МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Кваліфікаційна робота бакалавра

ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ ВЕДЕННЯ МОНІТОРИНГУ НАВЧАЛЬНИХ ДОСЯГНЕНЬ УЧНІВ АБО СТУДЕНТІВ

Здобувач освіти гр. ІН – 81 Олексій КОЛІСНИК

Науковий керівник,
кандидат технічних наук, доцент В'ячеслав МОСКАЛЕНКО

Завідувач кафедри,
доктор технічних наук, професор Анатолій ДОВБИШ

СУМИ 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____
Зав. кафедри Довбиш А.С.
"_____" _____2022 р.

ЗАВДАННЯ
до кваліфікаційної роботи бакалавра

Студента 4-го курсу, групи ІН-81 спеціальності 122 -Комп'ютерні науки, денної форми навчання Колісника О.М.

Тема: Інформаційна система для ведення моніторингу навчальних досягнень учнів або студентів

Затверджена наказом по СумДУ
№_____ від _____2022 р.

Зміст пояснювальної записки: 1) огляд існуючих аналогів; 2) проектування бази даних; 3) вибір мови програмування та фреймворків; 4) проектування архітектури додатку; 5) проектування архітектури системи; 6) програмна реалізація; 7) аналіз результатів виконаної роботи.

Дата видачі завдання "_____"_____2022 р.
Керівник випускної роботи _____ В'ячеслав МОСКАЛЕНКО
Завдання прийняв до виконання _____ Олексій КОЛІСНИК

РЕФЕРАТ

До складу кваліфікаційної роботи входить вступ, три основні розділи, висновки по всій роботі, список використаної літератури до складу якої входить 9 джерел. Кваліфікаційна робота містить в собі 176 сторінок на яких розміщені 25 таблиць, 47 рисунків та 3 додатки.

Метою даної роботи є створення системи моніторингу навчальних досягнень учнів та студентів, для зручності ведення оцінювання навчального процесу. При чому ця система має бути представлена, як веб-додаток, що дає можливість зберігання результатів роботи додатку, їх перегляду та редагування з боку викладачів.

Щоб у повному обсязі досягнути поставленої мети була вирішена низка завдань, а саме:

- Провести порівняльний аналіз існуючих аналогів;
- Сформувати представлення про функціонал додатку;
- Вибрати мову програмування та фреймворки для побудови веб-додатку;
- Побудувати зручну базу даних;
- Спроектувати інтерфейс, що дозволяє зручно взаємодіяти з користувачем;
- Розробити серверну частину системи, що відповідає за її функціонал.

Результатом роботи є розроблена система у вигляді веб-додатку, що надає користувачам різних ролей можливість у зручній формі проводити моніторинг результатів навчання.

ЗМІСТ

ВСТУП

Оцінювання навчальних досягнень учнів та студентів під час проходження навчання завжди є важливою частиною навчального процесу. При чому результати цього оцінювання є одним з головних результатів навчання. Тому не дивно, що викладачам, які проводять оцінювання, є дуже важливо донести результати роботи їх учнів та студентів. У наш час коли всю інформацію прийнято розповсюджувати, поширювати та передавати через мережу, не є дивним, що більшість навчальних закладів має або намагається створити власні веб-додатки, котрі містять у собі можливість донесення важливої інформації до учнів чи студентів у тому числі і їх результатів навчання. Враховуючи, що кількість навчальних закладів постійно зростає, бо постійно з'являються нові курси, приватні школи та подібні заклади. При чому часто учні та студенти не мають фізичної змоги перебувати біля викладачів, то попит на веб-додатки, що надають прозорі можливості для виставлення та перегляду оцінок згідно навчальних досягнень студентів тільки зростає.

Мета роботи – створення системи моніторингу навчальних досягнень учнів та студентів, для зручності ведення оцінювання навчального процесу. Дана система повинна бути створена у вигляді веб-додатку з можливістю зберігання результатів оцінювання та можливістю перегляду цих результатів для студентів або учнів, а також для власне самого оцінювання з боку вчителів. У наш час, коли вся інформація переноситься в мережу, а кожний навчальний заклад прагне зробити оцінювання більш прозорим для всіх, кількість веб-додатків, котрі спеціалізуються на відображені навчального процесу, буде тільки зростати. Тож на сьогодні розробка подібних сервісів є важливою та актуальною темою.

Завдання роботи – створити систему, котра буде містити у собі інформацію про досягнення студентів та учнів. При цьому надасть викладачам зручний спосіб виставлення оцінок за виконані завдання.

1. ОГЛЯД ВІДОМИХ АНАЛОГІВ

Одним з головних підготовчих етапів для створення додатку є знаходження можливих аналогів та проведення аналізу їх можливостей та функціоналу. Це варто робити для того, щоб отримати представлення про вже існуючі рішення. А саме, які можливості вони надають користувачу, які недоліки та переваги мають. Також після проведення аналізу можна підкреслити на що саме треба буде звернути увагу при розробці власного рішення.

Також після проведення аналізу аналогів, по ньому можна визначити, які ідеї були реалізовані вдало, а які не варто реалізовувати, або реалізовувати тільки після проведення серйозних доопрацювань. Звісно для отримання більшої кількості інформації краще за все брати декілька існуючих рішень, з подальшим їх порівнянням. А також у ході порівняльного аналізу варто зрозуміти, які нові можливості, порівняно з вже існуючими рішеннями можна надати майбутньому вед-додатку, котрих немає у проаналізованих додатках.

1.1. Аналіз існуючих аналогів

Для проведення аналізу беремо декілька прикладів вже існуючих рішень. Тож проведемо аналіз над інформаційними системами для ведення моніторингу навчальних досягнень учнів та студентів таких як ThinkWave та Edmodo.

1.1.1. ThinkWave

ThinkWave – це повноцінно хмарне рішення, призначене на моніторингу навчальних досягнень. Що дозволяє отримати доступ через будь-який браузер без встановлення додаткового програмного забезпечення. Дане рішення надає повний набір функціоналу, що необхідний для ведення звичайного документування результатів навчання. Крім цього цей додаток має інтуїтивно-простий у використанні інтерфейс, навчитися користуватися ним є доволі

швидкою задачею. Надає зручний фунціонал для додавання студентів до різних предметів, та у рамках цих предметів призначення різних завдань. Дане рішення є чудовим, як для окремих вчителів, котрі мають бажання полегшити спосіб ведення оцінювання, так і для доволі крупних навчальних закладів. Головна сторінка представлена на рисунку 1.1.1:

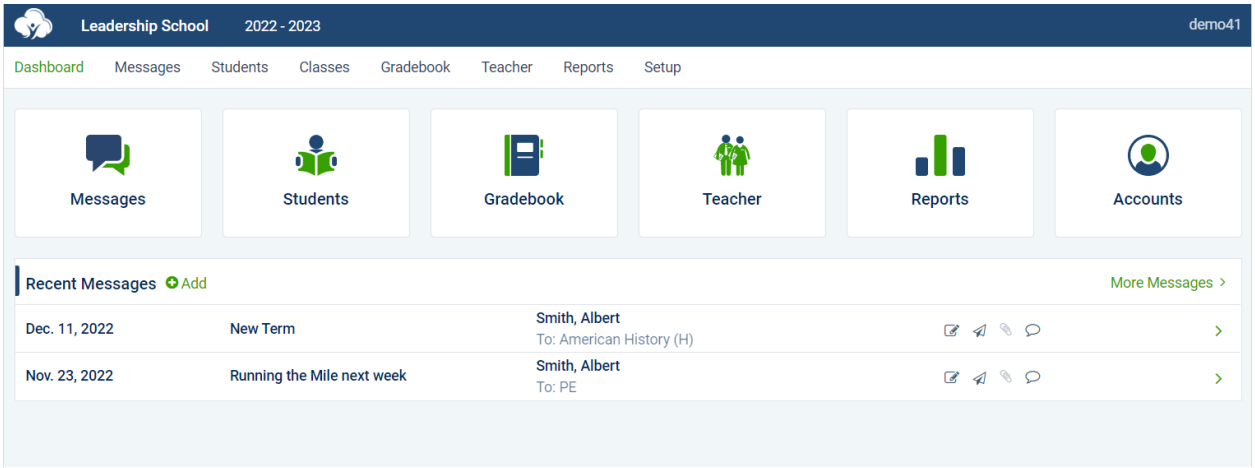

Рисунок 1.1.1 – Головна сторінка ThinkWave

Як можемо бачити, дане рішення пропонує шість головних розділів. З даних розділім розглянемо саме ті, що є головними для проведення оцінювання. Розділ під назвою “Classes” (див. Рис.1.1.2) містить у собі список предметів, з можливістю редагування їх даних та з можливістю створення нового предмету.

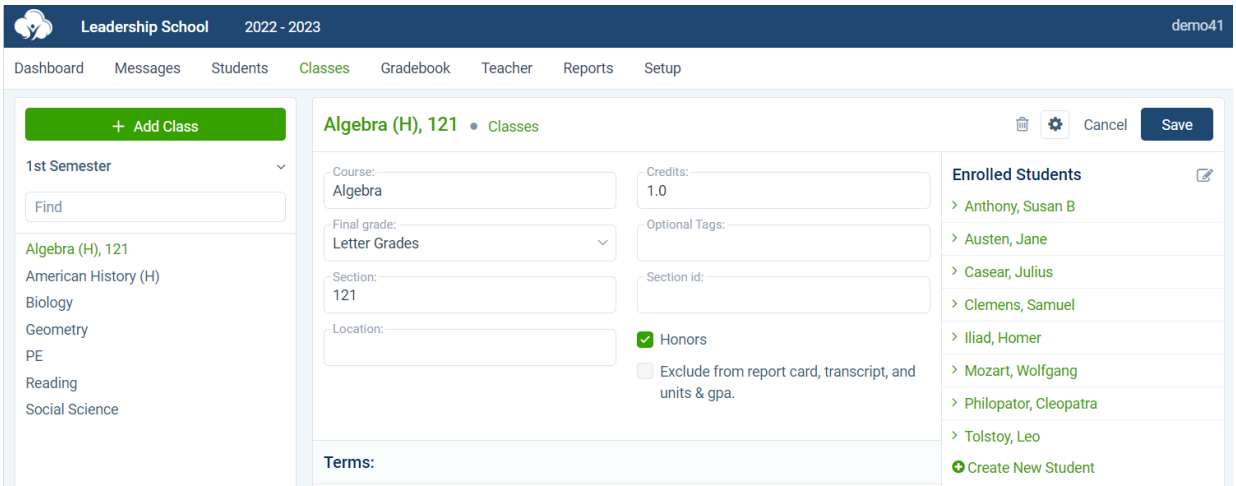

Рисунок 1.1.2 – Розділ “Classes” додатку ThinkWave

Розділ під назвою “Students” (див. Рис.1.1.3) містить у собі список студентів, з можливістю редагування їх даних та з можливістю створення нового студента. Аналогічні можливості є і у розділі “Teachers”. Також є можливість додання студента до певної групи, що покращує можливість групування студентів.

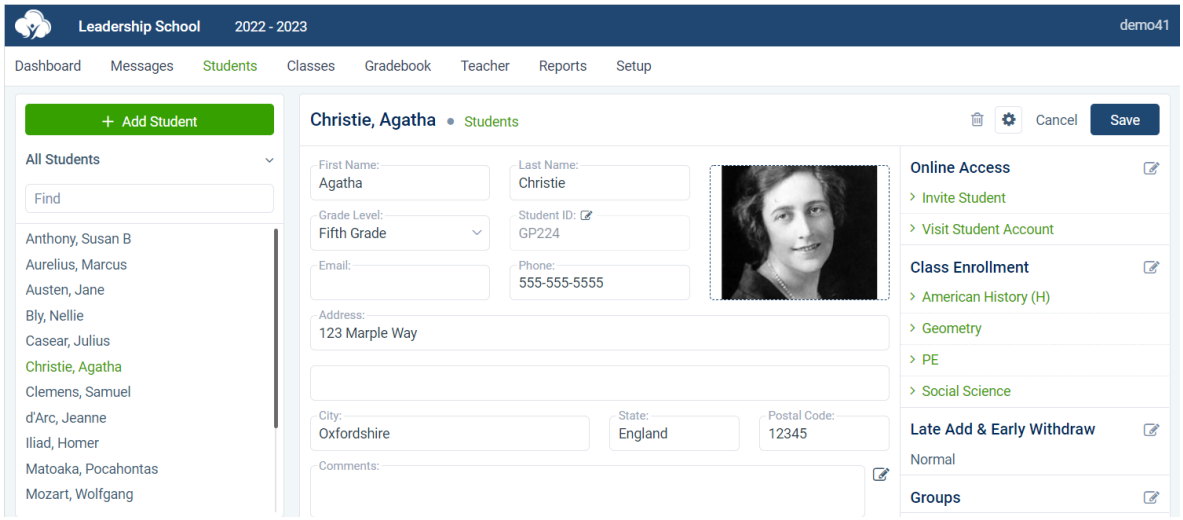

Рисунок 1.1.3 - Розділ “Students” додатку ThinkWave

Розділ під назвою “GradeBook” (див. Рис.1.1.4) містить у собі список предметів, для кожного предмету наявна таблиця, що містить у собі список студентів з оцінками. При чому є можливість додавання нових завдань та встановлення оцінок для кожного студента. Окрім цього наявна можливість проглянути оцінки по кожному студенту окремо.

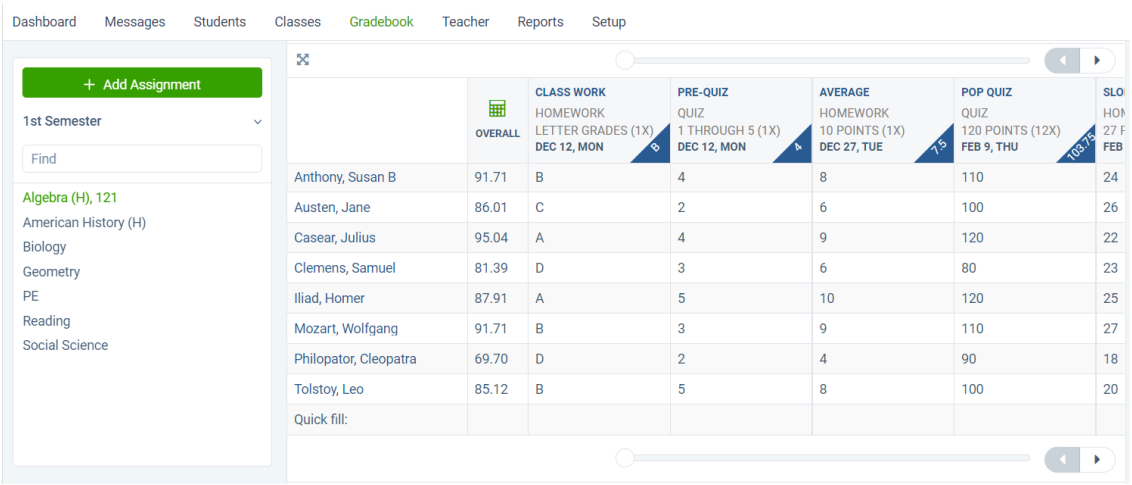

Рисунок 1.1.4 - Розділ “GradeBook” додатку ThinkWave

Крім цього на вкладках “Students”, “Teachers”, “Classes” та “GradeBook” надається можливість фільтрації відповідних об’єктів за їх назвами.

Як можемо бачити даний додаток дійсно надає користувачу доволі простий у використанні, але ефективний функціонал, що забезпечує можливість ведення оцінювання учнів або студентів по різним предметам. Основні розділи явно виокремлені та представлені у головному меню, що полегшує розуміння того, як влаштована система. До того ж існує система сповіщення студентів за допомогою повідомлень, що теж є доволі зручною функцією.

1.1.2. Edmodo

Edmodo – так само, як і попереднє рішення є повноцінно хмарним. Головне призначення якого полягає у моніторингу навчальних досягнень. Окрім функціоналу, що призначений безпосередньо для зберігання результатів навчання учнів та можливості їх перегляду, з можливістю їх редагування для вчителів, має ще й додаткові функції. Так наприклад даний додаток дозволяє створювати календарний план проведення занять, чи просто розкладу. Також надається можливість вести безпосередню розмову між викладачами та студентами за допомогою зручного месенджеру. Всі ці корисні функції оформлені у доволі простому та легкому у використанні інтерфейсі. Варто зазначити, що даний інтерфейс виконаний за стилістикою дуже схожою на сучасні соціальні мережі, це допомагає користувачам більш швидше опанувати базову роботу з цим вед-додатком. Головна сторінка представлена на рисунку 1.1.5:

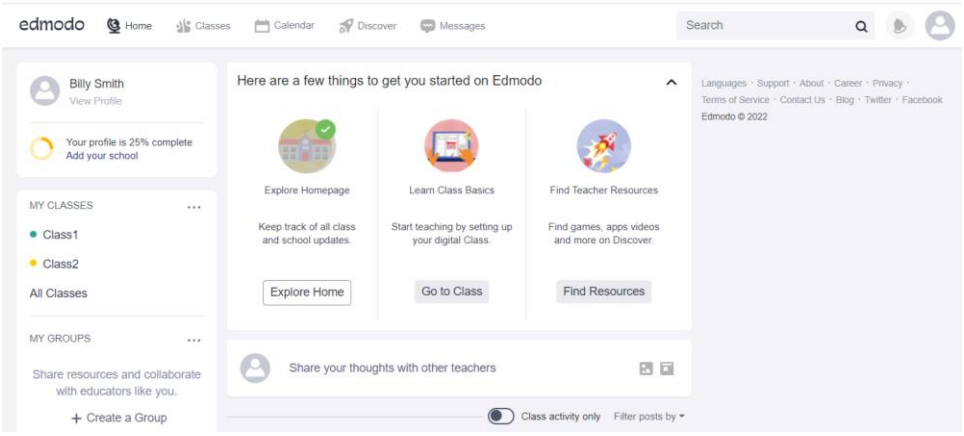

Рисунок 1.1.5 – Головна сторінка додатку Edmodo

Основний фунціонал, що відповідає за ведення журналу оцінювання знаходиться у розділі “Classes” (див. Рис.1.1.6). Цей розділ містить у собі список предметів, з можливістю редагування їх даних та з можливістю створення нового предмету.

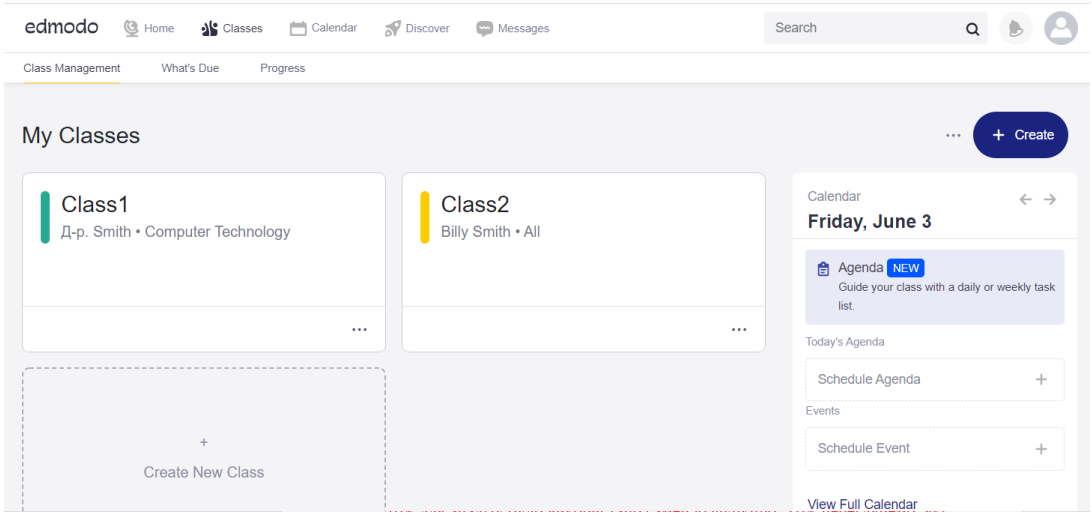

Рисунок 1.1.6 – Розділ “Classes” додатку Edmodo

До кожного предмету можна додавати студентів, це робиться на вкладці “Members” (див. Рис.1.1.7) відповідного предмету, якщо користувач ще не зареєстрований у системі, то можна попередньо створити для псевдо-аккаунти, які в подальшому учні можуть використати для реєстрації. Якщо ж студент вже є в системі, то додати його можна лише надіславши йому відповідний код предмету, і тільки учень може потім підтвердити своє

бажання долучитись до даного предмету. Аналогічно можна приєднувати і вчителів.

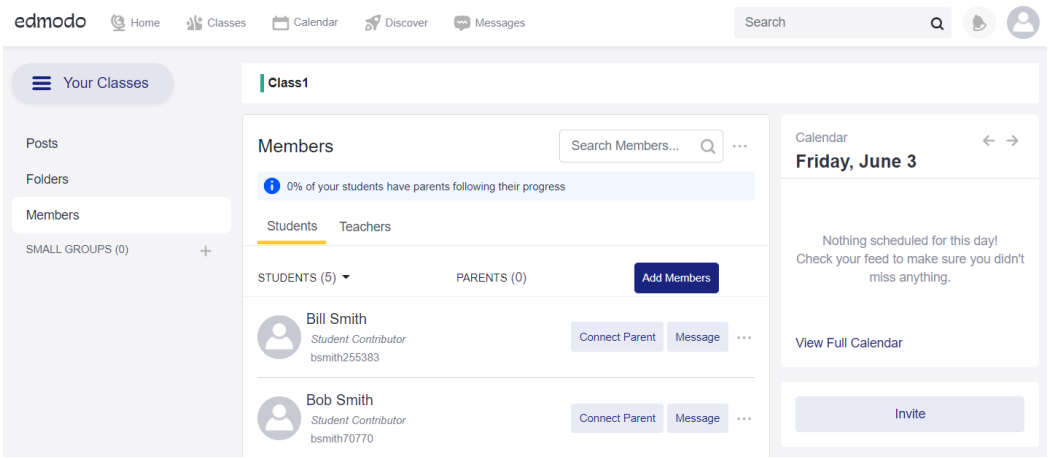

Рисунок 1.1.7 - Розділ “Members” додатку Edmodo

Підрозділ під назвою “Progress” (див. Рис.1.1.8), що входить до розділу “Classes” містить у собі список предметів, де для кожного предмету наявна таблиця, що містить у собі список студентів з оцінками. При чому є можливість додавання нових завдань та встановлення оцінок для кожного студента. Окрім цього наявна можливість проглянути оцінки по кожному студенту окремо.

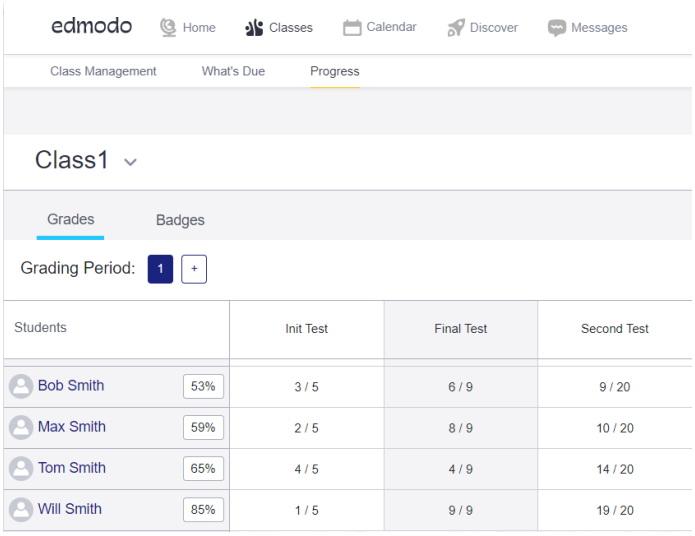

Рисунок 1.1.8 - Розділ “Progress” додатку Edmodo

Після аналізу даного рішення стає зрозуміло, що наданий користувачу функціонал у зручно представленому інтерфейсі дійсно практично повністю задовольняє потреби у веденні моніторингу навчальних досягнень. При чому існує цілий ряд додаткових інструментів, що покращують взаємозв'язок між викладачами та студентами, чудовим прикладом є вбудований месенджер. При тому існує проблема неможливості простого додавання вже зареєстрованого викладача чи студента до предмету. Що може породжувати незручності при створенні великої кількості предметів адміністратором.

1.2. Порівняння існуючих аналогів

Після проведення аналізу двох рішень, що включають в себе можливість моніторингу навчальних досягнень студентів, можемо перейти до їх порівняння. Це дасть можливість підкреслити сильні та слабкі сторони кожного з них та виокремити те, що буде корисно для власного рішення даного завдання.

Переваги ThinkWave – основними перевагами є зручний інтерфейс, наявність всього необхідного функціоналу для проведення оцінювання, можливість легко створювати нові об'єкти, можливість обмінюватися повідомленнями між викладачами та студентами засобами повідомлень. Не потребує встановлення додаткового програмного забезпечення на особистий пристрій.

Недоліки ThinkWave – не має локалізацій на інші мови, не надає достатнього функціоналу для великих навчальних закладів, не має можливості одночасно додавали більше одного студента засобами інтерфейсу.

Переваги Edmodo – має зручний інтерфейс та достатній функціонал, для базового опанування не потребує багато часу, має можливість планування часу занять, має вбудований месенджер. Не потребує встановлення додаткового програмного забезпечення на особистий пристрій. Надає можливість інтерфейсу створювати багато облікових записів учнів одночасно.

Недоліки Edmodo – для більш професіонального опанування потребує доволі багато часу, не має локалізацій на інші мови, при наявності існуючого облікового запису має незручний спосіб додавання студента до предмету. Студентам самим доводиться підтверджувати їх намір приєднатися до предмету.

Порівнявши переваги та недоліки можемо зробити висновки, що обидва мають купу вдалих моментів, на які варто спиратись при розробці власного додатку. Окрім недоліка рішення Edmodo, котрий полягає у ускладненому додаванні вже існуючого студента до предмету, всі інші можна вважати незначними, адже вони не створюють особливих незручностей.

Отримавши результат порівняння двох рішень було вирішено, що розробка власної інформаційної системи для моніторингу навчальних досягнень учнів та студенів, буде спиратися на приклади вдалої реалізації функціоналу та інтерфейсу, основних можливостей, обох додатків. Та до нашого веб-додатку не будуть включені ті частини, які є некритичними для повноцінного функціонування додатку.

1.3. Постановка задачі

Згідно теми роботи головним завданням є побудова вед-додатку, який є інформаційною системою призначення якої полягає у моніторингу навчальних досягнень учнів чи студентів. Це передбачає наявність функціоналу, який повністю вдовольнить потреби користувача, які необхідні для можливості ведення оцінювання, зберігання його результатів та подальшого їх перегляду та при необхідності зміни. При цьому він повинен бути простим у опануванні та використанні, тобто не потребувати багато часу для освоєння його функціоналу та мати зручний функціонал для виконання в ньому дій представлений у виді інтуїтивно зрозумілого інтерфейсу.

Отже для повного виконання завдання необхідно виконати усі наступні задачі:

1) Спроектувати зручну для використання та для зберігання базу даних.

2) Вибрати мову програмування та фреймворк для розробки серверної частини додатка.
3) Вибрати фреймворк та стилі для розробки інтерфейсу веб-додатка.
4) Розробити простий та інтуїтивно зрозумілий інтерфейс додатка.
5) Спроектувати архітектуру веб-додатку.
6) Реалізувати систему ідентифікації користувача.
7) Передбачити зручну систему пошуку необхідних елементів.
8) Передбачити чітке відображення в інтерфейсі таких елементів, як студент, викладач, предмет, група та завдання.
9) Надати викладачу можливість створення облікових записів студента, груп, предметів, а також завдань в рамках кожного предмету.
10) Надати викладачу можливість долучати студентів до груп та предметів, які їм необхідно викладати, а також видавати учням завдання.
11) Надати викладачу можливість встановлення та редагування оцінок студентів.

2. ВИЗНАЧЕННЯ МЕТОДУ ВИРІШЕННЯ ЗАДАЧ

2.1. Проектування бази даних

Враховуючи той факт, що для виконання поставленої задачі потрібно зберігати структурований набір даних, який має тісні взаємозалежності між своїми елементами, то є доцільно використовувати реляційну базу даних.

Серед існуючого переліку реляційних баз даних було вибрано Oracle Database. Ця база даних надає доволі широкі можливості. Одними з них є низька ціна операцій, повний перелік можливих типів полів, чудова масштабованість та широкий спектр інструментів для відновлення даних у випадку їх втрати.[1] Тож Oracle Database надає достатні можливості, котрі необхідні для повноцінного функціонування нашого веб-додатку.

Так як база даних була вибрана, то переходимо до самого проектування бази даних. Варто зазначити що на етапі проектування неможливо врахувати всі проблеми та нові рішення, які в подальшому можуть вплинути на структуру бази даних. Тому спроектована версія бази даних скоріш за все може отримати певні зміни. Для представлення бази даних доволі зручним є ERD діаграма. ERD (Entity relationship diagram) це зручний спосіб опису структури бази даних, де вказані таблиці, їх поля та стосунки між ними. Практично кожна гарна база даних має мати ERD діаграму.[2] Для побудови даних діаграм, гарно підходить CaseStudioPortable. Сама діаграма представлена на рисунку 2.1.1:

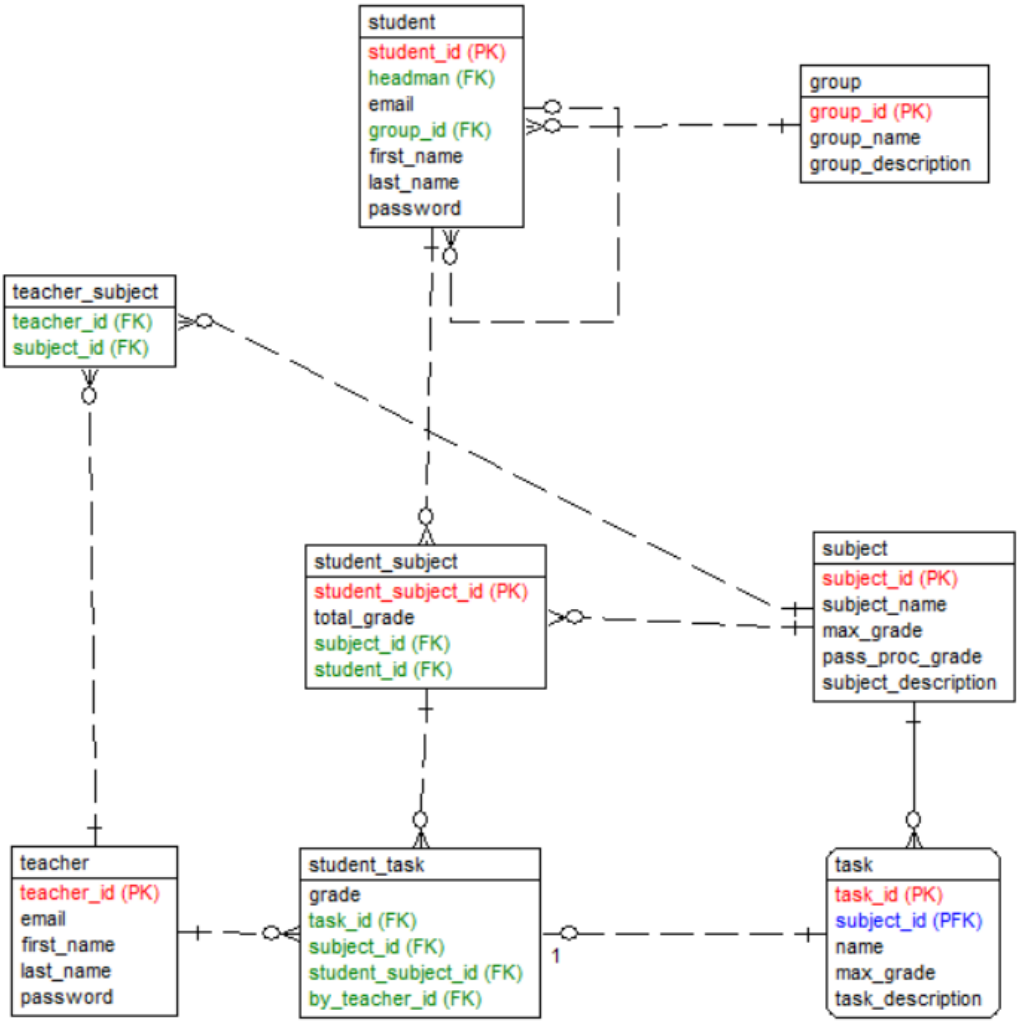

Рисунок 2.1.1 – ERD діаграма бази даних

Як можемо бачити, у спроектованій базі даних присутні, такі сутності, як студент, вчитель, предмет, група та задача. Це основні сутності, що необхідні для можливості ведення оцінювання навчальних досягнень викладачем по певному предмету, певної групи студентів.

2.2. Вибір мови програмування та основних фреймворків

Для побудови додатку, котрий передбачає ведення моніторинг навчальних досягнень учнів чи студентів, була вибрана мова програмування Java.

Java – це об’єктно-орієнтована мова програмування, котра була розроблена компанією Sun Microsystems, однією з головних її особливостей є те, що програми написані на цій мові транслюються у байт-код і потів виконуються на спеціальній віртуальній машині JVM.[3] Це забезпечує можливість не враховувати технічні особливості обладнання, це і зробило цю мову однією з найпопулярніших мов програмування у світі. Дана мова була вибрана через те, що вона є однією з найпопулярніших мов програмування у світі, це можна стверджувати оскільки вона постійно входить в топ 3 мов програмування кожного року. Ця мова має велику кількість можливостей, зручний фреймворків які на ній основані та через своє широке використання, купу корисної документації у мережі.

Спираючись на те, що мовою програмування була обрана Java та згідно завдання потрібно побудувати веб-додаток, то одним з найкращий фреймворків для цього завдання буде саме Spring Framework.

Spring Framework – це фреймворк з відкритим кодом, котрий надає інфраструктурну підтримку при розробці додатків на Java.[4] Це один з найпопулярніших фреймворків на мові Java, що використовується для побудови веб-додатків. Він надає розробнику обширний набір оптимізованого функціоналу, котрий зазвичай потрібно використовувати при створені різних веб-додатків, що значно полегшує процес розробки та позбавляє необхідності реалізовувати цей функціонал самому.[4] Через це у розробника є більше можливостей для роботи саме над бізнес логікою. Також, через те, що даний фреймворк є доволі популярним, він має велику кількість корисної документації та велику кількість обговорень, що значно полегшує пошук інформації.

Для відображення інформації будуть використовуватися Java Server Page (JSP) сторінки. Вони чудово підходять для роботи з Spring Framework. А також для побудови стилів на сторінках додатку буде використаний популярний фреймворк Bootstrap.

Bootstrap – це безкоштовний фреймворк, з відкритим кодом призначений для розробки дизайну сторінок, він побудований на HTML, CSS та JavaScript.[5] Головна його перевага полягає в тому, що він надає вже готові рішення для дизайну сторінок, які в основному лише треба адаптувати під власні потреби. Це суттєво зменшує час роботи розробника на дизайном сторінок.

2.3. Проектування архітектури додатку

Для проектування архітектури вед-додатку зручним інструментом є діаграма послідовності. Сама діаграма послідовностей призначення для того, щоб продемонструвати взаємодію об’єктів, котра є впорядкованою у часі.[6] Цей тип діаграм наглядно демонструє як об’єкти повинні взаємодіяти між собою для того, щоб функціонал був реалізований правильно. Щоб побудувати діаграми послідовностей, було використано сервіс draw.io, котрий надає зручний шаблон для даного типу діаграм.

У нашому вед-додатку буде два види користувачів, це викладач та студент, при чому викладач наділяється правами адміністратора, а студент має права лише на перегляд інформації та редагування своєї особистої інформації. Тож продемонструємо за допомогою діаграм послідовностей дії, які можуть виконувати ці два види користувачів.

Представлення у вигляді діаграм дій, котрі може виконувати викладач над іншими викладачами, студентами, групами та предметами представлена на рисунку 2.3.1 та на рисунку 2.3.2:

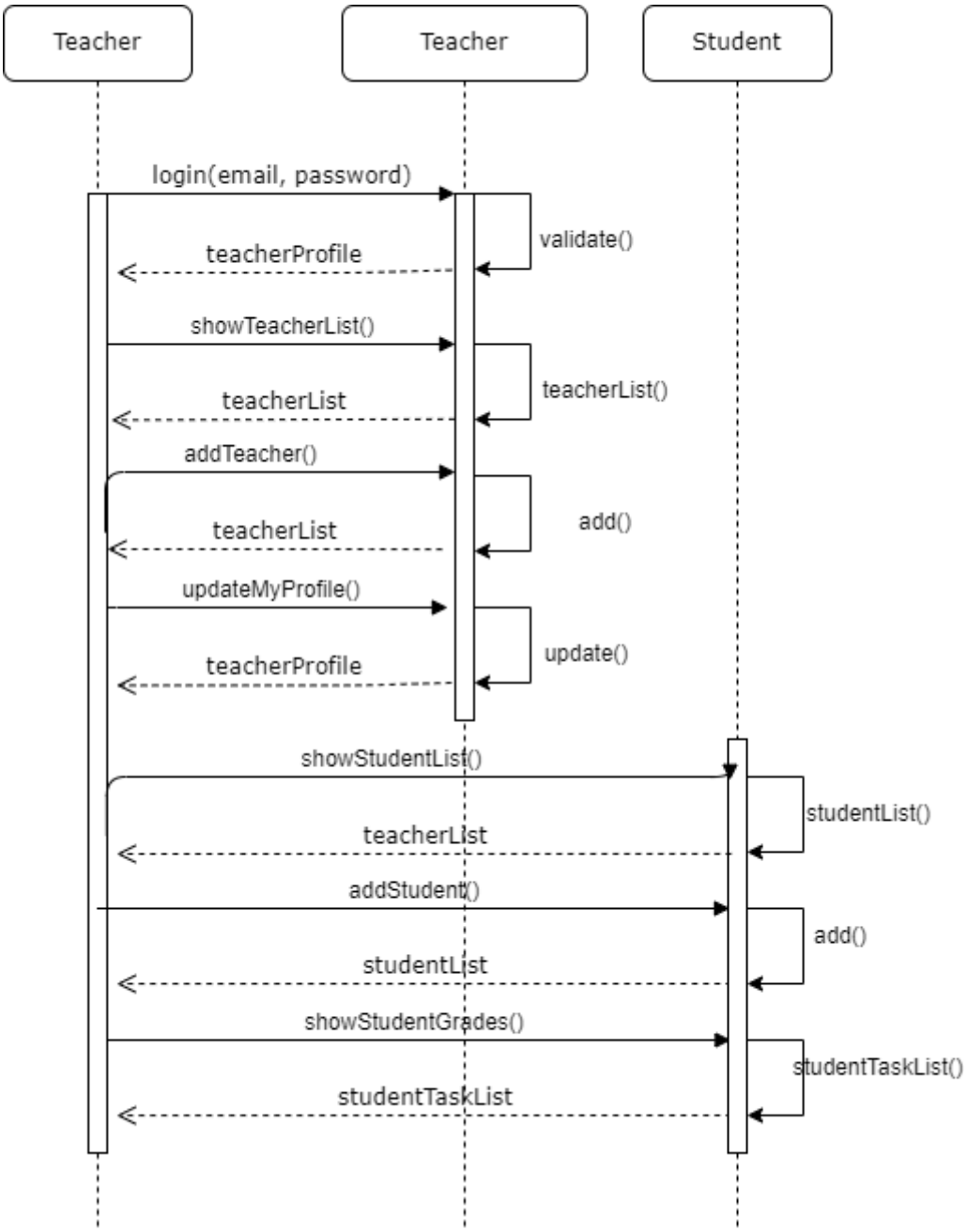

Рисунок 2.3.1 – Діаграма послідовностей для користувача викладач

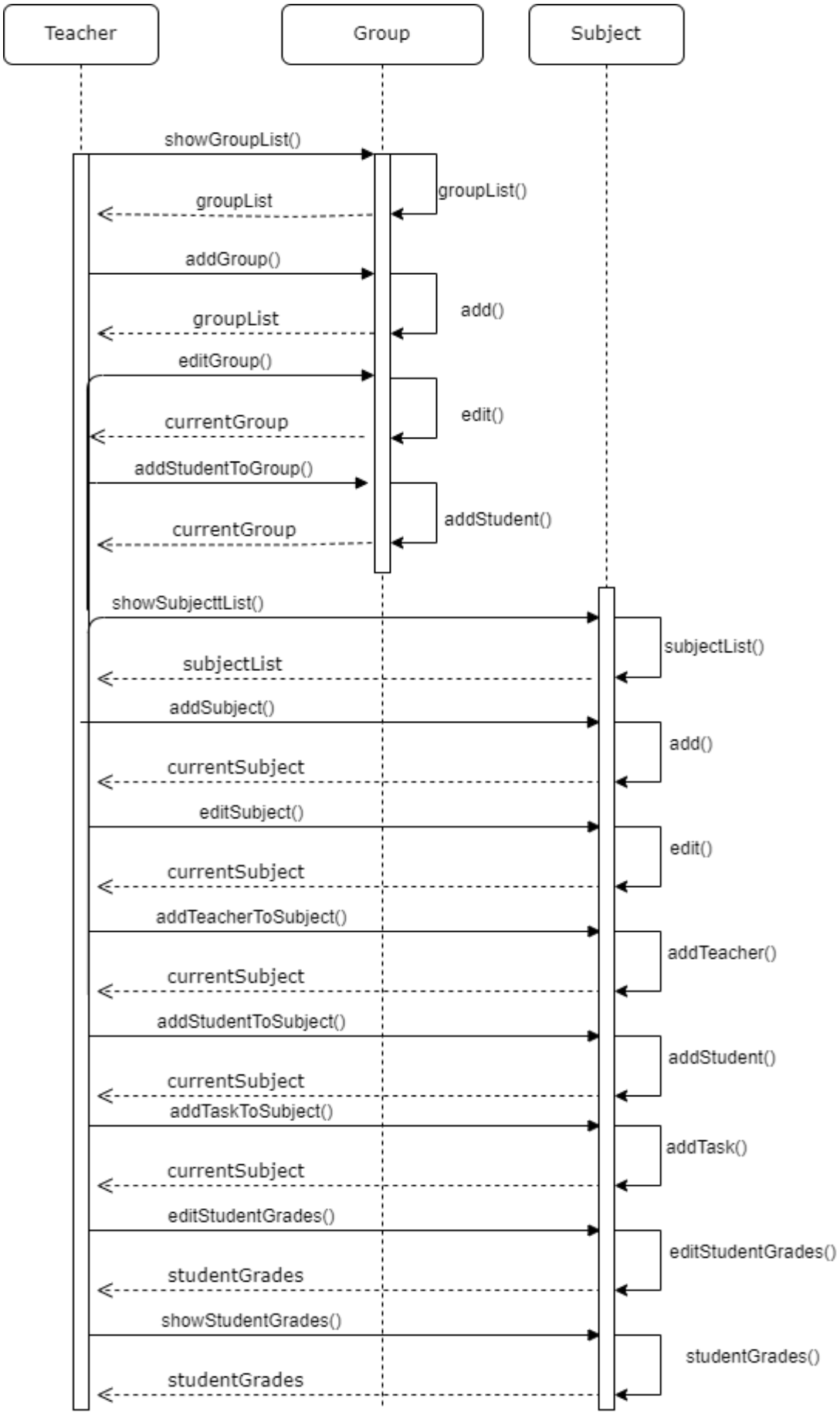

Рисунок 2.3.2 - Діаграма послідовностей для користувача викладач

Представлення у вигляді діаграм дій, котрі може виконувати студент над викладачами, іншими студентами, групами та предметами представлена на рисунку 2.3.3 та на рисунку 2.3.4:

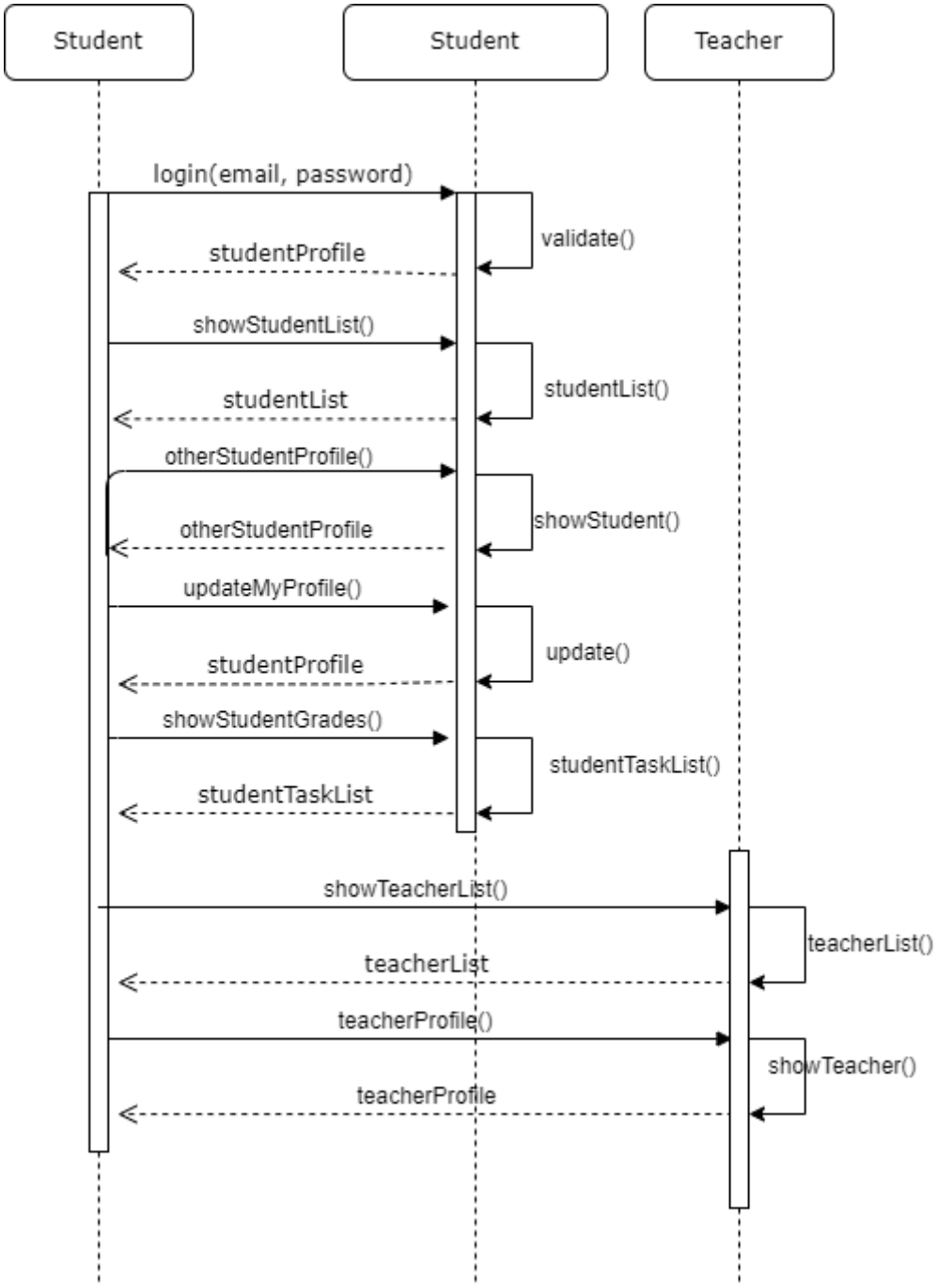

Рисунок 2.3.3 - Діаграма послідовностей для користувача студент

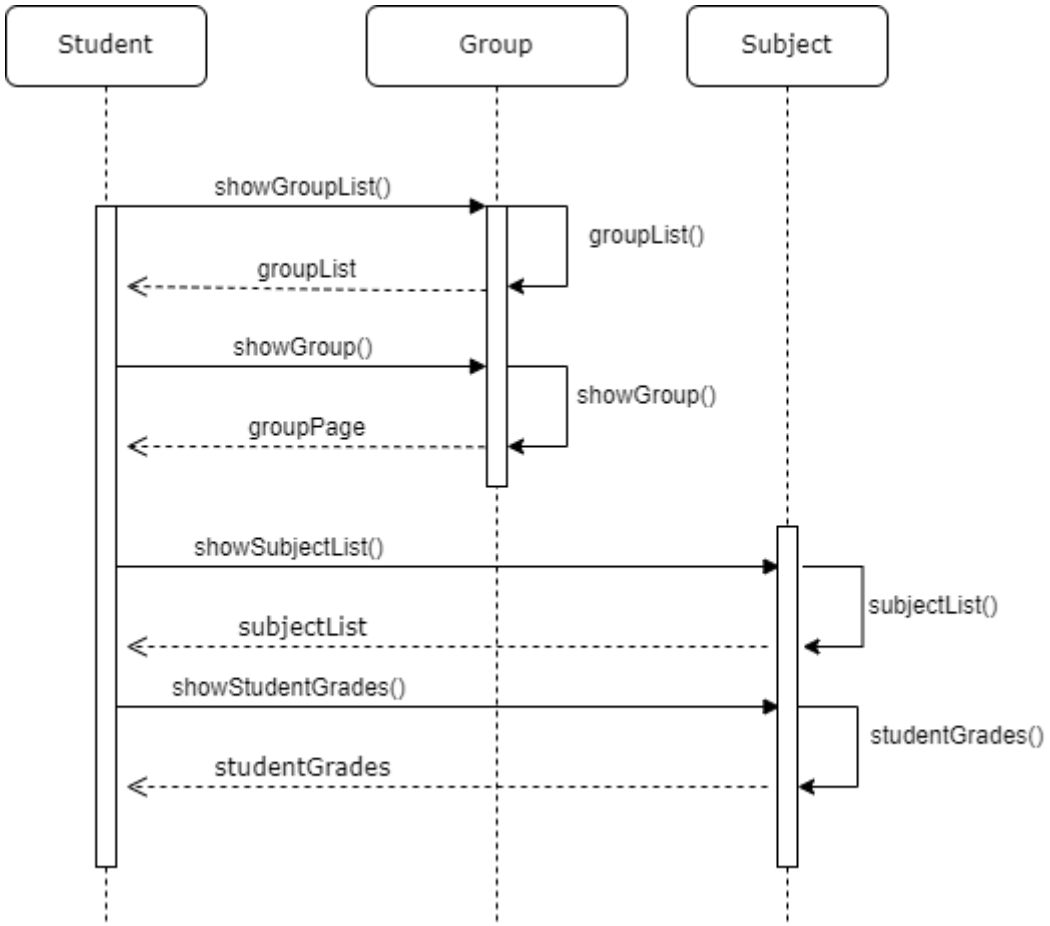

Рисунок 2.3.4 - Діаграма послідовностей для користувача студент

Як можемо бачити за допомогою діаграм послідовностей було наведено основний перелік дій, котрі можуть виконувати два типи користувачів. Але варто зазначити, що під час розробки додатку даний перелік може бути збільшеним.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1. Побудова бази даних

Для того, щоб додаток функціонував, одним із перших завдань є побудова бази даних, що буде зберігати всю необхідну інформацію. Оскільки ми вже маємо спроектовану діаграму на рисунку 2.1.1 у вигляді ERD діаграми то по ній і буде створена база даних. Основні сутності, такі як студент, викладач, предмет, група та задача є основними елементами системи тому вони мають свої виокремлені таблиці Student, Teacher, Subject, group та task. Їх опис представлений в таблицях 3.1.1 – 3.1.5.

Таблиця 3.1.1 – Поля таблиці Student.

Тип	Назва	Обмеження	Призначення
integer	student_id	not null , primary key	Ідентифікатор студента.
integer	headman	foreign key	Ідентифікатор куратора.
varchar2(30)	email	unique, not null	Електронна пошта студента.
varchar2(30)	first_name	not null	Ім’я студента.
varchar2(30)	last_name	not null	Прізвище студента.
integer	group_id	foreign key	Ідентифікатор групи студента.
varchar2(30)	password	not null	Пароль студента.

Таблиця 3.1.2 – Поля таблиці Teacher.

Тип	Назва	Обмеження	Призначення
integer	teacher_id	not null , primary key	Ідентифікатор викладача.
varchar2(30)	email	unique, not null	Електрона пошта викладача.
varchar2(30)	first_name	not null	Ім’я викладача.
varchar2(30)	last_name	not null	Прізвище викладача.
varchar2(30)	password	not null	Пароль викладача.

Таблиця 3.1.3 – Поля таблиці Subject.

Тип	Назва	Обмеження	Призначення
integer	subject_id	not null , primary key	Ідентифікатор предмета.
varchar2(30)	subject_name	not nul	Назва предмета.
integer	max_grade	not nul	Максимальна оцінка за предмет.
integer	pass_proc_grade	not nul	Прохідний бал у відсотках від максимальної оцінки.
varchar2(1000)	subject_description		Опис предмета.

Таблиця 3.1.4 – Поля таблиці Task.

Тип	Назва	Обмеження	Призначення
integer	task_id	not null, primary key	Ідентифікатор завдання.
integer	subject_id	not null, foreign key	Ідентифікатор предмета.
varchar2(30)	task_name	not null	Назва завдання.
integer	max_grade	not null	Максимальна оцінка за завдання.
varchar2(4000)	task_description		Опис завдання.

Таблиця 3.1.5 – Поля таблиці Group.

Тип	Назва	Обмеження	Призначення
integer	group_id	not null , primary key	Ідентифікатор групи.
varchar2(30)	group_name	not null	Назва групи.
varchar2(1000)	group_description		Опис групи.

Таблиці ж Student_Subject, Student_Task та Teacher_Subject є допоміжними та призначаються для створення зв’язків між основними сутностями. При цьому такі таблиці як Student_Subject та Student_Task окрім

зв’язків мають додаткову інформацію, що також використовується системо. Їх опис представлений в таблицях 3.1.6 – 3.1.8.

Таблиця 3.1.6 – Поля таблиці Student_Subject.

Тип	Назва	Обмеження	Призначення
integer	student_subject_id	not null, primary key	Ідентифікатор зв’язку студент-предмет.
integer	total_grade		Загальна оцінка по предмету у студента.
integer	student_id	not null, foreign key	Ідентифікатор студента.
integer	subject_id	not null, foreign key	Ідентифікатор предмета.

Таблиця 3.1.7 – Поля таблиці Student_Task.

Тип	Назва	Обмеження	Призначення
integer	student_subject_id	not null , foreign key	Ідентифікатор зв’язку студент-предмет.
integer	task_id	not null, foreign key	Ідентифікатор завдання.
integer	subject_id	not null, foreign key	Ідентифікатор предмета.
integer	grade		Оцінка отримана студентом.
integer	by_teacher_id	foreign key	Ідентифікатор викладача, що виставив оцінку.

Таблиця 3.1.8 – Поля таблиці Teacher_Subject.

Тип	Назва	Обмеження	Призначення
integer	subject_id	not null, foreign key	Ідентифікатор предмета.
integer	teacher_id	not null, foreign key	Ідентифікатор викладача.

Порівнявши спроектовану ERD діаграму та отриману базу даних можна зробити висновок, що вони повністю відповідають один одному.

3.2. Проектування інтерфейсу

У випадку коли йде розробка веб-додатку доволі велику роль має саме інтерфейс, адже для того, щоб можливо було реалізувати весь задуманий функціонал додатку, потрібно, щоб весь його інтерфейс давав можливість використання усього свого функціоналу. Саме через це слід заздалегідь визначитись з можливостями, що буде мати інтерфейс. Звісно у ході розробки деякі деталі можуть бути змінені та додані нові, бо передбачити все не можна, але приблизну структура можна визначити заздалегідь.

Оскільки наш веб-додаток заснований на тому, що ми маємо різних користувачів з різними правами то для того, щоб контролювати хто саме має доступ до конфіденційної інформації та, які дії може виконувати наявний користувач необхідно мати механізм авторизації. Вікно авторизації представлено на рисунку 3.2.1.

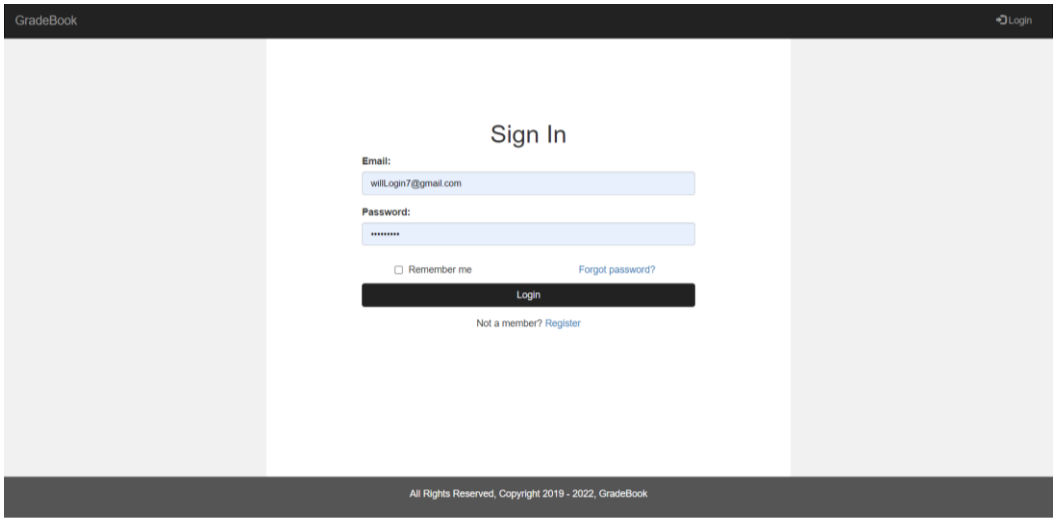

Рисунок 3.2.1 – Вікно авторизації

Після проведення вдалої авторизації користувач потрапляє на сторінку свого профілю. На власній сторінці користувач має поля, котрі містять його основну інформацію та назву ролі, форму для зміни власного паролю, та

кнопки, котрі дають можливість змінити особисту інформацію та видалити свій обліковий запис. Решта контенту сторінки залежить від ролі користувача.

Якщо користувач є викладачем, то на сторінці власного профілю (див. рис. 3.2.2) він додатково має список предметів у яких він викладає, посилання на кожний предмет, та можливість відмовитися від викладання предмету.

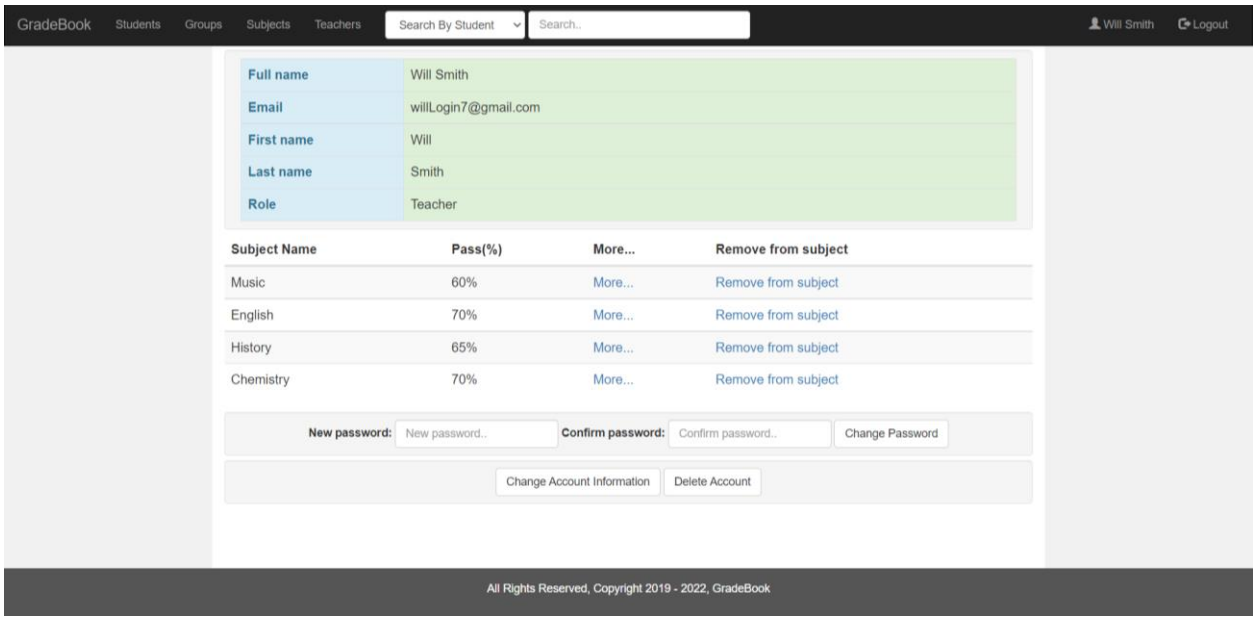

Рисунок 3.2.2 – Вікно власного профілю викладача

Якщо ж користувач є студентом, то на сторінці власного профілю (див. рис. 3.2.3), від додатково бачить власну групу та куратора з посиланнями на них, за умови їх наявності. Також на сторінці наведений список предметів, вивченням яких він займається. У списку представлена назва предмету, оцінка яку на даний момент має учень, власну оцінку у відсотках від максимальної, прохідний бал у відсотках, посилання на предмет, та посилання на детальний список оцінок, котрі отримав даний студент по даному предмету.

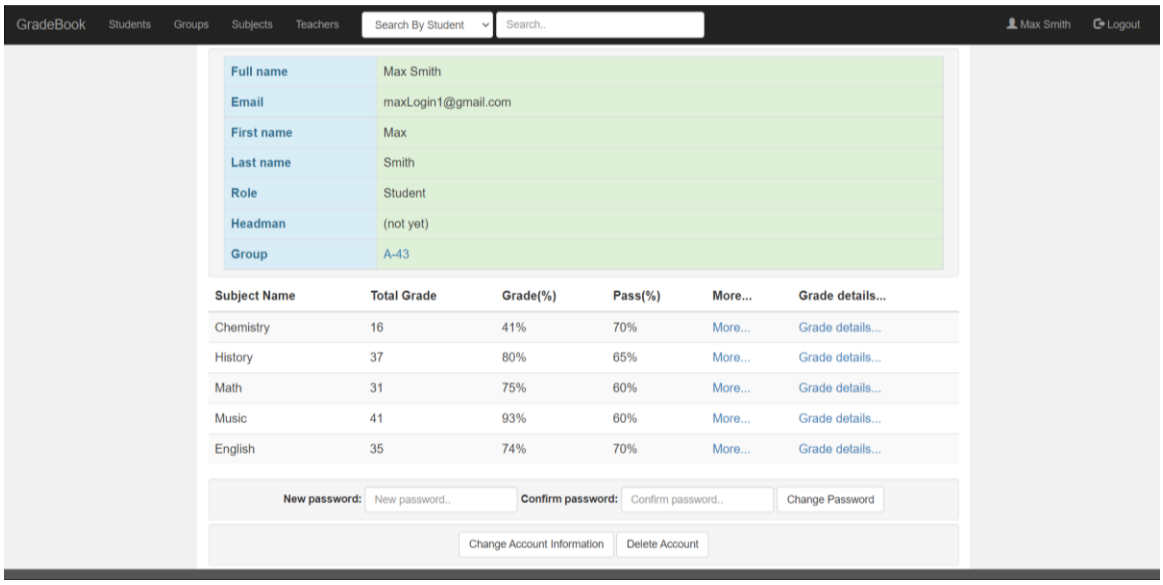

Рисунок 3.2.3 – Вікно власного профілю студента

Одним з основних способів взаємодії з додатком є його меню, у повному вигляді воно представлено на рисунку 3.2.4.

Рисунок 3.2.4 – Меню веб-додатку

Саме меню складається складається з трьох частин. Перша його частина (див. рис. 3.2.5) представлена чотирма пунктами меню, при натисканні на які користувач переходить на сторінку, що містить список об’єктів відповідного типу. Так при натисканні на кнопку “Students” відкривається список усіх студенів, “Groups” список усіх групп, “Subjects” список усіх предметів та “Teachers” список усіх викладачів. Детальний розбір контенту даних сторінок буде розглянутий у подальшому.

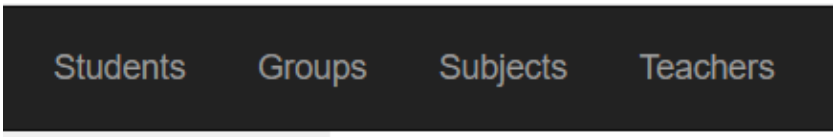

Рисунок 3.2.5 – Перша частина меню

Друга частина меню (див. рис. 3.2.6) представлена двома кнопками. Права кнопка, що містить ім’я та прізвище користувача при натисканні відриває профіль облікового запису авторизованого користувача. Ліва кнопка при натисканні припиняє дану робочу сесію та перенаправляє користувача до вікна авторизації.

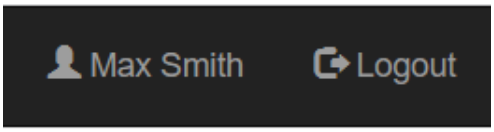

Рисунок 3.2.6 – Друга частина меню

Третьою частиною (див. рис. 3.2.7), є пошукова система, що дозволяє по часнині назви знаходити відповідного студента, предмет чи группу. Для зручності у правому випадаючому списку можна вибрати, об’єкти якого типу будуть шукатися. Відповідно після вибору типу пошуку у праве поле потрібно ввести частину назви группи та предмета або частину повного ім’я студента. Пошукова система не враховує регістр. Прикладі пошуку представлені на рисутках 3.2.8 – 3.2.10.

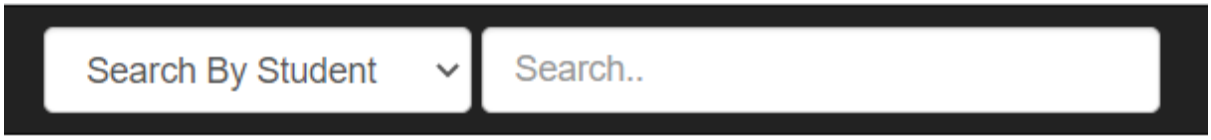

Рисунок 3.2.7 – Третя частина меню

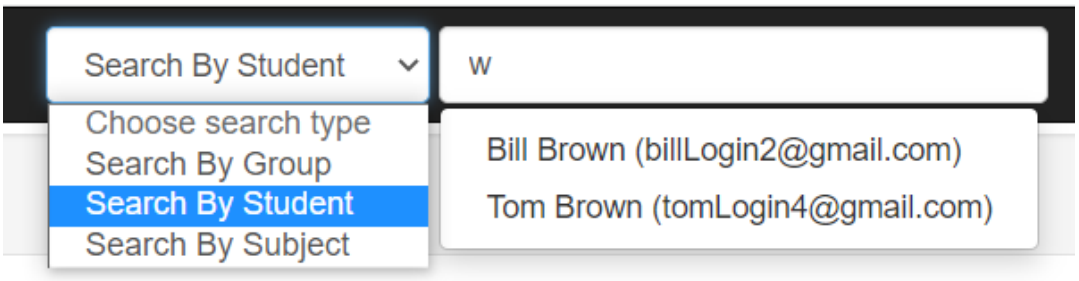

Рисунок 3.2.8 – Приклад пошуку студента

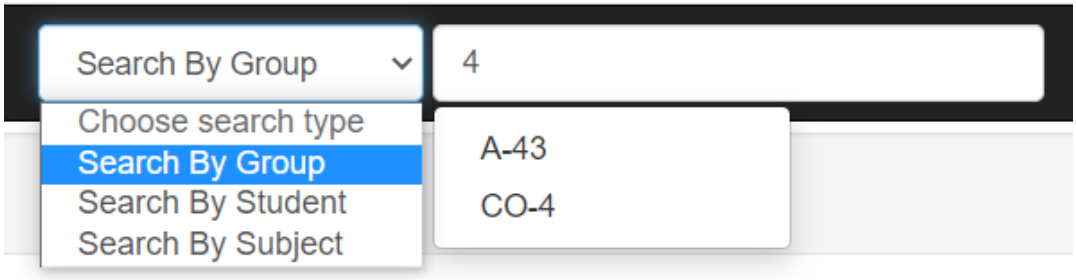

Рисунок 3.2.9 – Приклад пошуку групи

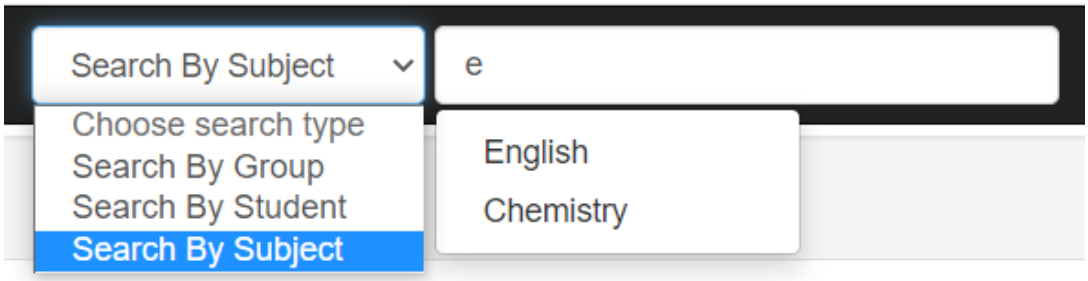

Рисунок 3.2.10 – Приклад пошуку викладача

Розділ “Students” містить у собі список усіх студентів (див. рис. 3.2.11), що наявні в системі. Даний список містить у собі основну інформацію по кожному студенту та відповідне посилання на їх профілі. Також якщо користувач є викладачем, він має можливість створити нового студента за допомогою відповідної кнопки “Add new student” що представлена на сторінці. Користувач із роллю студент такої можливості не має.

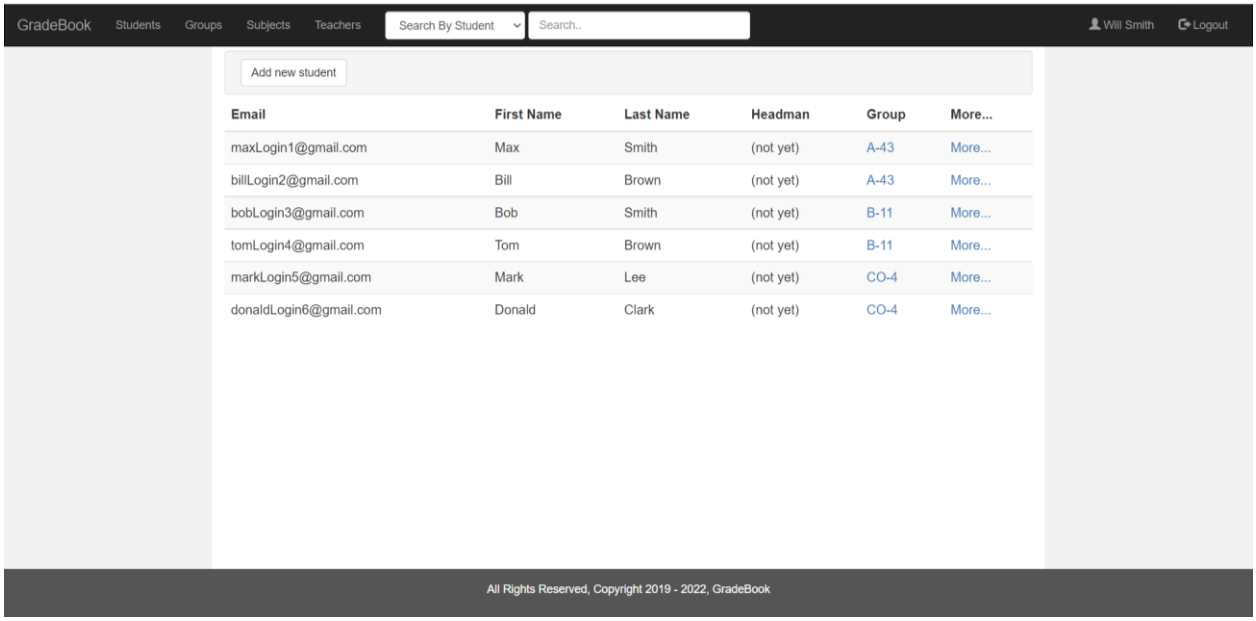

Рисунок 3.2.11 – Сторінка повного списку студентів

При спробі додання студента, з’являється вікно з відповідними полями (див. рис. 3.2.12), які необхідно заповнити, щоб створити нового студента. Поля “Headman” та “Group” представляють із себе випадаючі списки де представлені всі існуючі студенти та групи відповідно. Якщо при створені заздалегідь не відомо групу чи куратора ці поля можна залишити пустими та змінити з часом коли це буде потрібно. При умові заповнення всіх полів студента можна додати натисканням кнопки “Add” та відмінити дію натиснувши “Cancel”. Ця ж форма буде відкриватися при необхідності редагувати вже створеного студента, тільки при відкритті всі поля будуть заповнені інформацію даного студента, яку можна буде змінити за необхідністю та зберегти зміни.

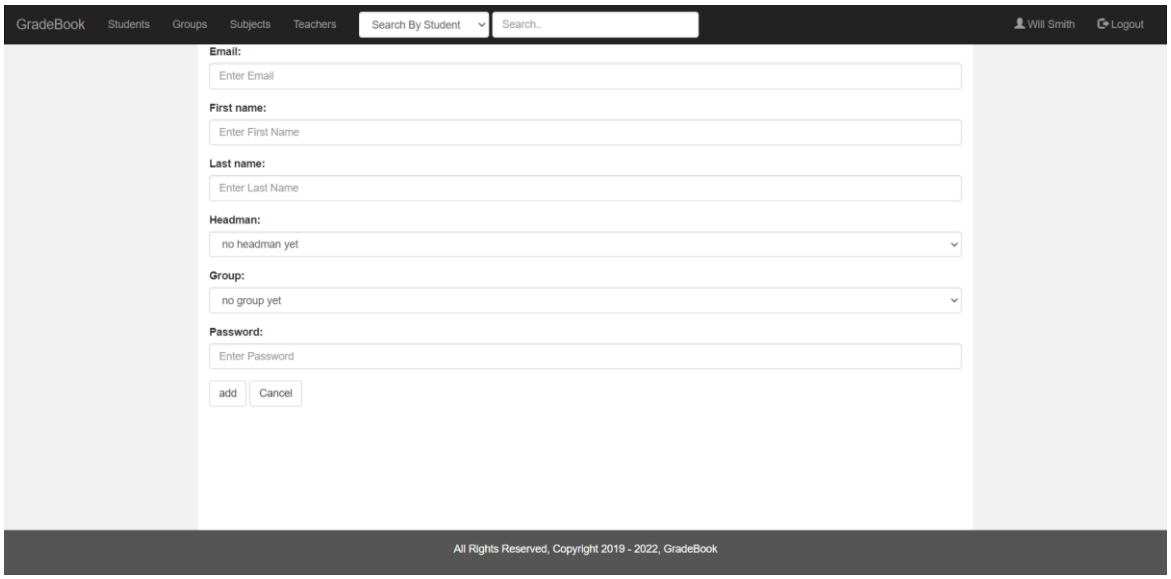

Рисунок 3.2.12 – Сторінка для створення-редагування студента.

Сторінка профілю іншого студента (див. рис. 3.2.13) аналогічна, сторінці профілю студента розглянутого раніше, але не має можливості зміни пароля, особистої інформації та видалення аккаунта. Якщо користувач є викладачем від має можливість відв’язати студента від предмета. Користувач із роллю студент таких можливостей не має.

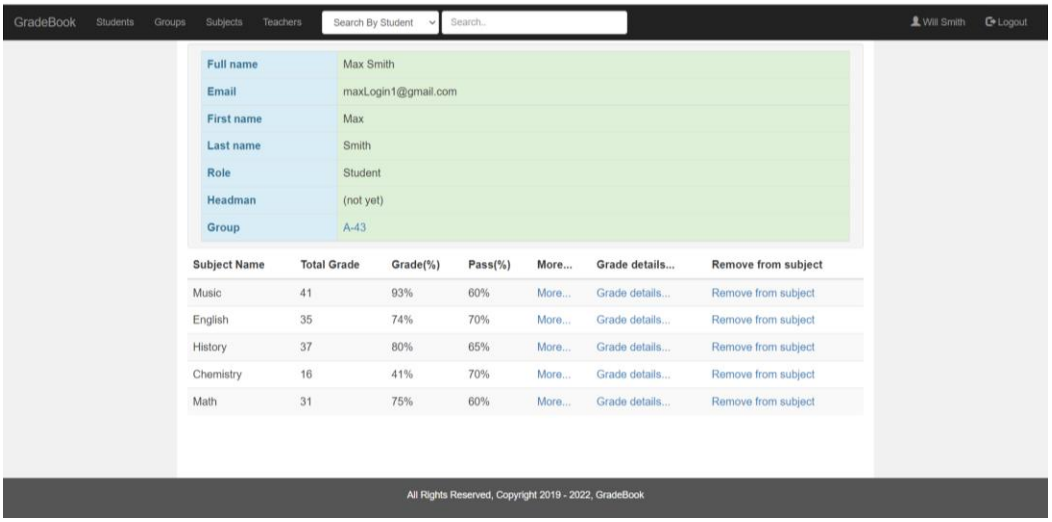

Рисунок 3.2.13 – Сторінка чужого профілю студента

Розділ “Teachers” містить у собі список усіх викладачів (див. рис. 3.2.14), що наявні в системі. Даний список містить у собі основну інформацію по

кожному викладачу та відповідне посилання на їх профілі. Також якщо користувач є викладачем, він має можливість створити нового викладача за допомогою відповідної кнопки “Add new teacher” що представлена на сторінці. Користувач із роллю студент такої можливості не має.

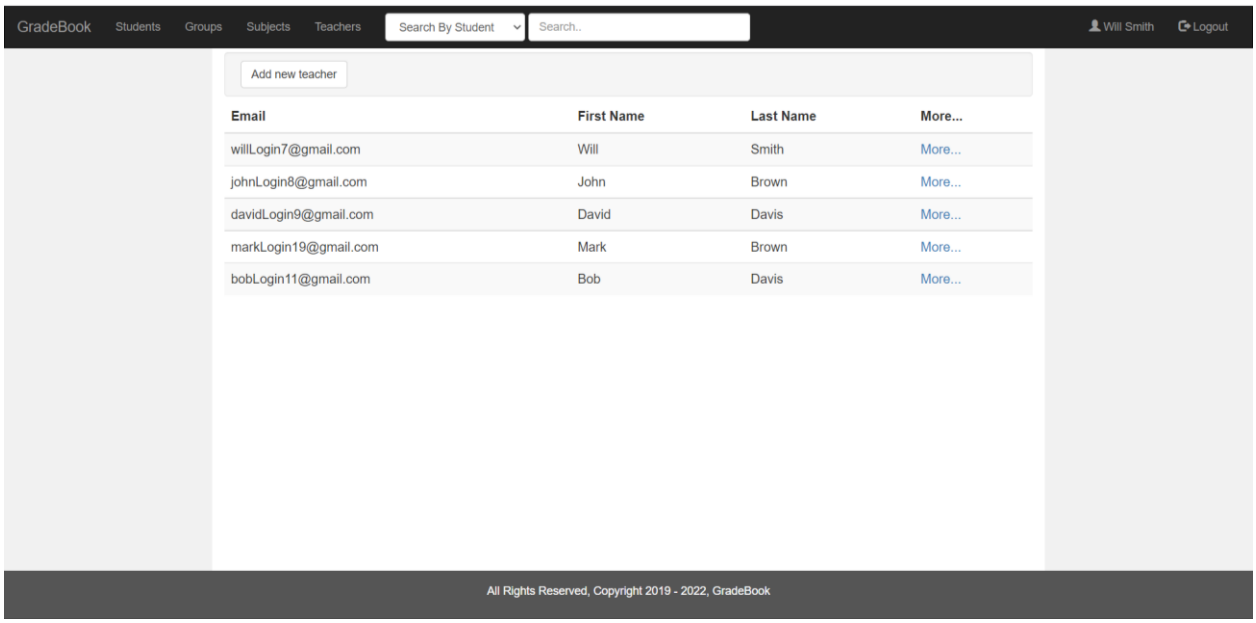

Рисунок 3.2.14 – Сторінка повного списку викладачів.

При спробі додання викладача, з’являється вікно з відповідними полями (див. рис. 3.2.15), які необхідно заповнити, щоб створити нову групу. Призначення кнопок аналогічне формі для створення студента.

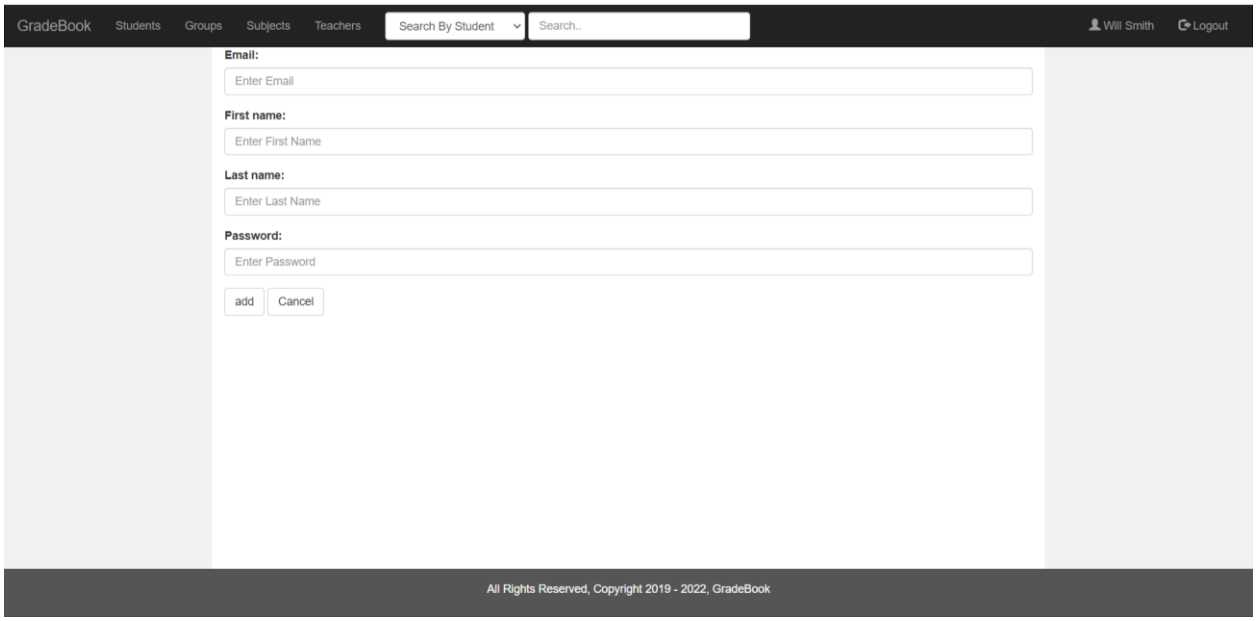

Рисунок 3.2.15 – Сторінка для створення-редагування викладача.

Сторінка профілю іншого викладача (див. рис. 3.2.16) аналогічна, сторінці профілю викладача розглянутого раніше, але не має можливості зміни пароля, особистої інформації та видалення аккаунта. Якщо користувач є викладачем від має можливість відв’язати викладача від предмета. Користувач із роллю студент таких можливостей не має.

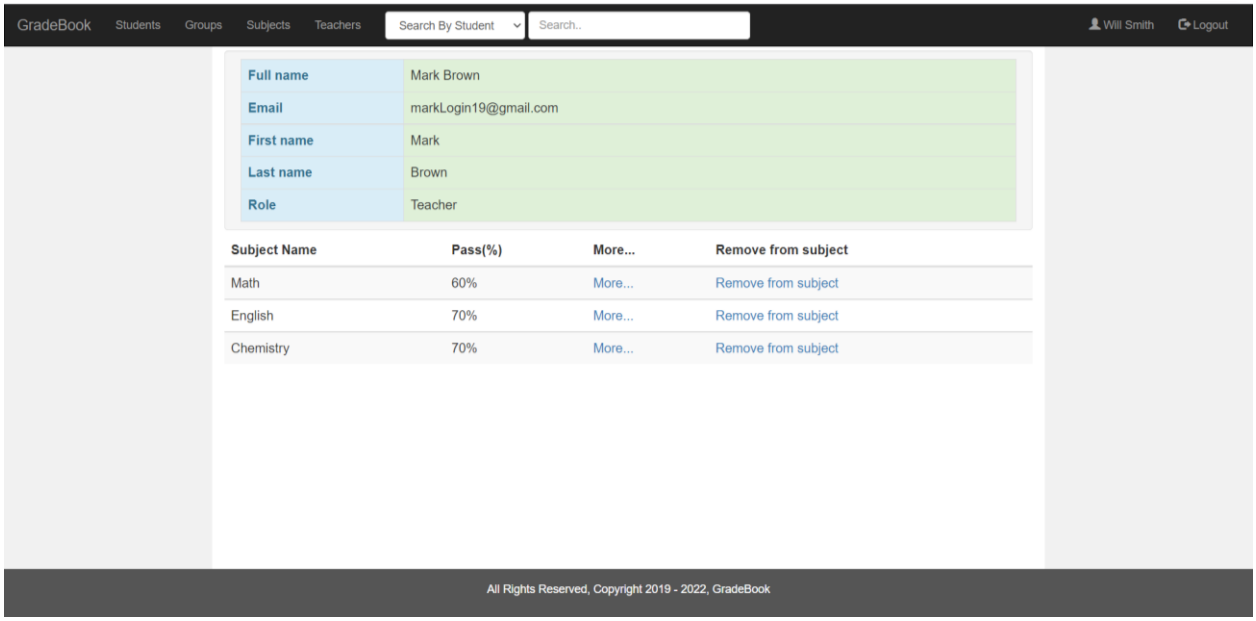

Рисунок 3.2.16 – Сторінка чужого профілю викладача.

Розділ “Groups” містить у собі список усіх груп (див. рис. 3.2.17), що наявні в системі. Даний список містить у собі назву групи, опис та відповідне посилання на її сторінку. Також якщо користувач є викладачем, він має можливість створити нову групу за допомогою відповідної кнопки “Add new group” що представлена на сторінці. Користувач із роллю студент такої можливості не має.

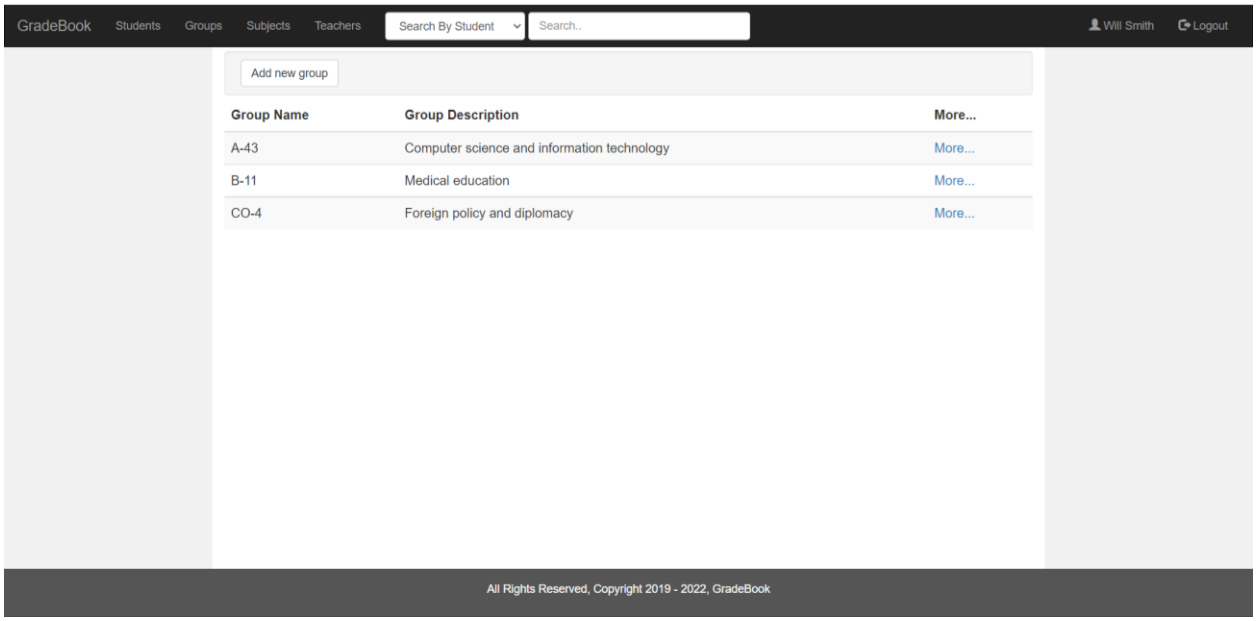

Рисунок 3.2.17 – Сторінка повного списку груп.

При спробі створення нової групи, з’являється вікно з відповідними полями (див. рис. 3.2.18), які необхідно заповнити, щоб створити нову групу. Призначення кнопок аналогічне формі для створення студента.

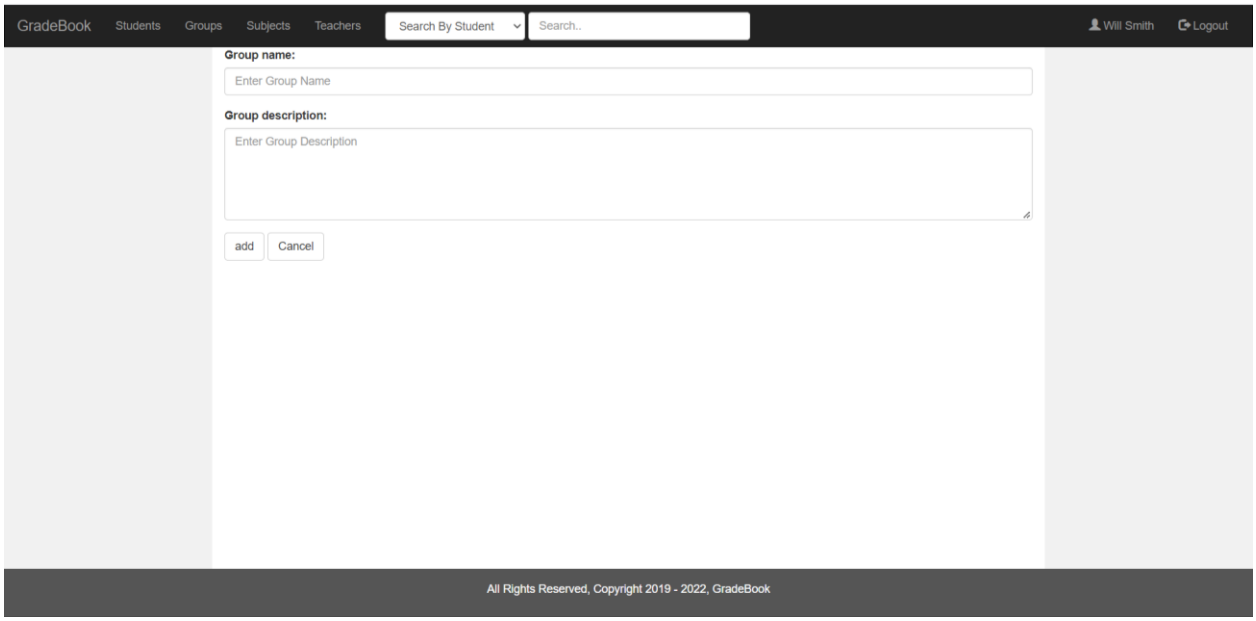

Рисунок 3.2.18 – Сторінка для створення-редагування групи.

Вікно групи представлене на рисунку 3.2.19. Воно містить у собі інформаційні поля з назвою групи та її описом. Також у кожній групі є список студентів, які до неї прив’язані. Якщо користувач є викладачем то він має можливість додати нового студента до групи через випадаючий список доступних студентів та кнопки “Add Student” або навпаки відв’язати деякого студента від даної групи за допомогою кнопки “Remove from group”. Також викладач може редагувати поля групи за допомогою кнопки “Change Group Info” та при необхідності видалити групу, при цьому всі студенти стануть відв’язані від неї. Редагування відбувається за тим же механізмом, що і для об’єктів типу студент. Користувач із роллю студент таких можливостей не має, та бачить лише інформаційні поля та список студентів групи.

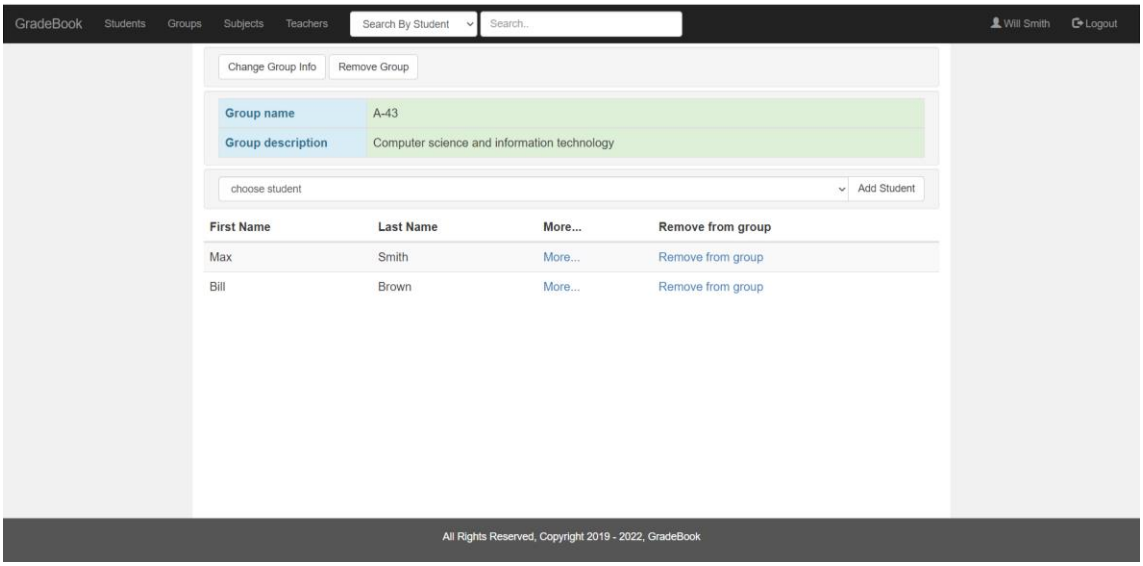

Рисунок 3.2.19 – Сторінка групи.

Розділ “Subjects” містить у собі список усіх предметів (див. рис. 3.2.20), що наявні в системі. Даний список містить у собі назву предмету, максимальну оцінку, прохідний бал у відсотках, опис та відповідне посилання на його сторінку. Також якщо користувач є викладачем, він має можливість створити новий предмет за допомогою відповідної кнопки “Add new subject” що представлена на сторінці. Користувач із роллю студент такої можливості не має.

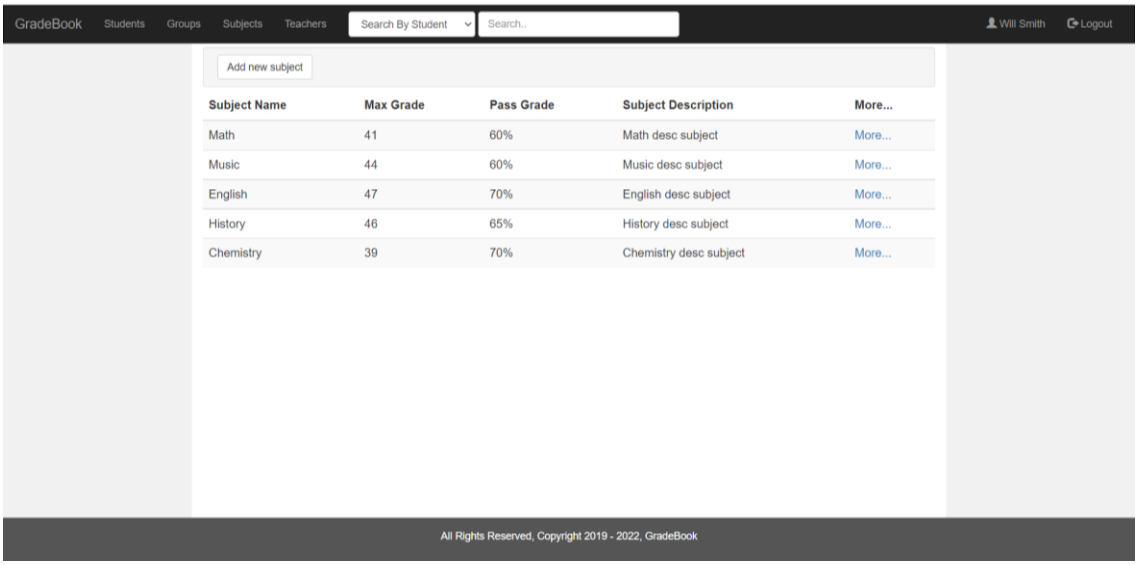

Рисунок 3.2.20 – Сторінка повного списку предметів.

При спробі створення нового предмету, з’являється вікно з відповідними полями (див. рис. 3.2.21), які необхідно заповнити, щоб створити нову групу. Призначення кнопок аналогічне формі для створення студента.

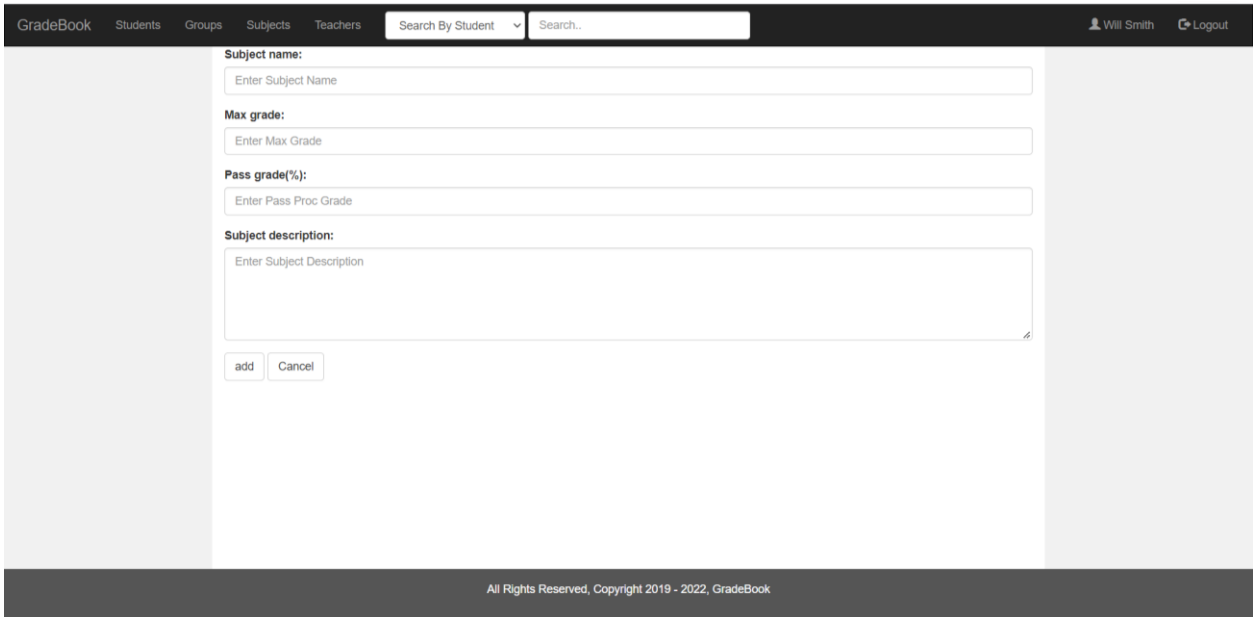

Рисунок 3.2.21 – Сторінка для створення-редагування предмету.

Вікно предмету представлене на рисунках 3.2.22 - 3.2.23. Воно містить у собі інформаційні поля з назвою предмета, максимальної оцінкою та його описом. Кожний предмет має три списки.

У першому представлені викладачі, котрі ведуть оцінювання по ньому. У цьому списку представлене ім’я та прізвище викладача та посилання на його профіль.

У другому представлені студенти, що проходять навчання по цьому предмету. У цьому списку представлене ім’я та прізвище студента, посилання на його профіль, оцінка за навчальні досягнення, ця ж оцінка у відсотках від максимального балу, та прохідний бал у відсотках. Також для кожного студента є посилання на сторінку де представлена детальна інформація про його навчальні досягнення по даному предмету.

У третьому представлені завдання, котрі автоматично представляються кожному студенту. У цьому списку представлена назва та опис завдання і максимальний бал за його виконання.

Якщо користувач є викладачем то він має можливість додати нового викладача чи студента до предмету. При додаванні студента усі завдання призначаються йому автоматично. Також викладач має право відв’язати іншого викладача чи студента від предмету. Ще у його розпорядженні є можливість редагувати поля предмету за допомогою кнопки “Change Subject Info” та при необхідності видалити предмет, при цьому всі студенти та викладачі стануть відв’язані від нього. Редагування відбувається за тим же механізмом, що і для об’єктів типу студент. Викладачу також доступна можливість створення нових завдань за допомогою відповідної кнопки “Add Task”, їх редагування за допомогою кнопки “Change” та навіть видалення по натисканню кнопки “Remove”.

Користувач із роллю студент таких можливостей не має, та бачить лише інформаційні поля та списки викладачів, студентів та завдань.

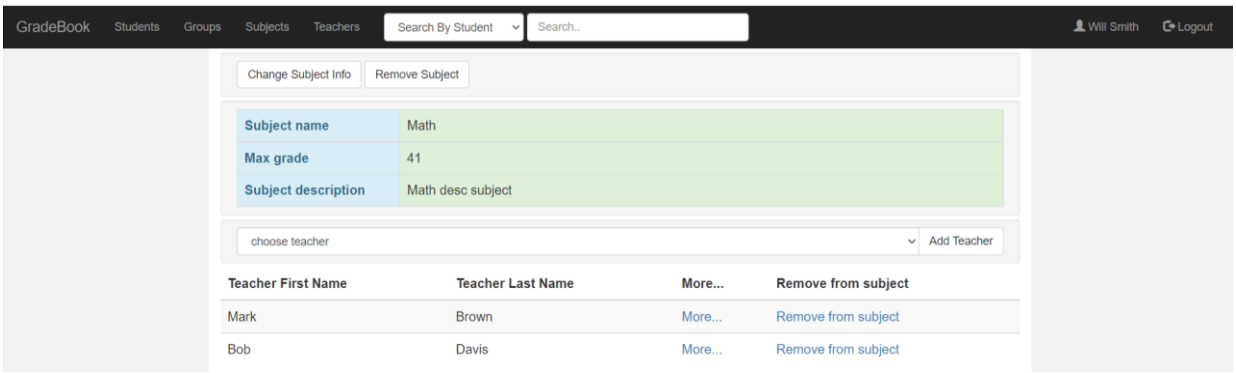

Рисунок 3.2.22 – Сторінка предмету, частина 1.

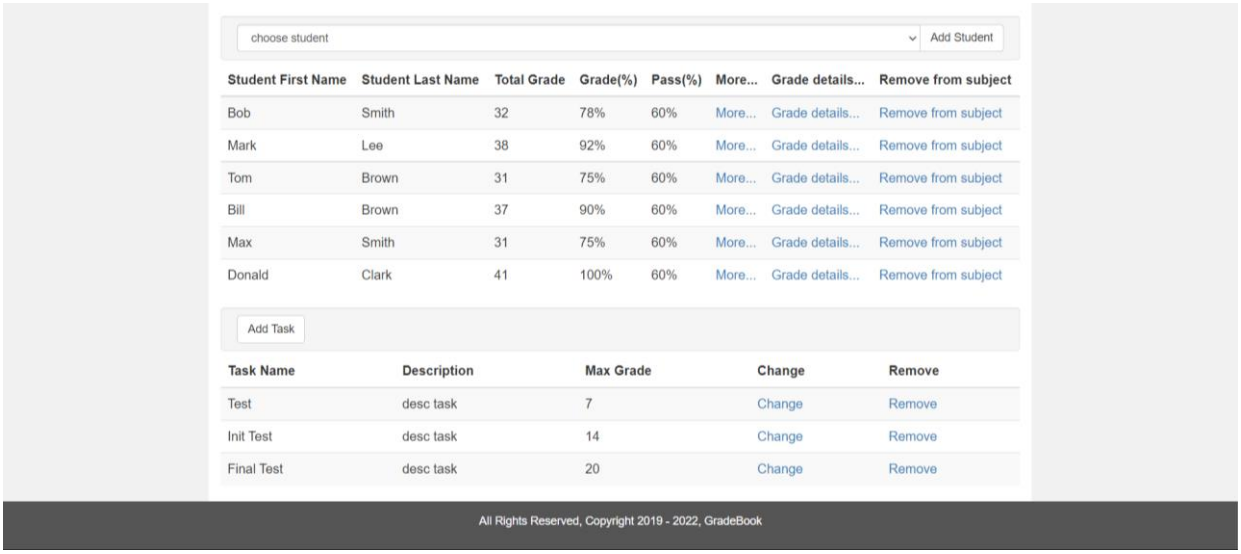

Рисунок 3.2.23 – Сторінка предмету, частина 2.

При спробі створення нового завдання, з’являється вікно з відповідними полями (див. рис. 3.2.24), які необхідно заповнити, щоб створити нове завдання. Призначення кнопок аналогічне формі для створення студента. Ця ж форма використовується для редагування цього ж типу.

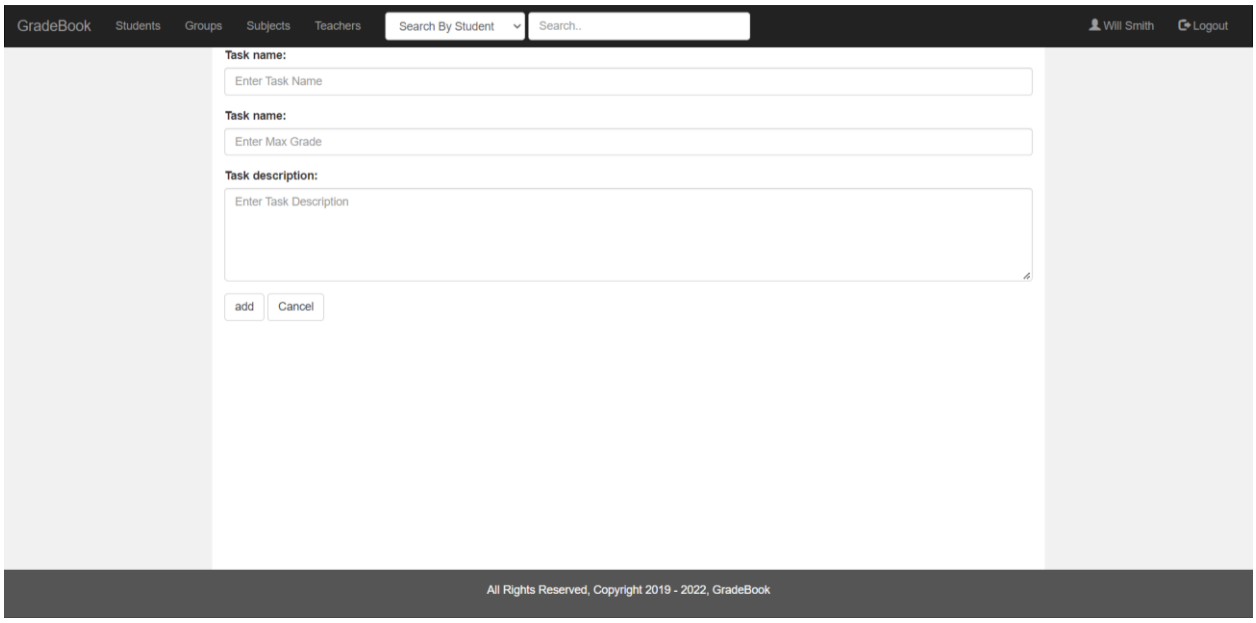

Рисунок 3.2.24 – Сторінка для створення-редагування завдання.

Дана сторінка (див. рис. 3.2.25) являє собою навчальну картку студента по певному предмету. На неї можна потрапити через профіль відповідного студента та через сторінку відповідного предмета.

Ця сторінка містить у собі посилання на предмет та студента, що проходить за ним навчання.

Також наявний список персональних завдань. У цьому списку представлена назва завдання, його короткий опис, максимальну оцінку, оцінку самого студента за дане завдання та ім’я та прізвище викладача, що оцінив завдання.

Якщо користувач є викладачем то він має можливість оцінювати кожне завдання змінюючи значення у полі “Grade” та натискаючи кнопку “Save”.

Користувач із роллю студент таких можливостей не має, та бачить лише інформаційні поля з посиланнями та список особистих завдань без можливості редагування.

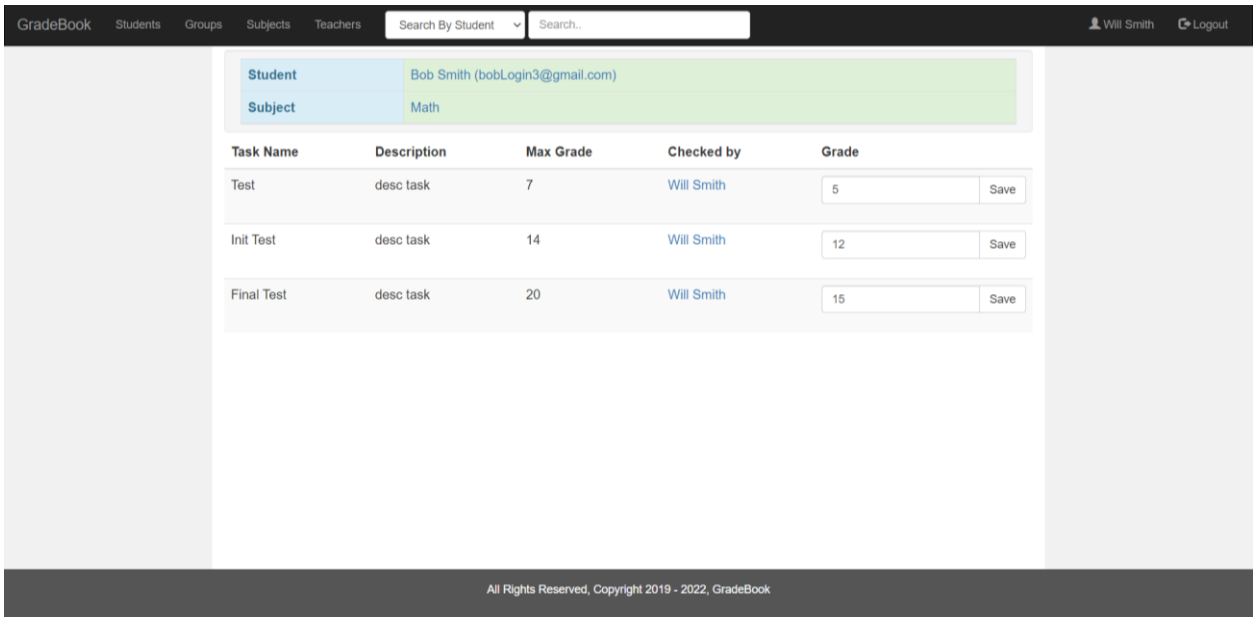

Рисунок 3.2.25 – Сторінка навчальної картки студента.

3.3. Розробка додатку

При розробці основною частиною веб-додатку практично завжди є його серверна частина, оскільки саме у цій частині розташована вся бізнес-логіка.

Більшість додатків мають багато спільних рис на їх серверній частині, але саме бізнес-логіка це те, що практично завжди є різним, адже вона завжди будується на основі того, які бізнес-процеси повинні відбуватися під час роботи веб-додатку. Саме тому для кращого опису етапу розробки більш доцільно описати його бізнес логіку. Також для графічного відображення цієї логіки зручно використовувати діаграми класів, які призначаються для опису структури системи, шляхом демонстрації класів системи їх атрибутів та методів, а також взаємовідносин між ними.[7] Зручність даних діаграм полягає в їх універсальності та простоті побудови і використання.

3.3.1. Класи Entities

Класи Entities призначені для зберігання даних об’єктів різних типів, які були отримані з бази даних. Головна їх ціль це оптимізувати структурування інформації для більш комфортного використання отриманих даних системою. До цих класів входять Group, Student, StudentInfoSet, StudentSubject, StudentTask, StudentTaskInfoSet, Subject, Task, Teacher, TeacherSubject. У вигляді діаграм класів вони представлені на рисунках 3.3.1 – 3.3.3. Самі ж ці класи не несуть в собі логічного навантаження, а є лише способами зручного збереження інформації, через це немає сенсу описувати методи, котрі вони мають.

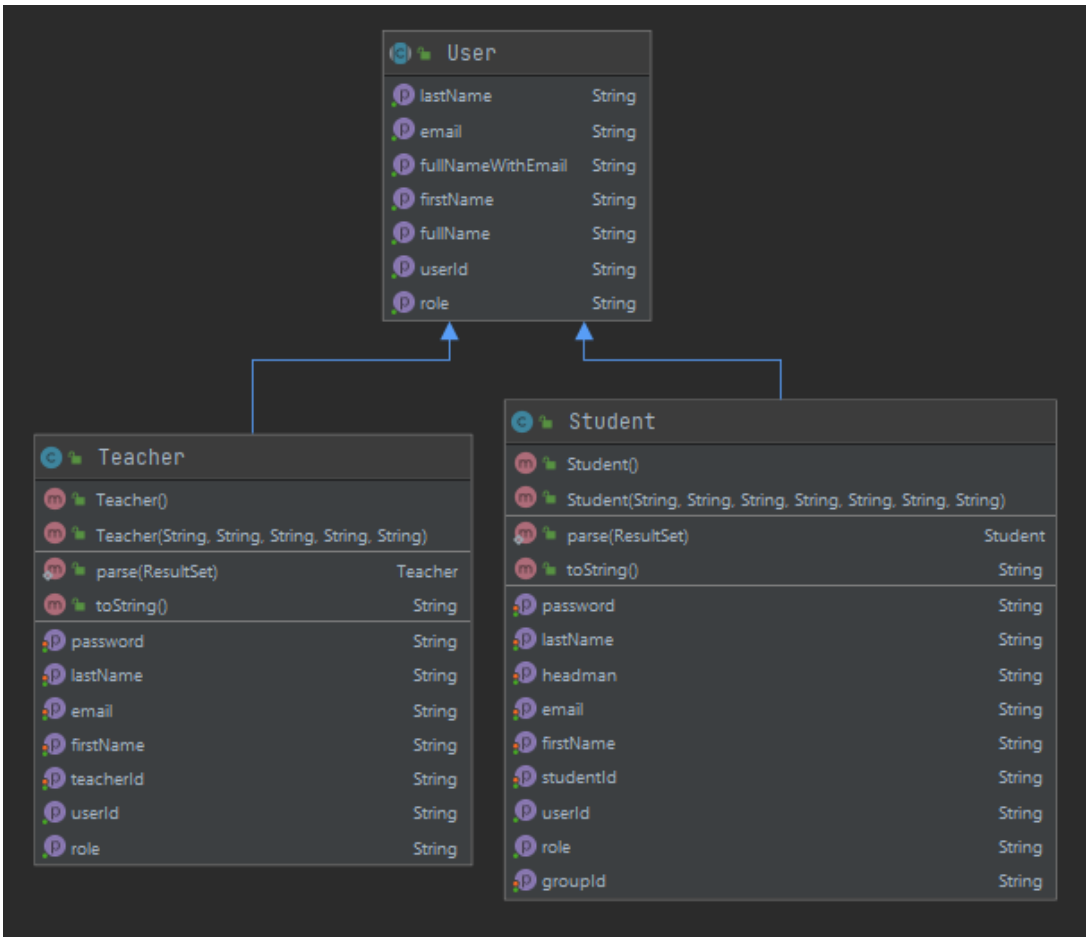

Рисунок 3.3.1 – Діаграма класів Entities, частина 1.

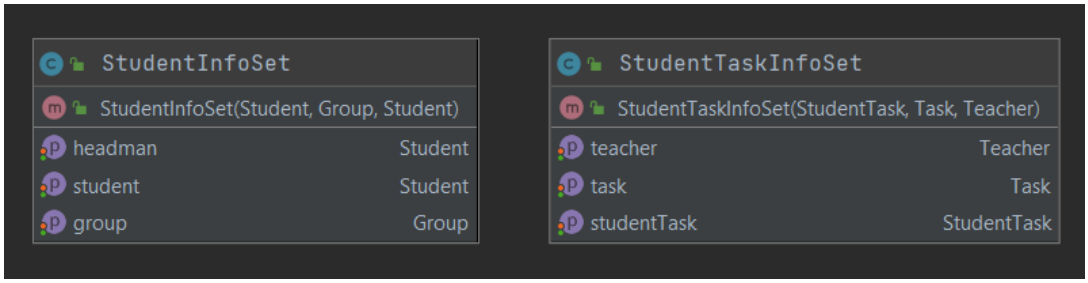

Рисунок 3.3.2 – Діаграма класів Entities, частина 2.

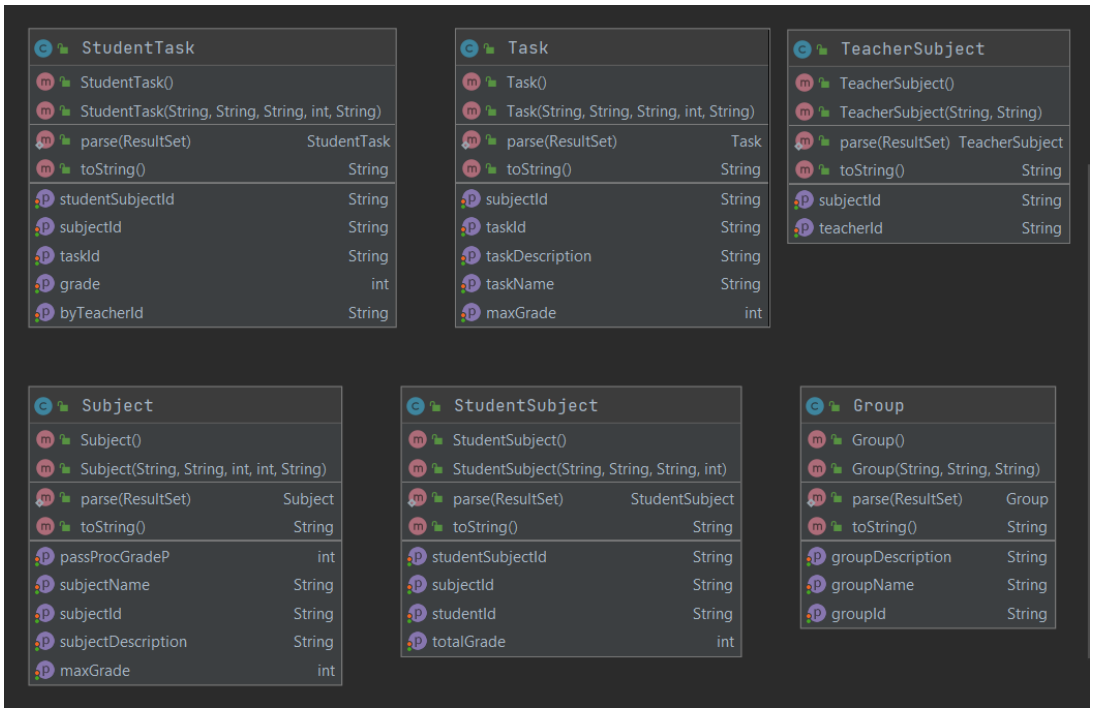

Рисунок 3.3.3 – Діаграма класів Entities, частина 3.

3.3.2. Класи DAO

Клас Oracle (див. таб. 3.3.1) призначений для встановлення та розриву зв'язку з базою даних, методи цього класу використовуються у всіх DAO класах проекту.

Таблиця 3.3.1 – Перелік усіх методів класу Oracle.

Тип	Назва	Призначення
void	connect	Здійснює приєднання до бази даних.
void	disconnect	Закриває з'єднання з базою даних.

Класи DAO призначені для виконання дій над базою даних, де кожний DAO клас має інструменти для роботи з відповідним до нього типом об'єктів. До них входять класи DAOGroupImpl, DAOStudentImpl, DAOStudentSubjectImpl, DAOStudentTaskImpl, DAOSubjectImpl, DAOTaskImpl, DAOTeacherImpl та DAOTeacherSubjectImpl. Це основні класи по роботі з базою даних в даному додатку, котрі забезпечують повноцінний обмін інформацією між додатком та базою даних. Тобто вони виконують

створення, редагування та видалення об’єктів визначених типів , де назви типів цих об’єктів та відповідних класів DAO збігаються. Також дані класи дають можливість отримання необхідних об’єктів, їх списків та навіть окремої інформації з бази даних, для подальшого використання результату додатком. У вигляді діаграм класів вони представлені на рисунках 3.3.4 – 3.3.7. Детальна інформація, що до методів кожного з таких класів наведена у таблицях 3.3.2 – 3.3.9.

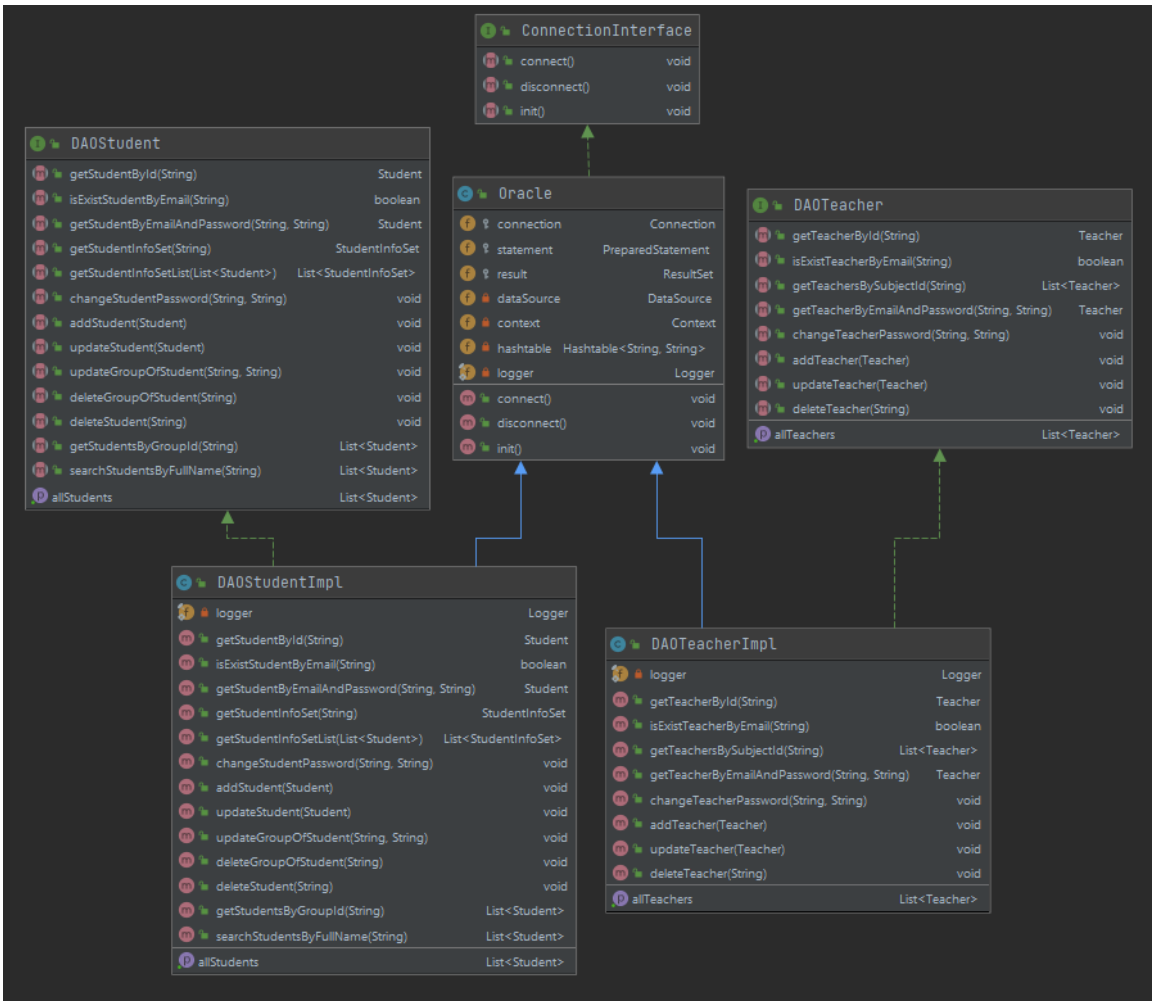

Рисунок 3.3.4 – Діаграма класів DAO, частина 1.

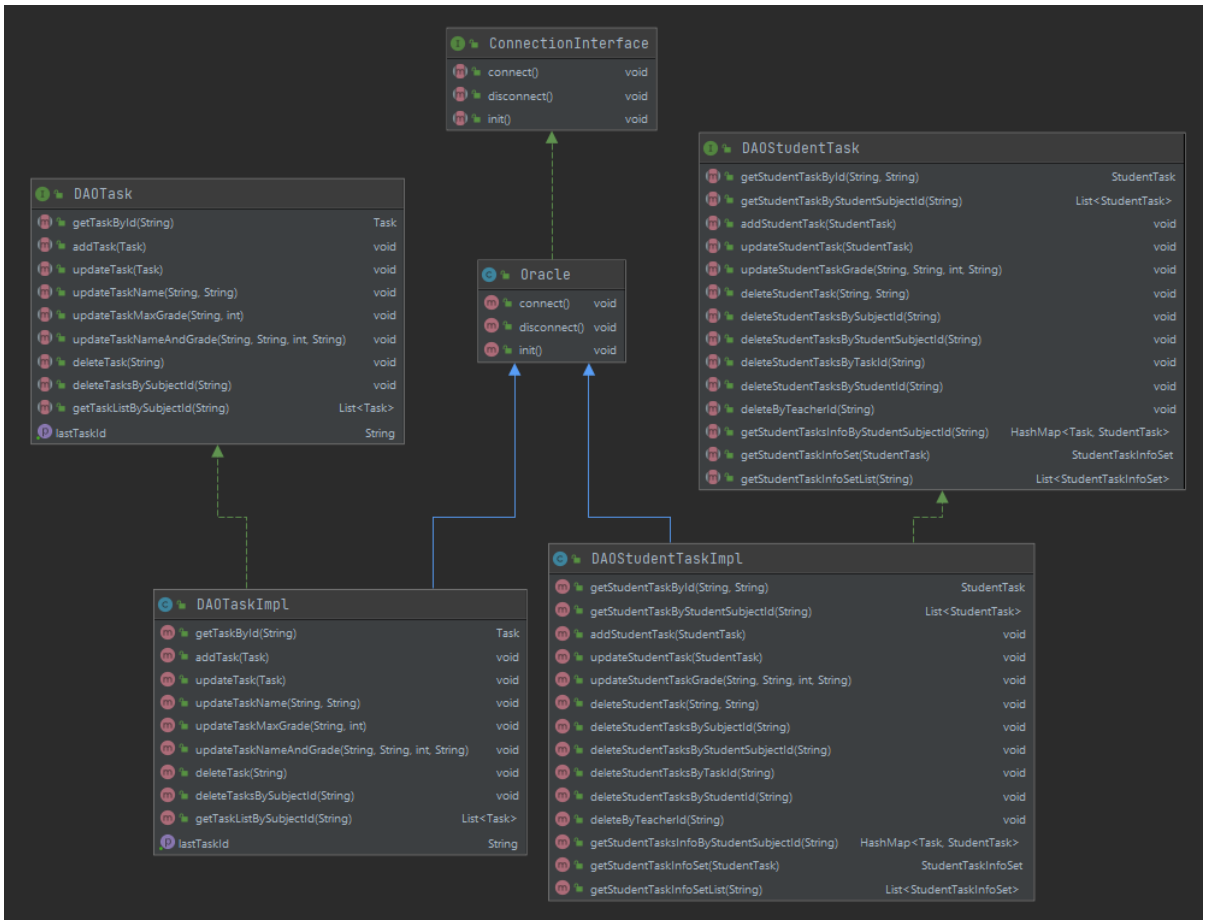

Рисунок 3.3.5 – Діаграма класів DAO, частина 2.

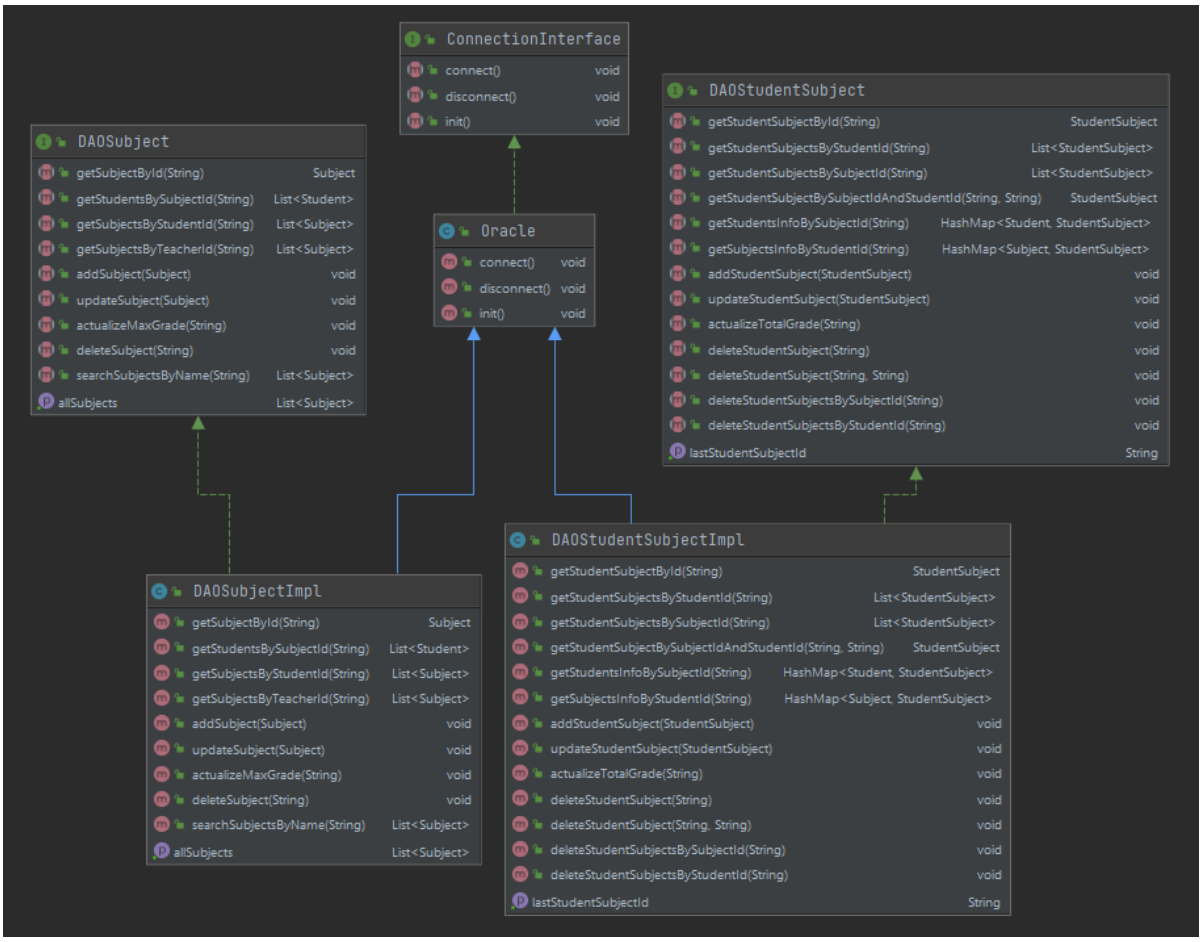

Рисунок 3.3.6 – Діаграма класів DAO, частина 3.

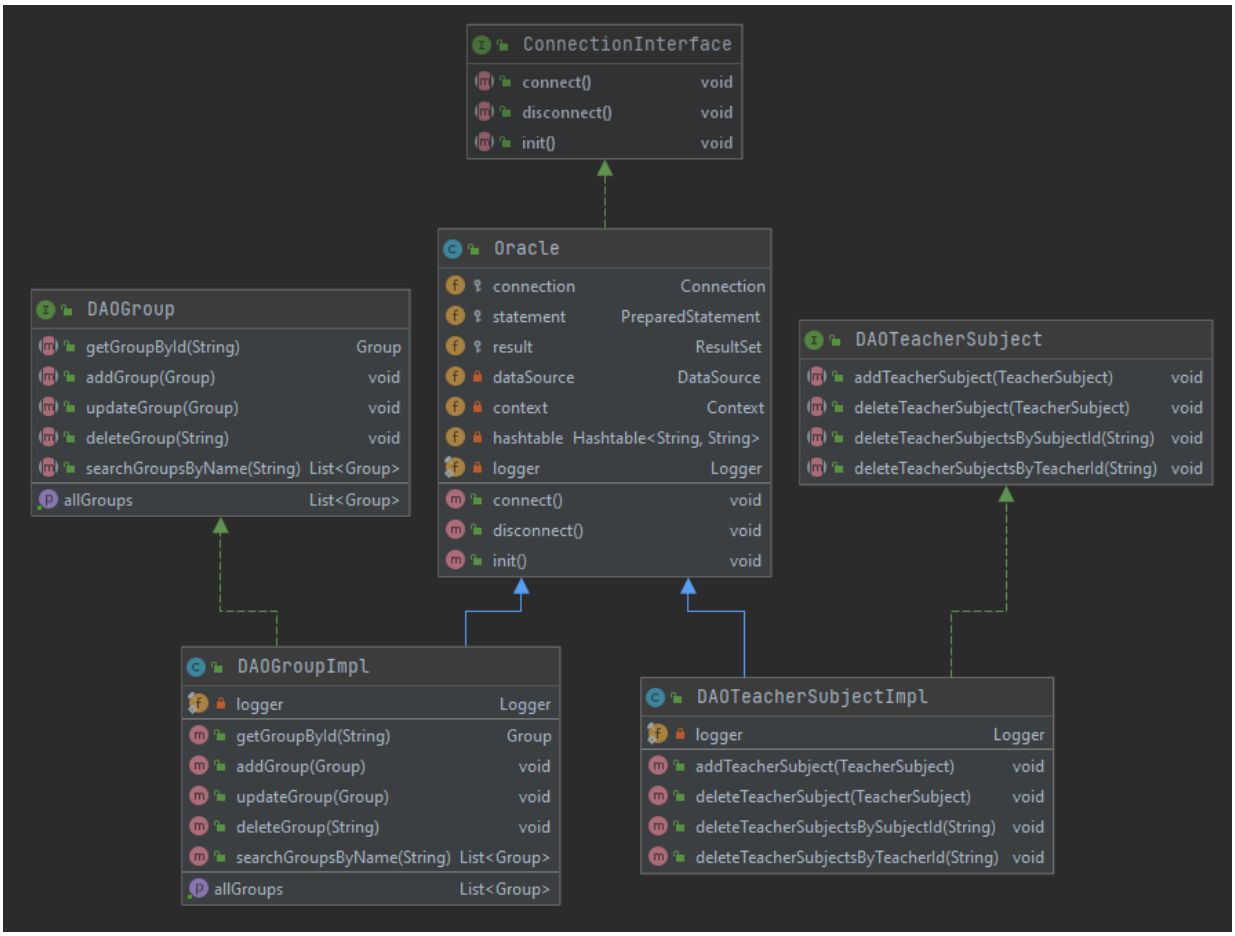

Рисунок 3.3.7 – Діаграма класів DAO, частина 4.

Таблиця 3.3.2 – Перелік усіх методів класу DAOGroupImpl.

Тип	Назва	Призначення
Group	getGroupById	Дістає дані групи з бази даних та повертає їх, як результат.
void	addGroup	Додає нову групу в базу даних.
void	updateGroup	Оновлює поля існуючої групи у базі даних.
void	deleteGroup	Видаляє групу з бази даних.
List<Group>	getAllGroups	Вибирає всі групи з бази даних та повертає їх список.
List<Group>	searchGroupsBy Name	Повертає список груп по частині їх назв.

Таблиця 3.3.3 – Перелік усіх методів класу DAOStudentImpl.

Тип	Назва	Призначення
Student	getStudentById	Дістає дані студента з бази даних та повертає їх, як результат.

Тип	Назва	Призначення

Продовження таблиці 3.3.3

Тип	Назва	Призначення
boolean	isExistStudentByEmail	Перевіряє чи є в базі даних студент с вказаною поштою, якщо такий є повертає істину.
Student	getStudentByEmailAndPassword	Шукає студента по поштовій адресі та паролю, у разі успіху повертає даного студента.
StudentInfoSet	getStudentInfoSet	Формує об’єкт, що містить у собі студента, його групу та куратора по ідентифікатору студента та повертає його.
List<StudentInfoSet>	getStudentInfoSetList	Формує список об’єктів, що містять у собі студента, його групу та куратора по ідентифікаторам студентів та повертає його.
void	changeStudentPassword	Змінює пароль студента в базі даних.
void	addStudent	Додає нового студента в базу даних.
void	updateStudent	Змінює поля існуючого студента у базі даних.
void	updateGroupOfStudent	Змінює групу студента у базі даних.
void	deleteGroupOfStudent	Видаляє у даного студента групу з бази даних.
void	deleteStudent	Видаляє студента з бази даних.
List<Student>	getStudentsByGroupId	Знаходить всіх студентів, що входять до вказаної групи та повертає їх список.
List<Student>	getAllStudents	Вибирає всіх студентів з бази даних та повертає їх список.
List<Student>	searchStudentsByFullName	Повертає список студентів по частині їх повного імені.

Таблиця 3.3.4 – Перелік усіх методів класу DAOStudentSubjectImpl.

Тип	Назва	Призначення
StudentSubject	getStudentSubjectById	Дістає зв’язок між предметом та студентом за ідентифікатором з бази даних.

Продовження таблиці 3.3.4

Тип	Назва	Призначення
List<StudentSubject>	getStudentSubjectsBySubjectId	Вибирає всі зв’язки студент-предмет для заданого предмета з бази даних та повертає їх список.
StudentSubject	getStudentSubjectBySubjectIdAndStudentId	Вибирає зв’язок студент-предмет для заданого предмета та студента з бази даних та повертає його.
HashMap<Student, StudentSubject>	getStudentsInfoBySubjectId	Формує список пар студент та студент-предмет та повертає його.
HashMap<Subject, StudentSubject>	getSubjectsInfoByStudentId	Формує список пар предмет та студент-предмет та повертає його.
void	addStudentSubject	Додає новий зв’язок студент-предмет в базу даних.
void	updateStudentSubject	Змінює дані існуючого зв’язку студент-предмет у базі даних.
void	actualizeTotalGrade	Актуалізує загальну оцінку студента за сумою оцінок всіх його завдань.
void	deleteStudentSubject	Видаляє існуючий зв’язок студент-предмет з бази даних.
void	deleteStudentSubjectsBySubjectId	Видаляє існуючий зв’язок студент-предмет за заданим предметом з бази даних.
void	deleteStudentSubjectsByStudentId	Видаляє існуючий зв’язок студент-предмет за заданим студентом з бази даних.
String	getLastStudentSubjectId	Вибирає найновіший ідентифікатор зв’язку студент-предмет та повертає його.

Таблиця 3.3.5 – Перелік усіх методів класу DAOStudentTaskImpl.

Тип	Назва	Призначення
StudentTask	getStudentTaskById	Дістає зв’язок між студентом та завданням за ідентифікаторами з бази даних.

Тип	Назва	Призначення

Продовження таблиці 3.3.5

Тип	Назва	Призначення
List<StudentTask>	getStudentTaskByStudentSubjectId	Вибирає всі зв’язки студент-завдання для заданого зв’язка студент-предмет з бази даних та повертає їх список.
void	addStudentTask	Додає новий зв’язок студент-завдання в базу даних.
void	updateStudentTaskGrade	Змінює оцінку студента за завдання у базі даних.
void	deleteStudentTask	Видаляє існуючий зв’язок студент-завдання з бази даних.
void	deleteStudentTasksBySubjectId	Видаляє існуючий зв’язок студент-завдання за заданим предметом з бази даних.
void	deleteStudentTasksByStudentSubjectId	Видаляє існуючий зв’язок студент-завдання за заданим зв’язком студент-предмет з бази даних.
void	deleteStudentTasksByTaskId	Видаляє існуючий зв’язок студент-завдання за заданим завданням з бази даних.
void	deleteStudentTasksByStudentId	Видаляє існуючий зв’язок студент-завдання за заданим студентом з бази даних.
void	deleteByTeacherId	Видаляє підписи заданого викладача на всіх зв’язках студент-завдання з бази даних.
HashMap<Task, StudentTask>	getStudentTasksInfoByStudentSubjectId	Формує список пар завдання та студент-завдання та повертає його.
StudentTaskInfoSet	getStudentTaskInfoSet	Формує об’єкт, що містить у собі завдання, зв’язок студент-завдання, викладача та повертає його.
List<StudentTaskInfoSet>	getStudentTaskInfoSetList	Формує список об’єктів, що містять у собі завдання, зв’язок студент-завдання, викладача та повертає його.

Таблиця 3.3.6 – Перелік усіх методів класу DAOSubjectImpl.

Тип	Назва	Призначення
Subject	getSubjectById	Дістає дані предмета з бази даних та повертає їх, як результат.
List<Student>	getStudentsBySubjectId	Вибирає всіх студентів за заданим предметом з бази даних та повертає їх список.
List<Subject>	getSubjectsByStudentId	Вибирає всі предмети за заданим студентом з бази даних та повертає їх список.
List<Subject>	getSubjectsByTeacherId	Вибирає всіх викладачів за заданим предметом з бази даних та повертає їх список.
void	addSubject	Додає новий предмет в базу даних.
void	updateSubject	Змінює дані існуючого предмета у базі даних.
void	actualizeMaxGrade	Актуалізує максимальну оцінку предмета за сумою максимальних оцінок всіх його завдань.
void	deleteSubject	Видаляє предмет з бази даних.
List<Subject>	getAllSubjects	Вибирає всі предмети з бази даних та повертає їх список.
List<Subject>	searchSubjectsByName	Повертає список предметів по частині їх повного імені.

Таблиця 3.3.7 – Перелік усіх методів класу DAOTaskImpl.

Тип	Назва	Призначення
Task	getTaskById	Дістає дані завдання з бази даних та повертає їх, як результат.
void	addTask	Додає нове завдання в базу даних.
void	updateTaskNameAndGrade	Змінює назву та оцінку існуючого завдання у базі даних.
void	deleteTas	Видаляє завдання з бази даних.
void	deleteTasksBySubjectId	Видаляє всі існуючі завдання за заданим предметом з бази даних.

Продовження таблиці 3.3.7

Тип	Назва	Призначення
List<Task>	getTaskListBySubjectId	Вибирає всі завдання за заданим предметом з бази даних та повертає їх список.
String	getLastTaskId	Вибирає найновіший ідентифікатор завдання та повертає його.

Таблиця 3.3.8 – Перелік усіх методів класу DAOTeacherImpl.

Тип	Назва	Призначення
Teacher	getTeacherById	Дістає дані викладача з бази даних та повертає їх, як результат.
boolean	isExistTeacherByEmail	Перевіряє чи є в базі даних викладач с вказаною поштою, якщо такий є повертає істину.
List<Teacher>	getTeachersBySubjectId	Вибирає всіх викладачів, що ведуть вказаний предмет з бази даних та повертає їх список.
Teacher	getTeacherByEmailAndPassword	Шукає викладача по поштовій адресі та паролю, у разі успіху повертає даного викладача.
void	changeTeacherPassword	Змінює пароль викладача в базі даних.
void	addTeacher	Додає нового викладача в базу даних.
void	updateTeacher	Змінює дані існуючого викладача у базі даних.
void	deleteTeacher	Видаляє викладача з бази даних.
List<Teacher>	getAllTeachers	Вибирає всіх викладачів з бази даних та повертає їх список.

Таблиця 3.3.9 – Перелік усіх методів класу DAOTeacherSubjectImpl.

Тип	Назва	Призначення
void	addTeacherSubject	Додає новий зв’язок між викладачем та предметом до бази даних.

Продовження таблиці 3.3.9

Тип	Назва	Призначення
void	deleteTeacherSubject	Видаляє існуючий зв’язок між викладачем та предметом з бази даних.
void	deleteTeacherSubjectsBySubjectId	Видаляє всі існуючі зв’язки між викладачами та вказаним предметом з бази даних.
void	deleteTeacherSubjectsByTeacherId	Видаляє всі існуючі зв’язки між вказаним викладачем та предметами з бази даних.

3.3.3. Класи Controllers

Класи Controllers призначені для обробки запитів, які здійснює користувач при взаємодії з інтерфейсом додатку. Залежно від запиту ці класи виконують набір дій, котрі повинні надавати користувачу результат, який він прагне отримати шляхом взаємодії з інтерфейсом. Класи контролери поділені між собою за спрямованістю дій котрі вони виконують. Так клас AddController відповідає за обробку тих запитів результатом яких є створення нових об’єктів, DeleteController за видалення існуючих об‘єктів, EditController за редагування інсуючих об’єктів, LoginController за авторизацію, LogoutController за вихід з облікового запису RedirectController за перенаправлення користувача на сторінки певних форм, SearchController за обробку пошукових запитів по типу та назві об’єктів та ShowController за відображення сторінок додатку з усією необхідною на них інформацією. У вигляді діаграм класів вони представлені на рисунках 3.3.8 – 3.3.9. Детальна інформація, що до методів кожного з таких класів наведена у таблицях 3.3.10 – 3.3.17.

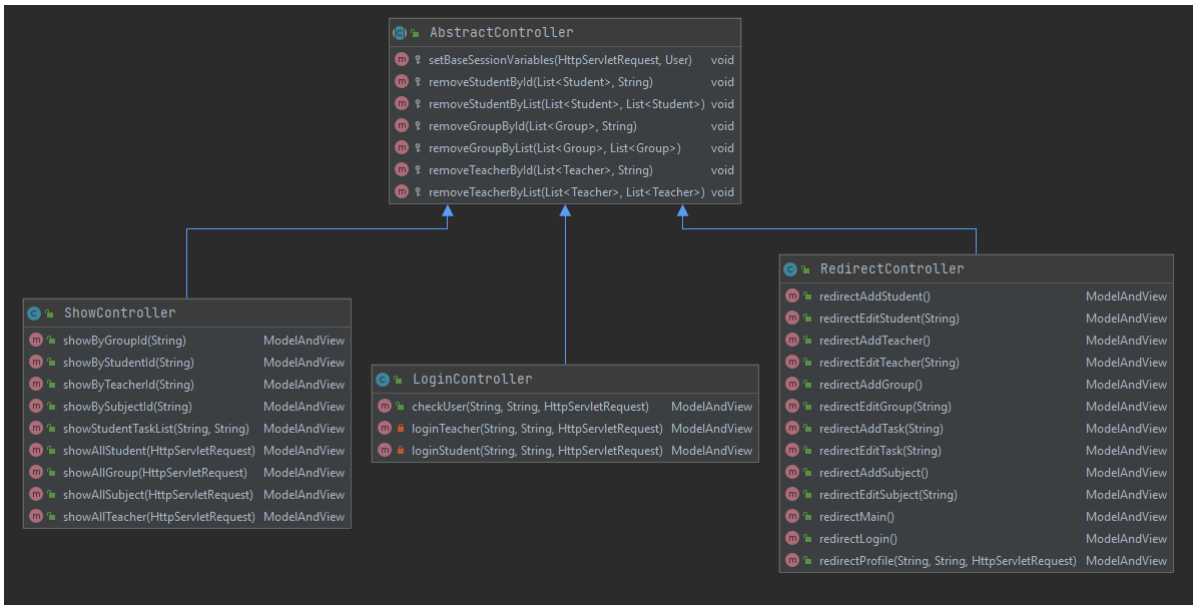

Рисунок 3.3.8 – Діаграма класів Controllers, частина 1.

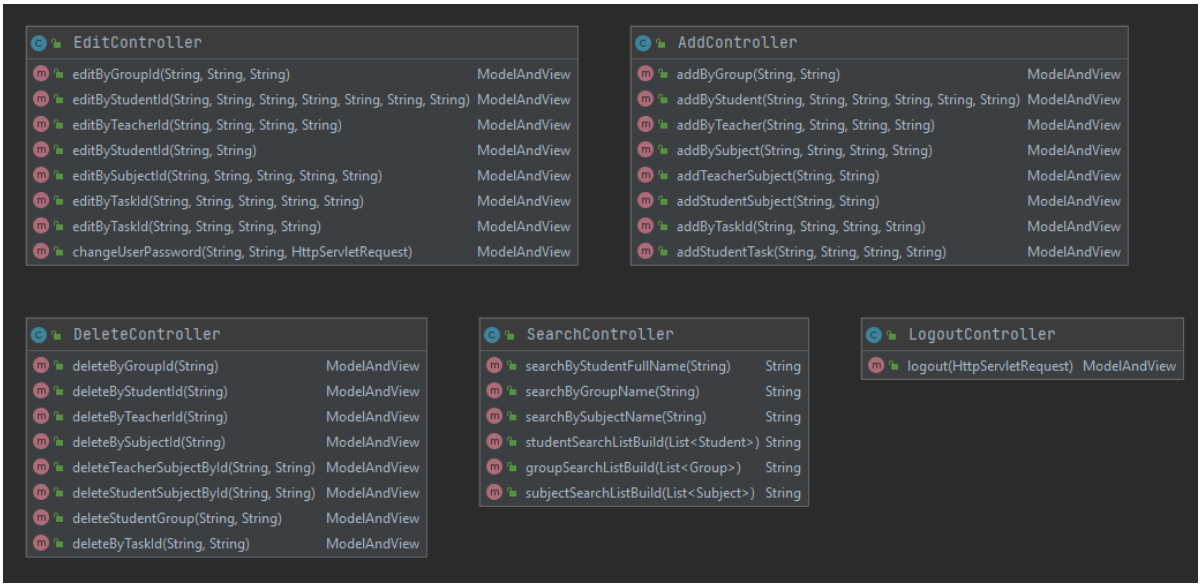

Рисунок 3.3.9 – Діаграма класів Controllers, частина 2.

Таблиця 3.3.10 – Перелік усіх методів класу AddController.

Тип	Назва	Призначення
ModelAndView	addNewGroup	Обробляє запит форми на створення нової групи, при наявності всіх даних нова група створюється.

Продовження таблиці 3.3.10

Тип	Назва	Призначення
ModelAnd View	addNewStudent	Обробляє запит форми на створення нового студента, при наявності всіх даних новий студент створюється.
ModelAnd View	addNewTeacher	Обробляє запит форми на створення нового викладача, при наявності всіх даних новий викладач створюється.
ModelAnd View	addNewSubject	Обробляє запит форми на створення нового предмета, при наявності всіх даних новий предмет створюється.
ModelAnd View	addNewTeacherSu bject	Обробляє запит на приєднання нового викладача до предмету, та у разі успіху створює новий зв’язок викладач-предмет.
ModelAnd View	addNewStudentSu bject	Обробляє запит на приєднання нового студента до предмету, та у разі успіху створює новий зв’язок студент-предмет.
ModelAnd View	addNewTask	Обробляє запит форми на створення нового завдання, при наявності всіх даних нове завдання створюється.

Таблиця 3.3.11 – Перелік усіх методів класу DeleteController.

Тип	Назва	Призначення
ModelAnd View	deleteByGroupId	Обробляє запит на видалення групи по ідентифікатору, у разі використання видаляє саму групу та посилання на неї з аккаунтів студентів.
ModelAnd View	deleteByStudentId	Обробляє запит на видалення студента по ідентифікатору, у разі використання видаляє самого студента, зв’язки студент-предмет та всі завдання зв’язані з студентом.

Продовження таблиці 3.3.11

Тип	Назва	Призначення
ModelAndView	deleteByTeacherId	Обробляє запит на видалення викладача по ідентифікатору, у разі використання видаляє самого викладача, зв’язки викладач-предмет зв’язані з викладачем.
ModelAndView	deleteBySubjectId	Обробляє запит на видалення предмет по ідентифікатору, у разі використання видаляє сам предмет, зв’язки викладач-предмет та студент-предмет та всі завдання зв’язані з даним предметом.
ModelAndView	deleteTeacherSubjectById	Обробляє запит від’єднання викладача від предмету та у разі успіху видаляє зв’язок викладач-предмет.
ModelAndView	deleteStudentSubjectById	Обробляє запит від’єднання студента від предмету та у разі успіху видаляє зв’язок студент-предмет.
ModelAndView	deleteStudentGroup	Обробляє запит від’єднання студента від групи та у разі успіху видаляє у студента посилання на групу.
ModelAndView	deleteByTaskId	Обробляє запит на видалення завдання по ідентифікатору, у разі використання видаляє саме завдання та всі зв’язки студент-завдання зв’язані з даним завданням.

Таблиця 3.3.12 – Перелік усіх методів класу EditController.

Тип	Назва	Призначення
ModelAndView	editGroup	Обробляє запит на редагування інформації групи та у разі успіху зміни зберігаються.

Продовження таблиці 3.3.12

Тип	Назва	Призначення
ModelAndView	editStudent	Обробляє запит на редагування інформації студента та у разі успіху зміни зберігаються.
ModelAndView	editTeacher	Обробляє запит на редагування інформації викладача та у разі успіху зміни зберігаються.
ModelAndView	editStudentGroup	Обробляє запит на редагування посилання на групу у студента та у разі успіху зміни зберігаються.
ModelAndView	editSubject	Обробляє запит на редагування інформації предмета та у разі успіху зміни зберігаються.
ModelAndView	editTask	Обробляє запит на редагування інформації завдання та у разі успіху зміни зберігаються.
ModelAndView	editStudentTask	Обробляє запит на редагування оцінки за завдання для вказаного студента та у разі успіху зміни зберігаються.
ModelAndView	changeUserPassword	Обробляє запит на зміну паролю студента чи викладача та у разі успіху зміни зберігаються.

Таблиця 3.3.13 – Перелік усіх методів класу LoginController.

Тип	Назва	Призначення
ModelAndView	checkUser	Обробляє запит на авторизацію користувача та залежно від ролі користувача передає управління методу loginTeacher або loginStudent.
ModelAndView	loginTeacher	Обробляє запит на авторизацію викладача та у разі наявності правильного логіну та паролю авторизація є успішною.
ModelAndView	loginStudent	Обробляє запит на авторизацію студента та у разі наявності правильного логіну та паролю авторизація є успішною.

Таблиця 3.3.14 – Перелік усіх методів класу LogoutController.

Тип	Назва	Призначення
ModelAndView	logout	Обробляє запит на вихід з поточного облікового запису та перенаправляє на сторінку авторизації.

Таблиця 3.3.15 – Перелік усіх методів класу RedirectController.

Тип	Назва	Призначення
ModelAndView	redirectAddStudent	Обробляє запит на відкриття вікна з формою для створення студента, будує її і показує користувачу.
ModelAndView	redirectEditStudent	Обробляє запит на відкриття вікна з формою для редагування студента, будує її і показує користувачу.
ModelAndView	redirectAddTeacher	Обробляє запит на відкриття вікна з формою для створення викладача, будує її і показує користувачу.
ModelAndView	redirectEditTeacher	Обробляє запит на відкриття вікна з формою для редагування викладача, будує її і показує користувачу.
ModelAndView	redirectAddGroup	Обробляє запит на відкриття вікна з формою для створення групи, будує її і показує користувачу.
ModelAndView	redirectEditGroup	Обробляє запит на відкриття вікна з формою для редагування групи, будує її і показує користувачу.
ModelAndView	redirectAddTask	Обробляє запит на відкриття вікна з формою для створення завдання, будує її і показує користувачу.
ModelAndView	redirectEditTask	Обробляє запит на відкриття вікна з формою для редагування завдання, будує її і показує користувачу.

Продовження таблиці 3.3.15

Тип	Назва	Призначення
ModelAndView	redirectAddSubject	Обробляє запит на відкриття вікна з формою для створення предмета, будує її і показує користувачу.
ModelAndView	redirectEditSubject	Обробляє запит на відкриття вікна з формою для редагування предмета, будує її і показує користувачу.

Таблиця 3.3.16 – Перелік усіх методів класу SearchController.

Тип	Назва	Призначення
@ResponseBody String	searchByStudentFullName	Обробляє запит на пошук студентів по частині їх імені, що надсилається пошуковою системою та повертає їх список у необхідному форматі.
@ResponseBody String	searchByGroupName	Обробляє запит на пошук груп по частині їх назв, що надсилається пошуковою системою та повертає їх список у необхідному форматі.
@ResponseBody String	searchBySubjectName	Обробляє запит на пошук предметів по частині їх назв, що надсилається пошуковою системою та повертає їх список у необхідному форматі.
String	studentSearchListBuild	Трансформує список студентів у правильний формат для відображення.
String	groupSearchListBuild	Трансформує список груп у правильний формат для відображення.
String	subjectSearchListBuild	Трансформує список предметів у правильний формат для відображення.

Таблиця 3.3.17 – Перелік усіх методів класу ShowController.

Тип	Назва	Призначення
ModelAndView	showByGroupId	Обробляє запит на показ сторінки вказаної групи, отримує всю необхідну інформації з бази даних та показує сторінку групи користувачу.
ModelAndView	showByStudentId	Обробляє запит на показ профіля студента, отримує всю необхідну інформації з бази даних та показує даний профіль користувачу.
ModelAndView	showByTeacherId	Обробляє запит на показ профіля викладача, отримує всю необхідну інформації з бази даних та показує даний профіль користувачу.
ModelAndView	showBySubjectId	Обробляє запит на показ сторінки вказаної групи, отримує всю необхідну інформації з бази даних та показує сторінку групи користувачу.
ModelAndView	showStudentTaskList	Обробляє запит на показ сторінки зі списком всіх завдань студента по вказаному предмету, отримує всю необхідну інформації з бази даних та показує сторінку з даною інформацією.
ModelAndView	showAllStudent	Обробляє запит на показ сторінки зі списком всіх студентів, отримує його з бази даних та показує сторінку з даним списком.
ModelAndView	showAllGroup	Обробляє запит на показ сторінки зі списком всіх груп, отримує його з бази даних та показує сторінку з даним списком.

Продовження таблиці 3.3.17

Тип	Назва	Призначення
ModelAnd View	showAllSubject	Обробляє запит на показ сторінки зі списком всіх предметів, отримує його з бази даних та показує сторінку з даним списком.
ModelAnd View	showAllTeacher	Обробляє запит на показ сторінки зі списком всіх вчителів, отримує його з бази даних та показує сторінку з даним списком.

По ходу цього розділу було повністю описано основну бізнес логіку додатка, з демонстрацією його структури за допомогою діаграм класів та переліком основним методів класів і їх функціонального призначення.

ВИСНОВКИ

У ході дипломної роботи був виконаний порівняльний аналіз двох існуючих аналогів та на його основі сформовані вимоги до додатку. Спираючись на вимоги та ідею самого додатку були сформовані завдання , котрі були виконані по ходу даної роботи. А саме, була спроектована база даних, результат проектування був представлений у вигляді ERD діаграми. За мову програмування, яка якій розробляється додаток, була обрана Java. Основними фреймворками було вибрано Spring Framework, що включає в себе масу інших фреймворків, JSP сторінки для побудови контенту сторінок додатку та Bootstrap для надання сторінкам додатку привабливого вигляду. Використовуючи діаграми послідовностей було наведено основні сценарії дій, котрі може виконувати користувач, тим самим було продемонстровано основний функціонал додатку.

У ході програмної реалізації, спираючись на ERD діаграму, була побудована база даних, що їй повністю відповідає. Структура бази даних була представлена у вигляді таблиць, з описом всіх полів, що належать їй. Також був представлений інтерфейс всього додатку, з описом кожної сторінки та функцій, що мають виконувати ті чи інші елементи на цих сторінках. Ця демонстрація інтерфейсу дозволяє в наочному прикладі переглянути, як саме звичайний користувач може взаємодіяти з нашою системою. Після представлення інтерфейсу було розроблено основну частину нашого додатку, котра відповідала за його функціональні можливості. Цей процес розробки спирався на представлені раніше діаграми послідовностей. Для демонстрації даного процесу були використані діаграми класів, та таблиці, що містять опис основних методів нашої інформаційної системи.

Згідно з результатами дипломної роботи можна зробити висновок, що було досягнуто її мети. А саме було побудовано систему моніторингу навчальних досягнень учнів та студентів. Котра спрямована на полегшення

ведення оцінювання навчального процесу. Вона представлена у вигляді веб-додатку та успішно використовує базу даних для зберігання результатів своєї роботи. Крім цього дана система надає достатній функціонал для ефективного користування.

Завдяки тому, що дані системи основані на обміні інформацією між студентами та викладачами, є велика кількість ідей котрі можна використовувати для їх вдосконалення. Через це розвиток систем даного виду буде актуальною темою ще тривалий час, це і забезпечує їх актуальність в наші дні.

СПИСОК ЛІТЕРАТУРИ

1. Oracle Database [Електронний ресурс] – Режим доступу до ресурсу: https://docs.oracle.com/cd/E11882_01/server.112/e40540/intro.htm#CNCPT00.
2. Entity Relationship diagram [Електронний ресурс] – Режим доступу до ресурсу: https://www.datensen.com/blog/er-diagram/what-is-entity-relationship-diagram-erd/.
3. Java Documentations [Електронний ресурс] – Режим доступу до ресурсу: https://docs.oracle.com/en/java/.
4. Spring framework documentations [Електронний ресурс] – Режим доступу до ресурсу: https://docs.spring.io/spring-framework/docs /current/reference/html/.
5. Bootstrap documentations [Електронний ресурс] – Режим доступу до ресурсу: https://getbootstrap.com/docs/5.2/getting-started/introduction/.
6. Sequence diagram [Електронний ресурс] – Режим доступу до ресурсу:https://www.sciencedirect.com/topics/computer-science/ sequence-diagram.
7. UML Class Diagrams [Електронний ресурс] – Режим доступу до ресурсу: https://www.codemag.com/article/0201061/UML-Class-Diagrams.
8. Rod Johnson, Alef Arendsen, Thomas Risberg. Professional Java Development with the Spring Framework - Amazon 2005.
9. Gerardus Blokdyk. Oracle Database A Complete Guide - 2019 Edition - Amazon 2021.

ДОДАТОК 1. РЕАЛІЗАЦІЯ БД

У цьому додатку наведений повний текст бази даних, що використовується нашої інформаційною системою.

```
CREATE SEQUENCE user_id_seq;
CREATE SEQUENCE subject_id_seq;
CREATE SEQUENCE task_id_seq;
CREATE SEQUENCE group_id_seq;
CREATE SEQUENCE student_subject_id_seq;

Create table  student(
   student_id  INTEGER NOT NULL ,
   headman INTEGER,
   email varchar2(30) UNIQUE,
   first_name varchar2(30),
   last_name varchar2(30),
   group_id INTEGER ,
   password varchar2(30),
primary key(student_id)
);

Create table  teacher(
   teacher_id INTEGER NOT NULL ,
   email varchar2(30) UNIQUE,
   first_name varchar2(30),
   last_name varchar2(30),
   password varchar2(30),
primary key(teacher_id)
);

Create table  subject(
   subject_id INTEGER NOT NULL ,
   subject_name varchar2(30),
   max_grade INTEGER NOT NULL,
   pass_proc_grade INTEGER NOT NULL,
   subject_description varchar2(1000),
primary key(subject_id)
);

Create table  task(
   task_id INTEGER NOT NULL ,
   subject_id INTEGER NOT NULL,
   task_name varchar2(30),
   max_grade INTEGER NOT NULL,
   task_description varchar2(4000),
primary key(task_id)
);

Create table  group_(
   group_id INTEGER NOT NULL ,
   group_name varchar2(30),
   group_description varchar2(1000),
primary key(group_id)
);
```

```
Create table  student_subject(
    student_subject_id INTEGER NOT NULL ,
    total_grade INTEGER ,
    student_id INTEGER NOT NULL ,
   subject_id INTEGER NOT NULL ,
primary key(student_subject_id)
);

Create table  student_task(
    student_subject_id INTEGER NOT NULL ,
    task_id INTEGER NOT NULL ,
   subject_id INTEGER NOT NULL ,
   grade INTEGER,
    by_teacher_id INTEGER
);

Create table teacher_subject(
   subject_id INTEGER NOT NULL ,
   teacher_id INTEGER NOT NULL
);
```

ДОДАТОК 2. ЗМІСТ JSP СТОРІНОК

У цьому додатку представлений лістинг коду всіх JSP сторінок, які відповідають за формування контенту сторінок, котрі відображаються користувачам нашого веб-додатку.

Сторінка “add-edit-group-page.jsp”

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>${action}</title>
</head>
<c:url value='/show/group-all' var="allGroupURL"/>
<body>
    <jsp:include page="header.jsp" />
    <div class="container-fluid text-center">
        <div class="row content  my-c">
            <div class="col-xs-8 col-xs-offset-2 text-left my">
                <c:choose>
                    <c:when test="${action eq 'edit'}">
                        <form action="<c:url
value='/${action}/group/${group.getGroupId()}' />" method="POST">
                            <div class="form-group">
                                <label for="groupName">Group name: </label>
                                <input type="text" class="form-control"
name="groupName" id="groupName" required value="${group.getGroupName()}"
placeholder="Enter Group Name">
                            </div>
                            <div class="form-group">
                                <label for="groupDescription">Group
description: </label>
                                <textarea name="groupDescription"
                                id="groupDescription"
                                class="form-control"
                                cols="30"
                                rows="5"
                                maxlength="1000"
                                placeholder="Enter Group Description"
required>${group.getGroupDescription()}</textarea>
                            </div>
                    </c:when>
                    <c:otherwise>
                        <form action="<c:url value='/${action}/group' />"
method="POST">
                            <div class="form-group">
                                <label for="groupNameA">Group name: </label>
                                <input type="text" class="form-control"
name="groupName" id="groupNameA" required value="" placeholder="Enter Group
Name">
                            </div>
                            <div class="form-group">
                                <label for="groupDescriptionA">Group
```

```
description: </label>
                                    <textarea name="groupDescription"
                                     id="groupDescriptionA"
                                     class="form-control"
                                    cols="30"
                                    rows="5"
                                    maxlength="1000"
                                    placeholder="Enter Group Description"
required></textarea>
                                </div>
                            </c:otherwise>
                        </c:choose>
                        <button type="submit" class="btn btn-
default">${action}</button>
                            <c:choose>
                                <c:when test="${action eq 'edit'}">
                                    <a href="<c:url
value='/show/group?groupId=${group.getGroupId()}'/>" class="btn btn-
default">Cancel</a>
                                </c:when>
                                <c:otherwise>
                                    <a href="<c:out value="${allGroupURL}"/>"
class="btn btn-default">Cancel</a>
                                </c:otherwise>
                            </c:choose>
                        </form>
                    </div>
                </div>
            </div>
            <jsp:include page="footer.jsp" />
</body>
</html>
```

Сторінка “add-edit-student-page.jsp”

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>${action}</title>
</head>

<body>
    <c:url value='/show/student-all' var="allStudentURL"/>
    <jsp:include page="header.jsp" />
    <div class="container-fluid text-center">
        <div class="row content my-c">
            <div class="col-xs-8 col-xs-offset-2 text-left my">
        <c:choose>
            <c:when test="${action eq 'edit'}">
            <form action="<c:url
value='/${action}/student/${studentInfoSet.getStudent().getStudentId()}' />"
method="POST">
                <div class="form-group">
                    <label for="loginName">Email: </label>
                    <input type="text" name="loginName" id="loginName"
```

```
class="form-control" required
value="${studentInfoSet.getStudent().getEmail()}" placeholder="Enter Email"/>
                </div>
                <div class="form-group">
                    <label for="firstName">First name: </label>
                    <input type="text" name="firstName" id="firstName"
class="form-control" required
value="${studentInfoSet.getStudent().getFirstName()}" placeholder="Enter
First Name"/>
                </div>
                <div class="form-group">
                    <label for="lastName">Last name: </label>
                    <input type="text" name="lastName" id="lastName"
class="form-control" required
value="${studentInfoSet.getStudent().getLastName()}" placeholder="Enter Last
Name"/>
                </div>
                <div class="form-group">
                    <label for="headman">Headman: </label>
                    <select name="headman" id="headman" class="form-control">
                        <c:choose>
                            <c:when test="${empty
studentInfoSet.getStudent().getHeadman()}">
                                <option value="NotYet" selected="selected">no
headman yet</option>
                            </c:when>
                            <c:otherwise>
                                <option
value="${studentInfoSet.getHeadman().getStudentId()}"
selected="selected">${studentInfoSet.getHeadman().getFullName()}</option>
                                <option value="NotYet">no headman
yet</option>
                            </c:otherwise>
                        </c:choose>

                        <c:forEach var="headman" items="${headmanList}">
                            <option
value="${headman.getStudentId()}">${headman.getFullName()}</option>
                        </c:forEach>
                    </select>
                </div>
                <div class="form-group">
                    <label for="groupId">Group: </label>
                    <select name="groupId" id="groupId" class="form-control">
                        <c:choose>
                            <c:when test="${empty
studentInfoSet.getStudent().getGroupId()}">
                                <option value="NotYet" selected="selected">no
headman yet</option>
                            </c:when>
                            <c:otherwise>
                                <option
value="${studentInfoSet.getGroup().getGroupId()}"
selected="selected">${studentInfoSet.getGroup().getGroupName()}</option>
                                <option value="NotYet">no headman
yet</option>
                            </c:otherwise>
                        </c:choose>
```

```
                        <c:forEach var="group" items="${groupList}">
                            <option
value="${group.getGroupId()}">${group.getGroupName()}</option>
                        </c:forEach>
                    </select>
                </div>
            </c:when>
            <c:otherwise>
                <form action="<c:url value='/${action}/student' />"
method="POST">
                    <div class="form-group">
                        <label for="loginNameA">Email: </label>
                        <input type="text" name="loginName" id="loginNameA"
class="form-control" required value="" placeholder="Enter Email"/>
                    </div>
                    <div class="form-group">
                        <label for="firstNameA">First name: </label>
                        <input type="text" name="firstName" id="firstNameA"
class="form-control" required value="" placeholder="Enter First Name"/>
                    </div>
                    <div class="form-group">
                        <label for="lastNameA">Last name: </label>
                        <input type="text" name="lastName" id="lastNameA"
class="form-control" required value="" placeholder="Enter Last Name"/>
                    </div>
                    <div class="form-group">
                        <label for="headmanA">Headman: </label>
                        <select name="headman" id="headmanA" class="form-
control">
                            <option value="NotYet" selected="selected">no
headman yet</option>
                            <c:forEach var="headman" items="${headmanList}">
                                <option
value="${headman.getStudentId()}">${headman.getFullName()}</option>
                            </c:forEach>
                        </select>
                    </div>
                    <div class="form-group">
                        <label for="groupIdA">Group: </label>
                        <select name="groupId" id="groupIdA" class="form-
control">
                            <option value="NotYet" selected="selected">no
group yet</option>
                            <c:forEach var="group" items="${groupList}">
                                <option
value="${group.getGroupId()}">${group.getGroupName()}</option>
                            </c:forEach>
                        </select>
                    </div>
                    <div class="form-group">
                        <label for="password">Password: </label>
                        <input type="text" name="password" id="password"
class="form-control" required value="" placeholder="Enter Password"/>
                    </div>
            </c:otherwise>
        </c:choose>
        <button type="submit" class="btn btn-default">${action}</button>
```

```
            <c:choose>
                <c:when test="${action eq 'edit'}">
                    <a href="<c:url
value='/show/student?studentId=${studentInfoSet.getStudent().getStudentId()}'
/>" class="btn btn-default">Cancel</a>
                </c:when>
                <c:otherwise>
                    <a href="<c:out value="${allStudentURL}"/>" class="btn
btn-default">Cancel</a>
                </c:otherwise>
            </c:choose>
        </form>
            </div>
        </div>
    </div>
    <jsp:include page="footer.jsp" />
</body>
</html>
```

Сторінка “add-edit-subject-page.jsp”

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>${action}</title>
</head>
<c:url value='/show/subject-all' var="allSubjectURL"/>
<body>
    <jsp:include page="header.jsp" />
    <div class="container-fluid text-center">
        <div class="row content my-c">
            <div class="col-xs-8 col-xs-offset-2 text-left my">
                <c:choose>
                    <c:when test="${action eq 'edit'}">
                        <form action="<c:url
value='/${action}/subject/${subject.getSubjectId()}' />" method="POST">
                            <div class="form-group">
                                <label for="subjectName">Subject name:
</label>
                                <input type="text" name="subjectName"
id="subjectName" class="form-control" required
value="${subject.getSubjectName()}" placeholder="Enter Subject Name">
                            </div>
                            <div class="form-group">
                                <label for="maxGrade">Max grade: </label>
                                <input type="text" name="maxGrade"
id="maxGrade" class="form-control" required value="${subject.getMaxGrade()}"
placeholder="Enter Max Grade">
                            </div>
                            <div class="form-group">
                                <label for="passProcGrade">Pass grade(%):
</label>
                                <input type="text" name="passProcGrade"
id="passProcGrade" class="form-control" required
value="${subject.getPassProcGradeP()}" placeholder="Enter Pass Proc Grade">
```

```
                            </div>
                            <div class="form-group">
                                <label for="subjectDescription">Subject
description: </label>
                                <textarea name="subjectDescription"
                                    id="subjectDescription"
                                    class="form-control"
                                    cols="30"
                                    rows="5"
                                    maxlength="1000"
                                    placeholder="Enter Subject Description"
required>${subject.getSubjectDescription()}</textarea>
                            </div>
                    </c:when>
                    <c:otherwise>
                        <form action="<c:url value='/${action}/subject' />"
method="POST">
                            <div class="form-group">
                                <label for="subjectNameA">Subject name:
</label>
                                <input type="text" name="subjectName"
id="subjectNameA" class="form-control" required value="" placeholder="Enter
Subject Name">
                            </div>
                            <div class="form-group">
                                <label for="maxGradeA">Max grade: </label>
                                <input type="text" name="maxGrade"
id="maxGradeA" class="form-control" required value="" placeholder="Enter Max
Grade">
                            </div>
                            <div class="form-group">
                                <label for="passProcGradeA">Pass grade(%):
</label>
                                <input type="text" name="passProcGrade"
id="passProcGradeA" class="form-control" required value="" placeholder="Enter
Pass Proc Grade">
                            </div>
                            <div class="form-group">
                                <label for="subjectDescriptionA">Subject
description: </label>
                                <textarea name="subjectDescription"
                                          id="subjectDescriptionA"
                                          class="form-control"
                                          cols="30"
                                          rows="5"
                                          maxlength="1000"
                                          placeholder="Enter Subject
Description" required></textarea>
                            </div>
                    </c:otherwise>
                </c:choose>
                <button type="submit" class="btn btn-
default">${action}</button>
                    <c:choose>
                        <c:when test="${action eq 'edit'}">
                            <a href="<c:url
value='/show/subject?subjectId=${subject.getSubjectId()}'/>" class="btn btn-
default">Cancel</a>
```

```
                        </c:when>
                        <c:otherwise>
                            <a href="<c:out value="${allSubjectURL}"/>"
class="btn btn-default">Cancel</a>
                        </c:otherwise>
                    </c:choose>
                </form>
            </div>
        </div>
    </div>
    <jsp:include page="footer.jsp" />
</body>
</html>
```

Сторінка “add-edit-task-page.jsp”

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>${action} Task</title>
</head>

<body>
<jsp:include page="header.jsp" />
<div class="container-fluid text-center">
    <div class="row content my-c">
        <div class="col-xs-8 col-xs-offset-2 text-left my">
        <c:choose>
            <c:when test="${action eq 'edit'}">
                <form action="<c:url
value='/${action}/task?subjectId=${task.getSubjectId()}&taskId=${task.getTask
Id()}' />" method="POST">
                    <div class="form-group">
                        <label for="taskName">Task name: </label>
                        <input type="text" class="form-control"
name="taskName" id="taskName" required value="${task.getTaskName()}"
placeholder="Enter Task Name"/>
                    </div>
                    <div class="form-group">
                        <label for="maxGrade">Task name: </label>
                        <input type="text" class="form-control"
name="maxGrade" id="maxGrade" required value="${task.getMaxGrade()}"
placeholder="Enter Max Grade">
                    </div>
                    <div class="form-group">
                        <label for="taskDescription">Task description:
</label>
                        <textarea name="taskDescription"
                            id="taskDescription"
                            class="form-control"
                            cols="30"
                            rows="5"
                            maxlength="4000"
                            placeholder="Enter Task Description"
required>${task.getTaskDescription()}</textarea>
```

```
                    </div>
            </c:when>
            <c:otherwise>
                <form action="<c:url
value='/${action}/task?subjectId=${subject.getSubjectId()}' />"
method="POST">
                    <div class="form-group">
                        <label for="taskNameA">Task name: </label>
                        <input type="text" class="form-control"
name="taskName" id="taskNameA" required value="" placeholder="Enter Task
Name">
                    </div>
                    <div class="form-group">
                        <label for="maxGradeA">Task name: </label>
                        <input type="text" class="form-control"
name="maxGrade" id="maxGradeA" required value="" placeholder="Enter Max
Grade">
                    </div>
                    <div class="form-group">
                        <label for="taskDescriptionA">Task description:
</label>
                        <textarea name="taskDescription"
                        id="taskDescriptionA"
                        class="form-control"
                        cols="30"
                        rows="5"
                        maxlength="4000"
                        placeholder="Enter Task Description"
required></textarea>
                    </div>
            </c:otherwise>
        </c:choose>
            <button type="submit" class="btn btn-default">${action}</button>
                <c:choose>
                    <c:when test="${action eq 'edit'}">
                        <a href="<c:url
value='/show/subject?subjectId=${task.getSubjectId()}'/>" class="btn btn-
default">Cancel</a>
                    </c:when>
                    <c:otherwise>
                        <a href="<c:url
value="/show/subject?subjectId=${subject.getSubjectId()}"/>" class="btn btn-
default">Cancel</a>
                    </c:otherwise>
                </c:choose>
            </form>
        </div>
    </div>
</div>
<jsp:include page="footer.jsp" />
</body>
</html>
```

Сторінка “add-edit-teacher-page.jsp”

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>${action}</title>
</head>

<body>
<c:url value='/show/teacher-all' var="allTeacherURL"/>
<jsp:include page="header.jsp" />
<div class="container-fluid text-center">
    <div class="row content my-c">
        <div class="col-xs-8 col-xs-offset-2 text-left my">
            <c:choose>
                <c:when test="${action eq 'edit'}">
                    <form action="<c:url
value='/${action}/teacher/${teacher.getTeacherId()}' />" method="POST">
                        <div class="form-group">
                            <label for="loginName">Email: </label>
                            <input type="text" name="loginName"
id="loginName" class="form-control" required value="${teacher.getEmail()}"
placeholder="Enter Email"/>
                        </div>
                        <div class="form-group">
                            <label for="firstName">First name: </label>
                            <input type="text" name="firstName"
id="firstName" class="form-control" required
value="${teacher.getFirstName()}" placeholder="Enter First Name"/>
                        </div>
                        <div class="form-group">
                            <label for="lastName">Last name: </label>
                            <input type="text" name="lastName" id="lastName"
class="form-control" required value="${teacher.getLastName()}"
placeholder="Enter Last Name"/>
                        </div>
                </c:when>
                <c:otherwise>
                        <form action="<c:url value='/${action}/teacher' />"
method="POST">
                            <div class="form-group">
                                <label for="loginNameA">Email: </label>
                                <input type="text" name="loginName"
id="loginNameA" class="form-control" required value="" placeholder="Enter
Email"/>
                            </div>
                            <div class="form-group">
                                <label for="firstNameA">First name: </label>
                                <input type="text" name="firstName"
id="firstNameA" class="form-control" required value="" placeholder="Enter
First Name"/>
                            </div>
                            <div class="form-group">
                                <label for="lastNameA">Last name: </label>
                                <input type="text" name="lastName"
id="lastNameA" class="form-control" required value="" placeholder="Enter Last
Name"/>
```

```
                            </div>
                            <div class="form-group">
                                <label for="password">Password: </label>
                                <input type="text" name="password"
id="password" class="form-control" required value="" placeholder="Enter
Password"/>
                            </div>
                </c:otherwise>
            </c:choose>
            <button type="submit" class="btn btn-default">${action}</button>
                <c:choose>
                    <c:when test="${action eq 'edit'}">
                        <a href="<c:url
value='/show/teacher?teacherId=${teacher.getTeacherId()}'/>" class="btn btn-
default">Cancel</a>
                    </c:when>
                    <c:otherwise>
                        <a href="<c:out value="${allTeacherURL}"/>"
class="btn btn-default">Cancel</a>
                    </c:otherwise>
                </c:choose>
            </form>
        </div>
    </div>
</div>
<jsp:include page="footer.jsp" />
</body>
</html>
```

Сторінка “footer.jsp”

```
<footer class="container-fluid text-center">
    <p>All Rights Reserved, Copyright 2019 - 2022, GradeBook</p>
</footer>
```

Сторінка “header.jsp”

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<script>
    function showResult(str) {
        if (str.length==0) {
            document.getElementById("searchResultId").innerHTML="";
            document.getElementById("searchResultId").style.display="none";
            return;
        }

        var xmlhttp=new XMLHttpRequest();
        xmlhttp.onreadystatechange=function() {
            if (this.readyState==4 && this.status==200 &&
this.responseText.length != 0) {

document.getElementById("searchResultId").innerHTML=this.responseText;
```

```
document.getElementById("searchResultId").style.display="block";
            }
        }

        var e = document.getElementById("searchTypeId");
        var searchType = e.value;
        var url = "";
        if(searchType == "byStudent") {
            url = "/Lab3GradeBook/search/by-student-full-
name?studentFullName=";
        } else if(searchType == "byGroup") {
            url = "/Lab3GradeBook/search/by-group-name?groupName=";
        } else {
            url = "/Lab3GradeBook/search/by-subject-name?subjectName=";
        }
        xmlhttp.open("GET",url+str,true);
        xmlhttp.send();
    }

</script>

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></scri
pt>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></s
cript>
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.j
s"></script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></s
cript>
    <style>
        .alert {border-radius: 0 !important;}
        .row.content.my-c {background: #f1f1f1;}
        body {background-color: #f1f1f1}
        td.desc-my {max-width: 300px;}
        td.grade-my {max-width: 150px;}
        th.alert.alert-info {width: 200px;}
        .jumbotron {margin-bottom: 0;}

        footer {
            background-color: #555;
            color: white;
            padding: 15px;
        }

        .navbar {
            margin-bottom: 0;
```

```
            border-radius: 0;
        }

        .col-xs-8.col-xs-offset-2.text-left.my {
            background: #fff;
            min-height: calc(100vh - 112px);
        }

        .row.content {
            height: available;
            min-height: 640px;
            background-color: #ffffff;
        }

        div.well {
            padding: 9px 19px 9px;
            margin: 5px 0 5px;
        }

        .table-bordered.tb-my {
            background-color: #ffffff;
            padding: 0;
            margin: 0;
            border-radius: 15px !important;
        }

        .profile {
            padding: 0;
            margin: 0 auto;
            text-align: center;
        }

        .form-group.input-group.my,
        .form-group.my,
        form.my {margin-bottom: 0;}

        .sidenav {
            padding-top: 20px;
            background-color: #f1f1f1;
            max-height: 10000px;
            min-height: 640px;
        }

        @media screen and (max-width: 767px) {
            .sidenav {
                height: auto;
                padding: 15px;
            }
            .row.content {height:auto;}
        }
    </style>
</head>

<%
    String username=(String)session.getAttribute("current_username");
    String userId=(String)session.getAttribute("current_user_id");
    String userRole=(String)session.getAttribute("current_user_role");
%>
```

```
<c:url value='/redirect/profile' var="profileURL">
    <c:param name="userId" value="<%=userId%>"/>
    <c:param name="userRole" value="<%=userRole%>"/>
</c:url>
<c:url value='/logout' var="logoutURL"/>
<c:url value='/show/student-all' var="allStudentURL"/>
<c:url value='/show/group-all' var="allGroupURL"/>
<c:url value='/show/subject-all' var="allSubjectURL"/>
<c:url value='/show/teacher-all' var="allTeacherURL"/>

<header>
    <nav class="navbar navbar-inverse">
    <div class="container-fluid">
        <div class="navbar-header">
            <a class="navbar-brand" href="">GradeBook</a>
        </div>

        <div class="collapse navbar-collapse" id="myNavbar">
            <ul class="nav navbar-nav">
                <li><a href="<c:out
value="${allStudentURL}"/>">Students</a></li>
                <li><a href="<c:out value="${allGroupURL}"/>">Groups</a></li>
                <li><a href="<c:out
value="${allSubjectURL}"/>">Subjects</a></li>
                <li><a href="<c:out
value="${allTeacherURL}"/>">Teachers</a></li>
            </ul>
            <div class="nav navbar-nav">
                <form name="search_form" class="navbar-form " role="search">
                    <div class="form-group">
                        <select class="form-control" name="searchType"
id="searchTypeId">
                            <option value="" disabled>Choose search
type</option>
                            <option value="byGroup">Search By Group</option>
                            <option value="byStudent"
selected="selected">Search By Student</option>
                            <option value="bySubject">Search By
Subject</option>
                        </select>
                        <div style="display: inline-block;">
                            <input type="text" class="form-control"
name="search-box" id="searchBoxId" placeholder="Search.." size="30"
onkeyup="showResult(this.value)"/>
                            <div class="drop-my dropdown" >
                                <ul class="dropdown-menu my-l"
id="searchResultId">
                                </ul>
                            </div>
                        </div>
                    </div>
                </form>
            </div>
            <ul class="nav navbar-nav navbar-right">
                <li>
                    <a href="<c:out value="${profileURL}"/>">
                    <span class="glyphicon glyphicon-user"></span>
<%=username%></a>
```

```
                </li>
                <li>
                    <a href="<c:out value="${logoutURL}"/>">
                    <span class="glyphicon glyphicon-log-out"></span>
Logout</a>
                </li>
            </ul>
        </div>
    </div>
    </nav>
</header>
```

Сторінка “login.jsp”

```
<%@ page import="org.example.dao.implementations.DAOStudentImpl" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Login page</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></scri
pt>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></s
cript>
        <style>
        footer {
            background-color: #555;
            color: white;
            padding: 15px;
        }

        .navbar {
            margin-bottom: 0;
            border-radius: 0;
        }

        .jumbotron {
            margin-bottom: 0;
        }

        .btn-block.my {
            background-color: #222222;
            color: #f1f1f1;
        }

        .text-center.my-login {
            padding: 100px 140px 0 140px;
        }

        #login-check {
```

```
            padding: 0;
        }

        .justify-content-center.my-login {
            height: 40px;
            vertical-align: middle;
            text-align: center;
            padding-bottom: 10px;
            padding-top: 10px;
        }

        .text-center.my-login-r {
            margin-top: 15px;
        }

        .row.content {height: 640px}

        .sidenav {
            padding-top: 20px;
            background-color: #f1f1f1;
            height: 100%;
        }

        @media screen and (max-width: 767px) {
            .sidenav {
                height: auto;
                padding: 15px;
            }
            .row.content {height:auto;}
        }
    </style>
</head>
<c:url value='/redirect/login' var="loginURL"/>
<body>
<header>
    <nav class="navbar navbar-inverse">
        <div class="container-fluid">
            <div class="navbar-header">
                <a class="navbar-brand" href="">GradeBook</a>
            </div>
            <div class="collapse navbar-collapse" id="myNavbar">
                <ul class="nav navbar-nav navbar-right">
                    <li>
                        <a href="<c:out value="${loginURL}"/>">
                            <span class="glyphicon glyphicon-log-in"></span>
Login</a>
                    </li>
                </ul>
            </div>
        </div>
    </nav>
</header>

<div class="container-fluid text-center">
    <div class="row content">
        <div class="col-sm-3 sidenav">
        </div>
        <div class="col-sm-6 text-center my-login">
```

```
            <h1>Sign In</h1>
            <c:if test="${not empty ExceptionMessage}">
            <div>
                <label>${ExceptionMessage}</label>
            </div>
            </c:if>
            <form name="username" action="<c:url value='/login' />"
method="POST">
                <div class="form-group text-left">
                    <label for="loginUserName">Email: </label>
                    <input type="text" value="" class="form-control"
name="loginUserName" required placeholder="Enter email" id="loginUserName"/>
                </div>
                <div class="form-group  text-left">
                    <label for="loginPassword">Password: </label>
                    <input type="password" value="" class="form-control"
name="loginPassword" required placeholder="Enter password"
id="loginPassword"/>
                </div>
                <div class="row mb-4">
                    <div class="col-md-6 d-flex justify-content-center">
                        <div class="checkbox">
                            <input   type="checkbox" value=""
id="loginCheck"/>
                            <label id="login-check" for="loginCheck">Remember
me</label>
                        </div>
                    </div>
                    <div class="col-md-6 d-flex justify-content-center my-
login">
                        <a href="#">Forgot password?</a>
                    </div>
                </div>
                <button type="submit" class="btn btn-block my text-
left">Login</button>
                <div class="text-center my-login-r">
                    <p>Not a member? <a href="#!">Register</a></p>
                </div>
            </form>

        </div>
        <div class="col-sm-3 sidenav">
        </div>
    </div>
</div>
<jsp:include page="footer.jsp" />
</body>
</html>
```

Сторінка “profile.jsp”

```
<%@ page import="org.example.entities.Student" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
```

```
    <title>Profile</title>
</head>
<body>
<c:url value='/edit/user-password' var="changePasswordURL" />
<%String userRole=(String)session.getAttribute("current_user_role");%>
<c:set var = "userRole" value = "<%=userRole%>"/>


    <jsp:include page="header.jsp" />
    <div class="container-fluid text-center">
        <div class="row content my-c" >
            <div class="col-xs-8 col-xs-offset-2 text-left my">
                <div class="well">
                    <table class="table table-bordered tb-my " >
                        <tbody>
                            <tr><th class="alert alert-info">Full
name</th><td class="bg-success">${user.getFullName()}</td></tr>
                            <tr><th class="alert alert-info">Email</th><td
class="bg-success">${user.getEmail()}</td></tr>
                            <tr><th class="alert alert-info">First
name</th><td class="bg-success">${user.getFirstName()}</td></tr>
                            <tr><th class="alert alert-info">Last
name</th><td class="bg-success">${user.getLastName()}</td></tr>
                            <c:choose>
                                <c:when test="${user.getRole() eq
'student'}">
                                    <tr><th class="alert alert-
info">Role</th><td class="bg-success">Student</td></tr>
                                </c:when>
                                <c:otherwise>
                                    <tr><th class="alert alert-
info">Role</th><td class="bg-success">Teacher</td></tr>
                                </c:otherwise>
                            </c:choose>
                            <c:if test="${user.getRole() eq 'student'}">
                                <c:choose>
                                    <c:when test="${empty
user.getHeadman()}">
                                        <tr><th class="alert alert-
info">Headman</th><td class="bg-success">(not yet)</td></tr>
                                    </c:when>
                                    <c:otherwise>
                                        <c:url value='/redirect/profile'
var="headmanURL">
                                            <c:param name="userId"
value="${studentInfoSet.getHeadman().getStudentId()}"/>
                                            <c:param name="userRole"
value="${studentInfoSet.getHeadman().getRole()}"/>
                                        </c:url>
                                        <tr><th class="alert alert-
info">Headman</th><td class="bg-success"><a href="<c:out
value="${headmanURL}"/>">${studentInfoSet.getHeadman().getFullName()}</a></td
></tr>
                                    </c:otherwise>
                                </c:choose>
                                <c:choose>
                                    <c:when test="${empty
user.getGroupId()}">
```

```
                                        <tr><th class="alert alert-
info">Group</th><td class="bg-success">(not yet)</td></tr>
                                    </c:when>
                                    <c:otherwise>
                                        <c:url value='/show/group'
var="studentGroupURL">
                                            <c:param name="groupId"
value="${studentInfoSet.getGroup().getGroupId()}"/>
                                        </c:url>
                                        <tr><th class="alert alert-
info">Group</th><td class="bg-success"><a href="<c:out
value="${studentGroupURL}"/>">${studentInfoSet.getGroup().getGroupName()}</a>
</td></tr>
                                    </c:otherwise>
                                </c:choose>
                            </c:if>
                        </tbody>
                    </table>
                </div>

            <c:choose>
                <c:when test="${user.getRole() eq 'student'}">
                    <c:choose>
                        <c:when test="${studentSubjectMap.size() eq 0}">
                            <label>Subjects: (not yet)</label><br>
                        </c:when>
                        <c:otherwise>
                            <table class="table table-striped">
                                <thead>
                                    <tr>
                                        <th>Subject Name</th>
                                        <th>Total Grade</th>
                                        <th>Grade(%)</th>
                                        <th>Pass(%)</th>
                                        <th>More...</th>
                                        <th>Grade details...</th>
                                        <c:if test="${userRole eq
'teacher'}">
                                            <th>Remove from subject</th>
                                        </c:if>
                                    </tr>
                                </thead>
                                <tbody>
                                    <c:forEach var="e"
items="${studentSubjectMap.entrySet()}">
                                        <tr>

<td>${e.getKey().getSubjectName()}</td>

<td>${e.getValue().getTotalGrade()}</td>
                                            <td>
                                                <script>

document.write(Math.round(parseInt(${e.getValue().getTotalGrade()/e.getKey().
getMaxGrade()*100})));
                                                </script>%
                                            </td>
```

```
<td>${e.getKey().getPassProcGradeP()}%</td>
                                            <c:url value='/show/subject'
var="subjectURL">
                                                <c:param name="subjectId"
value="${e.getKey().getSubjectId()}"/>
                                            </c:url>
                                            <td><a href="<c:out
value="${subjectURL}"/>">More...</a></td>
                                            <c:url value='/show/student-task-
list' var="studentTaskListURL">
                                                <c:param name="studentId"
value="${e.getValue().getStudentId()}"/>
                                                <c:param name="subjectId"
value="${e.getKey().getSubjectId()}"/>
                                            </c:url>
                                            <td><a href="<c:out
value="${studentTaskListURL}"/>">Grade details...</a></td>
                                            <c:if test="${userRole eq
'teacher'}">
                                                <td><a href="<c:url
value='/delete/student-
subject?studentId=${user.getStudentId()}&subjectId=${e.getKey().getSubjectId(
)}' />">Remove from subject</a></td>
                                            </c:if>
                                        </tr>
                                    </c:forEach>
                                </tbody>
                            </table>
                        </c:otherwise>
                    </c:choose>
                </c:when>
                <c:otherwise>
                    <c:choose>
                        <c:when test="${teacherSubjectMap.size() eq 0}">
                            <label>Subjects: (not yet)</label><br>
                        </c:when>
                        <c:otherwise>
                            <table class="table table-striped">
                                <thead>
                                    <tr>
                                        <th>Subject Name</th>
                                        <th>Pass(%)</th>
                                        <th>More...</th>
                                        <c:if test="${userRole eq
'teacher'}">
                                            <th>Remove from subject</th>
                                        </c:if>
                                    </tr>
                                </thead>
                                <tbody>
                                    <c:forEach var="subject"
items="${teacherSubjectMap}">
                                        <tr>

<td>${subject.getSubjectName()}</td>

<td>${subject.getPassProcGradeP()}%</td>
                                            <c:url value='/show/subject'
```

```
var="subjectURL">
                                                    <c:param name="subjectId"
value="${subject.getSubjectId()}"/>
                                                </c:url>
                                                <td><a href="<c:out
value="${subjectURL}"/>">More...</a></td>
                                                <c:if test="${userRole eq
'teacher'}">
                                                    <td><a href="<c:url
value='/delete/teacher-
subject?teacherId=${user.getTeacherId()}&subjectId=${subject.getSubjectId()}'
/>">Remove from subject</a></td>
                                                </c:if>
                                            </tr>
                                        </c:forEach>
                                    </tbody>
                                </table>
                            </c:otherwise>
                        </c:choose>
                    </c:otherwise>
                </c:choose>

                <c:if test="${isCurrentProfile eq 'Yes'}">
                <div class="well">
                    <div class="profile">
                        <form class="form-inline my" action="<c:out
value="${changePasswordURL}"/>" method="post">
                            <div class="form-group my">
                                <label for="newPassword">New password: </label>
                                <input type="password" class="form-control"
name="newPassword" id="newPassword" required placeholder="New password.."/>
                            </div>
                            <div class="form-group">
                                <label for="confirmPassword">Confirm password:
</label>
                                <input type="password" class="form-control"
name="confirmPassword" id="confirmPassword" required placeholder="Confirm
password.."/>
                            </div>
                            <button type="submit" class="btn btn-default">Change
Password</button>
                        </form>
                    </div>
                </div>
                </c:if>
                    <c:if test="${isCurrentProfile eq 'Yes'}">
                        <c:choose>
                            <c:when test="${user.getRole() eq 'student'}">
                                <div class="well">
                                    <div class="profile">
                                        <a href="<c:url
value='/redirect/edit/student?studentId=${user.getStudentId()}' />"
class="btn btn-default">Change Account Information</a>
                                        <a href="<c:url
value='/delete/student?studentId=${user.getStudentId()}' />" class="btn btn-
default">Delete Account</a>
                                    </div>
                                </div>
```

```
                        </c:when>
                        <c:otherwise>
                            <div class="well">
                                <div class="profile">
                                    <a href="<c:url
value='/redirect/edit/teacher?teacherId=${user.getTeacherId()}' />"
class="btn btn-default">Change Account Information</a>
                                    <a href="<c:url
value='/delete/teacher?teacherId=${user.getTeacherId()}' />" class="btn btn-
default">Delete Account</a>
                                </div>
                            </div>
                        </c:otherwise>
                    </c:choose>
                </c:if>
            </div>
        </div>
    </div>
    <jsp:include page="footer.jsp" />
</body>
</html>
```

Сторінка “group-list.jsp”

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>Groups</title>
</head>
<body>
<%String userRole=(String)session.getAttribute("current_user_role");%>
<c:set var = "userRole" value = "<%=userRole%>"/>
<jsp:include page="header.jsp" />

<div class="container-fluid text-center">
    <div class="row content my-c">
        <div class="col-xs-8 col-xs-offset-2 text-left my">
            <div class="well">
                <c:if test="${userRole eq 'teacher'}">
                    <a href="<c:url value='/redirect/add/group' />"
class="btn btn-default">Add new group</a>
                </c:if>
            </div>
            <table class="table table-striped">
                <thead>
                    <tr>
                        <th>Group Name</th>
                        <th>Group Description</th>
                        <th>More...</th>
                    </tr>
                </thead>
                <tbody>
                    <c:forEach var="group" items="${groups}">
                    <tr>
                        <td>${group.getGroupName()}</td>
```

```
                        <td>${group.getGroupDescription()}</td>
                        <c:url value='/show/group' var="groupURL">
                            <c:param name="groupId"
value="${group.getGroupId()}"/>
                        </c:url>
                        <td><a href="<c:out
value="${groupURL}"/>">More...</a></td>
                    </tr>
                    </c:forEach>
                </tbody>
            </table>
        </div>
    </div>
</div>
<jsp:include page="footer.jsp" />
</body>
</html>
```

Сторінка “student-list.jsp”

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>Students</title>
</head>
<body>
<%String userRole=(String)session.getAttribute("current_user_role");%>
<c:set var = "userRole" value = "<%=userRole%>"/>
<jsp:include page="header.jsp" />

<div class="container-fluid text-center">
    <div class="row content my-c">
        <div class="col-xs-8 col-xs-offset-2 text-left my">
            <div class="well">
                <c:if test="${userRole eq 'teacher'}">
                    <a href="<c:url value='/redirect/add/student' />"
class="btn btn-default">Add new student</a>
                </c:if>
            </div>
            <table class="table table-striped">
                <thead>
                    <tr>
                        <th>Email</th>
                        <th>First Name</th>
                        <th>Last Name</th>
                        <th>Headman</th>
                        <th>Group</th>
                        <th>More...</th>
                    </tr>
                </thead>
                <tbody>
                    <c:forEach var="studentInfoSet"
items="${studentInfoSetList}">
                        <tr>
```

```
<td>${studentInfoSet.getStudent().getEmail()}</td>

<td>${studentInfoSet.getStudent().getFirstName()}</td>

<td>${studentInfoSet.getStudent().getLastName()}</td>
                            <c:choose>
                                <c:when test="${empty
studentInfoSet.getStudent().getHeadman()}">
                                    <td>(not yet)</td>
                                </c:when>
                                <c:otherwise>
                                    <c:url value='/redirect/profile'
var="studentHeadmanURL">
                                        <c:param name="userId"
value="${studentInfoSet.getHeadman().getStudentId()}"/>
                                        <c:param name="userRole"
value="${studentInfoSet.getHeadman().getRole()}"/>
                                    </c:url>
                                    <td><a href="<c:out
value="${studentHeadmanURL}"/>">${studentInfoSet.getHeadman().getFullName()}<
/a></td>
                                </c:otherwise>
                            </c:choose>

                            <c:choose>
                                <c:when test="${empty
studentInfoSet.getStudent().getGroupId()}">
                                    <td>(not yet)</td>
                                </c:when>
                                <c:otherwise>
                                    <c:url value='/show/group'
var="studentGroupURL">
                                        <c:param name="groupId"
value="${studentInfoSet.getGroup().getGroupId()}"/>
                                    </c:url>
                                    <td><a href="<c:out
value="${studentGroupURL}"/>">${studentInfoSet.getGroup().getGroupName()}</a>
</td>
                                </c:otherwise>
                            </c:choose>

                            <c:url value='/redirect/profile'
var="studentURL">
                                <c:param name="userId"
value="${studentInfoSet.getStudent().getStudentId()}"/>
                                <c:param name="userRole"
value="${studentInfoSet.getStudent().getRole()}"/>
                            </c:url>
                            <td><a href="<c:out
value="${studentURL}"/>">More...</a></td>
                        </tr>
                    </c:forEach>
                </tbody>
            </table>
        </div>
    </div>
</div>
<jsp:include page="footer.jsp" />
```

```
</body>
</html>
```

Сторінка “student-task-list.jsp”

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>Subjects</title>

</head>
<body>
<%
    String userRole=(String)session.getAttribute("current_user_role");
    String userId=(String)session.getAttribute("current_user_id");
%>
<c:set var = "userRole" value = "<%=userRole%>"/>
<c:set var = "userId" value = "<%=userId%>"/>
<jsp:include page="header.jsp" />

<div class="container-fluid text-center">
    <div class="row content my-c">
        <div class="col-xs-8 col-xs-offset-2 text-left my">
            <div class="well">
                <table class="table table-bordered tb-my " >
                    <tbody>
                    <tr><th class="alert alert-info">Student</th><td
class="bg-success"><a href="<c:url
value='/show/student?studentId=${student.getStudentId()}'/>">${student.getFul
lNameWithEmail()}</a></td></tr>
                    <tr><th class="alert alert-info">Subject</th><td
class="bg-success"><a href="<c:url
value='/show/subject?subjectId=${subject.getSubjectId()}'/>">${subject.getSub
jectName()}</a></td></tr>
                    </tbody>
                </table>
            </div>

            <table class="table table-striped">
                <thead>
                    <tr>
                        <th>Task Name</th>
                        <th>Description</th>
                        <th>Max Grade</th>
                        <th>Checked by</th>
                        <th>Grade</th>
                    </tr>
                </thead>
                <tbody>
                    <c:forEach var="st" items="${studentTaskInfoMap}">
                        <tr>
                            <td>${st.getTask().getTaskName()}</td>
                            <td class="desc-my">
                                ${st.getTask().getTaskDescription()}
                            </td>
```

```
                                <td>${st.getTask().getMaxGrade()}</td>
                                <c:choose>
                                    <c:when test="${empty
st.getStudentTask().getByTeacherId()}">
                                        <td>(not yet)</td>
                                    </c:when>
                                    <c:otherwise>
                                        <td><a href="<c:url
value='/show/teacher?teacherId=${st.getStudentTask().getByTeacherId()}'/>"
>${st.getTeacher().getFullName()}</a></td>
                                    </c:otherwise>
                                </c:choose>
                                <c:choose>
                                    <c:when test="${userRole eq 'teacher'}">
                                        <td class="grade-my">
                                            <form action="<c:url
value='/edit/student-
task/${userId}?taskId=${st.getTask().getTaskId()}&studentSubjectId=${st.getSt
udentTask().getStudentSubjectId()}' />" method="POST">
                                                <div class="form-group input-
group">
                                                    <input type="text"
class="form-control" name="grade" required
value="${st.getStudentTask().getGrade()}" placeholder="Enter Grade">
                                                    <span class="input-group-
btn">
                                                        <button type="submit"
class="btn btn-default">Save</button>
                                                    </span>
                                                </div>
                                            </form>
                                        </td>
                                    </c:when>
                                    <c:otherwise>
                                        <td class="grade-
my">${st.getStudentTask().getGrade()}</td>
                                    </c:otherwise>
                                </c:choose>
                            </tr>
                        </c:forEach>
                    </tbody>
                </table>
            </div>
        </div>
    </div>
    <jsp:include page="footer.jsp" />
    </body>
    </html>
```

Сторінка “subject-list.jsp”

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>Subjects</title>
```

```
</head>
<body>
<%String userRole=(String)session.getAttribute("current_user_role");%>
<c:set var = "userRole" value = "<%=userRole%>"/>
<jsp:include page="header.jsp" />
<div class="container-fluid text-center">
    <div class="row content my-c">
        <div class="col-xs-8 col-xs-offset-2 text-left my">
            <div class="well">
                <c:if test="${userRole eq 'teacher'}">
                    <a href="<c:url value='/redirect/add/subject' />"
class="btn btn-default">Add new subject</a>
                </c:if>
            </div>
            <table class="table table-striped">
                <thead>
                    <tr>
                        <th>Subject Name</th>
                        <th>Max Grade</th>
                        <th>Pass Grade</th>
                        <th>Subject Description</th>
                        <th>More...</th>
                    </tr>
                </thead>
                <tbody>
                    <c:forEach var="subject" items="${subjects}">
                        <tr>
                            <td>${subject.getSubjectName()}</td>
                            <td>${subject.getMaxGrade()}</td>
                            <td>${subject.getPassProcGradeP()}%</td>
                            <td>${subject.getSubjectDescription()}</td>
                            <c:url value='/show/subject' var="subjectURL">
                                <c:param name="subjectId"
value="${subject.getSubjectId()}"/>
                            </c:url>
                            <td><a href="<c:out
value="${subjectURL}"/>">More...</a></td>
                        </tr>
                    </c:forEach>
                </tbody>
            </table>
        </div>
    </div>
</div>
<jsp:include page="footer.jsp" />
</body>
</html>
```

Сторінка “teacher-list.jsp”

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>Teachers</title>
</head>
```

```
<body>
    <%String userRole=(String)session.getAttribute("current_user_role");%>
    <c:set var = "userRole" value = "<%=userRole%>"/>
    <jsp:include page="header.jsp" />
    <div class="container-fluid text-center">
        <div class="row content my-c">
            <div class="col-xs-8 col-xs-offset-2 text-left my">
                <div class="well">
                    <c:if test="${userRole eq 'teacher'}">
                        <a href="<c:url value='/redirect/add/teacher' />"
class="btn btn-default">Add new teacher</a>
                    </c:if>
                </div>
                <table class="table table-striped">
                    <thead>
                        <tr>
                            <th>Email</th>
                            <th>First Name</th>
                            <th>Last Name</th>
                            <th>More...</th>
                        </tr>
                    </thead>
                    <tbody>
                        <c:forEach var="teacher" items="${teachers}">
                            <tr>
                                <td>${teacher.getEmail()}</td>
                                <td>${teacher.getFirstName()}</td>
                                <td>${teacher.getLastName()}</td>

                                <c:url value='/redirect/profile'
var="teacherURL">
                                    <c:param name="userId"
value="${teacher.getTeacherId()}"/>
                                    <c:param name="userRole"
value="${teacher.getRole()}"/>
                                </c:url>
                                <td><a href="<c:out
value="${teacherURL}"/>">More...</a></td>
                            </tr>
                        </c:forEach>
                    </tbody>
                </table>
            </div>
        </div>
    </div>
    <jsp:include page="footer.jsp" />
</body>
</html>
```

Сторінка “show-group-page.jsp”

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@page import="org.example.dao.implementations.DAOGroupImpl" %>
<%@page import="org.example.entities.Group" %>
<%@ page import="org.example.tools.custom.exceptions.WrongEntityIdException"
```

```
%>

<html>
<head>
    <title>Group Card</title>
</head>
<%String userRole=(String)session.getAttribute("current_user_role");%>
<c:set var = "userRole" value = "<%=userRole%>"/>

<body>
<jsp:include page="header.jsp" />
<div class="container-fluid text-center">
    <div class="row content my-c">
        <div class="col-xs-8 col-xs-offset-2 text-left my">


            <c:if test="${userRole eq 'teacher'}">
                <div class="well">
                    <a href="<c:url
value='/redirect/edit/group?groupId=${group.getGroupId()}' />" class="btn
btn-default">Change Group Info</a>
                    <a href="<c:url
value='/delete/group?groupId=${group.getGroupId()}' />" class="btn btn-
default">Remove Group</a>
                </div>
            </c:if>
            <div class="well">
                <table class="table table-bordered tb-my " >
                    <tbody>
                    <tr><th class="alert alert-info">Group name</th><td
class="bg-success"> ${group.getGroupName()}</td></tr>
                    <tr><th class="alert alert-info">Group
description</th><td class="bg-
success">${group.getGroupDescription()}</td></tr>
                    </tbody>
                </table>
            </div>
            <div>
                <c:if test="${userRole eq 'teacher'}">
                <div class="well">
                    <form class="my" action="<c:url value='/edit/student-
group?groupId=${group.getGroupId()}' />" method="POST">
                        <div class="form-group input-group my">
                            <select name="studentId" class="form-control">
                                <option value="" disabled
selected="selected">choose student</option>
                                <c:forEach var="as"
items="${availableStudents}">
                                    <option
value="${as.getStudentId()}">${as.getFullNameWithEmail()}</option>
                                </c:forEach>
                            </select>
                            <span class="input-group-btn">
                                <button type="submit" class="btn btn-
default">Add Student</button>
                            </span>
                        </div>
                    </form>
```

```
                </div>
                </c:if>
            </div>
            <table class="table table-striped">
                <thead>
                    <tr>
                        <th>First Name</th>
                        <th>Last Name</th>
                        <th>More...</th>
                        <c:if test="${userRole eq 'teacher'}">
                            <th>Remove from group</th>
                        </c:if>
                    </tr>
                </thead>
                <tbody>
                    <c:forEach var="student" items="${studentList}">
                        <tr>
                            <td>${student.getFirstName()}</td>
                            <td>${student.getLastName()}</td>
                            <c:url value='/redirect/profile'
var="studentURL">
                                <c:param name="userId"
value="${student.getStudentId()}"/>
                                <c:param name="userRole"
value="${student.getRole()}"/>
                            </c:url>
                            <td><a href="<c:out
value="${studentURL}"/>">More...</a></td>
                            <c:if test="${userRole eq 'teacher'}">
                                <td><a href="<c:url value='/delete/student-
group?studentId=${student.getStudentId()}&groupId=${group.getGroupId()}'
/>">Remove from group</a></td>
                            </c:if>
                        </tr>
                    </c:forEach>
                </tbody>
            </table>
        </div>
    </div>
</div>
<jsp:include page="footer.jsp" />
</body>
</html>
```

Сторінка “show-subject-page.jsp”

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@page import="org.example.dao.implementations.DAOGroupImpl" %>
<%@page import="org.example.entities.Group" %>
<%@ page import="org.example.tools.custom.exceptions.WrongEntityIdException"
%>

<html>
<head>
    <title>Subject Card</title>
```

```
</head>
<%String userRole=(String)session.getAttribute("current_user_role");%>
<c:set var = "userRole" value = "<%=userRole%>"/>

<body>
<jsp:include page="header.jsp" />
<div class="container-fluid text-center">
    <div class="row content my-c">
        <div class="col-xs-8 col-xs-offset-2 text-left my">

            <c:if test="${userRole eq 'teacher'}">
                <div class="well">
                    <a href="<c:url
value='/redirect/edit/subject?subjectId=${subject.getSubjectId()}' />"
class="btn btn-default">Change Subject Info</a>
                    <a href="<c:url
value='/delete/subject?subjectId=${subject.getSubjectId()}' />" class="btn
btn-default">Remove Subject</a>
                </div>
            </c:if>
            <div class="well">
                <table class="table table-bordered tb-my " >
                    <tbody>
                        <tr><th class="alert alert-info">Subject name</th><td
class="bg-success">${subject.getSubjectName()}</td></tr>
                        <tr><th class="alert alert-info">Max grade</th><td
class="bg-success">${subject.getMaxGrade()}</td></tr>
                        <tr><th class="alert alert-info">Subject
description</th><td class="bg-
success">${subject.getSubjectDescription()}</td></tr>
                    </tbody>
                </table>
            </div>
            <c:if test="${userRole eq 'teacher'}">
                <div class="well">
                    <form class="my" action="<c:url value='/add/teacher-
subject?subjectId=${subject.getSubjectId()}' />" method="POST">
                        <div class="form-group input-group my">
                            <select name="teacherId" class="form-control">
                                <option value="" disabled
selected="selected">choose teacher</option>
                                <c:forEach var="at"
items="${availableTeachers}">
                                    <option
value="${at.getTeacherId()}">${at.getFullNameWithEmail()}</option>
                                </c:forEach>
                            </select>
                            <span class="input-group-btn">
                                    <button type="submit" class="btn btn-
default">Add Teacher</button>
                                </span>
                        </div>
                    </form>
                </div>
            </c:if>
            <table class="table table-striped">
                <thead>
                    <tr>
```

```
                        <th>Teacher First Name</th>
                        <th>Teacher Last Name</th>
                        <th>More...</th>
                        <c:if test="${userRole eq 'teacher'}">
                            <th>Remove from subject</th>
                        </c:if>
                    </tr>
                </thead>
                <tbody>
                    <c:forEach var="teacher" items="${teacherList}">
                        <tr>
                            <td>${teacher.getFirstName()}</td>
                            <td>${teacher.getLastName()}</td>
                            <c:url value='/redirect/profile'
var="teacherURL">
                                <c:param name="userId"
value="${teacher.getTeacherId()}"/>
                                <c:param name="userRole"
value="${teacher.getRole()}"/>
                            </c:url>
                            <td><a href="<c:out
value="${teacherURL}"/>">More...</a></td>
                            <c:if test="${userRole eq 'teacher'}">
                                <td><a href="<c:url value='/delete/teacher-
subject?teacherId=${teacher.getTeacherId()}&subjectId=${subject.getSubjectId(
)}' />">Remove from subject</a></td>
                            </c:if>

                        </tr>
                    </c:forEach>
                </tbody>
            </table>
            <c:if test="${userRole eq 'teacher'}">
                <div class="well">
                    <form class="my" action="<c:url value='/add/student-
subject?subjectId=${subject.getSubjectId()}' />" method="POST">
                        <div class="form-group input-group my">
                            <select name="studentId" class="form-control">
                                <option value="" disabled
selected="selected">choose student</option>
                                <c:forEach var="as"
items="${availableStudents}">
                                    <option
value="${as.getStudentId()}">${as.getFullNameWithEmail()}</option>
                                </c:forEach>
                            </select>
                            <span class="input-group-btn">
                                <button type="submit" class="btn btn-
default">Add Student</button>
                            </span>
                        </div>
                    </form>
                </div>
            </c:if>
            <table class="table table-striped">
                <thead>
                    <tr>
                        <th>Student First Name</th>
```

```
                        <th>Student Last Name</th>
                        <th>Total Grade</th>
                        <th>Grade(%)</th>
                        <th>Pass(%)</th>
                        <th>More...</th>
                        <th>Grade details...</th>
                        <c:if test="${userRole eq 'teacher'}">
                            <th>Remove from subject</th>
                        </c:if>
                    </tr>
                </thead>
                <tbody>
                    <c:forEach var="e"
items="${studentSubjectMap.entrySet()}">
                        <tr>
                            <td>${e.getKey().getFirstName()}</td>
                            <td>${e.getKey().getLastName()}</td>
                            <td>${e.getValue().getTotalGrade()}</td>
                            <td>
                                <script>

document.write(Math.round(parseInt(${e.getValue().getTotalGrade()/subject.get
MaxGrade()*100})));
                                </script>%
                            </td>
                            <td>${subject.getPassProcGradeP()}%</td>
                            <c:url value='/redirect/profile'
var="studentURL">
                                <c:param name="userId"
value="${e.getKey().getStudentId()}"/>
                                <c:param name="userRole"
value="${e.getKey().getRole()}"/>
                            </c:url>
                            <td><a href="<c:out
value="${studentURL}"/>">More...</a></td>
                            <c:url value='/show/student-task-list'
var="studentTaskListURL">
                                <c:param name="studentId"
value="${e.getKey().getStudentId()}"/>
                                <c:param name="subjectId"
value="${subject.getSubjectId()}"/>
                            </c:url>
                            <td><a href="<c:out
value="${studentTaskListURL}"/>">Grade details...</a></td>
                            <c:if test="${userRole eq 'teacher'}">
                                <td><a href="<c:url value='/delete/student-
subject?studentId=${student.getStudentId()}&subjectId=${subject.getSubjectId(
)}' />">Remove from subject</a></td>
                            </c:if>
                        </tr>
                    </c:forEach>
                </tbody>
            </table>
            <div class="well">
                <c:if test="${userRole eq 'teacher'}">
                    <a href="<c:url
value='/redirect/add/task?subjectId=${subject.getSubjectId()}' />" class="btn
btn-default">Add Task</a>
```

```
                </c:if>
            </div>
            <table class="table table-striped">
                <thead>
                    <tr>
                        <th>Task Name</th>
                        <th>Description</th>
                        <th>Max Grade</th>
                        <c:if test="${userRole eq 'teacher'}">
                            <th>Change</th>
                            <th>Remove</th>
                        </c:if>
                    </tr>
                </thead>
                <tbody>
                    <c:forEach var="task" items="${taskList}">
                        <tr>
                            <td>${task.getTaskName()}</td>
                            <td class="desc-my">
                                ${task.getTaskDescription()}
                            </td>
                            <td>${task.getMaxGrade()}</td>
                            <c:if test="${userRole eq 'teacher'}">
                                <td><a href="<c:url
value='/redirect/edit/task?taskId=${task.getTaskId()}' />">Change</a></td>
                                <td><a href="<c:url
value='/delete/task?taskId=${task.getTaskId()}&subjectId=${task.getSubjectId(
)}' />">Remove</a></td>
                            </c:if>
                        </tr>
                    </c:forEach>
                </tbody>
            </table>
        </div>
    </div>
</div>
<jsp:include page="footer.jsp" />
</body>
</html>
```

ДОДАТОК 3. КОД ПРОГРАМИ

У цьому додатку представлений лістинг коду всіх класів нашого додатку, котрий відповідає за реалізацію всього функціоналу, що надається користувачу. Цей код представлений для кожного класу у окремому розділі.

Клас “AbstractController”

```
package org.example.controllers;

import org.example.entities.Group;
import org.example.entities.Student;
import org.example.entities.Teacher;
import org.example.entities.User;
import org.example.tools.strings.SessionAttributeName;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import java.util.List;

public abstract class AbstractController {
    protected void setBaseSessionVariables(HttpServletRequest request, User
user) {
        HttpSession session = request.getSession(false);
        session.setAttribute(

SessionAttributeName.CURRENT_USER_ID.getSessionAttributeName(),
                user.getUserId()
        );
        session.setAttribute(

SessionAttributeName.CURRENT_USERNAME.getSessionAttributeName(),
                user.getFullName()
        );
        session.setAttribute(

SessionAttributeName.CURRENT_USER_ROLE.getSessionAttributeName(),
                user.getRole()
        );
    }

    protected void removeStudentById(List<Student> studentList, final String
studentId) {
        studentList.removeIf(s -> (s.getStudentId().equals(studentId)));
    }

    protected void removeStudentByList(List<Student> fromStudentList, final
List<Student> removeStudentList) {
        for (Student s: removeStudentList) {
            removeStudentById(fromStudentList, s.getStudentId());
        }
    }

    protected void removeGroupById(List<Group> groupList, final String
```

```
groupId) {
        groupList.removeIf(g -> (g.getGroupId().equals(groupId)));
    }

    protected void removeGroupByList(List<Group> fromGroupList, final
List<Group> removeGroupList) {
        for (Group g: removeGroupList) {
            removeGroupById(fromGroupList, g.getGroupId());
        }
    }

    protected void removeTeacherById(List<Teacher> teacherList, final String
teacherId) {
        teacherList.removeIf(g -> (g.getTeacherId().equals(teacherId)));
    }

    protected void removeTeacherByList(List<Teacher> fromTeacherList, final
List<Teacher> removeTeacherList) {
        for (Teacher t: removeTeacherList) {
            removeTeacherById(fromTeacherList, t.getTeacherId());
        }
    }
}
```

Клас “AddController”

```
package org.example.controllers;

import org.example.dao.implementations.*;
import org.example.entities.*;
import org.example.tools.custom.exceptions.WrongEntityIdException;
import org.example.tools.strings.Strings;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.ModelAndView;

import java.sql.SQLException;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import static org.example.tools.strings.PageName.*;

@Controller
@RequestMapping(value = "/add")
public class AddController {

    private final DAOStudentImpl daoStudent;
    private final DAOTeacherImpl daoTeacher;
    private final DAOGroupImpl daoGroup;
    private final DAOSubjectImpl daoSubject;
    private final DAOTeacherSubjectImpl daoTeacherSubject;
    private final DAOStudentSubjectImpl daoStudentSubject;
```

```
    private final DAOStudentTaskImpl daoStudentTask;
    private final DAOTaskImpl daoTask;

    @Autowired
    public AddController(DAOStudentImpl daoStudent, DAOTeacherImpl
daoTeacher, DAOGroupImpl daoGroup, DAOSubjectImpl daoSubject,
DAOTeacherSubjectImpl daoTeacherSubject, DAOStudentSubjectImpl
daoStudentSubject, DAOStudentTaskImpl daoStudentTask, DAOTaskImpl daoTask) {
        this.daoStudent = daoStudent;
        this.daoTeacher = daoTeacher;
        this.daoGroup = daoGroup;
        this.daoSubject = daoSubject;
        this.daoTeacherSubject = daoTeacherSubject;
        this.daoStudentSubject = daoStudentSubject;
        this.daoStudentTask = daoStudentTask;
        this.daoTask = daoTask;
    }

    @RequestMapping(value = "/group")
    @GetMapping
    public ModelAndView addNewGroup(@RequestParam("groupName") String
groupName,
                                    @RequestParam("groupDescription") String
groupDescription) {

        ModelAndView modelAndView = new ModelAndView();
        try {
            daoGroup.addGroup(new Group(null, groupName, groupDescription));
            modelAndView.setViewName("redirect:/show/group-all");
        } catch (SQLException throwables) {
            throwables.printStackTrace();
            modelAndView.setViewName(ERROR_PAGE.getPageName());
        }

        return modelAndView;
    }

    @RequestMapping(value = "/student")
    @GetMapping
    public ModelAndView addNewStudent(@RequestParam("firstName") String
firstName,
                                      @RequestParam("loginName") String
email,
                                      @RequestParam("lastName") String
lastName,
                                      @RequestParam("headman") String
headman,
                                      @RequestParam("password") String
password,
                                      @RequestParam("groupId") String
groupId) {

        ModelAndView modelAndView = new ModelAndView();
        Student student = new Student(
                null,
                email,
                firstName,
                lastName,
```

```
                password,
                Strings.NOT_YET.getStrings().equals(headman) ? null : headman
,
                Strings.NOT_YET.getStrings().equals(groupId) ? null : groupId
        );

        try {
            daoStudent.addStudent(student);
            modelAndView.setViewName("redirect:/show/student-all");
        } catch (SQLException throwables) {
            throwables.printStackTrace();
            modelAndView.setViewName(ERROR_PAGE.getPageName());
        }

        return modelAndView;
    }

    @RequestMapping(value = "/teacher")
    @GetMapping
    public ModelAndView addNewTeacher(@RequestParam("firstName") String
firstName,
                                      @RequestParam("loginName") String email,
                                      @RequestParam("lastName") String
lastName,
                                      @RequestParam("password") String
password) {

        ModelAndView modelAndView = new ModelAndView();
        Teacher teacher = new Teacher(
                null,
                email,
                firstName,
                lastName,
                password
        );

        try {
            daoTeacher.addTeacher(teacher);
            modelAndView.setViewName("redirect:/show/teacher-all");
        } catch (SQLException throwables) {
            throwables.printStackTrace();
            modelAndView.setViewName(ERROR_PAGE.getPageName());
        }

        return modelAndView;
    }

    @RequestMapping(value = "/subject")
    @GetMapping
    public ModelAndView addNewSubject(@RequestParam("subjectName") String
subjectName,
                                      @RequestParam("maxGrade") String
maxGrade,
                                      @RequestParam("passProcGrade") String
passProcGrade,
                                     @RequestParam("subjectDescription")
String subjectDescription) {
```

```
        ModelAndView modelAndView = new ModelAndView();
        Subject subject = new Subject(
                null,
                subjectName,
                Integer.parseInt(maxGrade),
                Integer.parseInt(passProcGrade),
                subjectDescription
        );

        try {
            daoSubject.addSubject(subject);
            modelAndView.setViewName("redirect:/show/subject-all");
        } catch (SQLException throwables) {
            throwables.printStackTrace();
            modelAndView.setViewName(ERROR_PAGE.getPageName());
        }

        return modelAndView;
    }

    @RequestMapping(value = "/teacher-subject")
    @GetMapping
    public ModelAndView addNewTeacherSubject(@RequestParam("teacherId")
String teacherId,
                                             @RequestParam("subjectId") String
subjectId) {

        ModelAndView modelAndView = new ModelAndView();
        TeacherSubject teacherSubject = new TeacherSubject(
          subjectId,
          teacherId
        );

        try {
            daoTeacherSubject.addTeacherSubject(teacherSubject);
            modelAndView.setViewName("redirect:/show/subject?subjectId=" +
subjectId);
        } catch (SQLException throwables) {
            throwables.printStackTrace();
            modelAndView.setViewName(ERROR_PAGE.getPageName());
        }

        return modelAndView;
    }

    @RequestMapping(value = "/student-subject")
    @GetMapping
    public ModelAndView addNewStudentSubject(@RequestParam("studentId")
String studentId,
                                             @RequestParam("subjectId") String
subjectId) {

        ModelAndView modelAndView = new ModelAndView();
        StudentSubject studentSubject = new StudentSubject(
                null,
                studentId,
                subjectId,
                0
```

```
        );

        try {
            daoStudentSubject.addStudentSubject(studentSubject);

            String lastStudentSubjectId =
daoStudentSubject.getLastStudentSubjectId();
            StudentSubject lastStudentSubject =
daoStudentSubject.getStudentSubjectById(lastStudentSubjectId);
            List<Task> taskList =
daoTask.getTaskListBySubjectId(lastStudentSubject.getSubjectId());
            for (Task t: taskList) {
                daoStudentTask.addStudentTask(new StudentTask(
                        t.getTaskId(),
                        t.getSubjectId(),
                        lastStudentSubjectId,
                        0,
                        null
                ));
            }

            modelAndView.setViewName("redirect:/show/subject?subjectId=" +
subjectId);
        } catch (SQLException | WrongEntityIdException throwables) {
            throwables.printStackTrace();
            modelAndView.setViewName(ERROR_PAGE.getPageName());
        }

        return modelAndView;
    }

    @RequestMapping(value = "/task")
    @GetMapping
    public ModelAndView addNewTask(@RequestParam("subjectId") String
subjectId,
                                   @RequestParam("taskName") String
taskName,
                                   @RequestParam("maxGrade") String
maxGrade,
                                   @RequestParam("taskDescription") String
taskDescription) {

        ModelAndView modelAndView = new ModelAndView();
        Task task = new Task(
                null,
                subjectId,
                taskName,
                Integer.parseInt(maxGrade),
                taskDescription
        );

        try {
            daoTask.addTask(task);
            daoSubject.actualizeMaxGrade(subjectId);
            String lastTaskId = daoTask.getLastTaskId();
            Task lastTask = daoTask.getTaskById(lastTaskId);
            List<StudentSubject> studentSubjectList =
daoStudentSubject.getStudentSubjectsBySubjectId(lastTask.getSubjectId());
```

```
            for (StudentSubject s: studentSubjectList) {
                daoStudentTask.addStudentTask(new StudentTask(
                        lastTaskId,
                        s.getSubjectId(),
                        s.getStudentSubjectId(),
                        0,
                        null
                ));
            }

            modelAndView.setViewName("redirect:/show/subject?subjectId=" +
subjectId);
        } catch (SQLException | WrongEntityIdException throwables) {
            throwables.printStackTrace();
            modelAndView.setViewName(ERROR_PAGE.getPageName());
        }

        return modelAndView;
    }

    @RequestMapping(value = "/student-task")
    @GetMapping
    public ModelAndView addNewStudentTask(@RequestParam("studentSubjectId")
String studentSubjectId,
                                          @RequestParam("taskId") String taskId,
                                          @RequestParam("subjectId") String
subjectId,
                                          @RequestParam("grade") String grade) {

        ModelAndView modelAndView = new ModelAndView();
        StudentTask studentTask = new StudentTask(
                taskId,
                subjectId,
                studentSubjectId,
                Integer.parseInt(grade),
                null
        );

        try {
            daoStudentTask.addStudentTask(studentTask);
            modelAndView.setViewName("redirect:/show/subject?subjectId=" +
subjectId);
        } catch (SQLException throwables) {
            throwables.printStackTrace();
            modelAndView.setViewName(ERROR_PAGE.getPageName());
        }

        return modelAndView;
    }
}
```

Клас “DeleteController”

```
package org.example.controllers;

import org.example.dao.implementations.*;
```

```
import org.example.entities.*;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.ModelAndView;

import java.util.List;

import static org.example.tools.strings.PageName.ERROR_PAGE;

@Controller
@RequestMapping(value = "/delete")
public class DeleteController {

    private final DAOStudentImpl daoStudent;
    private final DAOTeacherImpl daoTeacher;
    private final DAOGroupImpl daoGroup;
    private final DAOSubjectImpl daoSubject;
    private final DAOTeacherSubjectImpl daoTeacherSubject;
    private final DAOStudentSubjectImpl daoStudentSubject;
    private final DAOStudentTaskImpl daoStudentTask;
    private final DAOTaskImpl daoTask;

    @Autowired
    public DeleteController(DAOStudentImpl daoStudent, DAOTeacherImpl
daoTeacher, DAOGroupImpl daoGroup, DAOSubjectImpl daoSubject,
DAOTeacherSubjectImpl daoTeacherSubject, DAOStudentSubjectImpl
daoStudentSubject, DAOStudentTaskImpl daoStudentTask, DAOTaskImpl daoTask) {
        this.daoStudent = daoStudent;
        this.daoTeacher = daoTeacher;
        this.daoGroup = daoGroup;
        this.daoSubject = daoSubject;
        this.daoTeacherSubject = daoTeacherSubject;
        this.daoStudentSubject = daoStudentSubject;
        this.daoStudentTask = daoStudentTask;
        this.daoTask = daoTask;
    }

    @RequestMapping(value = "/group")
    @GetMapping
    public ModelAndView deleteByGroupId(@RequestParam("groupId") String
groupId) {

        ModelAndView modelAndView = new ModelAndView();
        try {

            List<Student> studentList =
daoStudent.getStudentsByGroupId(groupId);
            for (Student s: studentList) {
                daoStudent.deleteGroupOfStudent(s.getStudentId());
            }
            daoGroup.deleteGroup(groupId);
            modelAndView.setViewName("redirect:/show/group-all");
        } catch (Exception throwables) {
            throwables.printStackTrace();
            modelAndView.setViewName(ERROR_PAGE.getPageName());
```

```
        }

        return modelAndView;
    }

    @RequestMapping(value = "/student")
    @GetMapping
    public ModelAndView deleteByStudentId(@RequestParam("studentId") String
studentId) {

        daoStudentTask.deleteStudentTasksByStudentId(studentId);
        daoStudentSubject.deleteStudentSubjectsByStudentId(studentId);
        daoStudent.deleteStudent(studentId);
        return new ModelAndView("redirect:/show/student-all");
    }

    @RequestMapping(value = "/teacher")
    @GetMapping
    public ModelAndView deleteByTeacherId(@RequestParam("teacherId") String
teacherId) {

        ModelAndView modelAndView = new ModelAndView();
        try {
            daoStudentTask.deleteByTeacherId(teacherId);
            daoTeacherSubject.deleteTeacherSubjectsByTeacherId(teacherId);
            daoTeacher.deleteTeacher(teacherId);
            modelAndView.setViewName("redirect:/show/teacher-all");
        } catch (Exception throwables) {
            throwables.printStackTrace();
            modelAndView.setViewName(ERROR_PAGE.getPageName());
        }

        return modelAndView;
    }

    @RequestMapping(value = "/subject")
    @GetMapping
    public ModelAndView deleteBySubjectId(@RequestParam("subjectId") String
subjectId) {

        ModelAndView modelAndView = new ModelAndView();
        try {
            daoStudentTask.deleteStudentTasksBySubjectId(subjectId);
            daoTask.deleteTasksBySubjectId(subjectId);
            daoTeacherSubject.deleteTeacherSubjectsBySubjectId(subjectId);
            daoStudentSubject.deleteStudentSubjectsBySubjectId(subjectId);
            daoSubject.deleteSubject(subjectId);
            modelAndView.setViewName("redirect:/show/subject-all");
        } catch (Exception throwables) {
            throwables.printStackTrace();
            modelAndView.setViewName(ERROR_PAGE.getPageName());
        }

        return modelAndView;
    }

    @RequestMapping(value = "/teacher-subject")
    @GetMapping
```

```
    public ModelAndView deleteTeacherSubjectById(@RequestParam("teacherId")
String teacherId,
                                                 @RequestParam("subjectId")
String subjectId) {

        ModelAndView modelAndView = new ModelAndView();
        TeacherSubject teacherSubject = new TeacherSubject(
                subjectId,
                teacherId
        );

        try {
            daoTeacherSubject.deleteTeacherSubject(teacherSubject);
            modelAndView.setViewName("redirect:/show/subject?subjectId=" +
subjectId);
        } catch (Exception throwables) {
            throwables.printStackTrace();
            modelAndView.setViewName(ERROR_PAGE.getPageName());
        }

        return modelAndView;
    }

    @RequestMapping(value = "/student-subject")
    @GetMapping
    public ModelAndView deleteStudentSubjectById(@RequestParam("studentId")
String studentId,
                                                 @RequestParam("subjectId")
String subjectId) {

        ModelAndView modelAndView = new ModelAndView();
        try {
            StudentSubject studentSubject =
daoStudentSubject.getStudentSubjectBySubjectIdAndStudentId(subjectId,
studentId);

daoStudentTask.deleteStudentTasksByStudentSubjectId(studentSubject.getStudent
SubjectId());
            daoStudentSubject.deleteStudentSubject(studentId, subjectId);

            modelAndView.setViewName("redirect:/show/subject?subjectId=" +
subjectId);
        } catch (Exception throwables) {
            throwables.printStackTrace();
            modelAndView.setViewName(ERROR_PAGE.getPageName());
        }

        return modelAndView;
    }

    @RequestMapping(value = "/student-group")
    @GetMapping
    public ModelAndView deleteStudentGroup(@RequestParam("studentId") String
studentId,
                                           @RequestParam("groupId")
String groupId) {

        ModelAndView modelAndView = new ModelAndView();
```

```
        try {
            daoStudent.deleteGroupOfStudent(studentId);
            modelAndView.setViewName("redirect:/show/group?groupId=" +
groupId);
        } catch (Exception throwables) {
            throwables.printStackTrace();
            modelAndView.setViewName(ERROR_PAGE.getPageName());
        }

        return modelAndView;
    }

    @RequestMapping(value = "/task")
    @GetMapping
    public ModelAndView deleteByTaskId(@RequestParam("taskId") String taskId,
                                       @RequestParam("subjectId") String
subjectId) {

        ModelAndView modelAndView = new ModelAndView();
        try {
            List<StudentSubject> studentSubjectList =
daoStudentSubject.getStudentSubjectsBySubjectId(subjectId);
            daoStudentTask.deleteStudentTasksByTaskId(taskId);
            daoTask.deleteTask(taskId);
            daoSubject.actualizeMaxGrade(subjectId);
            for (StudentSubject s: studentSubjectList) {

daoStudentSubject.actualizeTotalGrade(s.getStudentSubjectId());
            }
            modelAndView.setViewName("redirect:/show/subject?subjectId=" +
subjectId);
        } catch (Exception throwables) {
            throwables.printStackTrace();
            modelAndView.setViewName(ERROR_PAGE.getPageName());
        }

        return modelAndView;
    }
}
```

Клас “EditController”

```
package org.example.controllers;

import org.example.dao.implementations.*;
import org.example.entities.*;
import org.example.tools.custom.exceptions.WrongEntityIdException;
import org.example.tools.strings.PageName;
import org.example.tools.strings.Strings;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.ModelAndView;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import java.sql.SQLException;

import static org.example.tools.strings.PageName.ERROR_PAGE;
import static org.example.tools.strings.Role.STUDENT;
import static org.example.tools.strings.Role.TEACHER;
import static org.example.tools.strings.SessionAttributeName.*;

@Controller
@RequestMapping(value = "/edit")
public class EditController {

    private final DAOStudentImpl daoStudent;
    private final DAOTeacherImpl daoTeacher;
    private final DAOGroupImpl daoGroup;
    private final DAOSubjectImpl daoSubject;
    private final DAOStudentSubjectImpl daoStudentSubject;
    private final DAOTaskImpl daoTask;
    private final DAOStudentTaskImpl daoStudentTask;

    @Autowired
    public EditController(DAOStudentImpl daoStudent, DAOTeacherImpl
daoTeacher, DAOGroupImpl daoGroup, DAOSubjectImpl daoSubject,
DAOStudentSubjectImpl daoStudentSubject, DAOTaskImpl daoTask,
DAOStudentTaskImpl daoStudentTask) {
        this.daoStudent = daoStudent;
        this.daoTeacher = daoTeacher;
        this.daoGroup = daoGroup;
        this.daoSubject = daoSubject;
        this.daoStudentSubject = daoStudentSubject;
        this.daoTask = daoTask;
        this.daoStudentTask = daoStudentTask;
    }

    @RequestMapping(value = "/group/{groupId}")
    @GetMapping
    public ModelAndView editGroup(@RequestParam("groupName") String
groupName,
                                  @RequestParam("groupDescription")
String groupDescription,
                                  @PathVariable String groupId) {

        ModelAndView modelAndView = new ModelAndView();
        try {
            daoGroup.updateGroup(new Group(groupId, groupName,
groupDescription));

modelAndView.setViewName("redirect:/show/group?groupId="+groupId);
        } catch (SQLException throwables) {
            throwables.printStackTrace();
            modelAndView.setViewName(ERROR_PAGE.getPageName());
        }
        return modelAndView;
    }

    @RequestMapping(value = "/student/{studentId}")
```

```
    @GetMapping
    public ModelAndView editStudent(@RequestParam("loginName") String email,
                                    @RequestParam("firstName") String
firstName,
                                    @RequestParam("lastName") String
lastName,
                                    @RequestParam("headman") String
headman,
                                    @RequestParam("password") String
password,
                                    @RequestParam("groupId") String
groupId,
                                    @PathVariable String studentId) {

        ModelAndView modelAndView = new ModelAndView();
        Student student = new Student(
                studentId,
                email,
                firstName,
                lastName,
                password,
                Strings.NOT_YET.getStrings().equals(headman) ? null : headman
,
                Strings.NOT_YET.getStrings().equals(groupId) ? null : groupId
        );

        try {
            daoStudent.updateStudent(student);

modelAndView.setViewName("redirect:/redirect/profile?userId="+studentId+"&use
rRole="+student.getRole());
        } catch (SQLException throwables) {
            throwables.printStackTrace();
            modelAndView.setViewName(ERROR_PAGE.getPageName());
        }
        return modelAndView;
    }

    @RequestMapping(value = "/teacher/{teacherId}")
    @GetMapping
    public ModelAndView editTeacher(@RequestParam("loginName") String email,
                                    @RequestParam("firstName") String
firstName,
                                    @RequestParam("lastName") String
lastName,
                                    @PathVariable String teacherId) {

        ModelAndView modelAndView = new ModelAndView();
        Teacher teacher = new Teacher(
                teacherId,
                email,
                firstName,
                lastName,
                null
        );

        try {
            daoTeacher.updateTeacher(teacher);
```

```
modelAndView.setViewName("redirect:/show/teacher?teacherId="+teacherId);
        } catch (SQLException throwables) {
            throwables.printStackTrace();
            modelAndView.setViewName(ERROR_PAGE.getPageName());
        }
        return modelAndView;
    }

    @RequestMapping(value = "/student-group")
    @GetMapping
    public ModelAndView editStudentGroup(@RequestParam("studentId") String
studentId,
                                         @RequestParam("groupId") String
groupId) {

        ModelAndView modelAndView = new ModelAndView();

        try {
            daoStudent.updateGroupOfStudent(studentId, groupId);
            modelAndView.setViewName("redirect:/show/group?groupId=" +
groupId);
        } catch (SQLException throwables) {
            throwables.printStackTrace();
            modelAndView.setViewName(ERROR_PAGE.getPageName());
        }
        return modelAndView;
    }

    @RequestMapping(value = "/subject/{subjectId}")
    @GetMapping
    public ModelAndView editSubject(@RequestParam("subjectName") String
subjectName,
                                    @RequestParam("maxGrade") String
maxGrade,
                                    @RequestParam("passProcGrade") String
passProcGrade,
                                    @RequestParam("subjectDescription")
String subjectDescription,
                                    @PathVariable String subjectId) {

        ModelAndView modelAndView = new ModelAndView();
        Subject subject = new Subject(
                subjectId,
                subjectName,
                Integer.parseInt(maxGrade),
                Integer.parseInt(passProcGrade),
                subjectDescription
        );

        try {
            daoSubject.updateSubject(subject);

modelAndView.setViewName("redirect:/show/subject?subjectId="+subjectId);
        } catch (SQLException throwables) {
            throwables.printStackTrace();
            modelAndView.setViewName(ERROR_PAGE.getPageName());
        }
```

```
        return modelAndView;
    }


    @RequestMapping(value = "/task")
    @GetMapping
    public ModelAndView editTask(@RequestParam("taskId") String taskId,
                                 @RequestParam("subjectId") String
subjectId,
                                 @RequestParam("taskName") String
taskName,
                                 @RequestParam("maxGrade") String
maxGrade,
                                 @RequestParam("taskDescription") String
taskDescription) {

        ModelAndView modelAndView = new ModelAndView();

        try {
            daoTask.updateTaskNameAndGrade(taskId, taskName,
Integer.parseInt(maxGrade), taskDescription);
            daoSubject.actualizeMaxGrade(subjectId);

modelAndView.setViewName("redirect:/show/subject?subjectId="+subjectId);
        } catch (SQLException throwables) {
            throwables.printStackTrace();
            modelAndView.setViewName(ERROR_PAGE.getPageName());
        }
        return modelAndView;
    }

    @RequestMapping(value = "/student-task/{teacherId}")
    @GetMapping
    public ModelAndView editStudentTask(@RequestParam("taskId") String
taskId,
                                 @RequestParam("studentSubjectId") String
studentSubjectId,
                                 @RequestParam("grade") String grade,
                                 @PathVariable String teacherId) {

        ModelAndView modelAndView = new ModelAndView();
        StudentSubject studentSubject = null;
        try {
            daoStudentTask.updateStudentTaskGrade(taskId, studentSubjectId,
Integer.parseInt(grade), teacherId);
            daoStudentSubject.actualizeTotalGrade(studentSubjectId);
            studentSubject =
daoStudentSubject.getStudentSubjectById(studentSubjectId);
            modelAndView.setViewName("redirect:/show/student-task-
list?subjectId=" +
                    studentSubject.getSubjectId() + "&studentId=" +
studentSubject.getStudentId());
        } catch (SQLException | WrongEntityIdException throwables) {
            throwables.printStackTrace();
            modelAndView.setViewName(ERROR_PAGE.getPageName());
        }
        return modelAndView;
    }
```

```
    @RequestMapping(value = "/user-password")
    @GetMapping
    public ModelAndView changeUserPassword(@RequestParam("newPassword")
String newPassword,
                                           @RequestParam("confirmPassword")
String confirmPassword,
                                           HttpServletRequest request) {

        ModelAndView modelAndView = new ModelAndView();
        if(newPassword.equals(confirmPassword)) {
            HttpSession session = request.getSession(false);
            String currentUserId = (String)
session.getAttribute(CURRENT_USER_ID.getSessionAttributeName());
            String currentUserRole = (String)
session.getAttribute(CURRENT_USER_ROLE.getSessionAttributeName());

            User user = null;
            try {
                if (STUDENT.getRole().equals(currentUserRole)) {
                    daoStudent.changeStudentPassword(currentUserId,
newPassword);
                    user = daoStudent.getStudentById(currentUserId);
                } else if (TEACHER.getRole().equals(currentUserRole)) {
                    daoTeacher.changeTeacherPassword(currentUserId,
newPassword);
                    user = daoTeacher.getTeacherById(currentUserId);
                } else {
                }
            } catch (SQLException | WrongEntityIdException throwables) {
                throwables.printStackTrace();
            }
            modelAndView.addObject("user", user);
            modelAndView.addObject("userRole", currentUserRole);
        }

        modelAndView.addObject("isCurrentProfile", "Yes");
        modelAndView.setViewName(PageName.PROFILE_PAGE.getPageName());
        return modelAndView;
    }
}
```

Клас “LoginController”

```
package org.example.controllers;

import org.example.dao.implementations.DAOStudentImpl;
import org.example.dao.implementations.DAOTeacherImpl;
import org.example.entities.Student;
import org.example.entities.Teacher;
import org.example.tools.custom.exceptions.WrongLoginDataException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
```

```
import org.springframework.web.servlet.ModelAndView;

import javax.servlet.http.HttpServletRequest;

import java.sql.SQLException;

import static org.example.tools.strings.PageName.*;

@Controller
public class LoginController extends AbstractController{

    private final DAOStudentImpl daoStudent;
    private final DAOTeacherImpl daoTeacher;

    @Autowired
    public LoginController(DAOStudentImpl daoStudent, DAOTeacherImpl
daoTeacher) {
        this.daoStudent = daoStudent;
        this.daoTeacher = daoTeacher;
    }

    @RequestMapping(value = "/login")
    @GetMapping
    public ModelAndView checkUser(@RequestParam("loginUserName") String
email,
                                  @RequestParam("loginPassword") String
password,
                                  HttpServletRequest request) {

        ModelAndView modelAndView = new ModelAndView();
        try {
            if (daoTeacher.isExistTeacherByEmail(email)) {
                modelAndView = loginTeacher(email, password, request);
            } else if (daoStudent.isExistStudentByEmail(email)) {
                modelAndView = loginStudent(email, password, request);
            } else {
                modelAndView.addObject("ExceptionMessage", "You entered wrong
login!");
                modelAndView.setViewName(LOGIN_PAGE.getPageName());
            }
        } catch (SQLException e) {
            modelAndView.addObject("ExceptionMessage", "Sorry, we have
unexpected error!");
            modelAndView.setViewName(LOGIN_PAGE.getPageName());
        }
        return modelAndView;
    }

    private ModelAndView loginTeacher(String email, String password,
HttpServletRequest request) {
        ModelAndView modelAndView = new ModelAndView();
        try {
            Teacher teacher = daoTeacher.getTeacherByEmailAndPassword(email,
password);
            setBaseSessionVariables(request, teacher);
            modelAndView.setViewName("redirect:/show/teacher?teacherId=" +
teacher.getTeacherId());
        } catch (WrongLoginDataException e) {
```

```
            modelAndView.addObject("ExceptionMessage", "You entered wrong
password!");
            modelAndView.setViewName(LOGIN_PAGE.getPageName());
        }
        return modelAndView;
    }

    private ModelAndView loginStudent(String email, String password,
HttpServletRequest request) {
        ModelAndView modelAndView = new ModelAndView();
        try {
            Student student = daoStudent.getStudentByEmailAndPassword(email,
password);
            setBaseSessionVariables(request, student);
            modelAndView.setViewName("redirect:/show/student?studentId=" +
student.getStudentId());
        } catch (WrongLoginDataException e) {
            modelAndView.addObject("ExceptionMessage", "You entered wrong
password!");
            modelAndView.setViewName(LOGIN_PAGE.getPageName());
        }
        return modelAndView;
    }
}
```

Клас “LogoutController”

```
package org.example.controllers;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

import java.net.HttpCookie;

import static org.example.tools.strings.PageName.LOGIN_PAGE;

@Controller
public class LogoutController {

    @RequestMapping(value = "/logout")
    @GetMapping
    public ModelAndView logout(HttpServletRequest request) {
        HttpSession session = request.getSession(false);
        session.invalidate();
        return new ModelAndView("redirect:/redirect/login");
    }
}
```

Клас “RedirectController”

```
package org.example.controllers;

import org.example.dao.implementations.*;
import org.example.entities.*;
import org.example.tools.custom.exceptions.WrongEntityIdException;
import org.example.tools.strings.PageName;
import org.example.tools.strings.Role;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.ModelAndView;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

import java.util.HashMap;
import java.util.List;

import static org.example.tools.strings.SessionAttributeName.CURRENT_USER_ID;

@Controller
@RequestMapping(value = "/redirect")
public class RedirectController extends AbstractController{

    private final DAOStudentImpl daoStudent;
    private final DAOTeacherImpl daoTeacher;
    private final DAOGroupImpl daoGroup;
    private final DAOSubjectImpl daoSubject;
    private final DAOTaskImpl daoTask;
    private final DAOStudentSubjectImpl daoStudentSubject;

    @Autowired
    public RedirectController(DAOStudentImpl daoStudent, DAOTeacherImpl
daoTeacher, DAOGroupImpl daoGroup, DAOSubjectImpl daoSubject, DAOTaskImpl
daoTask, DAOStudentSubjectImpl daoStudentSubject) {
        this.daoStudent = daoStudent;
        this.daoTeacher = daoTeacher;
        this.daoGroup = daoGroup;
        this.daoSubject = daoSubject;
        this.daoTask = daoTask;
        this.daoStudentSubject = daoStudentSubject;
    }

    @RequestMapping(value = "/add/student")
    @GetMapping
    public ModelAndView redirectAddStudent() {
        ModelAndView modelAndView = new ModelAndView();

        List<Student> headmanList = null;
        List<Group> groupList =  null;
        try {
            headmanList = daoStudent.getAllStudents();
            groupList = daoGroup.getAllGroups();
        } catch (WrongEntityIdException throwables) {
```

```
            throwables.printStackTrace();
        }

        modelAndView.addObject("headmanList", headmanList);
        modelAndView.addObject("groupList", groupList);
        modelAndView.addObject("action", "add");
        modelAndView.setViewName("add-edit-student-page");
        return modelAndView;
    }

    @RequestMapping(value = "/edit/student")
    @GetMapping
    public ModelAndView redirectEditStudent(@RequestParam("studentId") String
studentId) {

        ModelAndView modelAndView = new ModelAndView();
        StudentInfoSet studentInfoSet = null;
        List<Student> headmanList = null;
        List<Group> groupList =  null;

        try {
            studentInfoSet = daoStudent.getStudentInfoSet(studentId);
            headmanList = daoStudent.getAllStudents();
            groupList = daoGroup.getAllGroups();
        } catch (WrongEntityIdException e) {
            e.printStackTrace();
        }

        removeStudentById(headmanList,
studentInfoSet.getStudent().getStudentId());
        if (studentInfoSet.getStudent().getHeadman() != null) {
            removeStudentById(headmanList,
studentInfoSet.getStudent().getHeadman());
        }

        if (studentInfoSet.getStudent().getGroupId() != null) {
            removeGroupById(groupList,
studentInfoSet.getStudent().getGroupId());
        }

        modelAndView.addObject("headmanList", headmanList);
        modelAndView.addObject("groupList", groupList);
        modelAndView.addObject("studentInfoSet", studentInfoSet);
        modelAndView.addObject("action", "edit");
        modelAndView.setViewName("add-edit-student-page");
        return modelAndView;
    }

    @RequestMapping(value = "/add/teacher")
    @GetMapping
    public ModelAndView redirectAddTeacher() {
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.addObject("action", "add");
        modelAndView.setViewName("add-edit-teacher-page");
        return modelAndView;
    }

    @RequestMapping(value = "/edit/teacher")
```

```
    @GetMapping
    public ModelAndView redirectEditTeacher(@RequestParam("teacherId") String
teacherId) {

        ModelAndView modelAndView = new ModelAndView();
        Teacher teacher = null;

        try {
            teacher = daoTeacher.getTeacherById(teacherId);
        } catch (WrongEntityIdException e) {
            e.printStackTrace();
        }

        modelAndView.addObject("teacher", teacher);
        modelAndView.addObject("action", "edit");
        modelAndView.setViewName("add-edit-teacher-page");
        return modelAndView;
    }

    @RequestMapping(value = "/add/group")
    @GetMapping
    public ModelAndView redirectAddGroup() {
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.addObject("action", "add");
        modelAndView.setViewName("add-edit-group-page");
        return modelAndView;
    }

    @RequestMapping(value = "/edit/group")
    @GetMapping
    public ModelAndView redirectEditGroup(@RequestParam("groupId") String
groupId) {

        ModelAndView modelAndView = new ModelAndView();
        Group group = null;
        try {
            group = daoGroup.getGroupById(groupId);
        } catch (WrongEntityIdException e) {
            e.printStackTrace();
        }
        modelAndView.addObject("group", group);
        modelAndView.addObject("action", "edit");
        modelAndView.setViewName("add-edit-group-page");
        return modelAndView;
    }

    @RequestMapping(value = "/add/task")
    @GetMapping
    public ModelAndView redirectAddTask(@RequestParam("subjectId") String
subjectId) {

        ModelAndView modelAndView = new ModelAndView();
        Subject subject = null;
        try {
            subject = daoSubject.getSubjectById(subjectId);
        } catch (WrongEntityIdException e) {
            e.printStackTrace();
        }
```

```
        modelAndView.addObject("subject", subject);
        modelAndView.addObject("action", "add");
        modelAndView.setViewName("add-edit-task-page");
        return modelAndView;
    }

    @RequestMapping(value = "/edit/task")
    @GetMapping
    public ModelAndView redirectEditTask(@RequestParam("taskId") String
taskId) {

        ModelAndView modelAndView = new ModelAndView();
        Task task = null;
        try {
            task = daoTask.getTaskById(taskId);
        } catch (WrongEntityIdException e) {
            e.printStackTrace();
        }
        modelAndView.addObject("task", task);
        modelAndView.addObject("action", "edit");
        modelAndView.setViewName("add-edit-task-page");
        return modelAndView;
    }

    @RequestMapping(value = "/add/subject")
    @GetMapping
    public ModelAndView redirectAddSubject() {
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.addObject("action", "add");
        modelAndView.setViewName("add-edit-subject-page");
        return modelAndView;
    }

    @RequestMapping(value = "/edit/subject")
    @GetMapping
    public ModelAndView redirectEditSubject(@RequestParam("subjectId") String
subjectId) {

        ModelAndView modelAndView = new ModelAndView();
        Subject subject = null;
        try {
            subject = daoSubject.getSubjectById(subjectId);
        } catch (WrongEntityIdException e) {
            e.printStackTrace();
        }
        modelAndView.addObject("subject", subject);
        modelAndView.addObject("action", "edit");
        modelAndView.setViewName("add-edit-subject-page");
        return modelAndView;
    }

    @RequestMapping(value = "/main")
    @GetMapping
    public ModelAndView redirectMain() {
        return new ModelAndView("main-page");
    }
```

```
    @RequestMapping(value = "/login")
    @GetMapping
    public ModelAndView redirectLogin() {
        return new ModelAndView(PageName.LOGIN_PAGE.getPageName());
    }

    @RequestMapping(value = "/profile")
    @GetMapping
    public ModelAndView redirectProfile(@RequestParam("userId") String
userId,
                                        @RequestParam("userRole") String
userRole,
                                        HttpServletRequest request) {


        ModelAndView modelAndView = new ModelAndView();
        HttpSession session = request.getSession(false);
        String isCurrentProfile =

session.getAttribute(CURRENT_USER_ID.getSessionAttributeName()).equals(userId
)?
                        "Yes" : "No";

        try {
            if (Role.STUDENT.getRole().equals(userRole)) {
                Student user = daoStudent.getStudentById(userId);
                StudentInfoSet studentInfoSet =
daoStudent.getStudentInfoSet(user.getStudentId());
                modelAndView.addObject("studentInfoSet", studentInfoSet);
                HashMap<Subject, StudentSubject> studentSubjectMap =
daoStudentSubject.getSubjectsInfoByStudentId(userId);
                modelAndView.addObject("studentSubjectMap",
studentSubjectMap);
                modelAndView.addObject("user", user);
            } else {
                Teacher user = daoTeacher.getTeacherById(userId);
                List<Subject> teacherSubjectMap =
daoSubject.getSubjectsByTeacherId(userId);
                modelAndView.addObject("teacherSubjectMap",
teacherSubjectMap);
                modelAndView.addObject("user", user);
            }
        } catch (WrongEntityIdException e) {
            e.printStackTrace();
        }

        modelAndView.addObject("userRole", userRole);
        modelAndView.addObject("isCurrentProfile", isCurrentProfile);
        modelAndView.setViewName(PageName.PROFILE_PAGE.getPageName());
        return modelAndView;
    }

}
```

Клас “SearchController”

```
package org.example.controllers;

import org.example.dao.implementations.DAOGroupImpl;
import org.example.dao.implementations.DAOStudentImpl;
import org.example.dao.implementations.DAOSubjectImpl;
import org.example.entities.Group;
import org.example.entities.Student;
import org.example.entities.Subject;
import org.example.tools.custom.exceptions.WrongEntityIdException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;

import java.sql.SQLException;
import java.util.List;

@Controller
@RequestMapping(value = "/search")
public class SearchController {

    private final DAOStudentImpl daoStudent;
    private final DAOGroupImpl daoGroup;
    private final DAOSubjectImpl daoSubject;

    @Autowired
    public SearchController(DAOStudentImpl daoStudent, DAOGroupImpl daoGroup,
DAOSubjectImpl daoSubject) {
        this.daoStudent = daoStudent;
        this.daoGroup = daoGroup;
        this.daoSubject = daoSubject;
    }

    @RequestMapping(value = "/by-student-full-name")
    @GetMapping
    public @ResponseBody String
searchByStudentFullName(@RequestParam("studentFullName") String
studentFullName) {
        String result;
        try {
            List<Student> studentSearchList =
daoStudent.searchStudentsByFullName(studentFullName);
            result = studentSearchListBuild(studentSearchList);
        } catch (WrongEntityIdException | SQLException throwable) {
            throwable.printStackTrace();
            result = "Error!";
        }
        return result;
    }

    @RequestMapping(value = "/by-group-name")
    @GetMapping
    public @ResponseBody String searchByGroupName(@RequestParam("groupName")
String groupName) {
```

```
        String result;
        try {
            List<Group> groupSearchList =
daoGroup.searchGroupsByName(groupName);
            result = groupSearchListBuild(groupSearchList);
        } catch (WrongEntityIdException | SQLException throwable) {
            throwable.printStackTrace();
            result = "Error!";
        }
        return result;
    }

    @RequestMapping(value = "/by-subject-name")
    @GetMapping
    public @ResponseBody String
searchBySubjectName(@RequestParam("subjectName") String subjectName) {
        String result;
        try {
            List<Subject> subjectSearchList =
daoSubject.searchSubjectsByName(subjectName);
            result = subjectSearchListBuild(subjectSearchList);
        } catch (WrongEntityIdException | SQLException throwable) {
            throwable.printStackTrace();
            result = "Error!";
        }
        return result;
    }

    public String studentSearchListBuild(List<Student> studentSearchList) {
        StringBuilder result = new StringBuilder();
        for (Student st: studentSearchList) {

            result.append("<li><a
href=\"/Lab3GradeBook/redirect/profile?userId=")

.append(st.getStudentId()).append("&userRole=").append(st.getRole()).append("
\">")

                    .append(st.getFullNameWithEmail())
                    .append("</a></li>");

        }
        return result.toString();
    }

    public String groupSearchListBuild(List<Group> groupSearchList) {
        StringBuilder result = new StringBuilder();
        for (Group gr: groupSearchList) {

            result.append("<li><a href=\"/Lab3GradeBook/show/group?groupId=")
                    .append(gr.getGroupId()).append("\">")
                    .append(gr.getGroupName()).append("</a></li>");
        }
        return result.toString();
    }

    public String subjectSearchListBuild(List<Subject> subjectSearchList) {
        StringBuilder result = new StringBuilder();
```

```
        for (Subject s: subjectSearchList) {

            result.append("<li><a
href=\"/Lab3GradeBook/show/subject?subjectId=")
                    .append(s.getSubjectId()).append("\">")
                    .append(s.getSubjectName()).append("</a></li>");
        }
        return result.toString();
    }
}
```

Клас “ShowController”

```
package org.example.controllers;

import org.example.dao.implementations.*;
import org.example.entities.*;
import org.example.tools.custom.exceptions.WrongEntityIdException;
import org.example.tools.strings.Role;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.ModelAndView;

import javax.servlet.http.HttpServletRequest;
import java.util.HashMap;
import java.util.List;

import static org.example.tools.strings.PageName.*;

@Controller
@RequestMapping(value = "/show")
public class ShowController extends AbstractController{

    private final DAOStudentImpl daoStudent;
    private final DAOTeacherImpl daoTeacher;
    private final DAOGroupImpl daoGroup;
    private final DAOSubjectImpl daoSubject;
    private final DAOTaskImpl daoTask;
    private final DAOStudentTaskImpl daoStudentTask;
    private final DAOStudentSubjectImpl daoStudentSubject;

    @Autowired
    public ShowController(DAOStudentImpl daoStudent, DAOTeacherImpl
daoTeacher, DAOGroupImpl daoGroup, DAOSubjectImpl daoSubject, DAOTaskImpl
daoTask, DAOStudentTaskImpl daoStudentTask, DAOStudentSubjectImpl
daoStudentSubject) {
        this.daoStudent = daoStudent;
        this.daoTeacher = daoTeacher;
        this.daoGroup = daoGroup;
        this.daoSubject = daoSubject;
        this.daoTask = daoTask;
        this.daoStudentTask = daoStudentTask;
        this.daoStudentSubject = daoStudentSubject;
```

```
    }

    @RequestMapping(value = "/group")
    @GetMapping
    public ModelAndView showByGroupId(@RequestParam("groupId") String
groupId) {

        ModelAndView modelAndView = new ModelAndView();
        Group group = null;
        List<Student> studentList = null;
        List<Student> availableStudents = null;

        try {
            group = daoGroup.getGroupById(groupId);
            studentList = daoStudent.getStudentsByGroupId(groupId);
            availableStudents = daoStudent.getAllStudents();
            removeStudentByList(availableStudents, studentList);
        } catch (WrongEntityIdException e) {
            e.printStackTrace();
        }

        modelAndView.addObject("group", group);
        modelAndView.addObject("studentList", studentList);
        modelAndView.addObject("availableStudents", availableStudents);
        modelAndView.setViewName("show-group-page");
        return modelAndView;
    }

    @RequestMapping(value = "/student")
    @GetMapping
    public ModelAndView showByStudentId(@RequestParam("studentId") String
studentId) {

        ModelAndView modelAndView = new ModelAndView();
        modelAndView.setViewName("redirect:/redirect/profile?userId=" +
                studentId + "&userRole=" + Role.STUDENT.getRole());
        return modelAndView;
    }

    @RequestMapping(value = "/teacher")
    @GetMapping
    public ModelAndView showByTeacherId(@RequestParam("teacherId") String
teacherId) {
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.setViewName("redirect:/redirect/profile?userId=" +
                teacherId + "&userRole=" + Role.TEACHER.getRole());
        return modelAndView;
    }

    @RequestMapping(value = "/subject")
    @GetMapping
    public ModelAndView showBySubjectId(@RequestParam("subjectId") String
subjectId) {

        ModelAndView modelAndView = new ModelAndView();
        Subject subject = null;
        HashMap<Student, StudentSubject> studentSubjectMap = null;
        List<Teacher> teacherList = null;
```

```
        List<Student> availableStudents = null;
        List<Teacher> availableTeachers = null;
        List<Task> taskList = null;

        try {
            subject = daoSubject.getSubjectById(subjectId);
            studentSubjectMap =
daoStudentSubject.getStudentsInfoBySubjectId(subjectId);
            availableStudents = daoStudent.getAllStudents();
            removeStudentByList(availableStudents,
daoSubject.getStudentsBySubjectId(subjectId));
            teacherList = daoTeacher.getTeachersBySubjectId(subjectId);
            availableTeachers = daoTeacher.getAllTeachers();
            removeTeacherByList(availableTeachers, teacherList);
            taskList = daoTask.getTaskListBySubjectId(subjectId);
        } catch (WrongEntityIdException e) {
            e.printStackTrace();
        }

        modelAndView.addObject("availableStudents", availableStudents);
        modelAndView.addObject("availableTeachers", availableTeachers);
        modelAndView.addObject("taskList", taskList);
        modelAndView.addObject("subject", subject);
        modelAndView.addObject("studentSubjectMap", studentSubjectMap);
        modelAndView.addObject("teacherList", teacherList);
        modelAndView.setViewName("show-subject-page");
        return modelAndView;
    }

    @RequestMapping(value = "/student-task-list")
    @GetMapping
    public ModelAndView showStudentTaskList(@RequestParam("subjectId") String
subjectId,
                                            @RequestParam("studentId") String
studentId) {

        ModelAndView modelAndView = new ModelAndView();
        Student student = null;
        Subject subject = null;
        StudentSubject studentSubject = null;
        List<StudentTaskInfoSet> studentTaskInfoMap = null;

        try {
            subject = daoSubject.getSubjectById(subjectId);
            student = daoStudent.getStudentById(studentId);
            studentSubject =
daoStudentSubject.getStudentSubjectBySubjectIdAndStudentId(subjectId,
studentId);
            studentTaskInfoMap = daoStudentTask.

getStudentTaskInfoSetList(studentSubject.getStudentSubjectId());
        } catch (WrongEntityIdException e) {
            e.printStackTrace();
        }

        modelAndView.addObject("student", student);
        modelAndView.addObject("subject", subject);
        modelAndView.addObject("studentSubject", studentSubject);
```

```
            modelAndView.addObject("studentTaskInfoMap", studentTaskInfoMap);
            modelAndView.setViewName("student-task-list");
            return modelAndView;
        }

        @RequestMapping(value = "/student-all")
        @GetMapping
        public ModelAndView showAllStudent(HttpServletRequest request) {
            ModelAndView modelAndView = new ModelAndView();
            try {
                List<Student> studentAllList = daoStudent.getAllStudents();
                List<StudentInfoSet> studentInfoSetList =
    daoStudent.getStudentInfoSetList(studentAllList);
                modelAndView.addObject("studentInfoSetList", studentInfoSetList);
                modelAndView.setViewName(STUDENT_LIST_PAGE.getPageName());
            } catch (WrongEntityIdException e) {
                modelAndView.setViewName(ERROR_PAGE.getPageName());
            }
            return modelAndView;
        }

        @RequestMapping(value = "/group-all")
        @GetMapping
        public ModelAndView showAllGroup(HttpServletRequest request) {
            ModelAndView modelAndView = new ModelAndView();
            try {
                List<Group> groupAllList = daoGroup.getAllGroups();
                modelAndView.addObject("groups", groupAllList);
                modelAndView.setViewName(GROUP_LIST_PAGE.getPageName());
            } catch (WrongEntityIdException e) {
                modelAndView.setViewName(ERROR_PAGE.getPageName());
            }
            return modelAndView;
        }

        @RequestMapping(value = "/subject-all")
        @GetMapping
        public ModelAndView showAllSubject(HttpServletRequest request) {
            ModelAndView modelAndView = new ModelAndView();

            try {
                List<Subject> subjectAllList = daoSubject.getAllSubjects();
                modelAndView.addObject("subjects", subjectAllList);
                modelAndView.setViewName(SUBJECT_LIST_PAGE.getPageName());
            } catch (WrongEntityIdException e) {
                modelAndView.setViewName(ERROR_PAGE.getPageName());
            }
            return modelAndView;
        }

        @RequestMapping(value = "/teacher-all")
        @GetMapping
        public ModelAndView showAllTeacher(HttpServletRequest request) {
            ModelAndView modelAndView = new ModelAndView();

            try {
                List<Teacher> teacherAllList = daoTeacher.getAllTeachers();
                modelAndView.addObject("teachers", teacherAllList);
```

```
            modelAndView.setViewName(TEACHER_LIST_PAGE.getPageName());
        } catch (WrongEntityIdException e) {
            modelAndView.setViewName(ERROR_PAGE.getPageName());
        }
        return modelAndView;
    }
}
```

Клас “Oracle”

```
package org.example.dao.connection;

import org.apache.log4j.Logger;
import org.example.tools.custom.exceptions.ConnectionFailedException;
import org.example.tools.custom.exceptions.WrongEntityIdException;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.sql.DataSource;
import java.sql.*;
import java.util.Hashtable;

public class Oracle implements ConnectionInterface{

    protected Connection connection = null;
    protected PreparedStatement statement = null;
    protected ResultSet result = null;
    private DataSource dataSource;
    private Context context;
    private Hashtable<String, String> hashtable = new Hashtable<>();
    private static final Logger logger = Logger.getLogger(Oracle.class);

    @Override
    public void connect() {
        try {
            hashtable.put(Context.INITIAL_CONTEXT_FACTORY,
"weblogic.jndi.WLInitialContextFactory");
            hashtable.put(Context.PROVIDER_URL, "t3://localhost:7001");
            context = new InitialContext(hashtable);
            dataSource = (DataSource) context.lookup("MyData");
            connection = dataSource.getConnection();
        } catch (SQLException | NamingException e) {
            return;
        }
    }

    @Override
    public void disconnect() {
        try {
            if(connection!=null){
                connection.close();
            }
            if(context!=null){
                context.close();
            }
```

```
        } catch (SQLException | NamingException e) {
            return;
        }
    }

    @Override
    public void init() {
    }
}
```

Клас “DAOGroupImpl”

```
package org.example.dao.implementations;

import org.apache.log4j.Logger;
import org.example.dao.connection.Oracle;
import org.example.dao.interfaces.DAOGroup;
import org.example.entities.Group;
import org.example.entities.Student;
import org.example.entities.StudentSubject;
import org.example.tools.custom.exceptions.WrongEntityIdException;
import org.springframework.stereotype.Repository;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import static org.example.tools.strings.Query.*;

@Repository
public class DAOGroupImpl extends Oracle implements DAOGroup {

    private static final Logger logger =
Logger.getLogger(DAOGroupImpl.class);

    @Override
    public Group getGroupById(String id) throws WrongEntityIdException {
        try {
            connect();
            statement = connection.prepareStatement(
                    GROUP_BY_ID.getQuery());

            statement.setInt(1, Integer.parseInt(id));

            result = statement.executeQuery();
            if(result.next()) {
                return Group.parse(result);
            } else {
                throw new WrongEntityIdException("desc ");
            }
        } catch (SQLException | WrongEntityIdException e) {
            e.printStackTrace();
            throw new WrongEntityIdException("desc ", e);
        } finally {
```

```
            disconnect();
        }
    }

    @Override
    public void addGroup(Group group) throws SQLException {
        try {
            connect();
            statement = connection.prepareStatement(ADD_GROUP.getQuery());
            statement.setString(1, group.getGroupName());
            statement.setString(2, group.getGroupDescription());
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new SQLException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public void updateGroup(Group group) throws SQLException {
        try {
            connect();
            statement = connection.prepareStatement(UPDATE_GROUP.getQuery());

            statement.setString(1, group.getGroupName());
            statement.setString(2, group.getGroupDescription());
            statement.setInt(3, Integer.parseInt(group.getGroupId()));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new SQLException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public void deleteGroup(String groupId) {
        try {
            connect();
            statement =
connection.prepareStatement(DELETE_GROUP_BY_ID.getQuery());
            statement.setInt(1, Integer.parseInt(groupId));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc");
        } finally {
            disconnect();
        }
    }

    @Override
    public List<Group> getAllGroups() throws WrongEntityIdException {
        try {
            connect();
            List<Group> list = new ArrayList<>();
```

```
            statement = connection.prepareStatement(ALL_GROUPS.getQuery());
            result = statement.executeQuery();
            while (result.next()) {
                list.add(Group.parse(result));
            }
            return list;
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new WrongEntityIdException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public List<Group> searchGroupsByName(String text) throws SQLException,
WrongEntityIdException {
        if (text == null || text.isEmpty()) {
            return Collections.emptyList();
        } else if (text.equals("*")) {
            return getAllGroups();
        }

        try {
            connect();
            List<Group> list = new ArrayList<>();
            statement =
connection.prepareStatement(SEARCH_BY_GROUP_NAME.getQuery());
            statement.setString(1, text);
            result = statement.executeQuery();
            while (result.next()) {
                list.add(Group.parse(result));
            }
            return list;
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new SQLException("desc", e);
        } finally {
            disconnect();
        }
    }
}
```

Клас “DAOStudentImpl”

```
package org.example.dao.implementations;

import org.apache.log4j.Logger;
import org.example.dao.connection.Oracle;
import org.example.dao.interfaces.DAOGroup;
import org.example.dao.interfaces.DAOStudent;
import org.example.entities.Group;
import org.example.entities.Student;
import org.example.entities.StudentInfoSet;
import org.example.tools.custom.exceptions.WrongEntityIdException;
import org.example.tools.custom.exceptions.WrongLoginDataException;
```

```
import org.springframework.stereotype.Repository;

import java.sql.SQLException;
import java.sql.Types;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import static org.example.tools.strings.Query.*;

@Repository
public class DAOStudentImpl extends Oracle implements DAOStudent {

    private static final Logger logger =
Logger.getLogger(DAOStudentImpl.class);

    @Override
    public Student getStudentById(String id) throws WrongEntityIdException {
        try {
            connect();
            statement = connection.prepareStatement(
                    STUDENT_BY_ID.getQuery());

            statement.setInt(1, Integer.parseInt(id));

            result = statement.executeQuery();
            if(result.next()) {
                return Student.parse(result);
            } else {
                throw new WrongEntityIdException("desc ");
            }
        } catch (SQLException | WrongEntityIdException e) {
            e.printStackTrace();
            throw new WrongEntityIdException("desc ", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public boolean isExistStudentByEmail(String email) throws SQLException {
        try {
            connect();
            statement = connection.prepareStatement(
                    EXIST_STUDENT_BY_EMAIL.getQuery());

            statement.setString(1, email);
            result = statement.executeQuery();
            return result.next();
        } catch (SQLException e) {
            e.printStackTrace();
            throw new SQLException("desc ", e);
        } finally {
            disconnect();
        }
    }

    @Override
```

```
    public Student getStudentByEmailAndPassword(String email, String
password) throws WrongLoginDataException {
        try {
            connect();
            statement = connection.prepareStatement(
                    STUDENT_BY_EMAIL_AND_PASSWORD.getQuery());

            statement.setString(1, email);
            statement.setString(2, password);

            result = statement.executeQuery();
            if(result.next()) {
                return Student.parse(result);
            } else {
                throw new WrongLoginDataException("desc ");
            }
        } catch (SQLException | WrongLoginDataException e) {
            e.printStackTrace();
            throw new WrongLoginDataException("desc ", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public StudentInfoSet getStudentInfoSet(String studentId) throws
WrongEntityIdException {
        DAOGroup daoGroup = new DAOGroupImpl();
        Student student = getStudentById(studentId);
        Student headman = null;
        if (student.getHeadman() != null) {
            headman = getStudentById(student.getHeadman());
        }
        Group group = null;
        if (student.getGroupId() != null) {
            group = daoGroup.getGroupById(student.getGroupId());
        }
        return new StudentInfoSet(student, group, headman);
    }

    @Override
    public List<StudentInfoSet> getStudentInfoSetList(List<Student>
studentList) throws WrongEntityIdException {
        List<StudentInfoSet> studentInfoSetList =  new ArrayList<>();
        for (Student s: studentList) {
            studentInfoSetList.add(getStudentInfoSet(s.getStudentId()));
        }
        return studentInfoSetList;
    }

    @Override
    public void changeStudentPassword(String studentId, String password)
throws SQLException {
        try {
            connect();
            statement =
connection.prepareStatement(CHANGE_STUDENT_PASSWORD.getQuery());
```

```
            statement.setString(1, password);
            statement.setInt(2, Integer.parseInt(studentId));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new SQLException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public void addStudent(Student student) throws SQLException {
        try {
            connect();
            statement = connection.prepareStatement(ADD_STUDENT.getQuery());

            if (student.getHeadman() == null ||
student.getHeadman().isEmpty()) {
                statement.setNull(1, Types.INTEGER);
            } else {
                statement.setInt(1, Integer.parseInt(student.getHeadman()));
            }

            statement.setString(2, student.getEmail());
            statement.setString(3, student.getFirstName());
            statement.setString(4, student.getLastName());

            if (student.getGroupId() == null ||
student.getGroupId().isEmpty()) {
                statement.setNull(5, Types.INTEGER);
            } else {
                statement.setInt(5, Integer.parseInt(student.getGroupId()));
            }

            statement.setString(6, student.getPassword());
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new SQLException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public void updateStudent(Student student) throws SQLException {
        try {
            connect();
            statement =
connection.prepareStatement(UPDATE_STUDENT.getQuery());

            statement.setString(1, student.getFirstName());
            statement.setString(2, student.getLastName());

            if (student.getHeadman() == null ||
student.getHeadman().isEmpty()) {
                statement.setNull(3, Types.INTEGER);
```

```
            } else {
                statement.setInt(3, Integer.parseInt(student.getHeadman()));
            }

            if (student.getGroupId() == null ||
student.getGroupId().isEmpty()) {
                statement.setNull(4, Types.INTEGER);
            } else {
                statement.setInt(4, Integer.parseInt(student.getGroupId()));
            }

            statement.setInt(5, Integer.parseInt(student.getStudentId()));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new SQLException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public void updateGroupOfStudent(String studentId, String groupId) throws
SQLException {
        try {
            connect();
            statement =
connection.prepareStatement(UPDATE_GROUP_OF_STUDENT.getQuery());

            if (groupId == null || groupId.isEmpty()) {
                statement.setNull(1, Types.INTEGER);
            } else {
                statement.setInt(1, Integer.parseInt(groupId));
            }
            statement.setInt(2, Integer.parseInt(studentId));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new SQLException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public void deleteGroupOfStudent(String studentId) throws SQLException {
        updateGroupOfStudent(studentId, null);
    }

    @Override
    public void deleteStudent(String studentId) {
        try {
            connect();
            statement =
connection.prepareStatement(DELETE_STUDENT_BY_ID.getQuery());
            statement.setInt(1, Integer.parseInt(studentId));
            statement.execute();
        } catch (SQLException e) {
```

```
            logger.info("desc");
        } finally {
            disconnect();
        }
    }

    @Override
    public List<Student> getStudentsByGroupId(String groupId) throws
WrongEntityIdException {
        try {
            connect();
            List<Student> list = new ArrayList<>();
            statement =
connection.prepareStatement(STUDENTS_BY_GROUP_ID.getQuery());
            statement.setInt(1, Integer.parseInt(groupId));
            result = statement.executeQuery();
            while (result.next()) {
                list.add(Student.parse(result));
            }
            return list;
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new WrongEntityIdException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public List<Student> getAllStudents() throws WrongEntityIdException {
        try {
            connect();
            List<Student> list = new ArrayList<>();
            statement = connection.prepareStatement(ALL_STUDENTS.getQuery());
            result = statement.executeQuery();
            while (result.next()) {
                list.add(Student.parse(result));
            }
            return list;
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new WrongEntityIdException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public List<Student> searchStudentsByFullName(String text) throws
SQLException, WrongEntityIdException {
        if (text == null || text.isEmpty()) {
            return Collections.emptyList();
        } else if (text.equals("*")) {
            return getAllStudents();
        }

        try {
            connect();
```

```
            List<Student> list = new ArrayList<>();
            statement =
connection.prepareStatement(SEARCH_BY_STUDENT_FULL_NAME.getQuery());
            statement.setString(1, text);
            result = statement.executeQuery();
            while (result.next()) {
                list.add(Student.parse(result));
            }
            return list;
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new SQLException("desc", e);
        } finally {
            disconnect();
        }
    }
}
```

Клас “DAOStudentSubjectImpl”

```
package org.example.dao.implementations;

import org.apache.log4j.Logger;
import org.example.dao.connection.Oracle;
import org.example.dao.interfaces.DAOStudentSubject;
import org.example.entities.Student;
import org.example.entities.StudentSubject;
import org.example.entities.Subject;
import org.example.tools.custom.exceptions.WrongEntityIdException;
import org.springframework.stereotype.Repository;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

import static org.example.tools.strings.Query.*;

@Repository
public class DAOStudentSubjectImpl extends Oracle implements
DAOStudentSubject {

    private static final Logger logger =
Logger.getLogger(DAOStudentSubjectImpl.class);

    @Override
    public StudentSubject getStudentSubjectById(String id) throws
WrongEntityIdException {
        try {
            connect();
            statement = connection.prepareStatement(
                    STUDENT_SUBJECT_BY_ID.getQuery());

            statement.setInt(1, Integer.parseInt(id));
```

```
            result = statement.executeQuery();
            if(result.next()) {
                return StudentSubject.parse(result);
            } else {
                throw new WrongEntityIdException("desc ");
            }
        } catch (SQLException | WrongEntityIdException e) {
            e.printStackTrace();
            throw new WrongEntityIdException("desc ", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public List<StudentSubject> getStudentSubjectsByStudentId(String
studentId) throws WrongEntityIdException {
        try {
            connect();
            List<StudentSubject> list = new ArrayList<>();
            statement =
connection.prepareStatement(STUDENT_SUBJECT_LIST_BY_STUDENT_ID.getQuery());
            statement.setInt(1, Integer.parseInt(studentId));
            result = statement.executeQuery();
            while (result.next()) {
                list.add(StudentSubject.parse(result));
            }
            return list;
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new WrongEntityIdException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public List<StudentSubject> getStudentSubjectsBySubjectId(String
subjectId) throws WrongEntityIdException {
        try {
            connect();
            List<StudentSubject> list = new ArrayList<>();
            statement =
connection.prepareStatement(STUDENT_SUBJECT_LIST_BY_SUBJECT_ID.getQuery());
            statement.setInt(1, Integer.parseInt(subjectId));
            result = statement.executeQuery();
            while (result.next()) {
                list.add(StudentSubject.parse(result));
            }
            return list;
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new WrongEntityIdException("desc", e);
        } finally {
            disconnect();
        }
    }
```

```
    @Override
    public StudentSubject getStudentSubjectBySubjectIdAndStudentId(String
subjectId, String studentId) throws WrongEntityIdException {
        try {
            connect();
            statement = connection.prepareStatement(
                    STUDENT_SUBJECT_BY_IDS.getQuery());

            statement.setInt(1, Integer.parseInt(subjectId));
            statement.setInt(2, Integer.parseInt(studentId));

            result = statement.executeQuery();
            if(result.next()) {
                return StudentSubject.parse(result);
            } else {
                throw new WrongEntityIdException("desc ");
            }
        } catch (SQLException | WrongEntityIdException e) {
            e.printStackTrace();
            throw new WrongEntityIdException("desc ", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public HashMap<Student, StudentSubject> getStudentsInfoBySubjectId(String
subjectId) throws WrongEntityIdException {
        try {
            connect();
            HashMap<Student, StudentSubject> studentSubjectMap = new
HashMap<>();
            statement =
connection.prepareStatement(STUDENT_SUBJECT_MAP_BY_SUBJECT_ID.getQuery());
            statement.setInt(1, Integer.parseInt(subjectId));
            result = statement.executeQuery();
            while (result.next()) {
                studentSubjectMap.put(Student.parse(result),
StudentSubject.parse(result));
            }
            return studentSubjectMap;
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new WrongEntityIdException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public HashMap<Subject, StudentSubject> getSubjectsInfoByStudentId(String
studentId) throws WrongEntityIdException {
        try {
            connect();
            HashMap<Subject, StudentSubject> studentSubjectMap = new
HashMap<>();
            statement =
connection.prepareStatement(STUDENT_SUBJECT_MAP_BY_STUDENT_ID.getQuery());
```

```
            statement.setInt(1, Integer.parseInt(studentId));
            result = statement.executeQuery();
            while (result.next()) {
                studentSubjectMap.put(Subject.parse(result),
StudentSubject.parse(result));
            }
            return studentSubjectMap;
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new WrongEntityIdException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public void addStudentSubject(StudentSubject studentSubject) throws
SQLException {
        try {
            connect();
            statement =
connection.prepareStatement(ADD_STUDENT_SUBJECT.getQuery());
            statement.setInt(1, studentSubject.getTotalGrade());
            statement.setInt(2,
Integer.parseInt(studentSubject.getStudentId()));
            statement.setInt(3,
Integer.parseInt(studentSubject.getSubjectId()));

            statement.execute();
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new SQLException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public void updateStudentSubject(StudentSubject studentSubject) throws
SQLException {
        try {
            connect();
            statement =
connection.prepareStatement(UPDATE_STUDENT_SUBJECT.getQuery());

            statement.setInt(1, studentSubject.getTotalGrade());
            statement.setInt(2,
Integer.parseInt(studentSubject.getStudentId()));
            statement.setInt(3,
Integer.parseInt(studentSubject.getSubjectId()));
            statement.setInt(4,
Integer.parseInt(studentSubject.getStudentSubjectId()));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new SQLException("desc", e);
        } finally {
            disconnect();
```

```
        }
    }

    @Override
    public void actualizeTotalGrade(String studentSubjectId) throws
SQLException {
        try {
            connect();
            statement =
connection.prepareStatement(ACTUALIZE_TOTAL_GRADE.getQuery());

            statement.setInt(1, Integer.parseInt(studentSubjectId));
            statement.setInt(2, Integer.parseInt(studentSubjectId));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new SQLException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public void deleteStudentSubject(String studentSubjectId) {
        try {
            connect();
            statement =
connection.prepareStatement(DELETE_STUDENT_SUBJECT_BY_ID.getQuery());
            statement.setInt(1, Integer.parseInt(studentSubjectId));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc");
        } finally {
            disconnect();
        }
    }

    @Override
    public void deleteStudentSubject(String studentId, String subjectId) {
        try {
            connect();
            statement =
connection.prepareStatement(DELETE_STUDENT_SUBJECT_BY_IDS.getQuery());
            statement.setInt(1, Integer.parseInt(studentId));
            statement.setInt(2, Integer.parseInt(subjectId));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc");
        } finally {
            disconnect();
        }
    }

    @Override
    public void deleteStudentSubjectsBySubjectId(String subjectId) {
        try {
            connect();
            statement =
```

```
connection.prepareStatement(DELETE_STUDENT_SUBJECT_BY_SUBJECT_ID.getQuery());
            statement.setInt(1, Integer.parseInt(subjectId));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc");
        } finally {
            disconnect();
        }
    }

    @Override
    public void deleteStudentSubjectsByStudentId(String studentId) {
        try {
            connect();
            statement =
connection.prepareStatement(DELETE_STUDENT_SUBJECT_BY_STUDENT_ID.getQuery());
            statement.setInt(1, Integer.parseInt(studentId));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc");
        } finally {
            disconnect();
        }
    }

    @Override
    public String getLastStudentSubjectId() throws WrongEntityIdException {
        try {
            connect();
            statement = connection.prepareStatement(
                    GET_LAST_STUDENT_SUBJECT_ID.getQuery());

            result = statement.executeQuery();
            if(result.next()) {
                return Integer.toString(result.getInt("last_id"));
            } else {
                throw new WrongEntityIdException("desc ");
            }
        } catch (SQLException | WrongEntityIdException e) {
            e.printStackTrace();
            throw new WrongEntityIdException("desc ", e);
        } finally {
            disconnect();
        }
    }
}
```

Клас “DAOStudentTaskImpl”

```
package org.example.dao.implementations;

import org.apache.log4j.Logger;
import org.example.dao.connection.Oracle;
import org.example.dao.interfaces.DAOGroup;
import org.example.dao.interfaces.DAOStudentTask;
import org.example.dao.interfaces.DAOTask;
```

```
import org.example.dao.interfaces.DAOTeacher;
import org.example.entities.*;
import org.example.tools.custom.exceptions.WrongEntityIdException;
import org.springframework.stereotype.Repository;

import java.sql.SQLException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

import static org.example.tools.strings.Query.*;

@Repository
public class DAOStudentTaskImpl extends Oracle implements DAOStudentTask {

    private static final Logger logger =
Logger.getLogger(DAOStudentTaskImpl.class);

    @Override
    public StudentTask getStudentTaskById(String studentSubjectId, String
taskId) throws WrongEntityIdException {
        try {
            connect();
            statement = connection.prepareStatement(
                    STUDENT_TASK_BY_ID.getQuery());

            statement.setInt(1, Integer.parseInt(studentSubjectId));
            statement.setInt(2, Integer.parseInt(taskId));

            result = statement.executeQuery();
            if(result.next()) {
                return StudentTask.parse(result);
            } else {
                throw new WrongEntityIdException("desc ");
            }
        } catch (SQLException | WrongEntityIdException e) {
            e.printStackTrace();
            throw new WrongEntityIdException("desc ", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public List<StudentTask> getStudentTaskByStudentSubjectId(String
studentSubjectId) throws WrongEntityIdException {
        try {
            connect();
            List<StudentTask> list = new ArrayList<>();
            statement =
connection.prepareStatement(STUDENT_TASKS_BY_STUDENT_SUBJECT_ID.getQuery());

            statement.setInt(1, Integer.parseInt(studentSubjectId));

            result = statement.executeQuery();
            while (result.next()) {
                list.add(StudentTask.parse(result));
            }
```

```
            return list;
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new WrongEntityIdException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public void addStudentTask(StudentTask studentTask) throws SQLException {
        try {
            connect();
            statement =
connection.prepareStatement(ADD_STUDENT_TASK.getQuery());
            statement.setInt(1,
Integer.parseInt(studentTask.getStudentSubjectId()));
            statement.setInt(2, Integer.parseInt(studentTask.getTaskId()));
            statement.setInt(3,
Integer.parseInt(studentTask.getSubjectId()));
            statement.setInt(4, studentTask.getGrade());
            statement.setInt(5,
Integer.parseInt(studentTask.getByTeacherId()));

            statement.execute();
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new SQLException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public void updateStudentTask(StudentTask studentTask) {

    }

    @Override
    public void updateStudentTaskGrade(String taskId, String
studentSubjectId, int newGrade, String teacherId) throws SQLException {
        try {
            connect();
            statement =
connection.prepareStatement(UPDATE_STUDENT_TASK_GRADE.getQuery());

            statement.setInt(1, newGrade);
            statement.setInt(2,  Integer.parseInt(teacherId));
            statement.setInt(3, Integer.parseInt(studentSubjectId));
            statement.setInt(4, Integer.parseInt(taskId));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new SQLException("desc", e);
        } finally {
            disconnect();
        }
    }
```

```
    @Override
    public void deleteStudentTask(String taskId, String studentTaskId) {
        try {
            connect();
            statement =
connection.prepareStatement(DELETE_STUDENT_TASK_BY_ID.getQuery());
            statement.setInt(1, Integer.parseInt(studentTaskId));
            statement.setInt(2, Integer.parseInt(taskId));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc");
        } finally {
            disconnect();
        }
    }

    @Override
    public void deleteStudentTasksBySubjectId(String subjectId) {
        try {
            connect();
            statement =
connection.prepareStatement(DELETE_STUDENT_TASK_BY_SUBJECT_ID.getQuery());
            statement.setInt(1, Integer.parseInt(subjectId));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc");
        } finally {
            disconnect();
        }
    }

    @Override
    public void deleteStudentTasksByStudentSubjectId(String studentSubjectId)
{
        try {
            connect();
            statement =
connection.prepareStatement(DELETE_STUDENT_TASK_BY_STUDENT_SUBJECT_ID.getQuer
y());
            statement.setInt(1, Integer.parseInt(studentSubjectId));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc");
        } finally {
            disconnect();
        }
    }

    @Override
    public void deleteStudentTasksByTaskId(String taskId) {
        try {
            connect();
            statement =
connection.prepareStatement(DELETE_STUDENT_TASK_BY_TASK_ID.getQuery());
            statement.setInt(1, Integer.parseInt(taskId));
            statement.execute();
        } catch (SQLException e) {
```

```
            logger.info("desc");
        } finally {
            disconnect();
        }
    }

    @Override
    public void deleteStudentTasksByStudentId(String studentId) {
        try {
            connect();
            statement =
connection.prepareStatement(DELETE_STUDENT_TASK_BY_STUDENT_ID.getQuery());
            statement.setInt(1, Integer.parseInt(studentId));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc");
        } finally {
            disconnect();
        }
    }

    @Override
    public void deleteByTeacherId(String teacherId) {
        try {
            connect();
            statement =
connection.prepareStatement(DELETE_STUDENT_TASK_BY_TEACHER.getQuery());

            statement.setInt(2,  Integer.parseInt(teacherId));
            statement.setInt(2,  Integer.parseInt(teacherId));

            statement.execute();
        } catch (SQLException e) {
            logger.info("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public HashMap<Task, StudentTask>
getStudentTasksInfoByStudentSubjectId(String studentSubjectId) throws
WrongEntityIdException {
        try {
            connect();
            HashMap<Task, StudentTask> studentTaskMap = new HashMap<>();
            statement =
connection.prepareStatement(STUDENT_TASK_MAP_BY_STUDENT_SUBJECT_ID.getQuery()
);
            statement.setInt(1, Integer.parseInt(studentSubjectId));
            result = statement.executeQuery();
            while (result.next()) {
                studentTaskMap.put(Task.parse(result),
StudentTask.parse(result));
            }
            return studentTaskMap;
        } catch (SQLException e) {
            logger.info("desc", e);
```

```
            throw new WrongEntityIdException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public StudentTaskInfoSet getStudentTaskInfoSet(StudentTask studentTask)
throws WrongEntityIdException {
        DAOTask daoTask = new DAOTaskImpl();
        DAOTeacher daoTeacher = new DAOTeacherImpl();
        Task task = daoTask.getTaskById(studentTask.getTaskId());
        Teacher teacher = null;
        if (studentTask.getByTeacherId() != null) {
            teacher =
daoTeacher.getTeacherById(studentTask.getByTeacherId());
        }
        return new StudentTaskInfoSet(studentTask, task, teacher);
    }

    @Override
    public List<StudentTaskInfoSet> getStudentTaskInfoSetList(String
studentSubjectId) throws WrongEntityIdException {
        List<StudentTask> studentTaskList =
getStudentTaskByStudentSubjectId(studentSubjectId);
        List<StudentTaskInfoSet> studentTaskInfoSetList =  new ArrayList<>();
        for (StudentTask s: studentTaskList) {
            studentTaskInfoSetList.add(getStudentTaskInfoSet(s));
        }
        return studentTaskInfoSetList;
    }
}
```

Клас “DAOSubjectImpl”

```
package org.example.dao.implementations;

import org.apache.log4j.Logger;
import org.example.dao.connection.Oracle;
import org.example.dao.interfaces.DAOSubject;
import org.example.entities.Student;
import org.example.entities.Subject;
import org.example.entities.Teacher;
import org.example.tools.custom.exceptions.WrongEntityIdException;
import org.springframework.stereotype.Repository;

import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import static org.example.tools.strings.Query.*;

@Repository
public class DAOSubjectImpl extends Oracle implements DAOSubject {
```

```
    private static final Logger logger =
Logger.getLogger(DAOSubjectImpl.class);

    @Override
    public Subject getSubjectById(String id) throws WrongEntityIdException {
        try {
            connect();
            statement = connection.prepareStatement(
                    SUBJECT_BY_ID.getQuery());

            statement.setInt(1, Integer.parseInt(id));

            result = statement.executeQuery();
            if(result.next()) {
                return Subject.parse(result);
            } else {
                throw new WrongEntityIdException("desc ");
            }
        } catch (SQLException | WrongEntityIdException e) {
            e.printStackTrace();
            throw new WrongEntityIdException("desc ", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public List<Student> getStudentsBySubjectId(String subjectId) throws
WrongEntityIdException {
        try {
            connect();
            List<Student> list = new ArrayList<>();
            statement =
connection.prepareStatement(STUDENTS_BY_SUBJECT_ID.getQuery());

            statement.setInt(1, Integer.parseInt(subjectId));

            result = statement.executeQuery();
            while (result.next()) {
                list.add(Student.parse(result));
            }
            return list;
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new WrongEntityIdException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public List<Subject> getSubjectsByStudentId(String studentId) throws
WrongEntityIdException {
        try {
            connect();
            List<Subject> list = new ArrayList<>();
            statement =
connection.prepareStatement(SUBJECTS_BY_STUDENT_ID.getQuery());
```

```
            statement.setInt(1, Integer.parseInt(studentId));

            result = statement.executeQuery();
            while (result.next()) {
                list.add(Subject.parse(result));
            }
            return list;
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new WrongEntityIdException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public List<Subject> getSubjectsByTeacherId(String teacherId) throws
WrongEntityIdException {
        try {
            connect();
            List<Subject> list = new ArrayList<>();
            statement =
connection.prepareStatement(SUBJECT_LIST_BY_TEACHER_ID.getQuery());
            statement.setInt(1, Integer.parseInt(teacherId));
            result = statement.executeQuery();
            while (result.next()) {
                list.add(Subject.parse(result));
            }
            return list;
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new WrongEntityIdException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public void addSubject(Subject subject) throws SQLException {
        try {
            connect();
            statement = connection.prepareStatement(ADD_SUBJECT.getQuery());
            statement.setString(1, subject.getSubjectName());
            statement.setInt(2, subject.getMaxGrade());
            statement.setInt(3, subject.getPassProcGradeP());
            statement.setString(4, subject.getSubjectDescription());
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new SQLException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public void updateSubject(Subject subject) throws SQLException {
```

```
        try {
            connect();
            statement =
connection.prepareStatement(UPDATE_SUBJECT.getQuery());

            statement.setString(1, subject.getSubjectName());
            statement.setInt(2, subject.getMaxGrade());
            statement.setInt(3, subject.getPassProcGradeP());
            statement.setString(4, subject.getSubjectDescription());
            statement.setInt(5, Integer.parseInt(subject.getSubjectId()));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new SQLException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public void actualizeMaxGrade(String subjectId) throws SQLException {
        try {
            connect();
            statement =
connection.prepareStatement(ACTUALIZE_MAX_GRADE.getQuery());

            statement.setInt(1, Integer.parseInt(subjectId));
            statement.setInt(2, Integer.parseInt(subjectId));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new SQLException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public void deleteSubject(String subjectId) {
        try {
            connect();
            statement =
connection.prepareStatement(DELETE_SUBJECT_BY_ID.getQuery());
            statement.setInt(1, Integer.parseInt(subjectId));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc");
        } finally {
            disconnect();
        }
    }

    @Override
    public List<Subject> getAllSubjects() throws WrongEntityIdException {
        try {
            connect();
            List<Subject> list = new ArrayList<>();
            statement = connection.prepareStatement(ALL_SUBJECTS.getQuery());
```

```
            result = statement.executeQuery();
            while (result.next()) {
                list.add(Subject.parse(result));
            }
            return list;
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new WrongEntityIdException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public List<Subject> searchSubjectsByName(String text) throws
SQLException, WrongEntityIdException {
        if (text == null || text.isEmpty()) {
            return Collections.emptyList();
        } else if (text.equals("*")) {
            return getAllSubjects();
        }

        try {
            connect();
            List<Subject> list = new ArrayList<>();
            statement =
connection.prepareStatement(SEARCH_BY_SUBJECT_NAME.getQuery());
            statement.setString(1, text);
            result = statement.executeQuery();
            while (result.next()) {
                list.add(Subject.parse(result));
            }
            return list;
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new SQLException("desc", e);
        } finally {
            disconnect();
        }
    }
}
```

Клас “DAOTaskImpl”

```
package org.example.dao.implementations;

import org.apache.log4j.Logger;
import org.example.dao.connection.Oracle;
import org.example.dao.interfaces.DAOTask;
import org.example.entities.Student;
import org.example.entities.Task;
import org.example.tools.custom.exceptions.WrongEntityIdException;
import org.springframework.stereotype.Repository;

import java.sql.SQLException;
import java.util.ArrayList;
```

```
import java.util.List;

import static org.example.tools.strings.Query.*;

@Repository
public class DAOTaskImpl extends Oracle implements DAOTask {

    private static final Logger logger = Logger.getLogger(DAOTaskImpl.class);

    @Override
    public Task getTaskById(String taskId) throws WrongEntityIdException {
        try {
            connect();
            statement = connection.prepareStatement(
                    TASK_BY_ID.getQuery());
            statement.setInt(1, Integer.parseInt(taskId));

            result = statement.executeQuery();
            if(result.next()) {
                return Task.parse(result);
            } else {
                throw new WrongEntityIdException("desc ");
            }
        } catch (SQLException | WrongEntityIdException e) {
            e.printStackTrace();
            throw new WrongEntityIdException("desc ", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public void addTask(Task task) throws SQLException {
        try {
            connect();
            statement = connection.prepareStatement(ADD_TASK.getQuery());
            statement.setInt(1, Integer.parseInt(task.getSubjectId()));
            statement.setString(2, task.getTaskName());
            statement.setInt(3, task.getMaxGrade());
            statement.setString(4, task.getTaskDescription());

            statement.execute();
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new SQLException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public void updateTask(Task task) {

    }

    @Override
    public void updateTaskName(String taskId, String newName) throws
SQLException {
```

```
        try {
            connect();
            statement =
connection.prepareStatement(UPDATE_TASK_NAME.getQuery());

            statement.setInt(1, Integer.parseInt(newName));
            statement.setInt(2, Integer.parseInt(taskId));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new SQLException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public void updateTaskMaxGrade(String taskId, int newMaxGrade) throws
SQLException {
        try {
            connect();
            statement =
connection.prepareStatement(UPDATE_TASK_MAX_GRADE.getQuery());

            statement.setInt(1, newMaxGrade);
            statement.setInt(2, Integer.parseInt(taskId));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new SQLException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public void updateTaskNameAndGrade(String taskId, String newTaskName, int
newMaxGrade, String taskDescription) throws SQLException {
        try {
            connect();
            statement = connection.prepareStatement(UPDATE_TASK.getQuery());

            statement.setInt(1, newMaxGrade);
            statement.setString(2, newTaskName);
            statement.setString(3, taskDescription);
            statement.setInt(4, Integer.parseInt(taskId));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new SQLException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public void deleteTask(String taskId) {
        try {
```

```
            connect();
            statement =
connection.prepareStatement(DELETE_TASK_BY_ID.getQuery());
            statement.setInt(1, Integer.parseInt(taskId));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc");
        } finally {
            disconnect();
        }
    }

    @Override
    public void deleteTasksBySubjectId(String subjectId) {
        try {
            connect();
            statement =
connection.prepareStatement(DELETE_TASK_BY_SUBJECT_ID.getQuery());
            statement.setInt(1, Integer.parseInt(subjectId));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc");
        } finally {
            disconnect();
        }
    }

    @Override
    public List<Task> getTaskListBySubjectId(String subjectId) throws
WrongEntityIdException {
        try {
            connect();
            java.util.List<Task> list = new ArrayList<>();
            statement =
connection.prepareStatement(TASKS_BY_SUBJECT_ID.getQuery());
            statement.setInt(1, Integer.parseInt(subjectId));
            result = statement.executeQuery();
            while (result.next()) {
                list.add(Task.parse(result));
            }
            return list;
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new WrongEntityIdException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public String getLastTaskId() throws WrongEntityIdException {
        try {
            connect();
            statement = connection.prepareStatement(
                    GET_LAST_TASK_ID.getQuery());

            result = statement.executeQuery();
            if(result.next()) {
```

```
                return Integer.toString(result.getInt("last_id"));
            } else {
                throw new WrongEntityIdException("desc ");
            }
        } catch (SQLException | WrongEntityIdException e) {
            e.printStackTrace();
            throw new WrongEntityIdException("desc ", e);
        } finally {
            disconnect();
        }
    }
}
```

Клас “DAOTeacherImpl”

```
package org.example.dao.implementations;

import org.apache.log4j.Logger;
import org.example.dao.connection.Oracle;
import org.example.dao.interfaces.DAOTeacher;
import org.example.entities.Teacher;
import org.example.tools.custom.exceptions.WrongEntityIdException;
import org.example.tools.custom.exceptions.WrongLoginDataException;
import org.springframework.stereotype.Repository;

import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import static org.example.tools.strings.Query.*;

@Repository
public class DAOTeacherImpl extends Oracle implements DAOTeacher {

    private static final Logger logger =
Logger.getLogger(DAOTeacherImpl.class);

    @Override
    public Teacher getTeacherById(String id) throws WrongEntityIdException {
        try {
            connect();
            statement = connection.prepareStatement(
                    TEACHER_BY_ID.getQuery());

            statement.setInt(1, Integer.parseInt(id));

            result = statement.executeQuery();
            if(result.next()) {
                return Teacher.parse(result);
            } else {
                throw new WrongEntityIdException("desc ");
            }
        } catch (SQLException | WrongEntityIdException e) {
            e.printStackTrace();
            throw new WrongEntityIdException("desc ", e);
        } finally {
```

```
            disconnect();
        }
    }

    @Override
    public boolean isExistTeacherByEmail(String email) throws SQLException {
        try {
            connect();
            statement = connection.prepareStatement(
                    EXIST_TEACHER_BY_EMAIL.getQuery());

            statement.setString(1, email);
            result = statement.executeQuery();
            return result.next();

        } catch (SQLException e) {
            e.printStackTrace();
            throw new SQLException("desc ", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public List<Teacher> getTeachersBySubjectId(String subjectId) throws
WrongEntityIdException {
        try {
            connect();
            List<Teacher> list = new ArrayList<>();
            statement =
connection.prepareStatement(TEACHER_LIST_BY_SUBJECT_ID.getQuery());
            statement.setInt(1, Integer.parseInt(subjectId));
            result = statement.executeQuery();
            while (result.next()) {
                list.add(Teacher.parse(result));
            }
            return list;
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new WrongEntityIdException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public Teacher getTeacherByEmailAndPassword(String email, String
password) throws WrongLoginDataException {
        try {
            connect();
            statement = connection.prepareStatement(
                    TEACHER_BY_EMAIL_AND_PASSWORD.getQuery());

            statement.setString(1, email);
            statement.setString(2, password);

            result = statement.executeQuery();
            if(result.next()) {
```

```
                return Teacher.parse(result);
            } else {
                throw new WrongLoginDataException("desc ");
            }
        } catch (SQLException | WrongLoginDataException e) {
            e.printStackTrace();
            throw new WrongLoginDataException("desc ", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public void changeTeacherPassword(String teacherId, String password)
throws SQLException {
        try {
            connect();
            statement =
connection.prepareStatement(CHANGE_TEACHER_PASSWORD.getQuery());

            statement.setString(1, password);
            statement.setInt(2, Integer.parseInt(teacherId));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new SQLException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public void addTeacher(Teacher teacher) throws SQLException {
        try {
            connect();
            statement = connection.prepareStatement(ADD_TEACHER.getQuery());
            statement.setString(1, teacher.getEmail());
            statement.setString(2, teacher.getFirstName());
            statement.setString(3, teacher.getLastName());
            statement.setString(4, teacher.getPassword());
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new SQLException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public void updateTeacher(Teacher teacher) throws SQLException {
        try {
            connect();
            statement =
connection.prepareStatement(UPDATE_TEACHER.getQuery());

            statement.setString(1, teacher.getEmail());
            statement.setString(2, teacher.getFirstName());
```

```
            statement.setString(3, teacher.getLastName());
            statement.setInt(4, Integer.parseInt(teacher.getTeacherId()));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new SQLException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public void deleteTeacher(String teacherId) {
        try {
            connect();
            statement =
connection.prepareStatement(DELETE_TEACHER_BY_ID.getQuery());
            statement.setInt(1, Integer.parseInt(teacherId));
            statement.execute();
        } catch (SQLException e) {
            logger.info("desc");
        } finally {
            disconnect();
        }
    }

    @Override
    public List<Teacher> getAllTeachers() throws WrongEntityIdException {
        try {
            connect();
            List<Teacher> list = new ArrayList<>();
            statement = connection.prepareStatement(ALL_TEACHERS.getQuery());
            result = statement.executeQuery();
            while (result.next()) {
                list.add(Teacher.parse(result));
            }
            return list;
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new WrongEntityIdException("desc", e);
        } finally {
            disconnect();
        }
    }
}
```

Клас “DAOTeacherSubjectImpl”

```
package org.example.dao.implementations;

import org.apache.log4j.Logger;
import org.example.dao.connection.Oracle;
import org.example.dao.interfaces.DAOTeacherSubject;
import org.example.entities.Subject;
import org.example.entities.TeacherSubject;
import org.springframework.stereotype.Repository;
```

```
import java.sql.SQLException;

import static org.example.tools.strings.Query.*;

@Repository
public class DAOTeacherSubjectImpl extends Oracle implements
DAOTeacherSubject {

    private static final Logger logger =
Logger.getLogger(DAOStudentSubjectImpl.class);

    @Override
    public void addTeacherSubject(TeacherSubject teacherSubject) throws
SQLException {
        try {
            connect();
            statement =
connection.prepareStatement(ADD_TEACHER_SUBJECT.getQuery());
            statement.setInt(1,
Integer.parseInt(teacherSubject.getSubjectId()));
            statement.setInt(2,
Integer.parseInt(teacherSubject.getTeacherId()));

            statement.execute();
        } catch (SQLException e) {
            logger.info("desc", e);
            throw new SQLException("desc", e);
        } finally {
            disconnect();
        }
    }

    @Override
    public void deleteTeacherSubject(TeacherSubject teacherSubject) {
        try {
            connect();
            statement =
connection.prepareStatement(DELETE_TEACHER_SUBJECT_BY_ID.getQuery());
            statement.setInt(1,
Integer.parseInt(teacherSubject.getSubjectId()));
            statement.setInt(2,
Integer.parseInt(teacherSubject.getTeacherId()));

            statement.execute();
        } catch (SQLException e) {
            logger.info("desc");
        } finally {
            disconnect();
        }
    }

    @Override
    public void deleteTeacherSubjectsBySubjectId(String subjectId) {
        try {
            connect();
            statement =
connection.prepareStatement(DELETE_TEACHER_SUBJECT_BY_SUBJECT_ID.getQuery());
```

```
            statement.setInt(1, Integer.parseInt(subjectId));

            statement.execute();
        } catch (SQLException e) {
            logger.info("desc");
        } finally {
            disconnect();
        }
    }

    @Override
    public void deleteTeacherSubjectsByTeacherId(String teacherId) {
        try {
            connect();
            statement =
connection.prepareStatement(DELETE_TEACHER_SUBJECT_BY_TEACHER_ID.getQuery());
            statement.setInt(1, Integer.parseInt(teacherId));

            statement.execute();
        } catch (SQLException e) {
            logger.info("desc");
        } finally {
            disconnect();
        }
    }
}
```

Клас “User”

```
package org.example.entities;

public abstract class User {
    public abstract String getEmail();
    public abstract String getFirstName();
    public abstract String getLastName();
    public String getFullName() {
        return getFirstName() + " " + getLastName();
    }
    public String getFullNameWithEmail() {
        return getFirstName() +
                " " + getLastName() +
                " (" + getEmail() + ")";
    }
    public abstract String getUserId();
    public abstract String getRole();
}
```

Клас “Group”

```
package org.example.entities;

import java.sql.ResultSet;
import java.sql.SQLException;
```

```
public class Group{
    private String groupId;
    private String groupName;
    private String groupDescription;

    public Group() {

    }

    public Group(String groupId, String groupName,  String groupDescription)
{
        this.groupId = groupId;
        this.groupName = groupName;
        this.groupDescription = groupDescription;
    }

    public String getGroupDescription() {
        return groupDescription;
    }

    public void setGroupDescription(String groupDescription) {
        this.groupDescription = groupDescription;
    }

    public String getGroupId() {
        return groupId;
    }

    public void setGroupId(String groupId) {
        this.groupId = groupId;
    }

    public String getGroupName() {
        return groupName;
    }

    public void setGroupName(String groupName) {
        this.groupName = groupName;
    }

    public  static Group parse(ResultSet result) throws SQLException {
        return new Group(
                result.getString("group_id"),
                result.getString("group_name"),
                result.getString("group_description")
        );
    }

    @Override
    public String toString() {
        return "Group{" +
                "groupId='" + groupId + '\'' +
                ", groupName='" + groupName + '\'' +
                ", groupDescription='" + groupDescription + '\'' +
                '}';
    }
}
```

Клас “Student”

```
package org.example.entities;

import org.example.tools.strings.Role;

import java.sql.ResultSet;
import java.sql.SQLException;


public class Student extends User {
    private String studentId;
    private String email;
    private String firstName;
    private String lastName;
    private String password;
    private String headman;
    private String groupId;

    public Student() {

    }

    public Student(String studentId, String email, String firstName, String
lastName, String password, String headman, String groupId) {
        this.studentId = studentId;
        this.email = email;
        this.firstName = firstName;
        this.lastName = lastName;
        this.password = password;
        this.headman = headman;
        this.groupId = groupId;
    }

    public String getStudentId() {
        return studentId;
    }

    public void setStudentId(String studentId) {
        this.studentId = studentId;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
```

```
}

public String getLastName() {
    return lastName;
}

@Override
public String getUserId() {
    return studentId;
}

@Override
public String getRole() {
    return Role.STUDENT.getRole();
}

public void setLastName(String lastName) {
    this.lastName = lastName;
}

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}

public String getHeadman() {
    return headman;
}

public void setHeadman(String headman) {
    this.headman = headman;
}

public String getGroupId() {
    return groupId;
}

public void setGroupId(String groupId) {
    this.groupId = groupId;
}

public static Student parse(ResultSet result) throws SQLException {
    return new Student(
            result.getString("student_id"),
            result.getString("email"),
            result.getString("first_name"),
            result.getString("last_name"),
            result.getString("password"),
            result.getString("headman"),
            result.getString("group_id")
    );
}

@Override
public String toString() {
```

```
        return "Student{" +
                "studentId='" + studentId + '\'' +
                ", email='" + email + '\'' +
                ", fistName='" + firstName + '\'' +
                ", lastName='" + lastName + '\'' +
                ", password='" + password + '\'' +
                ", headman='" + headman + '\'' +
                ", groupId='" + groupId + '\'' +
                '}';
    }
}
```

Клас “StudentInfoSet”

```
package org.example.entities;

public class StudentInfoSet {
    Student student;
    Group group;
    Student headman;

    public StudentInfoSet(Student student, Group group, Student headman) {
        this.student = student;
        this.group = group;
        this.headman = headman;
    }

    public Student getStudent() {
        return student;
    }

    public void setStudent(Student student) {
        this.student = student;
    }

    public Group getGroup() {
        return group;
    }

    public void setGroup(Group group) {
        this.group = group;
    }

    public Student getHeadman() {
        return headman;
    }

    public void setHeadman(Student headman) {
        this.headman = headman;
    }
}
```

Клас “StudentSubject”

```
package org.example.entities;

import java.sql.ResultSet;
import java.sql.SQLException;


public class StudentSubject {
    private String studentSubjectId;
    private String studentId;
    private String subjectId;
    private int totalGrade;

    public StudentSubject() {

    }

    public StudentSubject(String studentSubjectId, String studentId, String
subjectId, int totalGrade) {
        this.studentSubjectId = studentSubjectId;
        this.studentId = studentId;
        this.subjectId = subjectId;
        this.totalGrade = totalGrade;
    }

    public String getStudentId() {
        return studentId;
    }

    public void setStudentId(String studentId) {
        this.studentId = studentId;
    }

    public String getSubjectId() {
        return subjectId;
    }

    public void setSubjectId(String subjectId) {
        this.subjectId = subjectId;
    }

    public int getTotalGrade() {
        return totalGrade;
    }

    public void setTotalGrade(int totalGrade) {
        this.totalGrade = totalGrade;
    }

    public String getStudentSubjectId() {
        return studentSubjectId;
    }

    public void setStudentSubjectId(String studentSubjectId) {
        this.studentSubjectId = studentSubjectId;
    }
```

```
    public static StudentSubject parse(ResultSet result) throws SQLException
{
        return new StudentSubject(
                result.getString("student_subject_id"),
                result.getString("student_id"),
                result.getString("subject_id"),
                result.getInt("total_grade")
        );
    }

    @Override
    public String toString() {
        return "StudentSubject{" +
                "studentSubjectId='" + studentSubjectId + '\'' +
                ", studentId='" + studentId + '\'' +
                ", subjectId='" + subjectId + '\'' +
                ", totalGrade=" + totalGrade +
                '}';
    }
}
```

Клас “StudentTask”

```
package org.example.entities;

import java.sql.ResultSet;
import java.sql.SQLException;


public class StudentTask {
    private String taskId;
    private String SubjectId;
    private String studentSubjectId;
    private int grade;
    private String byTeacherId;

    public StudentTask() {

    }

    public StudentTask(String taskId, String subjectId, String
studentSubjectId, int grade, String byTeacherId) {
        this.taskId = taskId;
        this.SubjectId = subjectId;
        this.studentSubjectId = studentSubjectId;
        this.grade = grade;
        this.byTeacherId = byTeacherId;
    }

    public String getTaskId() {
        return taskId;
    }

    public void setTaskId(String taskId) {
        this.taskId = taskId;
    }
```

```
    public String getSubjectId() {
        return SubjectId;
    }

    public void setSubjectId(String subjectId) {
        SubjectId = subjectId;
    }

    public String getStudentSubjectId() {
        return studentSubjectId;
    }

    public void setStudentSubjectId(String studentSubjectId) {
        this.studentSubjectId = studentSubjectId;
    }

    public int getGrade() {
        return grade;
    }

    public void setGrade(int grade) {
        this.grade = grade;
    }

    public String getByTeacherId() {
        return byTeacherId;
    }

    public void setByTeacherId(String byTeacherId) {
        this.byTeacherId = byTeacherId;
    }

    public static StudentTask parse(ResultSet result) throws SQLException {
        return new StudentTask(
                result.getString("task_id"),
                result.getString("subject_id"),
                result.getString("student_subject_id"),
                result.getInt("grade"),
                result.getString("by_teacher_id")
        );
    }

    @Override
    public String toString() {
        return "Task{" +
                "taskId='" + taskId + '\'' +
                ", SubjectId='" + SubjectId + '\'' +
                ", studentSubjectId='" + studentSubjectId + '\'' +
                ", grade=" + grade +
                '}';
    }
}
```

Клас “StudentTaskInfoSet”

```
package org.example.entities;

public class StudentTaskInfoSet {
    StudentTask studentTask;
    Task task;
    Teacher teacher;

    public StudentTaskInfoSet(StudentTask studentTask, Task task, Teacher
teacher) {
        this.studentTask = studentTask;
        this.task = task;
        this.teacher = teacher;
    }

    public StudentTask getStudentTask() {
        return studentTask;
    }

    public void setStudentTask(StudentTask studentTask) {
        this.studentTask = studentTask;
    }

    public Task getTask() {
        return task;
    }

    public void setTask(Task task) {
        this.task = task;
    }

    public Teacher getTeacher() {
        return teacher;
    }

    public void setTeacher(Teacher teacher) {
        this.teacher = teacher;
    }
}
```

Клас “Subject”

```
package org.example.entities;

import java.sql.ResultSet;
import java.sql.SQLException;


public class Subject {
    private String subjectId;
    private String subjectName;
    private int maxGrade;
    private int passProcGradeP;
    private String subjectDescription;
```

```
    public Subject() {

    }

    public Subject(String subject_id, String subjectName, int maxGrade, int
passProcGradeP, String subjectDescription) {
        this.subjectId = subject_id;
        this.subjectName = subjectName;
        this.maxGrade = maxGrade;
        this.passProcGradeP = passProcGradeP;
        this.subjectDescription = subjectDescription;
    }

    public String getSubjectDescription() {
        return subjectDescription;
    }

    public void setSubjectDescription(String subjectDescription) {
        this.subjectDescription = subjectDescription;
    }

    public String getSubjectId() {
        return subjectId;
    }

    public void setSubjectId(String subjectId) {
        this.subjectId = subjectId;
    }

    public String getSubjectName() {
        return subjectName;
    }

    public void setSubjectName(String subjectName) {
        this.subjectName = subjectName;
    }

    public int getMaxGrade() {
        return maxGrade;
    }

    public void setMaxGrade(int maxGrade) {
        this.maxGrade = maxGrade;
    }

    public int getPassProcGradeP() {
        return passProcGradeP;
    }

    public void setPassProcGradeP(int passProcGradeP) {
        this.passProcGradeP = passProcGradeP;
    }

    public static Subject parse(ResultSet result) throws SQLException {
        return new Subject(
                result.getString("subject_id"),
                result.getString("subject_name"),
                result.getInt("max_grade"),
```

```
                result.getInt("pass_proc_grade"),
                result.getString("subject_description")
        );
    }

    @Override
    public String toString() {
        return "Subject{" +
                "subjectId='" + subjectId + '\'' +
                ", subjectName='" + subjectName + '\'' +
                ", maxGrade=" + maxGrade +
                ", passProcGrade=" + passProcGradeP +
                ", subjectDescription='" + subjectDescription + '\'' +
                '}';
    }
}
```

Клас “Task”

```
package org.example.entities;

import java.sql.ResultSet;
import java.sql.SQLException;

public class Task {
    private String taskId;
    private String SubjectId;
    private String taskName;
    private int maxGrade;
    private String taskDescription;

    public Task() {

    }

    public Task(String taskId, String subjectId, String taskName, int
maxGrade, String taskDescription) {
        this.taskId = taskId;
        this.SubjectId = subjectId;
        this.taskName = taskName;
        this.maxGrade = maxGrade;
        this.taskDescription = taskDescription;
    }

    public String getTaskDescription() {
        return taskDescription;
    }

    public void setTaskDescription(String taskDescription) {
        this.taskDescription = taskDescription;
    }

    public String getTaskId() {
        return taskId;
    }
```

```
    public void setTaskId(String taskId) {
        this.taskId = taskId;
    }

    public String getSubjectId() {
        return SubjectId;
    }

    public void setSubjectId(String subjectId) {
        SubjectId = subjectId;
    }

    public String getTaskName() {
        return taskName;
    }

    public void setTaskName(String taskName) {
        this.taskName = taskName;
    }

    public int getMaxGrade() {
        return maxGrade;
    }

    public void setMaxGrade(int maxGrade) {
        this.maxGrade = maxGrade;
    }

    public static Task parse(ResultSet result) throws SQLException {
        return new Task(
                result.getString("task_id"),
                result.getString("subject_id"),
                result.getString("task_name"),
                result.getInt("max_grade"),
                result.getString("task_description")
        );
    }

    @Override
    public String toString() {
        return "Task{" +
                "taskId='" + taskId + '\'' +
                ", SubjectId='" + SubjectId + '\'' +
                ", taskName='" + taskName + '\'' +
                ", maxGrade=" + maxGrade +
                ", taskDescription='" + taskDescription + '\'' +
                '}';
    }
}
```

Клас “Teacher”

```
package org.example.entities;

import org.example.tools.strings.Role;
```

```
import java.sql.ResultSet;
import java.sql.SQLException;

public class Teacher extends User {
    private String teacherId;
    private String email;
    private String firstName;
    private String lastName;
    private String password;

    public Teacher() {

    }

    public Teacher(String teacherId, String email, String firstName, String
lastName, String password) {
        this.teacherId = teacherId;
        this.email = email;
        this.firstName = firstName;
        this.lastName = lastName;
        this.password = password;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getTeacherId() {
        return teacherId;
    }

    public void setTeacherId(String teacherId) {
        this.teacherId = teacherId;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    @Override
    public String getUserId() {
        return teacherId;
    }

    @Override
    public String getRole() {
```

```
        return Role.TEACHER.getRole();
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public static Teacher parse(ResultSet result) throws SQLException {
        return new Teacher(
                result.getString("teacher_id"),
                result.getString("email"),
                result.getString("first_name"),
                result.getString("last_name"),
                result.getString("password")
        );
    }

    @Override
    public String toString() {
        return "Teacher{" +
                "teacherId='" + teacherId + '\'' +
                ", email='" + email + '\'' +
                ", firstName='" + firstName + '\'' +
                ", lastName='" + lastName + '\'' +
                ", password='" + password + '\'' +
                '}';
    }
}
```

Клас “TeacherSubject”

```
package org.example.entities;

import java.sql.ResultSet;
import java.sql.SQLException;

public class TeacherSubject {
    private String subjectId;
    private String teacherId;

    public TeacherSubject() {

    }

    public TeacherSubject(String subjectId, String teacherId) {
        this.subjectId = subjectId;
        this.teacherId = teacherId;
    }
```

```
    public String getSubjectId() {
        return subjectId;
    }

    public void setSubjectId(String subjectId) {
        this.subjectId = subjectId;
    }

    public String getTeacherId() {
        return teacherId;
    }

    public void setTeacherId(String teacherId) {
        this.teacherId = teacherId;
    }

    public static TeacherSubject parse(ResultSet result) throws SQLException
{
        return new TeacherSubject(
                result.getString("subject_id"),
                result.getString("teacher_id")
        );
    }

    @Override
    public String toString() {
        return "TeacherSubject{" +
                "subjectId='" + subjectId + '\'' +
                ", teacherId='" + teacherId +
                '}';
    }
}
```