МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

" Інформаційна система аналізу наукових тенденцій учбових кафедр"

Завідувач випускаючої кафедри — **Довбиш А.С.**

Керівник роботи — **Берест О.Б.**

Студент гр. ІН–73-9 — **Теницький О.В.**

СУМИ 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую _____

Зав. кафедрою Довбиш А.С.

“_____” _____2021 р.

Завдання

до випускної роботи

Студента четвертого курсу, групи ІН-73/9 спеціальності “Комп’ютерні науки” денної форми навчання Теницького Олександра Володимировича.

Тема: “**Інформаційна система аналізу наукових тенденцій учбових кафедр**”

Затверджена наказом по СумДУ

№_____ від _____2021 р.

Зміст пояснювальної записки:1) аналіз предметної області, формулювання мети, постановка завдань; 2) дослідження методів та інструментів розробки інформаційних систем; 3) проектування архітектури майбутнього web-додатку; 4) розробка інформаційної системи аналізу наукових тенденцій учбових кафедр; 5) аналіз результатів.

Дата видачі завдання “_____”_____2021 г.

Керівник випускної роботи _____ Берест О.Б.

Завдання прийняв до виконання _____ Теницький О.В.

РЕФЕРАТ

Записка: 53 стор., 41 рис., 1 таблиця, 10 додатків, 15 джерел.

Об'єкт дослідження — Інформаційна система аналізу наукових тенденцій учбових кафедр.

Мета роботи — розробити інформаційну систему обліку студентів та аналізу наукових тенденцій учбових кафедр для автоматизації організаційних процесів роботи з особистими справами студентів.

Результати — проведено дослідження предметної області, визначено мету, задачі та актуальність роботи, виконано вибір методів вирішення поставленої задачі; розроблено та реалізовано архітектуру web-додатку; спроектовано базу даних.

ІНФОРМАЦІЙНА СИСТЕМА, ОБЛІК СТУДЕНТІВ, АНАЛІЗ НАУКОВИХ ТЕНДЕНЦІЙ, ФРЕЙМВОРК LARAVEL, HTML, CSS, BOOTSTRAP, JAVASCRIPT

ЗМІСТ

ВСТУП

У сучасному суспільстві провідну роль виконує інформація, користь від її використання та технології, за допомогою яких її можна отримати. Разом зі збільшення обсягу бази знань виникає необхідність автоматизації збереження, обробки та передачі інформації. Інформаційні технології безперечно вважаються двигуном прогресу в усіх сферах діяльності людини: науковій, медичній, політичній, освітній і т.д.

Якщо розглядати місце інформаційних технологій в освітній сфері [1], то можна дійти висновку, що вони є інструментом як навчального процесу, так і організації функціонування закладу. Зараз важко уявити собі учбовий заклад, у якому немає комп'ютерного класу чи електронної бібліотеки, користуватися матеріалами якої можна не виходячи з дому, що значно полегшує процес навчання. Один із показників якості навчального процесу – це успішність студентів учбового закладу. Важливим фактором забезпечення високого це облік студентів та дослідження наукових тенденцій.

Основна мета проекту полягає у розробці інформаційної системи обліку студентів та аналізу наукових тенденцій учбових кафедр Машинобудівного коледжу Сумського державного університету, призначенням якої є автоматизація роботи з особистими даними студентів.

Для реалізації поставленої мети необхідно виконати наступні завдання:

- виконати аналіз аналогів інформаційної системи та сформулювати задачу для дослідження;
- спроектувати інформаційну систему обліку студентів та дослідження наукових тенденцій учбових кафедр;
- розробити інформаційну систему.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Дослідження предметної області

Нині стрімке впровадження інформаційних технологій у всіх сферах життєдіяльності суспільства стало не просто тенденцією, а запорукою успіху будь-якого підприємства. Тому до системи надання освітніх послуг висуваються вимоги не тільки до якості надання послуг, а й до функціонування самої установи.

У результаті постає питання автоматизації певних процесів функціонування учбового закладу, таких як: створення особистих справ студентів та заповнення їх даними, формування статистики та аналіз наукової діяльності студента. Метою автоматизації є скорочення непродуктивних витрат робочого часу, контроль якості роботи працівників освіти, контроль успішності студентів, підвищення якості службових звітів та зменшення часу на формування звітної документації.

Для досягнення мети проекту необхідно виконати наступні задачі:

- проаналізувати предметну область та визначити актуальність розробки;
- провести порівняння аналогів програмних продуктів заданої тематики;
- розробити інтерфейс користувача інформаційної системи;
- розробити функціональну частину програмного продукту;
- виконати тестування web-додатку.

1.2 Аналіз аналогів

Різноманітні системи обліку вже давно розміщені у мережі Інтернет, їх призначення і сфера застосування різноманітні: обрахунок продукції магазину, контролювання абонементів у спортивному клубі і т.д. Якщо розглядати системи обліку заданої предметної області – то вибір інформаційних систем досить невеликий, і всі вони не досконалі.

Для визначення вимог до розроблюваної інформаційної системи було проведено аналіз ринку аналогів програмних продуктів заданої предметної області. Це необхідно, щоб точніше зрозуміти специфіку направлення, визначити слабкі та сильні сторони існуючих проектів та зуміти в подальшому створити унікальний продукт. Для порівняння було взято наступні інформаційні системи: «Універсальна система обліку», «Система обліку студентів які отримують соціальну стипендію», «Досьє студента».

Основними критеріями оцінювання інформаційних систем є широта спектру функціональних можливостей, легкість у користуванні та інсталяції, адаптивність до предметної області застосування.

«Універсальна Система Обліку» - це функціональна інформаційна система, яка зберігає велику кількість персональних даних клієнтів, постачальників та інших організацій, а також відомості про компанію з детальним описом її функціонування та характеристик (рис.1.1).

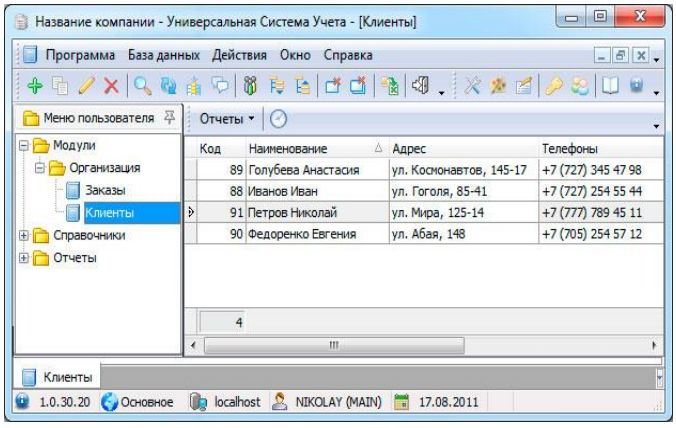


Рисунок 1.1 – Програма «Універсальна система обліку»

Web-додаток «Універсальна система обліку» дійсно є універсальною, може бути використана у будь-якій сфері життєдіяльності та застосовуватися на підприємствах різного напрямку. Програма не вимагає від користувача особливих знань інформаційних технологій та вмінь, і досить легка у інсталяції.

Головним недоліком цієї програми є висока вартість та сумнівна надійність розробника. Використання інформаційної системи для обліку юристів представлено на рисунку 1.2.

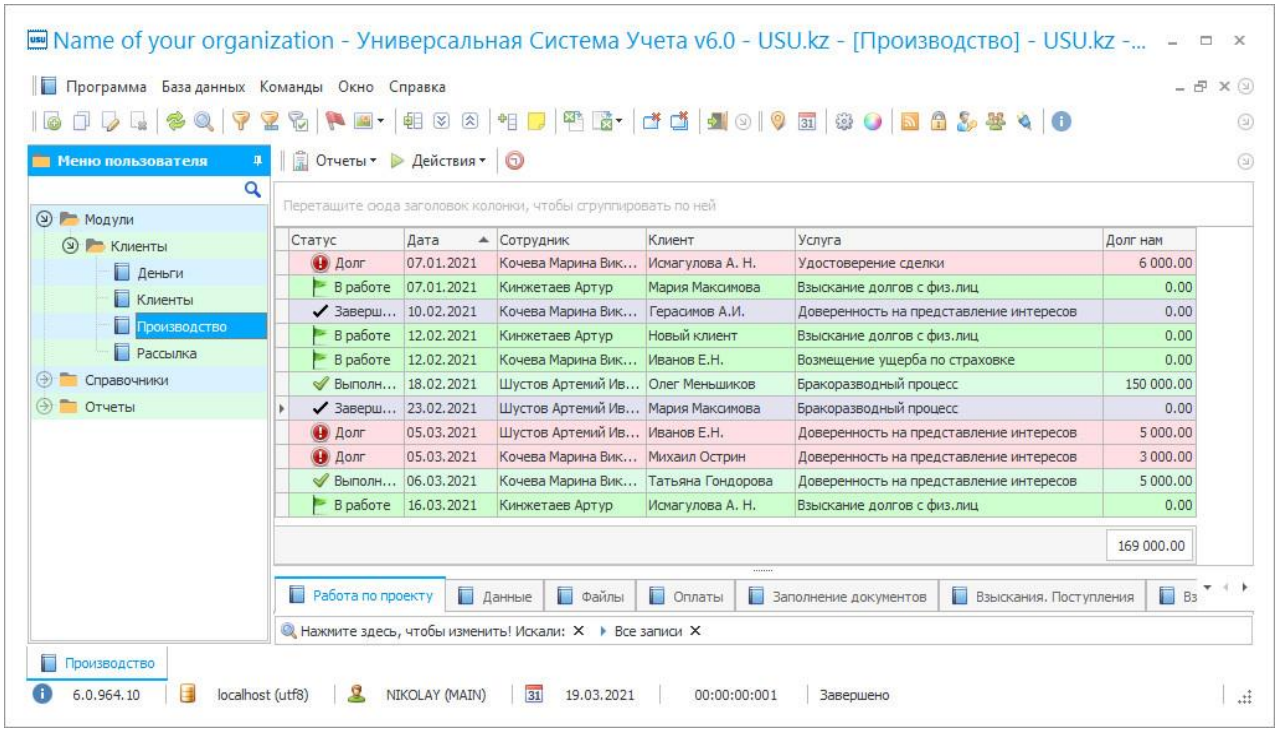


Рисунок 1.2 – Програма «Універсальна система обліку» для юристів

Наступним об’єктом аналізу є «Система обліку студентів які отримують соціальну стипендію» (рис.1.3). Дана система має дуже вузьку спеціалізацію, що слідує з назви, і призначена лише для певної категорії учасників учбового процесу.

Недоліки інформаційної системи наступні:

- складність в інсталяції;
- маленький спектр функцій;
- потреба у додатковому програмному забезпеченні;
- необхідність реєстрації в інших системах;
- висока вартість програми.

Для повноцінного функціонування «Системи обліку студентів які отримують соціальну стипендію» треба завантажити «Крипто автограф», що зображено на рисунку 1.3.

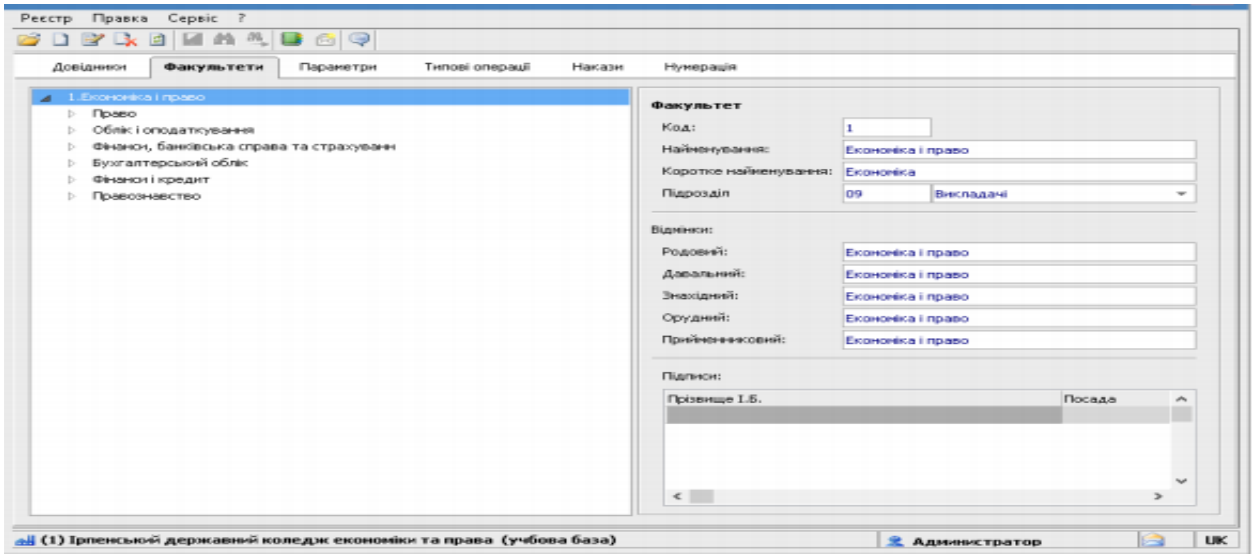


Рисунок 1.3 – Програма «Система обліку студентів які отримують соціальну стипендію»

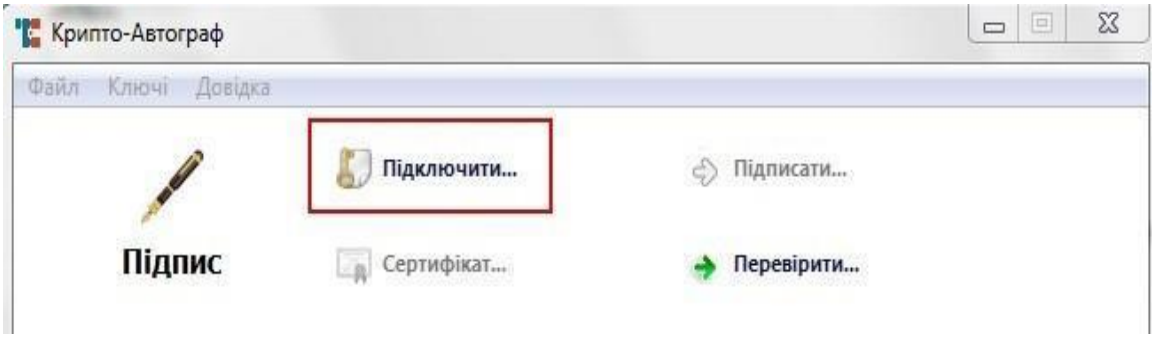


Рисунок 1.4 – Програма «Крипто автограф»

Останньою з трьох інформаційних систем для аналізу було взято web-сервіс «Досьє студента»[2].

Функціональні можливості даного сервісу:

- пошук студентів основних освітніх програм і слухачів програм додаткової освіти по ряду атрибутів (рис.1.5);
- перегляд особистих даних під час навчання;
- формування звітів успішності.

Перегляд інформації про студента у web-сервісі «Досьє студента» представлено на рисунку 1.6.

Головним недоліком даного сервісу є те, що він має обмежений доступ. Тобто користуватися ним можуть тільки ті користувачі системи, що були підключені до системи у групу «Користувачі досьє студента» заздалегідь.

До цієї групи можна включити ролі «Навчальні частини факультетів», «Навчально-методичне управління», «Керівництво ВНЗ» та ін. Всі користувачі, яким призначено одну з вищезазначених ролей, мають доступ до web-сервісу.

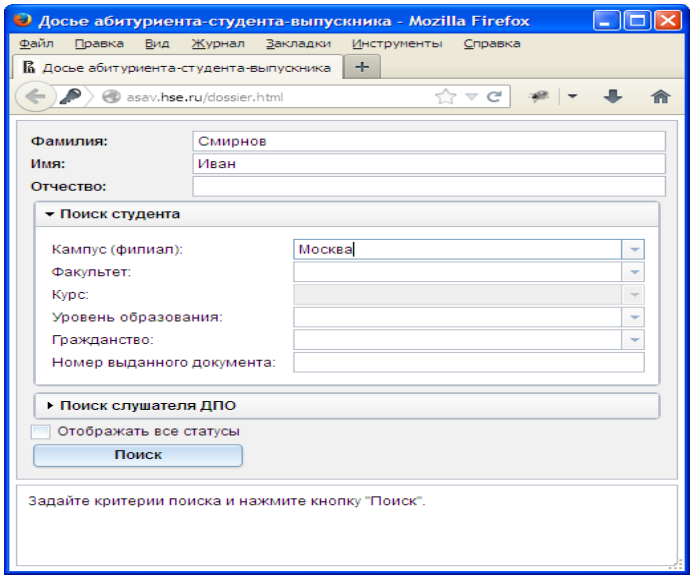


Рисунок 1.5 - Форма пошуку студента у web-сервісі «Досьє студента»

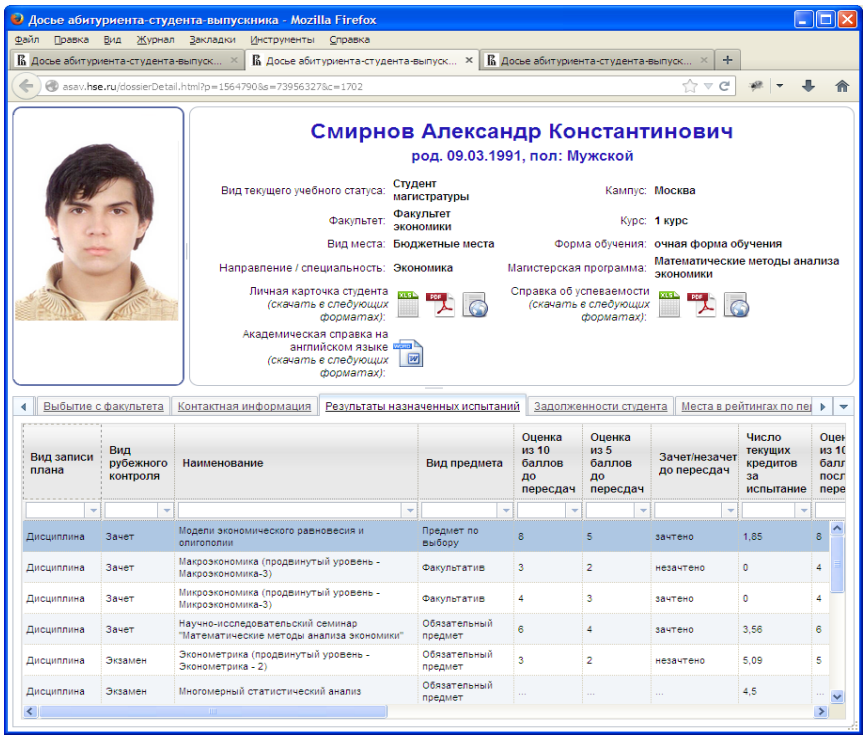


Рисунок 1.6 - Приклад досьє студента у web-сервісі «Досьє студента»

Після детального аналізу аналогів інформаційних систем, було визначено їх переваги та недоліки, які представлені в таблиці 1.1.

Таблиця 1.1 – Порівняльна таблиця характеристик аналогів web-сайтів

Характеристика/ ІС	**«Універсальна система обліку»**	**«Система обліку студентів які отримують соціальну стипендію»**	**«Досьє студента»**
Сучасний дизайн	-	-	-
Простота користування	+	+	-
Функціональність	+	-	+
Реєстрація користувачів	-	-	+
Перегляд та редагування особистих справ	+	+	+
Інструменти для аналізу даних	-	-	-
Формування звітів	+	-	+
Доступність	-	-	+

Завдяки даним з таблиці 1.1 можна зробити висновок, що під час розробки інформаційної системи треба реалізувати такі функціональні можливості як реєстрація користувачів, перегляд та редагування особистих справ, формування звітності та інструменти для аналізу даних системи. Важливим фактором успіху програмного продукту є його сучасність, зрозумілість та простота користування. Тому дизайн інформаційної системи буде розроблено відповідно до UI/UX-стандартів[3].

1.3 Постановка задачі

Метою дипломного проекту є розробка багатофункціонального додатку для обліку студентів Машинобудівного коледжу Сумського державного університету та дослідження тенденцій учбових кафедр.

Основні вимоги до майбутнього програмного продукту:

- забезпечити доступ до інтерфейсу та адміністративної панелі тільки для учбової частини коледжу;
- організувати виведення статистики за кафедрами та роками за ключовими словами;
- забезпечити можливість створювати та редагувати особисті дані студентів, предметів;
- створити форму для додавання наукових робіт до особистих справ студентів;
- забезпечити завантаження особистих справ студентів на персональний комп’ютер у вигляді Excel-таблиці;
- забезпечити можливість виставлення оцінок з предметів будь-якому студенту.

2 ВИБІР МЕТОДІВ ВИРІШЕННЯ ЗАВДАННЯ

2.1 Вибір засобів розробки системи

Сфера інформаційних технологій багата різноманітними технологіями та інструментами розробки програмних продуктів. Засоби реалізації обираються в залежності від виду програмного забезпечення, його призначення та області застосування.

Мовами програмування, які мають найважніший вплив на сферу інформаційних технологій є Visual Basic, C, C++, Java, C#, Python, JavaScript, PHP. Кожну з них можна використовувати в тій чи іншій мірі в якості алгоритмічної мови[4]. Для надання web-сторінкам певних сценаріїв, наприклад запис інформації з форми в базу даних використовують мову PHP.

Мова програмування PHP[5] – це серверна мова програмування, за допомогою якої можна створювати web-сайти, причому як невеликі лендінги, що складаються з однієї сторінки, так і гігантські системи, що використовують сотні і тисячі серверів. Електронна енциклопедія Wikipedia та соціальна мережа Facebook створені з використанням PHP. Будучи однією з перших мов програмування, орієнтованих на web-розробку, PHP пройшла тривалий шлях від самого початку зародження Інтернету. Тому вона залишається однією з найпопулярніших мов програмування в світі.

Основними перевагами мови PHP є наступні[6]:

- Орієнтація на web-розробку. З самого початку призначенням PHP була розробка web-орієнтованих додатків. Тому більшість функцій та процес ухвалення рішень створені для зручної роботи у web-середовищі.
- Кросплатформеність. Складнощі з перенесенням web-сайту написаному на PHP з операційної системи Windows на Mac OS X чи Linux, або навпаки, будуть мінімальними.
- Об’єктно-орієнтоване програмування. З кожною новою версією PHP все краще підтримує об’єктно-орієнтований підхід: від використання

лише певних елементів до повного переходу на ООП і появи JIT компіляції.

- Вбудованість html-коду. PHP - це мова, у код якої можна вбудовувати html-код web-сторінок, який буде коректно оброблятися PHP-інтерпретатором. Обробник PHP просто починає виконувати код після відкриваючого тегу (<?php) і продовжує виконання до того моменту, поки не з’явиться закриваючий тег (?>).

Для швидкого створення надійних та потужних додатків використовують PHP фреймворк Laravel. Laravel реалізує концепцію модель-вид-контролер[7] (MVC) та автоматизує загальні задачі. Також його можна уявити як широконалаштовау систему класів, що пов'язані між собою, яка призначена для розробки та керування web-застосунками.

Якщо порівнювати Laravel з іншими php-фреймворками, то він є найпопулярнішим серед них, хоча «під капотом» використовує функціонал написаний за допомогою PHP фреймворка Symfony. Популярність цього фреймворка обумовлена підтримкою будь-яких видів тестування «із коробки», легким масштабуванням, досить розвиненою еко-системою інструментів, від інструментів для деплойменту до мікро-фреймворків.

Фреймворк використовує кращі практики розробки і забезпечує великий вибір шаблонів проектування, що робить чистим і оптимізованим код. Однією з головних переваг Laravel є те, що він забезпечує набагато більшу продуктивність в порівнянні зі схожими фреймворками, це можливо, у тому числі завдяки потужній системі кешування.

Сучасний JavaScript[8] – це безпечна мова програмування, яка не надає доступ до пам'яті або процесору на низькому рівні, тому що спочатку була створений для браузерів, які не потребують цього. Можливості JavaScript залежать від оточення, в якому він працює. Наприклад, Node.JS підтримує функції читання та запису довільних файлів, виконання мережевих запитів і т.д. У браузері для JavaScript є все, що треба для маніпулювання веб-

сторінками та взаємодії між користувачем і web-сервером.

Браузер JavaScript надає наступні можливості:

- Додавати новий HTML-код на сторінку, змінювати існуючий вміст, модифікувати стилі.
- Реагувати на дії користувача, клацання миші, переміщення вказівника, натискання клавіш.
- Відправляти мережеві запити на віддалені сервери, завантажувати файли (технології AJAX і COMET).
- Отримувати і встановлювати cookies, задавати питання відвідувачеві, показувати повідомлення.
- Запам'ятовувати дані на стороні клієнта («local storage»).

Можливості JavaScript в браузері обмежені заради безпеки користувача, їх мета полягає у запобіганні доступу ненадійної web-сторінки до особистої інформації або нанесення шкоди даним користувача.

Приклади таких обмежень включають в себе:

- На web-сторінці JavaScript не може читати, записувати та копіювати довільні файли на жорсткий диск, запускати програми. Він не має прямого доступу до системних функцій ОС.
- Сучасні браузери дозволяють йому працювати з файлами, але з обмеженим доступом, і надають його, тільки якщо користувач виконує певні дії, такі як «перетягування» файлу в вікно браузера або його вибір за допомогою тега <input>.
- Існує багато способів взаємодії з камерою, мікрофоном та іншими пристроями, але вони вимагають явного дозволу від користувача.
- Різні вікна та вкладки не мають доступу один до одного.
- JavaScript може легко взаємодіяти з сервером, з якого було завантажено поточну сторінку.

Розробниками написано велику кількість інструментів, які взаємодіють з основною мовою JavaScript, які надають доступ до великої кількості додаткових можливостей:

- Програмні інтерфейси додатків (API), що вбудовані в браузери.
- Сторонні API дозволяють розробникам впроваджувати функціональні можливості в свої web-сайти від інших розробників.
- Також можна застосувати до HTML сторонні фреймворки і бібліотеки, що дозволяють прискорити створення web-додатків.

Для створення інтерфейсу користувача буде використовуватись JavaScript фреймворк Vue.js[9]. Тому що він використовується разом з фреймворком Laravel, має просту реактивність, що робить керування станами дуже простим і інтуїтивним, оптимізований рендеринг сторінок. Однією з його найважливіших переваг є використання віртуального DOM, що робить інтерфейс більш швидким.

Отже, для створення додатку буде використано такі технології:

- HTML – стандартна мова розмітки web-сторінок в Інтернеті.
- CSS3 – каскадні таблиці стилі, що необхідні для надання web-сторінці візуальних ефектів
- Мова JavaScript + Фреймворк Vue.js – для надання web-сторінкам динамічності та створення інтерфейсу користувача.
- Мова PHP + Фреймворк Laravel – для створення швидкого та надійного бекенду web-додатку.
- MySQL – реляційна база даних для збереження інформації[10].

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Проектування дизайну високого рівня

Розробка інформаційної системи починається з проектування дизайну високого рівня, щоб краще охарактеризувати архітектуру, яка буде використовуватися для створення програмного продукту. Діаграма High Level Design[11] описує основні компоненти системи та їх призначення (рис.3.1).

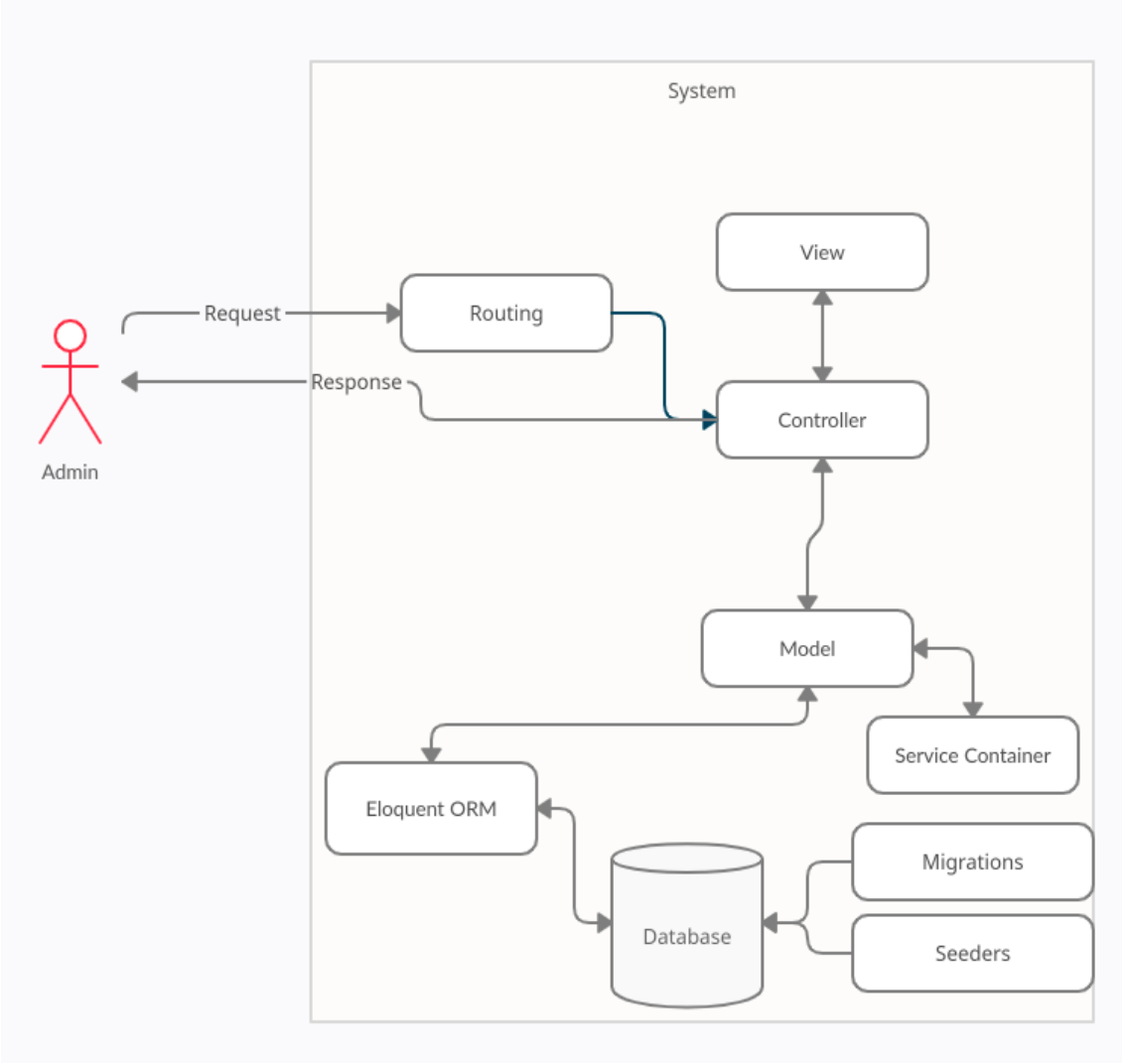


Рисунок 3.1 – Архітектура інформаційної системи

3.2 Проектування бази даних

Наступним етапом є проектування бази даних інформаційної системи[12], що складається з кількох етапів: виділення сутностей, визначення структури та зв’язків таблиць.

Для інформаційної системи обліку студентів та аналізу наукових тенденцій учбових кафедр Машинобудівного коледжу СумДУ було виділено 10 сутностей:

- Сутність Students – зберігає в собі всі дані про студентів, номер студентського квитка, ПІБ, дату народження та іншу необхідну інформацію, зв’язується з сутністю Groups зв`язком багато-до-одного, тобто в одній групі має бути багато студентів.
- Сутність Groups – містить інформацію про групи, та зв’язується з сутність Departments зв’язком багато-до-одного.
- Сутність Departments – зберігає інформацію про кафедри.
- Сутність Student work – зберігає інформацію про наукові роботи студентів, зв’язується з сутністю Departments зв’язком багато-до-одного.
- Сутність Student Student work – є проміжною сутністю для реалізацію зв’язку багато-до-багатьох сутностей Students і Student Work.
- Сутність Disciplines – зберігає в собі інформацію про дисципліни.
- Сутність Group Discipline– проміжна сутність, яка пов’язує сутність Groups і Lessons, тобто зберігає в собі дисципліни які підключені до певної групи.
- Сутність Student Discipline– зберігає в собі інформацію про предмети кожного студента, при додаванні дисципліни до групи, вона автоматично за допомогою тригера додасться до дисципліни студента, а також щоб була можливість підключати студентів до дисципліни за вибором.
- Сутність Users – зберігає в собі дані про користувачів системи.
- Сутність Migrations - зберігає в собі інформацію про виконані

міграції. Фреймворк Laravel має свою систему міграцій для простого перенесення проекту, тобто кожна таблиця має свою міграцію або схожі за суттю таблиці об’єднуються в міграцію. Це дозволяє створювати таблиці за допомогою самого фреймворку, без написання SQL-запитів

На рисунку 3.2 зображена ER-діаграма інформаційної системи, структуру якої складають таблиці бази даних та зв’язки між ними.

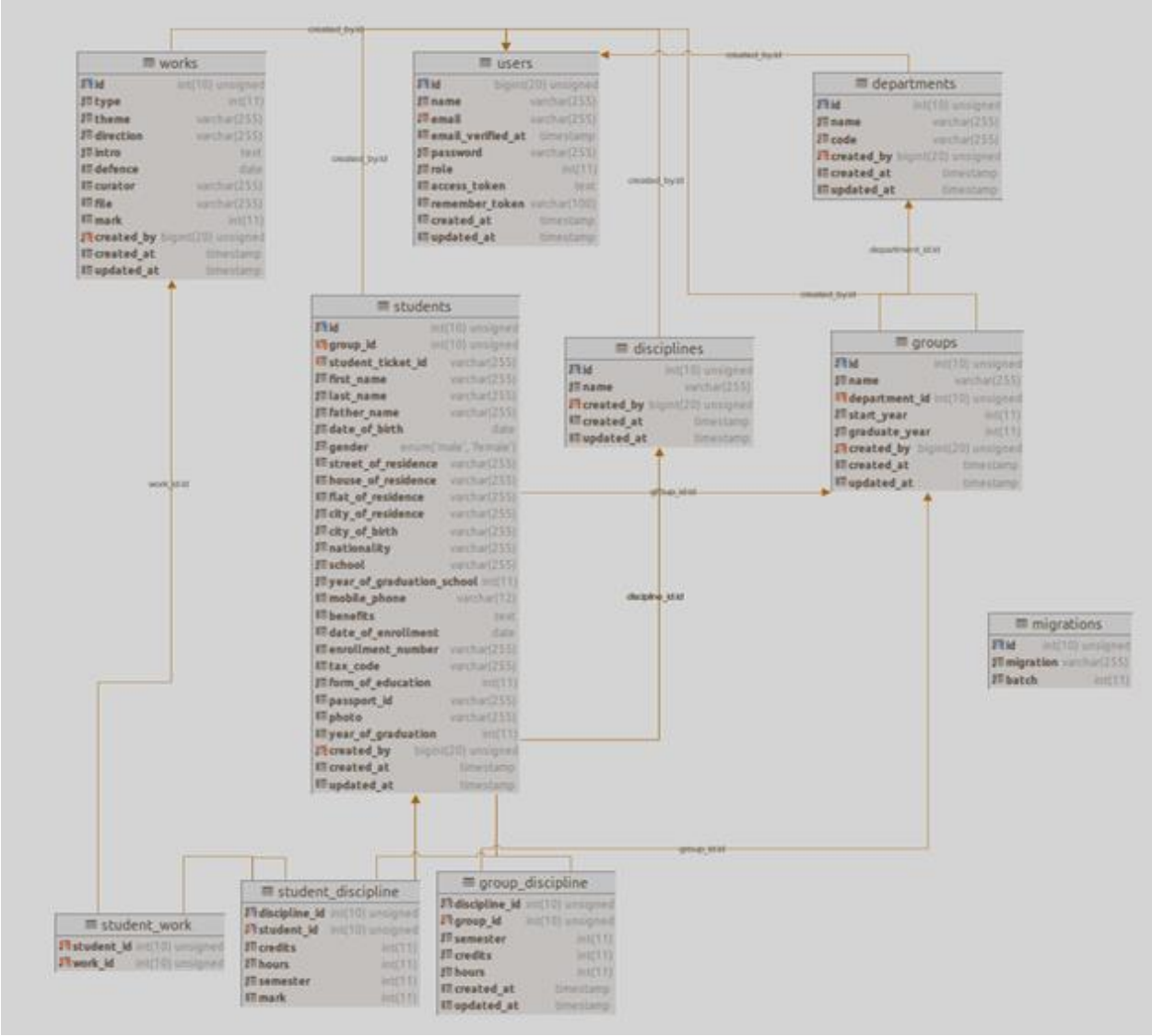


Рисунок 3.2 – ER-діаграма інформаційної системи

- Сутність Reset Passwords

Також всі сутності, крім міграцій, користувачів та скинутих паролів, зв’язані з сутностю користувачів. Таким чином ми реалізуєм логування змін в цих сотностях, тобто при кожній зміні буде записуватись ім’я користувача, хто вніс зміни.

База даних повинна бути в 3 нормальній формі. Відношення знаходиться в 3 Нормальній Формі (НФ) [13], коли знаходиться у 2НФ і кожен неключовий атрибут нетранзитивно залежить від первинного ключа. Отже, друге правило вимагає виносити всі не ключові поля, вміст яких може стосуватися кількох записів таблиці в окремі таблиці.

3.3 Реалізація інформаційної системи

Підготовчим етапом розробки програмного продукту є налаштування web-серверу Apache2[14]. Для цього треба створити віртуальний хост для проекту та встановити необхідні розширення: php8 та mysql. Далі завантажуємо менеджер залежностей для мови PHP – Composer. Використовуючи Composer, інсталюємо фреймворк Laravel та ініціюємо git-репозиторій[15]. Налаштування підключення до бази даних та поштового серверу виконуємо у файлі .env.

Структура створеного проекту Laravel представлена на рисунку 3.3.

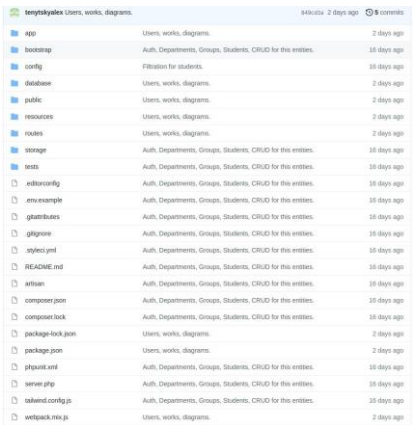


Рисунок 3.3 – Структура проекту

Наступним етапом є створення міграцій для таблиць за допомогою фреймворку. Приклад міграції зображено на рисунку 3.4.

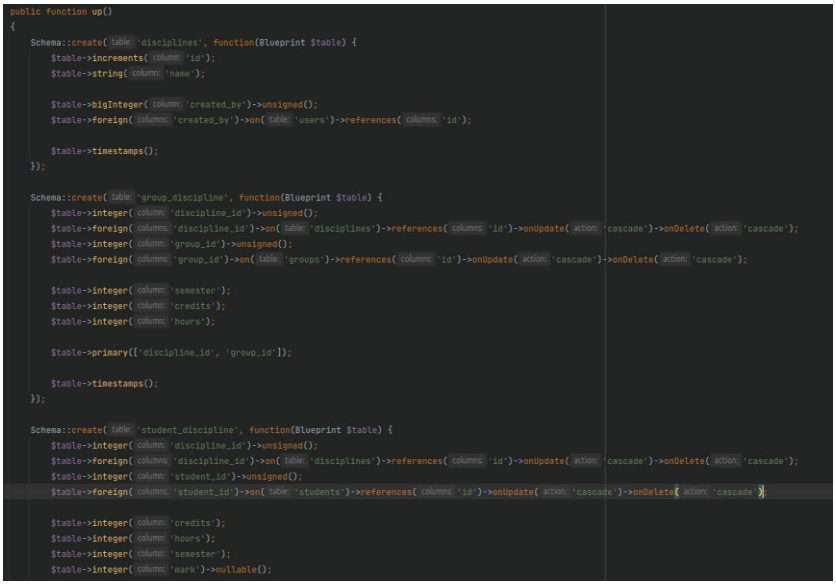


Рисунок 3.4 – Файл міграції

Коли користувач робить запит до системи, фреймворк знаходить відповідності між вказаною адресою та інструкціями, які описані у файлі маршрутизації (рис.3.5) та типом запиту, після чого буде задіяний необхідний контролер, який в залежності від отриманих даних викликає класи-сервіси та відображає необхідний вигляд (рис.3.6).

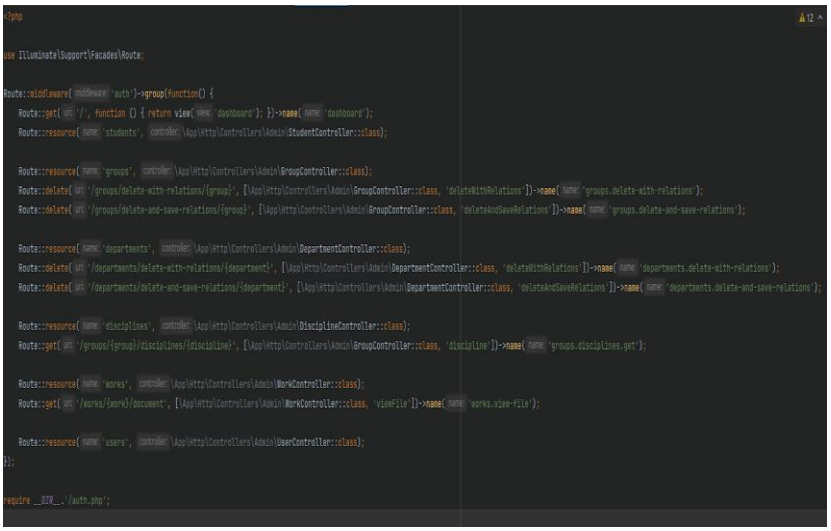
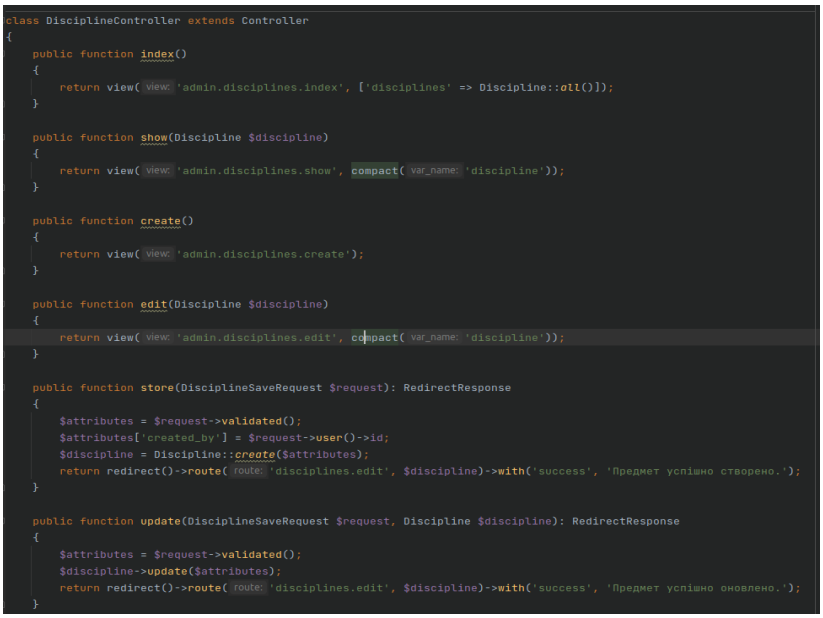


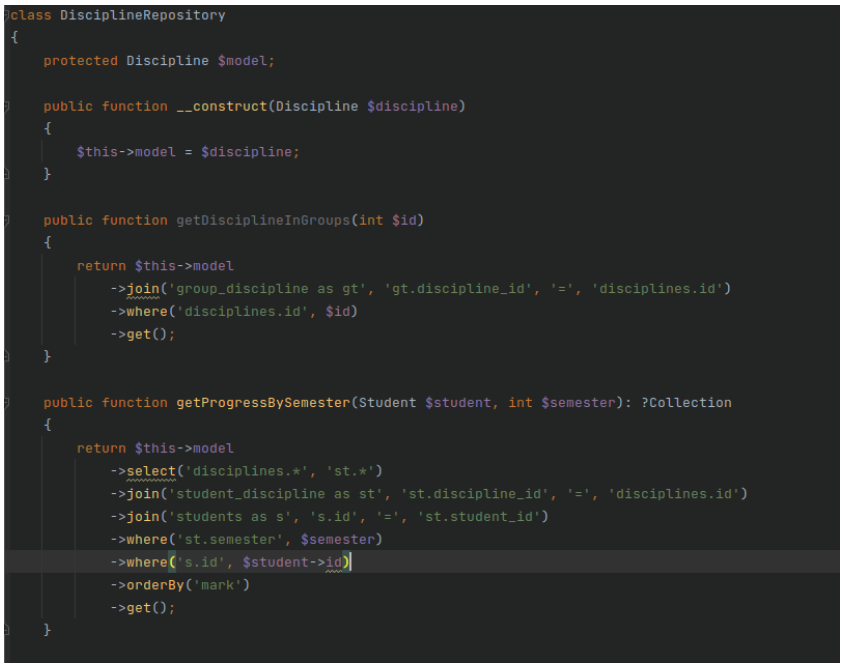
Рисунок 3.5 - Файл маршрутизації web.php



```php
class DisciplineController extends Controller
{
    public function index()
    {
        return view( view: 'admin.disciplines.index', ['disciplines' => Discipline::all()]);
    }

    public function show(Discipline $discipline)
    {
        return view( view: 'admin.disciplines.show', compact( var_name: 'discipline'));
    }

    public function create()
    {
        return view( view: 'admin.disciplines.create');
    }

    public function edit(Discipline $discipline)
    {
        return view( view: 'admin.disciplines.edit', compact( var_name: 'discipline'));
    }

    public function store(DisciplineSaveRequest $request): RedirectResponse
    {
        $attributes = $request->validated();
        $attributes['created_by'] = $request->user()->id;
        $discipline = Discipline::create($attributes);
        return redirect()->route( route: 'disciplines.edit', $discipline)->with('success', 'Предмет успішно створено.');
    }

    public function update(DisciplineSaveRequest $request, Discipline $discipline): RedirectResponse
    {
        $attributes = $request->validated();
        $discipline->update($attributes);
        return redirect()->route( route: 'disciplines.edit', $discipline)->with('success', 'Предмет успішно оновлено.');
    }
```

Рисунок 3.6 – Контролер для обробки дисциплін

Контролер передає дані в модель, що взаємодіє з базою даних та іншими сервісами, отримані дані від користувача на обробку. Класи репозиторії є частиною моделі та взаємодіють зі сховищем даних, вони є реалізацією патерну з такою ж назвою. Даний патерн застосовується для розділення бізнес-логіки додатку та сховища даних (рис.3.7).



```php
class DisciplineRepository
{
    protected Discipline $model;

    public function __construct(Discipline $discipline)
    {
        $this->model = $discipline;
    }

    public function getDisciplineInGroups(int $id)
    {
        return $this->model
            ->join('group_discipline as gt', 'gt.discipline_id', '=', 'disciplines.id')
            ->where('disciplines.id', $id)
            ->get();
    }

    public function getProgressBySemester(Student $student, int $semester): ?Collection
    {
        return $this->model
            ->select('disciplines.*', 'st.*')
            ->join('student_discipline as st', 'st.discipline_id', '=', 'disciplines.id')
            ->join('students as s', 's.id', '=', 'st.student_id')
            ->where('st.semester', $semester)
            ->where('s.id', $student->id)
            ->orderBy('mark')
            ->get();
    }
```

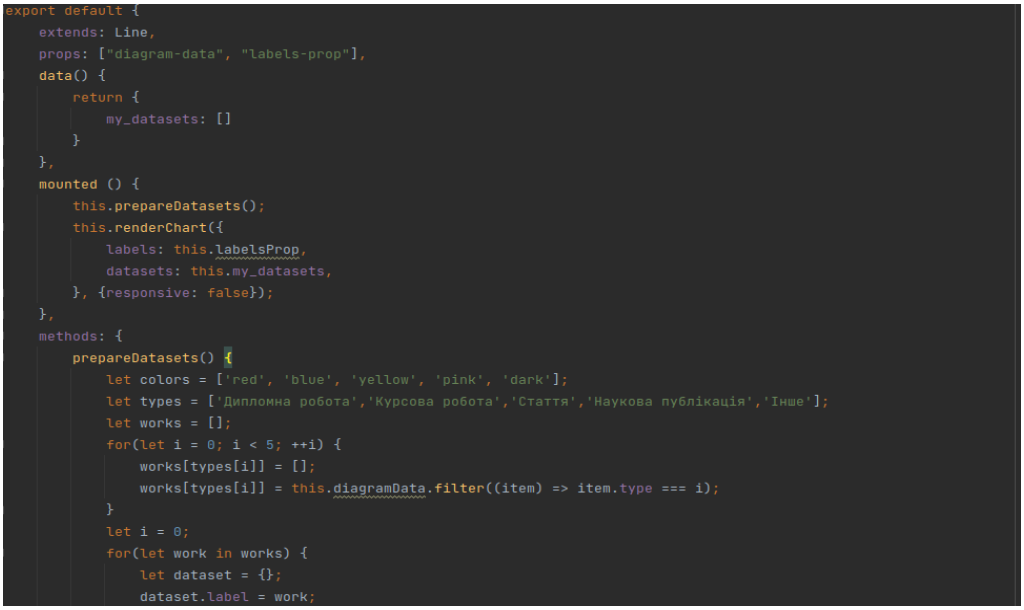
Рисунок 3.7 – Приклад класу репозиторію для дисциплін.

Після отримання даних, вони повертаються до контроллера, які він передає у вид (рис.3.8). В нашому випадку використовується шаблонізатор Blade який постачається разом з Laravel. Його перевагами є те що, він підтримує наслідування шаблонів та секцій, що виключає дублювання коду.

```
@extends('admin.layouts.admin')

@section('header-title')
    {{ __("Створення спеціальності") }}
@endsection

@section('content')
    <div class="py-12 max-w-7xl mx-auto sm:px-6 lg:px-8 admin-form">
        <form method="post" action="{{ route('departments.store') }}" enctype="multipart/form-data">
            @csrf
            <div class="form-group">
                <div class="row">
                    <div class="col-12 col-lg-6">
                        <label for="last-name">Назва</label>
                        <input type="text" name="name" class="form-control" {{ old('name') ?? '' }} id="last-name" placeholder="Введіть назву спеціальності">
                    </div>
                </div>
            </div>
            <div class="form-group">
                <div class="row">
                    <div class="col-12 col-lg-6">
                        <label for="code">Код спеціальності</label>
                        <input type="text" name="code" class="form-control" {{ old('code') ?? '' }} id="code" placeholder="Введіть код спеціальності">
                    </div>
                </div>
            </div>
            <button type="submit" class="btn btn-primary">Створити</button>
        </form>
    </div>
@endsection()
```

Рисунок 3.8 – Приклад виду для створення спеціальностей

У системі має бути реалізована функція побудови графіків, для цього було використано компонент-обгортку для Vue.js - chart.js. Приклад файлу для малювання лінійної діаграми залежностей зображено на рисунку 3.9.



```
export default {
    extends: Line,
    props: ["diagram-data", "labels-prop"],
    data() {
        return {
            my_datasets: []
        }
    },
    mounted () {
        this.prepareDatasets();
        this.renderChart({
            labels: this.labelsProp,
            datasets: this.my_datasets,
        }, {responsive: false});
    },
    methods: {
        prepareDatasets() {
            let colors = ['red', 'blue', 'yellow', 'pink', 'dark'];
            let types = ['Дипломна робота','Курсова робота','Стаття','Наукова публікація','Інше'];
            let works = [];
            for(let i = 0; i < 5; ++i) {
                works[types[i]] = [];
                works[types[i]] = this.diagramData.filter((item) => item.type === i);
            }
            let i = 0;
            for(let work in works) {
                let dataset = {};
                dataset.label = work;
```

Рисунок 3.9 – Приклад компонента для малювання діаграми

3.4 Демонстрація роботи web-додатку

Основними функціональними можливостями для користувача є створення та редагування студентів, груп, спеціальностей, дисциплін, студентських робіт, перегляд графіків наукових тенденцій, перегляд статистики успішності студентів та груп загалом.

Користувачі в системі розділені на 2 ролі: адміністратор та редактор. Адміністратор має право створювати та редагувати користувачів системи, редактор не має такої можливості.

При вході на сайт з’являється сторінка авторизації на якій ми повинні вказати електронну пошту та пароль користувача (рис.3.10).

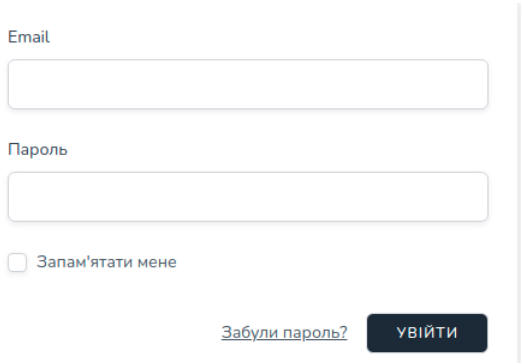


Рисунок 3.10 – Форма авторизації користувача

У випадку якщо користувач забув пароль, його можна відновити натиснувши на посилання «Забули пароль?», з’явиться сторінка для введення електронної пошти користувача, якщо пошта коректна – то буде відправлено електронний лист за вказаною електронною поштою з посиланням для скидання паролю (рис.3.11-3.12).

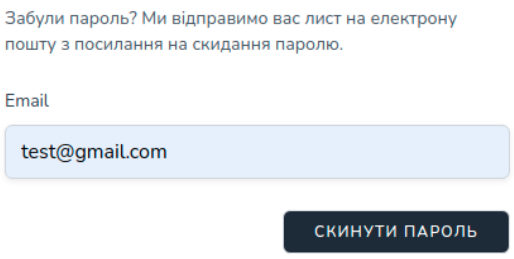


Рисунок 3.11 – Скидання паролю користувача

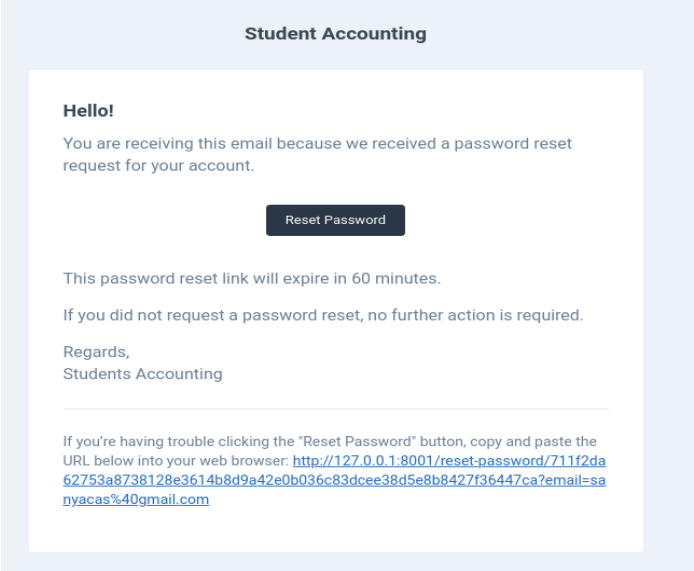


Рисунок 3.12 – Електронний лист з інструкціями для скидання паролю

Після переходу за посиланням з електронної пошти користувача з'явиться форма для зміни пароля, після чого можна буде авторизуватись у системі з новим паролем. При успішній авторизації з'являється головна сторінка інформаційної системи (рис.3.13).

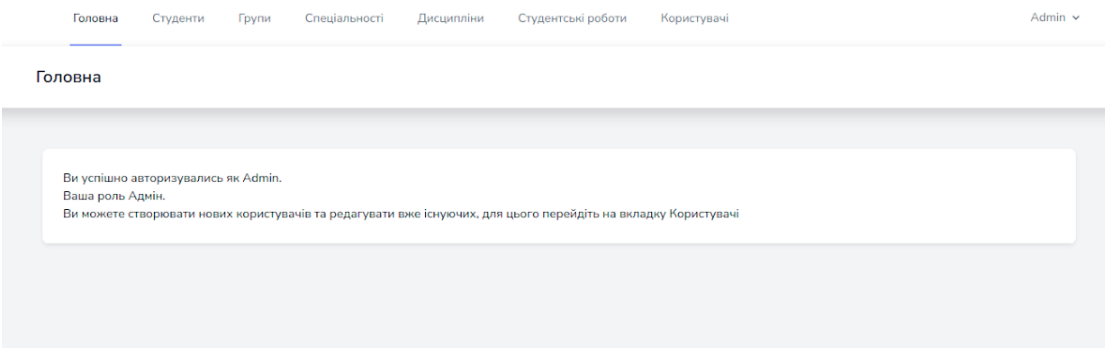


Рисунок 3.13 – Головна сторінка інформаційної системи

Меню web-додатку містить такі компоненти:

- студенти;
- групи;
- спеціальності;
- дисципліни;
- студентські роботи;
- користувачі.

Структуру сторінки «Спеціальності» складає розширений фільтр пошуку по назві або коду спеціальності (рис.3.14), кнопка створення нової спеціальності (рис.3.15) і таблиця відображення результатів запиту користувача. Після того як було створено нову спеціальність користувача буде перенаправлено на сторінку редагування (рис.3.16). Огляд web-сторінки «Спеціальності» представлено на рисунку 3.17.

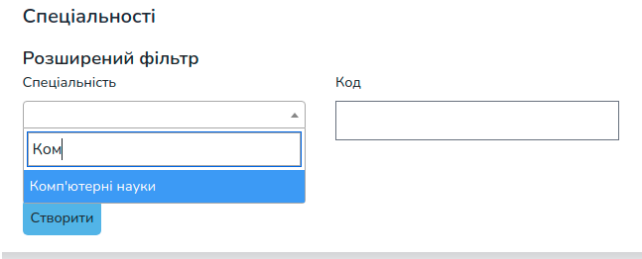


Рисунок 3.14 – Фільтрація спеціальностей

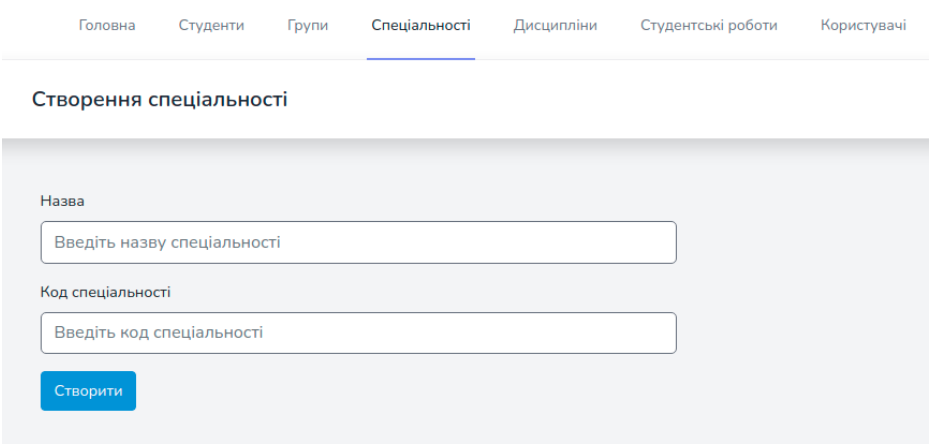


Рисунок 3.15 – Створення нової спеціальності

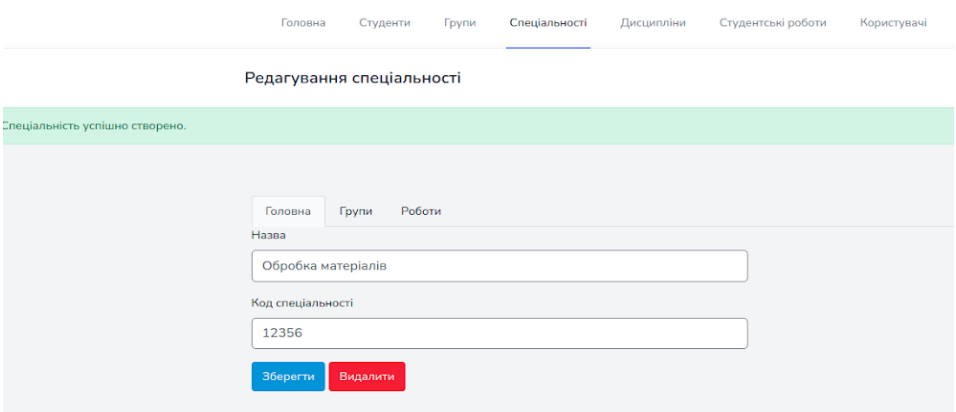


Рисунок 3.16 – Редагування спеціальності

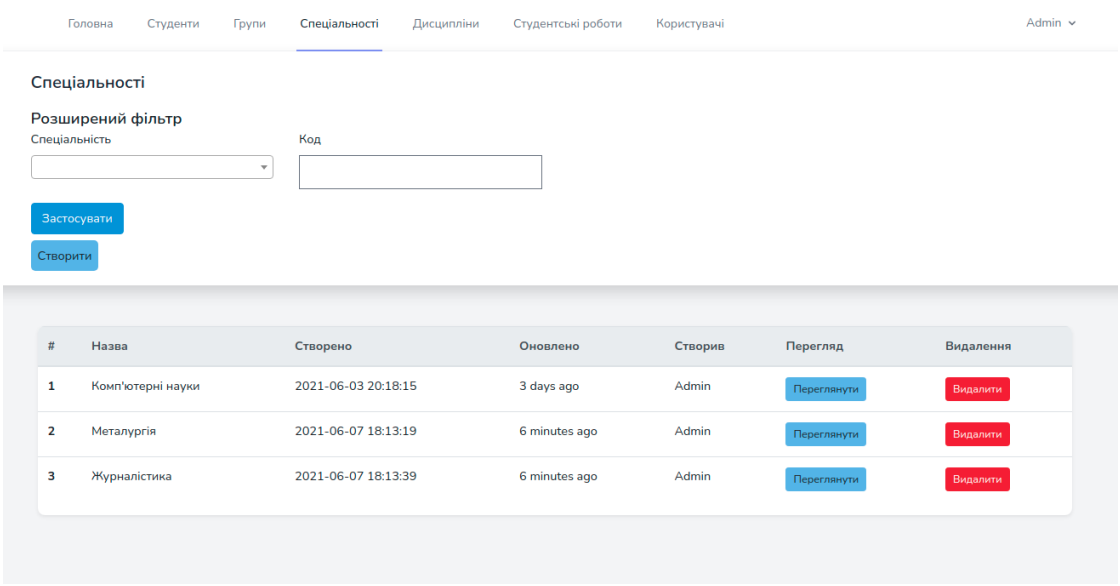


Рисунок 3.17 – Web-сторінка «Спеціальності»

Для того щоб переглянути які групи навчаються на спеціальності або наукові роботи студентів – треба перейти до пунктів меню редагування спеціальності «Групи» та «Роботи» відповідно (рис.3.18-3.19).

Виведення статистики наукових робіт виконується за кількістю робіт по типам та отриманим оцінкам. Також є можливість переглянути загальну тенденцію написання наукових робіт за роками та тенденцію за роками по типам наукових робіт.

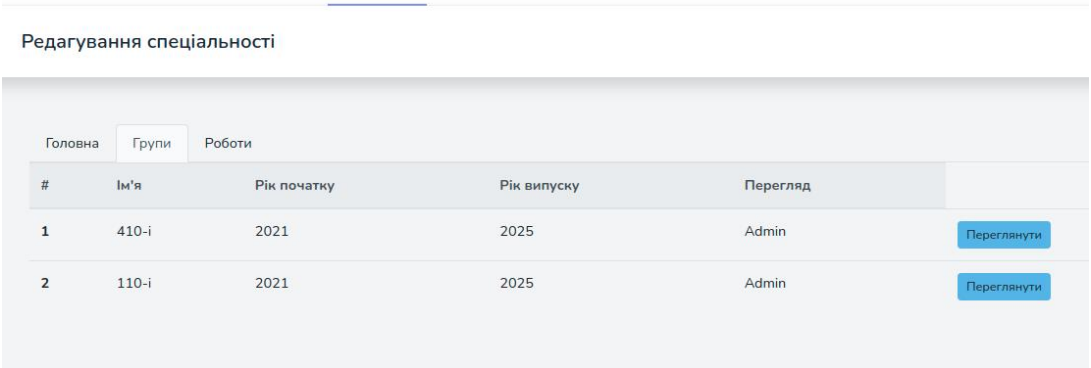


Рисунок 3.18 – Огляд груп спеціальності

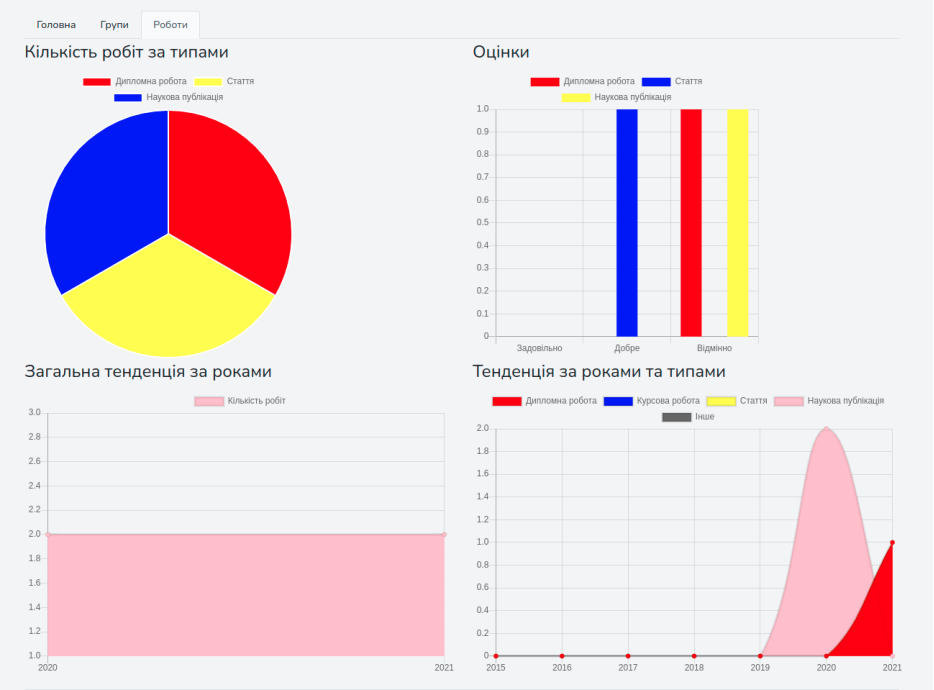


Рисунок 3.19 – Перегляд статистики наукових робіт

Призначення web-сторінки «Студенти» – це облік студентів учбового закладу, її елементами є розширений фільтр, можливості перегляду, створення, редагування та видалення студентів (рис.3.20-3.22). Фільтрація записів у базі даних виконується за групою, спеціальністю, формою навчання або роком випуску.

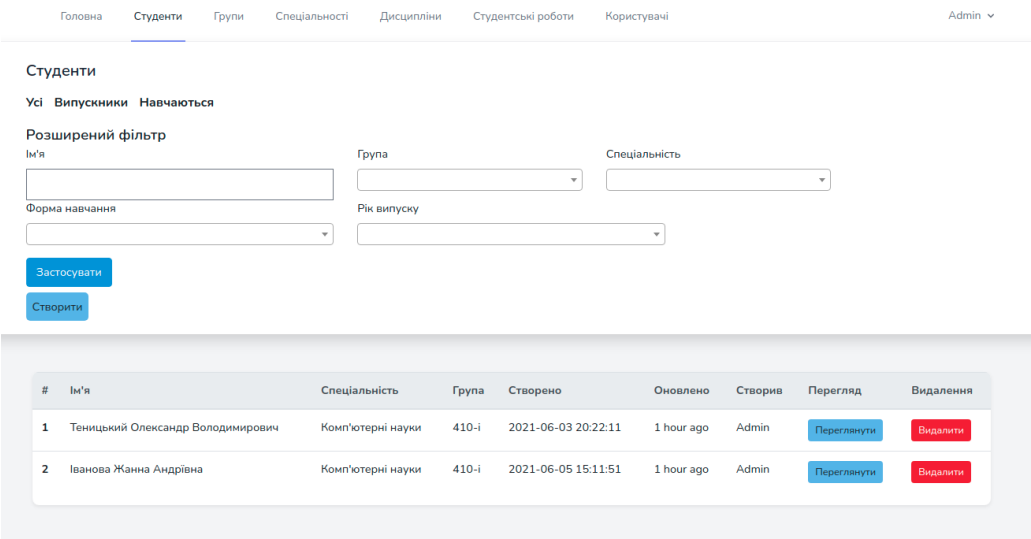


Рисунок 3.20 – Web-сторінка «Студенти»

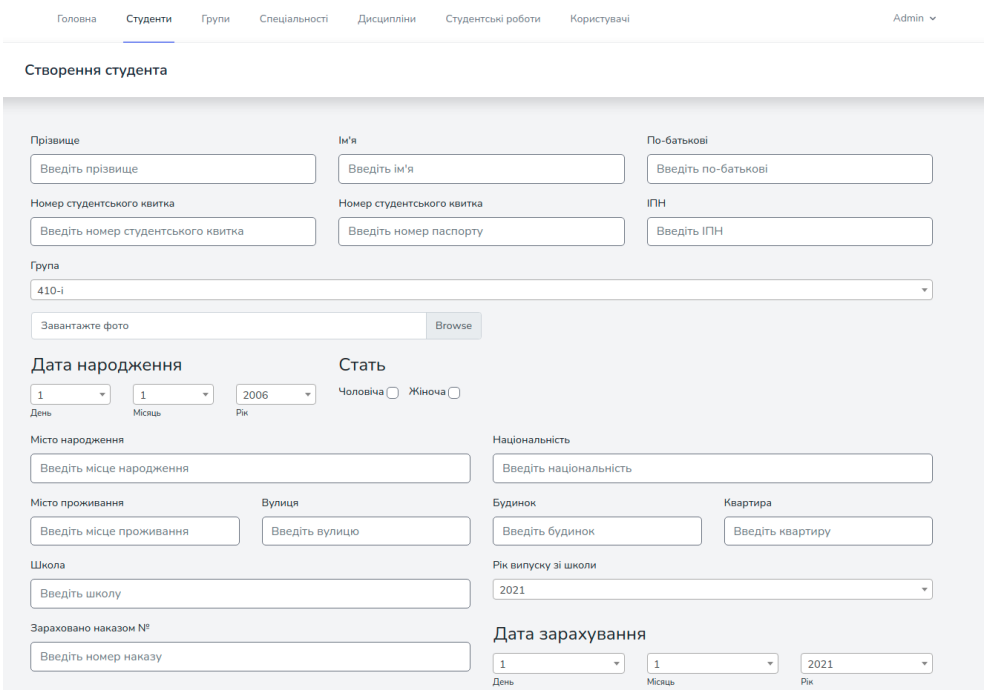


Рисунок 3.21 – Web-сторінка створення нового студента

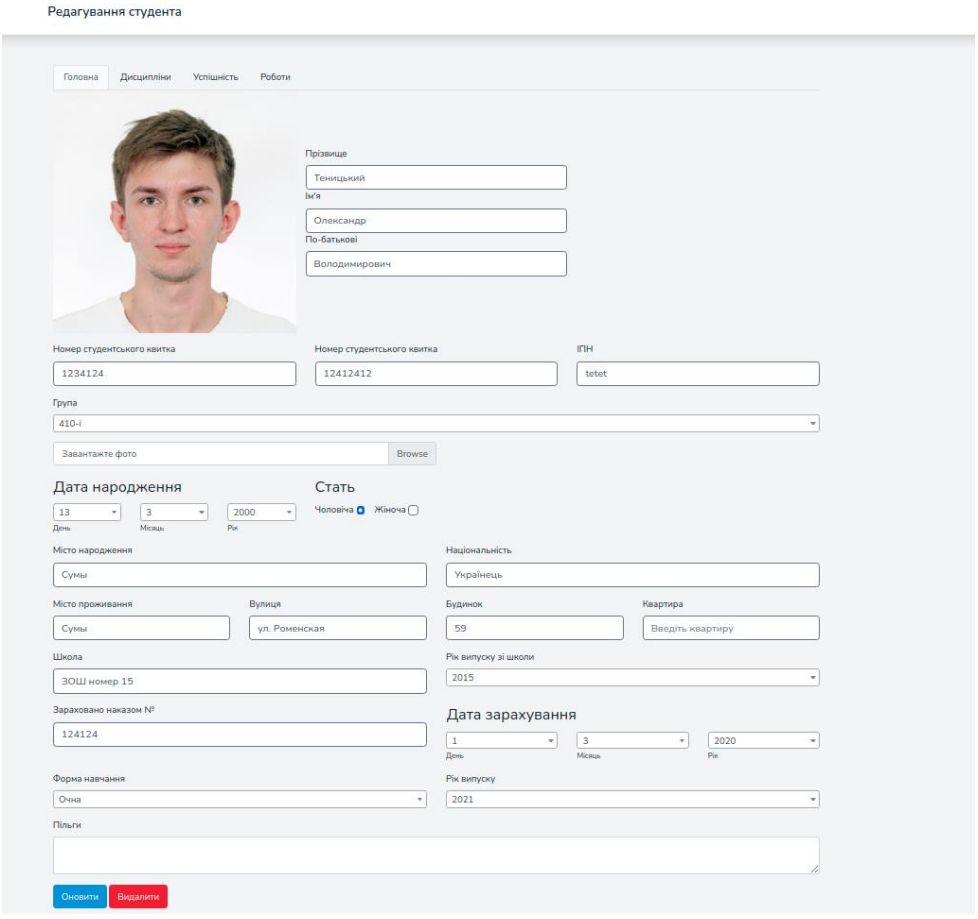


Рисунок 3.22 – Web-сторінка редагування студента

Окрім перегляду особистих даних студента користувач може переглядати, редагувати та видаляти дисципліни, що вивчає студент (рис.3.23). Якщо предмет є обов'язковим – то користувач може внести зміни тільки через інтерфейс групи.

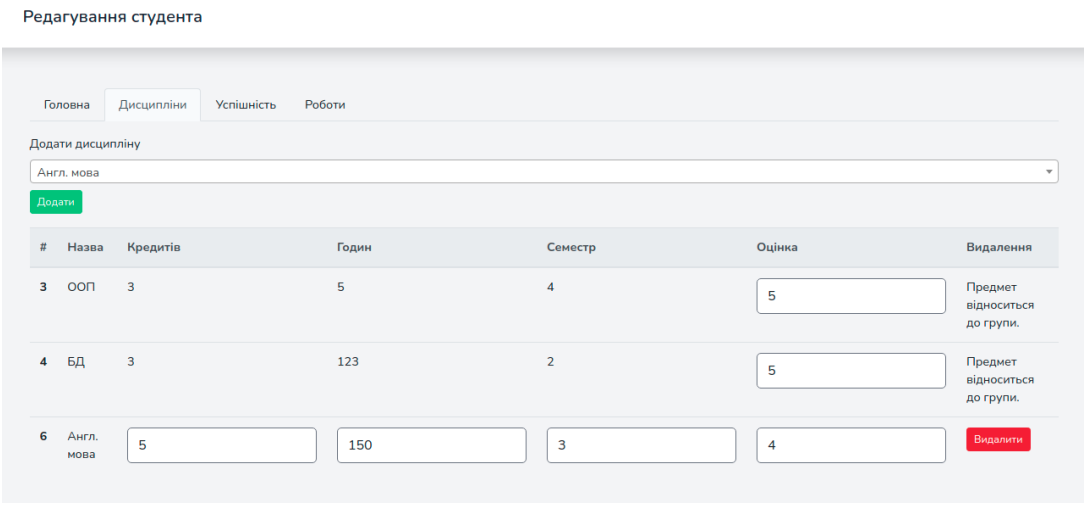


Рисунок 3.23 – Web-сторінка редагування дисциплін студента

Наступним пунктом меню редагування студента є перегляд успішності за семестрами, який можливим лише після виставлення оцінок у дисципліни (рис.3.24).

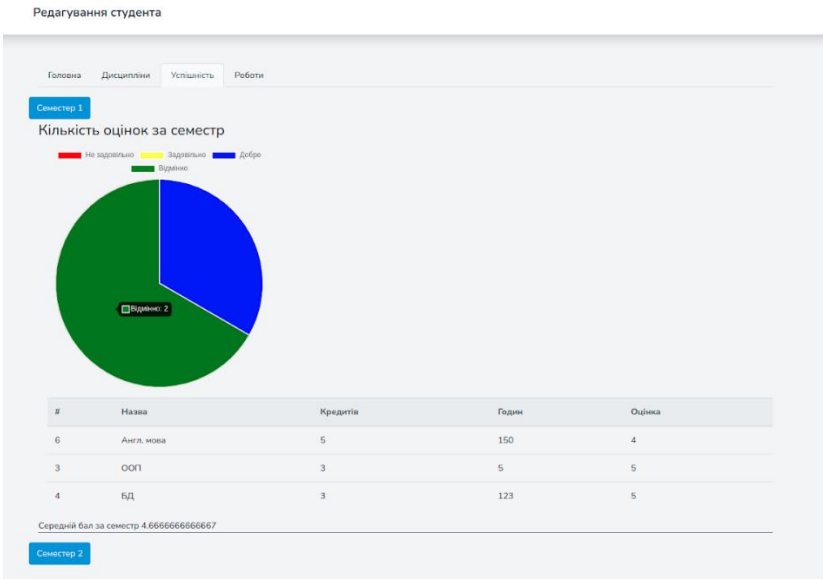


Рисунок 3.24 – Web-сторінка перегляду успішності студента

Останнім пунктом меню редагування інформації про студента є перегляд та додання його наукових робіт (рис.3.25). Діаграми на web-сторінці будуються за кількість робіт по типам та виставленими оцінками.

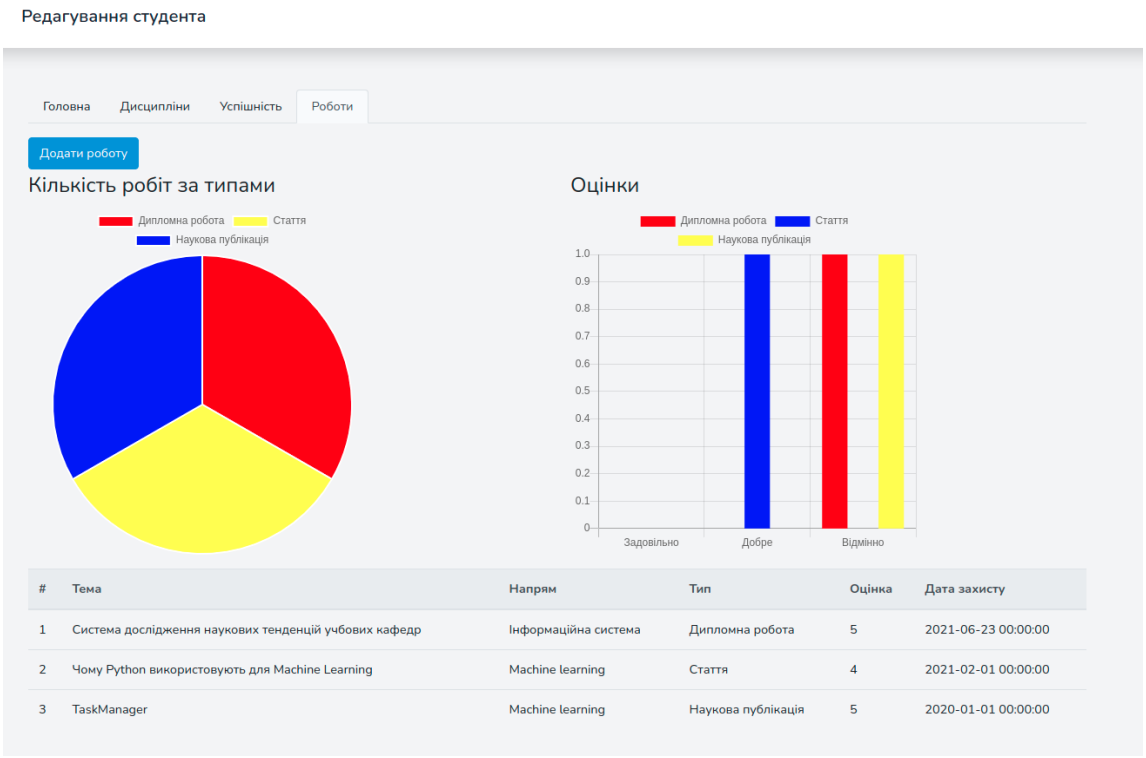


Рисунок 3.25 – Web-сторінка перегляду наукових робіт студента

Для того щоб створити нову групу треба перейти на пункт головного меню «Групи», що зображено на рисунку 3.26.

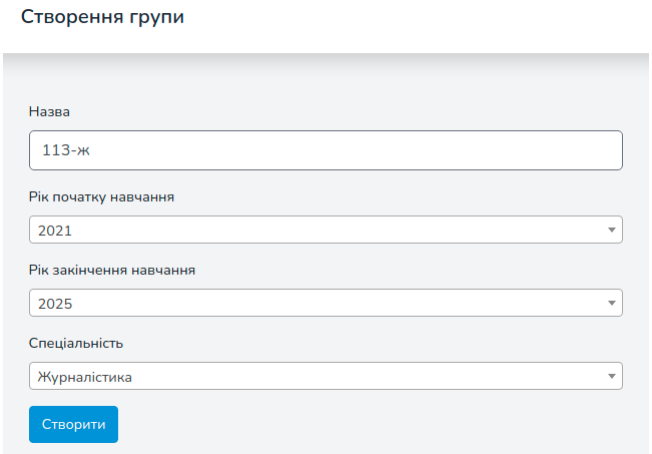


Рисунок 3.25 – Web-сторінка створення групи

Після успішного створення нової групи з’явиться можливість переглядати студентів групи, успішність та статистику по роботам. Перегляд та редагування підключених дисциплін є аналогічним як і на сторінці зі студентами.

Перегляд успішності студентів по групам зображено на рисунку 3.26.

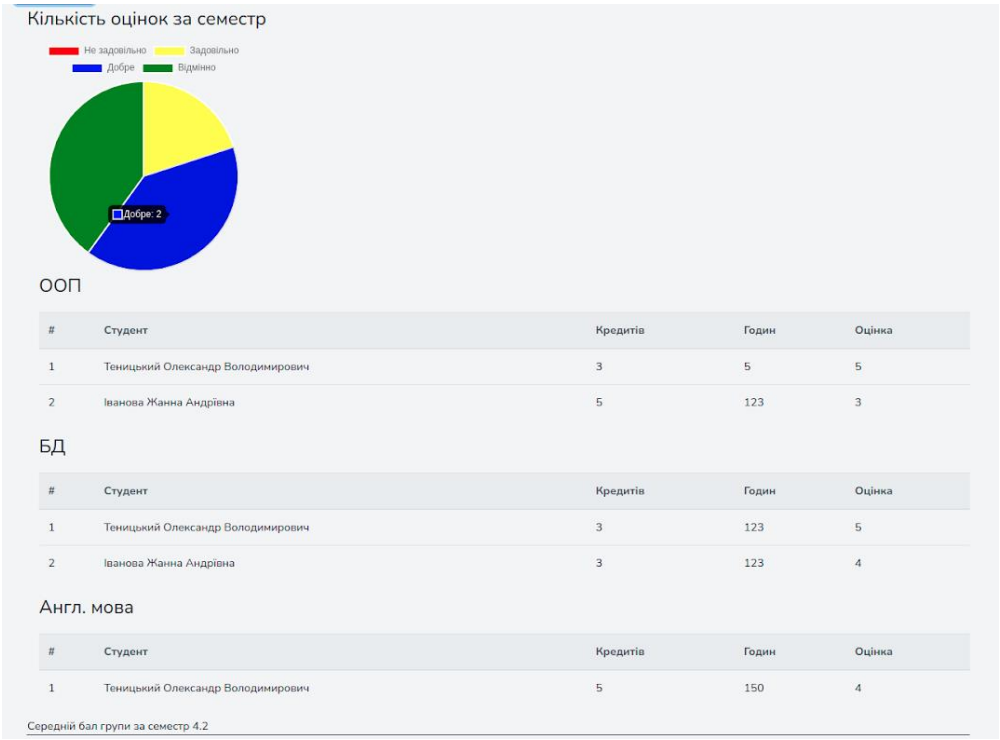


Рисунок 3.26 – Web-сторінка перегляду успішності студентів по групам

За допомогою сторінки редагування групи можна переглянути список студентів, що в ній навчаються (рис.3.27).

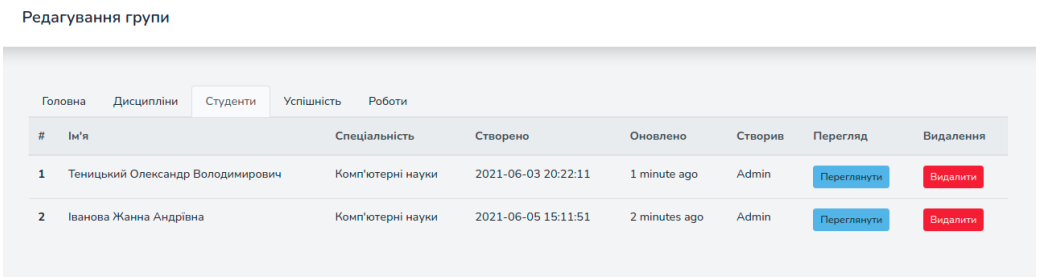


Рисунок 3.27 – Web-сторінка перегляду списку групи

Також на сторінці редагування групи є можливість переглянути статистику по науковим роботам, вона виглядає ідентично пункт меню на сторінці редагування спеціальності, але з іншими даними, які показують тенденцію певної групи.

Web-сторінка зі студентськими роботами складається з фільтру по групі, спеціальності, типу, року захисту та блок з відображення результатів запиту. У інформаційній системі є такі типи робіт як дипломна робота, курсова робота, наукова публікація, стаття та інше (рис.3.28). Реалізовано можливості створення, редагування та видалення студентських робіт (рис.3.29-3.30).

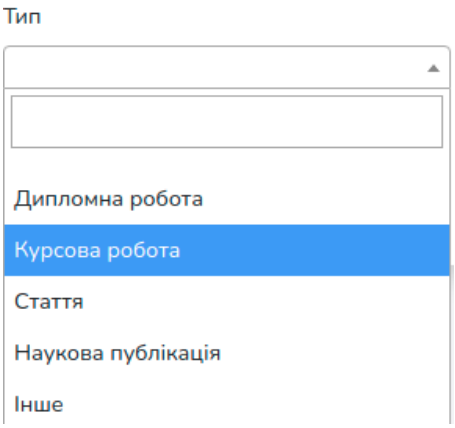


Рисунок 3.28 – Типи студентських робіт

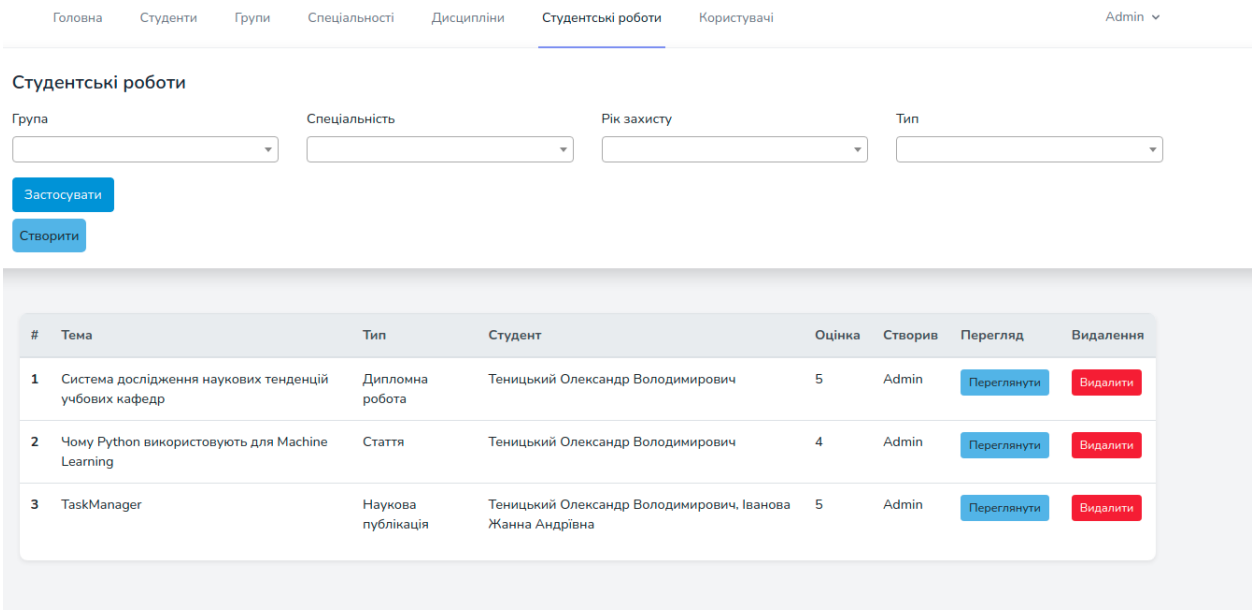


Рисунок 3.29 – Web-сторінка «Роботи»

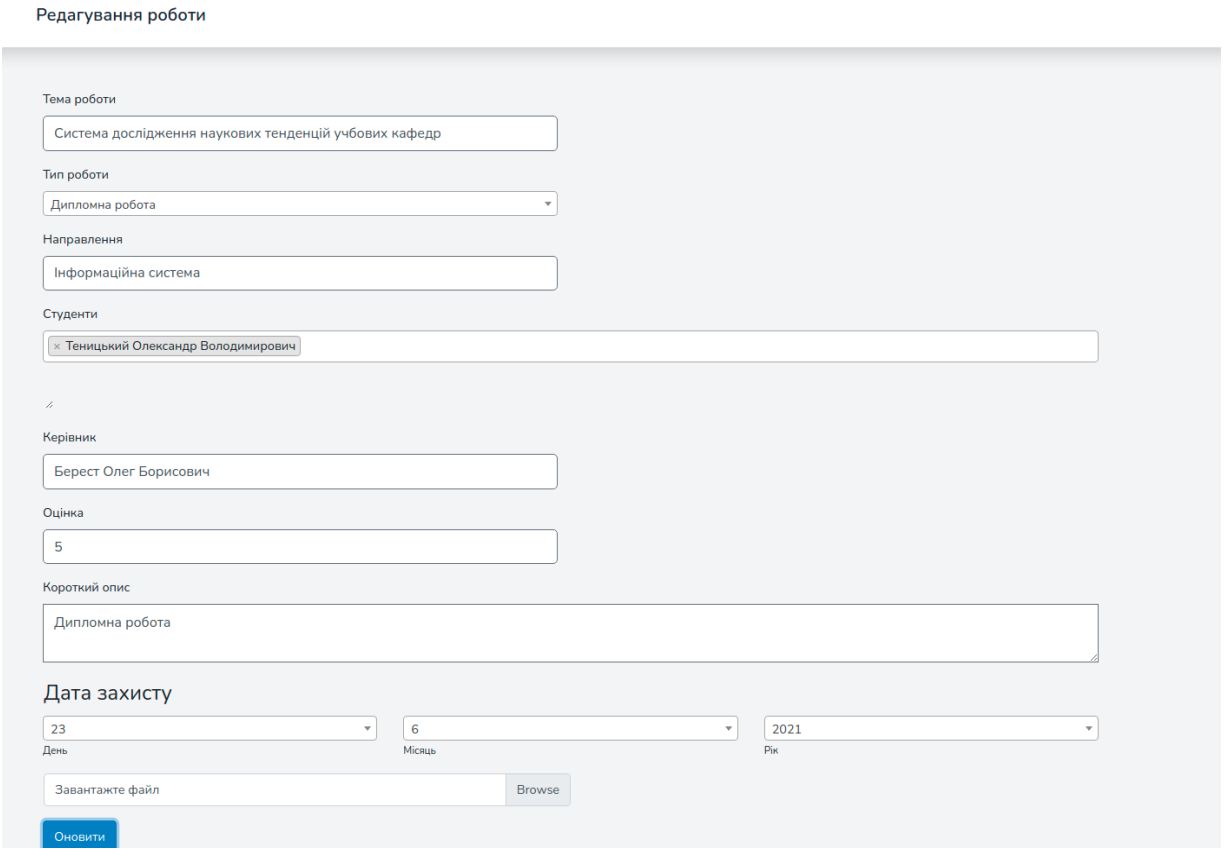


Рисунок 3.30 – Web-сторінка редагування студентських робіт

Для того щоб переглянути або завантажити файл студентської роботи треба перейти за відповідним посланням на сторінці робіт. Web-додаток підтримує такі типи файлів як .pdf, .doc, .docx, .zip, .ppt, .pptx. Web-сторінка перегляду роботи представлена на рисунку 3.31.

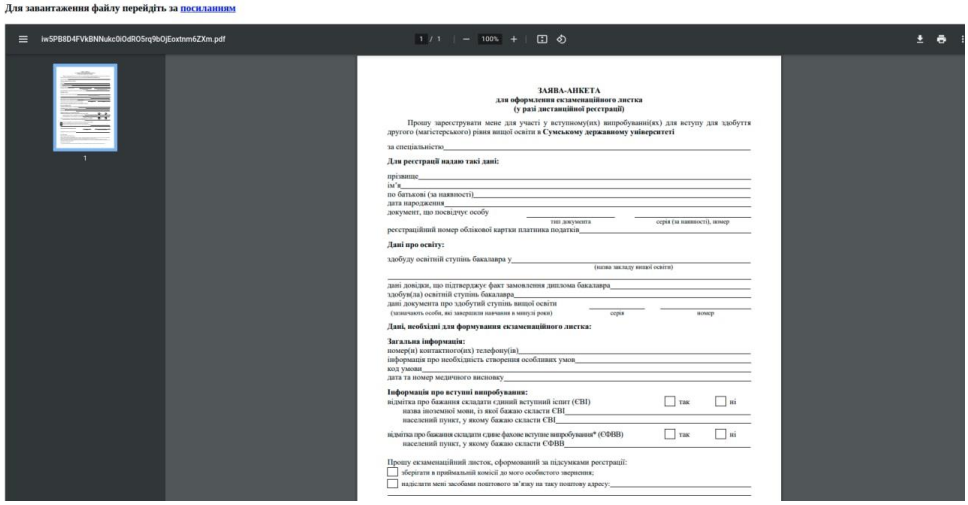


Рисунок 3.31 – Web-сторінка перегляду студентських робіт

На web-сторінці «Користувачі» можна переглядати, створювати та редагувати користувачів системи (рис.3.32-рис.3.33).

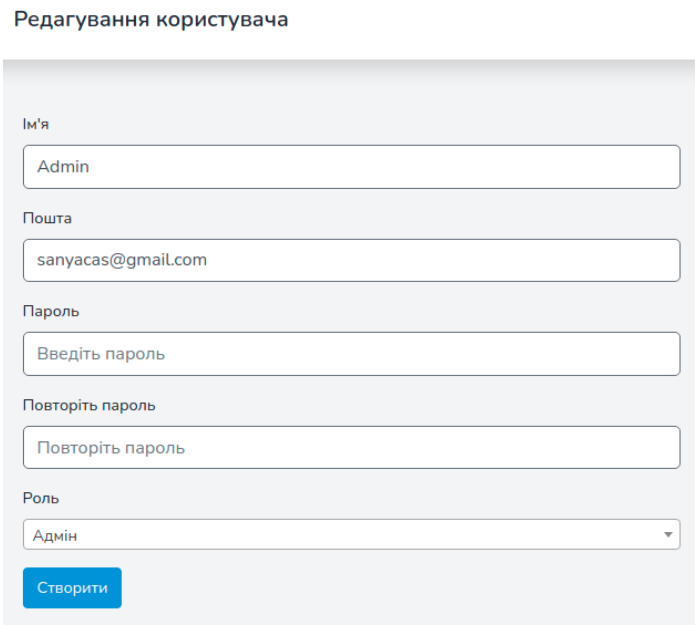


Рисунок 3.32 – Web-сторінка редагування користувача системи

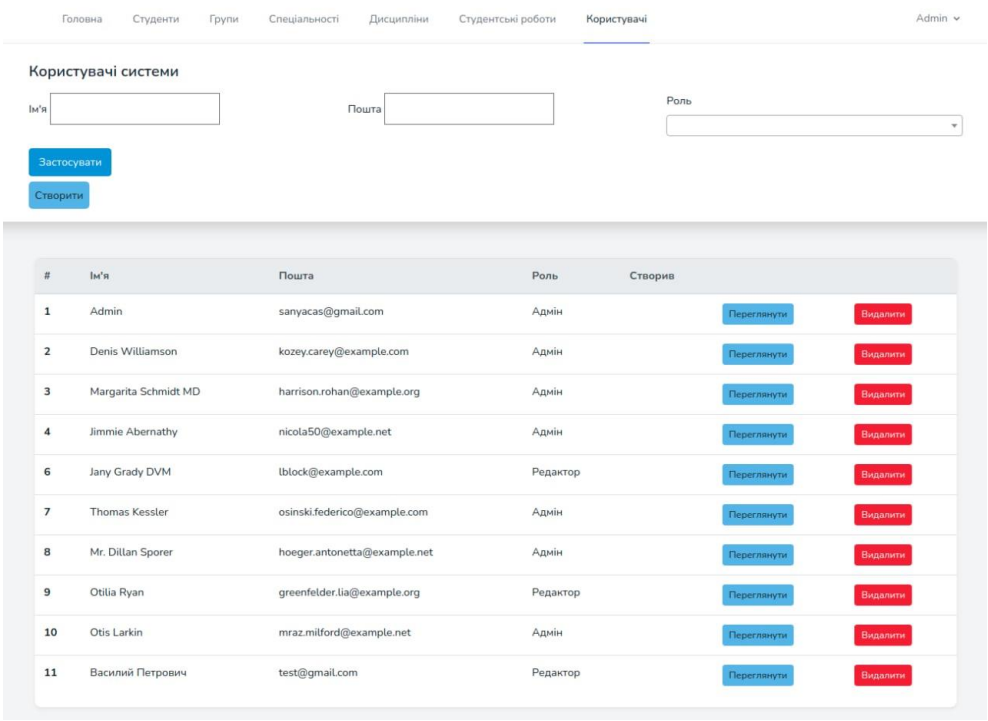


Рисунок 3.33 – Web-сторінка перегляду користувачів системи

Призначенням web-сторінки «Дисципліни» є відображення наявних дисциплін у базі даних, можливості додавання, редагування, видалення та пошуку дисциплін за назвою (рис.3.34-3.35).

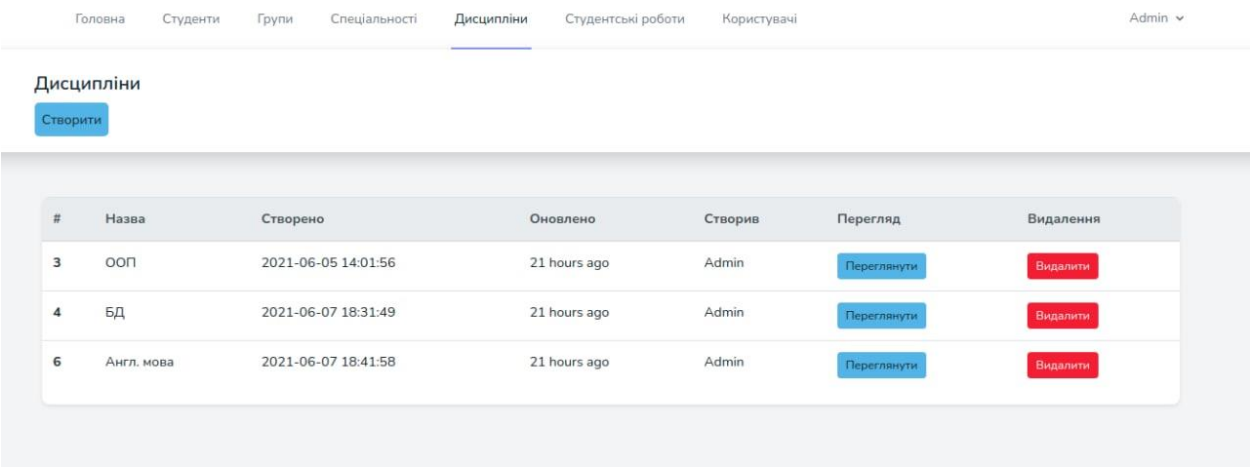


Рисунок 3.34 – Web-сторінка перегляду дисциплін

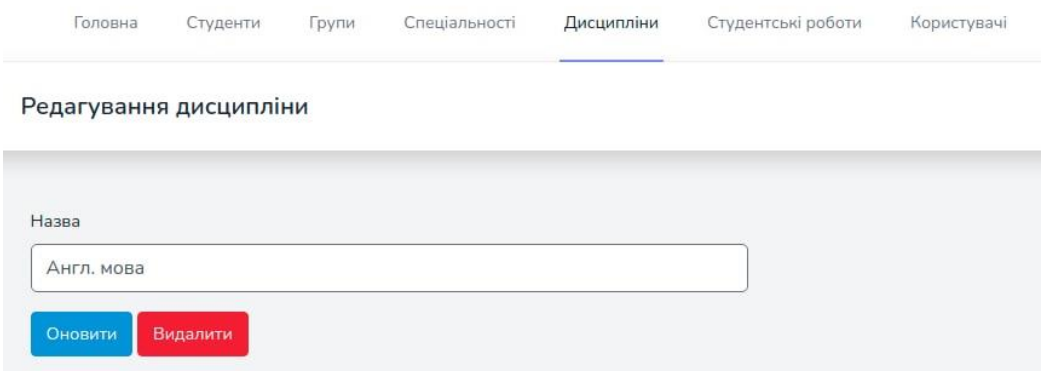


Рисунок 3.35 – Web-сторінка пошуку дисципліни системи

ВИСНОВОК

Результатом виконання кваліфікаційної роботи бакалавра є розроблена інформаційна система аналізу наукових тенденцій учбових кафедр.

Першим етапом виконання дипломного проекту було проведення дослідження предметної області систем обліку. У результаті чого визначено актуальність розробки, поставлено мету і задачі роботи. Наступним етапом досліджень був аналіз аналогів інформаційних систем. Під час виконання якого було сформульовано вимоги до розроблюваного програмного продукту.

У другому розділі було виконано порівняння інструментів вирішення проблеми та обрано наступні технології для розробки інформаційної системи:

- HTML – стандартна мова розмітки web-сторінок в Інтернеті.
- CSS3 – каскадні таблиці стилі, що необхідні для надання web-сторінці візуальних ефектів
- Мова JavaScript + Фреймворк Vue.js – для надання web-сторінкам динамічності та створення інтерфейсу користувача.
- Мова PHP + Фреймворк Laravel – для створення швидкого та надійного бекенду web-додатку.
- MySQL – реляційна база даних для збереження інформації

Під час виконання практичної частини дипломного проекту було створено діаграму High Level Design, реалізовано web-додаток і описано принцип роботи інформаційної системи.

Лістинг програмного коду основних модулів розробленого web-додатку представлено у додатках.

Застосування розробленого web-додатку закладом освіти позитивне вплине на якість роботи учбової частини та допоможе скоротити трудозатрати.

СПИСОК ЛІТЕРАТУРИ

1. Хомишин І. Ю. Сучасні інформаційні технології в освіті [Електронний ресурс] / І. Ю. Хомишин – Режим доступу до ресурсу: http://aphd.ua/publication-157/.
2. Веб-сервіс "Досьє студента" [Електронний ресурс] – Режим доступу до ресурсу: https://asav.hse.ru/documentation/university/6128D609A18E4FB1801147BC1E7CE445.html.
3. Кирик Т. А. Еволюційні процеси програмування [Електронний ресурс] / Т. А. Кирик – Режим доступу до ресурсу: http://ii.npu.edu.ua/files/Zbirnik_KOSN/2/1.pdf.
4. User Interface Design Guidelines: 10 Rules of Thumb [Електронний ресурс] – Режим доступу до ресурсу: https://www.interaction-design.org/literature/article/user-interface-design-guidelines-10-rules-of-thumb.
5. Кузнецов М. Самоучитель PHP 7 / М. Кузнецов, И. Симдянов. – Санкт-Петербург: БХВ-Петербург, 2018. – 450 с.
6. Lopez A. Learning PHP7 / Antonio Lopez., 2018. – 415 с.
7. Pitt C. Pro PHP MVC / Chris Pitt.. – 500 с.
8. Кантор И. Современный JavaScript [Електронний ресурс] / Илья Кантор. – 2018. – Режим доступу до ресурсу: https://learn.javascript.ru/intro.
9. Hanchett E. Vue.js in Action / E. Hanchett, B. Listwon. – New York: Manning Publications, 2019. – 375 с.
10. Коннолли Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Т. Коннолли, К. Бегг., 2016. – 1440 с.

11. Шевчук М. Что такое HLD [Електронний ресурс] / Марина Шевчук – Режим доступу до ресурсу: https://itsource.com.ua/blog/chto-takoe-hld-i-dlya-chego-moget-ponadobitsya/.
12. Exclusive MySQL Performance Tuning Tips For Better Database Optimization [Електронний ресурс] – Режим доступу до ресурсу: https://www.cloudways.com/blog/mysql-performance-tuning/.
13. Нормализация отношений [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/254773/.
14. Apache Software Foundation. Apache HTTP Server Documentation Version 2.5 / Apache Software Foundation.. – 1142 с.
15. Налаштування Git-репозиторію [Електронний ресурс] – Режим доступу до ресурсу: https://www.atlassian.com/ru/git/tutorials/setting-up-a-repository.

ДОДАТКИ

Додаток 1 – Student

```
<?php

class Student extends Model
{
 use HasFactory;

 const FULL_TIME_FORM = 0;
 const PART_TIME_FORM = 1;
 const REMOTE_FORM = 2;

 protected $fillable = ['group_id', 'student_ticket_id', 'first_name', 'last_name',
 'father_name', 'date_of_birth', 'gender', 'city_of_residence', 'street_of_residence',
'house_of_residence', 'flat_of_residence', 'city_of_birth',
 'nationality', 'school', 'year_of_graduation_school', 'mobile_phone', 'benefits',
 'date_of_enrollment', 'enrollment_number', 'tax_code', 'form_of_education',
 'passport_id', 'photo', 'year_of_graduation', 'created_by'];

 protected $dates = [
 'date_of_birth', 'date_of_enrollment'
 ];

 public function getFullNameAttribute(): string
 {
 return $this->first_name.' '.$this->last_name.' '.$this->father_name;
 }

 public function group(): BelongsTo
 {
 return $this->belongsTo(Group::class, 'group_id');
 }

 public function disciplines(): BelongsToMany
 {
 return $this->belongsToMany(Discipline::class, 'student_discipline', 'student_id',
'discipline_id')->withPivot('hours', 'credits', 'mark', 'semester');
 }

 public function createdBy(): BelongsTo
 {
 return $this->belongsTo(User::class, 'created_by');
 }

 public function getSemestersAttribute(): float|int
 {
 $startYear = $this->group->start_year;
 return (now()->year - $startYear + 1) * 2;
 }

 public function works(): BelongsToMany
 {
 return $this->belongsToMany(Work::class, 'student_work', 'student_id', 'work_id');
 }

 public function getStudyAttribute()
 {
 if($this->year_of_graduation > now())
 return false;
 return true;
 }
}
```

Додаток 2 – Group

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Relations\BelongsToMany;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Group extends Model
{
 use HasFactory;

 protected $fillable = ['name', 'department_id', 'start_year', 'graduate_year',
'created_by'];

 public function department(): BelongsTo
 {
 return $this->belongsTo(Department::class, 'department_id');
 }

 public function students(): HasMany
 {
 return $this->hasMany(Student::class, 'group_id');
 }

 public function disciplines(): BelongsToMany
 {
 return $this->belongsToMany(Discipline::class, 'group_discipline', 'group_id',
'discipline_id')->withPivot(['hours', 'credits', 'semester']);
 }

 public function createdBy(): BelongsTo
 {
 return $this->belongsTo(User::class, 'created_by');
 }

 public function getSemestersAttribute(): float|int
 {
 $startYear = $this->start_year;
 return (now()->year - $startYear + 1) * 2;
 }}
```

Додаток 3 – WorkRepository

```
<?php
namespace App\Repository;
use App\Models\Group;
use App\Models\Student;
use App\Models\Work;
use Illuminate\Database\Query\Builder;
use Illuminate\Support\Collection;
use Illuminate\Support\Facades\DB;
class WorkRepository
{
 protected Work $model;
 public function __construct(Work $discipline)
 {
 $this->model = $discipline;
 }
 public function getStatsPerYears(array $filter = []): ?Collection
 {
 return DB::table('works')
 ->select(DB::raw('count(*) total'), DB::raw('YEAR(defence) year'))
 ->when(isset($filter['group']), function(Builder $query) use ($filter) {
 $query->join('student_work as sw', 'sw.work_id', '=', 'works.id')
 ->join('students', 'students.id', '=', 'sw.student_id')
 ->where('students.group_id', $filter['group']);
 })
 ->when(isset($filter['department']), function(Builder $query) use ($filter) {
 $query->join('student_work as sw', 'sw.work_id', '=', 'works.id')
 ->join('students', 'students.id', '=', 'sw.student_id')
 ->join('groups', 'groups.id', '=', 'students.group_id')
 ->where('groups.department_id', $filter['department']);
 })
 ->whereYear('defence', '>', now()->year-6)
 ->groupBy('year')
 ->orderBy('year')
 ->get();
 }
```

```
 public function getStatsPerYearsForTypes(array $filter = []): ?Collection
 {
 return DB::table('works')
 ->select(DB::raw('count(*) total'), DB::raw('YEAR(defence) year'), 'type')
 ->when(isset($filter['group']), function(Builder $query) use ($filter) {
 $query->join('student_work as sw', 'sw.work_id', '=', 'works.id')
 ->join('students', 'students.id', '=', 'sw.student_id')
 ->where('students.group_id', $filter['group']);
 })
 ->when(isset($filter['department']), function(Builder $query) use ($filter) {
 $query->join('student_work as sw', 'sw.work_id', '=', 'works.id')
 ->join('students', 'students.id', '=', 'sw.student_id')
 ->join('groups', 'groups.id', '=', 'students.group_id')
 ->where('groups.department_id', $filter['department']);
 })
 ->whereYear('defence', '>', now()->year-6)
 ->groupBy('type', 'year')
 ->orderBy('year')
 ->get();
 }
}
```

Додаток 4 – MultipleLineWorkDiagram.vue

```
<script>
import {Line} from "vue-chartjs";

export default {
 extends: Line,
 props: ["diagram-data", "labels-prop"],
 data() {
 return {
 my_datasets: []
 }
 },
 mounted () {
 this.prepareDatasets();
 this.renderChart({
 labels: this.labelsProp,
 datasets: this.my_datasets,
 }, {responsive: false});
 },
 methods: {
 prepareDatasets() {
 let colors = ['red', 'blue', 'yellow', 'pink', 'dark'];
 let types = ['Дипломна робота','Курсова робота','Стаття','Наукова
публікація','Інше'];
 let works = [];
 for(let i = 0; i < 5; ++i) {
 works[types[i]] = [];
 works[types[i]] = this.diagramData.filter((item) => item.type === i);
 }
 let i = 0;
 for(let work in works) {
 let dataset = {};
 dataset.label = work;
 dataset.backgroundColor = colors[i];
 dataset.data = [];
 for(let i = 0; i < works[work].length; ++i) {
 let startYear = new Date().getFullYear();
 for(let j = startYear-6; j <= startYear; ++j) {
 if(works[work].find((item) => item.year === j))
dataset.data.push(works[work][i].total);
 else dataset.data.push(0);
 }
 }
 this.my_datasets.push(dataset);
 ++i;
 }
 }
 }
}
</script>
```

Додаток 5 – student.create.blade.php

```
@extends('admin.layouts.admin')

@section('header-title')
 {{ __("Створення студента") }}
@endsection

@section('content')
 <div class="py-12 max-w-7xl mx-auto sm:px-6 lg:px-8 admin-form">
 <form method="post" action="{{ route('students.store') }}" enctype="multipart/form-
data">
 @csrf
 <div class="form-group">
 <div class="row">
 <div class="col-4">
 <label for="last-name">Прізвище</label>
 <input type="text" name="first_name" class="form-control" id="last-name" value="{{
old('first_name') ?? '' }}" placeholder="Введіть прізвище" required>
 </div>
 <div class="col-4">
 <label for="first-name">Ім'я</label>
 <input type="text" name="last_name" class="form-control" id="first-name" value="{{
old('last_name') ?? '' }}" placeholder="Введіть ім'я" required>
 </div>
 <div class="col-4">
 <label for="father-name">По-батькові</label>
 <input type="text" name="father_name" class="form-control" id="father-name" value="{{
old('father_name') ?? '' }}" placeholder="Введіть по-батькові">
 </div>
 </div>
 </div>
 <div class="form-group">
 <div class="row">
 <div class="col-4">
 <label for="student-ticket-id">Номер студентського квитка</label>
 <input type="text" name="student_ticket_id" class="form-control" id="student-ticket-
id" value="{{ old('student_ticket_id') ?? '' }}" placeholder="Введіть номер
студентського квитка">
 </div>
 <div class="col-4">
 <label for="passport-id">Номер студентського квитка</label>
 <input type="text" name="passport_id" class="form-control" id="passport-id" value="{{
old('passport_id') ?? '' }}" placeholder="Введіть номер паспорту">
 </div>
 <div class="col-4">
 <label for="tax-code">ІПН</label>
 <input type="text" name="tax_code" class="form-control" id="tax-code" value="{{
old('tax_code') ?? '' }}" placeholder="Введіть ІПН">
 </div>
 </div>
 </div>
 <div class="form-group">
 <div class="row">
 <div class="col">
 <label for="group">Група</label><br>
 <select id="group" name="group_id" class="form-control w-50" required>
 @foreach($groups as $group)
 <option value="{{ $group->id }}" {{ old('group_id') == $group->id ? 'selected' : ''
}}>{{ $group->name }}</option>
 @endforeach
 </select>
 </div>
 </div>
 </div>
 <div class="form-group">
```

```
 <div class="row">
 <div class="col-12 col-lg-6">
 <input type="file" name="photo" id="custom-file" class="custom-file-input">
 <label class="ml-3 custom-file-label" for="custom-file">Завантажте фото</label>
 </div>
 </div>
 </div>
 <div class="form-group">
 <div class="row">
 <div class="col-12 col-lg-4">
 <div class="row">
 <legend class="px-3">Дата народження</legend>
 <div class="col-4">
 <select class="form-control" name="date_of_birth[day]" aria-describedby="day-help"
required>
 @foreach(range(1, 31) as $day)
 <option value="{{ $day }}" {{ old('birth.day') == $day ? 'selected' : '' }}>{{ $day
}}</option>
 @endforeach
 </select>
 <small id="day-help">День</small>
 </div>
 <div class="col-4">
 <select class="form-control" name="date_of_birth[month]" aria-describedby="month-
help" required>
 @foreach(range(1, 12) as $month)
 <option value="{{ $month }}" {{ old('birth.month') == $month ? 'selected' : '' }}>{{
$month }}</option>
 @endforeach
 </select>
 <small id="month-help">Місяць</small>
 </div>

 <div class="col-4">
 <select class="form-control" name="date_of_birth[year]" aria-describedby="year-help"
required>
 @foreach(range(now()->year-15, now()->year-70) as $year)
 <option value="{{ $year }}" {{ old('birth.year') == $year ? 'selected' : '' }}>{{
$year }}</option>
 @endforeach
 </select>
 <small id="year-help">Рік</small>
 </div>
 </div>
 </div>
 <div class="col-12 col-lg-2">
 <legend>Стать</legend>
 <label for="gender-male">Чоловіча</label>
 <input type="radio" name="gender" class="form-control" {{ old('gender') == 'male' ?
'checked' : '' }} id="gender-male" value="male" placeholder="Виберіть стать" required>
 <label for="gender-male">Жіноча</label>
 <input type="radio" name="gender" class="form-control" {{ old('gender') == 'female' ?
'checked' : '' }} id="gender-female" value="female" placeholder="Виберіть стать"
required>
 </div>
 </div>
 </div>
 <div class="form-group">
 <div class="row">
 <div class="col-6">
 <label for="city-of-birth">Місто народження</label>
 <input type="text" name="city_of_birth" class="form-control" value="{{
old('city_of_birth') ?? '' }}" id="city-of-birth" placeholder="Введіть місце
народження" required>
 </div>
 <div class="col-6">
 <label for="nationality">Національність</label>
```

```
 <input type="text" name="nationality" class="form-control" value="{{
old('nationality') ?? '' }}" id="nationality" placeholder="Введіть національність"
required>
 </div>
 </div>
 </div>
 <div class="form-group">
 <div class="row">
 <div class="col-12">
 <div class="row">
 <div class="col-3">
 <label for="city-of-residence">Місто проживання</label>
 <input type="text" name="city_of_residence" class="form-control" value="{{
old('city_of_residence') ?? '' }}" id="city-of-birth" placeholder="Введіть місце
проживання" required>
 </div>
 <div class="col-3">
 <label for="city-of-к">Вулиця</label>
 <input type="text" name="street_of_residence" class="form-control" value="{{
old('street_of_residence') ?? '' }}" id="city-of-birth" placeholder="Введіть вулицю">
 </div>
 <div class="col-3">
 <label for="city-of-birth">Будинок</label>
 <input type="text" name="house_of_residence" class="form-control" value="{{
old('house_of_residence') ?? '' }}" id="city-of-birth" placeholder="Введіть будинок">
 </div>
 <div class="col-3">
 <label for="city-of-birth">Квартира</label>
 <input type="text" name="flat_of_residence" class="form-control" value="{{
old('flat_of_residence') ?? '' }}" id="city-of-birth" placeholder="Введіть квартиру">
 </div>
 </div>
 </div>
 </div>
 </div>
 <div class="form-group">
 <div class="row">
 <div class="col-6">
 <label for="school">Школа</label>
 <input type="text" name="school" class="form-control" id="school" value="{{
old('school') ?? '' }}" placeholder="Введіть школу" required>
 </div>
 <div class="col-6">
 <label for="year-of-graduation-school">Рік випуску зі школи</label>
 <select name="year_of_graduation_school" class="form-control" id="year-of-graduation-
school" required>
 @foreach(range(now()->year, now()->year-70) as $year)
 <option value="{{ $year }}" {{ old('year_of_graduation_school') == $year ? 'selected'
: '' }}>
 {{ $year }}
 </option>
 @endforeach
 </select>
 </div>
 </div>
 </div>
 <div class="form-group">
 <div class="row">
 <div class="col-6">
 <label for="enrollment_number">Зараховано наказом №</label>
 <input type="text" name="enrollment_number" class="form-control" value="{{
old('enrollment_number') ?? '' }}" id="enrollment_number" placeholder="Введіть номер
наказу">
 </div>
 <div class="col-6">
 <div class="row">
 <legend class="px-3">Дата зарахування</legend>
```

```
 <div class="col-4">
 <select class="form-control" name="date_of_enrollment[day]" id="day-of-enrollment"
aria-describedby="day-help">
 @foreach(range(1, 31) as $day)
 <option value="{{ $day }}" {{ old('date_of_enrollment.day') == $day ? 'selected' : ''
}}>{{ $day }}</option>
 @endforeach
 </select>
 <small id="day-help">День</small>
 </div>
 <div class="col-4">
 <select class="form-control" name="date_of_enrollment[month]" id="month-of-birth"
aria-describedby="month-help">
 @foreach(range(1, 12) as $month)
 <option value="{{ $month }}" {{ old('date_of_enrollment.month') == $month ?
'selected' : '' }}>{{ $month }}</option>
 @endforeach
 </select>
 <small id="month-help">Місяць</small>
 </div>

 <div class="col-4">
 <select class="form-control" name="date_of_enrollment[year]" id="year-of-birth" aria-
describedby="year-help">
 @foreach(range(now()->year, now()->year-70) as $year)
 <option value="{{ $year }}" {{ old('date_of_enrollment.year') == $year ? 'selected' :
'' }}>{{ $year }}</option>
 @endforeach
 </select>
 <small id="year-help">Рік</small>
 </div>
 </div>
 </div>
 </div>
 </div>
 <div class="form-group">
 <div class="row">
 <div class="col-6">
 <label for="city-of-birth">Форма навчання</label>
 <select name="form_of_education" class="form-control" required>
 @foreach(__('site.students.form_of_education') as $key => $value)
 <option {{ old('form_of_education') == $key ? 'selected' : '' }} value="{{ $key
}}">{{ $value }}</option>
 @endforeach
 </select>
 </div>
 <div class="col-6">
 <label for="year-of-graduation">Рік випуску</label>
 <select id="year-of-graduation" name="year_of_graduation" class="form-control">
 @foreach(range(now()->year, now()->year-70) as $year)
 <option value="{{ $year }}" {{ old('year_of_graduation') == $year ? 'selected' : ''
}}>{{ $year }}</option>
 @endforeach
 </select>
 </div>
 </div>
 </div>
 <div class="form-group">
 <div class="row">
 <div class="col-12">
 <label for="city-of-birth">Пільги</label>
 <textarea name="benefits" class="form-control">{{ old('benefits') ?? '' }}</textarea>
 </div>
 </div>
 </div>
 <button type="submit" class="btn btn-primary">Створити</button>
 </form> </div>@endsection()
```

Додаток 6 – StudentDisciplineResource

```
<?php

namespace App\Http\Resources\Group;

use App\Models\Student;
use App\Repository\DisciplineRepository;
use Illuminate\Http\Resources\Json\JsonResource;
use Illuminate\Http\Resources\Json\ResourceCollection;

class StudentDisciplineResource extends JsonResource
{
 public function toArray($request)
 {
 $preparedArray = [
 'student_id' => $this->student_id,
 'discipline_id' => $this->id,
 'name' => $this->full_name,
 'hours' => $this->hours,
 'semester' => $this->semester,
 'mark' => $this->mark,
 'credits' => $this->credits,
 ];

 $disciplineRepository = app(DisciplineRepository::class);
 $preparedArray['users_discipline'] = $disciplineRepository-
>getDisciplineInGroups($this->id)->count() == 0;
 return $preparedArray;
 }
}
```

Додаток 7 – StudentSaveRequest

```
<?php
namespace App\Http\Requests\Admin\Students;
use Illuminate\Foundation\Http\FormRequest;
class StudentSaveRequest extends FormRequest
{
 public function authorize(): bool
 {
 return true;
 }

 public function rules(): array
 {
 $currentYear = now()->year;
 return [
 'group_id' => 'required|integer|exists:groups,id',
 'student_ticket_id' => 'nullable|max:30|unique:students,student_ticket_id',
 'first_name' => 'required|string|max:30',
 'last_name' => 'required|string|max:30',
 'father_name' => 'nullable|string|max:30',
 'date_of_birth.day' => 'integer|min:1|max:31',
 'date_of_birth.month' => 'integer|min:1|max:12',
 'date_of_birth.year' => 'integer|min:'.($currentYear-70).'|max:'.($currentYear-15),
 'gender' => 'in:male,female',
 'flat_of_residence' => 'nullable|string|max:100',
 'street_of_residence' => 'nullable|string|max:100',
 'house_of_residence' => 'nullable|string|max:50',
 'city_of_residence' => 'required|string|max:50',
 'city_of_birth' => 'required|string|max:50',
 'nationality' => 'required|string|max:50',
 'school' => 'required|string|max:50',
 'year_of_graduation_school' => 'required|integer|min:'.($currentYear-
70).'|max:'.$currentYear,
 'mobile_phone' => 'integer|nullable|max:12|min:12',
 'benefits' => 'nullable|string',
 'date_of_enrollment.day' => 'integer|min:1|max:31',
 'date_of_enrollment.month' => 'integer|min:1|max:12',
 'date_of_enrollment.year' => 'integer|min:'.($currentYear-70).'|max:'.$currentYear,
 'enrollment_number' => 'string|nullable|max:50',
 'tax_code' => 'string|nullable|max:50',
 'form_of_education' => 'required|in:0,1,2|integer',
 'passport_id' => 'nullable|string|max:20',
 'photo' => 'image|mimes:jpeg,png,jpg|max:2048',
 'year_of_graduation' => 'integer|min:'.($currentYear-70).'|max:'.$currentYear,
 ];
 }
}
```

Додаток 8 – StudentFilter

```
<?php

namespace App\Editing;

use App\Models\Department;
use App\Models\Group;
use App\Models\Student;
use Illuminate\Contracts\Pagination\LengthAwarePaginator;
use Illuminate\Database\Eloquent\Collection;
use Illuminate\Http\Request;
use Illuminate\Support\Fluent;

class StudentFilter
{
 protected ?array $filter;

 protected Student $model;

 public function __construct(Request $request)
 {
 $this->filter = $request->filter;
 $this->model = new Student();
 }

 public function getPaginator(): LengthAwarePaginator
 {
 $builder = $this->model;

 $filter = $this->filter();

 if($fullName = $filter->full_name) {
 $names = explode(' ', $fullName);
 foreach ($names as $name) {
 $builder = $builder->where('first_name', 'like', '%'.$name.'%')
 ->orWhere('last_name', 'like', '%'.$name.'%')
 ->orWhere('father_name', 'like', '%'.$name.'%');
 }
 }

 if($group = $filter->group)
 $builder = $builder->where('group_id', $group);

 if($department = $filter->department)
 $builder = $builder->join('groups', 'groups.id', '=', 'group_id')
 ->join('departments', 'departments.id', '=', 'groups.department_id')
 ->where('departments.id', $department);

 if($graduationYear = $filter->graduation_year)
 $builder = $builder->whereYear('year_of_graduation', $graduationYear);

 if(($formOfEducation = $filter->form_of_education) != null) {
 $builder = $builder->where('form_of_education', $formOfEducation);
 }

 if($study = $filter->study) {
 if($study == 'studying')
 $builder = $builder->whereDate('year_of_graduation', '<', now());
 else if($study == 'graduated')
 $builder = $builder->whereDate('year_of_graduation', '>', now());
 }

 return $builder->paginate(config('app.pagination.students'));
 }
```

```
 public function getGroups(): \Illuminate\Support\Collection
 {
 $collection = collect('');
 return $collection->merge(Group::all());
 }

 public function getDepartments(): \Illuminate\Support\Collection
 {
 $collection = collect('');
 return $collection->merge(Department::all());
 }

 protected function filter(): Fluent
 {
 $fluent = new Fluent();
 $fluent->full_name = $this->filter['full_name'] ?? null;
 $fluent->group = $this->filter['group'] ?? null;
 $fluent->department = $this->filter['department'] ?? null;
 $fluent->graduation_year = $this->filter['graduation_year'] ?? null;
 $fluent->form_of_education = $this->filter['form_of_education'] ?? null;
 $fluent->study = $this->filter['study'] ?? null;

 return $fluent;
 }
}
```

Додаток 9 – StudentDesciplineController

```
<?php
namespace App\Http\Controllers\Api;
use App\Http\Resources\Student\StudentDisciplineResource;
use App\Models\Discipline;
use App\Models\Group;
use App\Models\Student;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Request;
class StudentDisciplineController
{
 public function addDiscipline(Student $student, Request $request): JsonResponse
 {
 $attributes = $request->validate([
 'discipline_id' => 'required|integer',
 'hours' => 'required|integer',
 'credits' => 'required|integer',
 'semester' => 'required|integer'
 ]);

 $disciplineId = $attributes['discipline_id'];
 $hours = $attributes['hours'];
 $credits = $attributes['credits'];
 $semester = $attributes['semester'];

 if(!$disciplineId)
 abort(404);

 try {
 $student->disciplines()->attach([$disciplineId => ['hours' => $hours, 'credits' =>
$credits, 'semester' => $semester]]);
 } catch (\Exception $e) {}

 $discipline = $student->disciplines()->where('discipline_id', $disciplineId)-
>first();
 $discipline = new StudentDisciplineResource($discipline);
 return response()->json(['discipline' => $discipline]);
 }

 public function update(Student $student, Request $request): JsonResponse
 {
 $attributes = $request->validate([
 'id' => 'required|integer',
 'credits' => 'integer',
 'hours' => 'integer',
 'semester' => 'integer',
 'mark' => 'integer'
 ]);
 $id = $attributes['id'];
 unset($attributes['id']);
 $student->disciplines()->updateExistingPivot($id, $attributes);
 return response()->json('ok');
 }
 public function removeDiscipline(Student $student, Request $request): JsonResponse
 {
 $attributes = $request->validate([
 'discipline_id' => 'required|integer'
 ]);
 $student->disciplines()->detach($attributes['discipline_id']);
 return response()->json('ok');
 }
}
```

Додаток 10 – CreateDisciplinesTable

```
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateDisciplinesTable extends Migration
{
 public function up()
 {
 Schema::create('disciplines', function(Blueprint $table) {
 $table->increments('id');
 $table->string('name');

 $table->bigInteger('created_by')->unsigned();
 $table->foreign('created_by')->on('users')->references('id');

 $table->timestamps();
 });

 Schema::create('group_discipline', function(Blueprint $table) {
 $table->integer('discipline_id')->unsigned();
 $table->foreign('discipline_id')->on('disciplines')->references('id')-
>onUpdate('cascade')->onDelete('cascade');
 $table->integer('group_id')->unsigned();
 $table->foreign('group_id')->on('groups')->references('id')->onUpdate('cascade')-
>onDelete('cascade');

 $table->integer('semester');
 $table->integer('credits');
 $table->integer('hours');

 $table->primary(['discipline_id', 'group_id']);

 $table->timestamps();
 });

 Schema::create('student_discipline', function(Blueprint $table) {
 $table->integer('discipline_id')->unsigned();
 $table->foreign('discipline_id')->on('disciplines')->references('id')-
>onUpdate('cascade')->onDelete('cascade');
 $table->integer('student_id')->unsigned();
 $table->foreign('student_id')->on('students')->references('id')->onUpdate('cascade')-
>onDelete('cascade');

 $table->integer('credits');
 $table->integer('hours');
 $table->integer('semester');
 $table->integer('mark')->nullable();

 $table->primary(['discipline_id', 'student_id']);
 });

 }

 public function down()
 {
 Schema::drop('student_discipline');
 Schema::drop('group_discipline');
 Schema::drop('disciplines');
 }
}
```