Міністерство освіти і науки України
Сумський державний університет
Навчально-науковий інститут бізнесу, економіки та менеджменту
Кафедра економічної кібернетики

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему «Автоматизація діяльності співробітників середніх шкіл»

Виконав студент 2 курсу, групи ЕКм-91н.а
(номер курсу) (шифр групи)
Спеціальності 051 «Економіка»
(«Економічна кібернетика»)
_____Ситник В. М._____
(прізвище, ініціали студента)
Керівник професор, д.е.н. Олійник В. М.
(посада, науковий ступінь, прізвище, ініціали)

Суми – 2021 рік

РЕФЕРАТ

кваліфікаційної магістерської роботи на тему

«Автоматизація діяльності співробітників середніх шкіл»

студента Ситника Владислава Миколайовича

(прізвище, ім'я, по батькові)

Актуальність теми, обраної для дослідження, визначається тим, що в умовах діджиталізіції, а також умовах коронавірусу абсолютно першочерговою стає задача відмови ведення друкованих відомостей та автоматизація діяльності шляхом впровадження сучасних інформаційних технологій. Тому що, щоб задовольнити попередні проблеми був розроблений веб-додаток для автоматизації діяльності співробітників середніх шкіл.

Метою кваліфікаційної магістерської роботи є розробка веб-орієнтованої інформаційної системи для автоматизації діяльності співробітників середніх шкіл.

Об'єктом дослідження є діяльність співробітників середніх шкіл.

Предметом дослідження є шляхи реалізації системи для розробки веб-орієнтованої інформаційної системи для автоматизації діяльності співробітників середніх шкіл.

Задачами дослідження є аналіз стану автоматизації бізнес-процесів, розробка вимог до створюваної системи, проектування, реалізація та тестування веб-орієнтованої інформаційної системи для автоматизації діяльності співробітників середніх шкіл.

Інформаційною базою кваліфікаційної магістерської роботи є результати проходження переддипломної практики, що включає певний набір даних про створення веб-орієнтованих продуктів на базі ТОВ «Сігма Софтвеа» та інтернет-джерела, які надають інформацію щодо створення веб-застосунків.

Основний практичний результат кваліфікаційної магістерської роботи полягає у розробці веб-орієнтованої інформаційної системи для автоматизації діяльності співробітників середніх шкіл.

Одержані результати можуть бути використані в подальшій діяльності ТОВ «Сігма Софтвеа», при здійсненні реалізації продукції.

Ключові слова: веб-орієнтована система, Python, автоматизація, середня школа, веб-браузер, MySQL.

Зміст кваліфікаційної магістерської роботи викладено на 77 сторінках. Список використаних джерел із 43 найменувань, розміщений на 3 сторінках. Робота містить 14 таблиць, 38 рисунків, а також 6 додатків, розміщених на 13 сторінках.

Рік виконання кваліфікаційної роботи – 2021рік.

Рік захисту роботи – 2021рік.

Міністерство освіти і науки України

Сумський державний університет

Навчально-науковий інститут бізнесу, економіки та менеджменту

Кафедра економічної кібернетики

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.е.н., професор
_____О.В. Кузьменко
“__” _____ 2021 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ МАГІСТЕРСЬКУ РОБОТУ

(спеціальність 051 «Економіка («Економічна кібернетика»)»)

студенту 2 курсу, групи ЕК.м-91н.а

_____Ситника Владислава Миколайовича_____
(прізвище, ім’я, по батькові студента)

1. Тема роботи Автоматизація діяльності співробітників середніх шкіл_____
_____

затверджена наказом по університету від «___» _____ 20__ року №_____

2. Термін подання студентом закінченої роботи «__» травня 2021 року.

3. Мета кваліфікаційної роботи Розробка веб-орієнтованої інформаційної системи для автоматизації діяльності співробітників середніх шкіл_____

4. Об’єкт дослідження Діяльність співробітників середніх шкіл

5. Предмет дослідження Шляхи реалізації системи для розробки веб-орієнтованої інформаційної системи для автоматизації діяльності співробітників середніх шкіл.

6. Кваліфікаційна робота виконується на матеріалах ТОВ «Сігма Софтвеа» та інтернет-джерелах, які надають інформацію щодо створення веб-застосунків

7. Орієнтовний план кваліфікаційної роботи, терміни подання розділів керівникові та зміст завдань для виконання поставленої мети

Розділ 1 Аналіз предметної області та постановка задачі дослідження

(назва – термін подання)

У розділі 1 Провести огляд предметної області, навести приклади існуючих рішень та порівняти їх, описати постановку задачі дослідження.

(зміст конкретних завдань до розділу, які повинен виконати студент)

Розділ 2 Розробка математичного, алгоритмічного та програмно архітектурного забезпечення вирішення задачі автоматизації діяльності співробітників середніх шкіл

(назва – термін подання)

У розділі 2 Здійснити прогнозування за допомогою методу Хольта, проаналізувати підходи до підвищення забезпечення якості автоматизації, описати основні вимоги до прикладного програмного забезпечення, розробити структуру БД, вибрати інструментальні засоби для реалізації

(зміст конкретних завдань до розділу, які має виконати студент)

Розділ 3 Розробка програмного забезпечення для вирішення задачі автоматизації діяльності співробітників середніх шкіл

(назва – термін подання)

У розділі 3 Розробити структуру створення та редагування задач та шаблонів, редагування документів, додавання користувачів до системи, провести тестування, надати дані по результатам часу виконання задач

(зміст конкретних завдань до розділу, які повинен виконати студент)

8. Консультації з роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1			
2			
3			

9. Дата видачі завдання: «11» квітня 2021року

Керівник кваліфікаційної роботи  В. М. Олійник

(підпис) (ініціали, прізвище)

Завдання до виконання одержав  В. М. Ситник

(підпис) (ініціали, прізвище)

ЗМІСТ

ПЕРЕЛІК ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

БД – база даних;

ІС – інформаційна система;

ОС – операційна система;

ПЗ – програмне забезпечення;

UML – Unified Modeling Language;

НТП – науково-технічний прогрес;

ІТ – інформаційна технологія.

ВСТУП

У наш час неможливо уявити собі світ без інформаційних систем. У більшості систем наявні функції довгострокового зберігання та обробки інформації. На сучасному етапі інформація стала мірилом, яке визначає ефективність будь-якої сфери діяльності. Несвоєчасне отримання інформації або її втрата можуть обернутися втратою грошей. Для гнучкого, оперативного та ефективного управління фірмами, підприємствами та організаціями різних форм власності, телекомунікаційними засобами цивільного і військового призначення, інформаційно-обчислювальними, екологічними, радіолокаційними системами широко впроваджуються системи автоматизованого управління.

Автоматизація – один з багатьох спрямувань НТП, ціль якої вживання саморегульованих технічних прийомів, самодіючих й автоматизованих систем організації будь-якої діяльності. Обробка даних і доступ до них займає одне з найперших місць.

Обробка великої кількості інформації є проблемою як і для підприємств так і для закладів освіти. Впровадження нових ІТ є одним із рішень подібної проблеми.

Рутинну роботу з написання документів і зберігання їх в друкованому вигляді можна замінити зберіганням в електронному форматі.

Ефективно організувати процес управління та автоматизувати діяльність співробітників середніх шкіл — складна задача. Найбільш рутинні процеси повинні бути автоматизовані в першу чергу.

Метою кваліфікаційної магістерської роботи є розробка веб-орієнтованої інформаційної системи для автоматизації діяльності співробітників середніх шкіл.

Об’єктом дослідження є діяльність співробітників середніх шкіл.

Предметом дослідження є шляхи реалізації системи для розробки веб-орієнтованої інформаційної системи для автоматизації діяльності співробітників середніх шкіл.

Завданнями дослідження є:

— сформувати постановку задачі дослідження;

— розробити математичне забезпечення прогнозування ефективності впровадження системи за допомогою методу Хольту;
— розробити бізнес-процеси та бізнес-правила для системи автоматизації діяльності співробітників середніх шкіл;
— сформувати функціональні та нефункціональні вимоги для програмного рішення, що розробляється;
— спроектувати системну архітектуру для програмного рішення з урахуванням сформованих вимог;
— розробити та протестувати прототип системи;
— зібрати статистику та спрогнозувати час виконання задач.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Огляд предметної області

А. Автоматизації діяльності співробітників середніх шкіл

Для сьогодення якість освіти один з тих факторів який руйнівною силою впливає на наше майбутнє.

Як майже і все на світі, освіта як соціальне явище перебуває у динамічному стані. Технології, а також маніпуляції з інформацією, так само і покращення особистісних відносин у офіційно-діловому образі життя мають відношення до безлічі глобальних явищ не лише у конкретних особистостей а так позначаються на розвитку всього людства.

Освітня система повинна адаптуватися до цих змін всередині своєї галузі, адже ці зміни згодом впливають на суспільство.

Але, на жаль, в сфері освіти, унаслідок відсутності фінансування держави, а так само незацікавленості приватних осіб, є вкрай обмежена кількість програмних продуктів спрямованих на автоматизацію діяльності співробітників сфери освіти, як зарубіжних, так і вітчизняних.

І, якщо, для вишів дана проблема не стоїть досить гостро, так як при виникненні необхідності в даному продукті, його можливо розробити силами наукових співробітників або студентів, то для шкіл можливо тільки використання безкоштовних ресурсів або придбання дорогих програмних продуктів.

Система для середніх шкіл повинна охоплювати:

1) планування і управління процесів і завдань;
2) управління персоналом, учнями та батьками;
3) комунікацію між персоналом, учнями та батьками.

Планування є одним з найважливіших процесів. Управління одна із основних функцій планування. Розвиток підприємств, в тому числі і закладів освіти залежить від планування, постановки цілей і роботи кожного підрозділу. При плануванні

визначаються задачі, ресурси, які необхідні для їх досягнення та кінцеві дати, а також план їх реалізації.

Планування робочого дня співробітника також має великий пріоритет. Ефективність роботи в великій долі залежить від рівня буденної діяльності колективу правління. Перелічені нижче задачі можуть бути вирішені за допомогою планування та управління.

1) Збільшення ефективності роботи співробітників;

2) Інспектування дій і затрат часу на них;

3) Оптимізація робочого розкладу;

4) Вдосконалення технік використання часу;

5) Відсів неефективних видів діяльності;

6) Раціональне планування на виконання найважливіших справ;

7) Чіткий розподіл робочого та особистого часу.

Підсумовуючи, можна охарактеризувати планування таким процесом, який виконує функції управління, для врахування всіх можливих ризиків та варіантів розвитку майбутнього [1].

Останній пункт — комунікація — є чи не найважливішим у системі освіти.

Педагогічне спілкування - це професійне спілкування викладача з учнями на уроці і поза ним, що має певні педагогічні функції і спрямоване на створення психологічного клімату навчальної діяльності і відносин між педагогом і учнями та всередині учнівського колективу; його успіх визначає успіх в навчанні і вихованні.

Виховання за своєю суттю - комунікативний процес, основою якого є спілкування: через спілкування вчитель організовує поведінку і діяльність учнів, оцінює їх роботу і вчинки, інформує про події, що відбуваються, викликає відповідні переживання з приводу проступків, допомагає подолати труднощі, не втратити віру в свої можливості.

Найважливішим видом професійного спілкування вчителя поряд з уроком є позакласний захід (вечір, екскурсія, культпохід, класна година, збори). Ці види спілкування не повинні нагадувати урок, інакше школярі будуть уникати цих заходів, розглядаючи їх як примусове продовження уроків.

В індивідуальному спілкуванні з учнями перед педагогом завжди стоїть початкова психологічна установка: треба розгадати людину перед собою, відкрити його особистість, все приховане, що не є явним для звичайної людини. У кожного свій темперамент, здібності, характер: один дуже легко всім захоплюється, інакше все швидко набридає, один працьовитий, інший не проти полінуватися. І все це треба враховувати вчителю. Для кожного потрібен свій конкретно-специфічний, індивідуалізований стиль спілкування [2].

Б. Особливості автоматизації діяльності співробітників середніх шкіл

Налагодження інформаційних каналів, які використовуються для трансферу інформації про стан підприємства між різними прошарками управління та підрозділами проводиться на основі новітньої електронно-обчислювальної техніки.

Інформація використовується для створення необхідної документації.

Інформація повинна задовольняти наступні вимоги:

– по напрямку - втіха визначених потреб;

– по точності – лише перевірені ресурси з достовірною інформацією;

– по об'єкту та якості - стислість і чіткість визначень.

Введення ПЗ в школах сприятиме:

– зменшенню кількості однотипної роботи;

– збільшенню швидкості обробки інформації;

– забезпеченню точності інформації;

– підвищенню рівня інформаційної безпеки;

– збільшенню кількості інформації на електронних девайсах [3].

В. Функції та задачі, які повинно вирішувати система для співробітників середніх шкіл

Потрібно сформулювати список головних функцій та доручень які виконують співробітники середніх шкіл.

Автоматизація всієї діяльності навчального закладу, є величезною та складною задачею, найкращим шляхом для вирішення цієї проблеми буде автоматизація роботи переважаючого складу школи, проте необхідно не забувати про розширення в близькому майбутньому, все це необхідно закласти в оригінальну архітектуру.

Проаналізуємо закон Міністерства освіти та науки України «Про затвердження кваліфікаційних характеристик професій (посад) педагогічних та науково-педагогічних працівників навчальних закладів»

Витяг з розділу «Задачі та обов'язки заступника директора з виховної роботи»:

– репрезентує школу в державних органах; виконує функції менеджмента коштів;

– контролює фінансові витрати;

– відповідальний за розклад академічного процесу, правила внутрішнього режиму;

– відповідальний за зарахування нових учнів;

– відповідальний за коректний територіальний розподіл учнів;

– контролює діяльність організацій;

– висвітлює робітникам їх привілеї та повинності;

– відповідальний за правила внутрішнього розпорядку;

– відповідальний за навчальну базу [4].

Витяг з розділу «Задачі та обов'язки заступника директора з виховної роботи»:

– вивчає психологічні та інтелектуальні напрямки можливо розвитку учнів;

– використовує свої знання педагогічної і психологічної наук для проведення виховних та інших заходів;

– влаштовує діяльність студентських гуртків;

– організовує розважальні заходи;

– є членом педагогічних та методичних рад [5].

Витяг з розділу «Задачі та обов'язки заступника директора з методичної роботи»:

– відповідальний за планування методичної діяльності навчального закладу;

– нормалізує та узгоджує роботу наочних (циклових) комісій;

– застосовує і вдосконалює методи проведення навчального процесу;

– підсумовує та ділиться інформацією про прогресивні технології навчання;

Витяг з розділу «Задачі та обов’язки заступника директора з навчальної роботи»:

– формує поточне і далекосяжне планування навчальної роботи навчального закладу;

– використовує і поліпшує методи організації навчального процесу і модерних освітніх технологій;

– контролює об'єктивність оцінювання наслідків навчальної діяльності студентів;

– приділяє свою підтримку педагогічним працівникам;

– реалізує ревізію за навчальним навантаженням студентів;

– поповнює навчальний процес новими ідеями [6].

1.2 Опис існуючих рішень

А. Електронний журнал

Система, яка має зовнішній вигляд наближений до класного журналу та щоденника, створена Ukrschools, Україна.

Журнал оцінок забезпечує відображення отриманих учнем відміток. Батьки можуть проаналізувати якість навчання та скоректувати підготовку дитини з того чи іншого предмету.

Сервіс дозволяє зберігати будь-які навчальні матеріали - книжки, корисні посилання, відеофайли та інше.

Система виконує наступні функції. Батькам:

– Моніторинг виконання ДЗ та оцінка успішності учня

– СМС-повідомлення у випадку відсутності учня в школі

– Особистий чат з вчителем

– Розклад батьківських зборів

Учням:

– Моніторинг успішності

– База ДЗ, розлад

– Місце в рейтингу

Вчителям:

– Формування звітності

– Комунікація з батьками

Робота системи «Електронний щоденник» (далі по тексту — Система) ґрунтується лише на обробці інформації з паперових носіїв. Приклад головної сторінки представлено на рисунку 1.1.

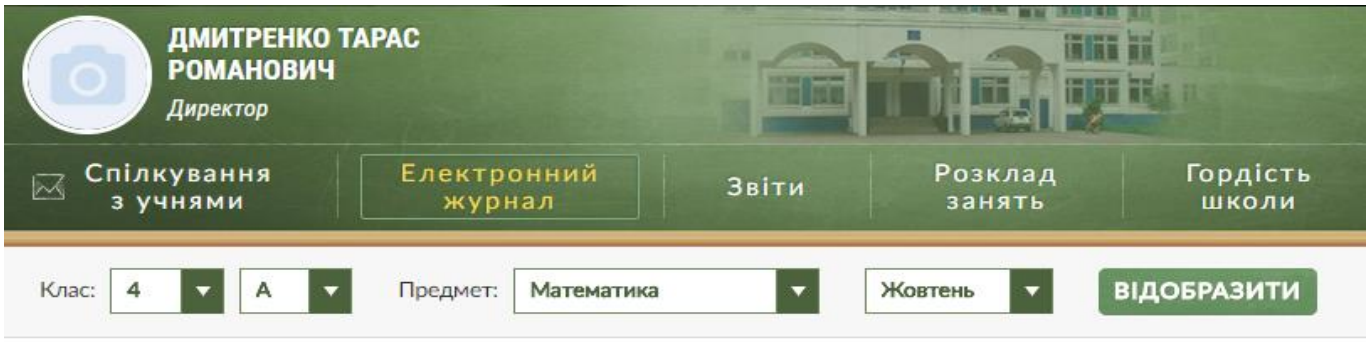

Рисунок 1.1 – Інтерфейс системи «Електронний щоденник»

Принцип роботи системи наступний, в кожній школі є свій оператор, щоранку, перед другим уроком він має обробити інформацію про учнів, які були відсутні на уроках, та розіслати інформацію про це батькам цих учнів за допомогою системи. Оператор опрацьовує інформацію про ДЗ учнів протягом 6 годин. Оновлення даних про успішність відбувається щотижнево, також при ручному внесені даних.

Під час реєстрації в системі для користувача генеруються логін та пароль через смс повідомлення на телефон. [7].

Б. Щоденник

Щоденник це проект, задачею якого, є розвиток освітньої системи на базі теперішніх інтернет-технологій, створений ТОВ «Щоденник», Україна. Інтерфейс системи представлено на рисунку 1.2.

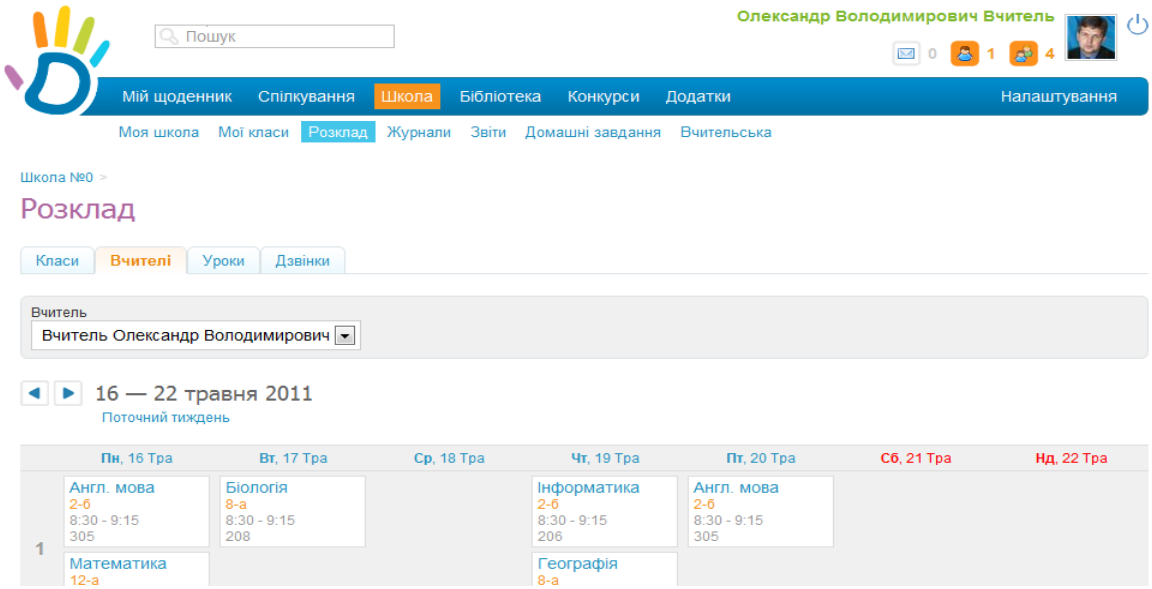

Рисунок 1.2 – Інтерфейс системи «Щоденник»

Доступні можливості:

1) Всі групи користувачів мають доступ до розкладу.

2) Система розкладу гнучка, що дозволяє змінювати дати початку та кінця обраного періоду.

3) Учням і вчителям доступний їх особистий розклад уроків. Вчителі також мають доступ до розкладів інших вчителів в школі.

4) Домашні завдання можуть бути виконані учнями та перевірені вчителями онлайн

5) Вчитель має безмежний доступ до бази домашніх завдань.

6) У бібліотеці Щоденника представлено декілька тисяч художніх творів, в першу чергу ті, які вивчаються за шкільною програмою.

7) В медіатеці програми наявні аудіо та звичайні книги, навчальні відео.

За допомогою каталогу дуже зручно знаходити потрібну інформацію.

Щоденник адаптував майже всі функції соціальних мереж під шкільний лад.

Кожен користувач має свій особистий профіль [8].

В. 1С: Підприємство

1С: Підприємство - продукт компанії «1С», призначений для автоматизації діяльності на підприємстві.

«1С: Підприємство» можна назвати базою даних з особливою оболонкою яка має свою власну мову програмування, яка може бути використана при взаємодії з іншими програмами.

Інтерфейс системи представлено на рисунку 1.3.

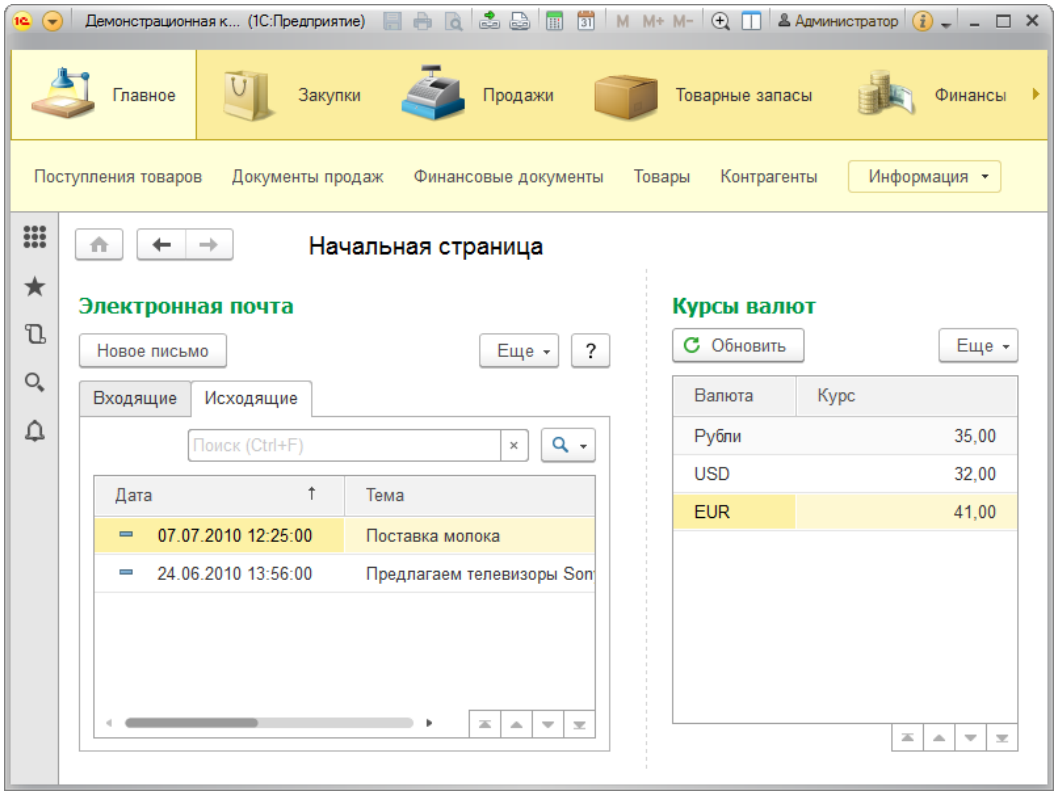

Рисунок 1.3 – Інтерфейс системи «1С: Підприємство»

Більшість конфігурацій на базі платформи версій 8.0 та 8.1 мають:

1) можливості загальної настройки програми (такої, як встановлення дати заборони редагування даних);

2) можливості індивідуальної настройки програми для кожного користувача (такий, як основний склад для автоматичної підстановки в документи);

3) безліч перемикаємих інтерфейсів;

4) безліч наборів прав (ролей), причому частина прав може призначатися користувачами в режимі підприємства;

5) Гнучкіше звіти, що налаштовуються, ніж у версії 7.7;

6) Можливість побудови довільних звітів (так звана «Консоль звітів»);

7) Вбудований універсальний обмін даними;

8) Вбудовані можливості поновлення через Інтернет.

Конфігурація «1С: Управління підприємством»

За допомогою програми можна автоматизувати підприємства будь-якого масштабу. Система сумісна тільки з платформою 8.3.

Ключовими перевагами нового флагманського рішення фірми «1С» є:

1) широкі функціональні можливості;

2) гнучка і продуктивна сучасна платформа «1С: Підприємство 8.3», що підтримує роботу через Інтернет, в тому числі «хмарні» технології і роботу на мобільних пристроях;

3) використання у виробництві унікальних методик, наприклад, Теорії обмеження, в тому числі і методу «барабан - буфер - мотузка»;

4) невисока вартість володіння і можливість отримання істотного економічного ефекту з ростом продуктивності праці і швидким поверненням інвестицій [9].

Г. Trello

Trello - програма для управління проектами невеликих груп, створено Fog Creek Software, США.

Trello – це менеджмент система яка має картки з задачами. Кожна задача демонструє стан будь-якого проекту.

Однією з головних переваг Trello – є можливість моніторингу багатопоточну кількість проектів та їх стан в на хвилину часу. Система дуже зручна в управлінні великою групою людей і моніторингу ходу виконання задач цією групою (рис. 1.4).

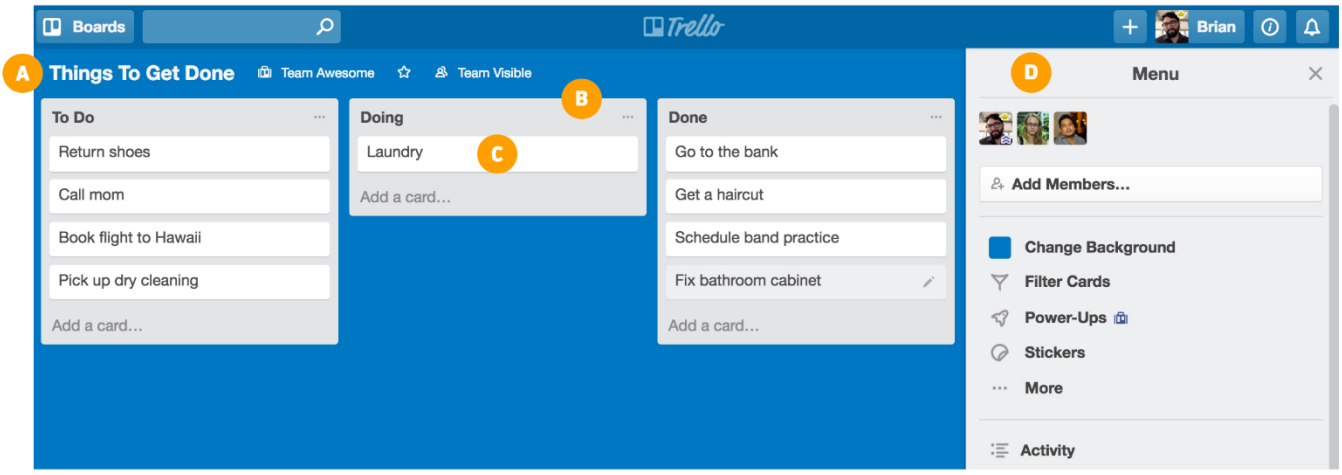

Рисунок 1.4 – Інтерфейс системи «Trello».

Задачки мають сотні можливостей. На задачку можна призначити виконавця обравши із доступних членів команди. Також можна завантажувати масиви інформації, голосувати, проводити дебати.

Цей інструмент є дуже надійним. Він дозволить зв'язати та керувати усіма проектами в одному менеджері [10].

Д. Порівняння розглянутих рішень

У таблиці 1.1 представлено порівняльний аналіз розглянутих систем управління задачами.

Таблиця 1.1 – Порівняльний аналіз розглянутих рішень

Параметр	Електронний журнал	Щоденник	1С: Підприємство	Trello
Управління задачами	Відсутнє	Відсутнє	Створення форм, звітів, діаграм та зв’язаних кроків для управління задачами.	Звіти, пріоритет, таблиця з картками
Управління навчальним процесом	Наявність журналів і щоденників, з можливістю перегляду, редагування, завантаження.	Наявність журналів і щоденників, з можливістю перегляду, редагування, завантаження. Можливість створення груп за інтересами.	Створення форм та звітів зв’язаних між собою алгоритмами.	Таблиця з картками

Продовження таблиці 1.1

Система комунікації	Особисті чати між батьками та вчителем. Почтові та смс повідомлення.	Особисті сторінки, групи та події.	Відсутнє	Коментарі до карток-задач
Розробка документації	Додавання файлів	Відсутнє, можливо додавати документи до бібліотеки	Відсутнє	Вкладення файлів до задач
Розгортання системи	Сайт	Сайт	Сервер	Сервер
Ціна використання	Платна	Безкоштовно	Платна корпоративна ліцензія	Платна корпоративна ліцензія

1.3 Постановка задачі дослідження

Після аналізу закону Міністерства освіти та науки України «Про затвердження кваліфікаційних характеристик професій (посад) педагогічних та науково-педагогічних працівників навчальних закладів» можна зробити висновок, що посада директора більшої мірою включає загальношкільну кадрову діяльність, матеріально-фінансові питання, регулює діяльність навчальних і педагогічних організацій, та спрямована великою частиною на позашкільні питання. Натомість, діяльність заступників директора є орієнтованою на шкільну та міжшкільну діяльність.

Виходячи з цього доцільно зосередитись на заступниках директора та автоматизувати їх діяльності за наступними напрямками:

– автоматизоване управління регулярними та не регулярними задачами, такими як: екскурсії, олімпіади, семінари, перевірку пожежної безпеки, перевірку обладнання кабінетів, перевірку спортивних приміщень та інвентарю, організації та координації роботи предметних (циклових комісій) та інше;

– надання засобів для колективної розробки навчально-методичної документації;

– надання засобів обліку співробітників.

Функціонування розробляємої системи допомогає збирати статистику, на основі якої можливо виконати прогнозування часу виконання задач, що в свою чергу зможе підвищити ефективність управління закладом освіти.

З урахуванням актуальності теми і загального стану проблеми необхідно вирішити наступні задачі:

1) оглянути літературні джерела та проаналізувати сучасний стан проблеми автоматизації діяльності співробітників середніх шкіл, розглянути деякі існуючі програмні рішення, виділити їх особливості, переваги та недоліки;

2) сформувати постановку задачі дослідження;

3) розробити математичне забезпечення прогнозування ефективності впровадження системи за допомогою методу Хольту;

4) розробити бізнес-процеси та бізнес-правила для системи автоматизації діяльності співробітників середніх шкіл;

5) сформувати функціональні та нефункціональні вимоги для програмного рішення, що розробляється;

6) спроектувати системну архітектуру для програмного рішення з урахуванням сформованих вимог;

7) розробити та протестувати прототип системи;

8) зібрати статистику та спрогнозувати час виконання задач.

2 РОЗРОБКА МАТЕМАТИЧНОГО, АЛГОРИТМІЧНОГО ТА ПРОГРАМНО АРХІТЕКУРНОГО ЗАБЕЗПЕЧЕННЯ ВИРІШЕННЯ ЗАДАЧІ АВТОМАТИЗАЦІЇ ДІЯЛЬНОСТІ СПІВРОБІТНИКІВ СЕРЕДНІХ ШКІЛ

2.1 Прогнозування за допомогою методу Хольта.

В теперішніх умовах існує безліч різноманітних методів прогнозування, які базуються виключно на аналізі попередніх значень хронологічної послідовності, а отже методів, що використовують принципи, що використовуються в технічному аналізі. Головним знаряддям цих методів є схема екстраполяції, коли властивості матриці, розпізнані через певний інтервал часу, перевищують її межі.

Процес прогнозування вимагає індивідуального підходу і, як правило, передбачає низку процедур.

1 Розбір часової послідовності на предмет факту присутності пропущених значень. Коректування цих значень.

2 Дефініція присутності тренду і його типу. Дефініція наявності періодичності в послідовності.

3 Перевірка послідовності на стаціонарність.

4 Розгляд послідовності на необхідність попередньої обробки Вибір моделі.

5 Визначення параметрів моделі. Передбачення на підставі обраної моделі.

6 Оцінка точності прогнозування моделі.

7 Аналіз характеру помилок обраної моделі.

8 Визначення адекватності обраної моделі і в разі необхідності її заміна і повернення до попередніх пунктів.

Що таке стаціонарність процесу? Це означає, що процес буде розвиватись весь час однаково, незалежно чи це минуле, теперішнє і майбутнє. Саме такий процес є пріоритетним з точки зору побудови прогнозів, але, на жаль, в житті таки процеси не реалістичні, всі процеси з часом зазнають змін.

У процесів в реальності з часом можуть змінюватися такі показники як математичне очікування, дисперсія, закон розподілу, але процеси, у яких ці

характеристики зазнають дуже повільних змін можна віднести до стаціонарних процесів. Це можна пояснити тим, що зміна на початковому і кінцеву етапі незначні і ними можна знехтувати.

Час спостереження впливає на ймовірність прийняття невірного рішення про стаціонарність системи. Але якщо розглядати стан процесу в пізні моменти часу, і прогноз базується на короткому інтервалі, то зменшення розміру вибірки може привести до покращення точності прогнозу.

Параметри послідовності певні на інтервалі спостереження у процесу зазнавшого змін, за його межами будуть змінюватися. Тобто довжина інтервала прогнозу прямопорційно вливає на похибку прогнозу. Виходячи з параметрів, які були згадані раніше, ми змушені обмежитись лише короткостроковим прогнозом.

Мінливість параметрів послідовності має гігантський вплив на оцінку по інтервалу спостереження, де ми одержуємо деяке усереднене їх значення, через те, що на цьому відрізкові вони залишалися змінними. Знайдені значення установки не будуть належати до кінцевого моменту часу цього інтервалу, а будуть відображати лише деяке їх середнє значення. Абсолютне усунення цього неприємного явища, на превеликий жаль, не є здійсненним, але його вплив можна зменшити, зменшуючи тривалість інтервалу спостереження, на якому проводиться оцінка параметрів моделі.

Вкорочувати до нескінченності цей інтервал теж не можна, так як при надмірному скороченні інтервалу навчання напевно буде знижуватися точність оцінки параметрів послідовності. Необхідно шукати компроміс між впливом помилок, пов'язаних з мінливістю характеристик послідовності і зростанням похибок, пов'язаних з надмірним скороченням інтервалу навчання.

Найпростіша модель являю собою:

$$X(t) = L(t) + r(t), \tag{2.1}$$

де, $X(t)$ – досліджуваний (модельований) процес,

$L(t)$ – рівень процесу, який змінюється,

$r(t)$ – випадкова величина з нульовим середнім значенням.

Модель включає в себе суму двох компонент, з яких нас цікавить рівень процесу L (t), саме його ми і спробуємо виділити.

Добре відомо, що при усередненні випадкової послідовності можна добитися зниження її дисперсії, тобто зменшити розмах її відхилення від середнього значення. Тому можна припустити, що якщо процес, описуваний нашої найпростішою моделлю піддати усереднення (згладжування), то ми зможемо якщо не зовсім позбутися від випадкової компоненти r (t), то хоча б помітно її послабити, виділивши тим самим цікавий для нас рівень L (t) .

Для цього звернемося до простого експоненціального згладжування (Simple Exponential Smoothing, SES).

$$S_t = \alpha X_t + (1 - \alpha) S_{t-1}, \tag{2.2}$$

У цьому добре відомому вислові ступінь згладжування задається коефіцієнтом альфа, який можна встановити в інтервалі від 0 до 1. При виборі значення альфа, рівним нулю, знову надходять величини вхідної послідовності X не зможуть надати ніякого впливу на результат згладжування. Результатом згладжування для будь-яких моментів часу буде постійна величина.

Таким чином, в цьому крайньому випадку ми повністю придушимо заважуючу випадкову компоненту, але при цьому і цікавий для нас рівень процесу буде згладжений до стану горизонтальної прямої лінії. Якщо значення коефіцієнта альфа встановити рівним одиниці, то на вхідну послідовність процес згладжування взагалі не буде надавати ніякого впливу. При цьому нас цікавить рівень L (t) не спотворений, а й випадкова компонента пригнічена не буде.

Інтуїтивно зрозуміло, що при виборі величини альфа необхідно одночасно задовольняти суперечливим вимогам. З одного боку значення альфа має бути близько до нуля, щоб ефективно придушити випадкову складову r(t). З іншого, щоб не спотворити інформацію, що цікавить нас компоненту L(t), бажано вибирати значення альфа, близьким до одиниці. Для знаходження оптимального значення альфа нам необхідно визначити критерій, за яким це значення можна буде оптимізувати.

При визначенні такого критерію згадаємо, прогнозування, а не просто про згладжування послідовності.

В даному випадку для простої моделі експоненціального згладжування прогнозом на будь-яку кількість кроків вперед прийнято вважати знайдене в даний момент часу значення S_t.

$$\hat{X}_t(m) = S_t, m = 1,2,3 \ldots, \tag{2.3}$$

де $\hat{X}_t(m)$ – прогноз в момент часу t на m кроків вперед.

Значить, прогнозом значення послідовності на момент часу t буде прогноз, зроблений на попередньому кроці на один крок вперед.

$$\hat{X}_t = \hat{X}_{t-1}(1) = S_{t-1}, \tag{2.4}$$

У цьому випадку в якості критерію для оптимізації значення коефіцієнта альфа можна використовувати помилку передбачення на один крок

$$e_t = X_t - \hat{X}_t = X_t - S_{t-1}, \tag{2.5}$$

Таким чином, якщо мінімізувати суму квадратів цих помилок по всій вибірці, то можна буде визначити оптимальне значення коефіцієнта альфа для даної послідовності. Природно, найкращим значенням альфа буде таке, при якому величина суми квадратів помилок виявиться мінімальною.

При застосуванні експоненціального згладжування необхідно тим чи іншим способом вибрати величину коефіцієнта згладжування. Але цього виявляється недостатньо. Так як при експоненційному згладжуванні поточне значення обчислюється на основі попереднього, то для нульового моменту часу виникає ситуація, коли таке значення ще відсутня. Тобто, для нульового моменту часу, якимсь чином повинні бути визначені початкове значення для S або для S1 і S2 в разі моделі лінійного росту.

У літературі можна зустріти різні рекомендації по вибору початкових значень. Наприклад, початкове значення для простого експоненціального згладжування може бути прирівняне першому елементу послідовності або в надії на згладжування випадкових викидів може бути знайдено як середнє значення трьох-чотирьох початкових елементів послідовності. Для моделі лінійного росту початкові значення S1 і S2 можуть бути визначені виходячи з припущення, що початковий рівень прогнозної кривої повинен бути дорівнює першому елементу послідовності, а нахил лінійного тренда дорівнює нулю [11].

Хольт розробив модель простого експоненціального стилю і додав в неї тенденцію. Для прогнозування часових рядів, коли є тенденція до зростання або падіння значень часового ряду використовується саме його метод. Окрім того, метод використовується для рядів, коли дані не є повним циклом, а сезонність ще не виділити.

Метод базується на оцінці ступеня лінійного зростання (або падіння) показника в часі. Фактор роста оцінюється за коефіцієнтом y_t, який і свою чергу обчислюється як експоненціально зважене середнє різниць між поточними експоненціально зваженими середніми значеннями процесу a_t і їх попередніми значеннями a_{t-1}. Що виділяє цей метод серед інших – це обчислення поточного значення експоненціально зваженого середнього a_t включає в себе обчислення минулого показника зростання y_{t-1}, адаптуючись таким чином до попереднього значення лінійного тренду.

Згладжування даних:

$$a_t = \alpha y_t + (1 - \alpha)(a_{t-1} + b_{t-1}), \tag{2.6}$$

Згладжування тренду:

$$b_t = \beta(a_t - a_{t-1}) + (1 - \beta)b_{t-1}, \tag{2.7}$$

Прогноз на період t + k:

$$y_{t+k} = a_t + b_t k, \tag{2.8}$$

Перевірка точності прогнозу:

$$\hat{y}_t = \frac{|y_t - y_t^*|}{y_t}, \tag{2.9}$$

де, a_t - згладжене значення прогнозованого показника для періоду t;

b_t - оцінка приросту тренда, що показує можливе зростання або спадання значень за один період;

α, β - параметри згладжування ($0 \leq \alpha \leq 1$; $0 \leq \beta \leq 1$);

y_t^* - прогнозу поточної ітерації;

k - кількість періодів часу, на які на які проводиться прогноз.

Параметри згладжування α і β вибираються суб'єктивно або шляхом мінімізації помилки прогнозу. При великих значеннях параметрів матиме місце більш швидкий відгук на зміни, що відбуваються. Чим більше параметр, тим більшого згладжування піддаються дані.

Для того, щоб скористатися рівняннями для отримання прогнозу, необхідно, визначити початкові умови. По-перше, початкова умова для згладжених даних можна задати рівним першому спостереженню, при цьому початкова умова для тренда (b_{t-1}) дорівнюватиме нулю. По-друге, початкова умова для згладжених даних можна визначити, як середнє для перших k спостережень. Тоді початкова умова для тренда можна оцінити нахилом лінії, утвореної цими k точками.

Було побудовано діаграму діяльності для алгоритму (рисунок 2.1)

Рисунок 2.1 – Діаграма діяльності, що описую принцип алгоритму Хольта

2.2 Підходи до підвищення забезпечення якості задачі автоматизації діяльності середньої школи.

Одними з ключових вимог стандартів ISO:9000 є застосування процесного підходу до організації системи управління якістю на підприємстві, яка передбачає моделювання процесів організації з метою їх подальшого аналізу та оптимізації [12]. Моделювання процесів здійснюється за допомогою спеціальної методології. IDEF0 – це сама найпоширеніша методологія функціонального моделювання, призначена для формалізації та графічного опису процесів. В рамках методології кожний процес характеризується певними цифровими параметрами, котрі характеризують проходження процесу і витрати на нього – показниками процесу. Одним із найважливіших показників процесу є показник вартості процесу

На даний час для розробки автоматизованої ІС існують два основних підходи, які відрізняються один від одного різними способами декомпозиції систем.

Перший – функціонально-модульний або структурний. Основою цього методу є принцип функціональної декомпозиції, за якою структура системи описується в термінах ієрархії її функцій і передачі інформації між окремими функціональними елементами.

Другий – об’єктно-орієнтований підхід, який використовує об’єктну декомпозицію. При цьому структура системи описується в термінах об’єктів та зв’язків між ними, а поведінка системи описується у термінах обміну повідомленнями між об’єктами.

Суть структурного підходу до розробки ІС полягає в його розбитті (поділі) на автоматизовані функції: система поділяється на функціональні підсистеми, які в свою чергу поділяються на підфункції, завдання тощо. Процес розбивки триватиме до очікування певних процедур. Система підтримує цілісний погляд, коли всі компоненти взаємопов’язані. При розробці системи "втрачається цілісність окремих завдань до загальної системи; виникають проблеми з підключенням інформації від окремих компонентів".

Усі сучасні методи структурного підходу базуються на ряді загальних принципів. В якості двох основних принципів використовуються наступні:

Перший принцип – «розділяй і пануй» – принцип вирішення складних проблем шляхом їх розбиття на безліч менших незалежних проблем, які легко зрозуміти і вирішити.

Другий принцип – ієрархічного упорядкування – принцип організації компонентів проблеми в ієрархічних деревних структурах, додаючи нові деталі на кожному рівні.

Поділ двох основних принципів не означає, що інші принципи є другорядними, оскільки ігнорування будь-якого з цих принципів може мати непередбачені наслідки (включаючи провал всього проекту). Найважливішими з цих принципів є наступні:

Принцип абстрагування – полягає у виділенні основних аспектів системи та відвернути їх від незначних;

Принцип формалізації – це необхідність жорсткого методологічного підходу до вирішення проблеми;

Принцип несуперечності – це валідність і послідовність елементів;

Принцип структурування даних – полягає в тому, що дані повинні бути структуровані та організовані ієрархічно.

Об'єктно-орієнтований підхід заснований на систематичному використанні моделей для лінгвістично незалежної розробки програмної системи на основі її прагматики. Прагматика визначається метою розробки програмного забезпечення. Реальні об'єкти та поняття, пов'язані з розробленою системою програмного забезпечення, беруть участь у формулюванні мети. В об'єктно-орієнтованому підході ці об'єкти та концепції замінюються їх моделями, тобто певними формальними конструкціями, які вони представляють у програмній системі.

Модель містить не всі особливості та властивості об’єкта (концепції), який вона представляє, а лише ті, які є суттєвими для розробленої програмної системи. Отже, модель "бідніша", ніж об'єкт або концепція, яку вона представляє. Однак головне навіть не в цьому, а в тому, що модель є формальною конструкцією: формальний характер моделей дає змогу визначити формальні взаємозв'язки між ними та формальні операції над ними. Це спрощує як розробку, так і вивчення (аналіз) моделей та їх реалізацію на комп’ютері. Зокрема, формальний характер моделей дозволяє одержати формальну модель розроблюваної програмної системи, як композицію формальних моделей її компонентів.

Тому об’єктно-орієнтований підхід допомагає вирішити такі проблеми, як зменшення складності програмного забезпечення; підвищити надійність програмного забезпечення; Забезпечити можливість зміни окремих компонентів програмного забезпечення без зміни будь-яких інших його компонентів; Надання можливості повторного використання окремих програмних компонентів.

Об’єктно-орієнтоване проектування засновано на принципах: виділення абстракцій, обмеження доступу, модульності, ієрархії, типізації, паралельності, стійкості, поліморфізму.

Абстрагування дозволяє керувати складністю системи, концентруватися на вагомих властивостях об’єкта, відрізняти його від об’єктів іншого типу. Воно залежить від теми та точки зору. Увага концентрується на зовнішніх факторах, дозволяє відділити поведінку об’єкта від його реалізації. Основа абстракції – клас та об’єкт.

Інкапсуляція – фізичне розташування властивостей та поведінки в межах однієї реалізації, абстракції, приховує її реалізацію за загальнодоступним інтерфейсом. Це поєднання властивостей та поведінки об’єкта.

Модульність – логічний розподіл складної системи на кілька слабо пов'язаних підсистем або модулів. Зменшує складність системи та дозволяє розробляти окремі модулі незалежно один від одного. Мета декомпозиції на модулі – зменшення строків та вартості розробки програмних систем за рахунок проектування модулів та їх неодноразового використання. Вимоги – простота, зрозумілість. Зміни повинні бути можливі без знання реалізації інших модулів і без впливу на їх поведінку. Визначення класів та об’єктів виконується в ході логічної, а модулів – в ході фізичної розробки. Ці дії пов’язанні та виконуються ітеративно.

Ієрархія – рангована або упорядкована система абстракцій, розташована по рівням у вигляді дерева. Ієрархія спадкування – структура з класів. Спадкування визначає відносини між класами, де клас розрізняє структуру або поведінку.

Розходження між не об’єктно-орієнтованими й об’єктно-орієнтованими системами проектування в основному пов'язані не з можливістю виразити в програмі необхідну функціональність (відповідно до теорії алгоритмів будь-яка функціональність може бути виражена на кожній мові програмування, або не може бути виражена на жодній мові програмування), а з виразністю мови програмування, зручністю складання програм, їхнього налагоджування і супроводу.

Переваги об’єктно-орієнтованого підходу обумовлюються наступними факторами:

Виразність: у не об’єктно-орієнтованій системі програмістові необхідно самому відображати операції над об'єктами (виклики методів, або оголошення підкласів)

уявні виклики відповідних функцій (при використанні структурно-орієнтованих оточень це забезпечується системою проектування).

Зручність: у не об’єктно-орієнтованій системі проектування програміст змушений вручну відслідковувати ієрархію класів при виклику методів і передачі їм параметрів; при змінах в ієрархії класів, він повинен вручну внести відповідні зміни в програму.

Захист від помилок: у не об’єктно-орієнтованій системі проектування програміст повинен щораз перевіряти правильність управління методами й об'єктами, ініціалізувати нові об'єкти, запобігати доступу до окремих атрибутів і методів.

Підтримка цілісності: при внесенні змін в оголошення об'єктів у не структурно-орієнтованому оточенні програміст повинен сам визначити вплив цих змін на програму і відповідним чином змінити її.

Проте навіть у випадку розробки прикладних програмних систем проектування з використанням методології об’єктно-орієнтованого проектування цих систем істотно спрощує їхню реалізацію і подальший супровід [13].

2.3 Основні вимоги до прикладного програмного забезпечення

А. Система бізнес правил

Перш ніж розробляти вимоги, спочатку потрібно створити систему ділових правил. Система правил ведення бізнесу - це логічний опис стандартів, процедур та іміджу дій, якими керується компанія в її роботі. Бізнес-правила використовуються для визначення взаємозв’язків між сутностями, дій, які можна виконувати над об’єктами, проміжних контрольних пунктів та робочих процесів. Правила бізнесу також описують обмеження щодо того, як система повинна враховувати дії. Правила повинні бути розроблені таким чином, щоб вони були зрозумілі як системним адміністраторам, так і розробникам, незалежно від призначення та мови програмування.

Наступна система бізнес-правил була розроблена під час аналізу предметної області.

Бізнес правила, які стосуються виключно користувачів:

1 Кожен користувач в системі характеризуються ПІБом, електронною поштою, ідентифікаційним кодом, датою народження, адресом, телефоном, роллю.

2 Наявні наступні ролі користувача: адміністратор, директор, заступник директора, вчитель, педагогічний персонал, технічний персонал, медичний персонал, учень. Кожна роль повинна бути зв’язана зі школою, тому що одна людина може працювати у декількох школах в системі одночасно на різних посадах.

3 Однакова пошта та ідентифікаційний номер не можуть використовуватися одним користувачем.

4 Для кожної задачі та сторінки документації можна вказати права доступу: школи, групи та користувачів, які матимуть доступ до цієї одиниці.

5 Адміністратор має адмін права доступу до кожної одиниці в системі.

6 Користувачі з роллю учні не повинні мати доступ до системи

7 Користувача можливо активувати, чи деактивувати. Активний користувач має доступ в систему, дезактивований – не має.

8 Для входу в систему виконується через google пошту.

Бізнес правила, які стосуються виключно задач:

1 Кожна задача характеризується назвою, описом, ключовими словами, творцем, зв’язаною школою, датою створення, строком виконання, статусом, виконавцями, користувачами які повідомляються про зміни в задачі.

2 Статус задачі може бути: нова, виконується, виконана, закрита.

3 Усі зміни в задачі повинні зберігатися, та повинно надходити письмо на електронну пошту всім користувачам, які вказаним в задачі.

4 Тільки вказані явно в задачі користувачі, або ролі які маються у користувачів та вказані в певній задачі, повинні мати доступ до цієї задачі.

5 На одного співробітника можливо призначити декілька задач, у кожній задачі можливо вказати декілька виконавців та кого повідомляти про зміні в задачі.

6 Задачі можливо переглянути в таблиці з сортуванням за назвою, школою, виконавцем, крайнім строком. Фільтрувати задачі можливо за назвою, описом, творцем, статусом, типом.

Бізнес правила, які стосуються виключно документації:

1 Документація являє собою вікі-сторінки. Версія сторінка документації характеризується назвою, номером версією, датою створення, змістом, коментарем, автором.

2 Для сторінки можливо вказати права доступу.

3 Сторінку можливо редагувати – це створить нову версію сторінки. Також видалити версію, видалити сторінку, змінити права доступу, змінити назву, роздрукувати.

4 На сторінку можливо підписатись – для підписаних користувачів буде відсилатись лист на електрону пошту про усі зміни на певній сторінці.

Б. Функціональні вимоги

Функціональні вимоги – описують функціонування системи, або, іншими словами, визначають «що» повинен робити програмний продукт. Деякі приклади: обчислення даних, обробка даних, опрацьовування даних та інші специфічні функції, які повинна виконувати система [14]. Функціональні вимоги системи, що розробляється:

1 Співробітник повинен мати можливість авторизуватись в системі, як користувач.

2 Співробітник повинен мати можливість створити або редагувати задачу: обрати тип, обрати школу, вказати ключові слова, вказати назву та зміст, обрати відповідальних, обрати тих, кого повідомляти про зміст.

3 Співробітник повинен мати можливість прийняти задачу на себе, помітити задачу як вирішену, закрити задачу.

4 Співробітник повинен мати можливість лишити коментар, редагувати свої коментарі.

5 Співробітник повинен мати можливість переглянути перелік задач, фільтрувати та сортувати задачі.

6 Співробітник повинен мати можливість сортувати задачі: за назвою, школою, типом, виконавцем.

7 Співробітник повинен мати можливість фільтрувати задачі: за назвою, за змістом, за творцем, за типом, за школою, за крайнім строком, за датою створення, за виконавцем.

8 Співробітник повинен мати можливість працювати з документацією: створити сторінку, редагувати сторінку, завантажити сторінку, перейменувати сторінку, задати права доступу до сторінки.

9 Адміністратор повинен мати можливість авторизуватись в системі, як адміністратор.

10 Адміністратор повинен мати можливість створювати шаблони задач.

11 Адміністратор повинен мати можливість переглянути перелік користувачів, переглянути інформаціє про користувача, редагувати інформацію про користувача.

12 Адміністратор повинен мати можливість редагувати будь яку задачу та сторінку документації.

Мова UML - це мова загального призначення, що розробляє, модель моделювання в галузі програмної інженерії, яка покликана забезпечити стандартний спосіб візуалізації дизайну системи.

У програмній інженерії діаграма класів в Уніфікованій мові моделювання (UML) - це тип статичної структурної діаграми, що описує структуру системи, показуючи класи системи, їх атрибути, операції (або методи) та взаємозв'язки між об'єктами. Універсальна мова об'єктного моделювання UML не залежить від мов програмування і, внаслідок цього, може підтримувати будь-яку об'єктно-орієнтовану мову програмування [16, 32, 33].

Для описаних вимог розроблена діаграма варіантів використання в нотації UML. Діаграма відображає відносини між акторами та прецедентами, і є складовою частиною моделі прецедентів (рис. 2.2).

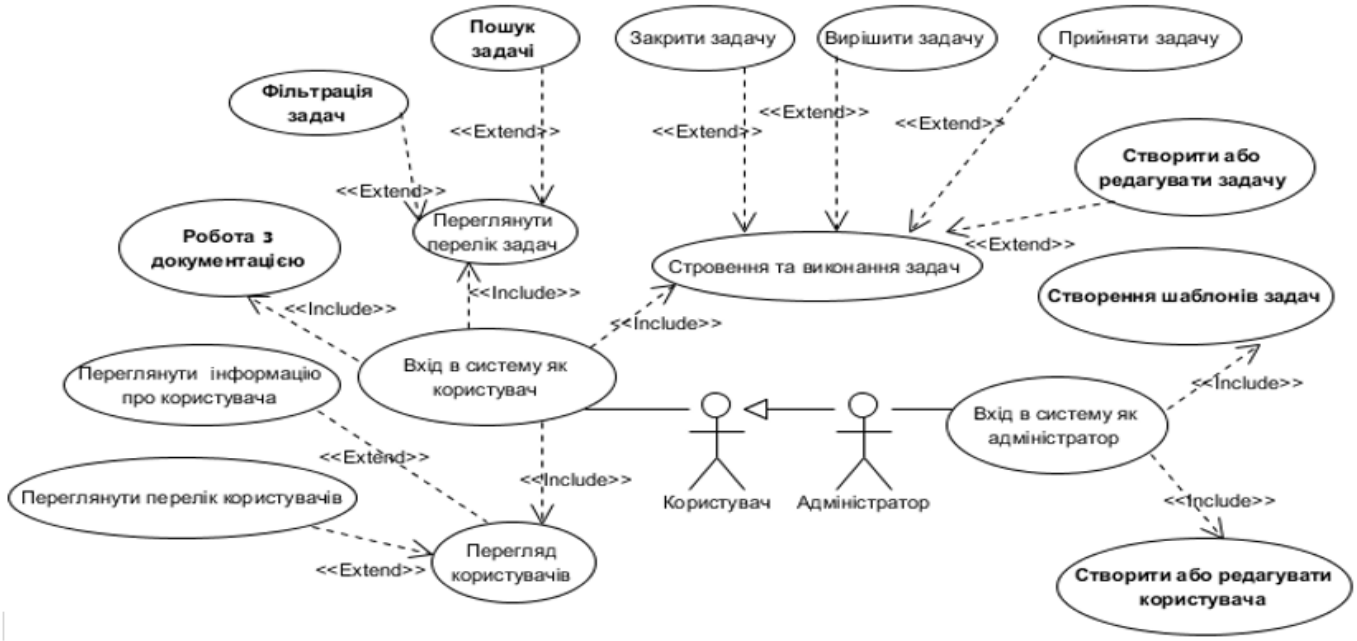

Рисунок 2.2 – Діаграма варіантів використання

Деталізація створення або редагування задач, створення шаблонів задач та робота з документацією (рис. 2.3):

Рисунок 2.3 – Діаграма варіантів використання для задач,
шаблонів та документації (деталізація)

Деталізація перегляду задач з пошуком та фільтрацією, та створення або редагування користувачів (рис. 2.4):

Рисунок 2.4 – Діаграма варіантів використання для перегляду задач та роботи з користувачами системи (деталізація)

В. Нефункціональні вимоги

Нефункціональні вимоги – У системній інженерії та інженерії вимог нефункціональна вимога (NFR) - це вимога, яка визначає критерії, за якими можна судити про роботу системи, а не про конкретну поведінку. Їм протиставляються функціональні вимоги, що визначають конкретну поведінку або функції. План реалізації функціональних вимог детально розроблений у проекті системи. План реалізації нефункціональних вимог детально описаний в архітектурі системи, оскільки вони, як правило, є архітектурно важливими вимогами [14].

Виділимо 2 параметри: перший - продуктивність (performance) та другий - зручність використання (usability). Розроблюєма система пов'язана з автоматизацією діяльності співробітників середніх шкіл, тому повинна здійснювати задані функції з високим рівнем швидкодії, наприклад, обробляти запити до серверної частини в діапазоні від 300 до 500 мс. І звісно ж, система повинна забезпечити зручність застосовування.

На базі створених функціональних та нефункціональних вимог була розроблена діаграма вимог в нотації UML та представлена на рисунку 2.5.

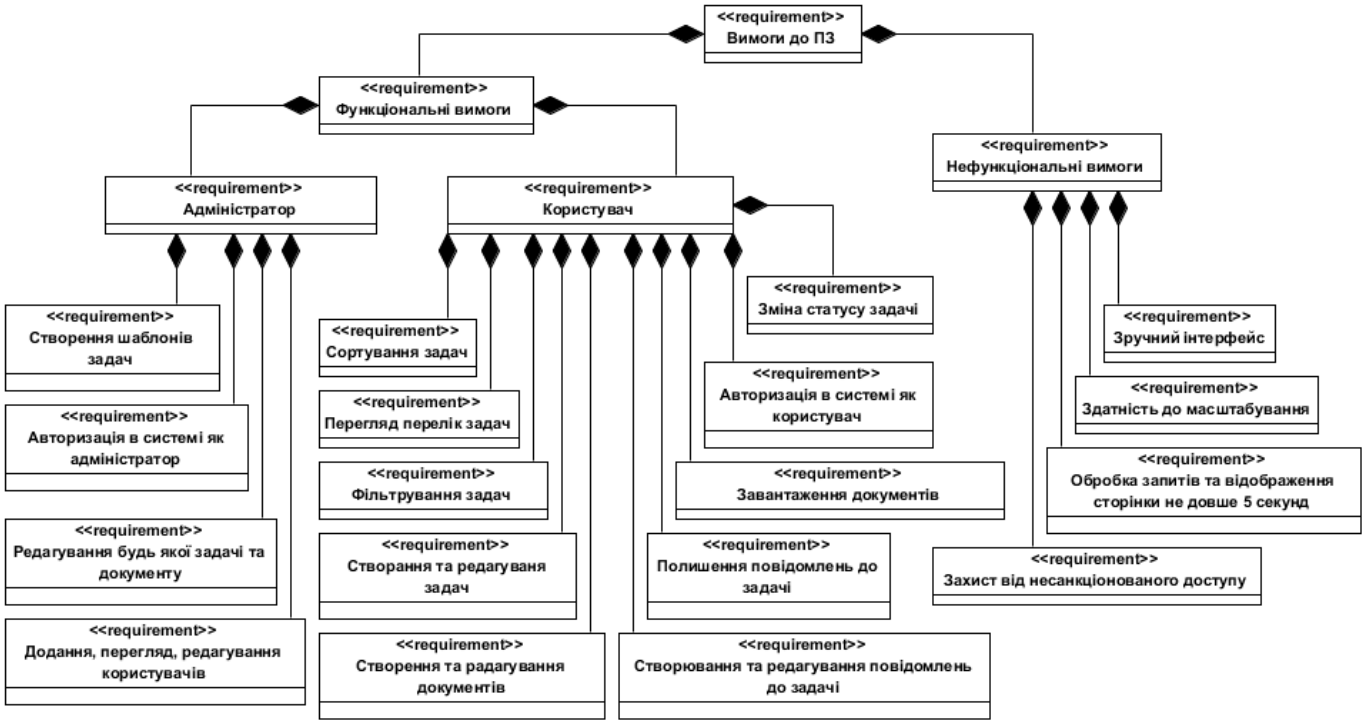

Рисунок 2.5 – Діаграма вимог

Г. Діаграми діяльності

Діаграма діяльності – це вид діаграми, що застосовується в UML для моделювання динамічних аспектів систем. По суті, це графічні зображення робочих процесів поетапних дій та дій [1] з підтримкою вибору, ітерації та паралельності. В Уніфікованій мові моделювання діаграми діяльності призначені для моделювання як обчислювальних, так і організаційних процесів (тобто робочих процесів), а також потоків даних, що перетинаються з відповідними видами діяльності. Хоча діаграми діяльності в основному показують загальний потік контролю, вони також можуть включати елементи, що показують потік даних між діями через один або кілька сховищ даних [15].

На рисунках 2.6-2.7 зображені діаграми діяльності в нотації UML для системи, що розроблюється. Були описані наступні специфікації поведінок, що можуть виконуватися: створення задачі в системі та взяття її до виконання, виконання задачі з використанням внутрісистемних засобів розробки документації.

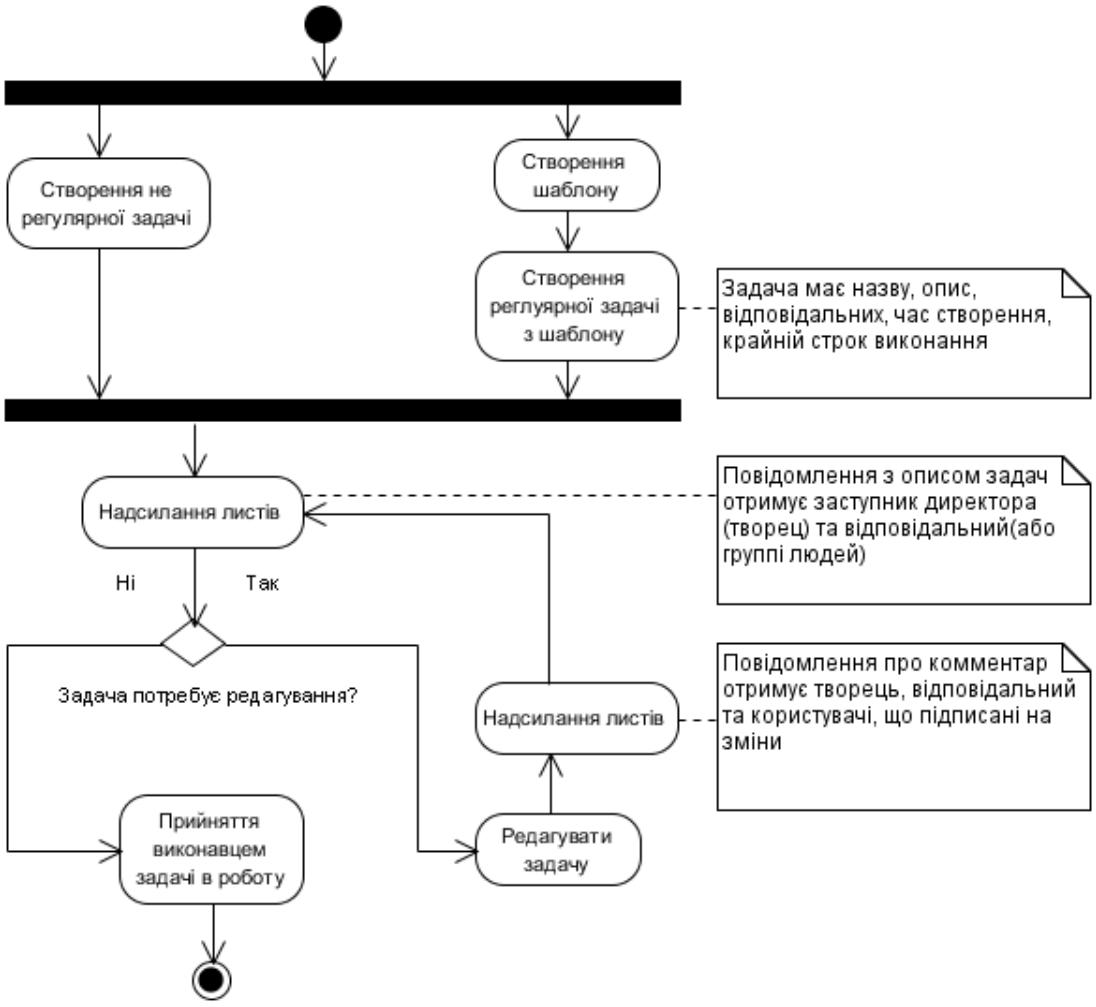

Рисунок 2.6 – Створення задачі в системі та взяття її до виконання

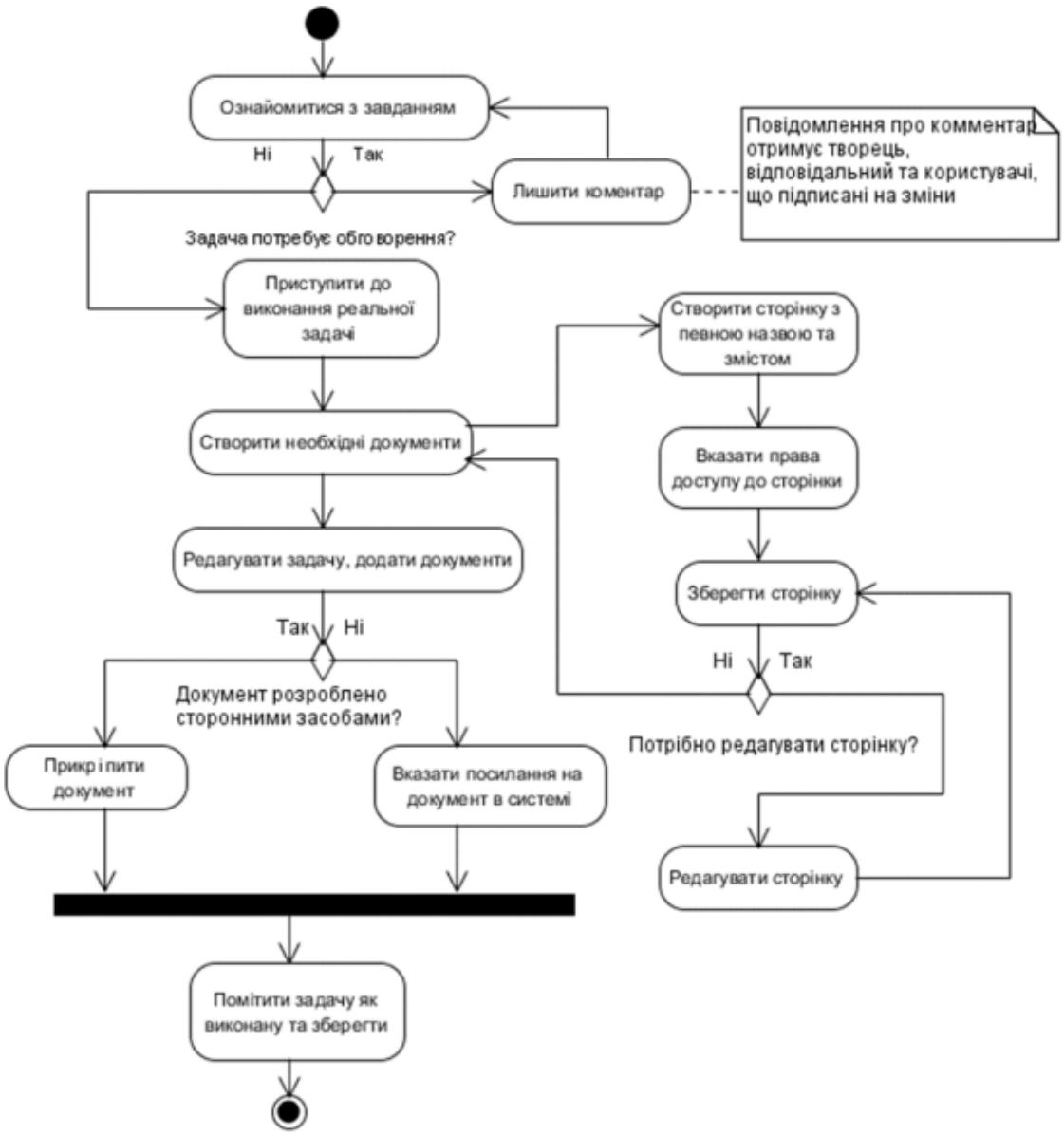

Рисунок 2.7 – Виконання задачі з використанням внутрісистемних засобів розробки документації

2.4 Розробка структури бази даних

На основі аналізу предметної області була розроблена логічна модель даних. В роботі була використана СУБД MySQL Server. База даних була створена шляхом генерації її на основі фізичної моделі даних за допомогою CASE-засобу ERWin [17].

Створена БД є сукупністю реляційних таблиць. Побудована концептуальна модель приведена на рисунку 2.8. Призначення основних таблиці у моделі даних наведено в таблиці 2.1.

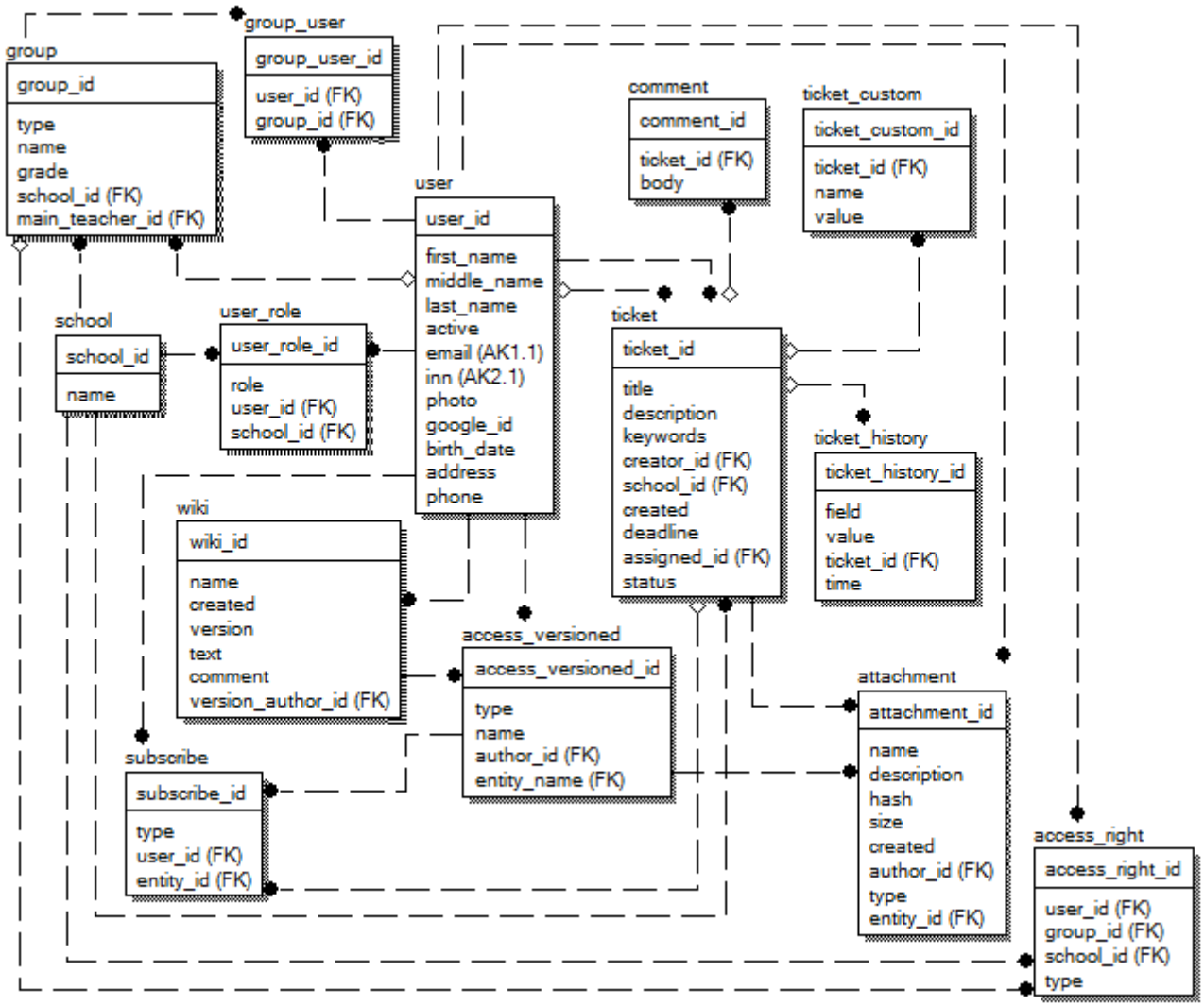

Рисунок 2.8 – Модель даних

Таблиця 2.1 – Призначення основних таблиць.

Назва	Призначення
Access right	Права доступу до задач.
Access versioned	Права доступу до сторінки документації.
Attachment	Файли прикріпленні до задачі або документації.
Comment	Коментар до задачі.
Group	Мінімальний набір основних даних про класи та гуртки, до яких належить користувач.
Subscribe	Підписка на повідомлення про зміни у задачі або документації.
Ticket	Основні відомості про задачу в системі.

Продовження таблиці 2.1

Ticket custom	Нестандарті поля для задач.
Ticket history	Дані про те, як змінювались значення полів у задачі.
User	Мінімальний набір основних текстових даних про користувача, такі як адреса, телефон та інше.
User role	Відомості про те, яку посаду та в якій школі займає користувач.
Wiki	Версія сторінки документації.

2.5 Вибір цільового варіанту архітектури програмного забезпечення

За для вибору архітектуру, спочатку необхідно розглянути існуючі варіанти.

«Товстий» клієнт (rich client) – це мережевий комп’ютер, що має безліч програм або ресурсів, що зберігаються локально, і мало залежить від мережевих ресурсів, таких як допоміжні накопичувачі, програвачі CD-RW / DVD або програмні програми. Як правило, користувачі віддають перевагу жирним клієнтським комп'ютерам перед тонкими клієнтами, тому що жирні клієнти дозволяють легко налаштовувати та забезпечувати б ільший контроль за встановленими програмами та конфігурацією системи. Оскільки вихід генерується локально, товстий клієнт також забезпечує більш складний графічний інтерфейс користувача (GUI) та зменшує навантаження сервера. [18].

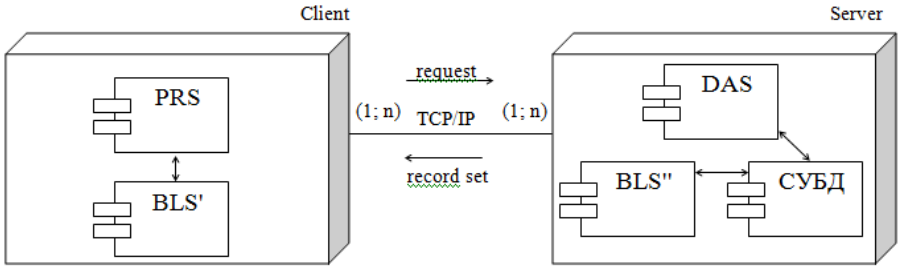

Рисунок 2.9 – Системна архітектура «товстий» клієнт

«Тонкий» клієнт (thin client) – це комп’ютер, який використовує ресурси, розміщені всередині центрального сервера, на відміну від жорсткого диска. Тонкий клієнт підключається до серверного середовища, в якому розміщується більшість

програм, пам'яті та конфіденційних даних, необхідних користувачеві. Тонкі клієнти також можуть підключатися до серверів, що базуються в хмарі. У багатьох випадках тонкий клієнтський комп'ютер є ефективною заміною персонального комп'ютера (ПК). Це також може бути найкращим рішенням, особливо тому, що це дозволяє IT-команді створити інфраструктуру віртуального робочого столу (VDI). За допомогою тонкого налаштування клієнта ви можете придбати нові робочі станції для співробітників, які працюють віддалено або власними силами, за нижчою вартістю, ніж якщо ви надасте кожному свій робочий стіл. Крім того, у вас є можливість централізувати своє рішення безпеки, захистивши сервер, до якого підключаються різні тонкі клієнти. [18].

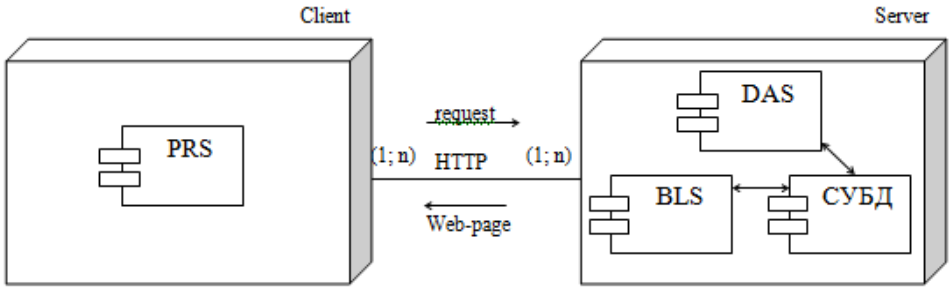

Рисунок 2.10 – Системна архітектура «тонкий» клієнт

Трирівнева архітектура довідкової системи забезпечує повне фізичне розділення програмних послуг та необхідних для них обчислювальних ресурсів. Ця архітектура системи дозволяє досягти максимального масштабування системи, вивантажити сервер баз даних і підвищити продуктивність. До недоліків можна віднести складність розгортання та адміністрування такої системи, високі вимоги до продуктивності серверів додатків та серверів баз даних [18].

Базова архітектура трирівневий «клієнт-сервер» з виділеним сервером додатків зображена на рисунку 2.11.

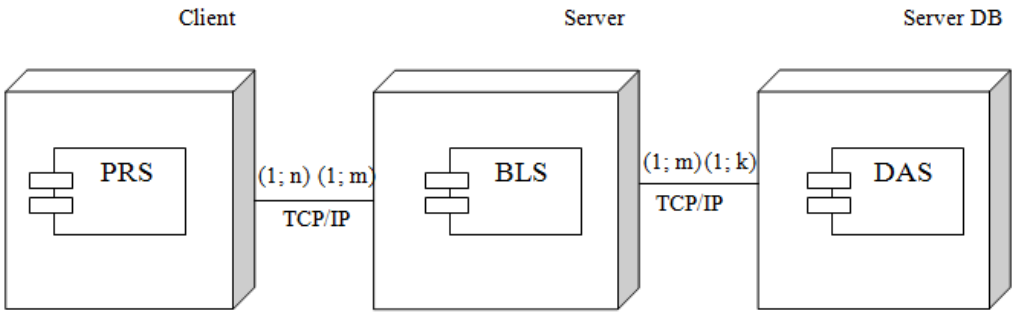

Рисунок 2.12 – Системна архітектура трирівневий «клієнт-сервер»

Програмне забезпечення розроблено та спроектовано для роботи в браузері на персональних комп'ютерах. Система повинна виконувати багато дій поза очима користувача: перевірка прав доступу, надсилання електронних листів, сортування та фільтрування великих обсягів даних. Система повинна бути розподіленою та стійкою до несправностей. Тому для цього завдання доцільно використовувати «тонку» архітектуру клієнта для передачі більшої частини обробки даних на сервер, але не надто багато, щоб збільшити складність розробки та обслуговування системи.

Задля підвищення надійності розробляємо системи та швидкості її розробки можливо використати відкрите програмне забезпечення для управління проектами Trac. Trac - засіб управління проектами та відстеження помилок в програмному забезпеченні. Є відкритим програмним забезпеченням, розробленим і підтримуваним компанією Edgewall Software. Він використовує мінімалістичний веб-інтерфейс, заснований на технології Wiki, і дозволяє організувати перехресні гіперпосилання між базою даних зареєстрованих помилок, системою контролю версій і вікі-сторінками. Це дає можливість використовувати Trac в тому числі і як веб-інтерфейс для доступу до системи контролю версій Subversion і Git, а також, через плагіни, до Mercurial, Bazaar і іншим.

Підтримуються бази даних SQLite, PostgreSQL, MySQL і MariaDB.

Trac написаний на мові програмування Python і в даний час поширюється по модифікованої ліцензії BSD. В якості системи HTML-шаблонів веб-інтерфейсу Trac до версії 0.11 використовував ClearSilver. Нові версії, починаючи з 0.11, використовують розроблену в Edgewall систему шаблонів Genshi [19, 37].

Для його модифікації і адаптації під існуючу задачу необхідно розробити плагін, який буде доповнювати систему необхідним функціоналом. Так само необхідно доопрацювати клієнтську частину, щоб підвищити зручність використання. Система, що розробляється, матиме архітектуру представлену на рисунку 2.13.

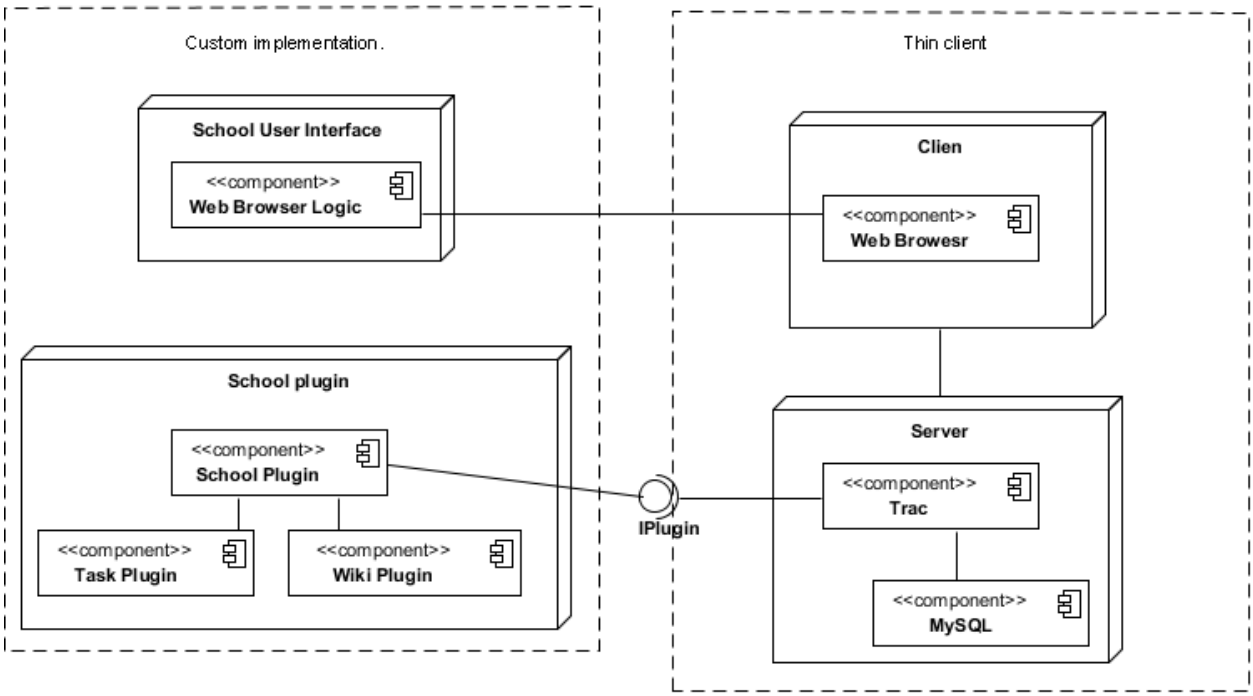

Рисунок 2.13 – Архітектура розробляємої системи

2.6 Вибір інструментальних засобів

A. Python

Python - інтерпретована об'єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Вбудовані структури даних високого рівня в поєднанні з динамічним набором тексту та динамічним прив'язуванням роблять його дуже привабливим для швидкої розробки додатків, а також для використання в якості мови сценаріїв або склеювання для з'єднання існуючих компонентів разом. Простий, легкий у засвоєнні синтаксис Python підкреслює читабельність і, отже, зменшує витрати на обслуговування програми. Python підтримує модулі та пакети, що

заохочує модульність програми та повторне використання коду. Інтерпретатор Python та обширна стандартна бібліотека доступні у вихідній або двійковій формі безкоштовно для всіх основних платформ і можуть вільно розповсюджуватися.

Часто програмісти закохуються в Python через підвищену продуктивність, яку він забезпечує. Оскільки не існує кроку компіляції, цикл редагування-тестування-налагодження неймовірно швидкий. Налагодження програм Python дуже просто: помилка або неправильний ввід ніколи не спричинить помилку сегментації. Натомість, коли інтерпретатор виявляє помилку, виникає виняток. Коли програма не вловлює виняток, інтерпретатор друкує трасування стека. Налагоджувач рівня джерела дозволяє перевіряти локальні та глобальні змінні, оцінювати довільні вирази, встановлювати точки зупинки, переходити через код за рядком за один раз тощо. Налагоджувач написаний на самому Python, що свідчить про інтроспективну силу Python. З іншого боку, часто найшвидшим способом налагодження програми є додавання кількох операторів друку до джерела: швидкий цикл редагування-тестування-налагодження робить цей простий підхід дуже ефективним. [20, 26, 27, 28].

Б. JavaScript

JavaScript спочатку був створений, щоб «оживити веб-сторінки». Програми на цій мові називаються сценаріями. Їх можна записати прямо в HTML веб-сторінки та запускати автоматично під час завантаження сторінки. Сценарії надаються та виконуються як звичайний текст. Їм не потрібна спеціальна підготовка чи складання для запуску. У цьому аспекті JavaScript сильно відрізняється від іншої мови, яка називається Java.

Сьогодні JavaScript може виконуватися не лише у браузері, але і на сервері, або фактично на будь-якому пристрої, що має спеціальну програму, яка називається механізмом JavaScript. У браузері є вбудований механізм, який іноді називають «віртуальною машиною JavaScript». [21, 29, 30, 31].

В. Мова розмітки гіпертексту HTML

HTML розшифровується як мова розмітки гіпертексту. Це дозволяє користувачеві створювати та структурувати розділи, абзаци, заголовки, посилання та

цитати для веб-сторінок та додатків. HTML не є мовою програмування, тобто він не має можливості створювати динамічну функціональність. Натомість це дозволяє упорядковувати та форматувати документи, подібно до Microsoft Word. Під час роботи з HTML ми використовуємо прості структури коду (теги та атрибути) для розмітки сторінки веб-сайту. Наприклад, ми можемо створити абзац, помістивши вкладений текст у початковий тег і закриття . Загалом, HTML - це мова розмітки, яка дійсно зрозуміла і проста у вивченні навіть для початківців початківців у створенні веб-сайтів. Ось що ви дізнаєтесь, прочитавши цю статтю: [22, 36].

Г. Мова опису зовнішнього вигляду документа CSS

Каскадні таблиці стилів, які залюбки називають CSS, - це проста мова дизайну, призначена для спрощення процесу зробити веб-сторінки презентабельними. CSS обробляє зовнішній вигляд частини веб-сторінки. За допомогою CSS ви можете керувати кольором тексту, стилем шрифтів, інтервалом між абзацами, розміром та розміщенням стовпців, які фонові зображення або кольори використовуються, дизайном макета, варіаціями відображення для різних пристроїв та розмірами екрану а також безліч інших ефектів. CSS легко вивчити і зрозуміти, але він забезпечує потужний контроль над поданням HTML-документа. Найчастіше CSS поєднується з мовами розмітки HTML або XHTML. [23].

Д. MySQL

MySQL - це система управління базами даних, яка дозволяє управляти реляційними базами даних. Це програмне забезпечення з відкритим кодом, яке підтримується Oracle. Це означає, що ви можете використовувати MySQL, не платячи ні копійки. Крім того, якщо ви хочете, ви можете змінити його вихідний код відповідно до ваших потреб. Незважаючи на те, що MySQL є програмним забезпеченням з відкритим кодом, ви можете придбати комерційну ліцензійну версію в Oracle, щоб отримати послуги преміум-підтримки. MySQL досить легко освоїти в порівнянні з іншим програмним забезпеченням для баз даних, таким як Oracle Database або Microsoft SQL Server. MySQL може працювати на різних платформах UNIX, Linux, Windows тощо. Ви можете встановити його на сервері або навіть на робочому столі. Крім того, MySQL надійний, масштабований і швидкий. [24, 34, 35].

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ АВТОМАТИЗАЦІЇ ДІЯЛЬНОСТІ СПІВРОБІТНИКІВ СЕРЕДНІХ ШКІЛ

3.1 Розробка функціоналу створення та редагування задач та шаблонів.

Основною одиницею в системі – є задача. Вона включає все необхідні базові поля для створення задач: тип; школа; назва; тіло; творець – користувач, який створив задачу, відповідальні – відповідальні за виконання цієї задачі користувачі (рис. 3.1).

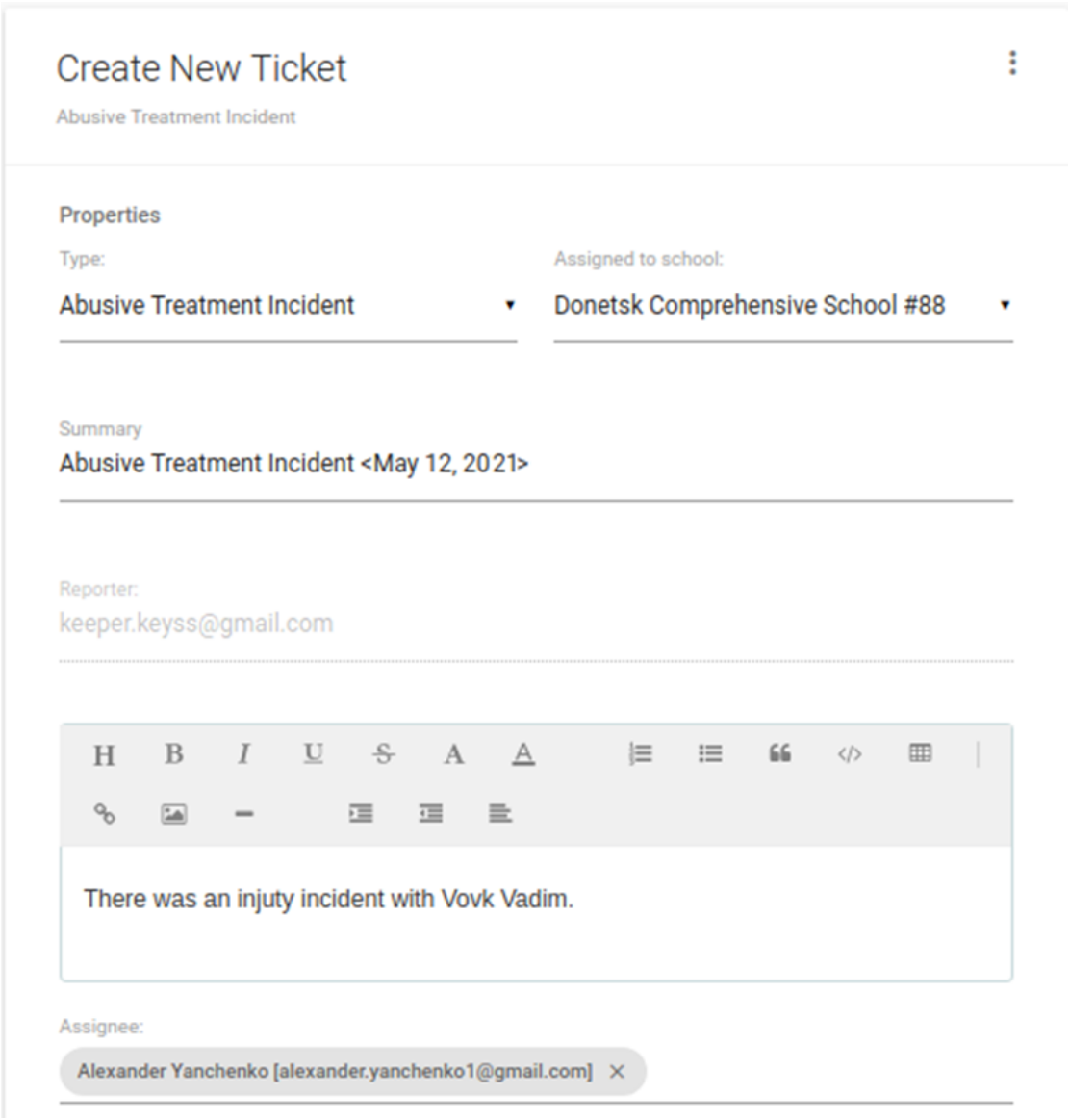

Рисунок 3.1 – Перша частина полів задачі.

Також задача включає до основних полів, такі як: користувачі, яких слід повідомляти про зміни в задачі; строк виконання задачі; ключові слова (рис 3.2).

Окрім основних полів, у кожного типу задачі можуть бути поля, які необхідні виключно для цієї задачі. Наприклад, для задачі «Неналежного поводження» такими полями є «повідомляти директора», тип інциденту, дата інциденту, причина інциденту, учень який приймав участь. Ці поля конфігуруються для кожного типу задачі особисто, та можуть являти собою: вибір зі списку варіантів, вибір зі списку варіантів з автоматичним доповненням, текстове поле, дата, перемикач (рис. 3.2).

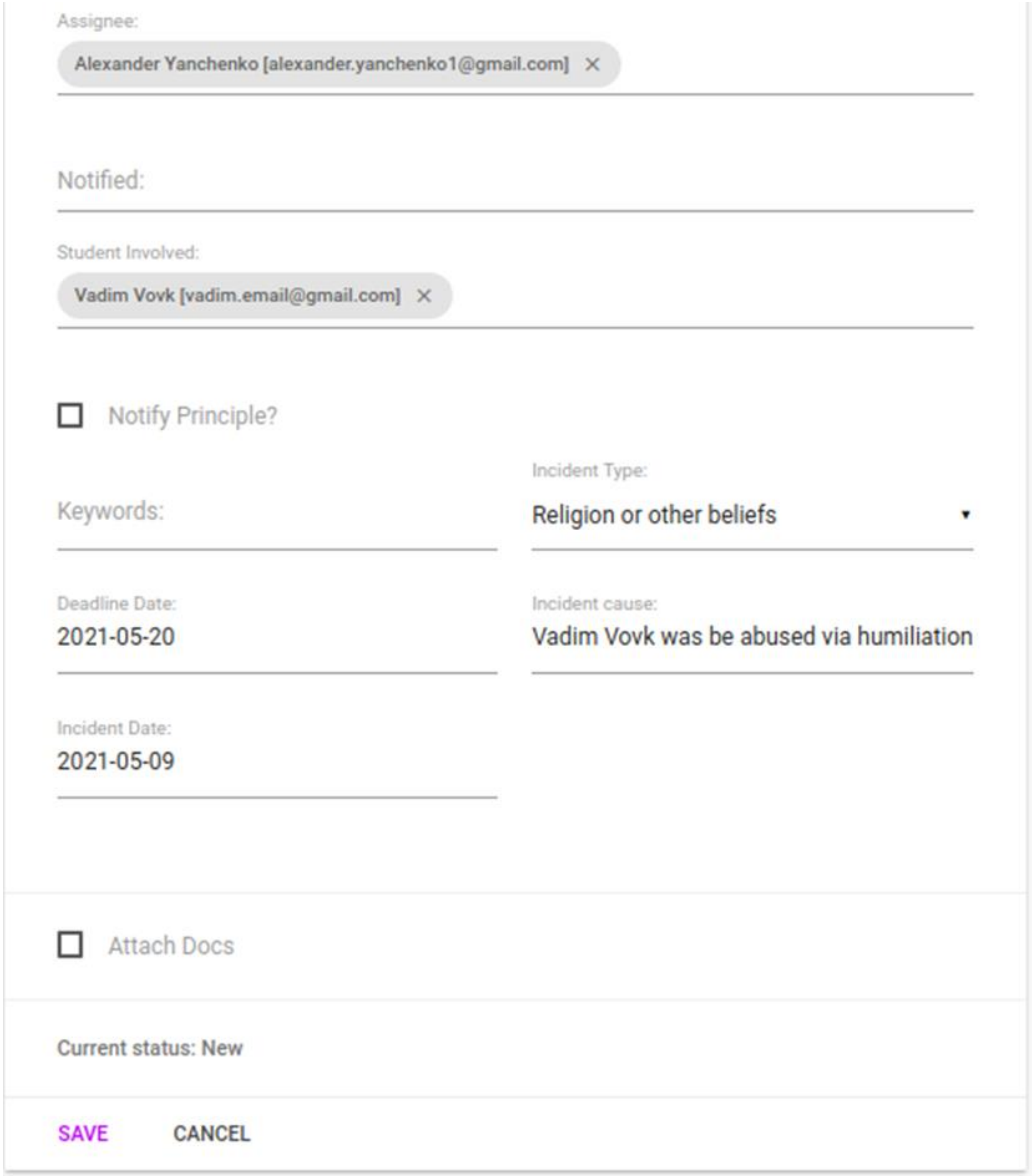

Рисунок 3.2 – Друга частина полів задачі.

Для того, щоб зменшити навантаження на заступника директора та автоматизувати створення задач, доцільно створити в системі таку одиницю, як шаблон задачі. Шаблон задачі має додаткові поля: частота створення та кількість днів перед крайнім строком (рис. 3.3). Частота створення вказує з якою періодичністю будуть створюватись типові задачі: кожний день, кожну неділю, кожен місяць, кожні три місяця, раз на пів року, раз на рік. У створеної з шаблону задачі вказується батьківський шаблон (рис. 3.4).

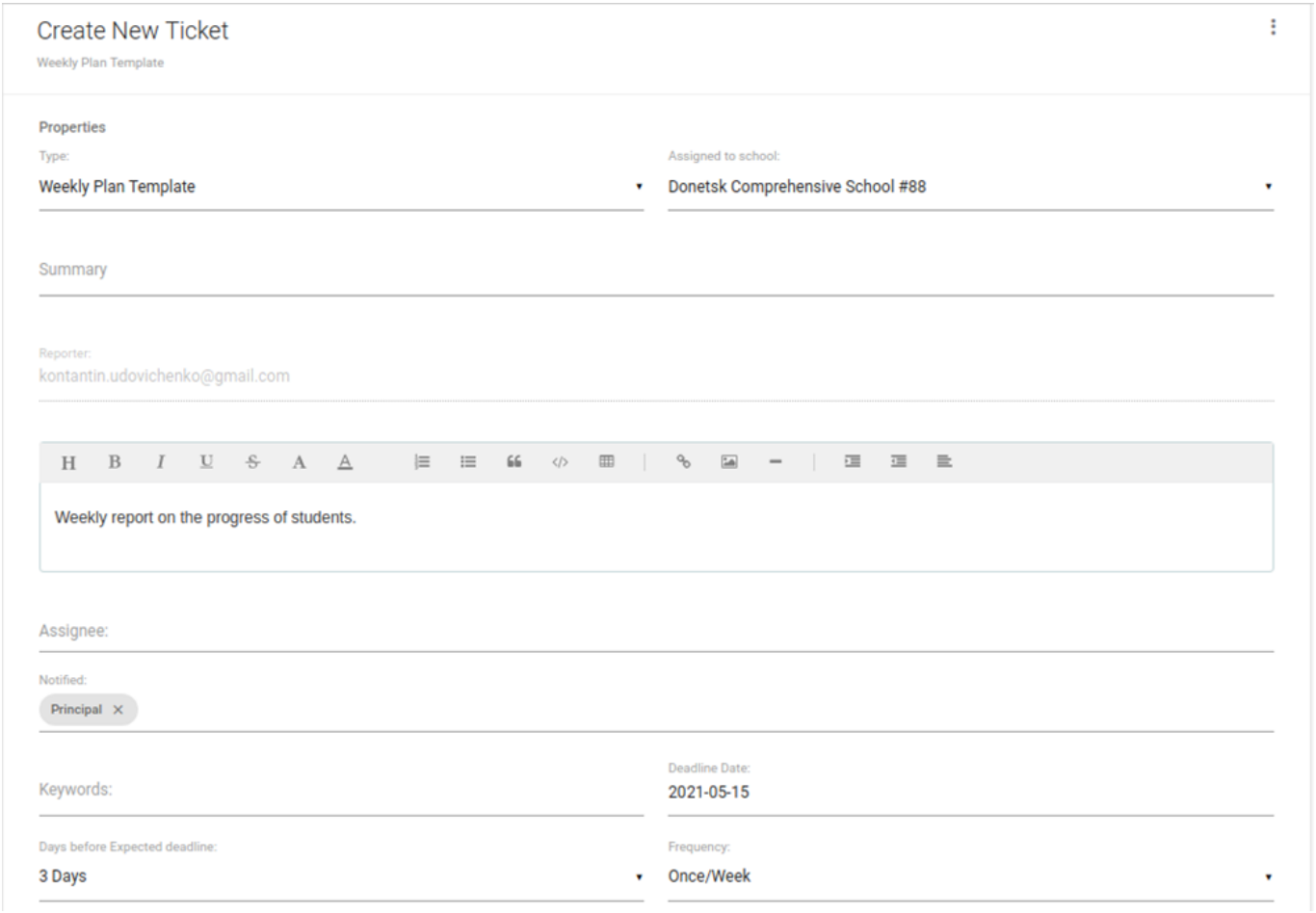

Рисунок 3.3 – Шаблон типової задачі.

До кожної задачі можливо додати коментар та прикріпити файл. Свої коментарі можливо редагувати або видалити. Прикріплені файл можливо завантажити або видалити (рис. 3.4).

Можливо переглянути список задач, які доступні користувачу в системі. Список можливо відсортувати за номером задачі, назвою, типом, відповідальним, строком закінчення, статусом (рис. 3.5).

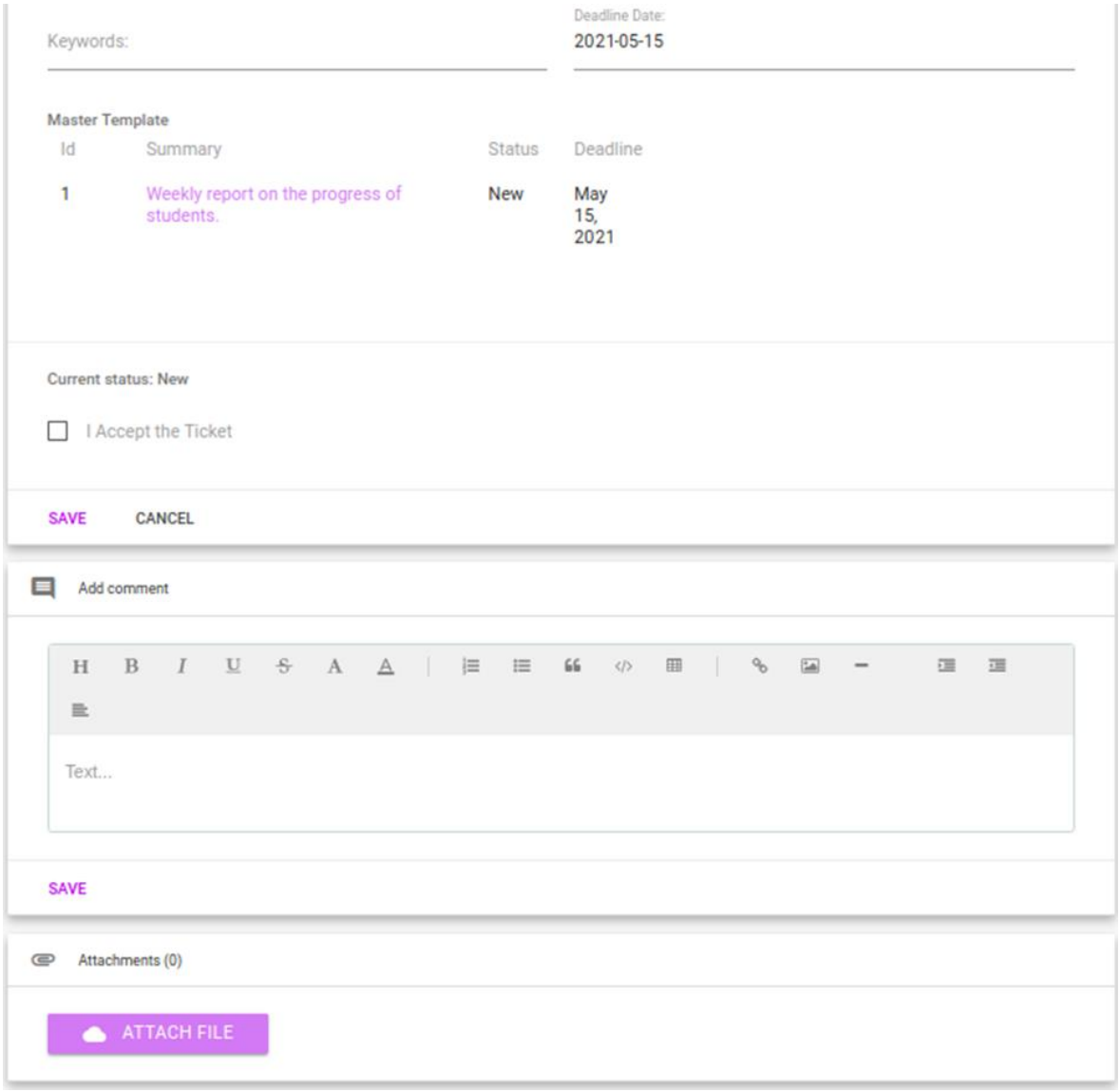

Рисунок 3.4 – Задача створена з шаблону.

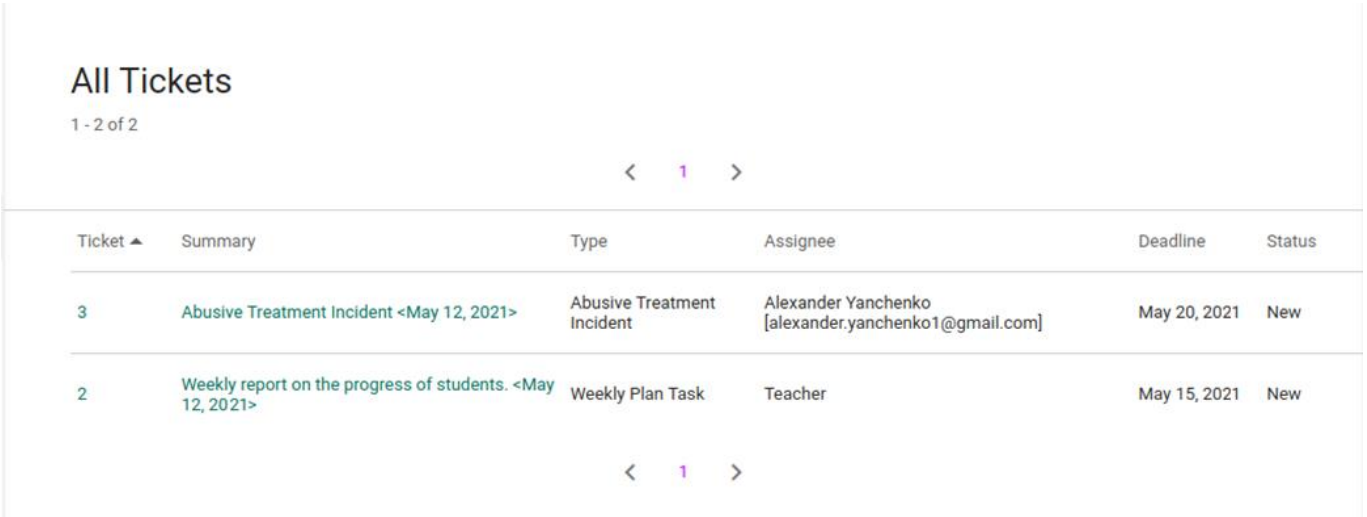

Рисунок 3.5 – Список доступних задач з можливістю сортування.

Про кожну зміну в задачі, все вказані в ній користувачі отримають повідомлення про зміни з описом задачі та вказанням які поля було змінено, їх попередні та нові значення. Приклад листа на рисунку 3.6.

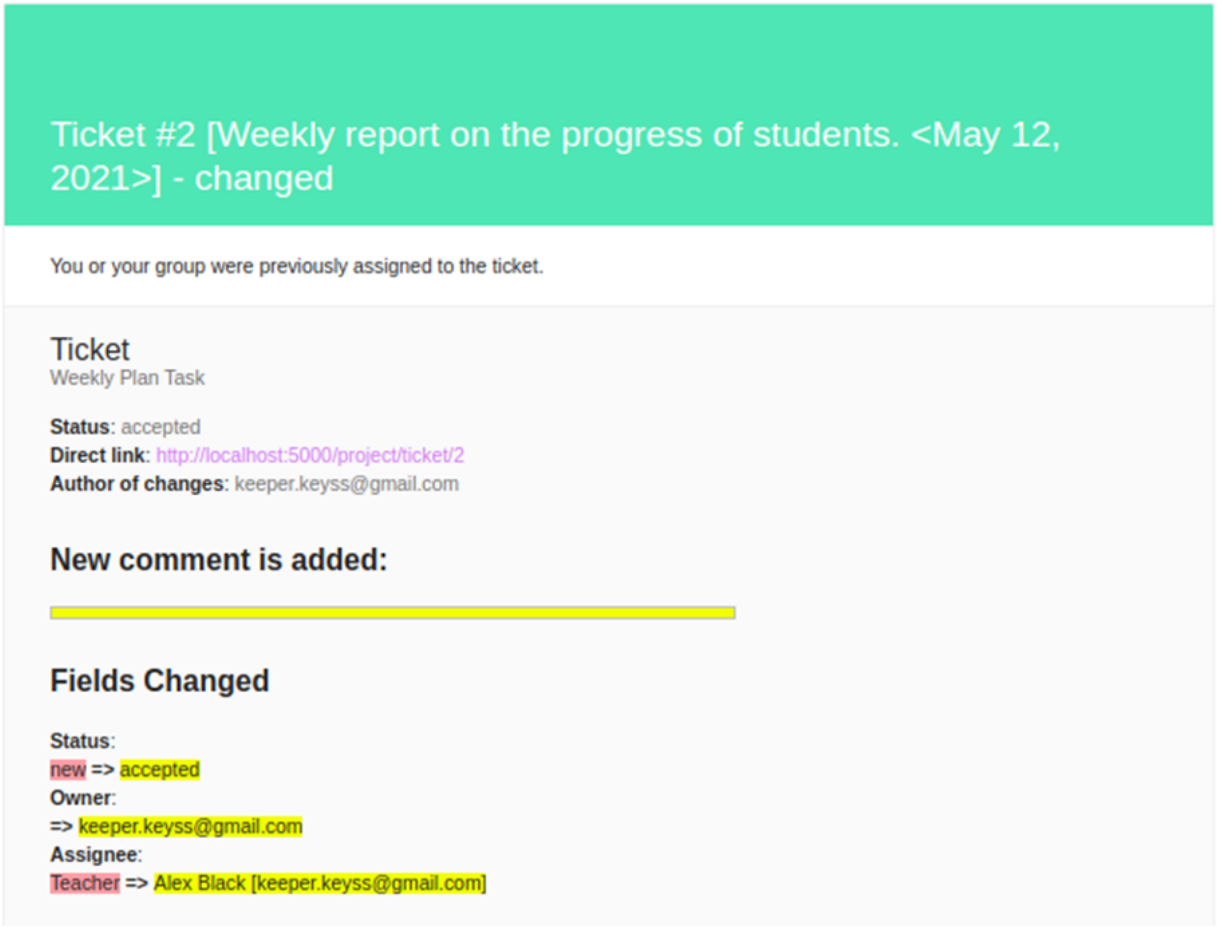

Рисунок 3.6 – Лист про зміни у задачі.

Користувач не має доступу до задачі, якщо він не створив її, не є виконавцем, або не належить до групи вказаних виконавців, та не є користувачем, який є в списку тих, кого повідомляти про зміні в цій задачі. Користувач не бачить цю задачу в списку та не має доступу до неї по прямому посиланню. При спробі переглянути таку задачу, він побачить повідомлення про нестачу прав для цієї дії (рис. 3.7).

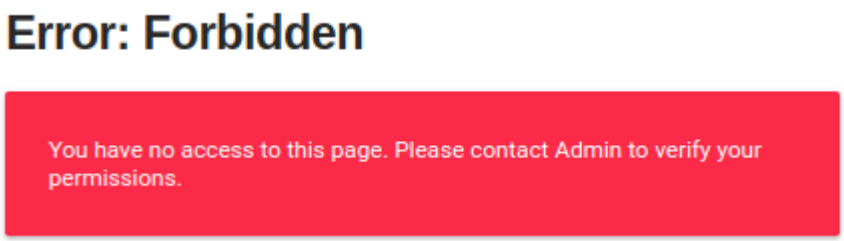

Рисунок 3.7 – Повідомлення про нестачу прав для перегляду задачі.

3.2 Розробка функціоналу створення та редагування документів.

Задачі дозволяють додавати документи різних видів: текстові, відео, аудіо. Але це не дуже зручно – необхідно надати доступ у всім користувачам перелічивши їх, а бо групи. Також це не дає можливості переглянути хоча б текстові документи онлайн – їх необхідно завантажувати. Тому с систему додано одиницю – сторінка документації. Документація підтримує wiki розмітку, керування версіями, та правами доступу. Wiki розмітка дозволяє використовувати такі типові засоби форматування, як стилі для тексту, колір тексту, колір фону, списки, цитування, створення таблиць, вирівнювання (рис. 3.8).

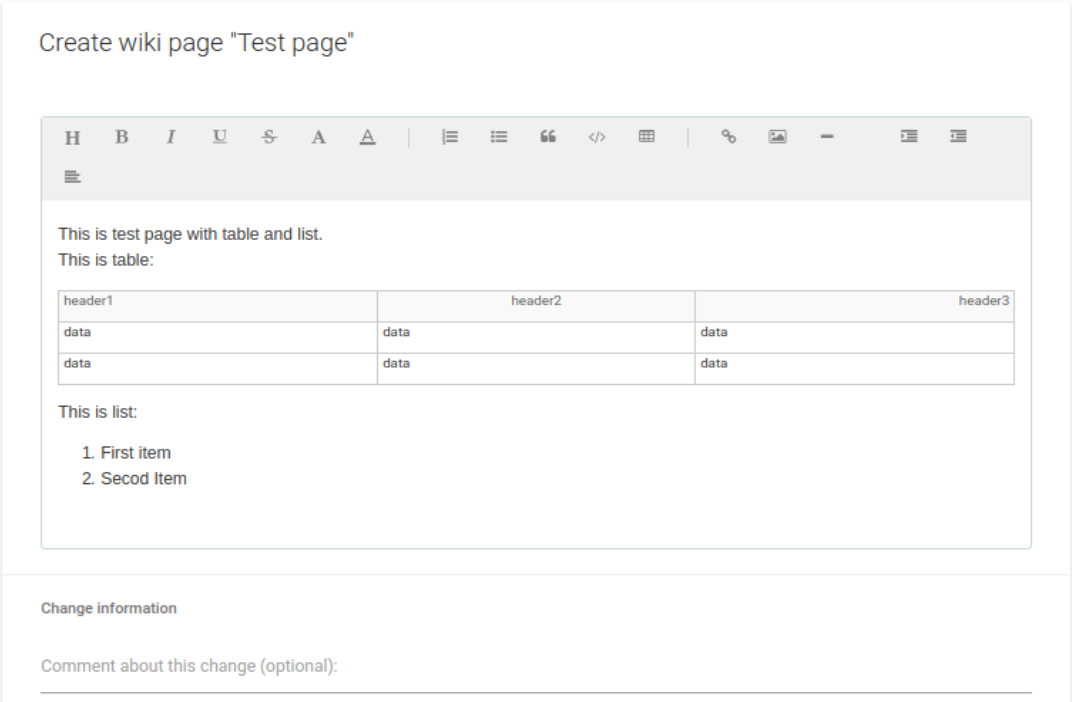

Рисунок 3.8 – Створення документації, форматування тексту.

Для того, щоб лімітувати доступ до певних документів використовується встановлення прав доступу. Вони дозволяють вказати яка зі шкіл, або всіх шкіл, та ролей, або певних користувачів, матимуть доступ до цього документу (рис. 3.9).

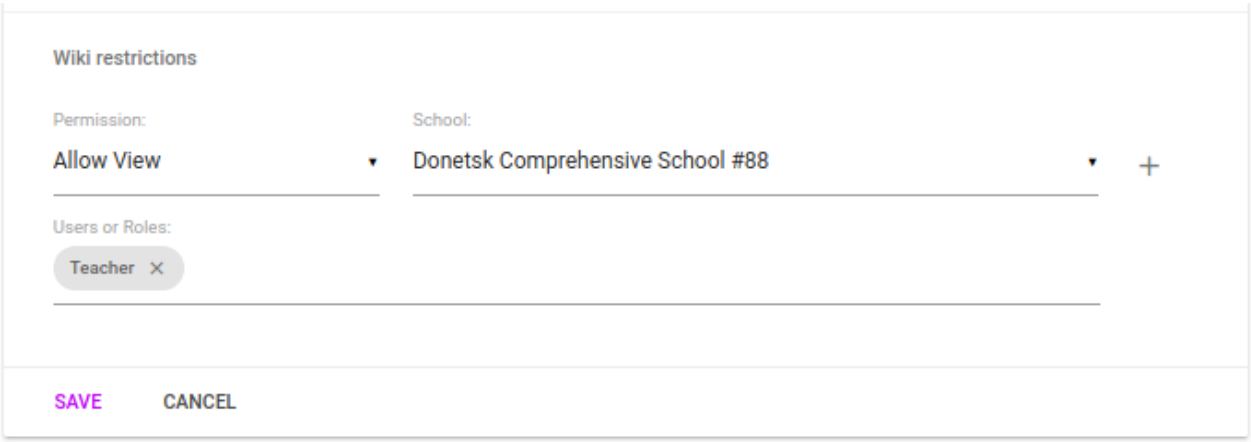

Рисунок 3.9 – Встановлення прав доступу до документу.

До кожного документу можливо задати матимуть ці користувачі доступ на перегляд, редагування чи не матимуть доступу зовсім (рис. 3.10).

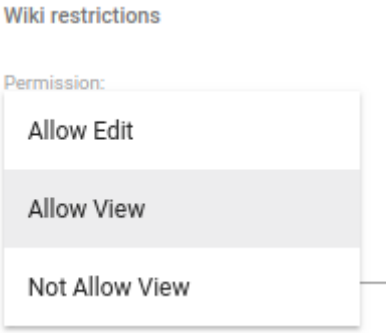

Рисунок 3.10 – Варіанти доступу до документу.

Повний перелік дій з документом: редагувати сторінку, перейменувати сторінку, видалити версію, видалити сторінку, додати файл, завантажити у вигляді тексту, завантажити у вигляді електронного документу – pdf, завантажити у вигляді текстового документу – open document (рис 3.11). Лише той, хто створив документ може його видалити та перейменувати. Редагувати можуть ті, хто має право на

редагування. Ті хто має право на перегляд – можуть переглянути сторінку, та завантажити її у вигляді файлу.

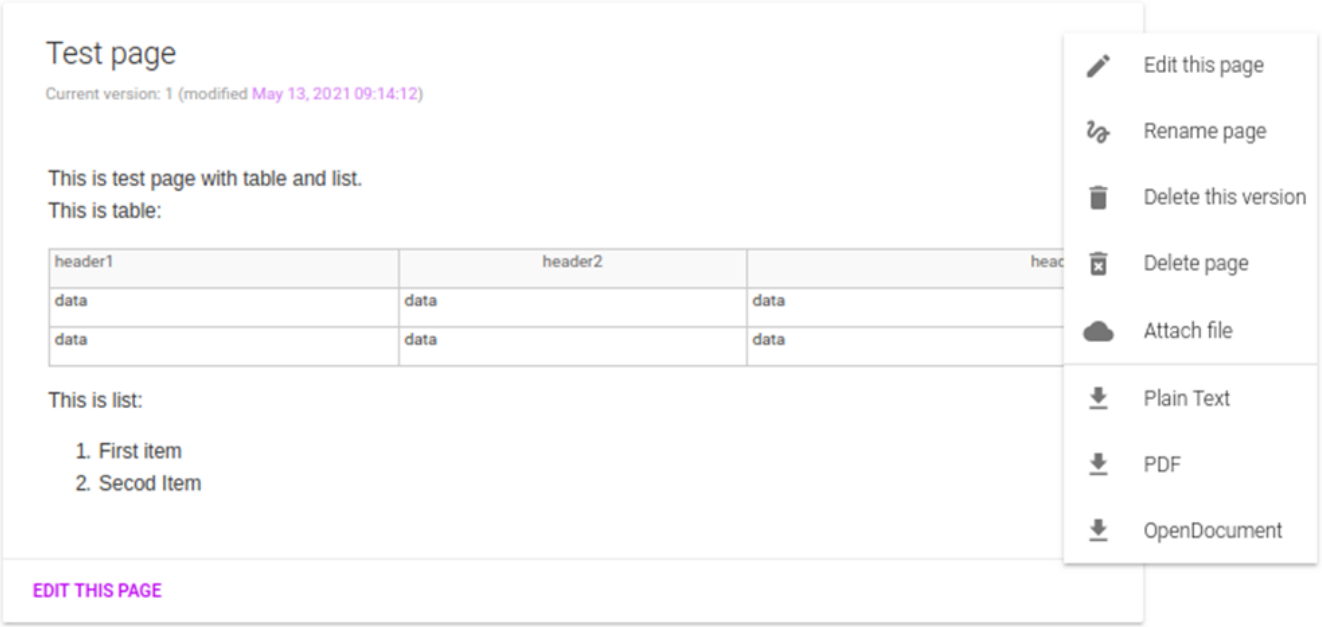

Рисунок 3.11 – Варіанти доступу до документу.

3.3 Розробка функціоналу додання користувачів до системи.

Для кожного користувача можливо вказати його ПІБ, дату народження, стать, додати фото (рис. 3.12).

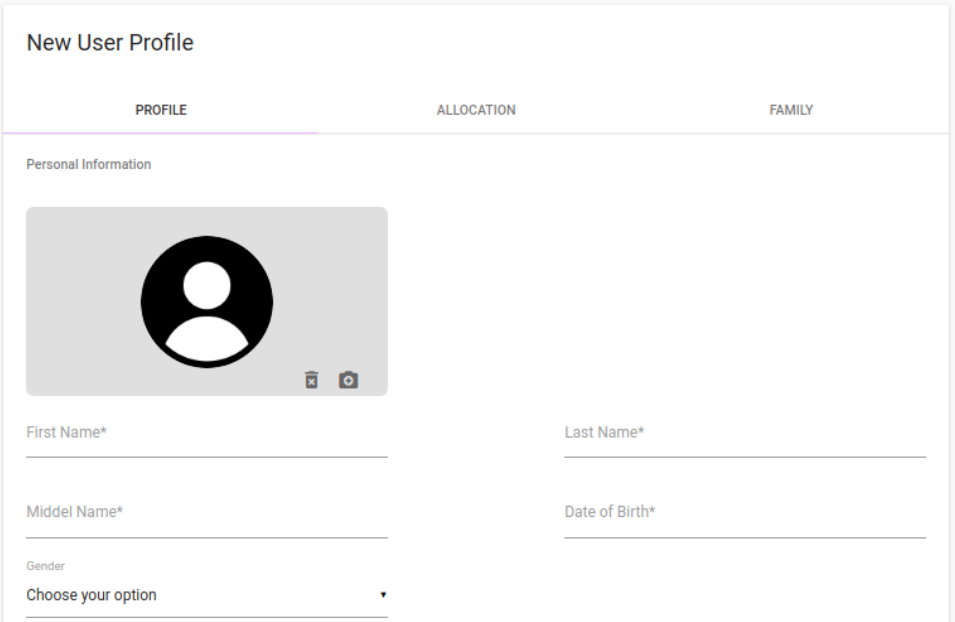

Рисунок 3.12 – Перша частина основних полів користувача.

Також користувачу можливо вказати адресу поштової скрині, мобільний та стаціонарний телефони, адресу, город, поштову адресу (рис 3.13).

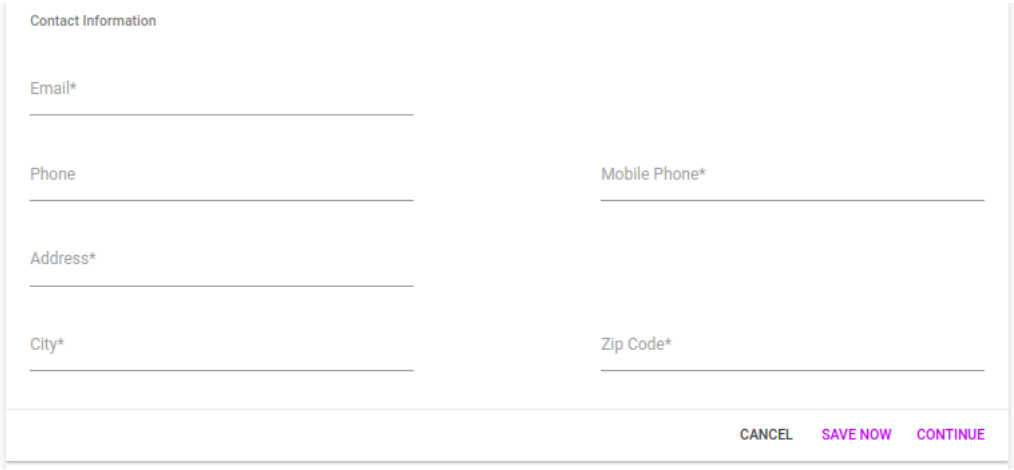

Рисунок 3.13 – Друга частина основних полів користувача.

Для кожного користувача необхідно вказати посаду та школу, де він її займає. При виборі учня або вчителя, в з’являться доступні виключно для них поля. У учня є групи та рік навчання. Групи – це шкільні об’єднання в яких учень є членом. У вчителя це поле вказує на шкільні об’єднання, які він курує (рис. 3.14).

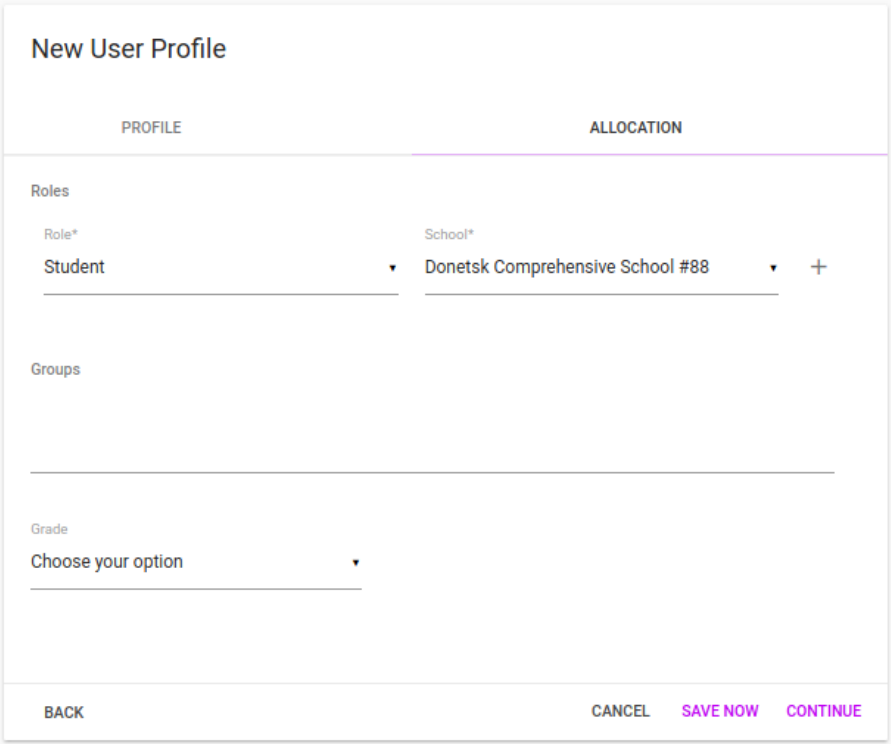

Рисунок 3.14 – Інформація про посаду користувача.

Вхід в систему виконується автоматично, якщо користувач має google пошту та э в системі як активний користувач.

3.4 Тестування.

Створення ефективних тест-кейсів може бути вкрай важливим для великих проектів, у разі, якщо поведінка частин додатку може змінюватися з різних причин. Мабуть, найбільш частою є проблема, коли велика група розробників працюють над одним і тим же, або суміжними модулями. Це може призводити до незапланованого зміни поведінки функцій, написаних іншими програмістами. Або робота в стислі терміни призводить до ненавмисному зміні критичних частин програми.

З часом проект наповнюється новими функціональними можливостями, що подовжує і ускладнює процес перевірки його роботи. Для автоматизації використовуються модульне тестування. Існують 2 підходи до побудови тестових сценаріїв:

– «White box» тестування – написання тестів грунтується на реалізації функціоналу. Тобто на ми перевіряємо за тими ж алгоритмами, на яких будуватиметься роботи модулів нашої системи. Такий підхід не гарантує коректність роботи системи в цілому.

– «Black box» тестування – створення сценаріїв базується на специфікаціях і вимогах до системи. Так можна перевірити правильність результатів роботи всього програми, проте подібний підхід не дозволяє відловити дрібні і рідкісні помилки. Розроблений інтерфейс є простим та інтуїтивно зрозумілим звичайним користувачам. Всі елементи керування є добре помітними, що полегшує роботу із застосуванням [25, 38, 39, 40].

Для перевірки працездатності ПЗ було створено набір тест-кейсів.

Тест-кейс – опис перевірки працездатності системи, або перелік кроків та очікуваних результатів. Заздалегідь представляються у вигляді таблиці, або списку, якщо використовується система управління тестами [41, 42].

В таблиці 3.1 представлені тест-кейси для перевірки коректної роботи системи.

Таблиця 3.1 – Тест-кейси перевірки системи

id	Назва	Кроки	Очікувані результати	Підтвердження
1	Створення задачі.	1. Вхід користувачем з правами заступника директора. 2. Відкриття сторінки створення задачі 3. Введення необхідних полів: тип, школа, назва, зміст, крайній строк. 3. Збереження задачі. 4. Відкриття створеної задачі за посиланням. 5. Перевірка полів задачі.	Задача була створена з усіма вказаними полями.	Успіх
2	Перевірка прав доступу.	1. Створення задачі. 2. Вхід в систему користувачем, який вказаний як той, кого повідомляти про зміни у задачі. 3. Перевірити можливість перегляду, але не редагування. 4. Вхід в систему відповідальним у задачі. 5. Перевірити можливість редагування задачі.	Вказаний як той, кого повідомляти – може переглянути задачу, але не редагувати. Відповідальний – може редагувати.	Успіх
3	Перевірка порядку дій.	1. Війти в систему як заступник директора. 2. Створення задачі, вказати відповідальним групу вчителів. 3. Вхід в систему вчителем. 4. Взяти задачу до виконання 5. Перевірити, що відповідальний змінився з групи «вчителі» на ім’я вчителя. 6. Перевірити, що заступник директора може лише переглянути задачу та не може редагувати. 7. Закрити задачу вчителем.	Вказаний в групі виконавців може взяти на себе задачу, творець матиме лише можливість переглянути задачу, виконуючий може закрити задачу.	Успіх
4	Перевірка списку задач.	1. Створити задачу. 2. Перевірити що вона з’явилась у списку. 3. Сортувати список. 4. Перевірити що задача залишилась у списку.	Задача після створення – з’являється у списку задач.	Успіх

Продовження таблиці 3.1

5	Створення документації.	1. Створити сторінку документації, що лише вчителя бачать цю сторінку. 2. Зайти в систему як вчитель 3. Перевірити наявність сторінки 4. Зайти в систему як мед. робітник. 5. Перевірити відсутність сторінки.	Сторінка документації створюється, права доступу працюють правильно.	Успіх
6	Створення користувача.	1. Вхід в систему як адміністратор. 2. Відкриття форми створення користувача 3. Заповнення усіх ключових полів та збереження форми. 4. Вхід новим користувачем в систему	Створення нового користувача та вхід ним в систему.	Успіх

Таким чином, було успішно перевірено створення задач, створення документації.

3.5 Дані по результатам часу виконання задач та прогноз часу виконання задач.

Кожна задача має дані коли вона була створена та коли вона була виконана. Ці дані необхідно зібрати та обробити. На їх підставі необхідно спрогнозувати час майбутнього виконання задач.

Тестові дані представлені в таблиці 3.2

Таблиця 3.2 – Тестові дані часу виконання задач

Номер тижня	1	2	3	4	5	6	7	8	9	10	11
Строк виконання, дні	10	9	10	9	8	7	7	7	6	6	5

В якості a_0 берем перше значення ряду, що дорівнює 10. α та β беремо рівним 0.3. a_0 та b_0 дорівнюють 10.

$$a_1 = \alpha y_1 + (1 - \alpha)(a_{t-1} + b_{t-1}) \quad (3.1)$$

$$b_1 = \beta(a_t - a_{t-1}) + (1 - \beta)b_{t-1} \quad (3.2)$$

$$y_1 = a_0 + b_0 \quad (3.3)$$

$$\hat{y} = \frac{|y_t - y_t^*|}{y_t} \quad (3.4)$$

Перша ітерація.

$$a_1 = 0.3 * 10 + (1 - 0.3) * 10 = 9.7$$
$$b_2 = 0.3 * (10 - 10) + (1 - 0.3) * 0 = -0.09$$
$$y_2 = 10 + 0$$
$$\hat{y}_1 = \frac{|10 - 10|}{10} = 1$$

Друга ітерація.

$$a_2 = 0.3 * 9 + (1 - 0.3) * 10 = 9.73$$
$$b_2 = 0.3 * (9.7 - 10) + (1 - 0.3) * 0 = -0.5$$
$$y_2 = 10 + 0 = 9.61$$
$$\hat{y}_2 = \frac{|9 - 10|}{9} = 0.04$$

Третя ітерація.

$$a_3 = 0.3 * 10 + (1 - 0.3) * (9.7 - 0.05) = 9.47$$
$$b_3 = 0.3 * (9.73 - 9.7) + (1 - 0.3) * (-0.05) = -0.115$$
$$y_3 = 9.7 - 0.09 = 9.67$$
$$\hat{y}_3 = \frac{|9 - 9.61|}{10} = 0.07$$

Четверта ітерація.

$$a_4 = 0.3 * 8 + (1 - 0.3) * (9.47 - 0.115) = 9.95$$
$$b_4 = 0.3 * (8.95 - 9.47) + (1 - 0.3) * (-0.115) = -0.237$$
$$y_4 = 9.47 - 0.115 = 9.335$$

$$\hat{y}_4 = \frac{|9 - 9.36|}{8} = 0.17$$

П’ята ітерація.

$$a_5 = 0.3 * 7 + (1 - 0.3) * (8.95 - 0.237) = 8.2$$
$$b_5 = 0.3 * (8.2 - 8.95) + (1 - 0.3) * (-0.237) = -0.39$$
$$y_5 = 8.95 - 0.24 = 8.71$$
$$\hat{y}_5 = \frac{|7 - 8.71|}{7} = 0.24$$

Шоста ітерація.

$$a_6 = 0.3 * 7 + (1 - 0.3) * (8.2 - 0.39) = 7.565$$
$$b_6 = 0.3 * (7.565 - 8.2) + (1 - 0.3) * (-0.39) = -0.464$$
$$y_6 = 8.2 - 0.39 = 7.8$$
$$\hat{y}_6 = \frac{|7 - 7.8|}{7} = 0.115$$

Сьома ітерація.

$$a_7 = 0.3 * 7 + (1 - 0.3) * (7.565 - 0.464) = 7.07$$
$$b_7 = 0.3 * (7.07 - 7.565) + (1 - 0.3) * (-0.464) = -0.473$$
$$y_7 = 7.565 - 0.464 = 7.1$$
$$\hat{y}_7 = \frac{|7 - 7.1|}{7} = 0.015$$

Восьма ітерація.

$$a_8 = 0.3 * 6 + (1 - 0.3) * (7.07 - 0.473) = 6.42$$
$$b_8 = 0.3 * (6.42 - 7.07) + (1 - 0.3) * (-0.473) = -0.527$$
$$y_8 = 7.07 - 0.473 = 6.6$$

$$\hat{y}_8 = \frac{|6 - 6.6|}{6} = 0.1$$

Дев’ята ітерація.

$$a_9 = 0.3 * 6 + (1 - 0.3) * (6.42 - 0.527) = 5.92$$
$$b_9 = 0.3 * (5.92 - 6.42) + (1 - 0.3) * (-0.527) = 0.517$$
$$y_9 = 6.42 - 0.527 = 5.4$$
$$\hat{y}_9 = \frac{|6 - 5.9|}{6} = 0.81$$

Десята ітерація.

$$a_{10} = 0.3 * 5 + (1 - 0.3) * (5.92 - 0.517) = 5.28$$
$$b_{10} = 0.3 * (5.29 - 5.92) + (1 - 0.3) * (-0.517) = -0.55$$
$$y_{10} = 5.92 - 0.517 = 5.4$$
$$\hat{y}_{10} = \frac{|6 - 5.4|}{6} = 0.08$$

Прогнозовано час виконання задачі на k+1=12.

$$y_{11} = 5.28 - 0.55 = 4.73$$

Побудуємо графі на основі цих даних(рис. 3.15).

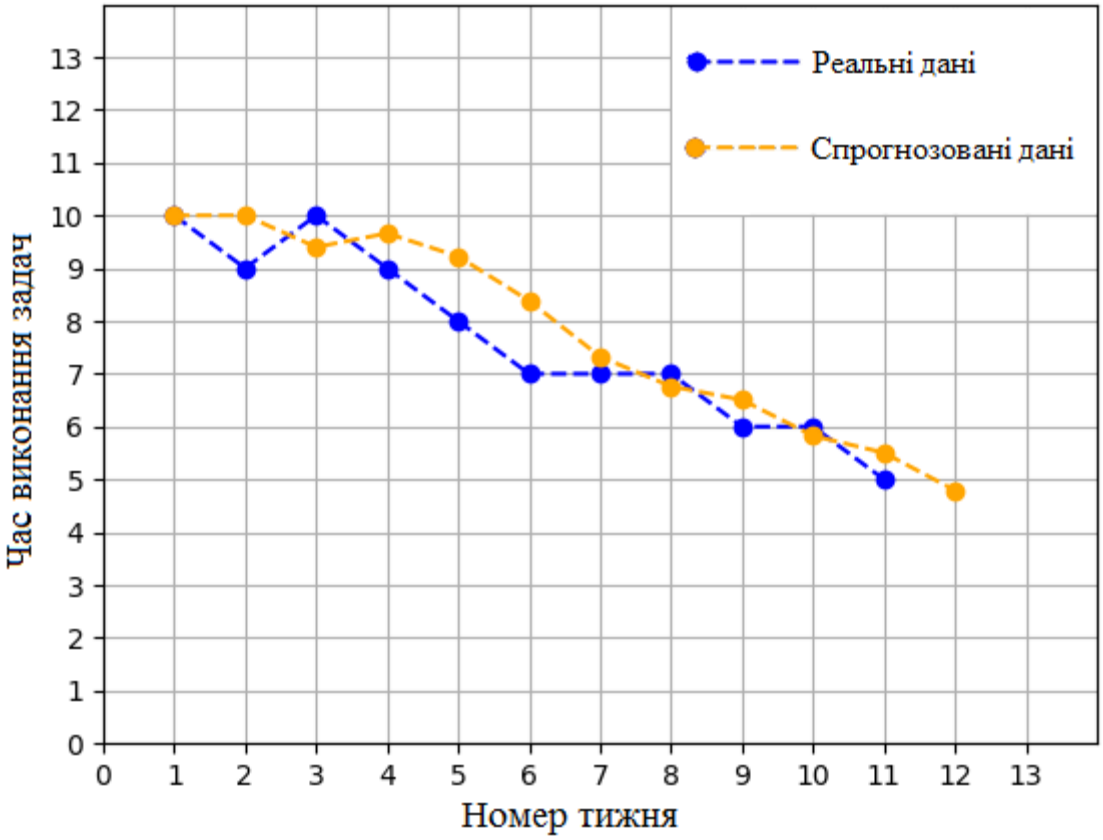

Рисунок 3.15 – Діаграма прогнозування часу виконання задач.

3.6 Техніко-економічне обґрунтування

А Обґрунтування доцільності розробки проекту

Даний ПЗ розроблений для інформаційної та аналітичної підтримки діяльності співробітників середніх шкіл. ПЗ орієнтований на заступників директора, які створюють задачі з управління закладом, та на педагогічний склад школи та персонал, які виконують ці задачі. Також ПЗ надає інформаційні засоби для розробки навчально-методичної документації.

Система вирішує наступні задачі:

– автоматизоване управління регулярними не регулярними та задачами, такими як: організація екскурсії, організація олімпіад, організація семінарів; моніторинг стану пожежної безпеки, моніторинг стану обладнання кабінетів, моніторинг стану спортивних приміщень та інвентарю, організації та координації роботи предметних (циклових комісій) та інше;

– надання інформаційних засобів для колективної розробки навчально-методичної документації;

– надання інформаційних засобів обліку співробітників.

У даному розділі представлене техніко-економічне обґрунтування проекту по розробці ПЗ для автоматизації діяльності співробітників середніх шкіл.

Б Оцінка конкурентоспроможності у порівнянні з аналогом

У якості програми для порівняння при розробці програмної компоненти був взятий інструмент Trello, створений Fog Creek Software (New York, USA).

Цей продукт був обраний у якості аналогу на основі таких факторів:

– схожий профіль;

– відповідність вимогам технічного завдання проекту;

– доступність для дослідження аналогу у зв'язку з його відкритим доступом.

Експлуатаційно-технічний рівень (ЕТР) розроблюваного продукту – це узагальнена характеристика його експлуатаційних властивостей, можливостей, ступеню новизни, що є основою якості продукту. Для визначення ЕТР продукту можна використати індекс експлуатаційно-технічного рівня $J_{\text{ЕТР}}$, який розраховується як сума часткових індексів, куди входять показники якості програмного продукту.

Тоді

$$J_{\text{ЕТР}} = \sum_{j=1}^{n} B_j \times X_j, \tag{3.5}$$

де $J_{\text{ЕТР}}$ – комплексний показник якості продукту за групою показників;

n – число показників, що розглядається;

B_j – коефіцієнти вагомості j-го показника у частках одиниці, що призначається у відповідності до потреб організації-замовника програмного продукту;

X_j – експертна оцінка j-го показника якості за вибраною шкалою оцінювання.

В таблиці 3.3 представлені результати розрахунку бально-індексним методом за п'ятибальною шкалою оцінювання.

Таблиця 3.3 – Розрахунок показника якості

Показник якості	Коефіцієнт вагомості, B_j	Проект		Аналог	
		X_j	$B_j \times X_j$	X_j	$B_j \times X_j$
Зручність роботи	0,17	4	0,68	3	0,51
Новизна	0,09	4	0,36	3	0,27
Відповідність профілю діяльності замовника	0,07	4	0,28	4	0,28
Ресурсна ефективність	0,08	3	0,24	3	0,24
Надійність ПЗ	0,09	3	0,27	2	0,18
Швидкість доступу до даних	0,2	5	0,1	3	0,06
Гнучкість настройки	0,14	4	0,56	3	0,42
Здатність до навчання персоналу	0,06	3	0,18	3	0,18
Співвідношення вартості/можливості	0,09	3	0,27	2	0,18
Узагальнений показник якості, J_{ETP}		2,94		2,32	

Відношення двох знайдених індексів називають коефіцієнтом технічного рівня A_k першого програмного продукту по відношенню до другого:

$$A_k = \frac{2,94}{2,32} = 1,27 \quad (3.6)$$

Так як коефіцієнт більше 1, то розробка проекту з технічної точки зору виправдана.

В. Планування комплексу робіт по розробці програмного забезпечення і оцінка трудомісткості робіт

Для розробки було задіяно дві людини: керівник проекту і виконавець (інженер-програміст).

Керівник виконує постановку задачі, курирує хід робіт і дає необхідні консультації при розробці системи. Виконавець відповідає за проектування інформаційного забезпечення, розробку бази даних, реалізацію алгоритмів, розробку інтерфейсних блоків і налагодження програми.

Вибір комплексу робіт по розробці проекту виконується згідно стандарту ISO/IEC 12207:2008 “System and software engineering — Software life cycle processes”, що встановлює стадії розробки програмних продуктів. Комплекс робіт наведений у таблиці 3.4.

Таблиця 3.4 – Комплекс робіт по розробці продукту

Зміст робот	Виконавці	Тривалість	Завантаження	
			Дні	%
1. Підготовка процесу розробки та аналізу вимог				
1.1 Дослідження та обґрунтування розробки				
1.1.1 Постановка задачі	Керівник Програміст	4	1 4	25 100
1.1.2 Збір вихідних даних	Керівник Програміст	10	2 10	20 100
1.2 Пошук аналогів та прототипів				
1.2.1 Аналіз існуючих методів вирішення задачі	Керівник Програміст	6	1 6	16 100
1.2.2 Обґрунтування необхідності розробки	Керівник Програміст	1	1 1	100 100
1.3 Аналіз вимог				
1.3.1 Визначення та аналіз вимог до програми	Керівник Програміст	6	2 6	33 100
1.3.2 Визначення структури вхідних і вихідних даних	Керівник Програміст	5	1 5	20 100

Продовження таблиці 3.4

1.3.3 Вибір технічних та програмних засобів реалізації	Керівник Програміст	3	1 3	33 100
1.3.4 Погодження та затвердження технічного завдання	Керівник Програміст	2	1 2	50 100
Підсумок по етапу 1	Керівник Програміст	37	10 37	27 100
2. Проектування				
2.1 Проектування програмної архітектури	Керівник Програміст	4	1 4	25 100
2.2 Технічне проектування компонентів програми	Керівник Програміст	8	0 8	0 100
Підсумок по етапу 2	Керівник Програміст	12	1 12	8 100
3. Програмування та тестування програмних модулів				
3.1 Програмування модулів	Керівник Програміст	19	0 19	0 100
3.2 Тестування програмних модулів	Керівник Програміст	10	0 10	0 100
3.3 Збірка та випробування програми	Керівник Програміст	5	1 5	20 100
3.4 Аналіз результатів випробувань	Керівник Програміст	3	1 3	33 100
Підсумок по етапу 3	Керівник Програміст	37	2 37	5 100
4. Оформлення робочої документації				
4.1 Проведення розрахунків показників безпеки життєдіяльності	Керівник Програміст	2	0 2	0 100
4.2 Проведення економічних розрахунків	Керівник Програміст	4	0 4	0 100
4.3 Оформлення пояснювальної записки	Керівник Програміст	16	4 16	25 100

Продовження таблиці 3.4

Підсумок по етапу 4	Керівник Програміст	22	4 22	18 100
Підсумок по проекту	Керівник Програміст	103	14 103	13 100

На основі даних таблиці 3.4 розроблений календарний графік виконання робіт по проекту (табл. 3.5, рис. 3.16), що показує послідовність виконання робіт.

Таблиця 3.5 – Календарний графік виконання робіт

Зміст роботи	Виконавці	Тривалість, дні	Графік робіт	
			Початок	Кінець
1 Постановка задачі	Керівник	1	06.02.2021	06.02.2021
	Програміст	4	06.02.2021	09.02.2021
2 Збір вихідних даних	Керівник	2	10.02.2021	13.02.2021
	Програміст	10	10.02.2021	22.02.2021
3 Аналіз існуючих методів розв’язання задачі	Керівник	1	23.02.2021	23.24.2021
	Програміст	6	23.02.2021	02.03.2021
4 Обґрунтування необхідності розробки	Керівник	1	03.03.2021	03.03.2021
	Програміст	1	03.03.2021	03.03.2021
5 Визначення і аналіз вимог до програми	Керівник	2	06.03.2021	07.03.2021
	Програміст	6	06.03.2021	14.03.2021
6 Визначення структури вхідних і вихідних даних	Керівник	1	15.03.2021	15.03.2021
	Програміст	5	15.03.2021	21.03.2021
7 Вибір технічних і програмних засобів реалізації	Керівник	1	22.03.2021	22.03.2021
	Програміст	3	22.03.2021	24.03.2021
8 Погодження і затвердження технічного завдання	Керівник	1	27.03.2021	27.03.2021
	Програміст	2	27.03.2021	28.03.2021

Продовження таблиці 3.5

9 Проектування програмної архітектури	Керівник	1	29.03.2021	29.03.2021
	Програміст	4	29.03.2021	03.04.2021
10 Технічне проектування компонентів програми	Керівник	0	-	-
	Програміст	8	04.04.2021	13.04.2021
11 Програмування модулів в обраному середовищі програмування	Керівник	0	-	-
	Програміст	19	14.04.2021	12.05.2021
12 Тестування програмних модулів	Керівник	0	-	-
	Програміст	6	12.05.2021	19.05.2021
13 Збірка та випробування програми	Керівник	1	22.05.2021	22.05.2021
	Програміст	5	22.05.2021	26.05.2021
14 Аналіз результатів випробувань	Керівник	1	29.05.2021	29.05.2021
	Програміст	2	29.05.2021	30.05.2021
15 Проведення розрахунків показників безпеки життєдіяльності	Керівник	0	-	-
	Програміст	1	31.05.2021	31.05.2021
16 Проведення економічних розрахунків	Керівник	0	-	-
	Програміст	2	01.06.2021	02.06.2021
17 Оформлення пояснювальної записки	Керівник	2	02.06.2021	06.06.2021
	Програміст	5	02.06.2021	09.06.2021

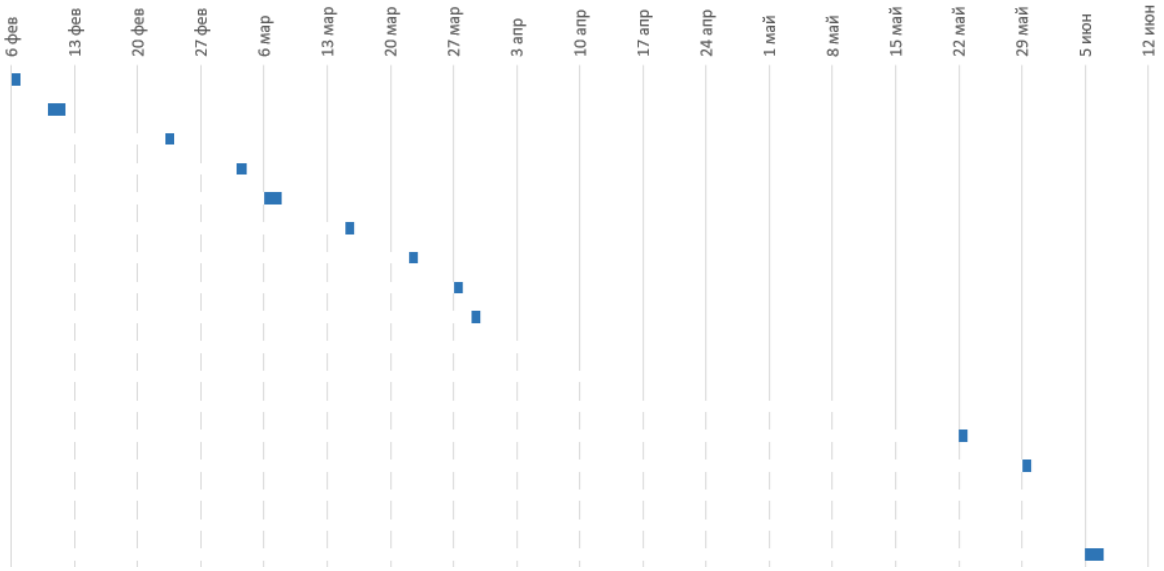

Рисунок 3.16 – Календарний графік виконання робіт керівника

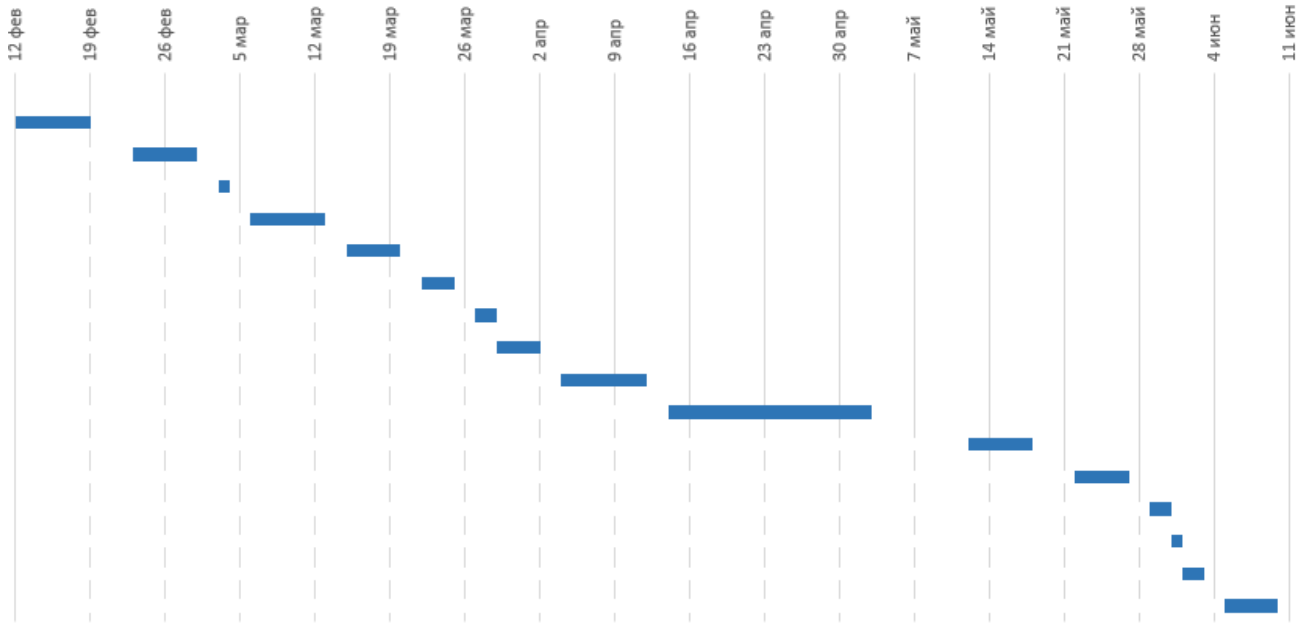

Рисунок 3.17 – Календарний графік виконання робіт програміста

Г Розрахунок затрат на розробку проекту

Капітальні вкладення в проекти, пов'язані з розробкою та впровадженням програмних продуктів, розраховуються за формулою

$$К = К_п + К_р, \quad (3.7)$$

де $К_п$ – капітальні вкладення на проектування (виробничі витрати), грн.;

$К_р$ – капітальні вкладення на реалізацію проекту, грн.

Виробничі витрати являють собою одноразові витрати на розробку забезпечувальних або функціональних систем, або їх елементів на всіх етапах проектування, а також витрати на їх удосконалення.

Сумарні витрати на проектування системи, її розробку і налагодження на комп'ютері визначаються за формулою 3.8:

$$К_п = \left((1 + W_d)(1 + W_c) + W_H\right) \sum_{i=1}^{m} З_{oi} + C_M + M_B, \quad (3.8)$$

де m – кількість робітників, які приймають участь у розробці проекту;

$З_{oi}$ – витрати на основну заробітну плату робітника I-II категорії, грн.;

W_d – коефіцієнт, що враховує додаткову заробітну плату в частках до основної заробітної плати ($W_d = 5\%$);

W_c – коефіцієнт, що враховує податок з фізичних осіб та військовий збір, в частках до суми основної та додаткової заробітної плати розробників ($W_c = 19{,}5\%$);

$W_Н$ – коефіцієнт, що враховує накладні витрати організації, в частках до основної заробітної плати розробників (приймається за фактичними даними,$W_Н = 0{,}6$);

$C_М$ – витрати на матеріали;

$M_В$ – витрати на використання машинного часу.

Витрати на основну заробітну плату працівника I-II категорії

$$З_{oi} = З_{дні} t_i, \quad (3.9)$$

де $З_{дні}$ – середньоденна заробітна плата працівника I-II категорії, грн/дн.;

t_i – кількість днів, відпрацьованих працівником I-II категорії.

Витрати часу на розробку системи по кожному виконавцю приймаються виходячи з його завантаження за календарним графіком виконання робіт (див. таблицю 3.6).

Розрахунок основної заробітної плати розробників проекту наведено з розрахунку, що в місяці в середньому 21 робочий день.

Таблиця 3.6 – Основна заробітна плата розробників проекту

Посада	Посадовий оклад, грн.	Середня денна ставка, грн.	Витрати часу на розробку, людино-днів	ОЗП, грн.
Керівник	10900	520	14	7280
Програміст	6300	300	103	30900
Усього				38180

З огляду на те, що проектована інформаційна система повинна бути запрограмована і налагоджена за допомогою комп'ютерів, до сумарних витрат на розробку додаються витрати на використання машинного часу, що обчислюються як:

$$M_B = t_{\text{МВ}} S_{\text{Мч}} K_{\text{М}}, \quad (3.10)$$

де $t_{\text{МВ}}$ - машинний час комп'ютера, необхідний для розробки програмного продукту; $t_{\text{МВ}}$ = 460 год. (з календарного графіку розробки);

$S_{\text{Мч}}$ – вартість 1 години машинного часу; $S_{\text{Мч}}$ = 6 грн/год;

$К_{\text{М}}$ – коефіцієнт мультипрограмності; $К_{\text{М}}$=1.

Таблиця 3.7 – Витрати на матеріали

Матеріали	Од. виміру	Необхідна кількість	Ціна за одиницю, грн.	Сума, грн.
Зошит загальний	шт.	1	29	29
Флеш пам’ять	шт.	1	150	150
Тонер для принтеру	шт.	1	150	150

Продовження таблиці 3.7

Папір офісний	пачка	1	95	95
Усього				424

Таким чином, капітальні вкладення на проектування дорівнюють:

$$К_п = (38180)\times((1+0,05)\times(1+0,195)+0,6)+424+460\times6\times1=73998,35 \text{ грн.}$$

Таблиця 3.8 – Витрати на розробку

Статті витрат	Сума, грн.
Основна заробітна плата	38180,00
Додаткова зарплата	1909,00
Відрахування на соціальні потреби	7445,10
Витрати на матеріали	424,00
Витрати на машинний час	2760,00
Накладні витрати організації	22908,00
Усього	73998,35

У зв'язку з тим, що для впровадження системи, що розглядається в даному проекті, не було витрат, пов'язаних з прокладанням лінії зв'язку, витрат на основне і допоміжне обладнання, витрат на реконструкцію і будівництво будівель, то дані витрати для впровадження системи не враховуються.

Також не приймаються в розрахунок витрати з підготовки та перепідготовки кадрів, витрати на створення інформаційної бази і витрати на придбання типових розробок.

Таким чином, при впровадженні системи, що розглядається в даному проекті, витрати на реалізацію визначаються витратами на обладнання і матеріали. У обладнання та матеріали входить комп'ютер. Вартість комп'ютера 17000 грн.

Тоді витрати на основне і допоміжне обладнання складають

$$K_0 = \sum_{j=1}^{n} C_{bj} Q_j Y_j, \quad (3.11)$$

де C_{bj} - балансова вартість і-го виду обладнання, грн. (при n=1 C_{bj}=17000 грн.);

Q_j – кількість одиниць j-го обладнання, шт. (1 шт.);

Y_j - коефіцієнт завантаження j-го виду обладнання при обробці інформації за рішенням завдань предметної області:

$$Y_j = \frac{T_j}{\Phi_{ефj}}, \quad (3.12)$$

де $\Phi_{ефj}$ - ефективний річний фонд часу роботи технічного засобу j-го виду, год./рік.

Час роботи технічного засобу j-го виду за рішенням s завдань, год./рік:

$$T_j = \sum_{k=1}^{s} t_{kj} \times U_k, \quad (3.13)$$

де t_{kj} - трудомісткість одноразової обробки інформації по k-й задачі на j-му виді технічних засобів, годин машинного часу (t_{kj}=6);

U_k - частота (періодичність) рішення k-й завдання, днів/рік ($U_k = 250$) на 2021 рік.

Витрати на реалізацію:

$$К_р = 17000 \times 1 \times 6 \times 250/(250 \times 8) = 12750 \text{ грн.}$$

Таким чином, сумарні витрати на розробку проекту складають:

$$К = К_п + К_р = 73998{,}35 + 12750 = 86748{,}35 \text{ грн}$$

Розрахуємо сумарні витрати, пов'язані з впровадженням аналога, з яким порівнюється розроблений програмний продукт. Вони складаються з наступних витрат:

– витрати на придбання програмного продукту (100000 грн.);

– витрати з оплати послуг на установку і супровід продукту (безкоштовно протягом перших 3-х років);

– витрати на основне і допоміжне обладнання (17000 грн.);

– витрати на підготовку користувача(підготовчий курс за рахунок компанії у якої покупається ПЗ).

Разом сумарні витрати, пов'язані з впровадженням аналога становлять 117000 грн.

Д. Розрахунок експлуатаційних витрат

До експлуатаційних витрат відносяться витрати, пов'язані із забезпеченням нормального функціонування проекту. Ці витрати називають також поточними витратами. Поточні витрати розраховуються за формулою

$$З_{тек} = З_{зп} + С_а + З_е + С_{рем} + З_м + З_н, \quad (3.14)$$

де $З_{зп}$ - витрати на зарплату основну та додаткову з відрахуваннями до соціальних фондів, грн.;

$С_а$ - амортизаційні відрахування від вартості обладнання і пристроїв системи, грн.;

$З_е$ - витрати на електроенергію, грн.;

$С_{рем}$ - витрати на поточний ремонт обладнання та пристроїв системи, грн.;

$З_м$ - витрати на матеріали і носії інформації, грн.;

$З_н$ - накладні витрати інформаційного відділу, грн.

Експлуатацію розробленої системи здійснюють фахівці. Витрати на їх заробітну плату основну і додаткову з відрахуваннями на соціальні потреби розраховують так:

$$С_{зп} = \sum_{i=1}^{m}(t_i З_i(1 + W_d)(1 + W_c)), \quad (3.15)$$

де t_i - час експлуатації системи і-м працівником, дні;

$З_i$ - середньоденна заробітна плата і-го працівника, грн./день.

Дані розрахунку заробітної плати фахівців наведені в таблицях 3.9 і 3.10.

Таблиця 3.9 - Дані по заробітній платі фахівців (для проекту)

Посада	Посадовий оклад, грн.	Середня денна ставка	Витрати часу на експлуатацію, людино-днів	Фонд з/п, грн
Заступник директора	7200	342,62	40	13704
Програміст	7500	357,14	30	13443
Усього				27147

$С_{зп1}$ = (40×247.62+30×357,14) ×1,05×1,195 = 34602,70 грн. (за рік).

Таблиця 3.10 - Дані по заробітній платі фахівців (аналог)

Посада	Посадовий оклад, грн.	Середня денна ставка	Витрати часу на експлуатацію	Фонд з/п, грн
Заступник директора	7200	342,62	60	20557
Програміст (сертифікований)	16300	776,19	40	31047
Усього				51604

$С_{зп2}$ = (60×247,62+40×538,1)×1,05×1,195 = 64750,62 грн. (за рік).

Сума амортизаційних відрахувань розраховується наступним чином:

$$\mathrm{C_a} = \sum_{j=1}^{n} \frac{\mathrm{C}_{bj} a_j g_j t_j}{F_{\text{еф}j}}, \quad (3.15)$$

де $\mathrm{C_{bj}}$ - балансова вартість і-го виду обладнання, грн.;

$\mathrm{t_j}$ - час роботи і-го виду обладнання, годин;

$\mathrm{F_{еф j}}$ - ефективний фонд часу роботи обладнання в рік, годин;

$\mathrm{a_j}$ - норма річних амортизаційних відрахувань для і-го виду обладнання;

$\mathrm{g_j}$ - кількість одиниць обладнання j-го виду.

Ефективний фонд часу роботи обладнання можна обчислити за формулою:

$$F_{\text{эф}} = D_p \times H_{\text{е}}, \quad (3.16)$$

де D_p - кількість робочих днів в році, D_p = 250;

$H_{\text{е}}$ - норматив середньодобової навантаження, год./ день, $H_{\text{Э}}$ = 8.

Таким чином, ефективний фонд часу роботи обладнання складає $F_{\text{эф}}$=250×8=2000 годин.

Дані для розрахунку:

a_j = 0,25 (використовується прискорена амортизація - 20-30%); g_j=1;

t_j(для проекту) = (45+30)*8=600 год.;

t_j(для аналогу) = (60+40)*8=800 год.;

$\mathrm{C}_{b1} = 17000$ грн.; $\mathrm{C}_{b2} = 17000$ грн.;

Сума амортизаційних відрахувань для проекту складає:

$$\mathrm{C}_{a1}=(17000\times0{,}25\times1\times600)/2000 = 1275 \text{ грн.}$$

Сума амортизаційних відрахувань для аналогу складає:

$$\mathrm{C}_{a2}=(17000\times0{,}25\times1\times800)/2000 = 1700 \text{ грн.}$$

Витрати на силову енергію розраховуються за формулою:

$$З_э = \sum_{j=1}^{n} N_j t_j g_j T_e, \quad (3.17)$$

де N_j - встановлена потужність j-го виду технічних засобів, кВт;

t_j - час роботи і-го виду технічних засобів, годин;

g_j - коефіцієнт використання встановленої потужності обладнання;

T_e - тариф на електроенергію, грн/кВт год.

В даний час тариф АК "Сумиобленерго" на електроенергію складає 1,68 грн./кВт год, встановлена потужність для комп'ютера дорівнює 0,27 кВт, таким чином витрати на силову енергію для проекту складуть:

$$З_э=0,27\times1\times600\times1,68=272,16 \text{ грн.},$$

Для аналогу:

$$З_э=0,27\times1\times800\times1,68=362,88 \text{ грн.}$$

Витрати на поточний ремонт обладнання розраховуються за формулою

$$З_{рем} = \sum_{j=1}^{n} \frac{C_{pi} C_{bi} T_{pi}}{F_{ефj}}, \quad (3.18)$$

де C_{pi} – норматив витрат на ремонт ($C_{pi} = 0,05$).

Витрати на поточний ремонт обладнання для проекту складають:

$$З_{рем1}=(0,05\times17000\times600)/2000 = 255 \text{ грн.},$$

для аналогу:

$$З_{рем2}=(0,05\times17000\times800)/2000 = 340 \text{ грн.}$$

Витрати на матеріали, які споживаються протягом року, складають 1% від балансової вартості основного устаткування і рівні 170 грн. для проекту і аналога.

Накладні витрати включають витрати на утримання адміністративного та управлінського персоналу, на утримання приміщення і т.д. Норматив накладних витрат становить 20% від прямих витрат, що включають перші п'ять статей витрат, представлених в таблиці 3.11.

Накладні витрати для проекту:

$$З_{н1} = (34602,70 +1275 +272,16+255+170)\times0,2 = 7314,97 \text{ грн.}$$

Накладні витрати для аналогу:

$$З_{н2} = (64750,62 +1700 +362,88+340+170)\times0,2 = 13464,70 \text{ грн.}$$

Таблиця 3.11 - Річні експлуатаційні витрати

Статті витрат	Витрати на проект, грн.	Витрати на аналог, грн.
Основна і додаткова зарплата з відрахуваннями	34602,70	64750,62
Амортизаційні відрахування	1275	1700
Витрати на електроенергію	272,16	362,88
Витрати на поточний ремонт	255	338,65
Витрати на матеріали	170	340
Накладні витрати	7314,97	13464,70

Продовження таблиці 3.11

Разом	43889,83	80788,20

Е. Оцінка ефективності розробленого проекту

Оцінка економічної ефективності варіантів проектних рішень елементів АІС ґрунтується на розрахунку показників порівняльної економічної ефективності капітальних вкладень. Річний економічний ефект від використання розроблюваної системи визначається по різниці приведених витрат на базовий і новий варіанти в розрахунку на річний обсяг виконуваних робіт:

$$\mathrm{E} = (З_1 \times \mathrm{A_к} - З_2) \times N, \quad (3.19)$$

де $З_1, З_2$ - наведені витрати на одиницю робіт, що виконуються за допомогою базового і проектованого варіантів процесу обробки інформації, грн.;

$\mathrm{A_к}$ - коефіцієнт експлуатаційної технічної еквівалентності, чи технічного рівня, $\mathrm{A_к} = 1{,}27$ (формула (3.6));

N - обсяг робіт, виконуваних за допомогою розробленого продукту (приймемо рівним 1).

Наведені витрати на одиницю робіт, виконуваних за базовим і розробляється варіантів, розраховуються за формулою:

$$З_i = C_j + E_H \times K_j, \quad (3.20)$$

де C_j - собівартість (поточні експлуатаційні витрати одиниці робіт), грн.;

E_H - нормативний коефіцієнт економічної ефективності (E_H=0,33);

K_j - сумарні витрати, пов'язані з впровадженням нового проекту.

Витрати на одиницю робіт по проекту:

$З_1$= 43889,83+0,33×86748,35= 72516,78 грн.

Витрати на одиницю робіт по аналогу:

$$З_2 = 80788,20+0,33\times117000 = 119398,20 \text{ грн.}$$

Економічний ефект від використання розроблюваної системи:

$$Е = 119398,20 \times1,27 – 72516,78 = 79118,93 \text{ грн.}$$

Зведені дані по розрахунку економічного ефекту наведені в таблиці 3.12.

Після визначення річного економічного ефекту необхідно розрахувати термін окупності витрат на розробку продукту за формулою:

$$Т_{ок} = К/Е, \quad (3.21)$$

Термін окупності складе:

$$Т_{ок} = 119398,20 / 79118,93 = 1,5 \text{ року.(18 місяців)}$$

Потім розрахуємо фактичний коефіцієнт економічної ефективності розробки (Еф) і порівняти його з нормативним значенням коефіцієнта ефективності капітальних вкладень Ен = 0,33 [43].

$$Еф = 1/Т_{ок}.= 1/1,5 = 0,5. \quad (3.22)$$

Таблиця 3.12 - Економічний ефект

Характеристика	Значення	
	Продукт-аналог (базовий)	Розроблюваний програмний продукт
Собівартість (поточні експлуатаційні витрати), грн.	80788,20	43889,83

Продовження таблиці 3.12

Сумарні витрати, пов'язані з впровадженням проекту, грн.	117000	86748,35
Наведені витрати на одиницю робіт, грн.	119398,20	72516,78
Економічний ефект від використання розроблюваної системи (програмного продукту), грн.	79118,93	

Фактичний коефіцієнт економічної ефективності розробки вийшов більше, ніж нормативний, тому розробка та впровадження розроблюваного продукту є ефективною.

Таким чином, в ході проробленої роботи знайдені всі необхідні дані, що доводять доцільність і ефективність розроблюваної системи.

ВИСНОВКИ

У ході виконання даної дипломної роботи було розглянуто проблеми автоматизації діяльності співробітників середніх шкіл з використанням сучасних засобів обробки інформації. В роботі зроблена якісна постановка задачі, розглянуті методи прогнозування. З використання обраного метода Хольта, розроблено алгоритмічне забезпечення для поставленої задачі. На підставі бізнес правил були розроблені функціональні та нефункціональні вимоги до програмного забезпечення, діаграми варіантів використання ПЗ, модель даних. Детально описані можливості програмного продукту, обрано цільовий варіант архітектури, розроблена архітектура системи, було описано обрані технічні засоби. Розроблено програмне забезпечення для задачі автоматизації діяльності співробітників середніх шкіл – програмні рішення для створення задач та документації адміністрацією навчального закладу. Розроблені тест-кейси для перевірки системи. Реалізовано математичний алгоритм прогнозування часу виконання задачі та побудова графіку на підставі існуючих даних, виконана перевірка результатів прогнозування за допомогою ручного перерахунку.

Розроблене програмне забезпечення може бути використано у діяльності співробітників середніх шкіл, що дозволить зберігати та швидко обробляти великі об’єми інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Планування на підприємстві: Підручник / М. В. Васильченко. - М.:Ексмо, 2017. - 52 с.

2 Педагогічна майстерня // http://xn--i1abbnckbmcl9fb.xn--p1ai/статті/210809/, 18.02.2018.

3 Щербаков В. А. Комплексный экономический анализ хозяйственной деятельности предприятия в рыночной экономике. – Новосибирск : НГАВТ, 2012.

4 Міністерство освіти науки україни. Наказ № 665 від 01 червня 2013 року «Про затвердження кваліфікаційних характеристик професій (посад) педагогічних та науково-педагогічних працівників навчальних закладів».

5 Навчальні матеріли онлайн. Посадові обов'язки заступника директора з виховної роботи середнього навчального закладу // http://pidruchniki.com/pedagogika/funktsiyi_zastupnika_directora_zagalnoosvitnogo_navchalno-vihovnogo_zakladu, 18.04.2018.

6 Посадові обов'язки завідуючого кабінетом // http://www.berezianka-school.edukit.zt.ua/normativno-pravova_baza_shkoli/posadovi_obovyazki_zaviduyucho-go_kabinetom, дата звернення 18.04.2018.

7 Електронний журнал // http://ukrschools.com.ua, 11.03.2018.

8 Щоденник // http://shodennik.ua, 11.03.2018.

9 1С:Підприємство // https://ru.wikipedia.org/wiki/1С:Підприємство, 11.03.2018.

10 Trello study // https://trello.com/b/VYZxnf5s/study-app

11 Лукашин Ю.П., Адаптивные методы краткосрочного прогнозирования временных рядов: Учебное пособие. - М.: Финансы и статистика, 2013. - 416 с.

12 Репін В.В., Елиферов В.Г. Процессный подход к управлению. Моделирование бизнес-процессов. – М. РИА «Стандарты и качество», 2014 – 408 с.

13 Гради Буч, Роберт А. Максимчук, Майкл У. Энгл, Бобби Дж. Янг, Джим Коналлен, Келли А. Хьюстон. Объектно-ориентированный анализ и проектирование с примерами приложений. — 3-е издание. — «Вильямс», 2010.

14 Кобёрн А. Современные методы описания функциональных требований к системам. — М.: Лори, 2012.

15 Румянцев М. Средства имитационного моделирования бизнес-процессов // Корпоративные системы. – 2017. - № 2

16 Буч Г., Рамбо Д., Якобсон І., Язык UML. Руководство пользователя. 2-е издание – М.: ДМК Пресс. – 496 с.

17 Маклаков С.В. BPwin и ERwin: CASE – средства разработки информационных систем. – М.: Диалог Мифи, 2010.

18 Фаулер М. Архитектура корпоративных программных приложений.: Пер. с англ. — М.: Издательский дом «Вильямс», 2016. — 544 с.

19 Trac. Головна сторінка // https://trac.edgewall.org, 13.05.2018.

20 Марк Саммерфилд. Python на практике.— М.: ДМК Пресс, 2014. — 338 с.

21 Флэнаган Д. 13.8.1. Чего не может JavaScript // JavaScript. Подробное руководство = JavaScript. The Definite Guide / Перевод А. Киселева. — 5-е изд. — СПб.: «Символ-Плюс», 2012. — С. 280, 281.

22 Питер Лабберс, Брайан Олберс, Фрэнк Салим. HTML5 для профессионалов: мощные инструменты для разработки современных веб-приложений. — М.: «Вильямс», 2011. — 272 с..

23 Дженнифер Н. Роббинс. HTML5, CSS3 и JavaScript. Исчерпывающее руководство. — М.: Эксмо, 2014. — 528 с.

24 Васвани. В. MySQL: использование и администрирование. — М.: «Питер», 2011. — 368 с.

25 Калбертсон Роберт, Браун Крис, Кобб Гэри. Быстрое тестирование. — М.: «Вильямс», 2012. — 374 с.

26 Python Developer’s Guide // https://devguide.python.org/

27 Learn Python Programming // https://www.programiz.com/python-programming

28 Python IDEs and Code Editors // https://www.programiz.com/python-programming/ide

29 The Modern JavaScript Tutorial // https://javascript.info/

30 Javascript Tutorial // https://www.tutorialspoint.com/javascript/index.htm

31 JavaScript // https://developer.mozilla.org/en-US/docs/Web/JavaScript

32 UML Tutorial // https://www.tutorialspoint.com/uml/index.htm

33 UML Class Diagram Tutorial // https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/

34 MySQL Tutorial // https://www.mysqltutorial.org/

35 MySQL 8.0 Reference Manual // https://dev.mysql.com/doc/refman/8.0/en/tutorial.html

36 HTML5 Tags/Elements // https://www.tutorialrepublic.com/html-reference/html5-tags.php

37 Trac Open Source Project // https://trac.edgewall.org/

38 Black Box Testing Vs. White Box Testing: Key Differences // https://www.guru99.com/back-box-vs-white-box-testing.html

39 Differences between Black Box Testing vs White Box Testing // https://www.geeksforgeeks.org/differences-between-black-box-testing-vs-white-box-testing/

40 Foundation Level Syllabus // https://www.istqb.org/downloads/syllabi/foundation-level-syllabus.html

41 What is a Test Case? // https://www.guru99.com/test-case.html

42 How to Write Test Cases: The Ultimate Guide with Examples // https://www.softwaretestinghelp.com/how-to-write-effective-test-cases-test-cases-procedures-and-definitions/

43 Шматко А.В., Москаленко В.В. «Технико-экономическое обоснование выполнения дипломного проекта». – М.: НТУ ХПИ, кафедра ПИиИТУ, 2016.

ДОДАТКИ

ДОДАТОК А

(обов’язковий)

SUMMARY

Sytnyk V.M. Automation of activities of secondary school employees. — Masters-level Qualification Thesis. Sumy State University, Sumy, 2021.

The purpose of the qualification work is to develop web-oriented information to automate the activities of secondary school employees. The state of automation of business processes is analyzed, requirements to the created system are developed, the web-oriented information system for automation of activity of employees of high schools is designed, implemented and tested.

Keywords: web-oriented system, Python, automation, secondary school, web browser, MySQL.

АНОТАЦІЯ

Ситник В. М. Автоматизація діяльності співробітників середніх шкіл. – Кваліфікаційна магістерська робота. Сумський державний університет, Суми, 2021 р.

Мета кваліфікаційної роботи – розробка веб-орієнтованої інформаційної для автоматизації діяльності співробітників середніх шкіл. Проаналізовано стан автоматизації бізнес-процесів, розроблені вимог до створюваної системи, спроектовано, реалізовано та протестовано веб-орієнтовану інформаційну систему для автоматизації діяльності співробітників середніх шкіл.

Ключові слова: веб-орієнтована система, Python, автоматизація, середня школа, веб-браузер, MySQL.

ДОДАТОК Б

(інформаційний)

КОНЦЕПТУАЛЬНА МОДЕЛЬ ТА ДЕКОМПОЗИЦІЯ ПЕРШОГО РІВНЯ

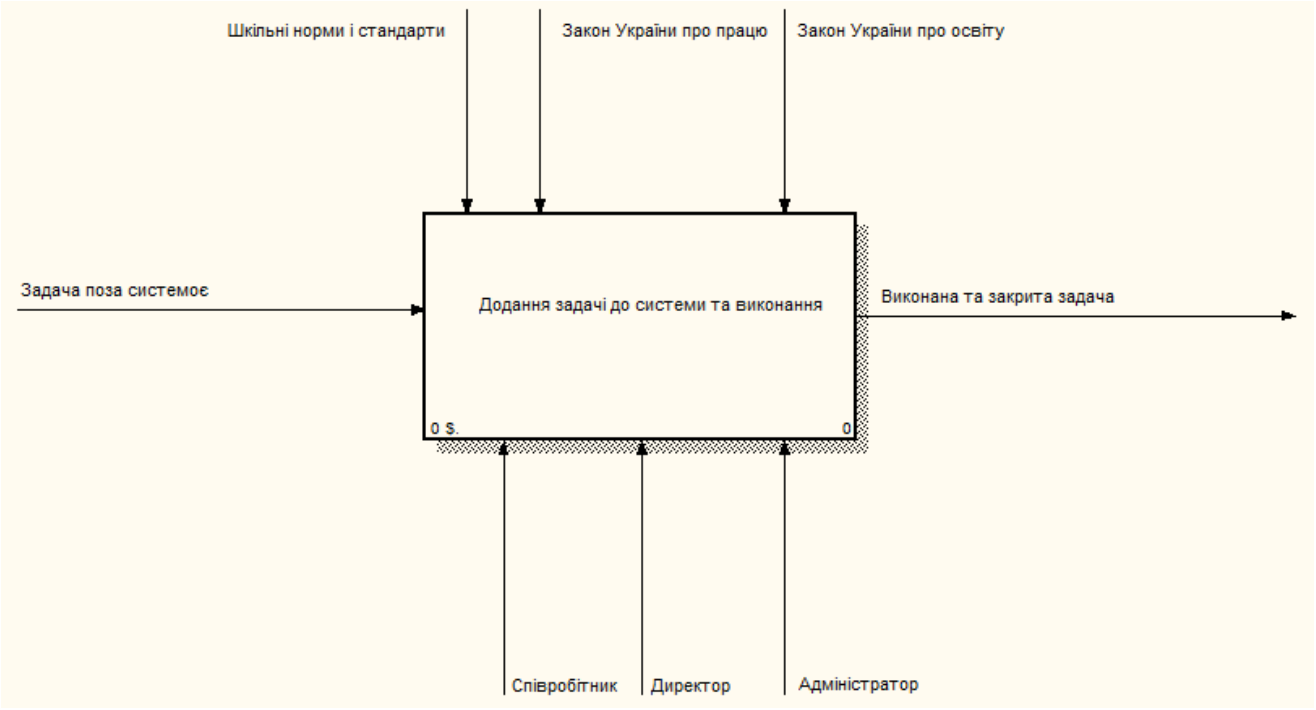

Рисунок Б.1 – Концептуальна модель

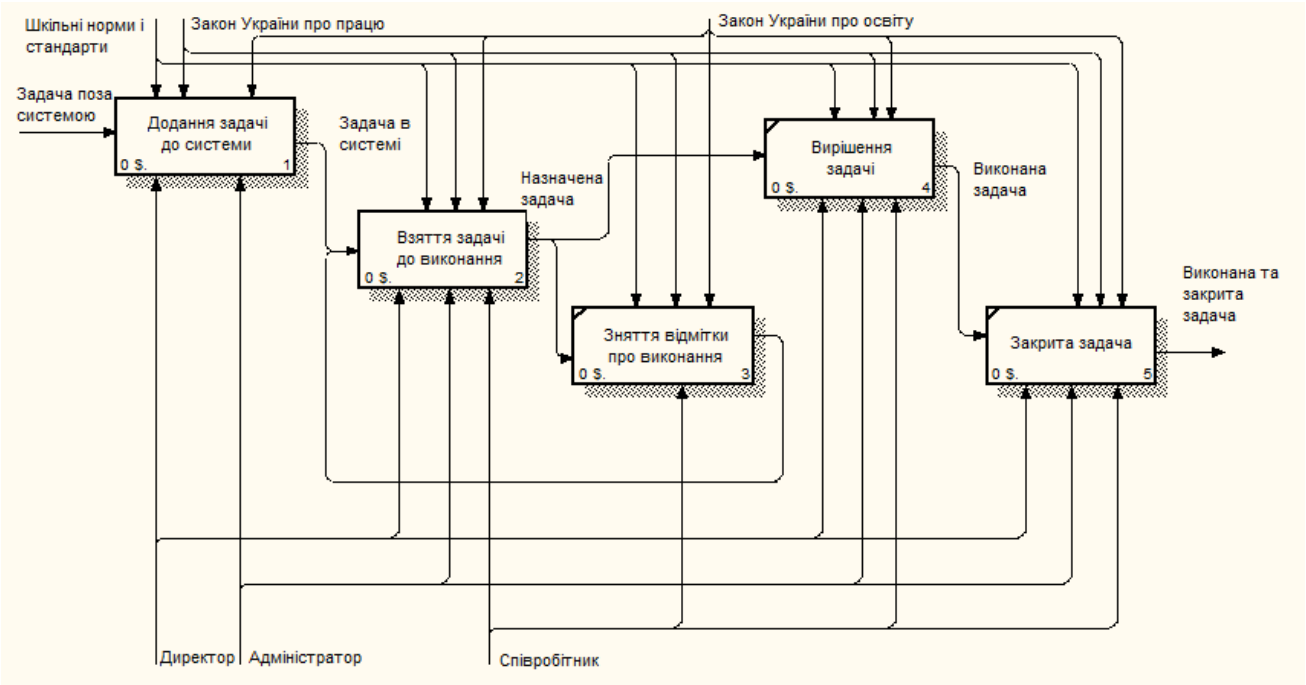

Рисунок Б.2 – Декомпозиція першого рівня

ДОДАТОК В

(інформаційний)

ДЕКОМПОЗИЦІЯ ДРУГОГО РІВНЯ (ДОДАННЯ ЗАДАЧІ В СИСТЕМУ ТА ВЗЯТТЯ ЗАДАЧІ ДО ВИКОНАННЯ).

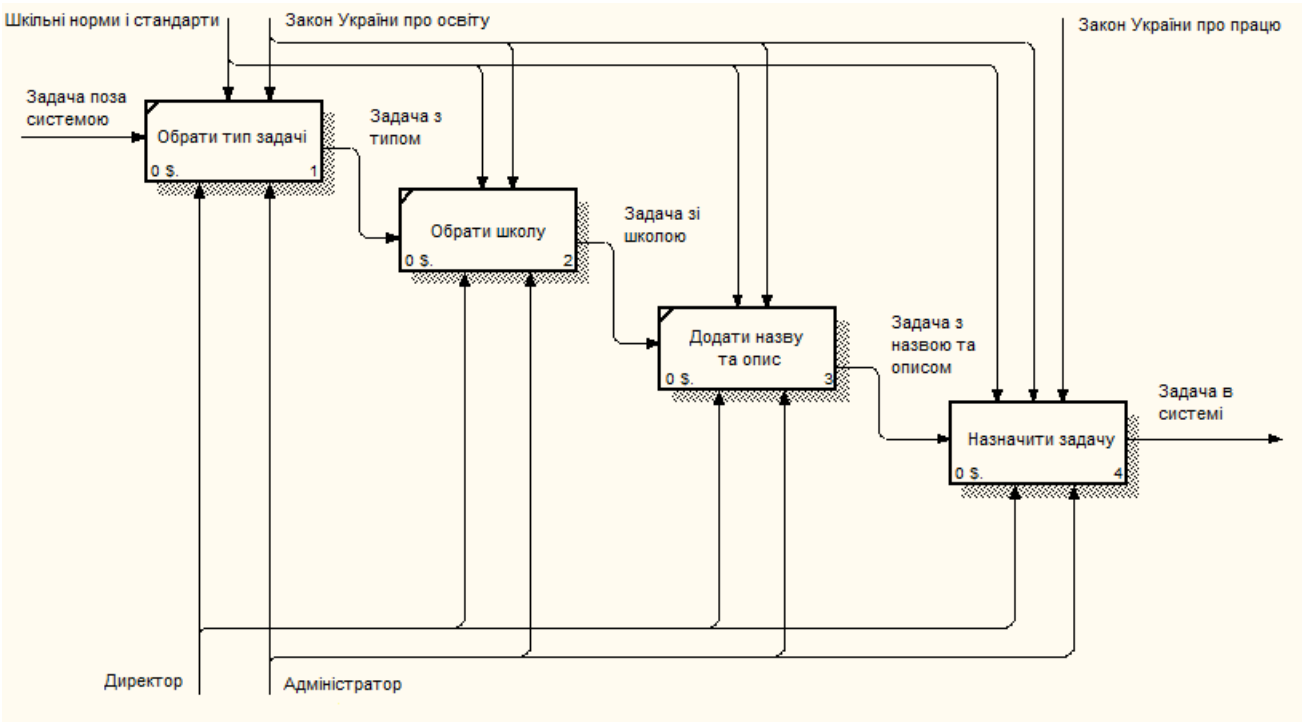

Рисунок В.1 – Додання задачі в систему

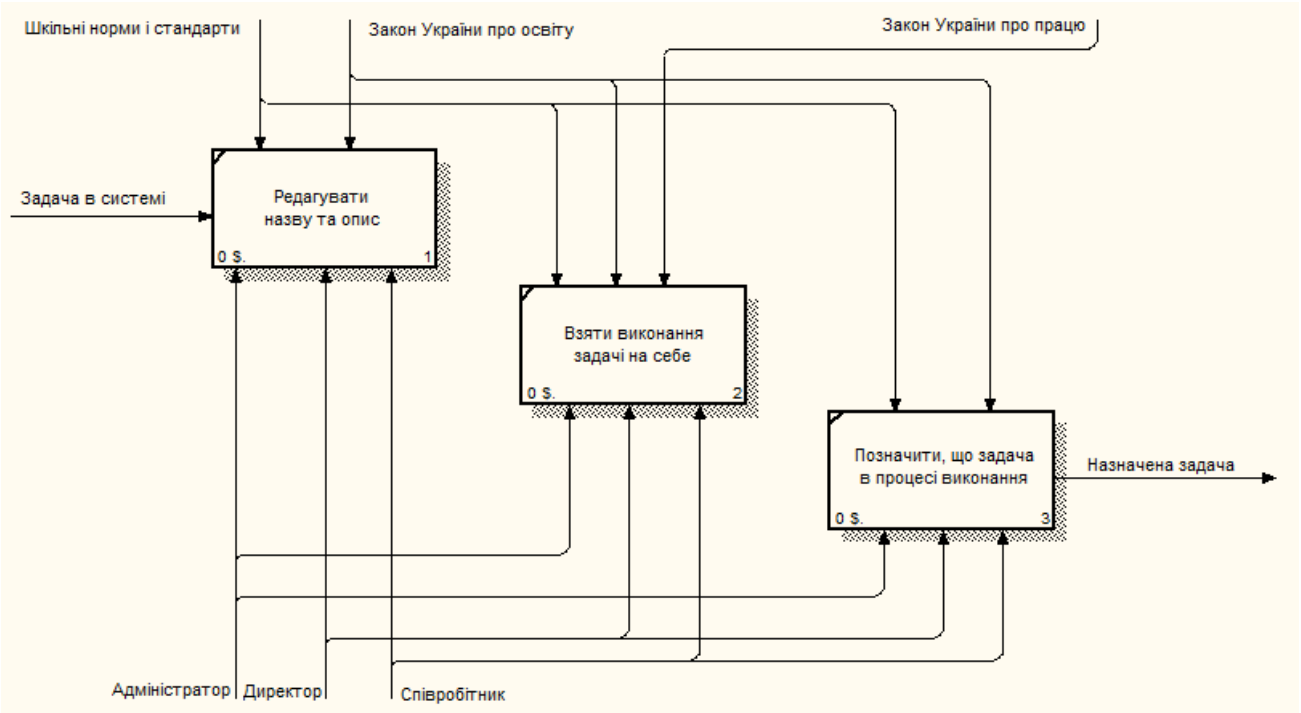

Рисунок В.2 – Взяття задачі до виконання

ДОДАТОК Г

(інформаційний)

ДЕКОМПОЗИЦІЯ ТРЕТЬОГО РІВНЯ (ПРИЗНАЧЕННЯ ЗАДАЧІ).

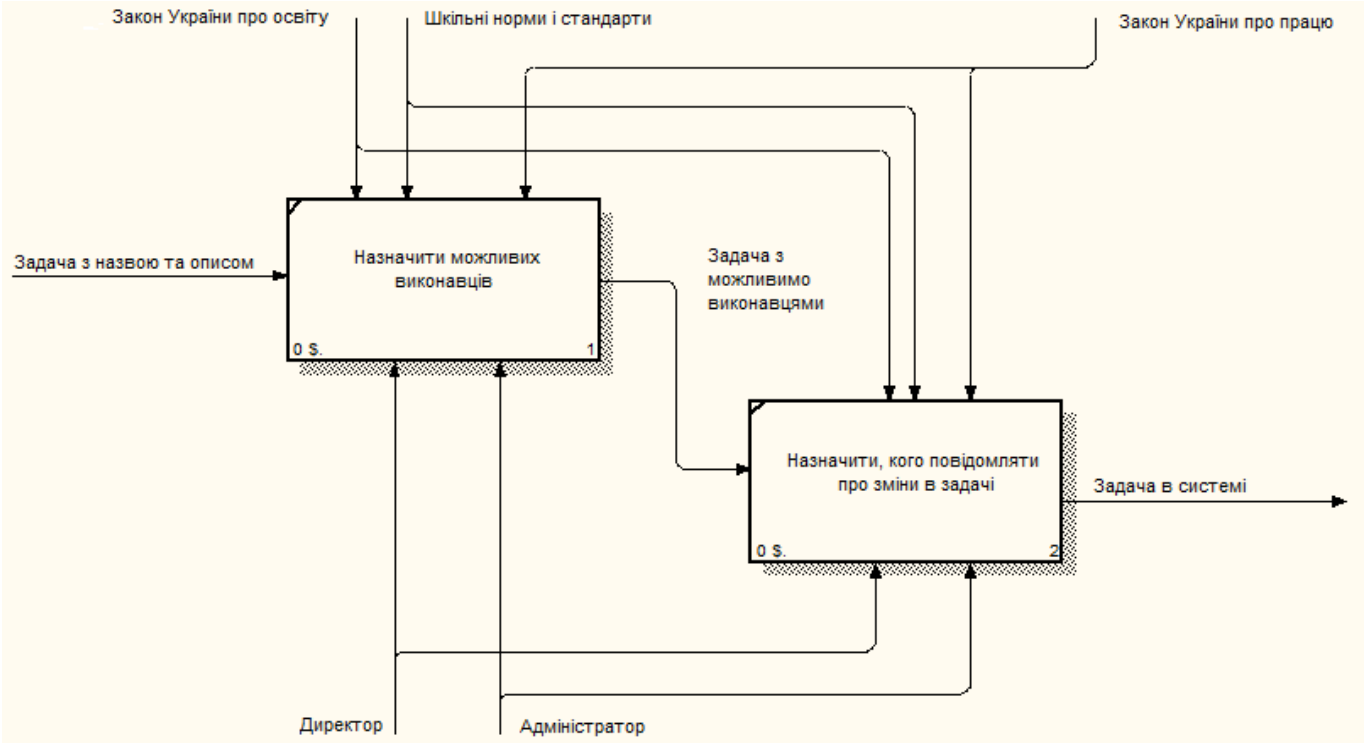

Рисунок Г – Призначення задачі

ДОДАТОК Д

(інформаційний)

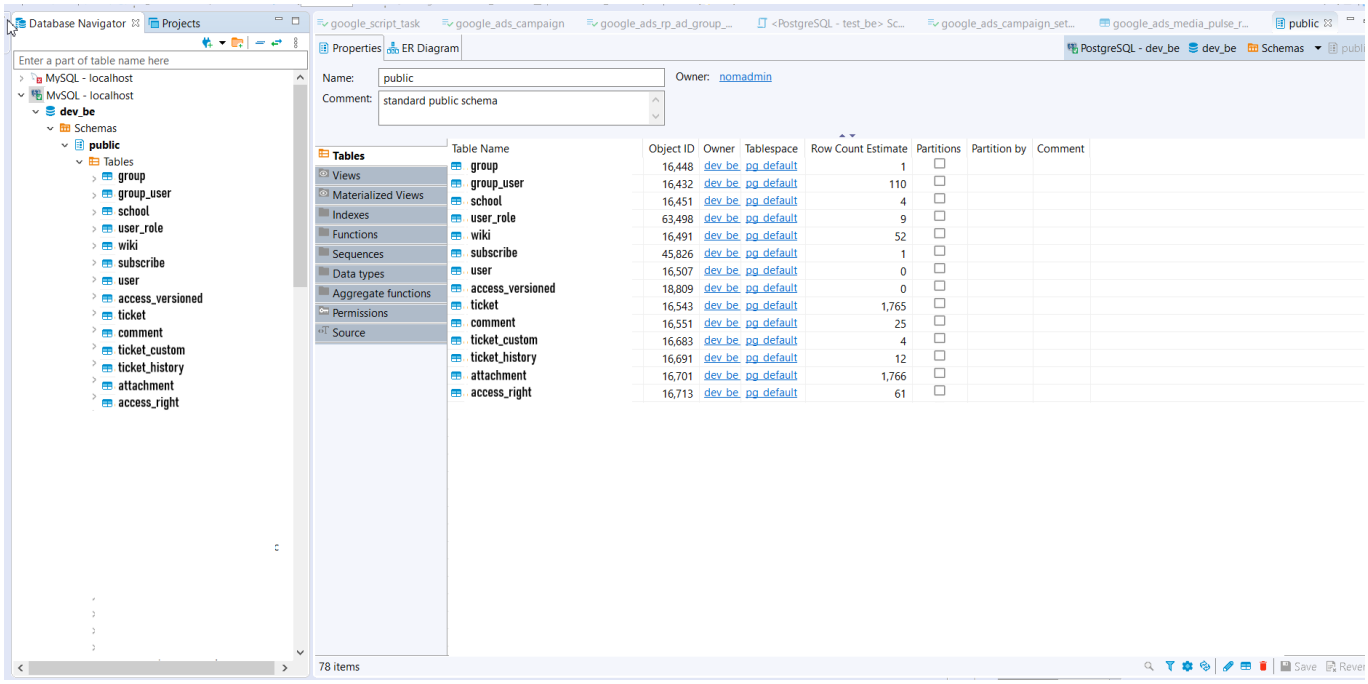

Рисунок Д – Структура БД

ДОДАТОК Е

(інформаційний)

ЛИСТИНГ

Python:

```
From __future__ import unicode_literals


from mysql.db import models


class AuthGroup(models.Model):
    name = models.CharField(unique=True, max_length=80)


    class Meta:
        managed = False
        db_table = 'auth_group'


class AuthGroupPermissions(models.Model):
    group = models.ForeignKey(AuthGroup)
    permission = models.ForeignKey('AuthPermission')


    class Meta:
        managed = False
        db_table = 'auth_group_permissions'
        unique_together = (('group_id', 'permission_id'),)



class AuthPermission(models.Model):
    name = models.CharField(max_length=255)
    content_type = models.ForeignKey('DjangoContentType')
    codename = models.CharField(max_length=100)
```

Продовження листингу:

```
class Meta:
        managed = False
        db_table = 'auth_permission'
        unique_together = (('content_type_id', 'codename'),)

class AuthUser(models.Model):
    password = models.CharField(max_length=128)
    last_login = models.DateTimeField(blank=True, null=True)
    is_superuser = models.IntegerField()
    username = models.CharField(unique=True, max_length=30)
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
    email = models.CharField(max_length=254)
    is_staff = models.IntegerField()
    is_active = models.IntegerField()
    date_joined = models.DateTimeField()

    class Meta:
        managed = False
        db_table = 'auth_user'


class AuthUserGroups(models.Model):
    user = models.ForeignKey(AuthUser)
    group = models.ForeignKey(AuthGroup)

    class Meta:
        managed = False
```

Продовження листингу:

```
        db_table = 'auth_user_groups'
        unique_together = (('user_id', 'group_id'),)


class AuthUserUserPermissions(models.Model):
    user = models.ForeignKey(AuthUser)
    permission = models.ForeignKey(AuthPermission)

    class Meta:
        managed = False
        db_table = 'auth_user_user_permissions'
        unique_together = (('user_id', 'permission_id'),)


class DjangoAdminLog(models.Model):
    action_time = models.DateTimeField()
    object_id = models.TextField(blank=True, null=True)
    object_repr = models.CharField(max_length=200)
    action_flag = models.SmallIntegerField()
    change_message = models.TextField()
    content_type = models.ForeignKey('DjangoContentType', blank=True, null=True)
    user = models.ForeignKey(AuthUser)

    class Meta:
        managed = False
        db_table = 'django_admin_log'
```

Продовження листингу:

```
class DjangoContentType(models.Model):
    app_label = models.CharField(max_length=100)
    model = models.CharField(max_length=100)

    class Meta:
        managed = False
        db_table = 'django_content_type'
        unique_together = (('app_label', 'model'),)


class DjangoMigrations(models.Model):
    app = models.CharField(max_length=255)
    name = models.CharField(max_length=255)
    applied = models.DateTimeField()

    class Meta:
        managed = False
        db_table = 'django_migrations'


class DjangoSession(models.Model):
    session_key = models.CharField(primary_key=True, max_length=40)
    session_data = models.TextField()
    expire_date = models.DateTimeField()

    class Meta:
        managed = False
        db_table = 'django_session'
```

Продовження листингу:

```
class (models.Model):
    id_title = models.SmallIntegerField(primary_key=True)
    name = models.CharField(max_length=200)
    ticket_id = models.ForeignKey(db_column='ticket_id')

    class Meta:
        managed = False
        db_table = 'title'


class (models.Model):
    id_description = models.SmallIntegerField(primary_key=True)
    school_id_name = models.CharField(max_length=40, blank=True, null=True)

    class Meta:
        managed = False
        db_table = 'description'


class (models.Model):
    ticket_history_id = models.SmallIntegerField(primary_key=True)
    school_id_name = models.CharField(max_length=30, blank=True, null=True)
    short_school_id_name = models.CharField(max_length=10, blank=True, null=True)

    class Meta:
        managed = False
        db_table = 'author'
```

Продовження листингу:

```
class (models.Model):
    group_user_id = models.IntegerField(primary_key=True)
    group_id = models.ForeignKey(db_column='group_id')
    group_name_id = models.ForeignKey(db_column='group_name_id', blank=True,
null=True)

    class Meta:
        managed = False
        db_table = 'group_name'



class (models.Model):
comment_id = models.SmallIntegerField(primary_key=True)
ticket_id = models.CharField(max_length=5)
class Meta:
managed = False
db_table = 'comment'
```

HTML:

```
<button id="btn_modal_window">Open Modal</button>
  <div id="my_modal" class="modal">
    <div class="modal_content">
      <span class="close_modal_window">×</span>
      <p>Modal window        !</p>
    </div>
</div>
```

CSS:

```
.modal {
    display: none;
    position: fixed;
    z-index: 1;
    left: 0;
    top: 0;
    width: 100%;
    height: 100%;
    overflow: auto;
    background-color: rgba(0,0,0,0.6);
    z-index: 1000;
}
.modal .modal_content {
    background-color: #fefefe;
    margin: 15% auto;
    padding: 20px;
    border: 1px solid #888;
    width: 80%;
    z-index: 99999;
}
.modal .modal_content .close_modal_window {
    color: #aaa;
    float: right;
    font-size: 28px;
    font-weight: bold;
    cursor: pointer;
}
```

JavaScript:

```
var modal = document.getElementById("my_modal");
 var btn = document.getElementById("btn_modal_window");
 var span = document.getElementsByClassName("close_modal_window")[0];

 btn.onclick = function () {
    modal.style.display = "block";
 }

 span.onclick = function () {
    modal.style.display = "none";
 }

 window.onclick = function (event) {
    if (event.target == modal) {
        modal.style.display = "none";
    }
}
```