НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра інженерії програмного забезпечення в енергетиці

«На правах рукопису»
УДК_____

«До захисту допущено»
зав. кафедри ІПЗЕ
_____Олександр КОВАЛЬ

«__»_____2024р.

Магістерська дисертація

За освітньою програмою «Інженерія програмного забезпечення інтелектуальни кібер-фізичних систем в енергетиці»

Спеціальності 121 Інженерія програмного забезпечення

на тему: «Система створення етапів дипломного проектування»

Виконав студент 2 курсу, групи ТВ-32мп

Омельченко Гліб Олександрович _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник

д. ф., ст. викл. Бандурка О. І. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент

професор, д. т. н., професор Отрох С. І. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2024

Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»

Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці
Рівень вищої освіти другий, магістерський
За освітньою програмою «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці»
Спеціальності 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
зав. кафедри
_____Олександр КОВАЛЬ
«__»_____ 202_р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ

Омельченку Глібу Олександровичу
(прізвище, ім'я, по батькові)

1. Тема дисертації: «Система створення етапів дипломного проектування»

Науковий керівник д. ф. ст. викл. Бандурка Олена Іванівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «_____»_____202_ року №____

2. Строк подання студентом дисертації «__»_____202_ р.

3. Вихідні дані до роботи персональний комп'ютер під керування операційної системи Microsoft Windows, мова програмування JavaScript

4. Перелік питань, які потрібно розробити систему створення етапів дипломного проектування, базу даних яка буде зберігати в собі основні дані

5. Орієнтований перелік ілюстративного матеріалу архітектура системи, графічне представлення інтерфейсу, приклади роботи програмного модулю

6. Орієнтований перелік публікацій Омельченко Г.О, Бандурка О.І, ДацюкО.А Система створення етапів дипломного проектування: III Всеукраїнська науково-практична конференці «Системні науки та інформатика»

7. Дата видачі завдання «__» _____ 202_ р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів виконання магістерської дисертації	Строки виконання етапів магістерської дисертації	Примітки
1	Отримання завдання	01.10.2023	Виконано
2	Дослідження предметної області	01.10.2023 - 01.11.2023	Виконано
3	Постановка вимог до проєктування системи	01.11.2023 - 15.11.2023	Виконано
4	Дослідження існуючих рішень	15.11.2023 - 01.12.2023	Виконано
5	Підготовка доповідей публікацій	01.02.2024 – 01.03.2024	Виконано
6	Розробка програмного продукту	01.03.2024 - 01.09.2024	Виконано
7	Тестування	01.09.2024 - 13.10.2024	Виконано
8	Захист програмного продукту	14.10.2024 - 18.10.2024	Виконано
9	Підготовка магістерської дисертації	19.10.2024 - 24.11.2024	Виконано
10	Передзахист магістерської дисертації	25.11.2024 - 29.11.2024	Виконано
11	Захист магістерської дисертації	16.12.2024 - 22.12.2024	Виконано

Студент _____ Гліб ОМЕЛЬЧЕНКО
(підпис) (ім’я, прізвище)

Науковий керівник _____ Олена БАНДУРКА
(підпис) (ім’я, прізвище)

РЕФЕРАТ

Структура і обсяг кваліфікаційної роботи. Магістерська дисертація складається зі вступу, п’яти розділів, висновків та 1 додатку. Робота містить посилання на 33 джерела, 67 рисунки та 4 таблиці. Основна частина роботи викладена на 95 сторінках.

Актуальність. Процес дипломного проектування відіграє ключову роль у підготовці студентів до майбутньої професійної діяльності, адже він дозволяє їм застосувати набуті знання на практиці. Зростає необхідність у створенні ефективної системи, яка б дозволяла організувати етапи проектування з урахуванням індивідуальних потреб студентів і вимог навчального закладу. Така система має не лише оптимізувати контроль за виконанням завдань, а й сприяти обранню студентами теми під конкретний напрямок, дозволяти вибір наукового керівника та забезпечувати гнучкий процес взаємодії. Актуальність теми полягає в необхідності підвищення якості підготовки студентів до дипломного проектування та в створенні умов для їхньої самостійної роботи в цифровому середовищі.

Метою роботи є розробка системи, яка забезпечуватиме оптимізацію етапів дипломного проектування, дозволятиме студентам обирати тему дипломної роботи під певний напрям, а також вибирати собі наукового керівника або пропонувати власну тему. Система також забезпечить можливість інтеграції засобів спілкування між студентами та науковими керівниками і дозволить обмінюватися робочими файлами у форматі Word для зручності написання та коригування звітів.

Методи дослідження. Для досягнення мети та вирішення поставлених завдань застосовувалися такі методи: аналіз і систематизація наукових джерел, методи порівняння і узагальнення, а також методи інженерного проектування та розробки інформаційних систем для побудови структури системи створення етапів дипломного проектування. Модель системи тестувалася на практичних

сценаріях з метою оцінки її ефективності.

Практичне значення одержаних результатів дослідження полягає в тому, що розроблена система дозволяє оптимізувати процес роботи на різних етапах дипломного проектування, а також сприяє самостійності студентів у виборі теми під певний напрямок і наукового керівника, при цьому даючи можливість пропонувати власні теми. Система забезпечує легке спілкування і обмін файлами між студентами і керівниками, що сприяє якості написання і редагування звітів та полегшує процес організації проектної діяльності студентів.

Апробація результатів дисертації. Основні положення та висновки дослідження висвітлювалися на III Всеукраїнській науково-практичній конференції «Системні науки та інформатика».

Ключові слова: дипломне проектування, вибір теми, вибір наукового керівника, система створення етапів дипломного проектування, обмін файлами, освітній процес, створення етапів.

ABSTRACT

Structure and Volume of the Qualification Work. The master's thesis consists of an introduction, five chapters, conclusions, and one appendix. The work includes references to 33 sources, 67 figures and 4 tables. The main body of the work is presented on 95 pages.

Relevance. The process of diploma project development plays a key role in preparing students for their future professional activities, as it allows them to apply acquired knowledge in practice. There is an increasing need to create an efficient system that organizes the stages of project development while considering the individual needs of students and the requirements of the educational institution. Such a system should not only optimize task management but also facilitate students' choice of topics within specific fields, allow them to select an academic supervisor, and support a flexible interaction process. The relevance of this topic lies in the need to enhance the quality of student preparation for diploma project development and to create conditions for their independent work in a digital environment.

The purpose of the work is to develop a system that will optimize the stages of diploma project development, enabling students to select a thesis topic within a specific area and to choose an academic supervisor or propose their own topic. The system will also enable communication tools between students and supervisors and facilitate the exchange of working files in Word format to improve the convenience of writing and editing reports.

Research Methods. To achieve the purpose and solve the defined tasks, the following methods were applied: analysis and systematization of scientific sources, methods of comparison and generalization, as well as engineering design methods and information systems development for constructing the structure of a diploma project staging system. The system model was tested in practical scenarios to evaluate its effectiveness.

Practical Significance of the Research Results lies in the fact that the developed system allows for the optimization of the process across various stages of diploma project development. It also supports students' independence in choosing topics within specific fields and in selecting an academic supervisor, while providing the possibility of proposing their own topics. The system facilitates smooth communication and file exchange between students and supervisors, which enhances the quality of report writing and editing, and simplifies the process of organizing student project activities.

Testing of the results of the dissertation. The main provisions and conclusions of the study were presented at the III All-Ukrainian Scientific and Practical Conference “System Sciences and Informatics”.

Keywords: diploma project development, diploma project staging system, topic selection, supervisor selection, file exchange, educational process, stage creation.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

MVC Model-view-controller — Модель–представлення–контролер

HTML Hypertext Markup Language — Мова розмітки гіпертекстових документів

IDE Integrated development environment — Інтегроване середовище розробки

CSS Cascading Style Sheet — Мова каскадних стилів

API Application programming interface — Програмний інтерфейс застосунку

HTTP Hyper Text Transfer Protocol — Протокол

SQL Structured query language — Мова структурованих запитів

JSON JavaScript Object Notation — Формат файлу

XML Extensible Markup Language — Розширена мова розмітки

REST Representational State Transfer — Передача представницького стану

СКБД Система керування базами даних

ВСТУП

Мета магістерської дисертації полягає в розробці та реалізації програмного забезпечення, яке допоможе автоматизувати процес створення етапів дипломного проектування.

Завданням магістерської дисертації було створити систему, яка зможе автоматизувати процес створення етапів дипломного проектування. Основним шляхом реалізації цієї задачі було обрано створити веб-додаток, який буде об’єднувати в собі бізнес-логіку самої системи та інтерфейс користувача для роботи.

Основним інструментом для виконання цього завдання та розробки бізнес- логіки є використання баз даних. Вони є ключовим елементом більшості сучасних інформаційних систем, адже навколо баз даних будується вся структура системи підприємства. Тому теоретичні основи та практичне застосування баз даних отримують значну увагу протягом усього життєвого циклу інформаційних систем. Тривалий час реляційні бази даних залишалися основним типом, і сьогодні вони вважаються класичними. Сучасні бази даних повинні вирішувати завдання масштабованості, зберігати й обробляти великі обсяги інформації, забезпечувати оперативний аналіз даних, інтеграцію з Інтернетом, контроль доступу користувачів і захист інформації під час її передачі в мережі.

Основні завдання магістерської дисертації:

- створити базу даних, яка буде включати в собі дані про користувачів системи, групи студентів, типи файлів, самі файли які додали користувачі системи, завдання студентів та дані про чати користувачів, також основні дані про практику студентів: такі як напрямки та теми;
- створити систему, яка буде забезпечувати безпеку даних та безпосередньо навігацію в самому додатку;
- передбачити можливість перегляду, редагування,

додаваннята пошуку введених даних;

- передбачити можливість забезпечення створення етапів дипломного проектування та автоматизації процесу обрання тем студентів;
- створити можливість формування звітів адміністраторами згідно оброблених даних всередині системи.

Остаточним результатом магістерської дисертації повинна бути розроблена та протестована система, яка дозволить автоматизувати процес створення етапів дипломного проектування, покращуючи ефективність та точність цього процесу.

1 ПОСТАНОВКА ЗАВДАННЯ ТА ОСНОВНІ ЗАВДАННЯ

Мета магістерської дисертації — полягає в розробці та реалізації програмного забезпечення, яке допоможе автоматизувати процес створення етапів дипломного проектування

Основним завданням такої системи є ефективне використання наявних ресурсів, мінімізація конфліктів та забезпечення створення етапів дипломного проектування.

1.1 Постановка завдання

Дипломне проектування є одним із найважливіших етапів навчального процесу для студентів вищих навчальних закладів. Воно вимагає значної організації, зокрема розподілу завдань, контрольних точок і взаємодії між студентом і науковим керівником. Однак у сучасних реаліях цей процес часто зустрічається з низкою проблем, пов'язаних із неефективною комунікацією, нестачею централізованого управління та труднощами в моніторингу прогресу студентів. Для вирішення цих питань пропонується створити систему "Система створення етапів дипломного проектування", яка автоматизує та оптимізує цей процес.

Основною метою розробки системи є автоматизація етапів дипломного проектування, що дозволить викладачам та студентам ефективніше керувати проектами [2], покращить комунікацію між ними та підвищить загальний рівень організації роботи. Система повинна надавати інструменти для створення та управління дипломними проектами, що включатимуть різні етапи — від вибору теми до надсилання готових звітів та щоденників.

Першим кроком у роботі системи є розробка етапів проекту. Викладачі повинні мати можливість створювати теми під напрямками для дипломного проекту – це означає що в системі будуть присутні такі речі як вибір теми, додавання матеріалів, написання звіту, його перевірка, редагування та фінальна

сдача. Кожен етап повинен бути чітко визначений із вказаними дедлайнами та завданнями, які студент повинен виконати.

Другий важливий аспект системи — це керування процесом виконання дипломного проекту. Викладачі можуть призначати студентам конкретні завдання для кожного етапу. Наприклад, під час початкового етапу студент може отримати завдання на вибір теми, підготовку пропозиції та плану дослідження. Кожен етап повинен містити інструкції та вимоги щодо виконання, а також інструменти для завантаження документів та інших матеріалів, необхідних для виконання завдань. Крім того, система повинна забезпечувати моніторинг прогресу студентів.

Викладачі мають можливість контролювати виконання завдань на кожному етапі, перевіряти матеріали, відслідковувати стан готовності магістерської дисертації. Це дозволить уникнути проблем з несвоєчасним виконанням завдань, що часто є причиною затримок у дипломному процесі. Студенти, зі свого боку, також можуть бачити свій прогрес і розуміти, на якому етапі вони знаходяться, що допоможе їм краще планувати час і виконувати завдання вчасно.

Важливим аспектом роботи системи є забезпечення комунікації між викладачами та студентами. Система повинна мати функції обміну повідомленнями, що дозволить студентам швидко зв'язуватися зі своїми науковими керівниками, ставити запитання або отримувати зворотній зв'язок щодо виконаних завдань. Окрім того, викладачі можуть коментувати завантажені студентами матеріали, вносити правки або надавати рекомендації щодо подальшої роботи.

Не менш важливою функцією системи є генерація звітів про дипломне проектування.

Запровадження такої системи дозволить значно покращити організацію дипломного проектування, зробити його прозорим та більш контрольованим як для студентів, так і для викладачів. Автоматизація рутинних процесів, зокрема

розподілу завдань, моніторингу прогресу та обміну інформацією, значно зменшить адміністративні витрати наукових керівників і дозволить їм більше зосередитися на змістовній частині дипломного проекту.

Система створення етапів дипломного проектування також допоможе студентам краще організувати свою роботу, розподіляти час та відповідальніше підходити до виконання завдань. Наявність чітких дедлайнів, завдань та можливості отримувати оперативний зворотний зв'язок дозволить студентам уникнути затримок та проблем із захистом магістерської дисертації.

Загалом, така система сприятиме підвищенню якості підготовки дипломних проектів, що позитивно вплине на загальний рівень знань студентів та поліпшить процес їхньої підготовки до реальної роботи в професійній сфері. Автоматизація процесів дипломного проектування — це важливий крок у напрямку модернізації вищої освіти та підвищення ефективності навчання.

1.2 Основні завдання

Основні завдання магістерської дисертації:

- створити базу даних, яка буде включати в собі дані про користувачів системи, групи студентів, типи файлів, самі файли які додали користувачі системи, завдання студентів та дані про чати користувачів, також основні дані про практику студентів: такі як напрямки та теми;
- створити систему, яка буде забезпечувати безпеку даних та безпосередньо навігацію в самому додатку;
- передбачити можливість перегляду, редагування, додавання та пошуку введених даних;
- передбачити можливість забезпечення створення етапів дипломного проектування та автоматизації процесу обрання тем студентів;
- створити можливість формування звітів адміністраторами згідно оброблених даних всередині системи.

Остаточним результатом магістерської дисертації повинна бути розроблена та протестована система, яка дозволить автоматизувати процес створення етапів дипломного проектування, покращуючи ефективність та точність цього процесу.

Висновки до розділу 1

У цьому розділі було визначено основні цілі та проблеми, які потребують вирішення в процесі дипломного проектування. Основною метою розробки системи є автоматизація етапів дипломного проектування, що дозволить поліпшити управління проектами та взаємодію між студентами і викладачами.

Зокрема, система має забезпечити чітке визначення етапів проекту, включаючи вибір теми, підготовку матеріалів, перевірку та фінальну здачу. Передбачено, що система також дозволить викладачам контролювати виконання завдань та моніторити прогрес студентів, що сприятиме уникненню затримок у виконанні проектів.

Важливим аспектом є впровадження механізмів комунікації, які дозволять студентам отримувати зворотний зв'язок від наукових керівників у реальному часі. Розробка системи забезпечить централізоване управління дипломними проектами, що зменшить адміністративні навантаження на викладачів і полегшить процес навчання.

Отже, визначені завдання формують основу для подальшої розробки та реалізації програмного забезпечення, яке сприятиме підвищенню ефективності дипломного проектування у вищих навчальних закладах.

2 АНАЛІЗ, ДОСЛІДЖЕННЯ ІСНУЮЧИХ МЕТОДІВ

Ефективне виконання магістерської дисертації потребує не лише практичних знань, а й глибокого розуміння теоретичних засад та існуючих рішень у відповідній галузі. Для цього необхідно провести ретельний аналіз теоретичних основ, що становлять базу для розробки сучасних інструментів і підходів до управління освітніми процесами.

Огляд існуючих рішень дозволяє визначити функціональні можливості, обмеження та потенційні сфери застосування наявних систем. Цей розділ спрямований на дослідження теоретичних аспектів, що лежать в основі процесу дипломного проектування, а також аналіз уже розроблених платформ.

2.1 Теоретична складова

Дипломне проектування є завершальним етапом підготовки фахівців у вищих навчальних закладах і являє собою комплексний процес, що вимагає системного підходу до організації та управління. Система створення етапів дипломного проектування базується на фундаментальних принципах проектного менеджменту та педагогічної науки, інтегруючи сучасні інформаційні технології для оптимізації освітнього процесу.

Основоположним аспектом системи є структурування процесу дипломного проектування на логічно пов'язані етапи, кожен з яких має чітко визначені цілі, завдання та очікувані результати. Це дозволяє забезпечити послідовний та методичний підхід до виконання магістерської дисертації, одночасно надаючи можливість контролю та корекції процесу на кожному етапі.

Теоретичний фундамент системи включає методології управління проектами, зокрема каскадну (waterfall) та гнучку (agile) моделі розробки, адаптовані до специфіки освітнього процесу. Важливим аспектом є також інтеграція принципів педагогічного дизайну, що забезпечує ефективну

взаємодію між студентом та науковим керівником, а також сприяє формуванню необхідних компетенцій майбутнього фахівця.

Система базується на сучасних теоріях управління знаннями та компетентнісному підході, що дозволяє формувати індивідуальні траєкторії виконання дипломних робіт з урахуванням специфіки обраної теми та особливостей студента. При цьому враховуються принципи академічної доброчесності та наукової етики, які є невід'ємною частиною сучасної освітньої парадигми.

2.1 Аналіз вхідних і вихідних даних

Як і будь-який веб-інтерфейс, дана програма повинна приймати, обробляти та виводити дані, а також їх зберігати.

Вхідними даними у програмі виступають додані дані кожним типом користувача на певному етапі. Система створення етапів дипломного проектування повинна мати можливість приймати, обробляти та зберігати різноманітні дані, необхідні для ефективного управління процесом виконання дипломної роботи. Ключовими вхідними даними є інформація, що вводиться користувачами різних ролей на кожному з етапів: дані про студентів, керівників, консультантів, напрямки тематик, файли з результатами етапів, записи про завдання та події, повідомлення й зворотній зв'язок. Система має забезпечувати зручні інтерфейси для введення, редагування та управління цими даними.

Для ефективного зберігання, систематизації та взаємозв'язку введених даних система повинна мати добре продуману структуру бази даних. Основними сутностями можуть бути студенти, керівники/консультанти, дипломні проекти, файли, повідомлення/зворотний зв'язок, пов'язані між собою відповідними зв'язками. Така структура дозволить відображати реальні взаємовідносини в процесі виконання дипломних робіт.

Вихідними даними мають бути добре структуровані та інтуїтивно зрозумілі таблиці, що відображають різні зрізи даних: загальний список дипломних проектів, деталізована інформація по кожному проекту, звіти по групах, напрямкам, статусах, календар подій. Користувачі матимуть змогу фільтрувати, сортувати та шукати дані за різноманітними критеріями.

Саме такий спосіб виводу даних, я вважаю, найбільш доцільним при даній постановці задачі, так як інформація подана в таблицях легше сприймається та дає змогу швидше знайти потрібну користувачеві інформацію.

2.2 Аналоги, які мають близький функціонал

У сучасному освітньому просторі існує ряд систем, що забезпечують підтримку процесу дипломного проектування [17]. Розглянемо більш детально аналоги, які мають близький функціонал.

Однією з таких систем є платформи для управління освітнім процесом, що дозволяють викладачам і студентам зручно організовувати комунікацію та відслідковувати прогрес роботи над проектом [19]. Крім цього, деякі системи надають доступ до інструментів для спільного редагування документів, що спрощує командну роботу [27].

2.2.1 Google Workspace - універсальна платформа для спільної роботи

Workspace - це потужний набір хмарних інструментів, які можуть стати відмінною основою для управління процесом створення магістерської дисертації [20]. Ключовими додатками, що входять до Google Workspace і можуть бути корисними, є Google Docs, Google Sheets, Google Classroom, Google Slides та Google Drive [15, 29]. Загальне відображення логотипів зображено на рисунку 2.1

Рисунок 2.1 – набір хмарних інструментів Google Workspace

Google Docs дозволяє студентам, що працюють над дипломом, спільно працювати над текстовими документами, такими як пояснювальні записки, звіти, розділи роботи тощо [17, 19]. Кілька людей можуть одночасно редагувати один і той же документ, залишаючи коментарі та пропозиції. Це суттєво спрощує процес колективного написання та узгодження різних частин дипломної роботи [30].

Google Sheets, у свою чергу, ідеально підходить для ведення різноманітних таблиць, необхідних для дипломного проекту [24]. Тут можна створювати календарі з дедлайнами, складати кошториси витрат, вести статистику та аналітику, будувати графіки та діаграми [21]. Доступ до цих даних можна надавати усім учасникам проекту для спільного використання.

Для створення презентацій, наприклад, для захисту дипломної роботи, дуже зручно використовувати Google Slides [23]. Цей інструмент дозволяє спільно працювати над слайдами, легко вставляти мультимедійний контент, а також ділитися презентацією з аудиторією безпосередньо з веб-браузера [29].

Об'єднуючим центром усіх цих додатків виступає Google Drive - єдине хмарне сховище, де можна зберігати та синхронізувати всі файли, пов'язані з дипломним проектом. Це забезпечує легкий доступ до потрібних матеріалів з будь-якого пристрою, а також можливість надавати спільний доступ іншим учасникам.

Крім загальних інструментів Google Workspace, для управління

дипломним проектуванням може бути корисним спеціалізований додаток Google Classroom. Цей сервіс, який є частиною Google Workspace, створений для освітніх потреб, але його функціональність може бути успішно застосована і до роботи над дипломними проектами.

Google Classroom дозволяє створювати окремий "клас" для групи студентів, що працюють над дипломним проектом. У межах цього класу можна публікувати різноманітні завдання, інструкції та дедлайни, пов'язані з етапами виконання дипломної роботи. Студенти, у свою чергу, можуть завантажувати свої виконані роботи, а викладач - надавати їм зворотний зв'язок та оцінки.

Таким чином, Google Classroom допомагає структурувати та організувати весь процес роботи над дипломом. Він забезпечує чіткість у постановці завдань, прозорість у відстеженні прогресу та зручність у зборі й перевірці виконаних робіт студентів.

Крім того, Google Classroom тісно інтегрований з іншими сервісами Google Workspace. Це дозволяє легко підключати Google Docs, Google Sheets, Google Slides та інші інструменти безпосередньо в межах класу, забезпечуючи єдине середовище для роботи над дипломним проектом.

Аналіз існуючих аналогів демонструє, що жодна з представлених систем не забезпечує повного покриття всіх аспектів процесу дипломного проектування, зокрема, в контексті специфічних вимог вітчизняної системи вищої освіти. Це обґрунтовує необхідність розробки нової системи, яка б враховувала як позитивний досвід існуючих рішень, так і специфічні потреби українських вищих навчальних закладів.

Однією з головних переваг цих інструментів є безкоштовний доступ і легка інтеграція. Оскільки Google Workspace надає комплекс сервісів, студенти та викладачі можуть користуватися ними без додаткових витрат. Інтеграція між сервісами створює цілісну екосистему, де усі необхідні функції доступні в єдиному просторі.

Google Workspace також забезпечує зручні можливості для спільної

роботи. Студенти можуть одночасно працювати над документами, таблицями та презентаціями, що значно полегшує колективну роботу. Крім того, ведення спільних календарів і планування задач сприяє ефективному управлінню часом та завданнями дипломного проекту.

Ще одна вагома перевага полягає у можливості чіткої організації процесу. Google Classroom дозволяє структуровано розподіляти завдання, встановлювати дедлайни та розбивати дипломну роботу на етапи. Це сприяє дисципліні в роботі й надає викладачам можливість відстежувати прогрес студентів, оцінювати виконані роботи та швидко надавати зворотний зв'язок.

Проте Google Workspace та Google Classroom мають і певні обмеження. Їхній базовий функціонал може бути недостатнім для управління складними проектами, оскільки вони не підтримують усіх можливостей спеціалізованих систем, призначених для проектного менеджменту.

Деякі розширені функції, як-от діаграми Ганта, управління ресурсами та розвинені аналітичні інструменти, можуть бути недоступними або обмеженими. Це знижує ефективність цих інструментів у складних проектах, які потребують детальної аналітики та планування.

Крім того, може виникнути потреба у координації з тими учасниками, які не користуються Google Workspace. Хоча ці сервіси пропонують значні можливості для колективної роботи, інколи виникає необхідність взаємодії з іншими зацікавленими сторонами, які можуть працювати у інших системах.

2.2.2 Jira - гнучка система управління проектами

Jira - це потужна платформа для управління проектами та відстеження завдань, яка може бути дуже корисною при роботі над дипломним проектом. Jira надає широкий спектр можливостей, що роблять її ефективним інструментом для структурування та контролю виконання дипломної роботи [16, 27, 33]. Систему схематично зображено на рисунку 2.2.

Рисунок 2.2 – Jira потужна платформа для управління проектами

Одна з ключових особливостей Jira - це можливість створювати окремі проекти для кожної дипломної роботи [30]. У межах таких проектів можна деталізувати всі необхідні завдання, прив'язуючи їх до конкретних етапів і дедлайнів [22]. Це дозволяє чітко визначати та розподіляти роботу, а також відслідковувати прогрес виконання.

Jira також надає широкі можливості для планування робіт [25]. Тут можна будувати детальні графіки Ганта, призначати відповідальних за кожне завдання, встановлювати пріоритети та моніторити статуси виконання [31]. Така структурованість допомагає тримати всі аспекти дипломного проекту під контролем.

Спільна робота та комунікація між учасниками - ще одна важлива перевага Jira [28]. Студенти, керівник та інші зацікавлені сторони можуть залишати коментарі до завдань, прикріплювати файли, обговорювати проблеми просто в межах платформи [18]. Це значно спрощує координацію та взаємодію в команді.

Крім того, Jira надає широкі можливості для звітності та аналітики. Тут можна створювати різноманітні звіти - від простих списків статусів до розгорнутих панелей ключових показників. Це дає змогу відслідковувати прогрес, виявляти проблемні зони та приймати обґрунтовані управлінські рішення.

Важливою характеристикою Jira є її гнучкість та масштабованість. Платформа достатньо адаптивна, щоб налаштовуватися під специфіку дипломного проектування, а також може легко масштабуватися відповідно до розміру та складності проекту. Це робить Jira придатною як для невеликих, так і для масштабних дипломних робіт. Окрім потужних можливостей самої Jira, її можна успішно інтегрувати з Google Classroom для ще ефективнішого управління дипломним проектуванням. Таке поєднання двох інструментів створює комплексне рішення, що дозволяє поєднати можливості управління проектами та переваги навчального середовища.

Google Classroom дозволяє створювати окремі "класи" для груп студентів, що працюють над дипломними роботами. У межах цих класів можна публікувати завдання, інструкції, дедлайни, а також збирати виконані студентами роботи. Інтеграція з Jira допомагає синхронізувати ці елементи між двома платформами.

Наприклад, коли в Jira створюється нове завдання, пов'язане з дипломним проектом, воно може автоматично публікуватися в Google Classroom для ознайомлення студентів. Своєю чергою, виконані студентами роботи можуть імпортуватися назад до Jira, де викладач зможе їх переглянути та оцінити.

Крім того, інтеграція Jira та Google Classroom забезпечує спільний доступ учасників проекту. Студенти та керівник можуть отримувати доступ до Jira безпосередньо через Google Classroom, що спрощує навігацію та взаємодію в єдиному середовищі.

Така взаємодія Jira та Google Classroom також дозволяє викладачу ефективно відстежувати прогрес студентів. Він може переглядати в Jira, на якій стадії знаходиться кожен студент, та надавати їм необхідний зворотний зв'язок прямо в межах цієї інтегрованої системи.

Загалом, інтеграція Jira та Google Classroom створює потужне рішення для управління дипломними проектами. Вона поєднує переваги гнучкої системи управління проектами з можливостями структурованого навчального

середовища, підвищуючи ефективність роботи над дипломом [26].

Jira надає широкий спектр інструментів для детального планування робіт, призначення відповідальних, моніторингу статусів і формування різноманітних звітів. Це допомагає тримати весь процес виконання дипломного проекту під контролем.

Jira достатньо адаптивна, щоб відповідати вимогам конкретного дипломного проекту . Платформу можна налаштувати під специфічні завдання, етапи, звітність тощо.

Jira забезпечує зручні інструменти для спільної роботи - коментарі, прикріплення файлів, обговорення безпосередньо в проекті. Це значно спрощує взаємодію між студентами та керівником [32].

Синхронізація з Google Classroom для кращої організації навчального процесу. Інтеграція Jira та Google Classroom дозволяє поєднати можливості управління проектами з перевагами структурованого навчального середовища.

Водночас, використання Jira також має певні обмеження та потенційні недоліки.

Jira є достатньо складним інструментом, тому для ефективного використання студентам та керівникам може знадобитися певний час на опанування платформи.

На відміну від безкоштовних Google Workspace та Classroom, Jira пропонує платні тарифні плани, що може створювати додаткові фінансові витрати для навчального закладу.

Хоча інтеграція Jira та Google Classroom може бути корисною, вона також додає додаткову складність у налаштування та підтримку такої гібридної системи.

Безкоштовна версія Jira має обмежений функціонал, тому для повноцінного використання може знадобитися перехід на платну ліцензію.

Загалом, Jira є потужним інструментом для управління проектами, включаючи дипломні роботи. Інтеграція з Google Classroom може значно

посилити можливості структурування, планування та спільної роботи над дипломним проектом [34]. Однак складність Jira, її вартість та необхідність інтеграції можуть бути перешкодами для деяких навчальних закладів. Тому вибір між Jira та іншими альтернативами слід робити з урахуванням конкретних потреб та ресурсів.

Висновки до розділу 2

Проведений аналіз предметної області системи створення етапів дипломного проектування виявив комплексний характер даної проблематики, що охоплює як теоретичні засади проектного управління, так і практичні аспекти організації освітнього процесу. Детальне вивчення вхідних та вихідних даних системи показало необхідність структурованого підходу до обробки різноманітної інформації, включаючи нормативні документи, персональні дані та результати проектування. Аналіз існуючих аналогів продемонстрував, що наявні рішення не повністю відповідають специфічним потребам вітчизняних навчальних закладів, що підтверджує актуальність розробки нової системи. Особливу увагу в подальшій розробці слід приділити питанням гнучкості налаштування етапів проектування та забезпечення ефективної комунікації між усіма учасниками процесу. Створення такої системи дозволить оптимізувати процес дипломного проектування та підвищити якість підготовки фахівців у вищих навчальних закладах.

3 АПАРАТ ВИРІШЕННЯ ДЛЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

Інтегрованим середовищем розробки у даній роботі було вибрано IntelliJ IDEA. Основну мову програмування для цієї підсистеми було обрано JavaScript. Для більш продуктивного та розширеного функціоналу використано фреймворки Node.js та Angular. Для взаємодії з базою даних — MySQL, та в ролі серверу — OpenServer.

3.1 Середовище розробки IntelliJ IDEA

IntelliJ IDEA — це потужне інтегроване середовище розробки (IDE), розроблене компанією JetBrains, яке надає розробникам широкий спектр інструментів для ефективної розробки програмного забезпечення. Вона підтримує різні мови програмування та технології, зокрема JavaScript, TypeScript, Angular, Node.js та інші. Інтуїтивний інтерфейс, зручні гарячі клавіші та широкі можливості налаштування роблять IntelliJ IDEA популярним вибором серед професійних розробників [10].

IntelliJ IDEA забезпечує розширену підтримку Angular, включаючи автозавершення коду для TypeScript, HTML та CSS, інтеграцію з Angular CLI та потужні інструменти для налагодження [3]. IDE також надає можливість інтеграції з інструментами тестування, такими як Karma і Protractor, що значно спрощує процес створення та підтримки тестів. Завдяки підсвічуванню синтаксису та перевірці помилок у реальному часі, розробники можуть швидко виявляти та виправляти проблеми в коді, підвищуючи ефективність своєї роботи.

Для розробки на Node.js IntelliJ IDEA пропонує вбудовану підтримку, що дозволяє розробникам легко створювати серверні додатки. IDE надає засоби для роботи з npm, включаючи можливість керування пакетами та автоматизації процесів. Завдяки вбудованому інструменту налагодження для Node.js, розробники можуть легко запускати та налагоджувати свої програми без

необхідності використовувати сторонні інструменти. Крім того, підтримка JavaScript і TypeScript на рівні платформи дозволяє легко працювати з кодом для сервера та клієнта в одному середовищі.

IntelliJ IDEA надає можливості для командної роботи, зокрема через інтеграцію з системами контролю версій (Git, SVN), що дозволяє легко відстежувати зміни у проєкті та працювати в команді. Крім того, IntelliJ забезпечує підтримку налаштувань проєктів для великих команд, що спрощує процес інтеграції нових членів команди та полегшує управління проєктами.

Таким чином, IntelliJ IDEA є чудовим вибором для розробки як Angular-додатків, так і Node.js-серверів завдяки широким можливостям, інтеграції з основними інструментами та простоті використання.

3.2 Серверна платформа Node.js

Node.js — це платформа для виконання JavaScript на сервері, яка базується на рушії V8 від Google Chrome. Вона дозволяє виконувати JavaScript [13] не тільки в браузері, але й на сервері, що робить її ідеальним інструментом для розробки високопродуктивних, масштабованих веб-додатків. Вперше випущена в 2009 році, Node.js набула великої популярності завдяки своїй легковагій архітектурі та подіям, керованим неблокуючими операціями введення/виведення [6].

Однією з ключових особливостей Node.js є її асинхронна модель введення/виведення, яка дозволяє обробляти багато запитів одночасно без блокування основного потоку. Це досягається через використання подієво-орієнтованого циклу подій (event loop), що дає можливість обслуговувати тисячі підключень із мінімальними витратами на ресурси. У порівнянні з традиційними серверними мовами, такими як PHP або Python, Node.js забезпечує значно кращу продуктивність при обробці великого числа одночасних запитів.

Однією з основних переваг використання Node.js є можливість писати JavaScript як на сервері, так і на клієнті, що спрощує процес розробки. Це дає змогу використовувати один стек технологій (так званий full-stack JavaScript), що робить код більш уніфікованим [3], а команду розробників — більш універсальною. Більше не потрібно перемикатися між різними мовами для різних частин системи, що полегшує розробку і підтримку коду.

Node.js відмінно підходить для розробки масштабованих систем, таких як веб-сервіси та API, оскільки дозволяє обробляти велику кількість одночасних підключень із мінімальним споживанням ресурсів. За рахунок горизонтального масштабування (розподіл навантаження на кілька серверів) Node.js може обробляти високі навантаження та підтримувати безперебійну роботу великих додатків.

Node.js має одну з найбільших екосистем пакетів завдяки менеджеру пакетів npm (Node Package Manager), що надає доступ до понад мільйона модулів і бібліотек. Це дозволяє розробникам легко використовувати готові рішення для багатьох типових завдань, таких як автентифікація користувачів, обробка запитів, інтеграція з базами даних та інше. npm спрощує управління залежностями та інтеграцію нових модулів у проєкт, що значно прискорює процес розробки.

Node.js має активну і велику спільноту розробників, що сприяє постійному оновленню платформи та розвитку нових інструментів. Відкрита екосистема, багатий набір бібліотек та популярність платформи роблять Node.js чудовим вибором для сучасних веб-додатків, включаючи системи реального часу (наприклад, чати або стримінгові сервіси), API для мобільних додатків та серверні системи.

Таким чином, Node.js [6] є потужною платформою для серверної розробки, яка поєднує продуктивність, масштабованість та гнучкість, що робить її ідеальним вибором для сучасних високонавантажених веб-додатків.

3.3 Фреймворк Angular

Angular — це потужний фронтенд-фреймворк, розроблений компанією Google, який дозволяє створювати динамічні односторінкові додатки (SPA) з високою продуктивністю. Використовуючи TypeScript як основну мову розробки, Angular пропонує структуровану і масштабовану архітектуру, що спрощує розробку, тестування та підтримку великих веб-додатків. Завдяки модульній системі, двосторонньому зв’язку даних та потужній підтримці інструментів, Angular став одним з найбільш популярних рішень для побудови складних інтерфейсів. Angular пропонує архітектуру, засновану на компонентах, де кожен елемент додатку розділяється на незалежні, багаторазово використовувані модулі. Це дозволяє створювати додатки з чіткою структурою та легко масштабовані. Компонентний підхід робить код більш організованим і зрозумілим, що спрощує як розробку, так і подальшу підтримку проєктів [8].

Двосторонній зв'язок даних (two-way data binding) є ще однією ключовою перевагою Angular. Це означає, що зміни у моделі даних автоматично відображаються в інтерфейсі користувача і навпаки. Така можливість значно скорочує кількість ручного коду та спрощує інтерактивність веб-додатків, що є важливим для розробки динамічних інтерфейсів.

Angular побудований на основі TypeScript, що є надмножиною JavaScript. TypeScript надає строгі типи, що підвищує надійність коду і полегшує виявлення помилок під час компіляції, ще до того, як код виконується. Це робить Angular особливо привабливим для великих проєктів, де важливо підтримувати чистоту коду і мінімізувати помилки під час його розробки та підтримки.

Однією з сильних сторін Angular є його модульна система. Angular розбиває додаток на незалежні модулі, що дозволяє з легкістю повторно використовувати компоненти і бібліотеки між різними частинами проєкту. Цей підхід сприяє кращій організації коду і полегшує масштабування додатків. Крім

того, Angular активно підтримує розробку з акцентом на тестування, що включає як юніт-тести, так і інтеграційні тести, забезпечуючи високу якість коду [8].

Angular інтегрується з потужними інструментами для розробки, такими як Angular CLI (Command Line Interface), який спрощує створення нових проєктів, генерування компонентів, модулів та сервісів. CLI також дозволяє легко налагоджувати та збирати проєкти, що значно зменшує кількість рутинної роботи. Також Angular відмінно працює з інструментами для тестування, такими як Jasmine та Karma, що дозволяє налаштовувати автоматизовані тести і забезпечувати стабільність коду.

Angular пропонує вбудовані механізми для підвищення продуктивності додатків, такі як лениве завантаження модулів (lazy loading) та дерево-трусінг (tree shaking), які зменшують розмір кінцевого файлу і покращують швидкість завантаження. Завдяки цим інструментам, додатки, створені на Angular, залишаються швидкими і добре оптимізованими навіть при зростанні складності та обсягу коду.

Angular має велику спільноту розробників та регулярну підтримку від Google, що забезпечує стабільний розвиток і регулярні оновлення фреймворку. Це також гарантує, що Angular буде залишатися актуальним вибором для розробки веб-додатків у найближчі роки, завдяки стабільній підтримці і активному внеску спільноти [8].

Таким чином, Angular є ідеальним вибором для розробки сучасних веб-додатків, що вимагають високої продуктивності, структурованого підходу і можливостей масштабування. Завдяки своїй модульній архітектурі, підтримці TypeScript та інтеграції з потужними інструментами, Angular дозволяє створювати стабільні, ефективні та легко підтримувані рішення.

3.4 Система керування базами даних MySQL

MySQL — це одна з найпопулярніших систем керування реляційними базами даних (СУБД) у світі. Вона використовує SQL (Structured Query Language) для керування, маніпуляції та отримання даних. MySQL є відкритим програмним забезпеченням і підтримує широкий спектр операційних систем, включаючи Linux, Windows та macOS. Завдяки своїй надійності, продуктивності та масштабованості, MySQL часто використовується в поєднанні з веб-технологіями, такими як PHP, Node.js і Angular, для створення повнофункціональних веб- додатків [7].

Основна архітектура MySQL базується на реляційній моделі, що дозволяє організовувати дані у вигляді таблиць, які можуть бути взаємопов’язаними між собою через ключі. Це робить MySQL чудовим вибором для проєктів, де важливо забезпечити цілісність даних, ефективне зберігання і швидкий доступ до інформації. За допомогою SQL-запитів розробники можуть створювати складні реляційні зв’язки між таблицями та проводити різні операції з даними, такі як фільтрація, сортування, агрегація та об’єднання [12].

MySQL добре відомий своєю високою продуктивністю і здатністю обробляти великі обсяги даних при високих навантаженнях. Це досягається за рахунок використання різних механізмів зберігання (storage engines), таких як InnoDB та MyISAM. InnoDB, наприклад, підтримує транзакції, забезпечуючи цілісність даних та збереження змін навіть у разі збою. MySQL також підтримує реплікацію баз даних, що дозволяє створювати резервні копії та розподіляти навантаження між кількома серверами, забезпечуючи високу доступність і швидкість доступу до даних у великих системах.

Одна з основних переваг MySQL — це підтримка транзакцій і відповідність принципам ACID (Atomicity, Consistency, Isolation, Durability). Це гарантує, що всі операції з базою даних будуть виконуватися атомарно, зберігаючи цілісність даних навіть у разі виникнення помилок або збоїв. Завдяки цьому MySQL є відмінним вибором для додатків, де важлива точність і

безпека операцій, наприклад, у фінансових системах або e-commerce платформах [7].

MySQL відмінно інтегрується з сучасними технологіями веб-розробки. Він широко використовується разом із серверними мовами програмування, такими як PHP, Node.js або Python, що дозволяє створювати динамічні додатки з базами даних. У поєднанні з фреймворками, такими як Angular, MySQL часто використовується для зберігання та керування даними користувачів, а також для обробки запитів в реальному часі через API.

MySQL забезпечує потужні засоби безпеки, включаючи систему користувачів і прав доступу, що дозволяє керувати доступом до баз даних та захищати дані від несанкціонованого доступу. Розробники можуть налаштувати детальні права для кожного користувача, визначаючи, хто може читати, змінювати або видаляти дані. Крім того, MySQL підтримує шифрування для забезпечення безпеки чутливих даних під час зберігання та передачі.

MySQL — це потужна, масштабована і надійна система керування базами даних, яка ідеально підходить для створення сучасних веб-додатків. Її реляційна архітектура, підтримка транзакцій, висока продуктивність та інтеграція з іншими веб-технологіями роблять MySQL вибором номер один для багатьох розробників по всьому світу. Завдяки активній підтримці спільноти та постійним оновленням MySQL залишається актуальним рішенням для будь-яких проєктів, незалежно від їхнього масштабу [7].

3.5 Локальний сервер Open Server

Open Server — це зручний інструмент для розгортання локального веб-сервера на персональному комп'ютері під керуванням Windows. Він створений для того, щоб надавати розробникам повноцінне середовище для тестування і розробки веб-додатків без необхідності використовувати реальні сервери. Open Server включає в себе всі необхідні компоненти для запуску веб-сервісів, такі як

веб-сервери Apache і Nginx [14], бази даних MySQL та PostgreSQL, і підтримує багато мов програмування, включаючи PHP, Python і Perl [9].

Open Server вирізняється своєю простотою у використанні та налаштуванні. Він має зручний інтерфейс, що дозволяє легко запускати і керувати різними компонентами сервера. Для встановлення не потрібно проходити складні конфігурації: достатньо завантажити Open Server і розпакувати його на комп'ютер. Всі налаштування вже попередньо задані, тому користувачі можуть одразу почати роботу над своїм проєктом. Крім того, Open Server підтримує багатопоточність, що дозволяє запускати декілька проєктів одночасно, а також перемикатися між різними версіями PHP або веб-серверів за потреби.

Open Server включає в себе всі необхідні інструменти для веб-розробки. Він підтримує популярні веб-сервери Apache та Nginx [14], які можна вибирати залежно від потреб проєкту. Крім того, Open Server дозволяє працювати з базами даних MySQL, MariaDB, PostgreSQL, SQLite, що дає гнучкість при розробці додатків різної складності. Система також підтримує мови програмування, такі як PHP, Python, Ruby, що дозволяє розробляти на різних платформах [9].

Висновки до розділу 3

У даному розділі були розглянуті важливі інструменти для веб-розробки, такі як IntelliJ IDEA, Node.js, Angular, MySQL та Open Server. IntelliJ IDEA забезпечує розробникам потужне середовище з інструментами для ефективного створення додатків на Angular та Node.js. Node.js надає продуктивну серверну платформу, що дозволяє створювати масштабовані та швидкодіючі системи. Angular є ідеальним фреймворком. Використання MySQL як надійної СУБД для управління даними та Open Server для локальної розробки забезпечує зручність тестування та налаштування проєктів в ізольованому середовищі.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Для реалізації системи створення етапів дипломного проектування. Такий підхід зручний як для користувачів, яким потрібен лише браузер і доступ до правильного посилання, так і для розробників, оскільки він надає широкі можливості для розширення функціоналу та оновлення системи при необхідності. Крім того, система може використовуватись локально за наявності відповідного програмного забезпечення. Базова модель роботи системи представлена на рисунку 4.1.

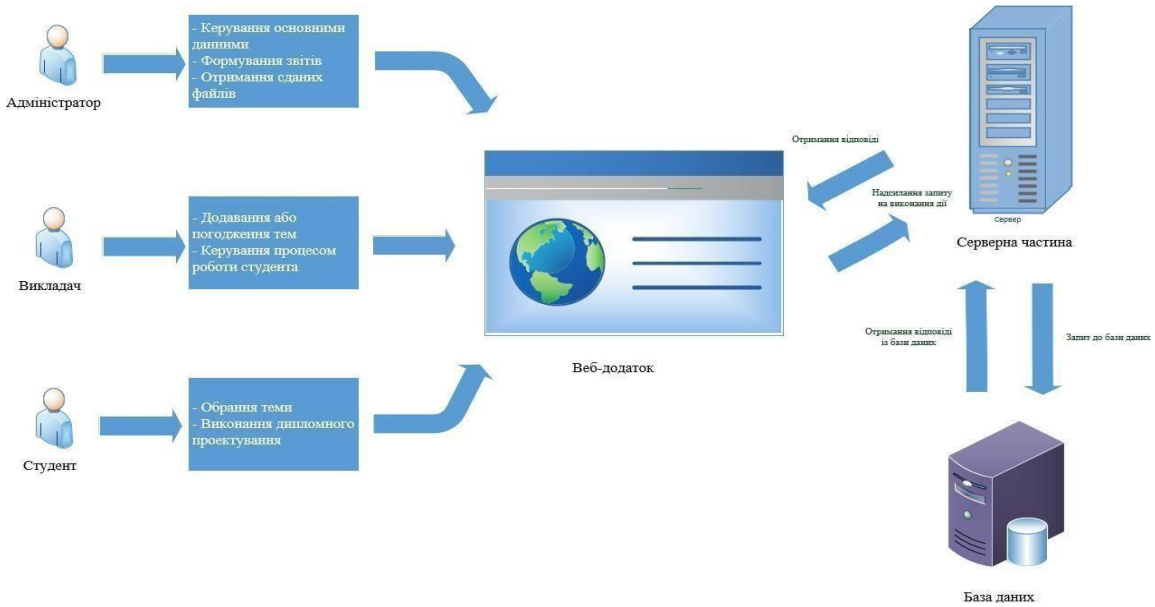

Рисунок 4.1 – Базова модель роботи з системою

4.1 Опис серверної частини системи

У даному підрозділі описується розробка серверної частини системи керування дипломною практикою. Серверна частина реалізована на базі Node.js з використанням фреймворку Express.js [11] для створення REST API. Основною метою серверної частини є забезпечення надійної та безпечної обробки запитів від клієнтської частини, управління даними та реалізація бізнес-логіки системи. Загальна модель архітектури серверної частини зображена на рисунку 4.2.

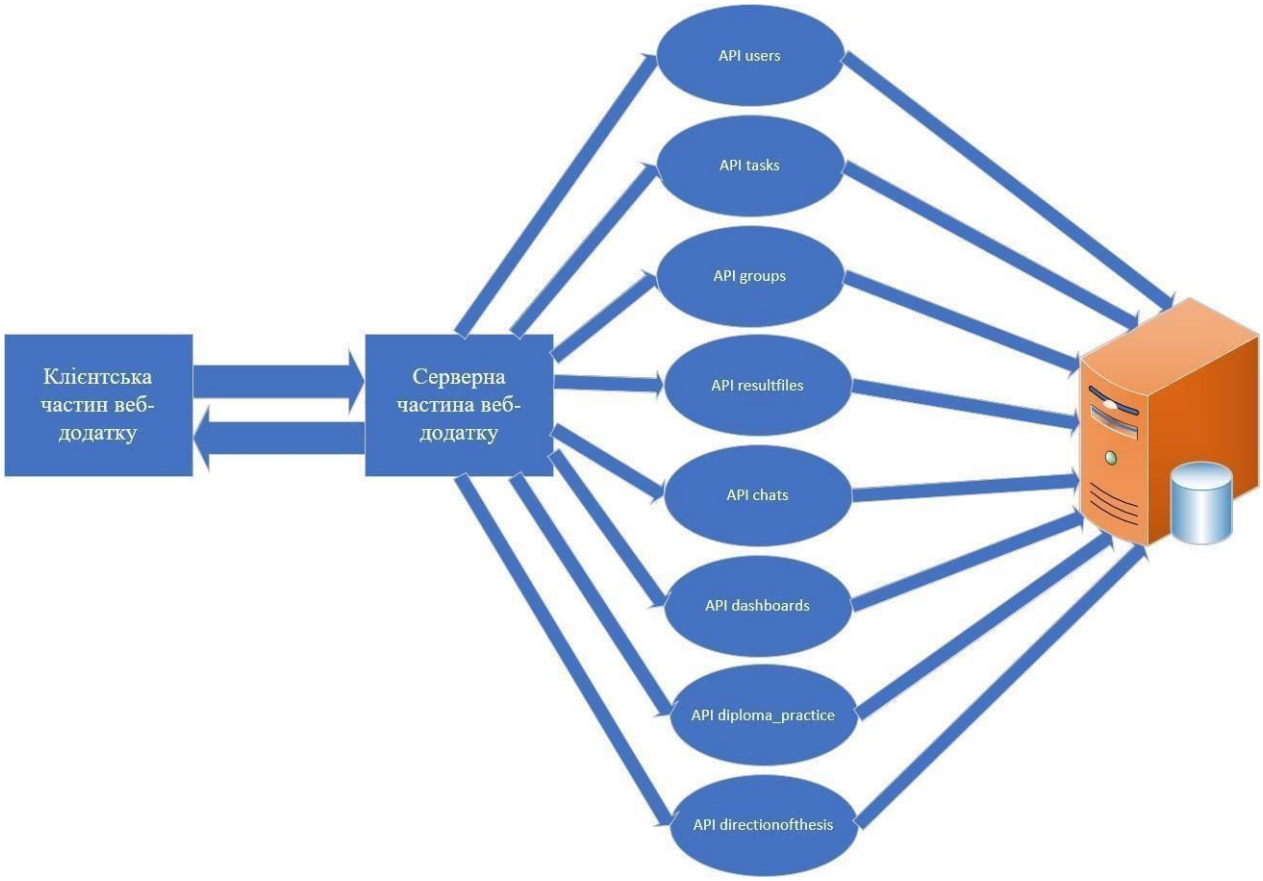

Рисунок 4.2 – Модель архітектури серверної частини

4.1.1 Архітектура та конфігурація серверної частини

Серверна частина додатку побудована за модульним принципом, що забезпечує чітке розділення відповідальності між різними компонентами системи. В якості системи управління базами даних використовується MySQL, що забезпечує надійне зберігання та швидкий доступ до даних. Для забезпечення безпечної взаємодії з клієнтською частиною налаштований CORS (Cross-Origin Resource Sharing), що дозволяє приймати запити лише з довірених джерел, у даному випадку з localhost:4200.

Для обробки HTTP-запитів використовуються стандартні middleware Express.js: express.json() для обробки JSON-даних та express.urlencoded() для обробки даних форм. Важливим компонентом системи є nodemailer, який забезпечує функціональність відправки повідомлень користувачам системи.

4.1.2 Система автентифікації та авторизації

В системі реалізована надійна система автентифікації на основі JSON Web Tokens (JWT). Вона забезпечує безпечну автентифікацію користувачів та контроль доступу до різних частин системи. Передбачено три основні ролі користувачів: студент (тип 1), викладач (тип 2) та адміністратор (тип 3). Кожна роль має свій набір прав та можливостей в системі.

Система автентифікації включає функції для входу користувачів, відновлення забутого паролю, зміни паролю та перевірки валідності токену. Особлива увага приділена безпеці: всі конфіденційні дані зберігаються з використанням безпечних практик, а доступ до критичних функцій системи суворо контролюється.

4.1.3 Безпека та обробка помилок

Особлива увага при розробці серверної частини приділена питанням безпеки та надійності роботи системи.

Використання prepared statements для SQL-запитів забезпечує захист від SQL-ін'єкцій.

Система має розвинений механізм обробки помилок, що включає автоматичне перепідключення до бази даних [1] у разі втрати з'єднання та інформативне логування помилок.

Всі критичні операції супроводжуються відповідними перевірками прав доступу та валідацією даних.

4.1.4 Комунікація та сповіщення

В системі реалізований механізм комунікації між користувачами через REST API, що забезпечує зручну взаємодію з клієнтською частиною додатку. Важливим компонентом є система email-нотифікацій, яка інформує

користувачів про важливі події: призначення тем, зміни в завданнях, відповіді на пропозиції тем тощо.

4.1.5 Реалізація API для роботи з користувачами системи

В рамках розробки серверної частини системи було реалізовано програмний інтерфейс (API) для роботи з користувачами, що забезпечує всі необхідні операції з управління користувачами та їх автентифікацією. Для створення надійного та масштабованого серверного додатку було обрано технологію Node.js у поєднанні з фреймворком Express.js. Основними технологічними компонентами реалізації стали модулі express для створення REST API endpoints, jsonwebtoken для реалізації JWT-автентифікації, nodemailer для відправки електронних листів, а також система управління базою даних MySQL.

Центральним елементом розробленого API є система автентифікації та управління безпекою. Вона включає в себе endpoint для автентифікації користувачів (/login), який перевіряє надані облікові дані та, у випадку успішної автентифікації, генерує JWT-токен з терміном дії 8 годин. Для випадків, коли користувач забув свій пароль, реалізовано механізм відновлення через електронну пошту. Авторизовані користувачі також мають можливість змінити свій пароль через відповідний endpoint.

Важливою частиною розробленого API є функціонал управління користувачами системи. Реалізовано можливість отримання списку всіх користувачів, а також окремі endpoints для отримання списків користувачів за їх типами: адміністратори, викладачі та студенти. Система підтримує додавання нових користувачів різних типів, оновлення інформації про існуючих користувачів та їх видалення. Особлива увага приділена роботі зі студентськими групами - реалізовано функціонал для управління зв'язками між студентами та їх навчальними групами, включаючи можливість зміни групи для студента.

Безпека даних та надійність системи забезпечується комплексом заходів. Впроваджено middleware для перевірки JWT-токенів та контролю доступу на основі ролей користувачів. Система запобігає створенню дублікатів облікових записів шляхом перевірки унікальності email-адрес. Всі вхідні дані проходять валідацію перед обробкою. Конфіденційні дані, такі як паролі від поштових сервісів та ключі для генерації токенів, зберігаються у змінних середовища.

Особлива увага приділена обробці помилок та виключних ситуацій. Система здійснює перевірку існування користувача при автентифікації, валідацію даних при створенні та оновленні користувачів, перевірку унікальності email- адрес та обробку помилок бази даних. Кожна помилка супроводжується відповідним HTTP-статусом та інформативним повідомленням, що полегшує діагностику проблем та покращує користувацький досвід.

Взаємодія з API відбувається через стандартні HTTP-методи. Наприклад, автентифікація користувача здійснюється через POST-запит до endpoint /login з передачею email та password у тілі запиту. Додавання нового користувача виконується через POST-запит до endpoint /add з передачею необхідних даних у форматі JSON, включаючи ім'я, email, контактний номер, тип користувача та пароль.

Таким чином, розроблений API модуль забезпечує повноцінну роботу з користувачами системи, надаючи весь необхідний функціонал для управління обліковими записами та автентифікацією.

Реалізовані механізми безпеки та обробки помилок забезпечують надійність та захищеність системи, а продумана архітектура дозволяє легко розширювати функціональність при подальшому розвитку проекту.

4.1.6 Реалізація API для роботи із завданнями в системі

В рамках розробки серверної частини системи було реалізовано програмний інтерфейс для управління завданнями магістерської дисертації. API

розроблено з використанням технології Node.js та фреймворку Express.js, що забезпечує надійну та ефективну обробку запитів, пов'язаних з управлінням завданнями студентів та адміністраторів системи.

Система управління завданнями розділена на два основні компоненти: управління індивідуальними завданнями студентів та управління загальними завданнями від адміністраторів. Для кожного з цих компонентів реалізовано повний набір CRUD-операцій (Create, Read, Update, Delete), що дозволяє гнучко керувати всіма аспектами завдань у системі. Важливою особливістю є те, що всі операції захищені механізмом автентифікації через JWT-токени, що забезпечує належний рівень безпеки при роботі з даними.

Для роботи з індивідуальними завданнями студентів реалізовано спеціалізований endpoint, який дозволяє отримувати список завдань конкретного студента. Запит здійснюється через POST метод з передачею логіну студента. Система автоматично зіставляє дані через зв'язки між таблицями users, diploma practice та tasks, повертаючи повну інформацію про завдання, включаючи дати початку та закінчення, назву завдання та його статус. Цей механізм забезпечує персоналізований доступ кожного студента до своїх завдань.

Особлива увага приділена функціоналу управління статусом завдань. Реалізовано спеціальний endpoint для зміни статусу завдання, що дозволяє відстежувати прогрес виконання кожного завдання. Це особливо важливо для моніторингу успішності проходження практики та своєчасного виявлення потенційних проблем. Система зберігає статус у форматі булевого значення, що дозволяє чітко визначати, чи виконано завдання.

Для адміністративної частини системи розроблено окремий набір endpoints для роботи з загальними завданнями. Ці завдання зберігаються в окремій таблиці tasks_from_admins і можуть бути доступні всім користувачам системи з відповідними правами. Реалізовано можливість додавання нових загальних завдань, їх редагування та видалення, що дозволяє адміністраторам

гнучко керувати навчальним процесом.

При розробці API особлива увага була приділена обробці помилок та валідації даних. Кожен endpoint повертає відповідні HTTP-статуси та інформативні повідомлення про результати операцій. У випадку успішного виконання операції повертається статус 200 та повідомлення про успіх, а при виникненні помилок - статус 500 та детальна інформація про помилку, що спрощує процес відлагодження та покращує користувацький досвід.

Структура запитів до бази даних оптимізована для ефективної роботи з пов'язаними таблицями. Використано INNER JOIN для зв'язування таблиць tasks, diploma practice та user, що дозволяє отримувати всю необхідну інформацію одним запитом. Такий підхід забезпечує високу продуктивність системи навіть при роботі з великою кількістю даних.

При реалізації функціоналу оновлення завдань враховано необхідність модифікації різних параметрів: назви завдання, дат початку та закінчення. Система дозволяє гнучко змінювати ці параметри, зберігаючи при цьому цілісність даних та зв'язки між різними частинами системи. Всі операції оновлення також захищені механізмом автентифікації, що запобігає несанкціонованій модифікації даних.

Таким чином, розроблений API модуль для роботи з завданнями забезпечує повний функціонал, необхідний для ефективного управління процесом магістерської дисертації. Реалізовані механізми безпеки, оптимізована структура запитів та продумана система обробки помилок роблять систему надійною та зручною у використанні як для студентів, так і для адміністраторів. адміністраторів.

4.1.7 Реалізація API для роботи із групами

В рамках розробки системи було реалізовано програмний інтерфейс для управління групами студентів. На відміну від попередніх модулів, цей компонент системи орієнтований виключно на адміністративний персонал, що

відображається в його архітектурі та системі прав доступу.

Ключовою особливістю даного модуля є його роль у забезпеченні організаційної структури навчального процесу. Групи є базовими одиницями для групування студентів, тому особлива увага приділена забезпеченню цілісності даних та запобіганню створення дублікатів у системі.

Реалізована система перевірки унікальності назв спеціальностей працює як при створенні нових записів, так і при оновленні існуючих. Це гарантує, що в системі не може бути двох спеціальностей з однаковою назвою, що важливо для коректної роботи інших модулів системи, зокрема модуля управління студентами та завданнями.

На відміну від інших компонентів системи, цей модуль використовує прямі запити до єдиної таблиці specialty, що спрощує структуру запитів та підвищує швидкість їх виконання. Це рішення обґрунтоване тим, що дані про спеціальності є відносно статичними та рідко змінюються.

Для зручності адміністрування реалізовано два способи отримання даних: загальний список всіх груп та деталізована інформація про конкретну групу за її ідентифікатором. Такий підхід оптимізує кількість даних, що передаються між клієнтом та сервером, дозволяючи отримувати тільки необхідну інформацію.

При видаленні групи система не перевіряє наявність пов'язаних записів в інших таблицях, оскільки ця логіка реалізована на рівні бази даних через зовнішні ключі. Це забезпечує додатковий рівень захисту цілісності даних та запобігає випадковому видаленню спеціальностей, які використовуються в системі.

4.1.8 Реалізація API для роботи із документообігом в системі

У рамках розробки системи документообігу було реалізовано REST API на базі Express.js [11] для управління документами та файлами. Дана частина системи відповідає за обробку запитів, пов'язаних із завантаженням,

отриманням та видаленням файлів різних типів, а також забезпечує необхідний рівень безпеки та автентифікації користувачів.

Особлива увага при розробці була приділена роботі з файлами та документами. Система використовує бібліотеку multer для обробки завантажених файлів, що дозволяє ефективно управляти процесом завантаження та зберігання документів. Реалізовано строгу валідацію типів файлів, що допускає лише документи формату Microsoft Word, а також встановлено обмеження на розмір файлів до 10 мегабайт, що запобігає перевантаженню серверних ресурсів.

Система надає широкий спектр функціональних можливостей для роботи з документами. Користувачі можуть завантажувати різні типи документів, включаючи звіти, щоденники практики та інші навчальні матеріали. Для кожного типу документів реалізовано специфічну логіку обробки та зберігання. Файли зберігаються у базі даних у форматі BLOB, що забезпечує їх надійне зберігання та швидкий доступ до них.

Адміністративний функціонал системи включає можливості для управління документами на рівні всієї системи. Адміністратори мають доступ до розширеного набору операцій, включаючи перегляд всіх завантажених файлів, керування правами доступу та видалення документів. Реалізовано також систему розмежування прав доступу, що дозволяє гнучко налаштовувати можливості різних категорій користувачів.

4.1.9 Реалізація API для роботи із напрямками дипломного проектування

В рамках розробки інформаційної системи було створено спеціалізований модуль для управління напрямками дипломних робіт, який забезпечує взаємодію між студентами, викладачами та адміністраторами системи. Даний модуль реалізований як окремий маршрутизатор Express.js та інтегрований в загальну структуру API системи.

Система надає різні рівні доступу до інформації про напрямки дипломних робіт. Адміністратори мають повний доступ до управління даними через endpoint'и '/get', '/add', '/update', '/delete' та '/getById', що дозволяє їм здійснювати повний контроль над напрямками дипломних робіт. Студенти мають обмежений доступ через спеціалізований endpoint '/getForStudent', який надає інформацію тільки про напрямки, доступні для їх спеціальності. Викладачі отримують доступ до інформації про напрямки через endpoint '/getDirectionForTeacher', що дозволяє їм бачити тільки ті напрямки, де вони призначені керівниками.

Реалізовано повний набір CRUD операцій, що дозволяє ефективно управляти даними про напрямки дипломних робіт. Операції створення та оновлення включають валідацію вхідних даних та перевірку прав доступу, що забезпечує цілісність даних та безпеку системи. При видаленні напрямків передбачено механізми перевірки зв'язків з іншими сутностями системи для запобігання порушення цілісності даних.

Архітектурне рішення передбачає можливість подальшого розширення функціоналу. Модульна структура коду та використання middleware дозволяють легко додавати нові можливості та модифікувати існуючі без необхідності значних змін в архітектурі системи. Це особливо важливо для адаптації системи до змінюваних вимог навчального процесу.

4.1.10 Реалізація API для роботи із обміну повідомлень та сповіщень

В рамках розробки інформаційної системи було реалізовано комплексний модуль комунікації, який забезпечує обмін повідомленнями між учасниками дипломного процесу та систему email-сповіщень. Даний модуль інтегрує функціонал внутрішнього чату з можливістю надсилання email-повідомлень, використовуючи сервіс Gmail через бібліотеку nodemailer.

Система комунікації побудована на основі структурованої архітектури бази даних, де повідомлення зберігаються в контексті конкретних бесід

(conversations), пов'язаних з дипломними практиками.

Особливу увагу приділено інтеграції email-сповіщень. Система автоматично надсилає повідомлення на електронну пошту при кожній новій взаємодії в чаті.

Архітектурне рішення передбачає можливість масштабування системи комунікації. Модульна структура дозволяє легко додавати нові канали комунікації та розширювати функціонал сповіщень. Особливу увагу приділено оптимізації запитів до бази даних для забезпечення швидкої роботи системи навіть при великій кількості повідомлень.

Реалізовано механізм визначення відправника та отримувача повідомлень на основі ролей користувачів у системі, що забезпечує коректну маршрутизацію повідомлень та email-сповіщень. Система автоматично визначає, хто є відправником та отримувачем на основі даних про дипломну практику та надсилає повідомлення відповідній стороні.

4.1.11 Реалізація API для роботи із інформаційною панелі статистики

Серверна стороні. В рамках розробки інформаційної системи реалізовано модуль аналітики, який забезпечує збір та надання статистичної інформації для різних категорій користувачів системи. Даний модуль є критично важливим компонентом для моніторингу ефективності дипломного процесу та прийняття управлінських рішень на основі актуальних даних.

Для адміністративного персоналу реалізовано комплексний endpoint '/info', який агрегує ключові показники системи через серію послідовних запитів до бази даних. Система збирає та надає інформацію про загальну кількість зареєстрованих спеціальностей, напрямків дипломних робіт та активних дипломних практик. Ця інформація дозволяє адміністраторам оцінювати масштаб роботи системи та планувати її розвиток.

Студентський інтерфейс аналітики, реалізований через endpoint '/infoForStudent', надає персоналізовану статистику щодо прогресу виконання

дипломної роботи. Система автоматично підраховує загальну кількість призначених завдань та кількість успішно виконаних задач. Використання складних SQL-запитів з JOIN операціями дозволяє ефективно збирати дані з пов'язаних таблиць tasks, diploma practice та user, забезпечуючи точність та актуальність інформації.

Для викладацького складу розроблено спеціалізований endpoint '/infoForTeacher', який надає детальну статистику щодо керованих дипломних робіт. Система відстежує кількість вільних тем, які ще не обрані студентами, та загальну кількість активних дипломних практик під керівництвом конкретного викладача. Реалізовані запити враховують складну структуру зв'язків між таблицями diploma practice, directionofthesis, specialty та user.

Особливу увагу в системі приділено оптимізації запитів до бази даних. Використання агрегатних функцій COUNT та складних JOIN операцій дозволяє отримувати необхідну статистику з мінімальним навантаженням на сервер. Кожен запит оптимізований для роботи з великими обсягами даних та включає необхідні умови фільтрації для забезпечення точності результатів.

4.1.12 Реалізація API для управління темами дипломного проектування та звітності

Основним компонентом системи є модуль управління темами дипломних практик. Він надає можливість викладачам створювати та публікувати теми для дипломних робіт, а студентам - переглядати доступні теми та обирати їх для своєї практики. Система також передбачає можливість скасування вибору теми як з боку викладача, так і з боку студента, що забезпечує необхідну гнучкість у процесі розподілу тем.

Важливою особливістю системи є реалізований механізм пропозиції тем студентами. Студенти можуть подавати власні пропозиції тем дипломних робіт, які розглядаються викладачами. Цей функціонал забезпечує більшу залученість студентів до процесу вибору напрямку своєї дипломної роботи та дозволяє

врахувати їхні особисті інтереси та преференції.

Для забезпечення ефективної комунікації між учасниками процесу в системі реалізовано автоматичне сповіщення через електронну пошту. Система надсилає повідомлення про важливі події: вибір теми студентом, затвердження або відхилення запропонованої теми, скасування закріплення теми за студентом. Це дозволяє всім учасникам процесу бути вчасно проінформованими про важливі зміни та події.

Модуль звітності є важливою складовою системи, що забезпечує можливість генерації різноманітних звітів у форматі Excel. Реалізовано можливість формування звітів про розподіл тем між студентами, списків студентів без обраних тем та інших необхідних для адміністрування процесу документів. Цей функціонал значно спрощує процес моніторингу та управління процесом дипломного проектування.

Таким чином, розроблена серверна частина системи забезпечує надійну та безпечну роботу всіх компонентів системи керування дипломною практикою, надаючи необхідний функціонал для всіх категорій користувачів. Модульна архітектура та використання сучасних технологій дозволяють легко масштабувати та підтримувати систему, а також розширювати її функціональність у майбутньому.

4.2 Опис клієнтської частини системи

У даному підрозділі описується розробка клієнтської частини системи системи створення етапів дипломного проектування. Клієнтська частина проєкту була реалізована за допомогою Angular — сучасного фреймворка для побудови односторінкових додатків (SPA).

Використовуючи компонентну архітектуру, Angular дозволяє організувати код в незалежні частини, які легко підтримувати, повторно використовувати та тестувати. Основним інструментом для роботи з даними

став Angular HTTP Client, який забезпечує взаємодію з сервером через API, надсилаючи запити і обробляючи відповіді в асинхронному режимі.

Для керування навігацією в додатку був використаний Angular Router, що забезпечує швидке перемикання між сторінками без перезавантаження браузера, зберігаючи плавний користувацький досвід. Важливою частиною також стало використання Angular Material для створення адаптивного і сучасного інтерфейсу, що забезпечує зручність та інтуїтивність для кінцевого користувача.

Загальна модель архітектури серверної частини зображена на рисунку 4.3.

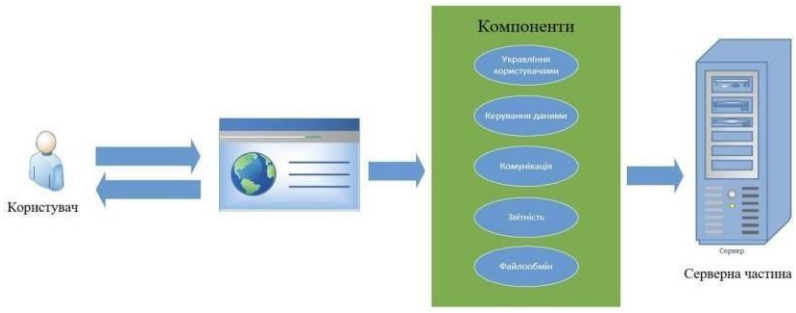

Рисунок 4.3 – Модель архітектури клієнтської частини

Аналізуючи структуру Angular компонентів у представленому проєкті, можна виділити типові архітектурні рішення та паттерни, які використовуються в системі управління дипломними роботами.

Компоненти побудовані за принципом Single Responsibility Principle, де кожен відповідає за конкретну функціональність системи. Типова структура включає компоненти для управління користувачами, групами, темами, оцінками та іншими сутностями системи. Кожен такий компонент реалізує стандартний набір CRUD операцій через уніфікований інтерфейс.

Архітектура системи базується на компонентному підході Angular, де кожен модуль містить набір взаємопов'язаних компонентів. Широко використовується Material Design через Angular Material, що забезпечує уніфікований зовнішній вигляд та поведінку елементів інтерфейсу. Типові елементи включають таблиці (MatTableDataSource), діалогові вікна (MatDialog),

форми та елементи навігації.

Система автентифікації та авторизації реалізована через JWT токени, що зберігаються в localStorage. Кожен компонент при ініціалізації перевіряє права доступу користувача та відповідно адаптує свій інтерфейс. Це забезпечує персоналізований доступ до функціоналу системи залежно від ролі користувача (студент, викладач, адміністратор).

Взаємодія з backend реалізована через сервіси, які інкапсулюють логіку HTTP-запитів. Кожен такий сервіс відповідає за роботу з конкретною сутністю системи. Сервіси використовують TypeScript інтерфейси для типізації даних, що забезпечує надійність та зручність розробки.

Обробка помилок централізована через спеціальний сервіс сповіщень (ScnackbarService), який відповідає за відображення повідомлень користувачу. Всі компоненти використовують єдиний підхід до обробки та відображення помилок, що забезпечує консистентний користувацький досвід.

Форми в системі реалізовані з використанням як Template-driven, так і Reactive forms підходів Angular. Кожна форма включає валідацію даних, відображення помилок та асинхронну взаємодію з сервером. Модальні вікна (діалоги) використовуються для створення та редагування даних, забезпечуючи зручний інтерфейс користувача.

Система маршрутизації побудована ієрархічно, де кожен модуль має свої дочірні маршрути. Це дозволяє організувати навігацію по системі логічно та забезпечити lazy loading модулів для оптимізації продуктивності.

Стан додатку управляється через комбінацію сервісів та RxJS Observables. Компоненти підписуються на зміни даних та оновлюють свій стан реактивно, що забезпечує ефективну синхронізацію даних між різними частинами додатку.

Компоненти активно використовують життєвий цикл Angular для управління ресурсами. Ініціалізація даних відбувається в ngOnInit, а очищення ресурсів (відписка від Observable) в ngOnDestroy, що запобігає витокам пам'яті.

Система підтримує інтернаціоналізацію через сервіс перекладів, де всі

текстові рядки винесені в окремі файли локалізації. Це спрощує додавання нових мов та підтримку існуючих перекладів.

Стилізація компонентів реалізована через SCSS з використанням змінних та міксинів. Кожен компонент має власні стилі, але використовує глобальні змінні для забезпечення консистентного вигляду. Material Design теми можуть бути кастомізовані під потреби проєкту.

Оптимізація продуктивності досягається через використання ChangeDetectionStrategy.OnPush, lazy loading модулів, віртуальний скролінг для довгих списків та кешування даних на клієнті. Це забезпечує швидку роботу системи навіть при великій кількості даних.

Тестування компонентів реалізоване через юніт-тести з використанням Jasmine та Karma. Кожен компонент має набір тестів, що перевіряють його функціональність в ізоляції від інших частин системи.

Система побудована з урахуванням можливості розширення. Нові функції можуть бути додані через створення нових компонентів або розширення існуючих. Модульна архітектура дозволяє легко інтегрувати нові можливості без значних змін в існуючому коді.

Компоненти активно використовують шаблони проєктування, такі як Dependency Injection, Observer, Factory та інші. Це забезпечує гнучкість архітектури та можливість легкого внесення змін у майбутньому.

Весь код написаний з дотриманням стилю програмування Angular та TypeScript, використовуючи строгу типізацію та лінтери для забезпечення якості коду. Документація генерується автоматично з коментарів у форматі JSDoc.

4.3 Опис бази даних системи

База даних включає в собі 13 таблиць для реалізації цієї системи, а саме: «Тип файлу», «Файли», «Тип користувача», «Користувачі», «Участники груп»,

«Групи», «Напрямки», «Дипломне проектування», «Запропоновані теми», «Повідомлення», «Чати», «Загальні етапи», «Завдання». Логічна модель подання бази даних зображена на рисунку 4.4.

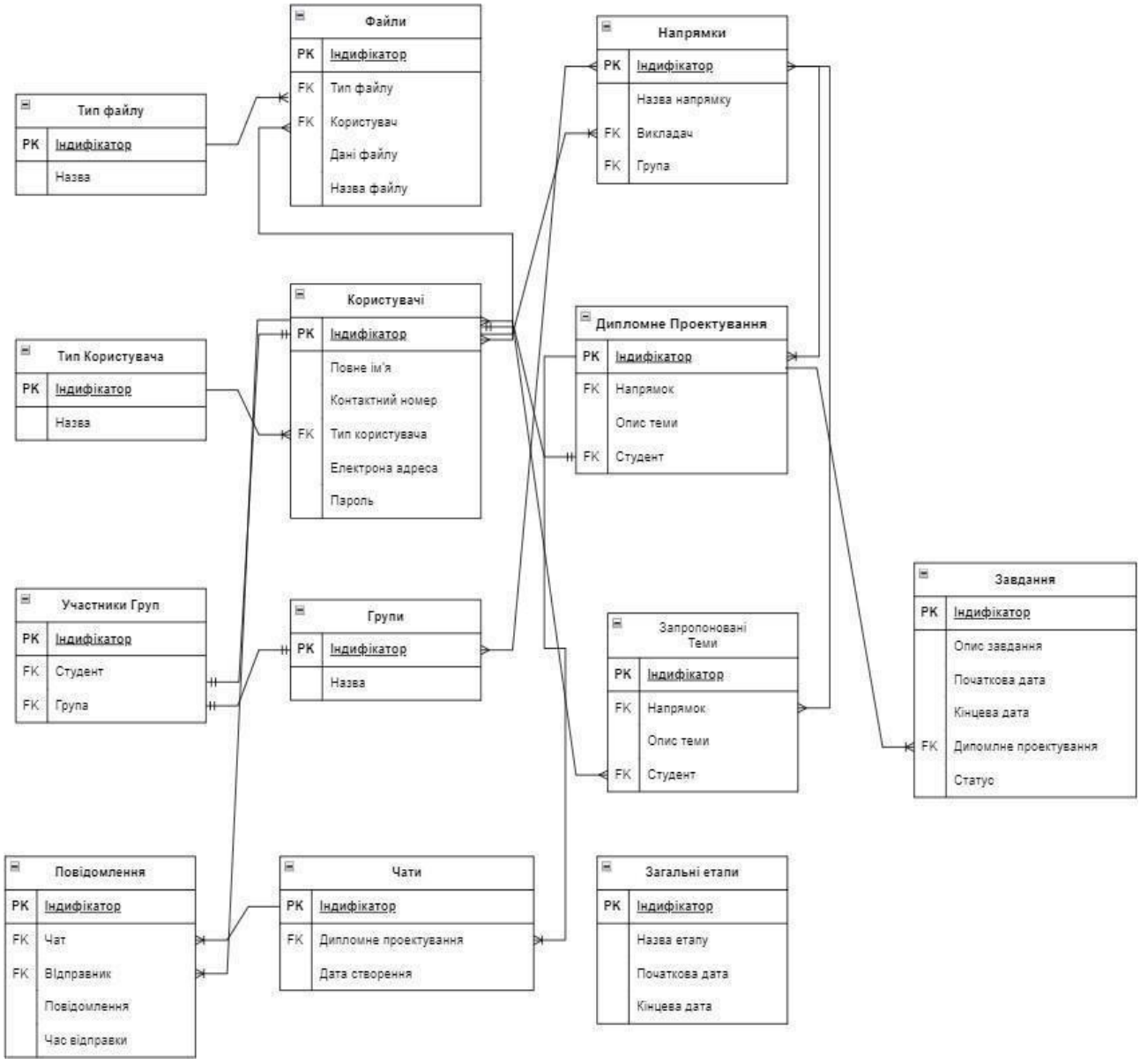

Рисунок 4.4 – Логічна модель бази даних

Представлена схема бази даних відображає структуру системи створення етапів дипломного проектування, що складається з декількох взаємопов'язаних компонентів. Також для більш чіткого розуміння ви можете переглянути ER - діаграму бази даних, яка зображена на рисунку 4.5.

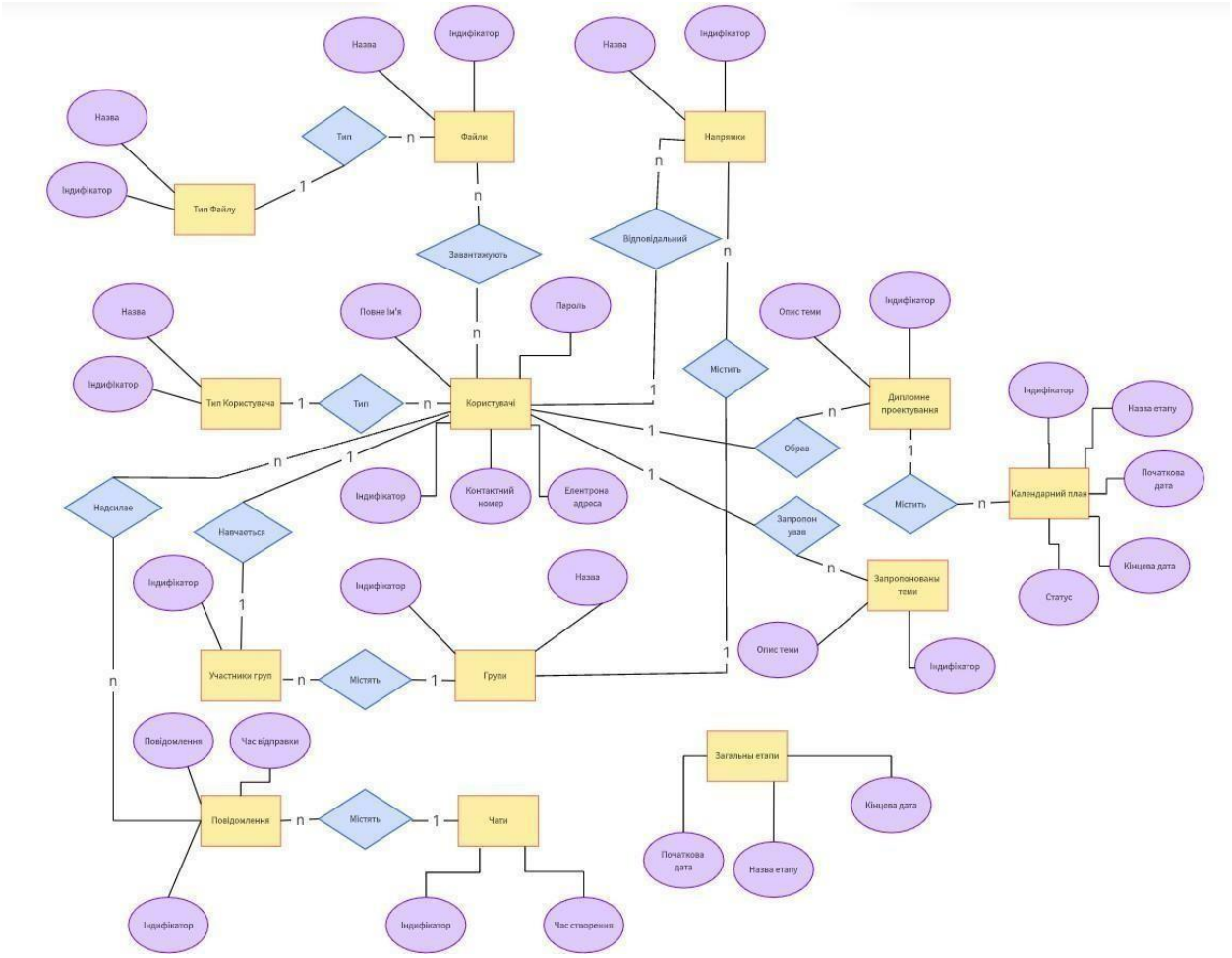

Рисунок 4.5 – ER - діаграма бази даних

В основі системи лежить таблиця "Користувачі", яка містить основну інформацію про учасників системи: ідентифікатор, повне ім'я, контактний номер, тип користувача, електронну адресу та пароІль. Тип користувача визначається через зв'язок з довідниковою таблицею "Тип Користувача", що дозволяє гнучко управляти ролями в системі.

Організаційна структура представлена таблицями "Групи" та "Учасники Груп". Таблиця "Групи" зберігає інформацію про навчальні групи, а "Учасники Груп" забезпечує зв'язок між студентами та їхніми групами через відповідні зовнішні ключі.

Центральним елементом схеми є таблиця "Дипломне Проектування", яка пов'язує напрямки дипломних робіт зі студентами. Вона містить інформацію

про обрані теми та їх описи. Таблиця "Напрямки" зберігає доступні напрямки дипломного проектування, включаючи інформацію про викладача та групу.

Для забезпечення можливості попереднього узгодження тем реалізована таблиця "Запропоновані теми", яка містить пропозиції тем від студентів з прив'язкою до напрямку та самого студента.

Система комунікації представлена таблицями "Чати", "Повідомлення" та "Файли". "Чати" створюються для кожного дипломного проектування, "Повідомлення" зберігають історію спілкування між учасниками, а "Файли" забезпечують можливість обміну документами. Структура файлової системи додатково деталізована таблицею "Тип файлу".

Для відстеження прогресу реалізовані таблиці "Завдання" та "Загальні етапи". "Завдання" містять конкретні задачі з термінами виконання та статусом, а "Загальні етапи" визначають основні віхи дипломного проектування з відповідними датами початку та завершення.

Всі таблиці мають первинні ключі (PK) для унікальної ідентифікації записів та необхідні зовнішні ключі (FK) для встановлення зв'язків між сутностями. Така структура забезпечує цілісність даних та дозволяє ефективно управляти всіма аспектами процесу дипломного проектування.

Схема бази даних розроблена з урахуванням принципів нормалізації, що мінімізує надлишковість даних та забезпечує їх узгодженість.

Використання зовнішніх ключів гарантує референційну цілісність та коректність зв'язків між різними сутностями системи. майбутньому. JSDoc.

4.4 Робота користувача з розробленою системою

Як тільки користувач відкрив веб-додаток, першим чином його зустрічає стартова сторінка. На ній ми можемо побачити опис основних функцій цієї системи та кнопки навігації, а саме «Надіслати пароль» та «Авторизуватися», як це зображено на рисунку 4.6.

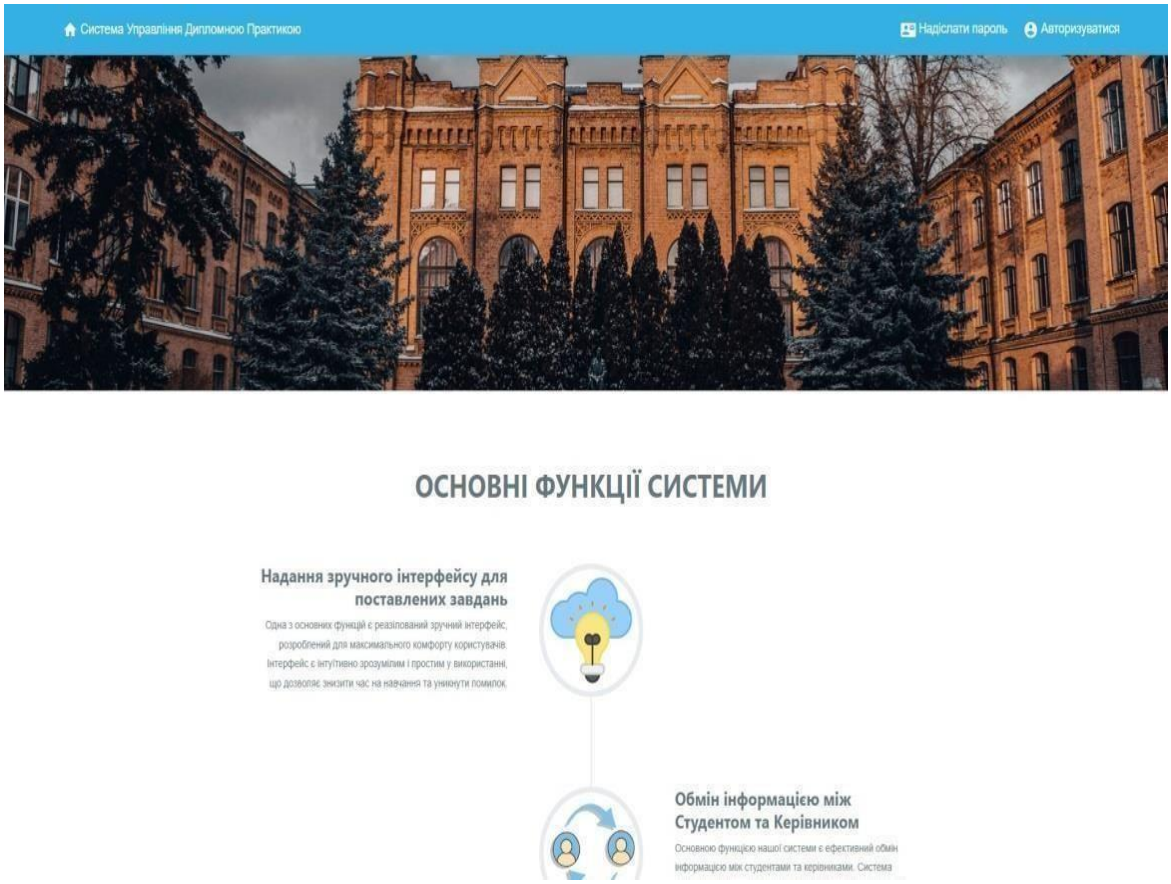

Рисунок 4.6 – Стартова сторінка

Якщо користувач має обліковий запис зі своїми даними, то в ситуації коли він забувся пароль – він має можливість автоматично, за допомогою реалізованої функції системи, надіслати його собі на електрону пошту, яка була підв’язана до нього. Для цього потрібно натиснути кнопку «Надіслати пароль» та вести адресу своєї електронної пошти, як це зображено на рисунку 4.7.

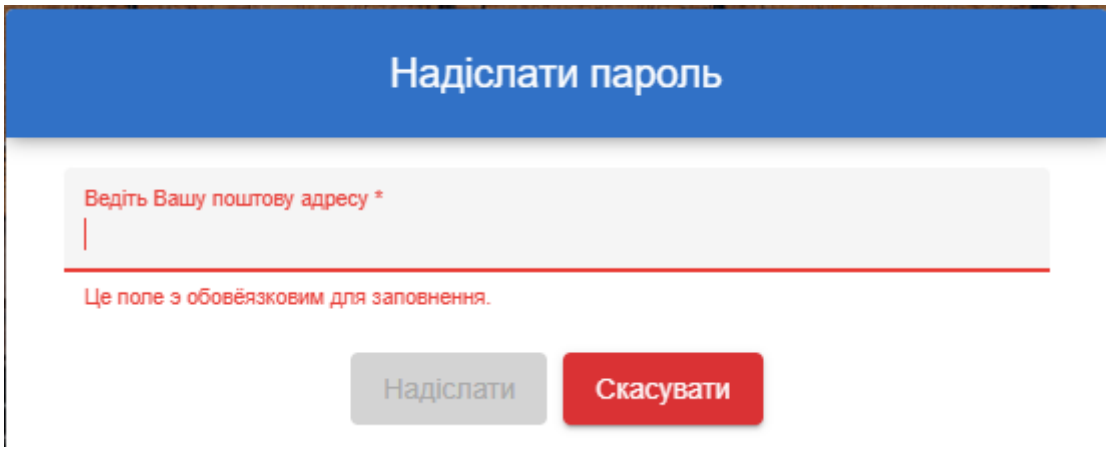

Рисунок 4.7 – Модальне вікно для надсилання паролю

Коли користувач впевнений, що він зареєстрований в системі – він звісно може авторизуватися. Для цього потрібно натиснути кнопку «Авторизуватися» та вести свою електронну пошту та пароль, як це зображено на рисунку 4.7.

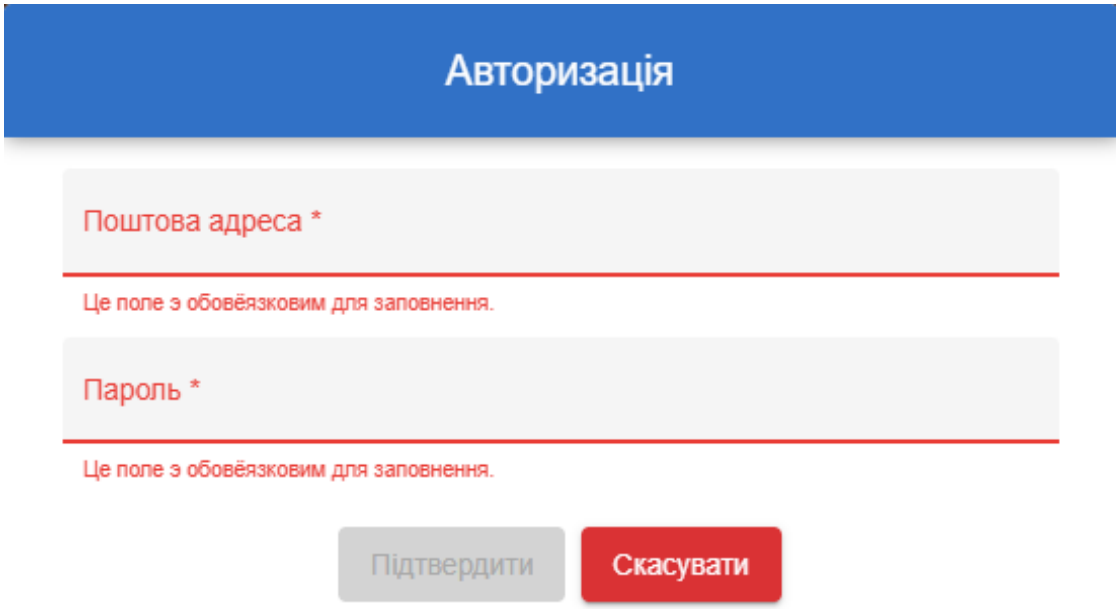

Рисунок 4.7 – Модальне вікно для авторизації

Після авторизації, в залежності від типу облікового запису користувача, він буде бачити різне наповнення навігаційного меню, Тобто якщо це адміністратор системи – то в нього буде можливість додавати дані і т.д, а якщо це студент або викладач, то він зможе виконувати певні специфічні дії.

4.4.1. Можливості адміністраторів в системі

Розпочнемо із опису можливостей адміністраторів системи. Після того, як ви увійшли в систему під обліком адміністратора - вас буде зустрічати

«Інформаційна панель» на ній відображено підрахунок ключових даних. По праву сторону розташована кнопка у вигляді іконки акаунта користувача, а по ліву сторону основне меню, як це зображено на рисунку 4.9.

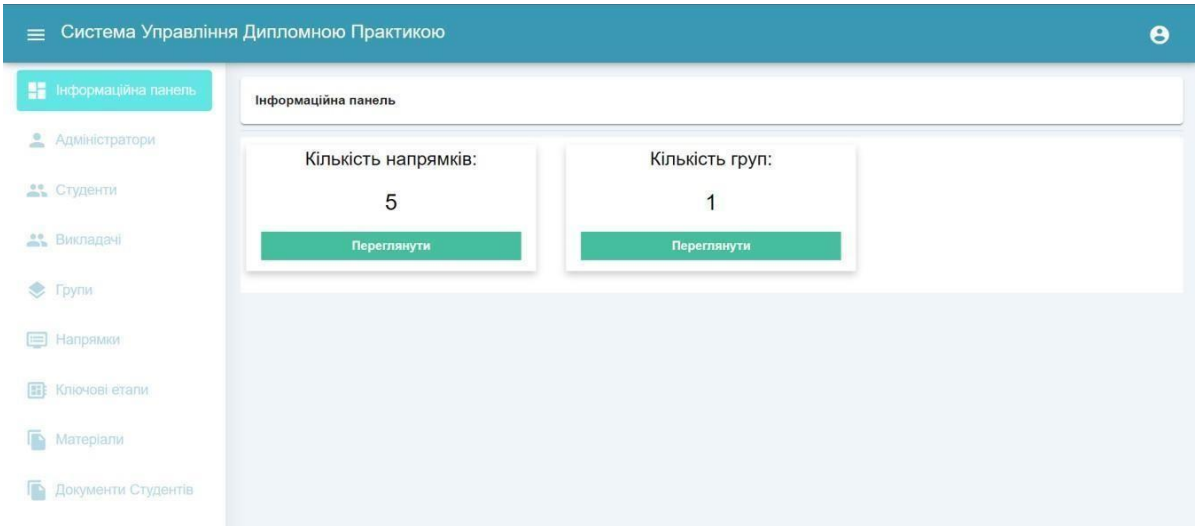

Рисунок 4.9 – Інтерфейс системи доступний для адміністраторів

Перш за все – це додавати нових користувачів системи, а саме адміністраторів, викладачів та студентів. Ви можете натиснути по ліву сторону «Адміністратори» та побачити дані про адміністраторів системи, як це зображено на рисунку 4.10.

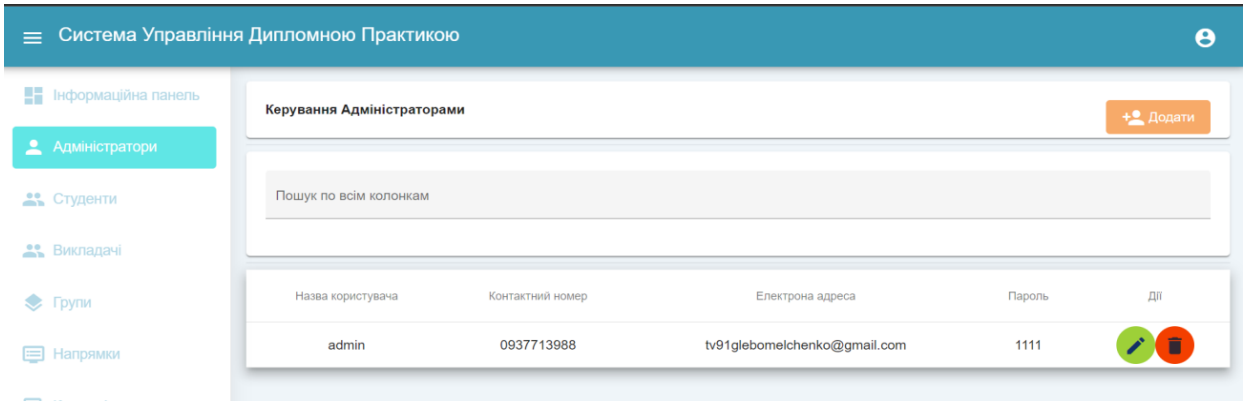

Рисунок 4.10 – Відображення адміністраторів в системі

У вас є можливість за потреби використовувати «Пошук по всім колонкам», цей пошук працює автоматично для всіх колонок які є наявні в таблиці. Також за потреби можна додати нового адміністратора натиснувши «Додати». Після чого потрібно буде вести дані нового користувача, та пройти перевірки на коректність ведених даних, як це зображено на рисунку 4.11.

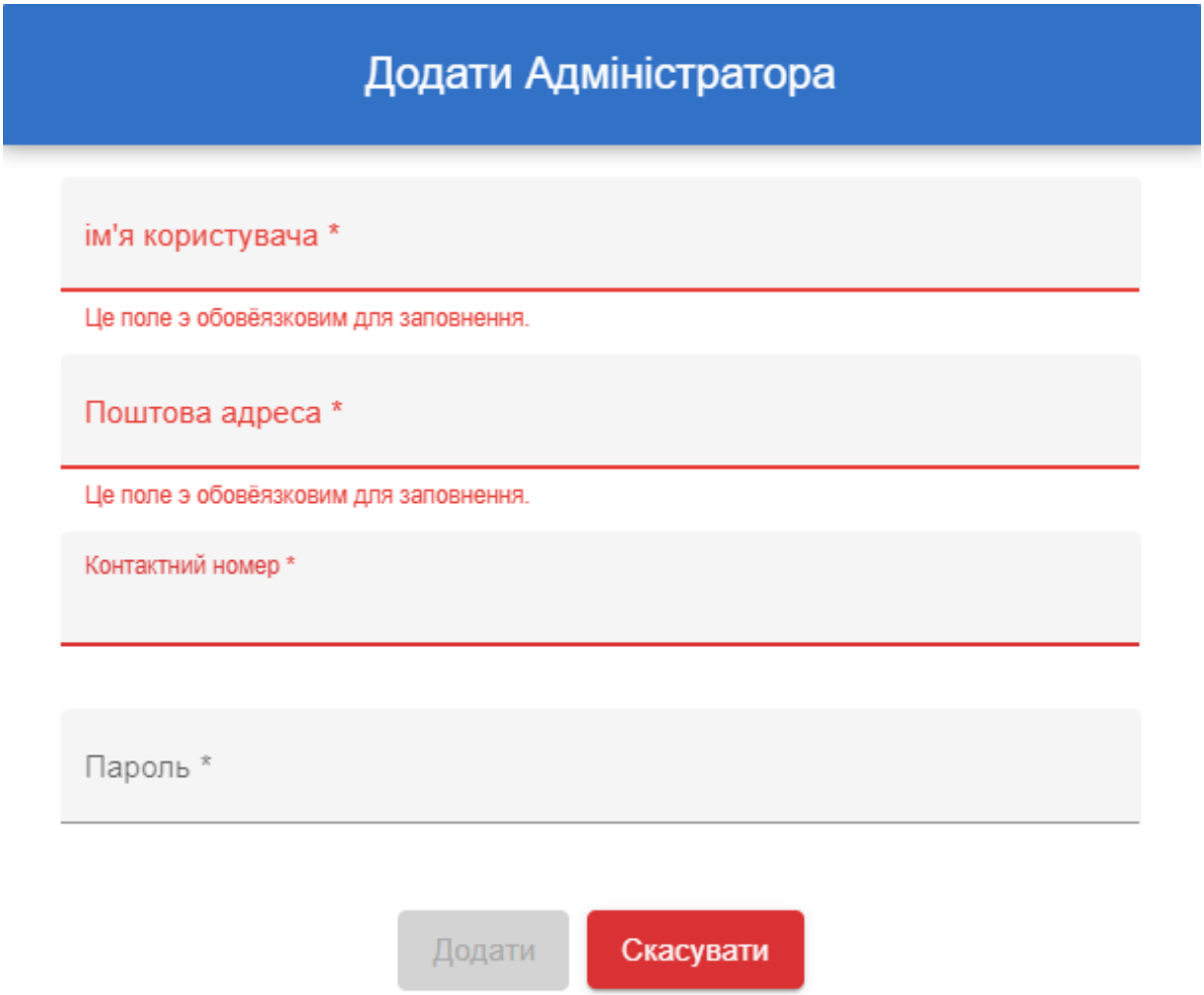

Рисунок 4.11 – Модальне вікно додавання адміністратора

Так само за потреби ви можете редагувати дані про вже наявного адміністратора, для цього потрібно натиснути на іконку редагування біля інформації потрібного вам користувача, як це зображено на рисунку 4.12, та після чого ви зможете вести нові дані, як це зображено на рисунку 4.13.

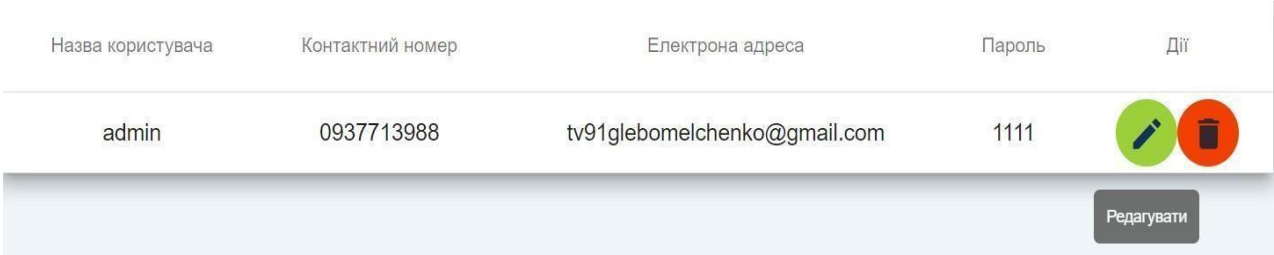

Рисунок 4.12 – Іконка редагування даних

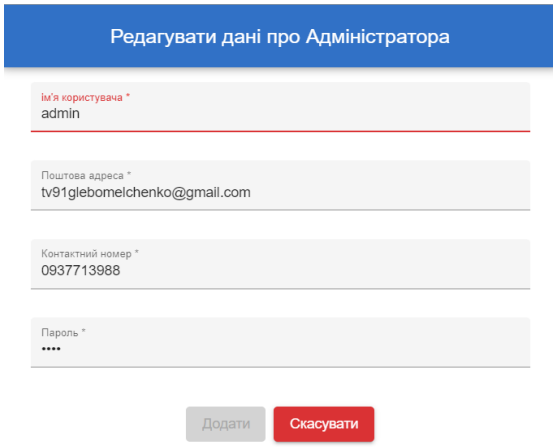

Рисунок 4.13 – Модальне вікно редагування даних адміністратора

Аналогічно до редагування даних – є можливість видалення даних, натиснувши на іконку «Видалити», як це зображено на рисунку 4.14, та після чого підтвердити цей крок або скасувати, як це зображено на рисунку 4.15.

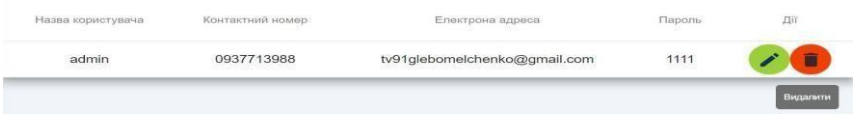

Рисунок 4.14 – Іконка видалення даних

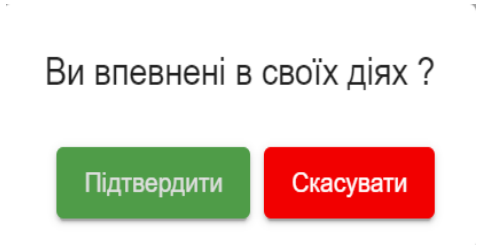

Рисунок 4.15 – Модальне вікно підтвердження

Аналогічно до можливостей та перегляду адміністраторів системи, створено такий же перегляд студентів та викладачів з усіма тими самими функціями, окрім надсилання повідомлень та під час додавання студента = вказування його групи, як це зображено на рисунку 4.16 та 4.17.

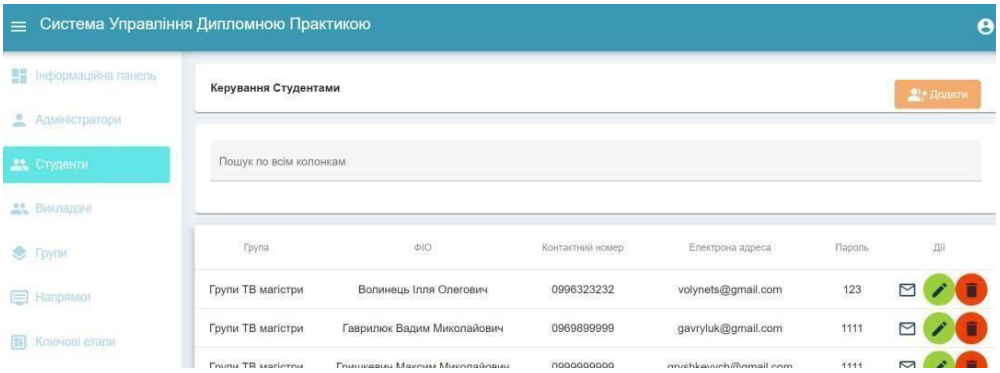

Рисунок 4.16 – Відображення студентів в системі

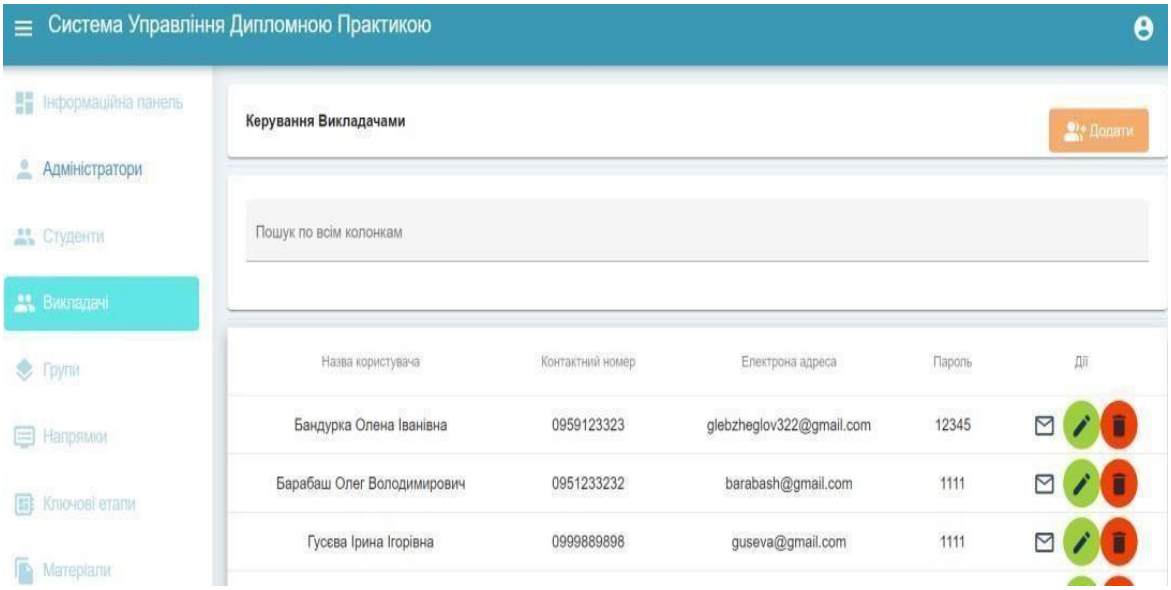

Рисунок 4.17 – Відображення викладачів в системі

Додатковою можливістю – є надсилання листів користувачам від імені системи, для цього потрібно натиснути іконку «Надіслати лист», як це зображено на рисунку 4.18. Після чого вести ваше повідомлення та надіслати, як це зображено на рисунку 4.19.

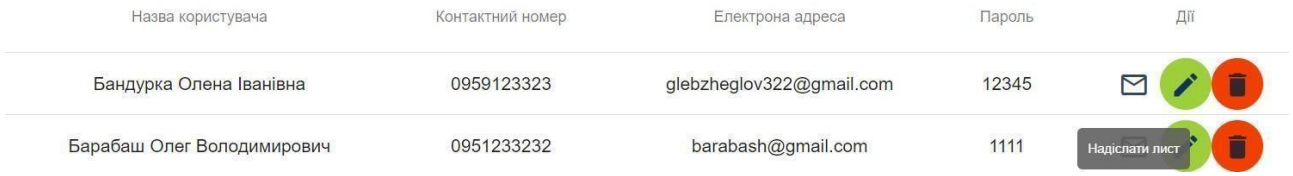

Рисунок 4.18 – Іконка надсилання листа

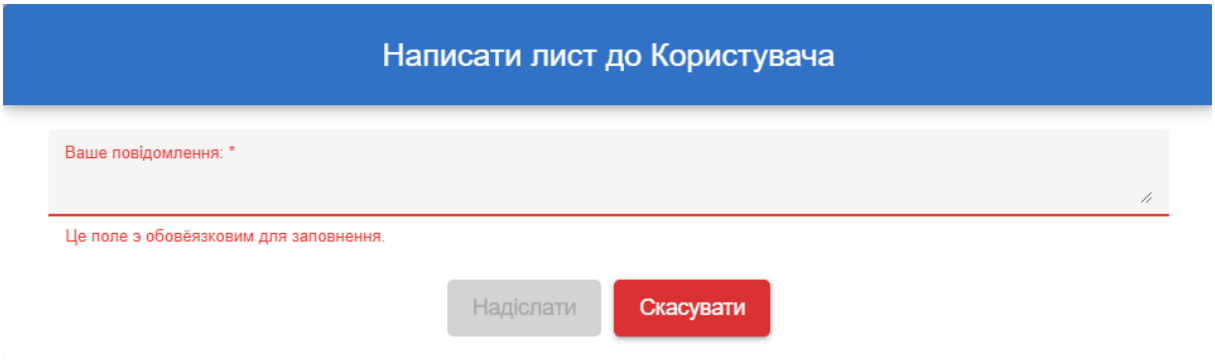

Рисунок 4.19 – Модальне вікно надсилання листа

Також однією з функцій є керування групами студентів, функції вже знайомі, це додавання їх, редагування та видалення із веденням назви групи. Відображення груп зображено на рисунку 4.20.

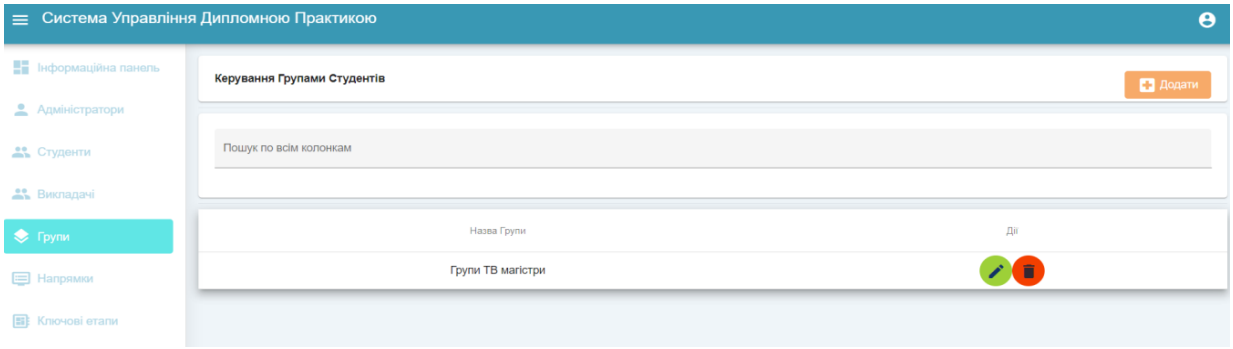

Рисунок 4.20 – Відображення груп студентів в системі

Однією із самих основних можливостей адміністратора – це створення напрямків. Відображення даних про напрямки зображено на рисунку 4.21.

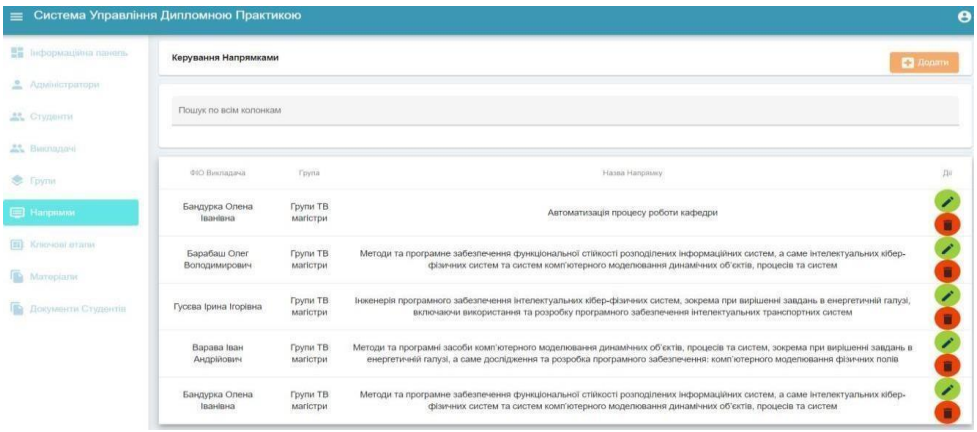

Рисунок 4.21 – Відображення даних напрямків в системі

Функції як і перед цим такі ж самі, а саме додавання, редагування та видалення даних, єдине для додавання напрямків вказується група студентів та викладач, який буде відповідальний за напрямок, ну і звісно назва напрямку.

Наступними можливостями є створення, видалення та редагування даних про «Ключові етапи» та «Матеріали».

Відображення даних про ключові етапи зображено на рисунку 4.22, серед особливостей це вказування початкової дати та кінцевої, інші функції такі ж самі. Відображення даних про ключові етапи зображено на рисунку 4.23, а ось щодо функцій для матеріалів – це можливість додавання файлів, та їх вивантаження, як зображено на рисунку 4.24.

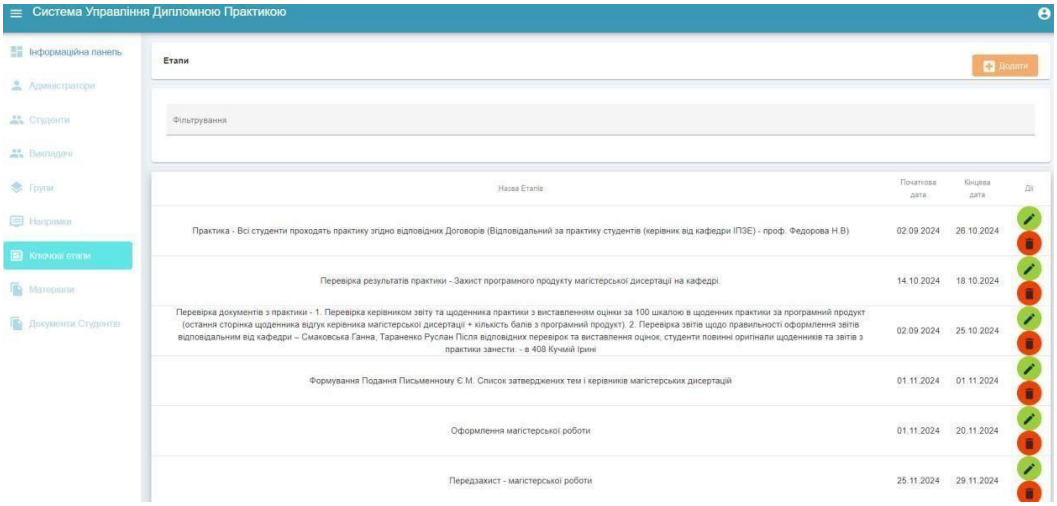

Рисунок 4.22 – Відображення даних про ключові етапи в системі

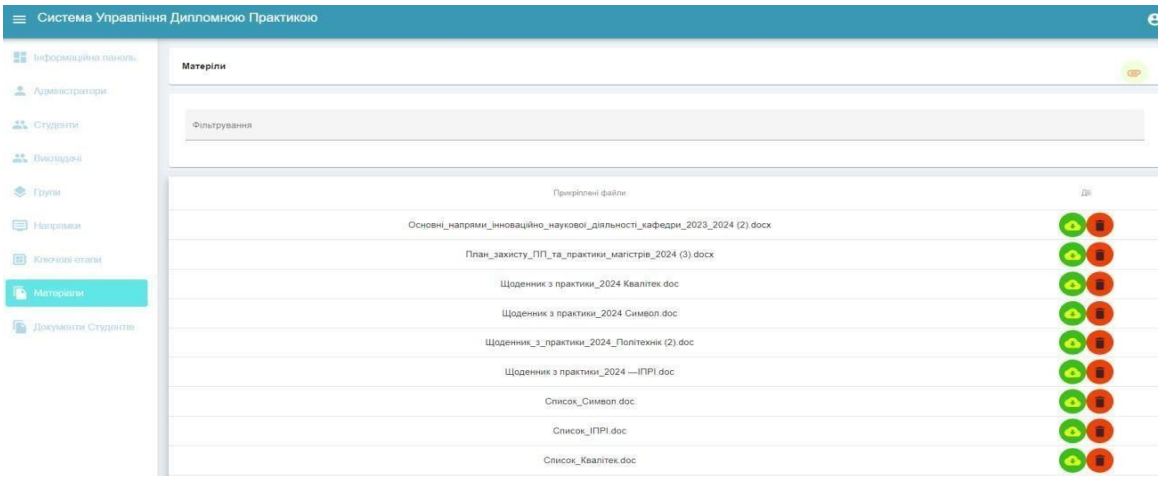

Рисунок 4.23 – Відображення даних про матеріали в системі

Для вивантаження файлів потрібно натиснути на іконку «Завантажити», після чого файл буде завантажений до вас на локальний комп’ютер.

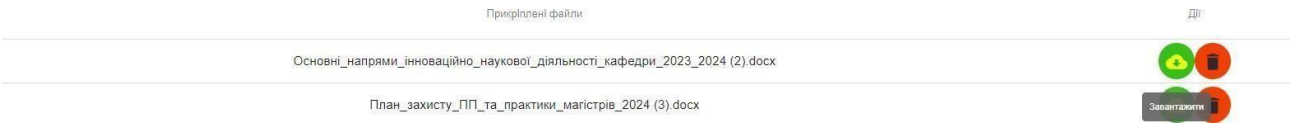

Рисунок 4.24 – Іконка завантаження файлів

Останнім пунктом перегляду даних з лівого меню – це «Документи Студентів» із можливістю пошуку їх по назві та також вивантаження собі на комп’ютер, як зображено на рисунку 4.25.

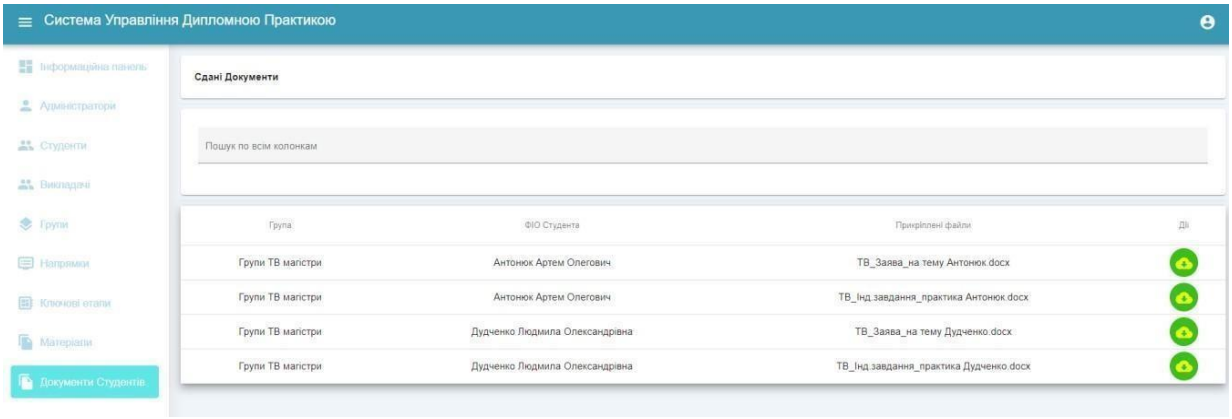

Рисунок 4.25 — Відображення даних про документи студентів в системі

Тепер перейдемо до меню справа, яке зображено на рисунку 4.26. Як ми бачимо доступно 3 функції, а саме «Змінити пароль», «Сформувати звіт» та «Вийти».

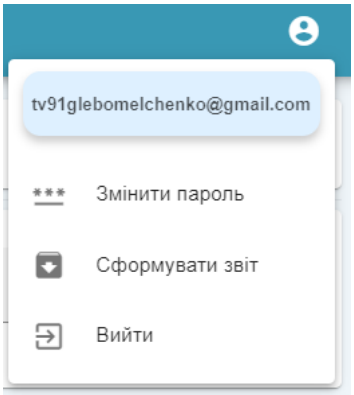

Рисунок 4.26 – Відображення меню акаунта адміністратора

Функція «Змінити пароль» потрібна для того щоб користувач при бажанні мав можливість змінити пароль від свого облікового запису. Процес змінення паролю відображено на рисунку 4.27.

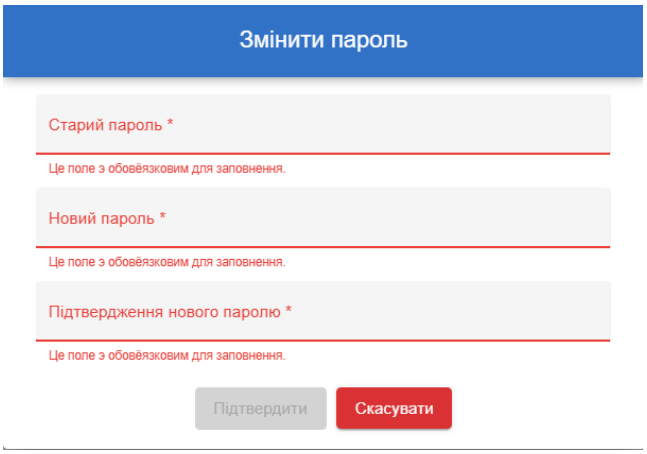

Рисунок 4.27 – Модальне вікно зміни паролю

Наступною кнопкою є «Сформувати звіт» - адміністратор має можливість сформувати 2 різних типи звіту для певної групи студентів, а саме звіт «Обрані теми», тобто базуючись на обраних тем студентів певної групи – ми можемо побачити загальну інформацію, та студенти які не обрали тему, тобто дані студентів, які цього не зробили, як це зображено на рисунку 4.28. Результатом звіту – є файл формату «.xlsx» з певними даними.

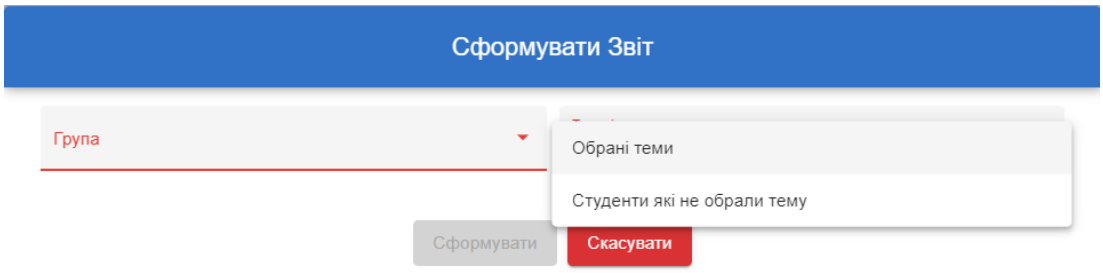

Рисунок 4.28 – Формування звіту

Останньою кнопкою є «Вийти», тобто вийти саме з облікового запису користувача із підтвердженням, як зображено на рисунку 4.26.

4.4.2 Можливості викладачів в системі

Перейдемо до опису можливостей викладачів в системі. Аналогічно як до адміністратору так і для викладача та студента – після того як ви увійшли в систему вас буде зустрічати «Інформаційна панель» на ній відображено підрахунок ключових даних. По праву сторону розташована кнопка у вигляді іконки акаунта користувача, а по ліву сторону основне меню, як це зображено на рисунку 4.29.

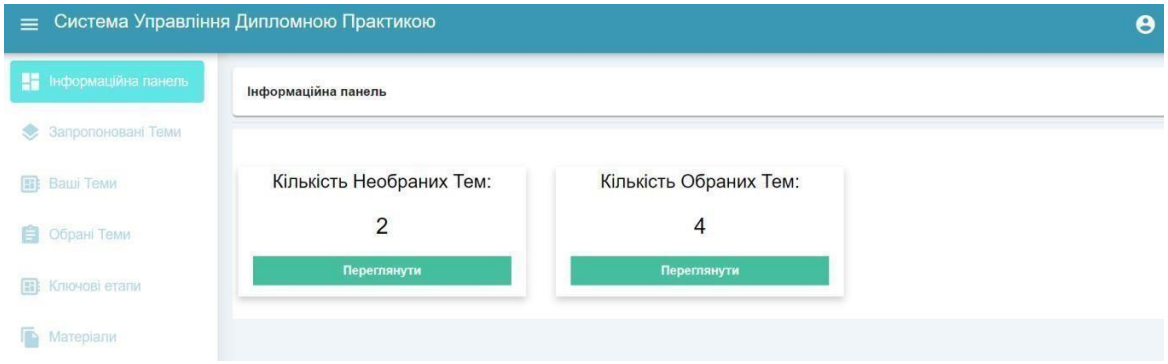

Рисунок 4.29 – Інтерфейс системи доступний для викладачів

Перейдемо до ключової можливості викладачів – це створення тем під певні напрямки студентам. Для цього потрібно обрати «Ваші Теми» по ліву сторону меню. Після чого ми можемо переглянути теми які ми створили, але їх ще ніхто не обрав, як це зображено на рисунку 4.30.

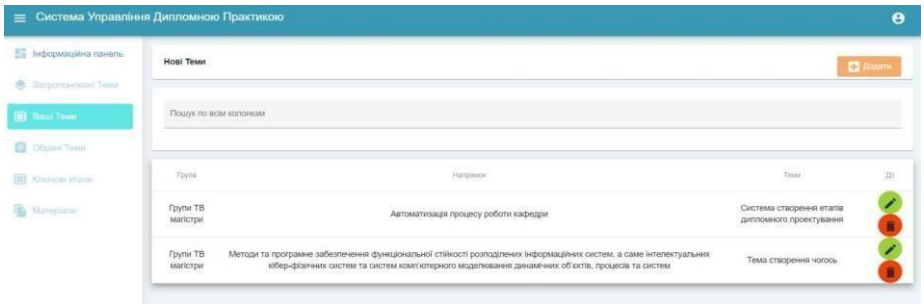

Рисунок 4.30 – Відображення тем викладача, які ще не обрані

Аналогічно для додавання потрібно натиснути кнопку «Додати» та обрати для якого напрямку під яку групу буде створена тема, як це зображено

на рисунку 4.31. Для редагування та видалення аналогічно як і перед цим є іконки «Редагувати» та «Видалити».

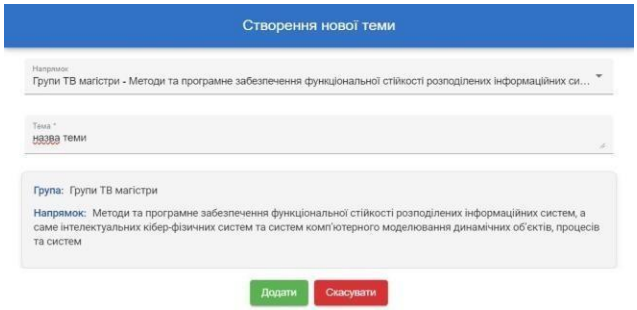

Рисунок 4.31 – Модульне вікно додавання теми

Наступною можливістю буде перегляд запропонованих тем. Для цього потрібно обрати по ліву сторону «Запропоновані теми». Після чого ви побачите теми, які студенти вирішили запропонувати викладачу під певний напрямок, як це зображено на рисунку 4.32.

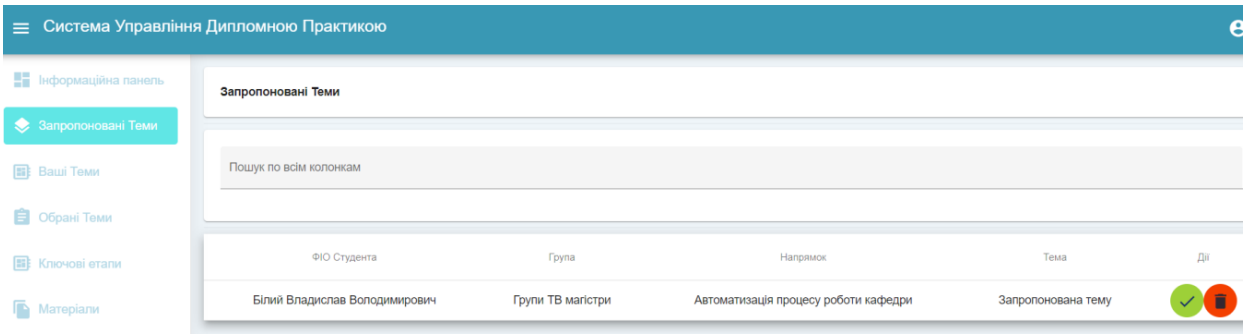

Рисунок 4.32 – Запропоновані теми студентів для викладача

Аналогічно для видалення потрібно натиснути іконку «Видалити». У випадку якщо тема вам підійшла – ви можете її підтвердити натиснувши іконку «Підтвердити», як це зображено на рисунку 4.33, та після чого у вас буде перепитано чи дійсно ви чого хочете, як це зображено на рисунку 4.15.

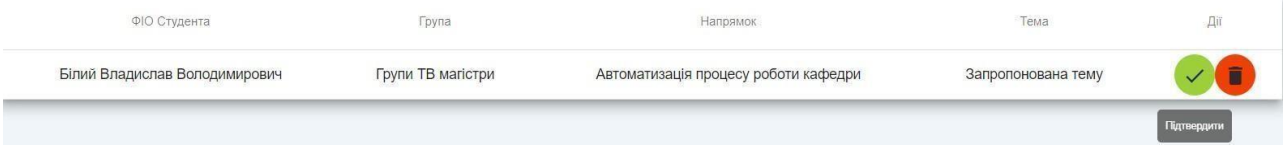

Рисунок 4.33 – Іконка підтвердження теми викладачем для студента

Також викладач має можливість переглянути «Матеріали» та «Ключові етапи», але він не може як адміністратор додавати, видаляти чи редагувати ці дані, лише перегляд, як це зображено на рисунках 4.34 та 4.35.

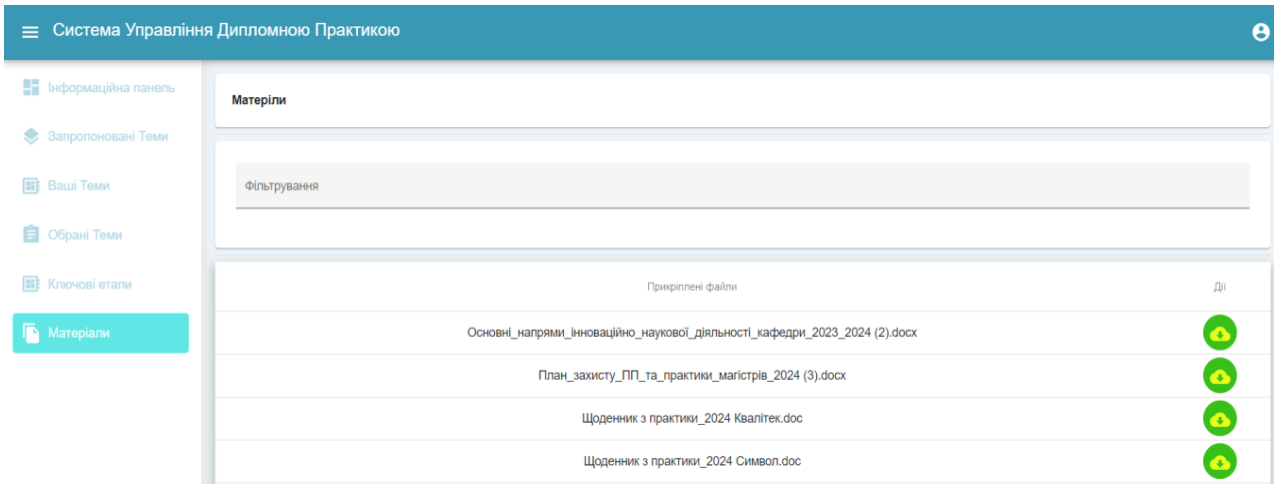

Рисунок 4.34 – Перегляд матеріалів викладачем та студентом

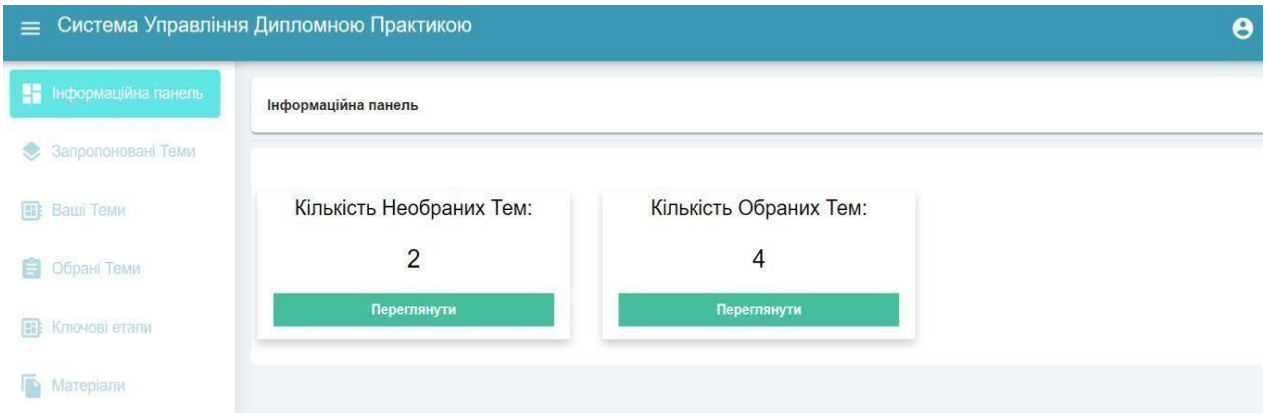

Рисунок 4.35 – Перегляд ключових етапів викладачем та студентом

Ну і нарешті це вже сам процес взаємодії викладача та студента. Викладач, має можливість переглянути обрані теми студентів, для цього потрібно натиснути по ліву сторону «Обрані Теми» та ви зможете переглянути теми які обрали вже студенти від певної групи під певний напрямок, як це зображено на рисунку 4.36.

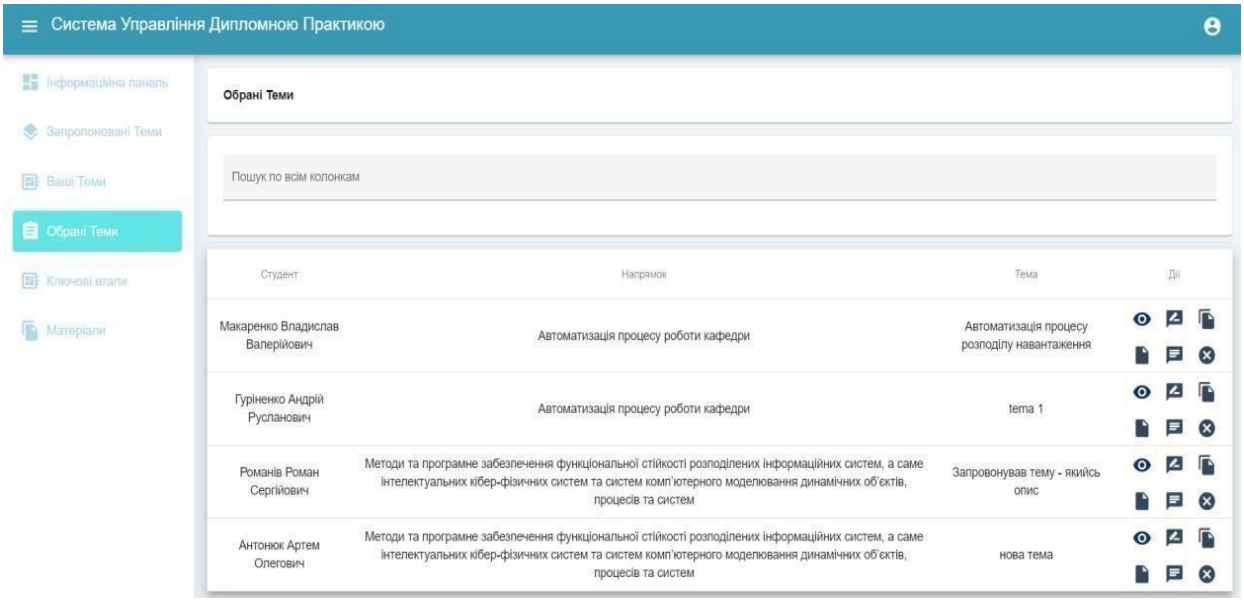

Рисунок 4.36 – Відображення обраних тем студентів для певного викладача

Для налагодження процесу роботи викладача із студентами було створено 6 видів кнопок у вигляді іконок. Давайте розберемо детально кожну із них.

Розпочнемо із іконки перегляду повної інформації обраної теми певним студентом, тобто це наявна інформація в системі безпосередньо про самого студента та назва напрямку разом із темою, як це зображено на рисунку 4.38. Для цього потрібно натиснути на іконку «Додаткова інформація» , як це зображено на рисунку 4.37.

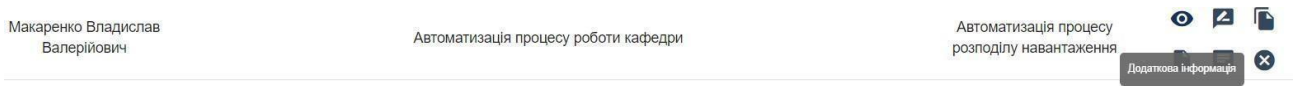

Рисунок 4.37 – Іконка для перегляду додаткової інформації про обрану тему

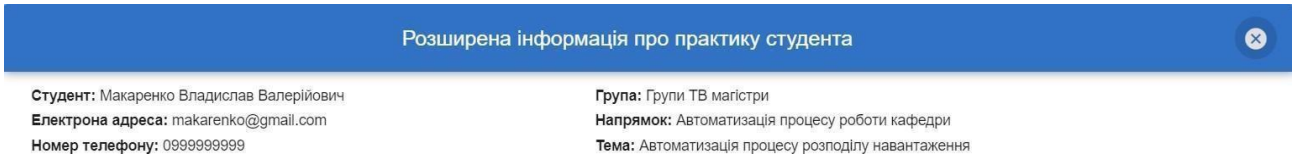

Рисунок 4.38 – Модульне вікно перегляду інформації про обрану тему

Для налагодження процесу роботи студента та викладача було створено можливість створення особистого календарного плана студента. Викладач має можливість додавати, редагувати та видаляти етапи. А також виставляти статус виконання етапа студентом, як це зображено на рисунку 4.40. Для цього потрібно натиснути на іконку «Календарний План», як це зображено на рисунку 4.39.

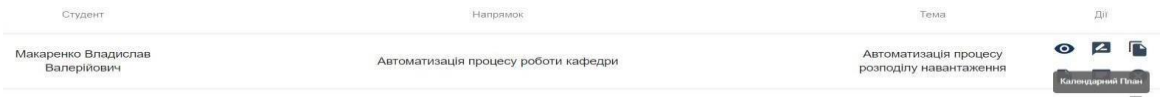

Рисунок 4.39 – Іконка для перегляду календарного плану студента

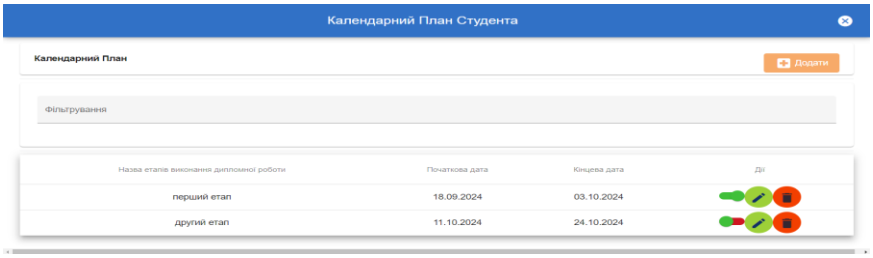

Рисунок 4.40 – Модульне вікно перегляду календарного плану студента

Також є можливість взаємодії студента і викладача для створення успішної версії щоденника та звіту студента. Викладач має можливість видалити, завантажити або вивантажити певний файл, як це зображено на рисунках 4.43 та 4.44. Для цього потрібно натиснути на іконку «Звіт», як це зображено на рисунку 4.42, або на іконку «Щоденник», як це зображено на рисунку 4.41.

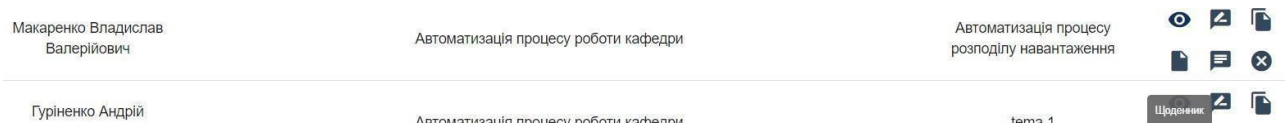

Рисунок 4.41 – Іконка для перегляду версій щоденника студента

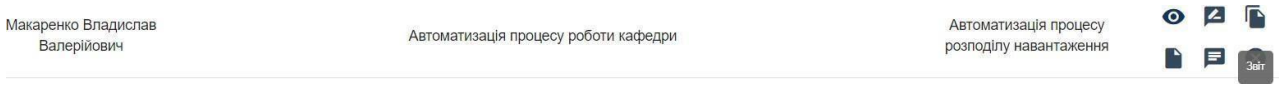

Рисунок 4.42 – Іконка для перегляду версій звіту студента

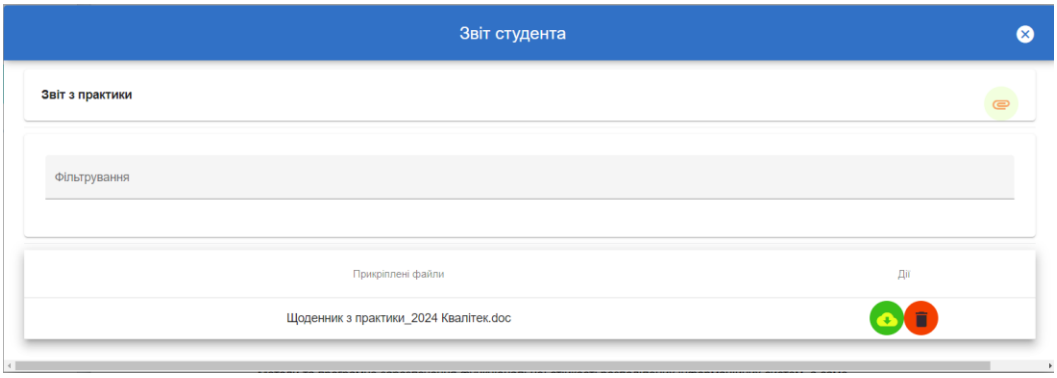

Рисунок 4.43 – Модульне вікно перегляду версій звіту студента

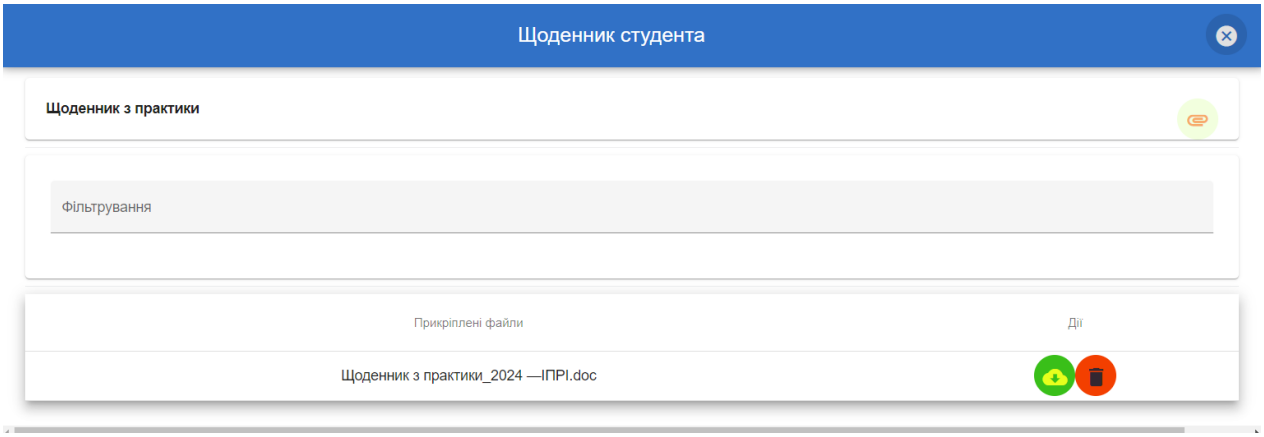

Рисунок 4.44 – Модульне перегляду версій щоденника студента

Також для забезпечення ефективної співпраці між викладачем та студентом було створено можливість листування між ними. Для цього потрібно натиснути на іконку «Чат», як це зображено на рисунку 4.45.

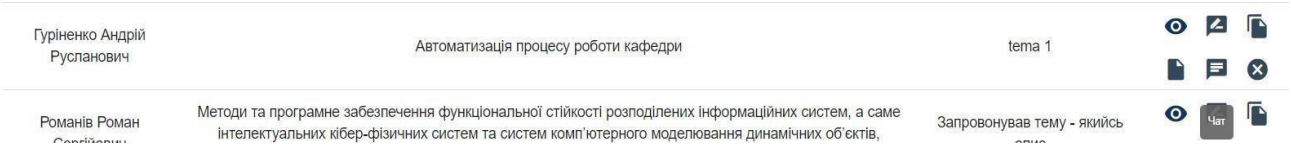

Рисунок 4.45 – Іконка для відкриття чату

Після чого ви зможете вести переписку, як це зображено на рисунку 4.46. Ви зможете побачити повідомлення відправлені вами та співрозмовником, а також дату та час коли повідомлення було відправлено.

Вони також дублюються у вас на електронній пошті, коли повідомлення відправили вам.

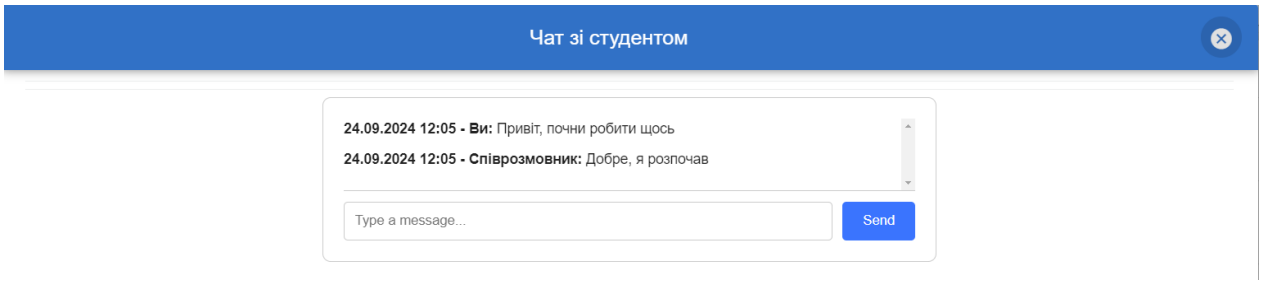

Рисунок 4.46 – Модульне вікно чату

У випадку коли ви припинили співпрацю зі студентом або бажаєте це, то ви маєте змогу відмовити студенту у співпраці.

Для цього потрібно натиснути на іконку «Відв’язати студента», як це зображено на рисунку 4.47. Після чого потрібно підтвердити як це зображено на рисунку 4.15.

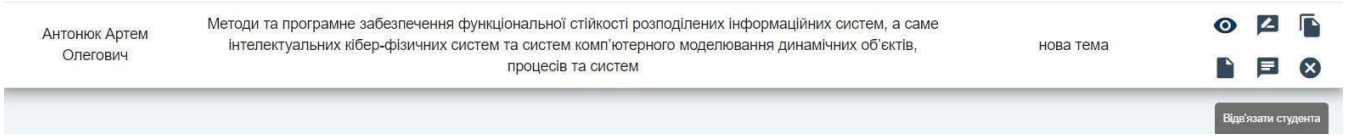

Рисунок 4.47 – Іконка для припинення співпраці зі студентом

І нарешті перейдемо до меню по праву сторону, яке викликається під час натискання іконки акаунта.

В цей раз воно виглядає так як це зображено на рисунку 4.48.

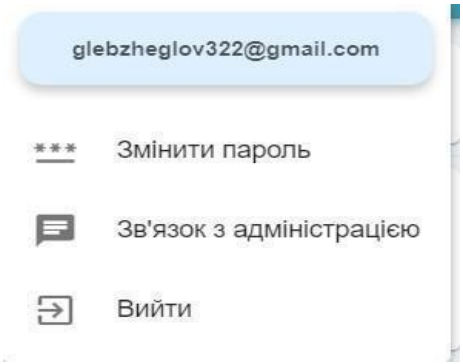

Рисунок 4.48 – Меню акаунта у викладача

Із особливого функціонала викладача це можливість написати адміністрації, для цього потрібно натиснути на «Зв’язок з адміністрацією», після цього ви зможете написати повідомлення, яке вам потрібно та натиснете «Відправити», як це зображено на рисунку 4.49.

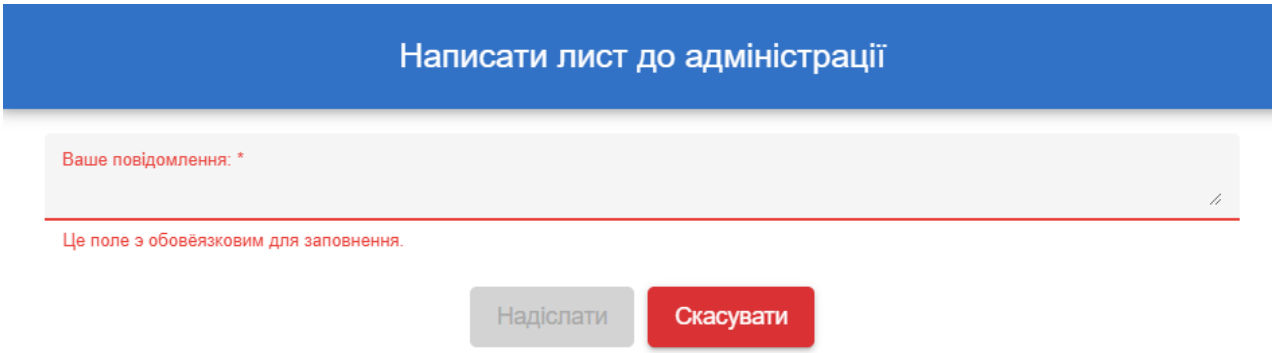

Рисунок 4.49 – Модульне вікно надсилання листа адміністрації

4.4.3 Можливості студентів в системі

Перейдемо до опису можливостей студентів в системі. Аналогічно як до адміністратору так і для викладача та студента – після того як ви увійшли в систему вас буде зустрічати «Інформаційна панель» на ній відображено підрахунок ключових даних. По праву сторону розташована кнопка у вигляді іконки акаунта користувача, а по ліву сторону основне меню, як це зображено на рисунку 4.50.

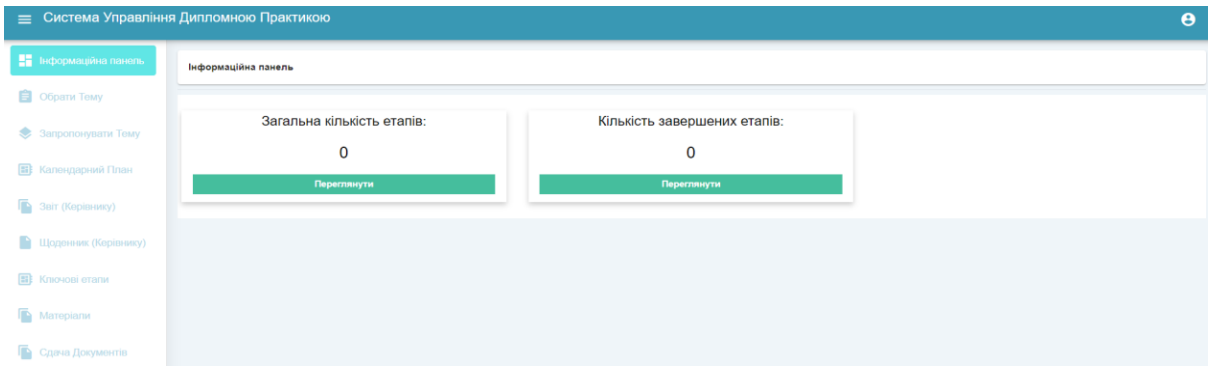

Рисунок 4.50 – Інтерфейс системи доступний для студентів

Перейдемо до ключової можливості студентів – це можливість обрати тему із наявних для його групи по різним напрямкам, як це зображено на

рисунку 4.51. Ці теми є вільні, тобто коли студент буде впевнений у своєму виборі та підтвердив його, то ця тема зникне для інших студентів у його групі. Для цього потрібно натиснути іконку «Обрати», як це зображено на рисунку 4.52.

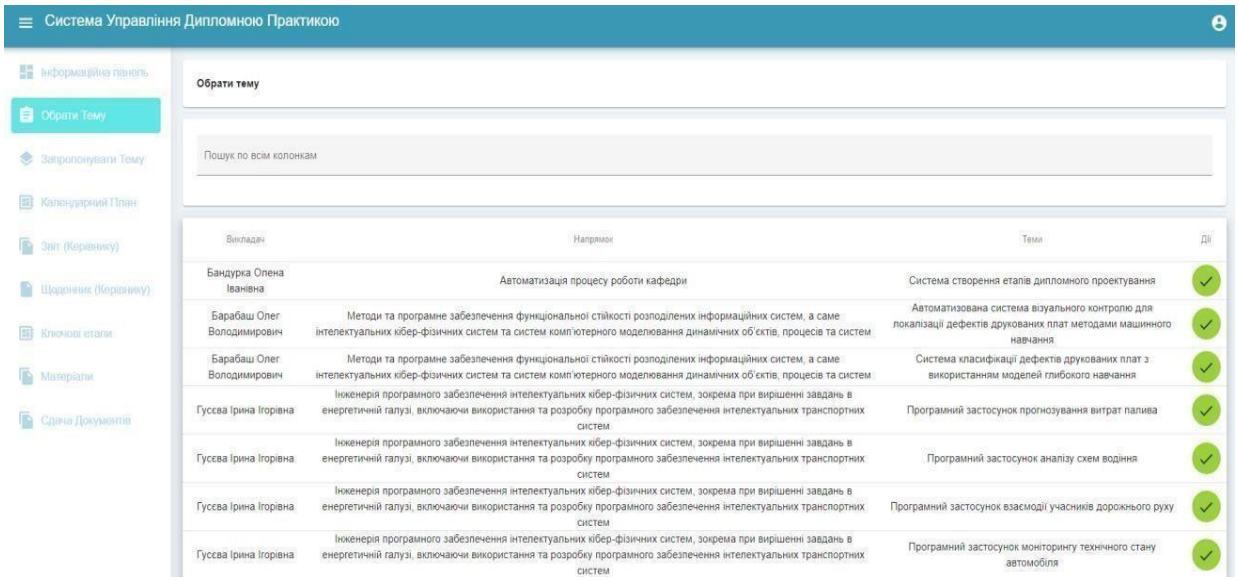

Рисунок 4.51 – Запропоновані теми студентів від викладачів

Рисунок 4.52 – Іконка обрання теми студентом

У випадку якщо студент хоче запропонувати викладачу свою тему, то у нього є ця можливість. Для цього потрібно натиснути зліва в меню «Запропонувати Тему» - після цього ви зможете переглянути створені напрямки для вашої групи під певними викладачами, як це зображено на рисунку 4.53. Для того щоб запропонувати треба натиснути «Запропонувати тему», як це зображено на рисунку 4.54. Та написати тему, яку ви бажаєте запропонувати викладачу, як це зображено на рисунку 4.55. Таким чином ми розглянули та закріпили це розуміння.

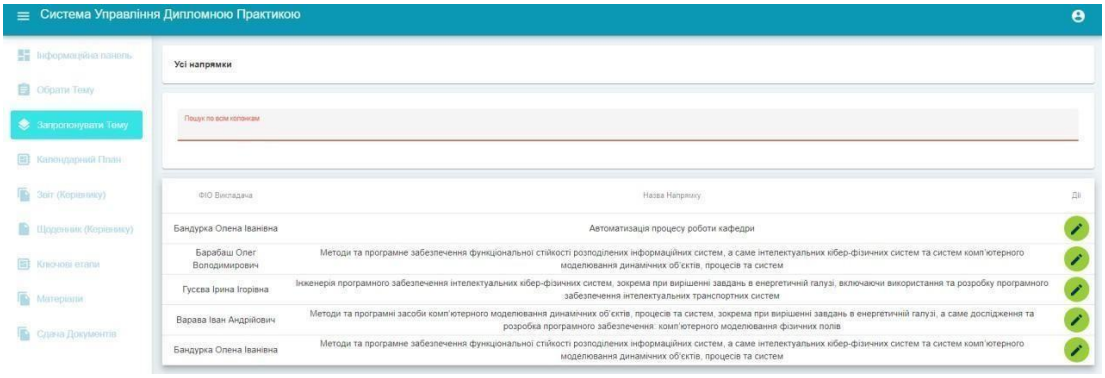

Рисунок 4.53 – Запропонувати тему викладачу під певний напрямок

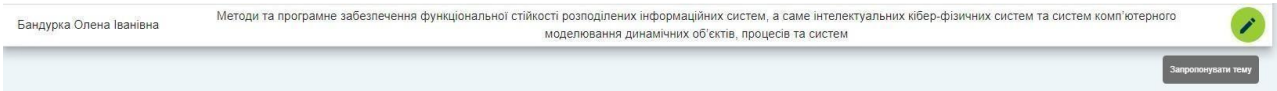

Рисунок 4.54 – Іконка запропонувати тему

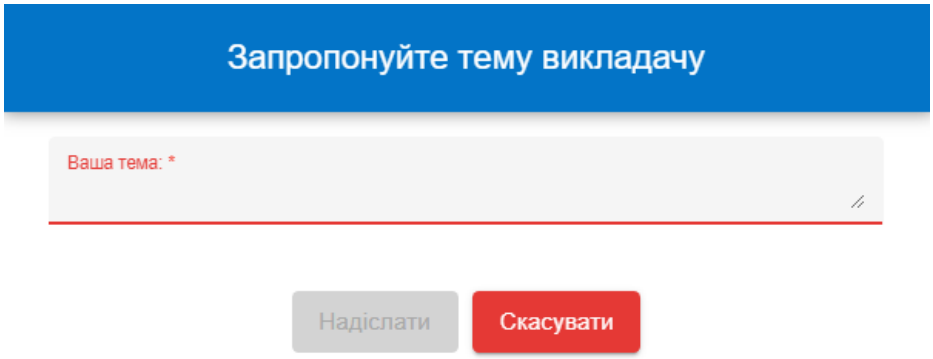

Рисунок 4.55 – Модульне вікно запропонування теми

У випадку якщо ви будете обирати якісь інші вкладки в меню до того як ви обрали тему, або вам погодили її, то ви будете бачити як зображено на рисунку 4.56.

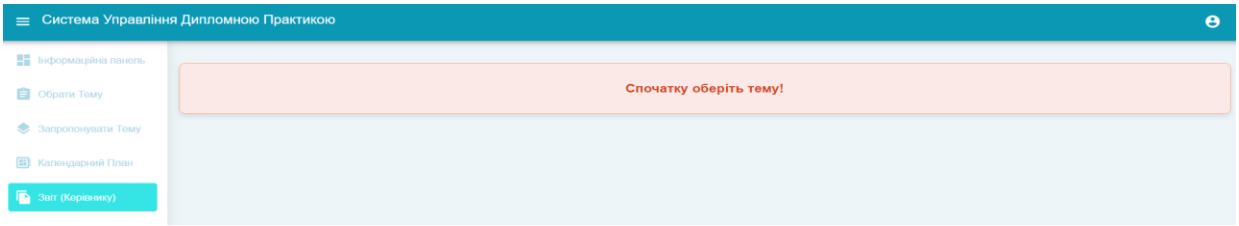

Рисунок 5.56 – Відображення інформації до обрання теми

Після того як ви будете мати закріплену тему під вами, то на вкладці «Обрати тему» ви зможете побачити детальну інформацію, як це зображено на рисунку 4.57.

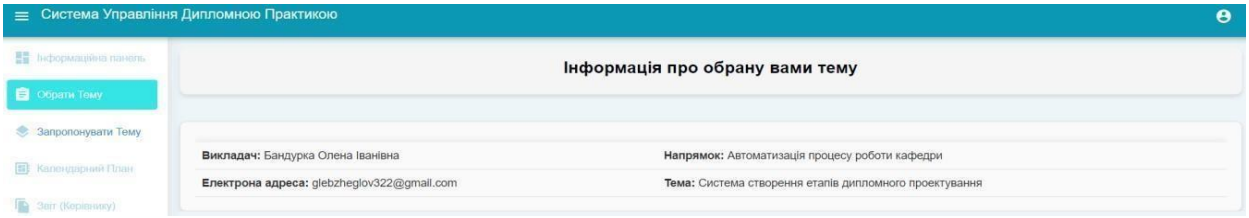

Рисунок 4.57 – Детальна інформація про обрану вами тему

А на вкладці «Запропонувати Тему» ви будете бачити повідомлення про те що тема вже обрана, як це зображено на рисунку 4.58.

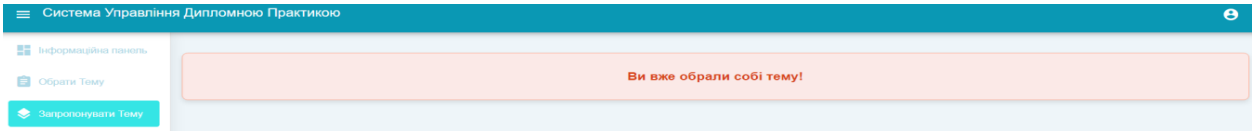

Рисунок 4.58 – Повідомлення про те що тема вже обрана

Перейдемо до перегляду календарного плану студента. Для цього потрібно по ліву сторону меню обрати вкладку «Календарний План». Після чого ви зможете побачити відображення цих даних, як це зображено на рисунку 4.59.

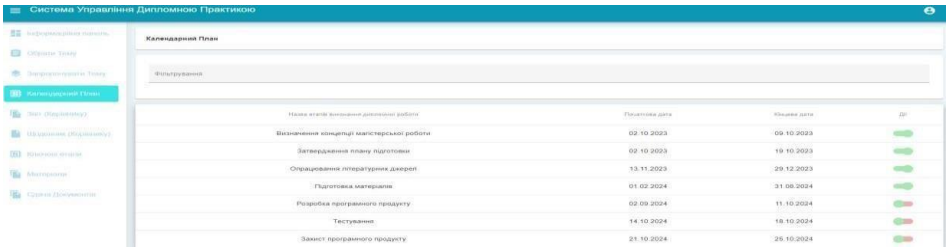

Рисунок 4.59 – Відображення календарного плану студента

Так само ви маєте змогу переглянути дані по ключовим етапам, як це зображено на рисунку 4.60. Таким же чином можна переглянути загальні

матеріали як це зображено на рисунку 4.23.

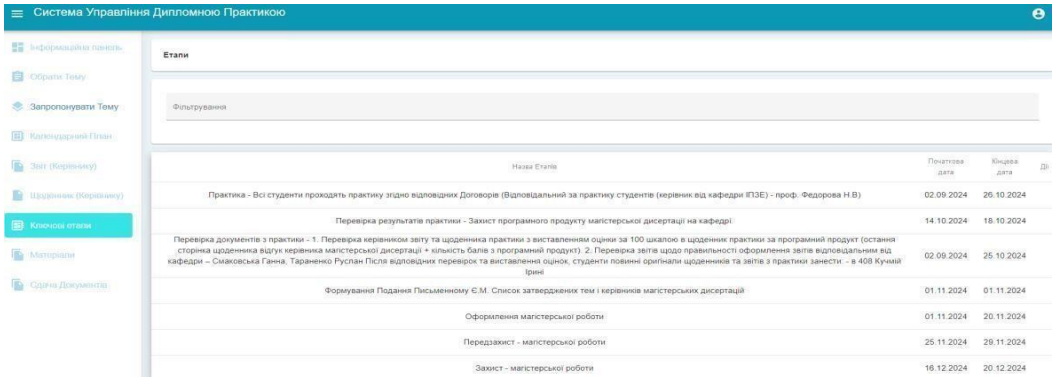

Рисунок 4.60 – Відображення ключових етапов для студента

Студент має можливість здавати версії звіту своєму викладачу на перевірку. Для цього потрібно натиснути на лівій вкладці меню «Звіт (Керівнику)». Та зможете переглянути дані по версіям звіту, як це зображено на рисунку 4.61.

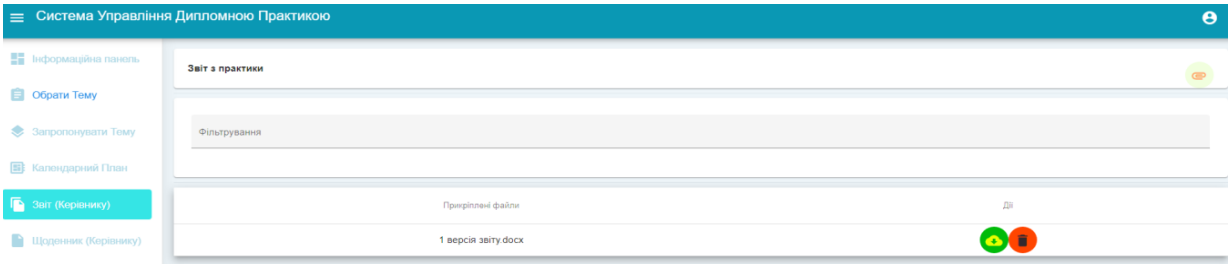

Рисунок 4.61 – Відображення зданих версій звітів студентом

Так само аналогічно це відноситься і до щоденника студента, як це зображено на рисунку 4.62.

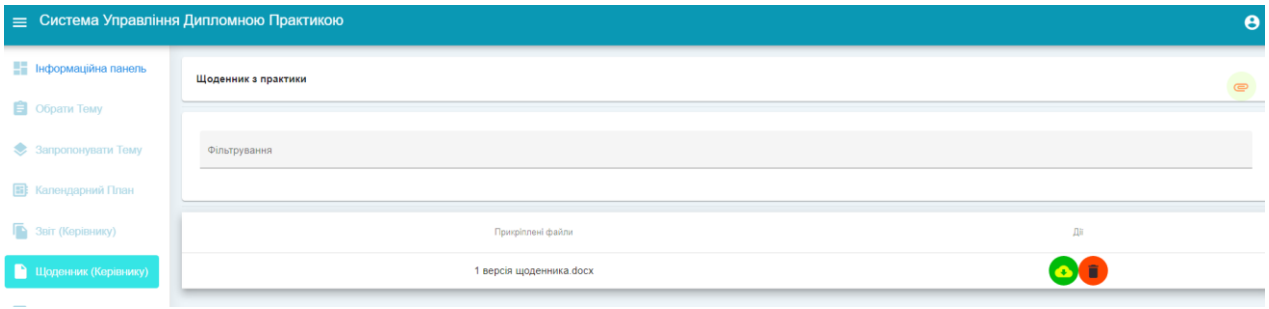

Рисунок 4.62 – Відображення зданих версій щоденника студентом

Також студент має змогу здавати документів по практиці студентом. Для цього вам потрібно обрати пункт меню зліва «Сдача Документів» та ви зможете їх переглянути, як це зображено на рисунку 4.63.

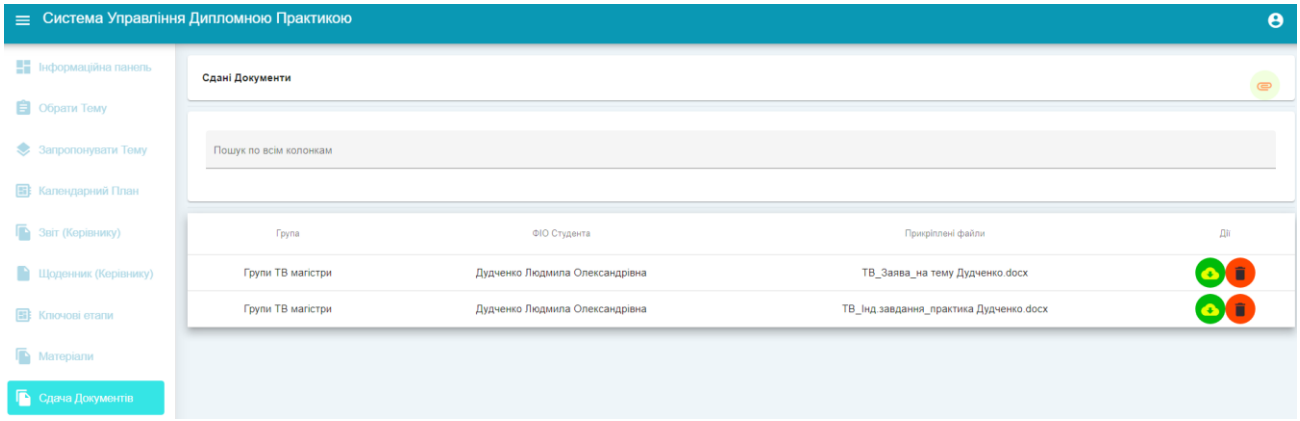

Рисунок 4.63 – Відображення зданих документів студентом

Перейдемо до меню акаунта користувача. Серед особливих можливостей у студента, це можливість відкрити чат з керівником. Переглянемо пункти меню студента, як це зображено на рисунку 4.64.

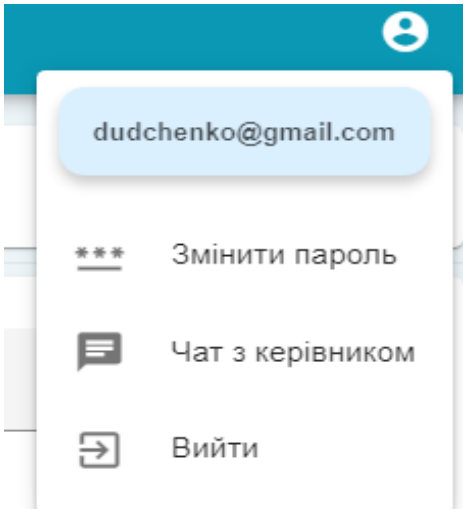

Рисунок 4.64 – Меню акаунта студента

Для відкриття переписки з викладачем, потрібно натиснути «Чат з керівником», та після чого ви зможете побачити вже знайоме відображення переписки, як це зображення на рисунку 4.65.

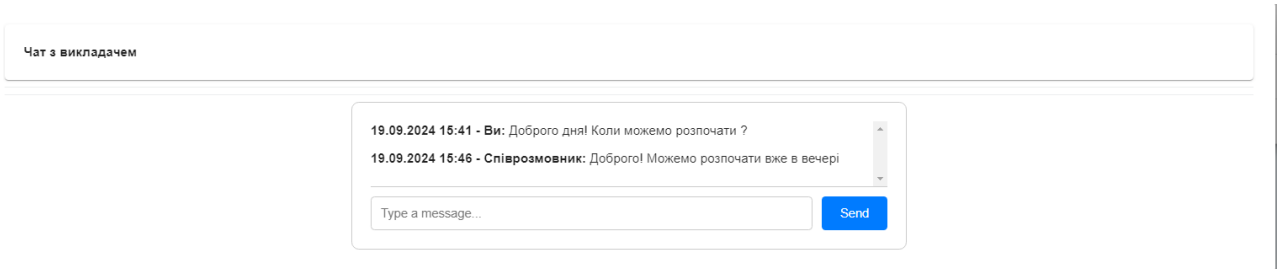

Рисунок 4.65 – Відображення переписки між студентом та викладачем

Висновки до розділу 4

В даному розділі було описано опис програмної реалізації дипломного проектування. Для реалізації системи, що створює етапи дипломного проектування було обрано веб-реалізацію. Такий спосіб дуже зручний і для кінцевого користувача, Та також має можливість локальне використання при наявності певного програмного забезпечення.

Серверна частина відповідає на основний функціонал системи, а саме: створено та реалізовано інформаційну безпеку в залежності від типу користувача; створення загальних та проміжних етапів для дипломного проектування; можливість автономної співпраці між викладачем та студентом; створена система сповіщень та листування; створення звітності.

Було надано опис реалізованій базі даних та надано логічну і ER - діаграму бази даних для даної системи. Та надано детальний опис реалізованих компонентів системи.

Також було описано роботу користувача з програмною системою. Для реалізації системи, що забезпечення створення етапів дипломного проектування було обрано веб-реалізацію. Такий спосіб дуже зручний і для кінцевого користувача, Та також має можливість локальне використання при наявності програмного забезпечення.

5 РОЗРОБКА СТАРТАП – ПРОЄКТУ

У цьому розділі розглядається процес розробки стартап-проєкту, спрямованого на автоматизацію та вдосконалення створення етапів дипломного проектування. Мета цього проєкту полягає в створенні програмного забезпечення, яке дозволить підвищити якість і зручність управління дипломними проєктами як для студентів, так і для викладачів. Система передбачає створення та інтеграцію основних етапів виконання проєкту, контроль проміжних результатів, забезпечення відповідальності та ефективності. Розділ містить опис ідеї проєкту, аналіз технологій, оцінку ринкових можливостей, розробку маркетингової програми і висновки з результатів досліджень, виконаних у процесі розробки стартап-проєкту.

4.5 Опис ідеї проєкта

Основна концепція проєкту передбачає розробку системи для полегшення керування дипломними проєктами. Цей стартап орієнтований на освітні заклади, які прагнуть поліпшити контроль за виконанням дипломних робіт, а також допомогти студентам краще організувати власний робочий процес. Система дозволяє викладачам створювати етапи проєктування, задавати терміни для кожного етапу та здійснювати контроль за виконанням завдань.

Ідея проєкту базується на принципах автоматизації та інтеграції сучасних технологій, що полегшить моніторинг виконання робіт та сприятиме своєчасному виявленню можливих затримок. Студенти, зі свого боку, отримують доступ до інструментів для структурованого планування, постановки цілей та контролю досягнень, що в цілому підвищить їх продуктивність і дисципліну. Викладачі отримають зручний засіб контролю та можливість відслідковувати прогрес у режимі реального часу, що значно зменшить адміністративне навантаження та підвищить якість взаємодії між

студентом і наставником.

Програмне забезпечення передбачає створення окремих профілів для студентів, викладачів і адміністрації навчального закладу. Студентам надається можливість структурувати роботу, переглядати цілі та контролювати свій прогрес. Викладачі можуть легко відстежувати роботу кожного студента на різних етапах проекту, залишати коментарі, проводити оцінку проміжних результатів і надавати рекомендації. Окрім цього, інструменти адміністрації дозволяють отримувати загальні звіти про статус дипломних проектів у масштабі всього закладу, що сприятиме кращому розумінню загальної продуктивності та підвищенню ефективності.

Переваги системи:

- спрощення процесу управління та моніторингу дипломного проектування;
- покращення комунікації між студентом і викладачем завдяки інтегрованій системі зворотного зв’язку;
- підвищення відповідальності студентів за рахунок регулярної звітності та прозорості оцінок;
- зручність у використанні та підтримка багатьох мов, що дозволить закладам адаптувати систему до локальних потреб.

4.6 Технологічний аудит проєкту

Для створення Технологічний аудит розглядає ключові аспекти технологій, необхідних для розробки цього продукту, оцінює відповідність обраного стеку вимогам безпеки, масштабованості, продуктивності та зручності в експлуатації. Система створюється на базі сучасних інструментів, зокрема Node.js для серверної частини, MySQL як реляційної бази даних і Angular з Angular Material для фронтенду.

Вибір Node.js як серверної платформи обумовлений її високою

продуктивністю, можливістю обробки великої кількості запитів та здатністю ефективно масштабуватися. Завдяки асинхронній обробці Node.js забезпечує швидку роботу з даними, що критично важливо для реального часу в комунікації між викладачами та студентами.

MySQL є надійною базою даних для збереження інформації про етапи проектування, коментарі, проміжні оцінки та результати кожного студента. Обрана модель бази даних забезпечує цілісність даних і дає змогу ефективно управляти великими обсягами інформації. Також було враховано можливість подальшого масштабування на випадок розширення користувачів системи.

Angular дозволяє створити інтерактивний та динамічний інтерфейс, що підвищить зручність використання системи. Застосування Angular Material забезпечує адаптивність дизайну, який підходить для різних пристроїв, таких як комп’ютери, планшети та смартфони, що є важливим для користувачів, які можуть працювати зі своїх особистих пристроїв.

Інтеграція компонентів між сервером і клієнтом здійснюється за допомогою REST API. Це забезпечує надійний спосіб обміну даними та гнучкість системи в майбутньому, оскільки дозволяє інтегруватися з іншими програмними продуктами.

Обраний підхід був визнаний найбільш оптимальним для реалізації основних функцій і забезпечення стабільної роботи системи.

4.7 Аналіз ринкових можливостей стартап-проєкту

Аналіз ринкових можливостей стартапу враховує потреби освітнього сектора в автоматизованих системах для управління дипломним проектуванням. Освітні заклади стикаються з проблемами контролю якості дипломних проектів, а також з необхідністю покращення комунікації між студентами та викладачами, що створює попит на подібне програмне забезпечення.

Ринок системи створення етапів дипломного проектування охоплює різні освітні установи та організації. Таблиця нижче відображає основні цільові сегменти та характеристику кожного з них.

Таблиця 5.1 – Цільові сегменти ринку

№	Сегмент ринку	Опис	Потреби та вимоги
1	2	3	4
1	Університети та коледжі	Заклади вищої та професійної освіти, які займаються підготовкою дипломних проектів студентів, а також інших наукових робіт	Потребують системи для моніторингу прогресу студентів, комунікації з викладачами, автоматизації оцінювання та можливості для індивідуального підходу до різних спеціальностей
2	Викладачі та наукові керівники	Викладачі, що працюють зі студентами на етапах проектування дипломів і бажають оптимізувати взаємодію, автоматизувати процес перевірки	Можливість спрощеного управління проектами декількох студентів, відстеження етапів роботи, підтримка зворотного зв’язку і автоматичне інформування про дедлайни
3	Студенти та аспіранти	Студенти та аспіранти, які реалізують дипломні та наукові проекти і потребують зручного інструменту для планування і контролю своїх завдань	Інтуїтивний інтерфейс для роботи з етапами проекту, можливість бачити коментарі від керівника, сповіщення про терміни та вимоги
4	Адміністрація освітніх установ	Керівництво навчальних закладів, відповідальне за якість навчання і контроль за підготовкою дипломних проектів	Отримання статистичних даних про прогрес проектів студентів, загальні звіти про ефективність, можливість інтеграції з іншими внутрішніми системами

Продовження таблиці 5.1

1	2	3	4
5	Професійні освітні центри та тренінги	Організації, які проводять короткострокові або довгострокові курси, тренінги для студентів і спеціалістів	Потребують інструментів для швидкого управління та моніторингу проектів у межах обмеженого часу, надання зворотного зв’язку учасникам курсів
6	ІТ-компанії (партнерські програми)	Компанії, які співпрацюють з університетами для підготовки спеціалістів та можуть брати участь у контролі етапів дипломного проектування	Інтеграція з корпоративними системами, доступ до етапів проектів студентів, можливість надавати рекомендації та оцінки, що дозволяють коригувати навчальні програми

Виділені сегменти охоплюють ключових потенційних користувачів, і кожен з них має свої специфічні вимоги. Розроблена система забезпечить потреби кожного сегменту, що сприятиме розширенню ринку й підвищенню конкурентоспроможності продукту.

Проведений аналіз системи показав, що основними її сильними сторонами є гнучкість, автоматизація процесів і зручність у використанні. Серед слабких сторін відзначено можливі обмеження при інтеграції з іншими освітніми системами, що може впливати на швидкість адаптації продукту. Проте ринок має значний потенціал для розвитку, а інноваційні підходи до автоматизації процесів управління навчанням забезпечують перспективу для успішної комерціалізації.

4.8 Розроблення ринкової стратегії програмного продукту

З розвитком технологій у галузі освіти спостерігається зростання попиту на інноваційні рішення, які покращують процеси навчання, управління

проектами та комунікацію між студентами та викладачами. Аналіз ринку EdTech (освітніх технологій) демонструє кілька ключових тенденцій, що впливають на попит на програмне забезпечення для управління дипломним проектуванням. Ці тенденції включають підвищену увагу до автоматизації, що забезпечує мінімізацію ручної роботи та зменшення ймовірності людських помилок, зручності у використанні, яка дозволяє навіть недосвідченим користувачам легко опановувати системи, адаптивності.

Таблиця 5.2 – Аналіз тенденцій ринку

№	Тенденція ринку	Опис	Вплив на стартап-проєкт
1	2	3	4
1	Зростання попиту на автоматизацію процесів	Заклади освіти шукають способи автоматизації рутинних процесів, таких як моніторинг завдань студентів, контроль за дотриманням дедлайнів, звітність і комунікація	Підтримка автоматичного відстеження етапів дипломного проектування, автоматичне сповіщення про важливі етапи і можливість генерації звітів по виконанню проектів
2	Популярність дистанційного навчання	З розвитком онлайн-освіти зростає потреба у зручних інструментах для планування, комунікації та контролю, що працюють віддалено	Створення інтерфейсу, що доступний з будь-якого пристрою та дозволяє працювати з проектом, виконуючи всі етапи контролю та комунікації дистанційно
3	Підвищення значущості персоналізації	Освітні установи та студенти очікують на індивідуальні рішення, які можна адаптувати під потреби кожного навчального закладу	Забезпечення можливості індивідуального налаштування програми під конкретний навчальний план, специфіку проектів і рівень кваліфікації

Продовження таблиці 5.2

1	2	3	4
4	Зростання інтеграції з іншими системами	Освітні установи прагнуть використовувати комплексні рішення, які інтегруються з LMS (Learning Management System) та іншими платформами	Налагодження API та підтримка інтеграцій для полегшення взаємодії з існуючими освітніми платформами, що вже використовуються в навчальних закладах
5	Попит на адаптивний дизайн	Сучасні освітні інструменти повинні працювати на різних пристроях, зокрема комп'ютерах, планшетах та смартфонах, щоб забезпечити мобільність користувачів	Створення адаптивного інтерфейсу, що буде зручним для різних пристроїв і підвищить доступність системи для користувачів, незалежно від їх місцезнаходження
6	Розвиток хмарних технологій	Хмарні рішення зменшують витрати на інфраструктуру, спрощують доступ до даних і покращують можливості для спільної роботи	Використання хмарних технологій для забезпечення доступу до даних у режимі реального часу, забезпечення безпеки і можливості роботи з великими обсягами інформації

Наведені тенденції вказують на основні напрямки розвитку ринку освітніх технологій і визначають ключові вимоги для успішного стартап-проєкту. Система створення етапів дипломного проектування відповідатиме поточним тенденціям, що сприятиме її конкурентоспроможності на ринку.

Ринкова стратегія спрямована на визначення ключових цільових сегментів і шляхів просування продукту. Основними цільовими сегментами для системи є університети, коледжі та інші освітні установи, які оптимізовують процеси дипломного проектування. Важливою особливістю ринкової стратегії є

її гнучкість: система може бути легко адаптована під специфічні потреби різних закладів освіти.

Для успішного запуску системи створення етапів дипломного проектування на ринок потрібно розробити комплексну ринкову стратегію, що охоплює визначення цільової аудиторії, позиціонування продукту, канали просування та цінову політику.

Нижче представлена таблиця, що детально описує кожен аспект ринкової стратегії для продукту.

Таблиця 5.3 – Ринкова стратегія програмного продукту

№	Елемент стратегії	Опис	Ціль
1	2	3	4
1	Цільова аудиторія	Основними споживачами продукту є університети, коледжі, професійні освітні заклади, а також приватні тренінгові компанії та онлайн-курси. Продукт орієнтований на адміністрацію навчальних закладів, викладачів та студентів.	Забезпечення глибокого розуміння потреб кожної групи користувачів, що дозволить адаптувати функціонал під конкретні вимоги кожного сегменту ринку.
2	Позиціонування продукту	Система позиціонується як спеціалізоване рішення для управління дипломними проектами, яке автоматизує процеси та дозволяє студентам взаємодіяти на всіх етапах проектування.	Підкреслити унікальність системи порівняно з універсальними інструментами для управління проектами, зокрема з фокусом на потреби освітньої сфери.

Продовження таблиці 5.3

1	2	3	4
3	Цінова стратегія	Використовується модель SaaS з підпискою, що дозволяє платити за користування щомісяця або щорічно. Передбачені знижки для освітніх закладів, що впроваджують систему на рівні факультетів або університетів.	Забезпечити доступність продукту для освітніх закладів із різними бюджетами та залучити більше користувачів за допомогою гнучкої системи знижок і тарифів.
4	Канали просування	Основні канали – це освітні конференції, семінари, онлайн-вебінари, платформи EdTech, а також цільова реклама в соціальних мережах. Крім того, планується співпраця з університетами та проведення безкоштовних демо-сесій.	Залучити якнайбільше потенційних користувачів через демонстрацію переваг системи та побудову довіри через освітні заходи та спільноти.
5	Точки контакту	Вебсайт продукту, підтримка користувачів, соціальні мережі, відео на YouTube, де представлені функції та переваги системи, блоги для обговорення проблем управління проектами.	Забезпечити інформативну підтримку для користувачів на всіх етапах ознайомлення з продуктом, щоб знизити бар’єри для впровадження та підтримувати лояльність.

Продовження таблиці 5.3

1	2	3	4
6	Стратегія підтримки клієнтів	Впроваджена цілодобова підтримка, персоналізовані консультації та навчальні матеріали, доступні на сайті. Створено навчальний центр з відеоінструкціями для викладачів і студентів.	Задовольнити потреби клієнтів, зокрема допомогти з інтеграцією, налаштуванням і подальшим користуванням, щоб забезпечити ефективну роботу та збільшити задоволення користувачів.
7	Аналіз результативності	Регулярний аналіз показників, таких як кількість нових користувачів, рівень задоволеності, частота використання функцій і повернення клієнтів.	Виявити сильні та слабкі сторони продукту, адаптувати ринкову стратегію відповідно до потреб ринку і забезпечити сталий ріст користувацької бази.

Ця стратегія дозволяє зосередитися на конкретних потребах освітніх закладів, підкреслюючи ключові переваги продукту для ринку та формуючи привабливу пропозицію для основних груп користувачів.

4.9 Розроблення маркетингової програми стартап-проєкту

Маркетингова програма покликана забезпечити привернення уваги цільової аудиторії та підвищення впізнаваності продукту. Основні елементи програми включають розробку рекламних кампаній, цільовий контент і регулярну взаємодію з аудиторією для створення позитивного іміджу продукту. Маркетингова програма стартапу орієнтована на комплексне охоплення

цільової аудиторії та ефективне просування продукту на ринку.

Нижче представлена таблиця з основними компонентами маркетингової програми.

Таблиця 5.4 – Маркетингові програми стартап-проєкту

№	Компонент програми	Опис	Ціль
1	2	3	4
1	Цільова реклама в соціальних мережах	Використання соціальних мереж (Facebook, LinkedIn, Instagram) для розміщення таргетованої реклами, що охоплює аудиторію освітніх фахівців, викладачів і студентів	Залучити цільову аудиторію, зацікавлену в оптимізації освітнього процесу, і підвищити обізнаність про продукт
2	Контент-маркетинг	Публікація статей, блогів, аналітичних матеріалів про ефективне управління освітніми проектами, кейсів використання системи та успішних прикладів її впровадження в навчальних закладах	Підвищити довіру до бренду, показати експертність у темі управління освітніми проектами та залучити органічний трафік
3	Email-розсилка	Регулярні розсилки з інформацією про нові функції, кейси успішного використання, інноваційні підходи в управлінні проектами для викладачів і адміністрації	Побудувати прямий канал комунікації з клієнтами, підтримувати інтерес до продукту, забезпечити інформованість про оновлення та новини.

Продовження таблиці 5.4

1	2	3	4
4	Цілодобова підтримка клієнтів	Налагоджена служба підтримки, яка відповідає на запити користувачів	Забезпечити безперебійну роботу продукту, запобігти їх переходу до конкурентів
5	Онлайн-вебінари та демо-сесії	Проведення онлайн-вебінарів з демонстрацією системи, навчальними сесіями для викладачів і адміністрації, а також Q&A сесіями для обговорення питань потенційних клієнтів	Допомогти клієнтам краще зрозуміти функціонал продукту, підвищити їх зацікавленість, надати можливість взаємодії з командою проекту
6	Пошукова оптимізація (SEO)	Оптимізація вебсайту та блогу для пошукових систем, щоб потенційні клієнти могли легко знайти інформацію про продукт при пошуку освітніх технологій або управління проектами	Підвищити видимість продукту в органічних пошуках, збільшити кількість відвідувачів сайту, залучити нових клієнтів
7	Співпраця з освітніми закладами	Партнерство з університетами, коледжами та тренінговими центрами, з пропозицією безкоштовного тестування, знижок та спеціальних умов для тривалого використання продукту	Залучити освітні заклади, забезпечити швидке впровадження системи в освітній процес і створити базу лояльних користувачів

Запропонована маркетингова програма дозволяє комплексно підходити до просування продукту, поєднуючи цифрові канали, освітні заходи та тісну співпрацю з клієнтами. Така стратегія спрямована на довгострокове утримання користувачів і забезпечення сталого зростання стартапу.

Висновки до розділу 5

У даному розділі детально розглядається процес розробки стартап-проєкту системи для створення етапів дипломного проектування, що включає всебічний аналіз ринкових можливостей, вибір технологій та стратегію виходу на ринок. Основна ідея проекту полягає у створенні ефективного інструменту для управління етапами дипломних робіт, який забезпечує зручність для студентів, викладачів та адміністрації навчальних закладів. Розробка продукту почалася з проведення технологічного аудиту, що дозволив визначити сучасні інструменти та платформи, найбільш придатні для реалізації всіх необхідних функцій системи. Особливу увагу було приділено вибору технологій для організації інтерфейсу, управління базами даних і забезпечення інтеграції з існуючими освітніми платформами.

Аналіз ринку показав, що освітній сектор все більше потребує цифрових рішень для управління проектною діяльністю, особливо у сфері дипломних робіт, що передбачає багатоступеневий процес та потребу у співпраці. Існує значний потенціал для комерційного успіху продукту, оскільки багато навчальних закладів стикаються з необхідністю впровадження зручних і функціональних інструментів, які автоматизують ключові процеси та забезпечують простий контроль над прогресом студента.

Крім того, стратегія враховує особливості просування продукту через соціальні мережі, освітні платформи та партнерські програми з навчальними закладами. Завдяки продуманій маркетинговій програмі продукт отримує високу впізнаваність і забезпечує довготривалі відносини з клієнтами.

ВИСНОВКИ

В ході виконання завдання магістерської дисертації було створено систему, яка автоматизує процес створення етапів дипломного проектування. Основним шляхом реалізації цієї задачі було обрано створити веб-додаток, який буде об’єднувати в собі бізнес-логіку самої системи та інтерфейс користувача для роботи.

Було проведено детальний аналіз поточного процесу створення етапів дипломного проектування на кафедрі. Це допомогло зрозуміти проблеми та недоліки минулого підходу, а також було визначено та реалізовоно ключові вимоги до нової системи.

Проведено повне тестування системи на виконання усіх перелічених функцій та також протестовано на різні помилки і дефекти, виконано аналіз та виправлено їх.

Було застосовано та продумано певні алгоритми для роботи з таблицями, які несуть у собі звітну функцію, розділено проект на певну кількість файлів та проведено роботу з їхнім підключенням один до одного.

Було реалізовано наступні завдання:

- створено базу даних, яка буде включає в собі дані про користувачів системи, групи студентів, типи файлів, самі файли, завдання студентів та дані про чати, також основні дані про практику студентів такі як напрямки та теми;
- створено систему, яка забезпечує безпеку даних та безпосередньо навігацію в самому додатку;
- створено можливість перегляду, редагування, додавання та пошуку введених даних;
- створено можливість забезпечення створення етапів дипломного проектування та автоматизації процесу обрання тем студентів;
- створено функціонал для формування звітів — перегляд обраних тем студентів та список студентів, які ще не обрали тему.

Остаточним результатом магістерської дисертації є створена та впроваджена система, яка дозволяє автоматизувати процес створення етапів дипломного проектування на кафедрі та покращує ефективність та точність цього процесу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. К. Дейт. Введення в систему бази даних. М.: Наука, 1980, с. 463.
2. Грассіг І., Кемерер К. Основи управління програмними проектами. М.: Видавництво ІД «Вільямс», 2004. — 432 с.
3. Мартін Р. Чистий код: Створення, аналіз та рефакторинг. К.: Видавництво "Ранок", 2018. — 464 с.
4. Фаулер М. Архітектура корпоративних програм. Патерни проектування. М.: ДМК Пресс, 2020. — 624 с.
5. Шнхайт Дж. Основи баз даних. М.: Питер, 2011. — 608 с.
6. Node.js офіційна документація. [Електронний ресурс]. – Режим доступу: https://nodejs.org/ (дата звернення: 02.09.2024).
7. MySQL офіційна документація. [Електронний ресурс]. – Режим доступу: https://dev.mysql.com/doc/ (дата звернення: 12.09.2024).
8. Angular офіційна документація. [Електронний ресурс]. – Режим доступу: https://angular.io/ (дата звернення: 15.09.2024).
9. Open Server документація. [Електронний ресурс]. – Режим доступу: https://ospanel.io/ (дата звернення: 20.09.2024).
10. JetBrains IntelliJ IDEA для розробників на JavaScript. [Електронний ресурс]. – Режим доступу: https://www.jetbrains.com/ru-ru/idea/download/ (дата звернення: 05.09.2024).
11. Express.js — Фреймворк для Node.js. [Електронний ресурс]. – Режим доступу: https://expressjs.com/ (дата звернення: 15.09.2024).
12. Реляційні бази даних. [Електронний ресурс]. – Режим доступу: https://www.mysql.com/ (дата звернення: 20.09.2023).
13. JavaScript для веб-розробки. [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/uk/docs/Web/JavaScript (дата звернення: 15.09.2024).
14. Використання Nginx для веб-розробки. [Електронний ресурс]. –

Режим доступу: https://www.nginx.com/ (дата звернення: 25.09.2024).

15. Google Workspace довідковий центр. [Електронний ресурс]. – Режим доступу: https://support.google.com/workspace/ (дата звернення: 02.09.2024).

16. Jira офіційна документація. [Електронний ресурс]. – Режим доступу: https://www.atlassian.com/software/jira/guides (дата звернення: 02.09.2024).

17. Гриценко В.Г. Інформаційно-аналітична система управління освітнім процесом ЗВО. Черкаси: ЧНУ, 2023. — 298 с.

18. Лаврищева К.М. Програмна інженерія. Київ: НаУКМА, 2023. — 374 с.

19. Коваль Т.І. Автоматизація процесів управління проектами у закладах вищої освіти. Київ: КНУ, 2023. — 286 с.

20. Гужва В.М. Інформаційні системи в міжнародному бізнесі. Київ: КНЕУ, 2023. — 458 с.

21. Томашевський В.М. Моделювання систем. Київ: КПІ, 2023. — 352 с.

22. Луцький М.Г. Інформаційні технології управління проектами. Київ: НАУ, 2023. — 302 с.

23. Павленко П.М. Автоматизовані системи технічної підготовки виробництва. Київ: НАУ, 2023. — 268 с.

24. Карпенко М.Ю. Технічна підтримка прийняття рішень. Харків: ХНУМГ, 2023. — 245 с.

25. Пономаренко В.С. Проектування інформаційних систем. Харків: ХНЕУ, 2023. — 442 с.

26. Федорчук Є.Н. Програмування та підтримка веб-застосувань. Львів: НУ "Львівська політехніка", 2023. — 312 с.

27. Управління проектами в умовах цифрової трансформації / За ред. С.Д. Бушуєва. Київ: КНУБА, 2023. — 386 с.

28. Гайна Г.А. Основи проектування баз даних. Київ: КНУБА, 2023. — 204с.

29. Технології розробки інформаційних систем: підручник / М.М. Поліщук та ін. Київ: КПІ, 2023. — 332 с.

30. Методичні рекомендації з управління ІТ-проектами / В.В. Литвинов та ін. Чернігів: ЧНТУ, 2023. — 184 с.

31. Фісун М.Т. Інтелектуальні інформаційні системи. Миколаїв: ЧНУ, 2023. — 266 с.

32. Цюцюра С.В. Системи автоматизованого проектування та управління. Київ: КНУБА, 2023. — 248 с.

33. Сучасні методології управління проектами в організаціях України / За ред. І.В. Чумаченка. Харків: НТУ "ХПІ", 2023. — 324 с.

ДОДАТОК А Лістинг програми

```
// серверна частина
// конект до БД
const mysql = require('mysql'); require('dotenv').config(); function
createConnection() { return mysql.createConnection({
port: process.env.DB_PORT, host: process.env.DB_HOST,
user: process.env.DB_USERNAME, password: process.env.DB_PASSWORD,
database: process.env.DB_NAME
});
}
let connection;
function handleDisconnect() { connection = createConnection();
connection.connect((err) => {
if (err) {
console.error('Error connecting to the database:', err); setTimeout(handleDisconnect,
2000);
} else {
console.log('Connected to the database');
}
});

connection.on('error', (err) => { console.error('Database connection error:', err);
if (err.code === 'PROTOCOL_CONNECTION_LOST') {
handleDisconnect();
} else { throw err;
}
});
}
handleDisconnect(); module.exports = connection;
```

```
// сервер
require('dotenv').config(); const http = require('http'); const app = require('./index');
const socketIO = require('socket.io'); const server = http.createServer(app);
server.listen(process.env.PORT, process.env.HOST, () => {
console.log(`Server running
://${process.env.HOST}:${process.env.PORT}`);
});

// index.js
const express = require('express'); var cors = require('cors');

const connection = require('./connection') const userRoute = require('./routes/user');
const specialityRoute = require('./routes/speciality');
const directionofthesisRoute = require('./routes/directionofthesis'); const
dashboardRoute = require('./routes/dashboard');
const diploma_practiceRoute = require('./routes/diploma_practice');
const tasksRoute = require('./routes/tasks');
const resultfileRoute = require('./routes/resultfile'); const chatRoute =
require('./routes/chat');
const app = express(); app.use(cors())
app.use(express.urlencoded({extended: true})) app.use(express.json());
app.use('/user', userRoute); app.use('/speciality', specialityRoute);
app.use('/directionofthesis', directionofthesisRoute); app.use('/dashboard',
dashboardRoute); app.use('/diploma_practice', diploma_practiceRoute);
app.use('/tasks', tasksRoute);
app.use('/resultfile', resultfileRoute); app.use('/chat', chatRoute);
app.use(cors({
origin: 'http://localhost:4200',
```

```
methods:  'GET,HEAD,PUT,PATCH,POST,DELETE',
credentials: true
}));
app.listen(process.env.PORT, () => {
})
module.exports = app;

// сервіс авторизації require('dotenv').config()
const jwt = require('jsonwebtoken'); function authenticateToken(req, res, next) {
const authHeader = req.headers['authorization']
const token = authHeader && authHeader.split(' ')[1] if (token == null) return
res.sendStatus(401);
jwt.verify(token, process.env.ACCESS_TOKEN, (err, response) => { if(err) return
res.sendStatus(403);
res.locals = response next()
})
}
module.exports = {authenticateToken: authenticateToken}

// routs chat
const express = require('express');
const connection = require('../connection'); const router = express.Router();
const nodemailer = require('nodemailer') require('dotenv').config();
// Authentification middleware
var auth = require('../services/authentification'); var checkRole =
require('../services/checkRole'); var transporter = nodemailer.createTransport({
service: 'gmail', auth: {
user: process.env.EMAIL_LOGIN, pass: process.env.EMAIL_PASSWORD
}
```

```
});
router.post('/messagesFromChat', auth.authenticateToken, (req, res) => { const
practice_id = req.body.practice_id;
connection.query(
'SELECT ms.`id` as id, ms.`sender_id` as `sender_id`, ms.`content` as
`content`, ms.`created_at` as `created_at`, ms.`conversation_id` as `conversation_id`,
us.`login` as email FROM messages ms INNER JOIN `user` us ON (us.`ID` =
ms.`sender_id`) WHERE conversation_id = ( SELECT cv.`id` FROM
`conversations` cv WHERE cv.`diploma_practice_id` = ? ) ORDER BY created_at
ASC',
[practice_id], (err, results) => {
if (err) {
console.error('Error fetching messages:', err);
return res.status(500).json({ error: 'Помилка отримання повідомлень'
});
                                                    }
                                                    res.json(results);
                                                }
                                    );
                                    });
```

```
router.post('/sendmessage', auth.authenticateToken, (req, res) => { const {
practice_id, senderLogin, content } = req.body; connection.query(
'INSERT INTO messages (conversation_id, sender_id, content) VALUES (
(SELECT `id` FROM `conversations` WHERE `diploma_practice_id` = ?) ,
(SELECT
`ID` from `user` WHERE `login` = ?), ?)', [practice_id, senderLogin, content], (err,
result) => {
if (err) {
console.error('Error sending message:', err);
return res.status(500).json({ error: 'Помилка під час
відправки повідомлення' });

else {
var query = "SELECT uss.`login` as teacher_login, us.`login` as student_login
FROM `diploma practice` dp INNER JOIN `user` us ON (us.`ID` = dp.`student_id`)
INNER JOIN `directionofthesis` dr ON (dr.`ID` = dp.`directionofthesis_id`) INNER
JOIN `specialty` sp ON (sp.`ID` = dr.`group_id`) INNER JOIN `user` uss ON
(uss.`ID` = dr.`teacher_id`)";
connection.query(query, (err, result)=> { if(!err){
var from; var to;
if (result[0].teacher_login == senderLogin) { from = result[0].teacher_login;
to = result[0].student_login;
} else {
from = result[0].student_login; to = result[0].teacher_login;
}
var mailOptions = {
from: process.env.EMAIL_LOGIN, to: to,
subject: 'Вам надіслали повідомленя з "Система Керування Дипломною
Практикою"',
```

```
html: '<p><b>'+ from + ': ' +content+'</b></p>'
}
transporter.sendMail(mailOptions, function (error, info) { if(error) {
console.log(error)
} else {
console.log('Email sent: ' + info.response);}
})
res.json({ message: 'Повідомлення надіслано успішно!' });
} else {
return  res.status(500).json(err);
}
})
}
}
);
});
router.post('/sendMessageToAdmins', auth.authenticateToken, (req, res) => { const {
content, teacherLogin} = req.body;
var mailOptions = { from: teacherLogin,
to: process.env.EMAIL_LOGIN, subject: 'Питання від викладача!', html:
'<p><b>'+content+'</b></p>'
}
transporter.sendMail(mailOptions, function (error, info) { if(error) {
console.log(error)
} else {
console.log('Email sent: ' + info.response);
}
})
return res.json({ message: 'Повідомлення надіслано успішно!' });
```

```
});router.post('/sendMessageFromAdmins', auth.authenticateToken, (req, res) => {
const { content, email} = req.body;
var mailOptions = {

from: process.env.EMAIL_LOGIN, to: email,

subject: 'Вам надіслано повідомлення від "Система
Управління Дипломною Практикою"!',

html: '<p><b>'+content+'</b></p>'

}

transporter.sendMail(mailOptions, function (error, info) { if(error) {

console.log(error)

} else {

console.log('Email sent: ' + info.response);

}

})

return res.json({ message: 'Повідомлення надіслано успішно!' });

});

module.exports = router;

// dashboard

// API для роботи із таблицею faculty const express = require('express')

const connection = require('../connection'); const router = express.Router();

const jwt = require('jsonwebtoken'); require('dotenv').config();

//authentification

var auth = require('../services/authentification')

//checkRoles

var checkRole = require('../services/checkRole') router.get('/info',
auth.authenticateToken , (req, res)=> { var specialityCount;
var directionofthesisCount; var diplomapracticeCount;

var query = "SELECT COUNT(*) as specialtyCount FROM `specialty`"

connection.query(query, (err, result)=> {

if(!err){
```

```
specialityCount = result[0].specialtyCount;
} else {
return  res.status(500).json(err);
}
})
query = "SELECT COUNT(*) as diplomapracticeCount FROM `diploma practice`"
connection.query(query, (err, result)=> { if(!err){
diplomapracticeCount = result[0].diplomapracticeCount;
} else {
return  res.status(500).json(err);
}
})
query = "SELECT COUNT(*) as directionofthesisCount FROM
`directionofthesis`"
connection.query(query, (err, result)=> { if(!err){
directionofthesisCount = result[0].directionofthesisCount; var data = {
specialityCount: specialityCount,
directionofthesisCount: directionofthesisCount, diplomapracticeCount:
diplomapracticeCount
}
return res.status(200).json(data)
} else {
return  res.status(500).json(err);
}
})
})
router.post('/infoForStudent', auth.authenticateToken , (req, res)=> { var tasksCount;
var completeTasksCount;
```

```
let info_regarding_student = req.body;
var query = "SELECT COUNT(*) as tasksCount FROM `tasks`ts INNER JOIN
`diploma practice` dp ON(dp.`ID` = ts.`diplomapractice_id`) INNER JOIN `user` us
ON(us.`ID` = dp.`student_id`) WHERE us.`login` = ?"
connection.query(query,[info_regarding_student.login], (err, result)=> { if(!err){
tasksCount = result[0].tasksCount;
} else {
return  res.status(500).json(err);
}
})
query = "SELECT COUNT(*) as tasksCount FROM `tasks`ts INNER JOIN
`diploma practice` dp ON(dp.`ID` = ts.`diplomapractice_id`) INNER JOIN `user` us
ON(us.`ID` = dp.`student_id`) WHERE us.`login` = ? AND ts.`status` = 'true'"
connection.query(query,[info_regarding_student.login], (err, result)=> { if(!err){
completeTasksCount = result[0].tasksCount;
var data = {
tasksCount: tasksCount, completeTasksCount: completeTasksCount
}
return res.status(200).json(data)
} else {
return  res.status(500).json(err);
}
})
})
router.post('/infoForTeacher', auth.authenticateToken , (req, res)=> { var
newThemeCount
var practiceCount;
let info_regarding_teacher = req.body;
var query = "SELECT COUNT(*) as practiceCount FROM `diploma practice` dp
```

```
INNER JOIN `directionofthesis` dr ON (dr.`ID` = dp.`directionofthesis_id`) INNER JOIN `specialty` sp ON (sp.`ID` = dr.`group_id`) INNER JOIN `user` uss ON (uss.`ID`
= dr.`teacher_id`) WHERE uss.`login` = ? AND dp.`student_id` IS NULL"
connection.query(query,[info_regarding_teacher.login], (err, result)=> { if(!err){
practiceCount = result[0].practiceCount;
} else {
return  res.status(500).json(err);
}
})
query = "SELECT COUNT(*) as newThemeCount FROM `diploma practice` dp
        sp        ON      (sp.`ID`=     dr.`group_id`)   INNER                JOIN
        `user`    uss     ON      (uss.`ID` = dr.`teacher_id`) WHERE uss.`login` = ?"
connection.query(query,[info_regarding_teacher.login], (err, result)=> { if(!err){
newThemeCount = result[0].newThemeCount; var data = {
newThemeCount:  practiceCount,  practiceCount:  newThemeCount
}
return res.status(200).json(data)
} else {
return  res.status(500).json(err);
}
})
})
module.exports = router
//diploma practice
// API для роботи із таблицею faculty const express = require('express')
const connection = require('../connection'); const router = express.Router();
const nodemailer = require('nodemailer') const jwt = require('jsonwebtoken');
require('dotenv').config();
```

```
const bodyParser = require('body-parser'); const XLSX = require('xlsx');
const fs = require('fs'); const path = require('path');
//authentification
var auth = require('../services/authentification')
//checkRoles
var checkRole = require('../services/checkRole') var transporter =
nodemailer.createTransport({
service: 'gmail', auth: {
user: process.env.EMAIL_LOGIN, pass: process.env.EMAIL_PASSWORD
}
});
// БАЗОВІ CRUD ЗАПИТИ
// get
router.post('/get', auth.authenticateToken, checkRole.checkRoleTeacher, (req, res)=>
{
let informationTeacher = req.body;
let query = "SELECT dr.`ID` as directionId, dp.`ID` as ID, dp.`description` as description, dr.`name` as directionofthesis_name, sp.`Name` as group_name, dp.`student_id` as student_id FROM `diploma practice` dp INNER JOIN `directionofthesis` dr ON (dr.`ID` = dp.`directionofthesis_id`) INNER JOIN `specialty` sp ON (sp.`ID` = dr.`group_id`) INNER JOIN `user` uss ON (uss.`ID` = dr.`teacher_id`) WHERE uss.`login` = ? AND dp.`student_id` IS NULL";
connection.query(query,[informationTeacher.login], (err, result)=> { if(!err){
return res.status(200).json(result);
} else {
return res.status(500).json(err);
}
})
})
```

```
router.post('/getForTeacher', auth.authenticateToken,
checkRole.checkRoleTeacher, (req, res)=> {
let informationTeacher = req.body;

let query = "SELECT dp.`ID` as practice_id, dp.`description` as description, us.`Name` as student_name, us.`contact_number` as contact_number, us.`login` as student_email, dr.`name` as directionofthesis_name, sp.`Name` as group_name FROM
`diploma practice` dp INNER JOIN `user` us ON (us.`ID` = dp.`student_id`) INNER JOIN `directionofthesis` dr ON (dr.`ID` = dp.`directionofthesis_id`) INNER JOIN `specialty` sp ON (sp.`ID` = dr.`group_id`) INNER JOIN `user` uss ON (uss.`ID` = dr.`teacher_id`) WHERE uss.`login` = ?";
connection.query(query,[informationTeacher.login], (err, result)=> { if(!err){
return res.status(200).json(result);
} else {
return res.status(500).json(err);
}
})
})
router.post('/getForStudent', auth.authenticateToken, checkRole.checkRoleStudent, (req, res)=> {
let informationStudent = req.body;
let query = "SELECT uss.`login` as teacher_login, dr.`ID` as directionId, dp.`ID` as ID, dp.`description` as description, dr.`name` as directionofthesis_name, sp.`Name` as group_name, uss.`name` as teacher_name FROM `diploma practice` dp INNER JOIN `directionofthesis` dr ON (dr.`ID` = dp.`directionofthesis_id`) INNER JOIN `specialty` sp ON (sp.`ID` = dr.`group_id`) INNER JOIN `user` uss ON (uss.`ID` = dr.`teacher_id`) WHERE dp.`student_id` is NULL AND dr.`group_id` = ( SELECT gm.`specialty_id` FROM `group_member`gm INNER JOIN `user` us ON (gm.`student_id` = us.`ID`) WHERE us.`login` = ?)";
connection.query(query,[informationStudent.login], (err, result)=> { if(!err){
```

```
return  res.status(200).json(result);
} else {
return  res.status(500).json(err);
}
})
})
router.post('/checkPracticeForStudent',
                                                                    auth.authenticateT
oken, checkRole.checkRoleStudent, (req, res)=> {
let informationStudent = req.body;
let query = "SELECT uss.`login` as teacher_email, dr.`ID` as directionId, dp.`ID` as ID, dp.`description` as description, dr.`name` as directionofthesis_name,  sp.`Name` as group_name, uss.`name` as teacher_name FROM `diploma practice` dp INNER JOIN `directionofthesis` dr ON (dr.`ID` = dp.`directionofthesis_id`) INNER JOIN `specialty` sp ON (sp.`ID` = dr.`group_id`) INNER JOIN `user` uss ON (uss.`ID` = dr.`teacher_id`) WHERE dp.`student_id` = (SELECT `ID` FROM `user` WHERE `login` = ?)";
connection.query(query,[informationStudent.login], (err, result)=> { if(!err){
return res.status(200).json(result);
} else {
return  res.status(500).json(err);
}
})
})
router.post('/setNullForTeacher',
                                                                    auth.authenticateT
oken, checkRole.checkRoleTeacher, (req,res) => {
let diplomapractice = req.body;
console.log(diplomapractice);
```

```
query = "UPDATE `diploma practice` SET `student_id` = NULL WHERE `diploma practice`.`ID` = ?"
connection.query(query, [diplomapractice.id], (err, results) => { if(!err) {
query = "DELETE FROM `conversations` WHERE `conversations`.`diploma_practice_id` = ?"
connection.query(query, [diplomapractice.id], (err, results) => { if(!err) {
query = "DELETE FROM `tasks` WHERE `tasks`.`diplomapractice_id` = ?"
connection.query(query, [diplomapractice.id], (err, results) => { if(!err) {
var mailOptions = {
from: process.env.EMAIL_LOGIN, to: diplomapractice.student_email,
subject: 'Вам надіслали повідомленя з "Система Керування Дипломною Практикою"',
html: '<p><b>Викладач припинив за вами співпрацю. Будь ласка, оберіть собі нову тему!</b></p>'
}
transporter.sendMail(mailOptions, function (error, info) { if(error) {
console.log(error)
} else {
console.log('Email sent: ' + info.response);
}
})
return res.status(200).json({message: "Успішно!"});
} else {
return res.status(500).json(err);
}
})
} else {
```

```
return  res.status(500).json(err);
}
})
} else {
return  res.status(500).json(err);
}
})
})
router.post('/setStudent',              auth.authenticateToken,
                                        checkRole.checkRoleStudent, (req,res) => {
let diplomapractice = req.body; console.log(diplomapractice);
query = "UPDATE `diploma practice` SET `student_id` = (SELECT `ID` FROM
`user` WHERE `login` = ?) WHERE `diploma practice`.`ID` = ?"
connection.query(query,                 [diplomapractice.student_email,
                                        diplomapractice.id], (err, results) => {
if(!err) {
query                   = "INSERT INTO `conversations` (`id`,
                        `created_at`,
`diploma_practice_id`) VALUES (NULL, current_timestamp(), ?);"
connection.query(query, [diplomapractice.id], (err, results) => {
if(!err) {
var mailOptions = {
from:  process.env.EMAIL_LOGIN,   to:  diplomapractice.teacher_login,
ubject: 'Вам надіслали повідомленя з "Система Керування Дипломною Практикою"',
html: '<p><b>Студент `'+diplomapractice.student_email+'` обрав вашу тему -
'+diplomapractice.theme+' </b></p>'
}
transporter.sendMail(mailOptions, function (error, info) { if(error) {
console.log(error)
} else {
```

```
console.log('Email sent: ' + info.response);
}
})
query = "DELETE from `topic_proposal` WHERE `student_id` = (SELECT `ID`
FROM `user` WHERE `login` = ?)";
connection.query(query, [diplomapractice.student_email], (err, results)=> {
if(!err) {
return res.status(200).json({message: "Успішно!"});
} else {
return  res.status(500).json(err);}
})
} else {
return  res.status(500).json(err);
}
})
} else {
return  res.status(500).json(err);
}
})
})


=> {
// add
router.post('/add', auth.authenticateToken, checkRole.checkRoleTeacher, (req,res)

let diplomapractice = req.body; console.log(diplomapractice);
query =         "INSERT INTO     `diploma   practice` (`ID`,
                `directionofthesis_id`,`description`) VALUES (NULL, ?, ?);"
connection.query(query,
```

```
                                                    [diplomapractice.directionofthesi
s_id, diplomapractice.description], (err, results) => {
if(!err) {
return  res.status(200).json({message:  "Успішно!"});
} else {
return  res.status(500).json(err);
}
})
})
// update
router.patch('/update', auth.authenticateToken, checkRole.checkRoleTeacher, (req,
res)=> {
let diplomapractice = req.body;
let query = "UPDATE `diploma practice` SET `directionofthesis_id` = ?,
`description` = ? WHERE `diploma practice`.`ID` = ?"
connection.query(query,[diplomapractice.directionofthesis_id,
diplomapractice.description, diplomapractice.id], (err, result)=> { if(!err){
if (result.afffectedRows == 0) {
return res.status(404).json({message : "Не знайдено такої практики"});} else {
return res.status(200).json({message : "Успішно!"});
}
} else {
return  res.status(500).json(err);
}
})
})
// delete
router.post('/delete', auth.authenticateToken, checkRole.checkRoleTeacher, (req,
res)=> {
```

```
let speciality = req.body;
let query = "delete from `diploma practice` where `diploma practice`.`ID` = ?";
connection.query(query, [speciality.id], (err, result) => {
if(!err) {
return res.status(200).json({message : "Успішно!"})
} else {
return  res.status(500).json(err);
}
})
})
router.post('/getTopicProposal',
                                                    auth.authenticateT
oken, checkRole.checkRoleTeacher, (req, res)=> {
let informationTeacher = req.body;
query = "SELECT dr.`ID` as direction_id, us.`ID` as student_id, uss.`name` as teacher_name, dp.`ID` as practice_id, dp.`description` as description, us.`Name` as student_name, us.`contact_number` as contact_number, us.`login` as student_email, dr.`name` as directionofthesis_name, sp.`Name` as group_name FROM
`topic_proposal` dp INNER JOIN `user` us ON (us.`ID` = dp.`student_id`) INNERJOIN `directionofthesis` dr ON (dr.`ID` = dp.`directionofthesis_id`) INNER JOIN
`specialty` sp ON (sp.`ID` = dr.`group_id`) INNER JOIN `user` uss ON (uss.`ID` = dr.`teacher_id`) WHERE uss.`login` = ?"
connection.query(query,[informationTeacher.login], (err, result)=> { if(!err){
return res.status(200).json(result);
} else {
return  res.status(500).json(err);
}
```

```
})
})
router.post('/setTopicProposal',
                                                    auth.authenticateT
oken, checkRole.checkRoleStudent, (req,res) => {
let diplomapractice = req.body; console.log(diplomapractice);
query          = "INSERT INTO `topic_proposal` (`ID`,
               `directionofthesis_id`,
`description`, `student_id`) VALUES (NULL, ?, ?, (SELECT `ID` from `user` where
`login` = ?));"
connection.query(query,
                                                    [diplomapractice.directio
n_id, diplomapractice.description, diplomapractice.student_login], (err, results) => {
if(!err) {
return  res.status(200).json({message:  "Успішно!"});
} else {
return  res.status(500).json(err);
}
})
})
router.post('/deleteTopicProposole',
                                                    auth.authenticateT
oken, checkRole.checkRoleTeacher, (req,res) => {let diplomapractice = req.body;
console.log(diplomapractice);
query = "delete from `topic_proposal` where `ID` = ?" connection.query(query,
[diplomapractice.id], (err, results) => {
if(!err) {
var mailOptions = {
from:  process.env.EMAIL_LOGIN,   to:  diplomapractice.student_email,
```

```
subject: 'Вам надіслали повідомленя з "Система Керування Дипломною Практикою"',
html: '<p><b>Викладач `'+diplomapractice.teacher_name+'` Відмовив Вам тему - '+diplomapractice.description+'. За таким напрямком - '+diplomapractice.directionofthesis_name+' </b></p>'
}
transporter.sendMail(mailOptions, function (error, info) { if(error) {
console.log(error)
} else {
console.log('Email sent: ' + info.response);
}
})
return res.status(200).json({message: "Успішно!"});
} else {
console.log(err)
return  res.status(500).json(err);
}
})
})
router.post('/approveTopicProposole', auth.authenticateToken,
checkRole.checkRoleTeacher, (req,res) => {
let diplomapractice = req.body; console.log(diplomapractice);
query =  "INSERT INTO  `diploma  practice` (`ID`,
                `directionofthesis_id`,
`description`, `student_id`) VALUES (NULL, ?, ?, ?);"
connection.query(query,
                                [diplomapractice.directio
n_id, diplomapractice.description, diplomapractice.student_id], (err, results) => {
if(!err) {
query  =  "INSERT  INTO  `conversations`  (`id`,
```

```
                    `created_at`,
`diploma_practice_id`) VALUES (NULL, current_timestamp(), ?);"
connection.query(query, [results.insertId], (err, results) => {
if(!err) {
var mailOptions = {
from: process.env.EMAIL_LOGIN, to: diplomapractice.student_email,
subject: 'Вам надіслали повідомленя з "Система Керування Дипломною Практикою"',
html: '<p><b>Викладач `'+diplomapractice.teacher_name+'` Підтвердив Вам тему - '+diplomapractice.description+'. За таким напрямком - '+diplomapractice.directionofthesis_name+' </b></p>'
}
transporter.sendMail(mailOptions, function (error, info) { if(error) {
console.log(error)
} else {
console.log('Email sent: ' + info.response);
}
})
query = "DELETE from `topic_proposal` WHERE `student_id` = (SELECT `ID` FROM `user` WHERE `login` = ?)";
connection.query(query, [diplomapractice.student_email], (err, results)=> {

if(!err) {
return res.status(200).json({message: "Успішно!"});
} else {
return res.status(500).json(err);}
})
} else {
return res.status(500).json(err);
```

```
}
})
} else {
return  res.status(500).json(err);
}
})
})
router.post('/generateReport', (req, res) => { var data = req.body;
console.log(data); var query = "";
if (data.typeReport == 'allPractice') query = 'SELECT sp.`Name` as `Група`, us.`Name` as `Студент`, uss.`Name` as `Викладач`, dr.`name` as `Напрямок`, dp.`description` as `Тема` FROM `diploma practice` dp INNER JOIN `user` us ON (us.`ID` = dp.`student_id`) INNER JOIN `directionofthesis` dr ON (dr.`ID` = dp.`directionofthesis_id`) INNER JOIN `specialty` sp ON (sp.`ID` = dr.`group_id`) INNER JOIN `user` uss ON (uss.`ID` = dr.`teacher_id`) WHERE sp.`ID` = ?';
if (data.typeReport == 'nullStudent') query = 'SELECT sp.`name` as `Група`, us.`name` as \'ФІО\', us.`contact_number` as `Контактний номер`, us.`login` as `Електрона Адреса` FROM `user` us INNER JOIN `group_member` gm ON (us.`ID`
= gm.`student_id`) INNER JOIN `specialty` sp ON (sp.`ID` = gm.`specialty_id`) LEFT JOIN `diploma practice` dp ON (us.`ID` = dp.`student_id`) WHERE dp.`student_id` IS NULL AND us.`user_type` = \'1\' AND sp.`ID` = ?';
connection.query(query, [data.group_id], (err, results) => { if (err) {
return res.status(500).send('Помилка');
}
const ws = XLSX.utils.json_to_sheet(results); const wb = XLSX.utils.book_new();
XLSX.utils.book_append_sheet(wb, ws, 'Data'); const filePath = path.join(_dirname,
'data.xlsx'); XLSX.writeFile(wb, filePath); res.download(filePath, 'data.xlsx', (err) =>
{
if (err) {
```

```
console.log('Error in file download:', err);
}

fs.unlinkSync(filePath);
});
});
});
module.exports = router
```