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

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

"На правах рукопису"
УДК_____

«До захисту допущено»
Завідувач кафедри
_____ Наталія АУШЕВА
“___”_____2023р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою
“Цифрові технології в енергетиці”
зі спеціальності 122 “Комп’ютерні науки”

на тему “Веб-платформа для керування процесами розробки університетських проектів”

Виконав: студент 2 курсу, групи ТР-з21мп
Кадирбеков Ільяс Юсуфович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник професор, д.т.н., Костянтин П’ЯНИХ _____
(посада, вчене звання, науковий ступінь, ім’я ПРІЗВИЩЕ) (підпис)

Рецензент_____ _____
(посада, вчене звання, науковий ступінь, ім’я ПРІЗВИЩЕ) (підпис)

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

Київ — 2023

Національний технічний університет України
“Київський політехнічний інститут ім. Ігоря Сікорського”

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ
Рівень вищої освіти другий (магістерський)
За освітньою програмою “Цифрові технології в енергетиці”
Спеціальності 122 Комп’ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Наталія АУШЕВА
(підпис)

«____» _____ 2023 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ

Кадирбекову Ільясу Юсуфовичу
(прізвище, ім’я, по батькові)

1. Тема дисертації «Веб-платформа для керування процесами розробки університетських проектів»

Науковий керівник П’яних Костянтин Євгенович, д.т.н., професор
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “08” листопада 2023 року №5202-с

2. Строк подання студентом дисертації 18 грудня 2023р

3. Вихідні дані до роботи мова програмування JavaScript, платформа Node.js, середовище розробки WebStorm, фреймворк React.js.

4. Перелік питань, які потрібно розробити можливість реєстрації, авторизації та створення проектів, створення модулів власника проекту та користувача проекту та з’єднання їх, комунікація на проекті.

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

6. Орієнтований перелік публікацій V Міжнародна науково-практична інтернет-конференція “Integration of Education, Science and Business in Modern Environment: Summer Debates”, 3-4 серпня 2023.

7. Дата видачі завдання «24» жовтня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	строки виконання етапів магістерської дисертації	Примітка
1	Затвердження теми роботи	14.03.23	Вик.
2	Вивчення та аналіз задачі	20.03.23-31.08.23	Вик.
3	Розробка архітектури та загальної структури системи	01.09.23-10.09.23	Вик.
4	Розробка структур окремих підсистем	11.09.23-22.09.23	Вик.
5	Програмна реалізація системи	23.09.23-12.10.23	Вик.
6	Оформлення пояснювальної записки	28.11.23-08.12.23	Вик.
7	Захист програмного продукту	24.10.23	Вик.
8	Передзахист	04.10.23	Вик.
9	Захист	03.01-10.01.24	Вик.

Студент _____ Ільяс КАДИРБЕКОВ
(підпис) (ім’я ПРІЗВИЩЕ)

Науковий керівник _____ Костянтин П’ЯНИХ
(підпис) (ім’я ПРІЗВИЩЕ)

РЕФЕРАТ

Актуальність теми. Сучасний світ переживає стрімкий розвиток технологій. Це створює великий попит на технічних спеціалістів, які мають практичний досвід у програмуванні та розробці програмного забезпечення. Платформа дозволяє студентам отримати цей досвід, підвищуючи їхню привабливість для роботодавців. Також платформа сприяє співпраці між університетами, студентами та потенційними роботодавцями. Це може сприяти створенню більш пристосованих до потреб ринку праці програм та проектів.

Мета роботи — розробити та впровадити веб-платформу, що сприяє співпраці студентів та викладачів для участі в університетських проектах.

Завдання дослідження:

- проаналізувати потреби студентів та викладачів, створити порівняльний аналіз існуючих платформ, проаналізувати технологічний стек;
- розробити програмну систему для керування процесами розробки університетських проектів;
- протестувати прототип платформи та оцінити вплив платформи;
- провести апробацію розробленої програмної системи, експериментально довести її коректність.

Об’єкт дослідження — система управління університетськими проектами.

Предмет дослідження — розробка та впровадження веб-платформи для управління університетськими проектами, яка об’єднує студентів та викладачів для спільної роботи над проектами, спрощуючи їхню взаємодію та полегшуючи процес управління.

Практична цінність отриманих в роботі результатів полягає в покращенні управління університетськими проектами, забезпечуючи більш ефективну комунікацію та співпрацю між студентами та викладачами. Студенти отримають можливість набувати практичний досвід у реальних проектах, що покращить їхню

підготовку для майбутніх кар’єрних викликів. Успішна реалізація і використання платформи підвищить репутацію університету через залучення у цей процес студентів та професорського складу.

Апробація результатів дисертації. Основні положення даної роботи доповідались та обговорювались на:

V Міжнародній науково-практичній інтернет-конференції “Integration of Education, Science and Business in Modern Environment: Summer Debates”, 3-4 серпня 2023.

Дисертація складається з вступу, п’яти розділів та висновків. Повний обсяг дисертації складає 98 сторінок, в тому числі 92 сторінка основного тексту, 5 таблиць, 14 рисунків, 2 сторінки списку використаних джерел у кількості 30 найменувань.

Ключові слова: університетські проекти, веб-платформа для розробки, управління проектами, студентські проекти, співпраця університету, практичне навчання, освітній процес.

ABSTRACT

Actuality of theme. The modern world is experiencing a rapid development of technologies. This creates a high demand for technical specialists who have hands-on experience in programming and software development. The platform allows students to gain this experience, increasing their attractiveness to employers. The platform also facilitates cooperation between universities, students and potential employers. This can contribute to the creation of programs and projects more adapted to the needs of the labor market.

The purpose of the work is to develop and implement a web platform that facilitates the cooperation of students and teachers to participate in university projects.

Objectives of the study:

- analyze the needs of students and teachers, create a comparative analysis of existing platforms, analyze the technology stack;
- develop a software system for managing university project development processes;
- test the prototype of the platform and evaluate the impact of the platform;
- test the developed software system, experimentally prove its correctness.

The object of the study is the university project management system.

The subject of the study is the development and implementation of a web platform for university project management, which unites students and teachers to work together on projects, simplifying their interaction and facilitating the management process.

The practical value of the results obtained in the work is to improve the management of university projects, ensuring more effective communication and cooperation between students and teachers. Students will have the opportunity to gain practical experience in real projects, which will improve their preparation for future career challenges. The successful implementation and use of the platform will increase

the university's reputation through the involvement of students and professors in this process.

Approbation of the results of the dissertation. The main provisions of this work were reported and discussed at:

V International Scientific and Practical Internet Conference "Integration of Education, Science and Business in Modern Environment: Summer Debates", August 3-4, 2023.

The dissertation consists of an introduction, five chapters and conclusions. The full volume of the dissertation is 98 pages, including 92 pages of the main text, 5 tables, 14 figures, 2 pages of the list of used sources in the amount of 30 names.

Keywords: university projects, web platform for development, project management, student projects, university cooperation, practical training, educational process

ЗМІСТ

ВСТУП 10
1 ПОСТАНОВКА ЗАДАЧІ ДИСЕРТАЦІЇ 14
Висновки до розділу 1 19
2 РЕАЛІЗАЦІЯ ТА ПОБУДОВА РОЗРОБЛЕНОЇ СИСТЕМИ 20
2.1 Опис аналогічних систем 21
2.1.1 Система GitHub Classroom 22
2.1.2 Система Moodle 23
2.1.3 Порівняння існуючих систем з розробленим забезпеченням 24
2.2 Опис програмних засобів 25
2.2.1 Середовище розробки WebStorm 26
2.2.2 Мова програмування JavaScript 27
2.2.3 Бібліотека React.js 29
2.2.4 Фреймворк Next.js 32
2.2.5 Мова програмування TypeScript 33
2.2.6 Платформа Node.js. Фреймворк Nest.js 35
2.2.7 Протокол WebSocket 37
2.2.8 База даних MongoDB 38
2.3 Опис принципів розробки 40
2.3.1 Принципи розробки SOLID 40
2.3.2 Принципи ООП 45
2.4 Опис методів та алгоритмів 50

2.4.1 Алгоритм сортування QuickSort 50
2.4.2 Алгоритм шифрування bcrypt 52
Висновки до розділу 2 53
3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ 54
3.1 Серверна частина розробленої системи 54
3.2 Клієнтська частина розробленої системи 60
Висновки до розділу 3 63
4 РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ 65
4.1 Інсталяція програмного забезпечення 65
4.2 Вимоги до обчислюваної техніки 66
4.3 Демонстрація функціоналу програмної системи 67
Висновки до розділу 4 76
5 РОЗРОБЛЕННЯ СТАРТАПУ ДЛЯ КЕРУВАННЯ ПРОЦЕСАМИ РОЗРОБКИ УНІВЕРСИТЕТСЬКИХ ПРОЕКТІВ 78
5.1 Опис ідеї стартапу 79
5.2 Технологічний аудит ідеї проекту 84
5.3 Аналіз ринкових можливостей запуску стартап-проекту 86
Висновки до розділу 5 90
ВИСНОВКИ 91
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 93
ДОДАТОК А 95

ВСТУП

Сучасний університетський освітній процес необхідно постійно модернізувати та адаптувати до вимог ринку праці та індустрії інформаційних технологій. Однак однією з найбільших викликів для багатьох студентів залишається відсутність можливості набути практичний досвід у сфері розробки програмного забезпечення протягом навчання. Завданням студентів університетів є не лише теоретичне вивчення дисциплін, але і можливість застосування здобутих знань у практиці, співпраця в команді, інтерактивне вирішення реальних завдань.

У цьому контексті, велике значення набуває створення веб-платформи для керування процесами розробки університетських проектів. Ця дипломна робота присвячена розробці такої платформи, яка сприятиме покращенню інтерактивної взаємодії між студентами та викладачами, а також надасть можливість студентам здобути практичний досвід у розробці програмного забезпечення в рамках університетських проектів.

Розробка веб-платформи для керування університетськими проектами у сфері програмної розробки відповідає актуальній потребі університетського освітнього процесу. Головна проблема полягає у недостатній можливості студентів отримати практичний досвід у програмуванні під час навчання, який є ключовим для їхньої професійної підготовки. Наявні рішення не завжди забезпечують можливість ефективної інтеграції практичних знань у навчальний процес. Крім того, відсутність засобів, що б дозволяли студентам активно брати участь у реальних проектах, обмежує їхні можливості для практичного застосування набутих знань. Шляхом критичного аналізу і порівняння існуючих рішень з потребами студентів у практичній підготовці, стає очевидною необхідність в розробці веб-платформи, яка забезпечить ефективну співпрацю, навчання та практичний розвиток студентів у сфері програмного забезпечення.

Головною метою цього дослідження є створення інноваційної веб-

платформи, яка дозволить студентам реєструвати та брати участь у реальних проектах, які здійснюються на базі їхнього навчального закладу. Платформа дозволить студентам та викладачам співпрацювати, формувати команди для розробки проектів, вести процес розробки, а також отримувати доступ до інформації про стан проекту, комунікацію між учасниками та інші засоби для ефективної роботи над програмними продуктами.

Завдання дослідження включають:

- аналіз потреб: дослідження потреб студентів та викладачів університету щодо практичної підготовки у програмній сфері;
- огляд існуючих рішень: критичний огляд наявних веб-платформ для управління проектами та їх адаптація до вимог університетського середовища;
- співробітництво студентів та викладачів: розробка механізмів співпраці між студентами та викладачами у процесі навчання та реалізації проектів;
- технічна реалізація: розробка та імплементація функціоналу веб-платформи, що відповідає виявленим потребам;
- оцінка ефективності: проведення тестування та оцінка роботи платформи для визначення її відповідності вимогам та можливостей для покращення.

Об’єктом дослідження є процес розробки та управління університетськими проектами у сфері програмної розробки. Цей процес породжує проблемну ситуацію у навчальному середовищі та є предметом уваги даного дослідження.

Предмет дослідження магістерської дисертації — розробка та впровадження веб-платформи для керування університетськими проектами з метою підвищення практичної підготовки студентів у програмній сфері. Конкретно, фокус дослідження ускладнюється в аспекті реалізації співпраці студентів та викладачів, формування команд для проектної діяльності та ефективного управління процесами розробки.

Для досягнення поставленої мети використовуються два алгоритми. Використання алгоритму QuickSort для сортування проектів за різними характеристиками (час розробки, складність, статус) сприяє швидкому

відображенню даних для користувачів. QuickSort є швидким та ефективним алгоритмом сортування, що дозволяє швидко організовувати дані за певними параметрами, полегшуючи їхнє відображення на веб-платформі. Використання алгоритму шифрування bcrypt для зберігання облікових даних користувачів є важливим з точки зору безпеки. Алгоритм bcrypt використовує унікальні хеш-функції та “сіль” для створення хешів паролів, ускладнюючи процес розшифрування навіть у випадку злому бази даних. Це забезпечує високий рівень захисту облікових записів користувачів та унеможливлює зловмисникам доступ до конфіденційної інформації. Обидва вищезгадані методи використовуються для забезпечення ефективності, швидкості та безпеки платформи. QuickSort оптимізує відображення проектів за різними характеристиками, а bcrypt гарантує безпечне зберігання облікових даних користувачів, забезпечуючи високий рівень захисту від несанкціонованого доступу.

Результатом цього дослідження є створення функціональної та зручної веб-платформи, яка не тільки полегшить доступ студентів до реальних проектів, але й сприятиме підвищенню їхнього професіоналізму в сфері програмної розробки. Крім того, платформа може відкрити нові можливості для університетів у реалізації інноваційних проектів та співпраці з індустрією.

У роботі буде представлено аналіз сучасного стану освітнього процесу в галузі програмної розробки, визначені основні вимоги та функціональні можливості платформи, розглянуті аспекти безпеки даних, а також способи впровадження та подальшого розвитку цього інноваційного проекту. Дипломна робота “Веб-платформа для керування процесами розробки університетських проектів” спрямована на створення реального інструменту для підтримки студентів у їхньому активному навчанні та розвитку, а також на підвищення якості університетської освіти в сфері програмної розробки.

Отримані результати дослідження мають значний практичний вплив на багато аспектів, зокрема: покращення університетських проектів, розвиток навичок студентів, ефективне керування проектами, стимулювання інновацій.

Рекомендації щодо практичного застосування результатів дослідження включають створення адаптивної та інтуїтивно зрозумілої платформи для максимальної доступності користувачів, впровадження механізмів заохочення активності учасників та постійне оновлення функціоналу з урахуванням потреб користувачів. Також важливо підтримувати безпеку та конфіденційність даних користувачів через застосування найсучасніших методів шифрування та забезпечення надійності та стабільності платформи.

Основні положення даної роботи доповідались та обговорювались на V Міжнародній науково-практичній інтернет-конференції “Integration of Education, Science and Business in Modern Environment: Summer Debates”, 3-4 серпня 2023.

Робота складається зі вступу, 5 розділів, висновків до розділів, загальних висновків, списку використаних джерел, списку джерел ілюстративного матеріалу, 1 додатку на 8 сторінках. Загальний обсяг дисертації — 98 сторінки. Основний зміст викладено на 92 сторінках. Роботу проілюстрован 5 таблицями, 14 рисунками.

1 ПОСТАНОВКА ЗАДАЧІ ДИСЕРТАЦІЇ

Об’єкт: Веб-додаток для керування процесами розробки університетських проектів.

Предмет: Моделі та методи розробки програмного продукту.

Мета: Створити інноваційну веб-платформу, яка сприятиме ефективному керуванню та співпраці університетських проектів, а також надасть студентам можливість здобути практичний досвід у розробці програмного забезпечення в рамках їхнього навчання.

Розглянемо дев’ять задач, які потрібно розв’язати для подолання мети:

1. Розробити інтерфейс користувача.

Цей додаток має добре продуманий інтерфейс для реєстрації та авторизації користувачів, забезпечуючи зручність та безпеку. Реєстраційна сторінка надає можливість користувачам заповнити особисті дані, такі як ім’я, прізвище, курс, факультет, інформацію про себе та посилання на роботу або проект. Крім того, ця сторінка включає вибір ролі в системі (студент чи викладач) та автоматичне створення профілю відповідно до обраної ролі.

Створення та редагування проектів на цій платформі реалізоване через сторінку з докладними можливостями налаштування проекту. Користувачі мають можливість обирати тип проекту, додавати детальний опис завдання, встановлювати вимоги до учасників, вказувати інформацію про команду, терміни виконання та інші важливі характеристики. Ця сторінка оптимізована для легкого та зручного використання, надаючи учасникам максимум можливостей для управління проектами та взаємодії у команді.

Основна ідея цього інтерфейсу — створення доступного середовища для користувачів будь-якого рівня, щоб вони могли ефективно працювати з системою, забезпечуючи максимальну простоту та зручність у використанні.

2. Реєстрація та аутентифікація користувачів.

Користувацька реєстрація на платформі охоплює як студентів, так і

викладачів, надаючи можливість обирати відповідну роль у системі згідно їхніх потреб. Процес реєстрації детально налаштований, забезпечуючи зручне заповнення особистих даних та вимагаючи лише необхідної інформації. Приділено особливу увагу захисту та збереженню конфіденційності персональних даних кожного користувача.

Механізми шифрування та використання передових захисних протоколів впроваджені для забезпечення безпеки даних у надійному середовищі. Усі особисті дані обробляються згідно з вимогами законодавства з охорони особистих даних, що гарантує повну конфіденційність та безпеку для кожного користувача платформи.

3. Управління проектами.

Система управління проектами орієнтована на розширений пошук та фільтрацію доступних проектів. Користувачі можуть використовувати різні параметри, такі як тип проекту, його складність, використання певних технологій або предметну область для ефективного пошуку відповідних проектів. Крім того, система надає можливість створювати команди для спільної розробки проектів та взаємодії між учасниками команд.

Проектна система заснована на детальному аналізі та обробці інформації, що дозволяє користувачам ефективно відфільтрувати проекти за конкретними параметрами. Користувачі зможуть створювати команди та співпрацювати для успішної реалізації обраних проектів, що сприятиме ефективному об’єднанню зусиль та досягненню спільних цілей.

4. Моніторинг та комунікація.

Метою модуля моніторингу та комунікації є створення повноцінного інструментарію для систематичного відстеження та контролю над процесом розробки проекту. Цей інструмент забезпечує систему управління завданнями, статуси завдань та їх виконавців, а також строкові рамки для виконання кожного завдання.

У режимі реального часу відображаються дані про прогрес у виконанні

завдань, дозволяючи всім учасникам бути на одній хвилі та вчасно реагувати на можливі зміни. Окрім цього, платформа надає можливість обміну повідомленнями та взаємодії між учасниками команди та іншими учасниками проекту. Це сприяє не лише ефективній роботі, а й сприяє побудові зручного спілкувального середовища для обговорення ідей, вирішення можливих проблем та спільної роботи над проектом.

5. Безпека.

У цьому розділі обговорюється важливість захисту платформи від можливих загроз та небезпек для конфіденційності даних користувачів та всієї системи в цілому. Передбачається застосування широкого спектру заходів безпеки, таких як криптографічне забезпечення для захисту особистих даних користувачів, виявлення можливих шляхів проникнення, а також встановлення системи моніторингу для вчасного виявлення та реагування на будь-які спроби порушення безпеки платформи.

Крім того, розроблено систематичне проведення аудиту безпеки для виявлення потенційних вразливостей та здійснення відповідних заходів для їх виправлення. Великий акцент зроблено на постійний моніторинг та аналіз активності користувачів для реагування на будь-які підозрілі або ненормальні дії, що можуть загрожувати безпеці системи.

Усі ці заходи спрямовані на забезпечення високого рівня безпеки, недопущення можливих загроз та збереження даних користувачів у безпечному середовищі.

6. Розгортування та тестування.

На етапі розгортання та тестування платформи на тестовому сервері відбувається інтенсивне тестування кожного функціонального блоку і компонента системи. Основна мета полягає не лише в підтвердженні відповідності функціональності заявленим вимогам, але й в ідентифікації можливих уразливостей та аномалій у роботі платформи.

Під час тестування на витривалість важливо забезпечити, щоб система

функціонувала стабільно в різних сценаріях навантаження та обставинах. Це включає оцінку роботи системи під високими навантаженнями, різними об’ємами даних та різноманітними умовами експлуатації.

Завданням цього етапу є забезпечення того, що система буде готова до фактичного впровадження, має високу надійність і стійкість до незапланованих ситуацій, що можуть виникнути у процесі експлуатації. Тестування відповідності функціональності специфікаціям допоможе у виявленні та виправленні будь-яких неузгоджень, а тестування на витривалість дозволить переконатися у стабільності системи у різних умовах роботи.

7. Документація.

На цьому етапі планується створення докладної та повноцінної документації, яка охопить всі аспекти встановлення, налаштування та подальшого використання платформи для користувачів. Ця документація включає в себе не лише технічні деталі, а й пошагові інструкції, які допоможуть новим користувачам ознайомитися з усіма можливостями платформи та почати роботу з нею з легкістю.

Документація побудована у формі, що легко сприймається, з уважним розгортанням всіх ключових відомостей. Користувачі мають доступ до інструкцій, які допомагають їм розібратися у всіх етапах встановлення та функціонування платформи.

Крім того, ця документація постійно оновлюється, враховуючи відгуки користувачів, нові можливості та покращення платформи. Мета полягає у забезпеченні максимально доступного та зрозумілого інформаційного ресурсу для користувачів, який стимулює продуктивне використання платформи.

8. Публікація та впровадження.

Після завершення технічного розгортання на основному сервері та запуску платформи, важливим етапом є її впровадження в університетську спільноту. Це не лише про встановлення доступу користувачам, але й про активний процес адаптації й ознайомлення з нововведеннями.

Проведення впровадження передбачає ряд заходів: рекламні кампанії, тренінги для користувачів, вебінари чи навіть демонстрації можливостей платформи на відкритих заходах. Організація подій, спрямованих на презентацію та показ можливостей платформи, допоможе привернути увагу до нової системи та створити позитивний імідж серед користувачів.

Також важливим аспектом є підтримка та навчання користувачів, особливо тих, хто раніше не мав досвіду з подібними системами. Створення довідкової документації, інструкцій та проведення тренінгів може сприяти більш швидкому та якісному освоєнню платформи, що стане ключовим чинником для успішного використання цієї системи в університетському середовищі.

9. Моніторинг та підтримка.

На підтримку функціонування платформи, планується систематичне оновлення, спрямоване на збереження її ефективності та безпеки. Регулярні оновлення дозволять враховувати та імплементувати нові технологічні вдосконалення. Моніторинг роботи платформи надасть можливість оперативно реагувати на будь-які аномалії або недоліки, що можуть виникнути.

Збирання відгуків користувачів є ключовою складовою для вдосконалення платформи. Аналізуючи отриману зворотню інформацію, можна ідентифікувати та вирішувати виявлені недоліки, а також впроваджувати нові функції, щоб забезпечити максимальне задоволення потреб користувачів. Цей постійний процес удосконалення ґрунтується на активному спілкуванні зі спільнотою користувачів та гнучкій реакції на їхні потреби.

Крім того, надання оперативної та ефективної підтримки користувачам є пріоритетним завданням. Це включає надання вичерпних консультацій та вирішення будь-яких проблем чи питань, що виникають у процесі використання платформи. Такий підхід до підтримки користувачів дозволяє забезпечити їм безперебійний та комфортний досвід взаємодії з платформою.

Висновки до розділу 1

Мета проекту — створення інноваційної веб-платформи для керування університетськими проектами, спрямованої на покращення співпраці між студентами та викладачами, а також надання студентам можливості отримати практичний досвід у розробці програмного забезпечення.

Задачі проекту включають в себе розробку інтерфейсу користувача, реєстрацію та аутентифікацію користувачів, управління проектами, моніторинг та комунікацію, забезпечення безпеки, розгортування та тестування, публікацію та впровадження, а також моніторинг та підтримку.

Технологічний стек проекту включає такі компоненти, як React.js для фронтенду, Nest.js для бекенду, MongoDB для бази даних, GitHub для CI/CD та контролю версій, Docker для контейнеризації, Heroku для розміщення та хмарна платформа для розгортання веб-додатку.

Успішне виконання проекту передбачає реалізацію усіх вищезазначених завдань, забезпечення безпеки та конфіденційності даних користувачів, публікацію платформи та її впровадження в університетську спільноту, а також подальший моніторинг та підтримку.

Проект має великий потенціал для поліпшення університетського середовища, сприяючи співпраці та навчанню студентів в галузі розробки програмного забезпечення. Наявність чіткої постановки завдання та визначеного технологічного стеку допоможе здійснити ефективну розробку та розгортання платформи.

2 РЕАЛІЗАЦІЯ ТА ПОБУДОВА РОЗРОБЛЕНОЇ СИСТЕМИ

Розділ “Реалізація та побудова розробленої системи” присвячений детальному аналізу ключових компонентів, функціональності та технологічних аспектів нашого проекту. Цей розділ відображає технічну сутність та структуру системи, розкриває принципи її функціонування та методи, використані при розробці.

У першому підрозділі “Опис аналогічних систем” дозволить порівняти розроблену систему з існуючими аналогами. Зосередимось на ключових відмінностях та перевагах розробленого рішення порівняно з існуючими аналогами.

У підрозділі “Опис програмних засобів” докладно розглянемо програмні складові системи, вказуючи на їхню роль у функціонуванні системи та їхню взаємодію між собою. Від кожного програмного засобу до реалізації певної функціональності — стежимо за кожним елементом, який забезпечує стабільність та ефективність системи.

У третьому підрозділі “Опис принципів розробки” подамо висновки щодо основних принципів, які визначили архітектуру та структуру системи. Розкриємо концептуальні підходи, що використовувалися під час проектування та розробки, а також основні методики, що лягли в основу роботи.

“Опис методів та алгоритмів” завершить цей розділ, подаючи вичерпну інформацію про використані методи, алгоритми та підходи під час розробки системи. Розкриємо їхню роль та вплив на функціональність та продуктивність системи.

Цей розділ створений для більш глибокого розуміння технічних аспектів проекту та його основних складових, щоб забезпечити повноту та ясність його представлення.

2.1 Опис аналогічних систем

У сучасному освітньому середовищі університети та навчальні заклади активно впроваджують інноваційні підходи до навчання та підготовки студентів, надаючи їм можливість активно займатися практичною роботою і розвивати навички в сфері розробки програмного забезпечення. Один з ефективних способів досягнення цієї мети — впровадження веб-платформ для керування процесами розробки університетських проектів.

У цьому контексті, огляд аналогічних систем для керування проектами має велике значення. Це допомагає розробникам та освітнім закладам визначити оптимальний набір функцій, інструментів та можливостей, які мають бути включені до власного веб-додатку для кращого задоволення потреб студентів та викладачів у керуванні проектами.

У цьому розділі розглянемо декілька аналогічних систем, які вже використовуються в університетському середовищі або можуть бути адаптовані для використання в цілях навчання та керування проектами. Розглянемо їх переваги, недоліки, основні функції та можливості, що вони пропонують.

Зокрема, цей вступ допомагає поглибити розуміння того, які конкретно функції та можливості веб-платформ для управління проектами важливі для студентів, викладачів та університетських програм загалом. Ретельний аналіз аналогічних систем дозволяє ідентифікувати потреби освітнього середовища та зважити на найкращі практики, щоб створити оптимальний та пристосований до потреб інструмент для поліпшення навчального процесу та розвитку навичок у сфері програмування.

Крім того, це дослідження може послужити основою для розробки нових підходів до організації та проведення практичних занять у сфері інформаційних технологій. Залучення студентів до реальних проектів за допомогою цієї платформи може стати важливим інструментом для формування їхньої практичної експертизи та підготовки до майбутньої кар'єри у сфері IT.

2.1.1 Система GitHub Classroom

GitHub Classroom — це інструмент від GitHub, призначений для використання в освітніх закладах та університетах. Він спрощує процес створення та управління завданнями для студентів, особливо в контексті програмування та розробки програмного забезпечення. GitHub Classroom базується на використанні системи контролю версій Git та хостингу коду на GitHub. Основна ідея полягає в тому, щоб надати викладачам інструмент, який дозволить їм створювати завдання, спільно працювати зі студентами над кодом, та оцінювати їх роботу.

Основні характеристики GitHub Classroom:

- інтеграція з GitHub: GitHub Classroom інтегрований з популярною платформою GitHub, що дозволяє студентам та викладачам працювати з кодом у звичному середовищі;
- простота створення завдань: викладачі можуть легко створювати завдання, створювати репозиторії для студентів та призначати завдання групам;
- контроль версій: GitHub Classroom надає можливість викладачам відстежувати зміни у коді студентів та відновлювати попередні версії, що допомагає в оцінці роботи;
- автоматичні перевірки: GitHub Classroom може використовувати автоматичні перевірки, щоб оцінювати завдання, що допомагає викладачам заощадити час;
- спільна робота: студенти можуть спільно працювати над завданнями у командах, використовуючи можливості спільного доступу до репозиторіїв;
- звіти та статистика: GitHub Classroom надає викладачам звіти та статистику щодо активності студентів та результатів оцінювання.

GitHub Classroom становить важливий інструмент для освітніх закладів, оскільки сприяє полегшенню викладання програмування та розробки програмного забезпечення. Ця система не лише забезпечує можливості співпраці та контролю версій, але й надає викладачам інструменти для ефективної оцінки та керування навчальним процесом. Інтеграція з відомою платформою GitHub робить GitHub

Classroom потужним засобом для розвитку навичок учнів у сфері програмування та сприяє покращенню якості навчання. Така система дозволяє вчителям та студентам зосередитись на суттєвих аспектах навчання, забезпечуючи зручну та продуктивну платформу для спільної роботи та навчання.

2.1.2 Система Moodle

Moodle — це відкрите програмне забезпечення для управління навчальним процесом та навчальними матеріалами. Додаток створений для освітніх закладів, викладачів та студентів, щоб спростити процес навчання та навчання онлайн. Moodle надає інструменти для створення віртуальних класів, завдань, тестів, форумів та багато інших функцій для підтримки навчання.

Ключові моменти:

- відкрите програмне забезпечення: Moodle — це відкрите програмне забезпечення з ліцензією GNU GPL, що означає, що можна використовувати його безкоштовно та адаптувати під свої потреби;
- легкий доступ: студенти та викладачі можуть отримати доступ до Moodle з будь-якого пристрою з підключенням до Інтернету, що робить навчання більш доступним та зручним;
- можливості спільної роботи: Moodle надає інструменти для спільної роботи студентів, включаючи форуми для обговорення та спільних завдань;
- оцінювання та звітність: в Moodle є функціональність для створення тестів, завдань та оцінювання студентів, за допомоги якої можна відстежувати академічну продуктивність та генерувати звіти;
- розширюваність: Moodle має велику спільноту користувачів і розробників, яка створює різноманітні розширення та додатки для розширення функціональності;
- адаптивний дизайн: Moodle має адаптивний дизайн, що дозволяє користувачам зручно використовувати платформу на різних пристроях, включаючи смартфони та планшети;

- захист даних: Moodle надає засоби для захисту даних студентів та конфіденційності;
- інтеграція з іншими системами: Moodle може бути легко інтегрований з іншими системами, такими як системи електронного навчання, бібліотечні системи, та інші.

Висновуючи, Moodle виявляється потужним інструментом для сучасної освіти, завдяки своїм універсальним можливостям. Відкритість програми, легкість використання та широкий спектр функціональності роблять його привабливим для університетів та навчальних закладів. Можливості спільної роботи, оцінювання та аналізу, адаптивний дизайн та захист даних — це лише кілька переваг, що роблять Moodle популярним серед освітніх платформ. Його потенціал у поєднанні з іншими системами та гнучкість в розширенні функціоналу роблять цю платформу важливим інструментом для модернізації навчального процесу.

2.1.3 Порівняння існуючих систем з розробленим забезпеченням

Нова система, створена з використанням таких технологій, як React.js, Nest.js Docker, Heroku та MongoDB, являє собою сучасне, добре адаптоване та масштабоване рішення для оптимізації керування процесами розробки університетських проектів.

Ключові моменти:

- простота використання: розроблений додаток має інтуїтивний і простий інтерфейс, що дозволяє новим користувачам легко розпочати використовувати систему без великого навчання;
- спільна робота: додаток розроблений для спільної роботи в команді. Кожен учасник може бачити стан завдань, долучати коментарі, призначати завдання та вносити зміни;
- гнучкість: налаштування програмного додатку під потреби користувача;
- безкоштовна версія: розроблений додаток пропонує безкоштовну версію для особистого користування, що дозволяє спробувати систему без витрат;

- безпека: система забезпечує безпеку даних та може бути використаний як для особистих, так і для корпоративних цілей;
- моніторинг прогресу: можливість відслідковувати, наскільки далеко учасник просунувся у виконанні завдань та проеку;
- уведення дедлайнів: додаток дозволяє встановлювати терміни виконання завдань.

В цьому підрозділі досліджено та порівняно кілька систем управління проектами, спеціально адаптованих для використання в університетському навчальному середовищі. Moodle, GitHub Classroom та нова розроблена система мають свої переваги та унікальні характеристики.

Moodle відкрите програмне забезпечення з широким функціоналом та підтримкою спільної роботи, GitHub Classroom інтегрований з GitHub, спрощуючи спільну роботу над кодом, а розроблена система пропонує сучасне та гнучке рішення з акцентом на простоту використання та спільну роботу.

Кожна з цих систем має свої унікальні переваги, але нова розроблена система надає можливість об’єднати ключові аспекти для досягнення найкращого результату: простота використання, спільна робота, безпека та гнучкість налаштування. Розроблена система є відмінним вибором для управління проектами в університетському середовищі, надаючи студентам та викладачам ефективний та зручний інструмент для співпраці та розвитку навичок у сфері розробки програмного забезпечення.

2.2 Опис програмних засобів

Програмні засоби — це різноманітні програми, інструменти та системи, які використовуються для створення, управління, аналізу, виконання чи підтримки програмного забезпечення. Це можуть бути інтегровані середовища розробки, фреймворки, бібліотеки, бази даних, серверні платформи, різноманітні API,

системи контролю версій, тестувальні засоби та багато іншого.

Ці програмні засоби використовуються розробниками для створення програмного забезпечення, а також для управління, відлагодження, випробування та підтримки розроблених продуктів. Засоби надають різні можливості та функціонал для забезпечення продуктивності та ефективності розробки, управління проектами, аналізу даних та рішень, а також забезпечення безпеки програмних продуктів.

Звичайно, програмні засоби розширюються й удосконалюються відповідно до зростаючих потреб у сфері програмного забезпечення. Це дозволяє розробникам мати доступ до різноманітних інструментів, які відповідають конкретним завданням та технічним вимогам. Від великих платформ до спеціалізованих утиліт, програмні засоби стають ключовою складовою у процесі розробки, сприяючи якісному створенню та оптимізації програмних продуктів.

2.2.1 Середовище розробки WebStorm

WebStorm є однією з провідних інтегрованих середовищ розробки (IDE), яка спеціалізується на веб-розробці та створена компанією JetBrains. Це потужне інструментальне забезпечення, спрямоване на полегшення роботи розробників у написанні, редагуванні і тестуванні веб-додатків та веб-сайтів. WebStorm надає різноманітні інструменти та можливості, що допомагають прискорити та полегшити процес веб-розробки.

Однією з ключових переваг WebStorm є його здатність працювати з різними мовами програмування, зокрема JavaScript, HTML і CSS, і надавати інтегровану підтримку для фреймворків і бібліотек, таких як React.js, Angular, і Vue.js. Інтелектуальне автодоповнення дозволяє швидко написати код, а також вбудовані інструменти для аналізу коду та автоматичного виправлення помилок допомагають у підтримці високої якості написання.

WebStorm має додаткові функції, які полегшують розробку, такі як вбудовані дебагери для виявлення та виправлення помилок, інструменти для

спільної роботи над проектами, включаючи системи контролю версій, а також інструменти для автоматизації рутинних задач, які роблять розробку більш продуктивною.

Ще однією важливою особливістю є те, що WebStorm постійно оновлюється. Компанія JetBrains регулярно випускає нові версії програми з покращеннями та новими функціями, що дозволяє розробникам використовувати найсучасніші інструменти у своїй роботі.

Інтерфейс WebStorm простий та зручний у використанні, що дозволяє розробникам ефективно керувати проектами та фокусуватися на написанні якісного коду. Завдяки величезному спектру можливостей та постійним оновленням, WebStorm залишається одним із популярних виборів серед веб-розробників для створення сучасних та високоякісних веб-додатків.

В цілому, WebStorm виступає як сучасне та високоефективне інтегроване середовище розробки, призначене для веб-розробників. Середовище пропонує розгалужені можливості роботи з різними мовами програмування та фреймворками, забезпечуючи інтуїтивний інтерфейс та ряд інструментів для збільшення продуктивності. Його постійні оновлення та спрощене використання роблять його важливим інструментом для створення якісних веб-додатків.

2.2.2 Мова програмування JavaScript

JavaScript, який часто скорочується як JS, це мова програмування, яка є однією з єдиних технологій всесвітньої мережі. Практично усі веб-сайти використовують JavaScript на клієнтській стороні додатку для поведінки веб-сторінок, часто включають сторонні бібліотеки. Практично усі основні веб-переглядачі мають механізм для виконання JavaScript на будь-яких пристроях користувачів.

По-іншому JavaScript називають мовою скриптів або сценаріїв. Скрипти — це набір інструкцій або команд, які виконуються у момент завантаження сторінки. Браузер самостійно розпізнає код JavaScript, для цього не потрібне навіть

переведення мови програмування у машинний код, тобто компіляції.

JavaScript є однією з найпопулярніших мов програмування у світі програмування. Розроблена в 1995 році компанією Netscape, ця мова відразу ж здобула велику популярність завдяки своїй універсальності та можливостям веб-розробки. JavaScript була визнана як стандарт мови програмування для веб-браузерів, що визначило її основну роль у фронтенді. Давайте розглянемо детальніше цю мову.

JavaScript, часто сплутувана з Java, насправді це зовсім інша мова програмування. Вона є мовою скриптів, яка використовується для додавання функціональності на веб-сторінки. Початково використовувалася для створення простих анімацій чи перевірки форм, але з часом її потужність зросла, і мова стала основою для багатьох веб-застосунків та сайтів.

JavaScript є мовою програмування високого рівня. Ця мова підтримує об’єктно-орієнтований, процедурний та функціональний підходи до програмування, що робить її дуже гнучкою для використання. Одним з переважових аспектів JavaScript є те, що мова програмування інтегрується безпосередньо у HTML і може працювати у будь-якому сучасному браузері.

Ще одна важлива риса JavaScript — це асинхронність. Мова може взаємодіяти з веб-сторінкою, не блокуючи її роботу. Це дозволяє виконувати декілька операцій одночасно, що робить її чудовим вибором для веб-розробки.

JavaScript має широкий спектр застосувань. Від створення інтерактивних елементів у веб-дизайні до розробки повноцінних веб-додатків, ця мова показує себе як потужний інструмент. На сьогоднішній день, завдяки розвитку фреймворків та бібліотек, таких як React.js, Angular і Vue.js, JavaScript використовується для розробки великих масштабів веб-застосунків та односторінкових додатків (SPA).

Однією з ключових особливостей JavaScript є можливість маніпулювання DOM (Document Object Model). Це означає, що розробник може змінювати вміст, стиль та структуру веб-сторінки в реальному часі, що робить JavaScript потужним

інструментом для створення динамічних веб-сторінок.

JavaScript постійно розвивається. Із випуском нових версій EcmaScript (стандарту мови), з'являються нові функції та покращення, що розширюють можливості цієї мови та сприяють її поширенню в інших сферах програмування, таких як розробка серверних застосунків.

У підсумку, усе у світі веб-розробки не уявляється без JavaScript. Ця мова програмування, починаючи як простий інструмент для анімації та перевірки форм, перетворилася на потужний інструмент, що використовується для розробки найскладніших веб-додатків та створення динамічних веб-сторінок. Її гнучкість, асинхронність та невід'ємна роль у веб-розробці роблять JavaScript однією з найпопулярніших та ключових мов програмування у цьому сегменті. Стійке оновлення та розвиток її стандарту дозволяють впроваджувати нові функції та забезпечують постійне зростання її можливостей. JavaScript — це мова, яка тримає руку на пульсі розвитку веб-технологій та продовжує залишатися ключовим інструментом для веб-розробників у всьому світі.

2.2.3 Бібліотека React.js

React.js є найпопулярнішою бібліотекою JavaScript, яка прискорює та спрощує розробку інтерфейсу користувача за допомогою методів швидкого доступу та компонентів, які можна перевикористовувати. Цю бібліотеку можна використовувати для усіх інтерфейсних веб-програм та мобільних додатків.

React.js — це відкрита JavaScript бібліотека, яка використовується для створення користувацьких інтерфейсів. Розроблена компанією Facebook, ця бібліотека дозволяє розробникам будувати веб-додатки з високопродуктивними та ефективними інтерфейсами. React призначений для створення односторінкових додатків (SPA) та динамічних інтерфейсів, забезпечуючи швидкий рендерінг та ефективне управління станом додатку.

Однією з ключових особливостей React.js є використання компонентної архітектури. Ця архітектура дозволяє створювати компоненти, які є

самостійними, повторно використовуваними блоками, які керують своїм власним станом та рендеряться відповідно до зміни цього стану. Це робить розробку та підтримку додатків більш простою та ефективною.

Ще однією важливою концепцією в React є віртуальний DOM (Document Object Model). React.js використовує віртуальний DOM для оптимізації процесу оновлення елементів на веб-сторінці. Замість оновлення всіх елементів при зміні, DOM оновлює тільки ті частини, які дійсно зазнали змін, що забезпечує велику ефективність рендерінгу великих додатків.

React.js також підтримує використання JSX — розширення синтаксису JavaScript, яке дозволяє використовувати HTML-подібний код для опису структури UI компонентів. Це дозволяє розробникам писати код більш зрозуміло та ефективно, що сприяє покращенню розробки та підтримки проектів на React.

Однією з переваг використання React є широкий спектр інструментів та бібліотек, які розробники можуть використовувати для підтримки своїх проектів. Наприклад, Redux дозволяє ефективно керувати станом додатку, а React Router використовується для управління маршрутизацією в односторінкових додатках. Також треба підкреслити, що React.js має компонентну архітектуру, яка дозволяє створювати невеликі та самодостатні компоненти, що полегшує управління станом додатку та поліпшує його масштабованість. Такий підхід дозволяє швидко реагувати на зміни в додатку та сприяє його ефективній розробці. Реалізація віртуального DOM є ще однією перевагою. React порівнює стани реального та віртуального DOM, виявляючи зміни та оновлюючи лише ті частини інтерфейсу, які були змінені. Це підвищує продуктивність та швидкодію додатків, особливо при роботі з великим обсягом даних. Крім того, JSX (JavaScript XML) у React спрощує створення користувацького інтерфейсу, оскільки це розширення синтаксису JavaScript, що дозволяє використовувати синтаксис, схожий на HTML. Це робить код більш читабельним та легким у розумінні. Наявність широкої спільноти розробників, великої кількості розширень та додаткових бібліотек, таких як Redux та React Router, додає гнучкості та розширює можливості React у

створенні різноманітних додатків. React також має велику кількість ресурсів для вивчення. Це включає офіційну документацію, блоги, книги, курси та форуми, що допомагають розробникам поглибитися в цю технологію та вдосконалювати свої навички. Ці переваги роблять React.js потужним та популярним інструментом для розробки сучасних веб-додатків.

Хоча React.js є популярним інструментом для розробки веб-додатків, цей інструмент також має свої недоліки, які можуть впливати на процес розробки та використання цієї бібліотеки. Одним з недоліків React.js є висока крутизна початкового навчання для новачків. Фреймворк вимагає від розробників розуміння концепцій, таких як JSX, віртуальний DOM та компоненти, що може виявитися складним для тих, хто тільки починає. Ще одним аспектом є швидкість зміни технологій та версій React. Це може створювати проблеми з сумісністю при оновленні проекту на новішу версію бібліотеки або при використанні старих функцій, які можуть бути видалені у майбутніх оновленнях. Також, хоча віртуальний DOM покращує продуктивність, іноді може призводити до надмірного використання пам’яті. Для великих додатків або додатків з великою кількістю даних це може стати проблемою. Недоліком React також може бути перевантаження широким спектром варіантів та бібліотек, які пропонуються спільнотою. Це може призвести до вибору неправильного інструменту або невірного підходу для конкретного завдання, що ускладнює розробку. Незважаючи на ці недоліки, з правильним розумінням та використанням, React.js є потужним інструментом для створення високоякісних веб-додатків.

У підсумку, React.js є потужним інструментом для створення інтерфейсів веб-додатків, що відзначається своєю ефективністю та продуктивністю. Бібліотека надає розробникам зручні інструменти, такі як компонентна архітектура, віртуальний DOM та розмітка JSX, що полегшують процес створення веб-додатків. Однак, важливо враховувати деякі недоліки, такі як висока крутизна початкового навчання та можливість проблем із сумісністю при оновленнях. Незважаючи на це, з правильним розумінням та використанням, React.js

залишається потужним інструментом для створення високоякісних веб-додатків.

2.2.4 Фреймворк Next.js

Next.js — це популярний фреймворк React.js для створення універсальних веб-додатків, який надає широкий набір інструментів для розробки модерних проектів. Цей фреймворк базується на React.js і забезпечує ряд переваг і можливостей для розробників.

Одна з ключових особливостей Next.js — це можливість використання рендерингу на стороні сервера (SSR) та генерації статичних сторінок (SSG). Це дозволяє оптимізувати продуктивність, забезпечувати кращий SEO та прискорювати завантаження сторінок. Комбінація SSR та SSG робить Next.js ідеальним вибором для створення швидких та ефективних веб-додатків.

Інтегрована підтримка маршрутизації у Next.js є дуже потужним інструментом для роботи з URL-адресами та навігацією. Організація маршрутів дозволяє створювати компоненти, які відповідають конкретним URL, що спрощує навігацію в додатку та дозволяє створювати дружні до користувача URL-адреси.

Next.js також надає можливість збільшити швидкість розробки завдяки автоматичному перевантаженню та гарячій перезавантаженню (hot reloading), що дозволяє відразу бачити зміни в коді без перезавантаження сторінки.

Фреймворк має вбудовану підтримку TypeScript, що дозволяє розробникам писати типізований код, забезпечуючи більшу безпеку та читабельність коду.

Окрім цього, Next.js має ряд плагінів та розширень, які спрощують роботу з асинхронним завантаженням даних, управлінням станом додатку та інтеграцією з різними серверними API.

У цілому, Next.js виступає як важливий інструмент для розробки веб-додатків, пропонуючи широкий набір зручних функцій для розробників. Відмінність у використанні SSR та SSG дозволяє покращити продуктивність та SEO, а підтримка маршрутизації та автоматичного перезавантаження сприяють зручності розробки. Розширення TypeScript та набір плагінів роблять Next.js

гнучким та пристосованим до різних вимог проектів. Загалом, цей фреймворк має потужний набір функцій, що сприяють створенню високоякісних та сучасних веб-додатків.

2.2.5 Мова програмування TypeScript

TypeScript — це мова програмування, що виступає як розширення JavaScript, розроблена компанією Microsoft. Це мова з використанням статичних типів, яка додає строгу типізацію до JavaScript, полегшуючи розробку складних програмних продуктів. TypeScript компілюється у JavaScript та забезпечує розширений функціонал, який дозволяє використовувати нові функції, типи та засоби відладки.

Однією з ключових переваг TypeScript є його система типізації. Вона дозволяє визначати типи даних для змінних, параметрів функцій та інших об’єктів у програмі. Це полегшує виявлення помилок у коді на етапі розробки та покращує його читабельність. Більшість інструментів редагування коду (IDE) підтримують автодоповнення та виявлення помилок, що робить роботу з TypeScript більш продуктивною.

Іншою важливою перевагою є те, що TypeScript підтримує екосистему JavaScript. Він включає в себе всі функції JavaScript, а також дозволяє використовувати будь-які наявні бібліотеки та фреймворки. Це робить TypeScript універсальним для розробки в різних середовищах.

Типи даних у TypeScript можуть бути вбудованими (як ‘number’, ‘string’, ‘boolean’), включаючи спеціальні типи, такі як ‘enum’, ‘void’, ‘any’, ‘null’ та ‘undefined’. TypeScript також підтримує розширену систему типів, дозволяючи розробникам визначати власні складні типи даних.

Хоча TypeScript має безліч переваг, мова програмування також має кілька обмежень, які варто врахувати при його використанні. Одним з недоліків є необхідність часу на вивчення і прийняття концепцій строго типізованих мов програмування. Це може стати проблемою для новачків або для розробників, які

звикли працювати з динамічно типізованими мовами, де не потрібно дбати про оголошення типів змінних. Ще однією проблемою може бути висока вимогливість до організації коду. TypeScript вимагає декларації типів для змінних, функцій та об’єктів, що може призвести до збільшення обсягу коду та часу, витраченого на розробку. Також варто врахувати, що при перетворенні TypeScript у JavaScript можливі деякі втрати продуктивності через додаткові обгортки та перевірки типів, що може вплинути на продуктивність в деяких сценаріях використання. Наприклад, при великому обсязі проекту перевірки типів можуть займати значний час під час компіляції, що призведе до збільшення часу розробки та запуску додатку. Також TypeScript може не завжди автоматично вгадувати типи даних у всіх ситуаціях, що може призвести до неочікуваної поведінки у великих проектах. Незважаючи на ці обмеження, TypeScript продовжує бути актуальним в середовищі розробки завдяки своїм перевагам, і багато з цих недоліків можна уникнути, правильно організовуючи розробку та використовуючи кращі практики при впровадженні цієї мови програмування. Завдяки строгій типізації, великій спільноті користувачів та постійному розвитку, TypeScript став популярним інструментом для розробки великих проектів. Його можливості сприяють покращенню стабільності, безпеки та продуктивності при розробці програмного забезпечення. Використання TypeScript може значно зменшити кількість помилок у коді та полегшити процес обслуговування проектів.

Отже, TypeScript є потужним інструментом для розробки програмного забезпечення, завдяки своїй строгій типізації та здатності підтримувати розширений функціонал JavaScript. Його переваги полягають у зменшенні помилок на етапі розробки, покращенні стабільності та безпеки програм. Проте варто враховувати обмеження, такі як складність вивчення для новачків та певні труднощі з обслуговуванням коду. Оптимальне використання TypeScript дозволяє забезпечити підвищену якість програмних продуктів та покращення ефективності розробки, зокрема великих проектів.

2.2.6 Платформа Node.js. Фреймворк Nest.js

Node.js — це відкрите, серверне середовище виконання JavaScript, побудоване на движку V8, що розроблений Google. Ця платформа дозволяє виконувати JavaScript на сервері, що раніше було характерно тільки для веб-браузерів. Основним призначенням Node.js є створення масштабованих та швидких мережевих додатків.

Node.js використовується для побудови серверів, мережевих додатків, API та інших засобів, спрямованих на роботу з мережею. Вона надає зручний і ефективний спосіб створення додатків, що працюють в реальному часі, наприклад, чати, стрімінгові мережеві додатки або інші застосунки з великою кількістю одночасних підключень.

Однією з головних переваг Node.js є його швидкодія. Використання движка V8, який володіє високою продуктивністю завдяки JIT-компіляції, робить Node.js особливо ефективним для операцій з великим обсягом вводу/виводу (I/O).

Ще однією ключовою перевагою є можливість використання однієї мови програмування (JavaScript) як на клієнтському, так і на серверному рівні, що спрощує розробку та підтримку додатків.

Проте, у Node.js є й деякі недоліки, наприклад, одночасне виконання великої кількості операцій I/O може призвести до блокування основного потоку та зниження продуктивності. Також, через асинхронний характер Node.js, код може стати складнішим через використання зворотних викликів або обіцянь (promises) для управління асинхронними операціями.

У висновку можна сказати, що Node.js є потужним інструментом для розробки серверних додатків, що дозволяє використовувати JavaScript на серверному рівні. Його можливості управління великою кількістю одночасних з’єднань та висока швидкодія роблять його ідеальним для мережевих додатків та операцій з великим обсягом вводу/виводу. Однак існують певні обмеження, такі як можливість блокування основного потоку та складність управління асинхронними операціями, які потребують уважної оптимізації для оптимальної

продуктивності та ефективної роботи програмного забезпечення. Загалом, Node.js є важливим інструментом для створення сучасних та масштабованих серверних додатків.

Nest.js — це прогресивний фреймворк для будівництва ефективних, надійних та масштабованих серверних додатків на основі Node.js. Фреймворк базується на засадах SOLID, архітектурі модульних та уніфікованих елементів, що сприяє створенню добре структурованих та легко розширюваних застосунків.

Однією з ключових особливостей Nest.js є його використання TypeScript, що дозволяє використовувати сучасні підходи до програмування, включаючи типізацію, зручність рефакторингу та більш безпечний код.

Фреймворк пропонує модульну структуру, що сприяє підтримці масштабованості та розширюваності. Це включає в себе використання модулів для групування функцій, що пов’язані за функціональністю, що робить проект більш структурованим та зрозумілим.

Nest.js використовує концепцію провайдерів для впорядкування компонентів додатку. Він також пропонує вбудовану підтримку HTTP роутінгу, модульний підхід до міжпроцесорної комунікації, аутентифікацію та авторизацію через різноманітні стратегії, що спрощує розробку та підтримку додатків.

Крім того, Nest.js вбудовує підтримку тестування за допомогою модульних та інтеграційних тестів, що допомагає забезпечити стабільність та надійність коду.

Однак, для деяких розробників Nest.js може виявитися складним через високий рівень абстракції та певні особливості. Також, через його концепцію, що базується на декораторах та метаданих, процес вивчення може зайняти більше часу.

Nest.js визначається не лише як фреймворк для створення серверних додатків на Node.js, а як ціла екосистема, спрямована на підвищення продуктивності, надійності та зручності розробки програмного забезпечення. Його основні переваги, зокрема використання TypeScript, модульна структура та

підтримка провайдерів, забезпечують високий рівень абстракції та зручності у створенні добре організованих застосунків. Інтересним є те, як Nest.js інтегрує різноманітні компоненти, які часто використовуються у серверних додатках, такі як HTTP роутінг, аутентифікація, модульність та інші, у єдину, логічну структуру, що дозволяє зосередитися на функціоналі самого додатку, не втрачаючи час на організацію складних компонентів. Навіть якщо деякі розробники відзначають певну складність вивчення через високий рівень абстракції та особливості Nest.js, важливо відзначити, що цей фреймворк пропонує новий рівень ефективності та структурованості, що робить його привабливим для багатьох розробників, які шукають надійність, зручність у використанні та здатність розширювати свої додатки. Таким чином, Nest.js не лише відображає технологічний прогрес у світі розробки програмного забезпечення, а й демонструє новий підхід до побудови серверних додатків, що спрощує та оптимізує процес розробки, забезпечуючи при цьому високу якість та стабільність програмного забезпечення.

2.2.7 Протокол WebSocket

WebSocket — це протокол зв’язку, який дозволяє двом або більше пристроям встановлювати з’єднання через мережу Інтернет та обмінюватися даними в реальному часі. Розроблений як розширення HTTP, WebSocket забезпечує можливість бідиректорійного зв’язку між клієнтом та сервером, що дозволяє передавати повідомлення без попереднього запиту від клієнта.

Однією з ключових особливостей WebSocket є постійне з’єднання між клієнтом і сервером, що дозволяє обмінюватися даними в реальному часі без значного навантаження на мережу та сервер. Це особливо корисно для веб-додатків, які вимагають негайної взаємодії між користувачами, таких як чати, онлайн-ігри, фінансові додатки тощо.

WebSocket використовує однакові порти HTTP (80) та HTTPS (443), що дозволяє йому працювати на більшості мереж, оскільки більшість фірм блокує порти, які не використовуються. Він також підтримується більшістю сучасних

веб-браузерів та є стандартом, розробленим і підтримуваним W3C.

У підсумку, WebSocket — це інноваційний протокол зв'язку, що перетворює спосіб, яким веб-додатки взаємодіють між собою та з користувачами. Цей інструмент відкриває нову еру веб-розробки, дозволяючи обмінюватися даними в реальному часі між клієнтом та сервером. Його ключова перевага — постійне з'єднання, що робить можливим спілкування без попередніх запитів, дозволяючи створювати веб-додатки з вищим рівнем інтерактивності, такі як чати, онлайн-ігри та фінансові сервіси. WebSocket забезпечує зручну альтернативу для розв'язання завдань, де миттєва взаємодія користувача з додатком відіграє ключову роль. Однак даний протокол також має важливі переваги з точки зору мережевої стабільності, оскільки використовує стандартні порти HTTP та підтримується більшістю сучасних браузерів. Технологія WebSocket, заснована на розвитку й стандартизації, відображає потужний інструмент у веб-розробці. Її універсальність, зручність та низьке навантаження на мережу дозволяють створювати інтерактивні та високоефективні веб-додатки. Цей протокол стає критичним для веб-сервісів, що прагнуть забезпечити реально часову взаємодію та сприяти новому поколінню веб-інтерфейсів.

2.2.8 База даних MongoDB

MongoDB — це потужна, гнучка та широко використовувана система керування базами даних (СКБД) типу NoSQL, яка базується на концепції документ-орієнтованих баз даних. Розроблена компанією MongoDB Inc., вона надає можливості ефективного зберігання та організації великих обсягів структурованої, напівструктурованої і навіть неструктурованої інформації.

Однією з ключових особливостей MongoDB є використання JSON-подібних документів для зберігання даних, що робить її особливо придатною для роботи з веб-додатками, де дані можуть бути представлені у форматі JSON. Ця база даних може працювати як на одному сервері, так і у режимі розподіленої системи, що дає можливість масштабування при великому обсязі даних та навантаженні.

MongoDB використовує гнучкі колекції документів замість таблиць у традиційних реляційних базах даних. Це дозволяє зберігати різноманітні дані в одному документі, що спрощує її структурування та підтримку. MongoDB також має багатий набір можливостей для роботи з даними, включаючи запити, індексацію, агрегаційні функції та можливість розробки складних операцій обробки даних.

Однією з переваг використання MongoDB є гнучкість в роботі з даними: можливість швидко міняти структуру даних без перерви у роботі системи. Представлена база даних також дозволяє розробникам працювати з великими обсягами неструктурованих даних, які виникають в різних додатках, зберігаючи швидкість доступу до цих даних.

Проте MongoDB також має свої недоліки. Наприклад, ця база даних не підтримує транзакції між кількома колекціями, що може бути проблемою у випадку необхідності змінювати дані в кількох місцях одночасно. Також, збільшення обсягу даних може призвести до збільшення навантаження на сервер та його ресурсів.

У підсумку, MongoDB, з її потужними можливостями та гнучкістю, стала невід’ємною частиною сфери зберігання та обробки даних. Її здатність працювати з різноманітними типами даних у вигляді JSON-подібних документів робить її ідеальним вибором для веб-додатків. MongoDB забезпечує ефективне зберігання та організацію великих обсягів даних, дозволяючи змінювати структуру даних без перерви у роботі системи. Її гнучка структура спрощує управління даними, але одночасно може призвести до перевантаження сервера при збільшенні обсягу даних. Незважаючи на це, MongoDB залишається потужним та важливим інструментом для розробників, що працюють з великими обсягами структурованих та неструктурованих даних, надаючи широкі можливості управління та обробки інформації.

2.3 Опис принципів розробки

Опис принципів розробки — це важлива частина, яка допомагає зрозуміти основні принципи та методи, що лежать в основі створення програмного забезпечення. Цей розділ присвячений розгляданню ключових концепцій, які визначають структуру, архітектуру та процеси розробки програмних продуктів. Від організації коду до вибору підходів у розробці, цей розділ відкриває принципи, які дозволяють створювати ефективні, стабільні та легко розширювані програмні рішення. Вивчення цих принципів не лише допомагає розуміти основи розробки, але й дає змогу вдосконалювати та оптимізувати процес створення програмного забезпечення.

2.3.1 Принципи розробки SOLID

Принципи SOLID в програмуванні — це п’ять основних концепцій, що визначають принципи об’єктно-орієнтованого програмування (ООП). Ці принципи були представлені Робертом Мартіном і його колегами та стали фундаментом для створення якісних, легких у розумінні та підтримці програмних рішень. Давайте розглянемо кожен з них детальніше:

1. Принцип єдиної відповідальності (Single Responsibility Principle — SRP).

Принцип єдиної відповідальності (SRP) є одним з ключових принципів об’єктно-орієнтованого програмування, який визначає, що кожен клас або модуль повинен бути відповідальним лише за одну конкретну функцію або аспект програми. Цей принцип розробки вперше був визначений Робертом Мартіном.
SRP визначає, що кожен об’єкт або клас має мати лише одну причину для зміни. Ідея полягає в тому, щоб розділити функціонал програми на невеликі, незалежні компоненти, кожен з яких відповідає лише за певний аспект функціоналу програми. Це дозволяє полегшити розуміння, розвиток та тестування коду.

Цей принцип застосовується не лише до окремих класів, а й до модулів, функцій, методів та інших структурних елементів програми. Використання SRP

сприяє створенню більш зрозумілих, гнучких та підтримуваних систем.

Розглянемо приклад: веб-додаток, що обробляє замовлення. Існують класи, відповідальні за роботу з користувачами, товаром, замовленнями та оплатою. Застосування SRP означає, що кожен з цих класів буде відповідальний тільки за свій власний аспект: наприклад, клас, який працює з користувачами, не буде надмірно залучений до обробки оплати або замовлення товарів.

Однак SRP також може бути викликаний проблемами при його застосуванні. Недотримання цього принципу може призвести до перевантаження класів або методів занадто багатьма обов'язками, що робить систему складнішою у розумінні та підтримці.

Узагальнюючи, SRP є важливим принципом, який сприяє створенню чистого, зрозумілого та підтримуваного коду, дозволяючи легко масштабувати програму та зменшувати її залежності, забезпечуючи більшу гнучкість у внесенні змін.

2. Принцип відкритості/закритості (Open/Closed Principle — OCP).

Принцип відкритості/закритості (Open/Closed Principle — OCP) є одним із ключових принципів об'єктно-орієнтованого програмування, визначеним Бертраном Майерсом. Цей принцип визначає, що програмні об'єкти або сутності повинні бути відкритими для розширення, але закритими для змін. Грубо кажучи, це означає, що коли потрібно внести зміни до системи, вони повинні відбуватися через розширення, а не зміну вже існуючого коду.

Принцип OCP залежить від уявлення про модулі, які мають бути розроблені таким чином, що забезпечують можливість їх розширення без зміни вихідного коду. Це здійснюється за допомогою створення абстракцій та інтерфейсів, що дозволяють додавати нову функціональність шляхом наслідування чи реалізації нових класів без внесення змін до вже існуючих класів.

Один із способів досягнення відкритості для розширення та закритості для змін полягає в застосуванні патернів проектування, таких як стратегія, декоратор, адаптер та інші. Наприклад, використання стратегії дозволяє створити новий

алгоритм і внести його у виконання, не змінюючи вихідного коду.

Важливість OCP в програмуванні полягає в тому, що він сприяє створенню коду, який менше схильний до помилок та більш гнучкий у розширенні. Відділення існуючої функціональності від нового функціоналу допомагає підтримувати стабільність системи під час внесення змін.

Тим не менш, дотримання принципу OCP може вимагати більшої уваги до структури програми та архітектури системи на початковому етапі проектування, а також викликати додаткові витрати часу на створення абстракцій та інтерфейсів для забезпечення можливості розширення. Однак ці зусилля зазвичай варто витратити, оскільки вони сприяють покращенню підтримки та масштабованості системи з часом.

3. Принцип підстановки Лісков (Liskov Substitution Principle – LSP).

Принцип підстановки Лісков (Liskov Substitution Principle — LSP) є одним з п’яти принципів SOLID і був вперше сформульований Барбарою Лісков. Цей принцип стверджує, що об’єкти повинні бути замінними на свої підтипи, не змінюючи правильність програми. Іншими словами, якщо S є підтипом T, то об’єкти типу T можуть бути замінені об’єктами типу S без будь-яких небажаних наслідків для коректності програми, де T — це батьківський тип, а S — це його підтип.

Цей принцип забезпечує стабільність в програмі і дозволяє використовувати узагальнені типи, не змінюючи коректності роботи програми, що є дуже важливим для поліпшення читабельності та підтримки коду. Якщо програма коректно виконується з об’єктом певного класу, то вона також повинна виконуватися з будь-яким підтипом цього класу без будь-яких змін у логіці програми.

Застосування LSP передбачає, що методи підкласів повинні поводитися так само, як методи батьківського класу, не порушуючи їхньої специфікації. Якщо підкласи змінюють поведінку методів, це може призвести до неправильного функціонування програми, оскільки код, що використовує батьківський клас,

може не очікувати таких змін.

Дотримання принципу LSP сприяє створенню систем з більшою гнучкістю, спрощує тестування та підтримку коду, робить програми більш розширюваними та масштабованими. Однак недотримання цього принципу може призвести до введення помилок, складнощів у розробці та розширенні програмного забезпечення.

Для досягнення принципу LSP важливо ретельно планувати ієрархію класів та дотримуватися концепції спадкування. Ретельна увага до того, як підкласи спадкуються від батьківських класів, є ключовим аспектом впровадження цього принципу в розробку програмного забезпечення.

4. Принцип розділення інтерфейсу (Interface Segregation Principle — ISP).

Принцип розділення інтерфейсу (Interface Segregation Principle — ISP) — це один із п’яти базових принципів SOLID, який розробив Роберт Мартін. Цей принцип стверджує, що клієнти не повинні залежати від інтерфейсів, які вони не використовують. Головна ідея полягає у тому, що інтерфейси повинні бути спеціалізованими, а не загальними, аби мінімізувати залежності і забезпечити впровадження принципу одинарної відповідальності (SRP).

ISP прагне уникнути “жирних” інтерфейсів, де класи змушені реалізувати методи, які вони не використовують. Замість цього, він спонукає до створення більш спеціалізованих інтерфейсів, призначених для конкретних використань, що зменшує залежність клієнтів від зайвого функціоналу.

При використанні цього принципу важливо ретельно розробляти інтерфейси таким чином, щоб вони були чіткими та конкретними для певних завдань. Це дозволяє класам використовувати лише ті методи, які є необхідними для їхньої роботи, уникнувши непотрібного функціоналу. Такий підхід полегшує розширення та рефакторинг, оскільки зміни в інтерфейсах не впливають на всю систему.

Практичне застосування принципу ISP полягає в розробці невеликих та спеціалізованих інтерфейсів, які відповідають конкретним потребам класів. Це

дозволяє створювати більш гнучкі та змінювані системи, які легко адаптуються до нових вимог та зберігають свою стабільність. Однак недотримання цього принципу може призвести до появи “жирних” інтерфейсів, які ускладнюють розуміння та підтримку коду, а також ускладнюють процеси рефакторингу та розширення системи.

5. Принцип інверсії залежності (Dependency Inversion Principle — DIP).

Принцип інверсії залежності (Dependency Inversion Principle — DIP) — це один із п’яти ключових принципів SOLID, сформульований Робертом Мартіном. DIP стверджує, що високорівневі модулі системи не повинні залежати від деталей реалізації низькорівневих модулів; обидва типи модулів повинні залежати від абстракцій. Даний принцип передбачає, що абстракції не повинні залежати від деталей, а деталі мають залежати від абстракцій.

Застосування DIP полягає у створенні абстракцій, які описують функціональність, та використанні цих абстракцій для реалізації різних деталей. Він робить систему більш гнучкою та забезпечує відокремлення високорівневих модулів від деталей їхньої реалізації. Це досягається шляхом створення інтерфейсів та абстракцій, що дозволяють використовувати узагальнені концепції, замість прив’язки до конкретних реалізацій.

Принцип DIP забезпечує спрощення та зменшення залежностей між класами. Він сприяє створенню коду, який легко розширюється та підтримується. Цей принцип допомагає забезпечити відкрите розширення, закрите змінення (принцип O в SOLID) і забезпечує рівень абстракції, що дозволяє використовувати різні реалізації для конкретних завдань.

Наприклад, у веб-застосунках DIP може бути застосований через використання інтерфейсів для специфікації, а потім реалізації різних сервісів, замість прив’язки контролерів напряму до конкретних сервісів. Це дозволяє змінювати конкретні реалізації сервісів, не змінюючи логіку контролерів, що робить систему більш модульною та легко розширюваною.

Принципи SOLID — це важливий набір концепцій, що визначають

фундаментальні принципи об’єктно-орієнтованого програмування. Ці принципи допомагають розробникам створювати більш якісне та гнучке програмне забезпечення, спрощуючи процеси розробки, тестування та підтримки систем.

Принципи, такі як SRP, OCP, LSP, ISP та DIP, кожен в своєму контексті, спрямовані на розділення функціоналу, створення гнучких інтерфейсів, уникнення залежностей від непотрібного функціоналу, забезпечення високої абстракції та розширюваності програмних рішень.

Впровадження цих принципів у розробку дозволяє створювати системи, які легше змінювати та розширювати з часом, спрощуючи процеси додавання нового функціоналу та підтримки існуючого коду. Застосування SOLID сприяє покращенню архітектури програм та робить їх більш стійкими до змін.

2.3.2 Принципи ООП

Об’єктно-орієнтоване програмування (ООП) — це парадигма програмування, яка спирається на концепції об’єктів та їх взаємодію в програмних системах. В основі цієї парадигми лежить моделювання програмних об’єктів як відображення реальних об’єктів чи концепцій. Це дозволяє створювати складні програми, структуровані у вигляді взаємодіючих об’єктів, кожен з яких має свої властивості та можливості.

У своїй сутності, ООП спрямоване на полегшення розробки та підтримки програмних продуктів шляхом організації коду у логічні блоки (об’єкти), які взаємодіють між собою. Це дозволяє розробникам створювати більш швидкі, масштабовані та легко змінювані програми.

Ключові концепції ООП включають в себе класи, об’єкти, спадкування, інкапсуляцію, поліморфізм та абстракцію. Класи визначають структуру об’єкта, а об’єкти є екземплярами цих класів, маючи унікальні властивості та можливості. Спадкування дозволяє створювати нові класи на основі вже існуючих, а інкапсуляція дозволяє обмежувати доступ до певних частин коду.

ООП надає розробникам потужний інструментарій для розв’язання

складних завдань програмування, забезпечуючи зручність, читабельність та підтримку коду від початкового створення до підтримки та розширення програм. Використання ООП дозволяє створювати більш стабільні, модульні та легкі для супроводу програми. Розглянемо наступні чотири принципи.

1. Принцип абстракції

Принцип абстракції є одним із фундаментальних принципів об’єктно-орієнтованого програмування (ООП) і використовується для управління складністю програмних систем. Цей принцип полягає в виділенні найважливіших аспектів чи характеристик об’єкту та ігноруванні деталей, які не є релевантними на даному етапі розробки програми. Абстракція дозволяє моделювати складні системи з точки зору їх взаємодії та функціоналу, приховуючи внутрішню реалізацію.

Цей принцип сприяє створенню ієрархій об’єктів, де кожен рівень абстракції представляє відповідний рівень деталізації. На вищому рівні абстракції упроваджуються загальні поняття та методи, що мають загальне застосування, тоді як на нижчих рівнях уточнюються деталі конкретної реалізації.

Важливим елементом абстракції є використання інтерфейсів та абстрактних класів, які визначають набір методів та властивостей, не вказуючи на їхню конкретну реалізацію. Це дозволяє використовувати ці компоненти відокремлено від їхньої внутрішньої реалізації, сприяючи гнучкості та модульності програми.

Принцип абстракції дозволяє вбудовувати структуру та функціонал програми так, що вона може бути легко розширена та модифікована без необхідності змінювати внутрішню реалізацію. Це забезпечує зручність підтримки, відлагодження та розширення програмних продуктів.

Абстракція також дозволяє розробникам працювати над окремими частинами програми, використовуючи їхні абстрактні описи без необхідності вдаватися у деталі всієї системи. Це сприяє покращенню ефективності розробки та спрощує взаємодію між різними командами, які працюють над різними частинами програми.

2. Принцип інкапсуляції

Принцип інкапсуляції є одним із фундаментальних принципів об’єктно-орієнтованого програмування (ООП), що визначає основи створення об’єктів та їхньої внутрішньої організації в програмах.

Інкапсуляція полягає у згортанні даних та методів, які з ними пов’язані, в єдиний компонент, надійно захищений від зовнішнього доступу та змін. Цей принцип дозволяє створювати об’єкти, які взаємодіють один з одним через визначені інтерфейси, приховуючи свою внутрішню реалізацію.

Головна мета інкапсуляції — забезпечення контролю доступу до даних та методів об’єкта, зниження залежності від внутрішньої реалізації та ускладнення непрямих взаємодій між об’єктами. Це досягається шляхом використання модифікаторів доступу, таких як public, private та protected, які визначають, хто може отримати доступ до внутрішніх частин об’єкта.

Інкапсуляція включає у себе не тільки приховання даних, але і реалізацію методів, які взаємодіють з цими даними. Це означає, що об’єкт має власну внутрішню логіку, яка відповідає за обробку даних та забезпечення їхньої цілісності.

Інкапсуляція дозволяє розділити рівень інтерфейсу та рівень реалізації, що спрощує використання об’єктів у програмі та полегшує розвиток та модифікацію системи. Вона забезпечує можливість змінювати внутрішню реалізацію об’єкта без впливу на код, який використовує цей об’єкт.

Інкапсуляція допомагає створювати більш стійкі та надійні програми, де доступ до даних та методів об’єкта контролюється, а непрямі залежності між об’єктами зменшуються, що полегшує тестування та підтримку коду.

3. Принцип спадкування

Принцип спадкування, один із ключових принципів об’єктно-орієнтованого програмування (ООП), визначає можливість створення нового класу (підкласу) на основі вже існуючого класу (базового класу) шляхом успадкування його властивостей і методів.

Цей принцип ґрунтується на ідеї використання вже наявного коду для створення нових об’єктів та розширення їх функціональності без необхідності повторної реалізації вже існуючих функцій. Основна концепція полягає в тому, що підклас (дочірній клас) може успадковувати властивості і методи свого базового класу, а також розширювати або перевизначати їх.

Спадкування дозволяє створювати ієрархію класів, де кожен новий клас може використовувати властивості і методи базового класу, спадковуючи їх. Це спрощує розробку, оскільки необхідність виправлення помилок або зміни вже існуючого коду зменшується.

Принцип спадкування допомагає уникнути дублювання коду та покращує структуру програми шляхом створення загальних класів, що можуть бути успадковані різними класами. Це сприяє полегшенню утримання коду та зниженню його повторного використання, що є важливими аспектами у побудові ефективних та масштабованих програм.

Однак, використання спадкування може призвести до залежності між класами та ускладнити їх подальне розширення. В деяких випадках, де використання спадкування є недоцільним, рекомендується використовувати інші методи, такі як композиція об’єктів або інші підходи до структури програмного коду.

4. Принцип поліморфізму

Принцип поліморфізму — це ключовий принцип об’єктно-орієнтованого програмування (ООП), який дозволяє об’єктам реагувати на поведінку, спільну для їхнього типу, різними способами. Слово “поліморфізм” походить від грецьких слів “poly” (багато) та “morphē” (форма), що означає багатоформність. Цей принцип дає можливість об’єктам одного класу мати різну реалізацію методів залежно від контексту, в якому вони використовуються.

Поліморфізм в ООП надає можливість використовувати об’єкти підрізного класу як об’єкти базового класу. Це означає, що об’єкт може представляти різні класи в залежності від контексту використання.

Одним із прикладів поліморфізму є перевантаження методів. Метод з однаковим ім’ям може виконувати різні дії залежно від типу об’єкта, для якого він викликається. Наприклад, в класі “Фігура” можуть бути методи “площа” або “периметр”, які будуть перевизначені в класах-нащадках (таких як “Коло” або “Квадрат”) для обчислення площі чи периметра конкретної фігури.

Ще одним прикладом є реалізація поліморфізму через інтерфейси або абстрактні класи. Об’єкти, які реалізують один і той же інтерфейс чи успадковують абстрактний клас, можуть мати власну реалізацію для методів, визначених у цьому інтерфейсі чи абстрактному класі.

Використання поліморфізму дозволяє розширювати функціональність програми без змінення вже існуючого коду. Це сприяє покращенню читабельності, обслуговування та розширення програм, що є важливими аспектами в розробці великих та складних проектів.

Об’єктно-орієнтоване програмування (ООП) базується на чотирьох ключових принципах: абстракції, інкапсуляції, спадкування та поліморфізму. Ці принципи визначають спосіб побудови програмного коду, що спрямований на створення більш гнучких, модульних та легких для супроводу програм. Абстракція дозволяє управляти складністю системи, виділяючи головні аспекти об’єктів та ігноруючи деталі, що не є важливими на даному етапі. Цей принцип сприяє створенню логічних моделей з використанням інтерфейсів та абстрактних класів, полегшуючи розширення та модифікацію систем. Інкапсуляція забезпечує контроль доступу до даних та методів об’єктів, знижуючи залежність від їхньої внутрішньої реалізації. Цей принцип спрощує використання об’єктів у програмі та полегшує розвиток системи. Спадкування дає можливість створювати нові класи на основі вже існуючих, успадковуючи їхню функціональність. Це зменшує дублювання коду та спрощує розробку, забезпечуючи створення загальних класів, що можуть бути успадковані різними класами. Поліморфізм дозволяє об’єктам реагувати на спільні для них типи поведінки різними способами, що полегшує розширення функціональності без змінення вже існуючого коду. Використання

цих принципів дозволяє створювати програми, що є більш читабельними, ефективними у плані обслуговування та більш гнучкими для майбутнього розвитку. Однак необхідно ретельно враховувати контекст та обставини для правильного застосування цих принципів, оскільки неконтрольоване їх використання може призвести до надмірної складності програми.

2.4 Опис методів та алгоритмів

Зважаючи на важливість технічних аспектів у створенні веб-платформи для керування університетськими проектами, підрозділ присвячено опису ключових стратегій, інструментів та підходів, що використовуються для реалізації та оптимізації функціональності платформи. Цей підрозділ детально вивчає методи розробки програмного забезпечення, алгоритми оптимізації та використання технологій для забезпечення ефективного управління проектами в університетському середовищі.

Під цим підрозділом об’єднано аналіз ключових інструментів веб-розробки, методів взаємодії користувачів та баз даних, а також огляд підходів до оптимізації роботи платформи з метою підвищення її функціональності та ефективності.

Підрозділ виконує важливу роль у демонстрації інструментів та стратегій, що лежать в основі розробки та функціонування платформи, спрямованої на полегшення співпраці та управління університетськими проектами.

2.4.1 Алгоритм сортування QuickSort

Алгоритм QuickSort є ефективним інструментом для сортування проектів за різними характеристиками, такими як час розробки, складність чи статус. QuickSort є одним з швидких та ефективних алгоритмів сортування, особливо коли маємо справу з великою кількістю елементів.

Основна ідея QuickSort полягає у поділі масиву на менші частини, засновані

на порівняннях з обраним елементом, який називається опорним. Він рекурсивно сортує частини менші та більші за опорний елемент, доки весь масив не буде відсортований.

Для застосування QuickSort до сортування проектів за різними характеристиками, потрібно визначити, які поля або характеристики проектів будуть ключами сортування. Наприклад, якщо ви хочете відсортувати проекти за часом розробки, то поле з інформацією про час розробки може бути ключем для QuickSort.

Після визначення ключа сортування, можна застосувати алгоритм QuickSort до списку проектів, порівнюючи значення вибраного ключа для кожного проекту. За допомогою порівнянь та рекурсивних викликів QuickSort проекти можуть бути швидко та ефективно відсортовані за вибраною характеристикою.

Цей підхід допоможе організувати проекти відповідно до потрібних критеріїв та забезпечити користувачам зручний доступ до інформації про проекти на вашій веб-платформі.

Таким чином, алгоритм QuickSort є потужним інструментом для швидкого та ефективного сортування проектів за різними характеристиками. Використання цього алгоритму дозволяє організувати дані з великою кількістю елементів за певними параметрами, такими як час розробки чи статус. Основна концепція QuickSort полягає у рекурсивному поділі масиву на менші частини та їх послідовному порівнянні з опорним елементом для досягнення повного сортування. При використанні цього алгоритму для сортування проектів необхідно обрати ключі сортування, які відповідають певним критеріям, наприклад, часу розробки. Застосування QuickSort допомагає створити систему, яка розташовує проекти відповідно до вибраної характеристики, забезпечуючи зручний доступ до інформації для користувачів. Цей алгоритм забезпечує швидке та ефективне сортування, що робить його важливим інструментом для оптимізації відображення даних на веб-платформі, покращуючи користувацький досвід та забезпечуючи зручність у використанні системи.

2.4.2 Алгоритм шифрування bcrypt

Використання алгоритму шифрування bcrypt для зберігання облікових даних користувачів є важливим кроком у забезпеченні безпеки даних.

Алгоритм bcrypt є одним з найбільш надійних і рекомендованих для зберігання паролів у захищеній формі. Основна перевага bcrypt полягає у його здатності до уповільнення обчислень, що ускладнює брутфорс атаки та знижує ризик перебору паролів. Також, bcrypt автоматично включає в себе сіль (salt), що додає до пароля додатковий рівень безпеки.

При використанні bcrypt для зберігання паролів користувачів, сам пароль спочатку хешується (перетворюється в незворотний вигляд), і отриманий хеш зберігається в базі даних. Кожен хеш унікальний і не може бути відновлений до початкового паролю, що робить його безпечним для зберігання.

Основні переваги використання bcrypt:

- уповільнення обчислень — це ускладнює зловмисникам спроби перебору паролів;
- використання солі (salt) — кожен пароль має свою унікальну сіль, що ускладнює масові атаки на базу даних;
- рекомендований стандарт — bcrypt вважається одним з найбільш надійних алгоритмів для зберігання паролів.

Алгоритм bcrypt став ключовим інструментом для збереження паролів користувачів, пропонуючи високий рівень безпеки та надійності. Його унікальність полягає у спеціальній схемі хешування, що уповільнює атаки на паролі та включає у себе сіль, що робить хешування навіть більш складним. Використання bcrypt у системі зберігання паролів є ключовим кроком для підвищення безпеки даних користувачів, зменшення ризику атак та забезпечення конфіденційності.

Висновки до розділу 2

Аналіз аналогічних систем виявився ключовим етапом у процесі розробки. Це дозволило не лише зрозуміти, які підходи вже існують на ринку, а й вивчити їхні сильні та слабкі сторони. Виокремлення різних систем дало можливість зрозуміти, які особливості варто взяти на увагу, а що можна уникнути чи покращити власними зусиллями.

Опис програмних засобів приніс важливість вибору правильних інструментів для власного проекту. Оцінка їхньої ефективності, сумісності та можливостей покращила шлях до реалізації проекту. Крім цього, вірний вибір програмних засобів впливає на продуктивність та успішність розробки.

Принципи розробки стали фундаментом системи. ООП та SOLID принципи допомогли побудувати чітку архітектуру та забезпечити модульність коду. Ці принципи також вплинули на розподіл відповідальностей та взаємодію компонентів системи, що сприяє легкості підтримки та розвитку програми у майбутньому.

Опис методів та алгоритмів надав необхідний інструментарій для вибору найефективніших підходів до обробки даних та виконання операцій. Розуміння алгоритмів сортування, шифрування та інших методів вирішення завдань стало ключовим елементом в розробці оптимальних стратегій.

Усі ці підрозділи взаємодіяли та впливали один на одного, створюючи комплексний, добре структурований та працездатний продукт. Підрозділи спільно допомогли збудувати систему, яка відповідає вимогам якісного програмного продукту, має високий ступінь безпеки, стійкість до атак та готова до майбутніх змін і розширень.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

В розділі “Опис програмної реалізації” надається детальний огляд технічних аспектів та ключових рішень, що використовуються в процесі створення програмного продукту.

Цей розділ є важливим компонентом, оскільки розділ надає глибоке розуміння технічних рішень, їхніх характеристик і взаємодії, що дозволяє краще осмислити функціональні можливості продукту.

У цьому розділі представлен опис програмного продукту, включаючи в себе використані мови програмування, архітектурні рішення, вибрані фреймворки та бібліотеки, методи розробки, а також алгоритми, що застосовуються. Розглянуті технічні деталі, які дозволили створити функціональність програмного продукту та забезпечили його працездатність.

В цьому розділі також розглянуті можливі складнощі, з якими зіткнулися під час розробки програмного продукту, і способи, якими ці складнощі були вирішені. Описуючи технічні аспекти розробки, цей розділ покликаний поглибити розуміння основних концепцій, що лежать в основі створення програмного продукту.

3.1 Серверна частина розробленої системи

Сервер є критичною складовою програмної системи, надаючи надійну основу для функціонування програм та забезпечуючи взаємодію з клієнтськими пристроями. Цей підрозділ присвячений детальному огляду та аналізу інфраструктури серверної частини програмного продукту.

Розглянемо аспекти апаратного та програмного забезпечення, що становлять основу сервера, а також розкриємо ключові технології та механізми,

використані для забезпечення продуктивності, масштабованості та безпеки системи. Описуючи технічні деталі, які входять до складу серверної архітектури, цей розділ розкриє основні принципи функціонування та ефективності програмної системи. Детально розглянуто конфігурацію сервера, використані технології, можливості масштабування, аспекти безпеки та захисту даних, що відіграють ключову роль у забезпеченні стабільної та безпечної роботи програмного продукту.

Цей підрозділ підкреслить важливість серверної частини, яка є фундаментом для забезпечення надійності, продуктивності та безпеки програмної системи.

Структура серверної частини системи представлена на рисунку 3.1.

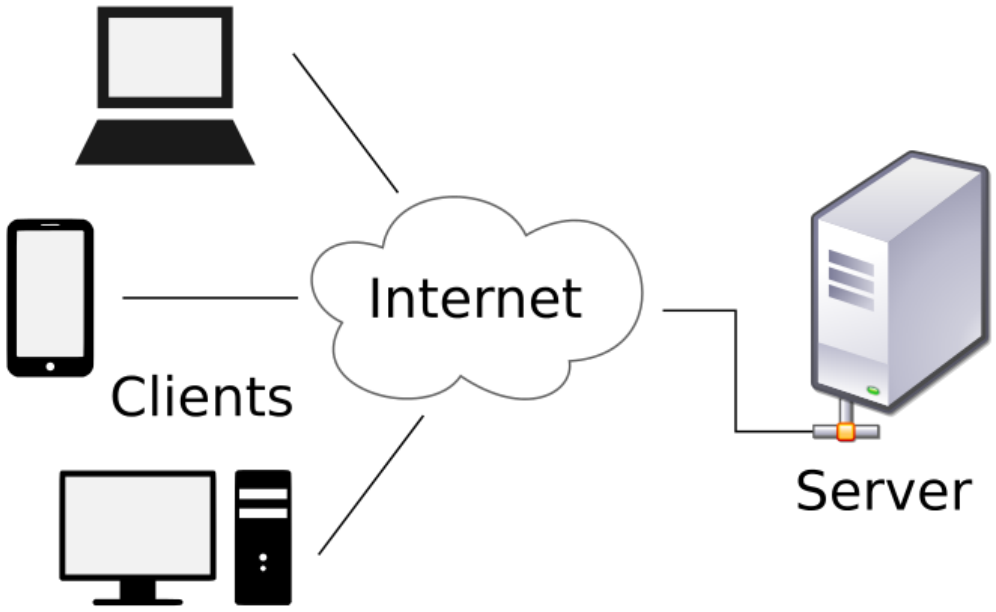


Рисунок 3.1 — Структура серверної частини розробленої системи

Серверна частина системи є центральним елементом у клієнт-серверній архітектурі. Цей компонент забезпечує обробку запитів від клієнтів, зберігання та керування даними, виконання бізнес-логіки та надання відповідей клієнтам.

Сервер отримує запити від клієнтів і виконує відповідні операції відповідно до бізнес-логіки програми. Сервер зберігає та керує базою даних, виконує операції зчитування, запису, видалення та модифікації інформації. Серверна частина відповідає за забезпечення безпеки даних та обробки запитів, включаючи автентифікацію та авторизацію користувачів. Також сервер містить логіку програми, відповідає за внутрішні операції системи та виконання різноманітних операцій над даними згідно з правилами програми.

Серверна частина є основою для функціонування клієнт-серверних систем і відіграє ключову роль у забезпеченні коректної та ефективної взаємодії між клієнтами та даними.

Діаграма класів має наступний вигляд та представлена на рисунку 3.2:

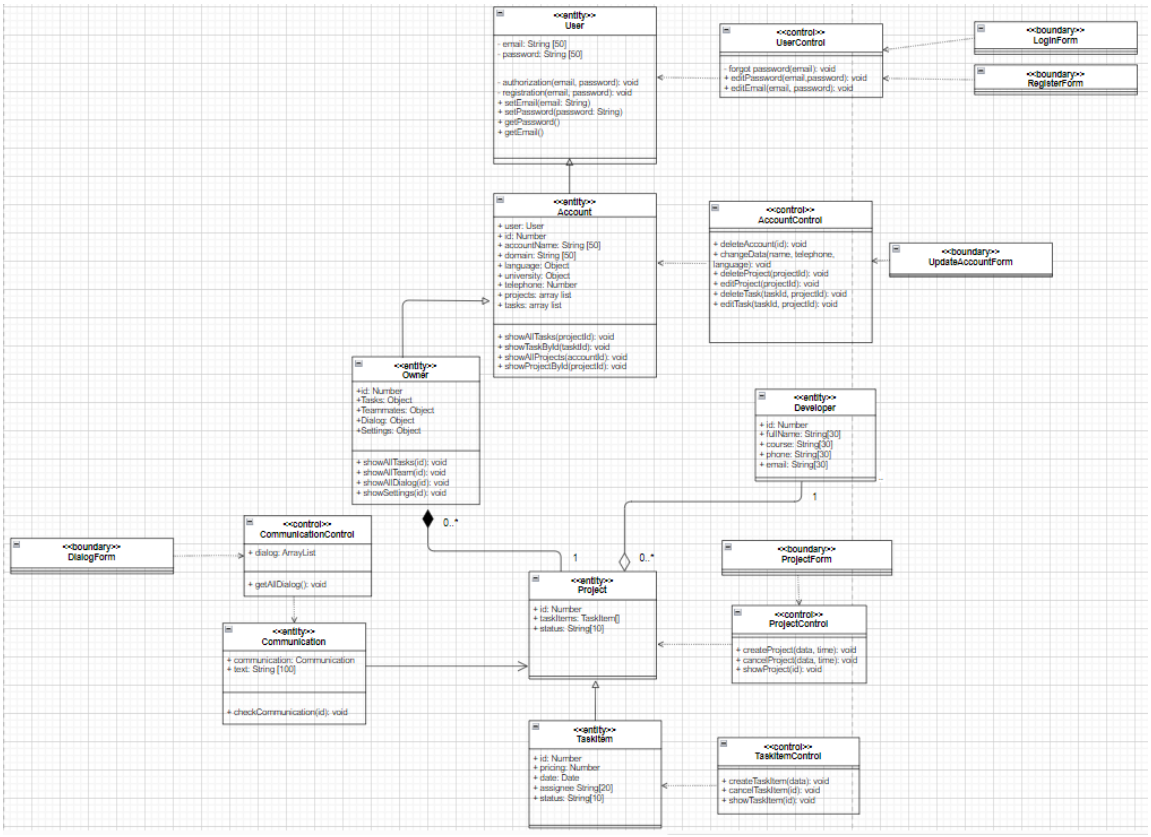


Рисунок 3.2 — Діаграма класів

Центральним елементом програмного додатку виступає система авторизації та реєстрації, що розглядається як основна складова класу USER. Цей клас відіграє ключову роль у функціоналі додатку, забезпечуючи аутентифікацію користувача та генерацію та перевірку JWT токенів, які служать ідентифікаторами користувачів.

Процес авторизації та реєстрації включає введення користувачем електронної адреси та пароля, передачу цих даних на сервер, їх перевірку в базі даних та у випадку позитивного результату — формування та передачу зашифрованого токену на клієнтську частину для подальшого збереження та використання в заголовках запитів. У випадку негативної перевірки, сервер генерує відповідний статус помилки, який також відправляється на клієнтську частину.

Додаток реалізує дві ролі з обмеженнями на доступ до відповідних запитів: роль DEVELOPER з обмеженим доступом до загальних URI та роль OWNER з розширеними можливостями управління контентом. Крім того, функціонал для інших сутностей, таких як PROJECT, TASKITEM, включає можливості з доступу, збереження, редагування та видалення інформації. При збереженні проекту система отримує id проекту для подальшого призначення власника завдання.

Для моделювання об’єктних даних та взаємодії з базою даних MongoDB та Node.js використовується інструмент Mongoose. Mongoose використовується для зручного взаємодії з базою даних MongoDB.

Цей інструмент надає можливість розробникам Node.js створювати моделі даних, використовуючи JavaScript-подібну синтаксичну структуру, що полегшує роботу з даними. Крім того, Mongoose дозволяє визначати схеми для об'єктів, що зберігаються в базі даних, та забезпечує валідацію цих схем.

Також цей інструмент дозволяє контролювати структуру даних, перевіряти їх коректність перед збереженням у базі та розробляти потужні моделі, які відображають логіку програми і спрощують взаємодію з базою даних. Використання Mongoose дозволяє ефективно керувати даними, встановлювати

зв'язки між колекціями документів та оптимізувати роботу з базою даних MongoDB в застосунку, забезпечуючи стабільність та продуктивність системи.

Структура бази данних має наступний вигляд та представлена для огляду на рисунку 3.3.

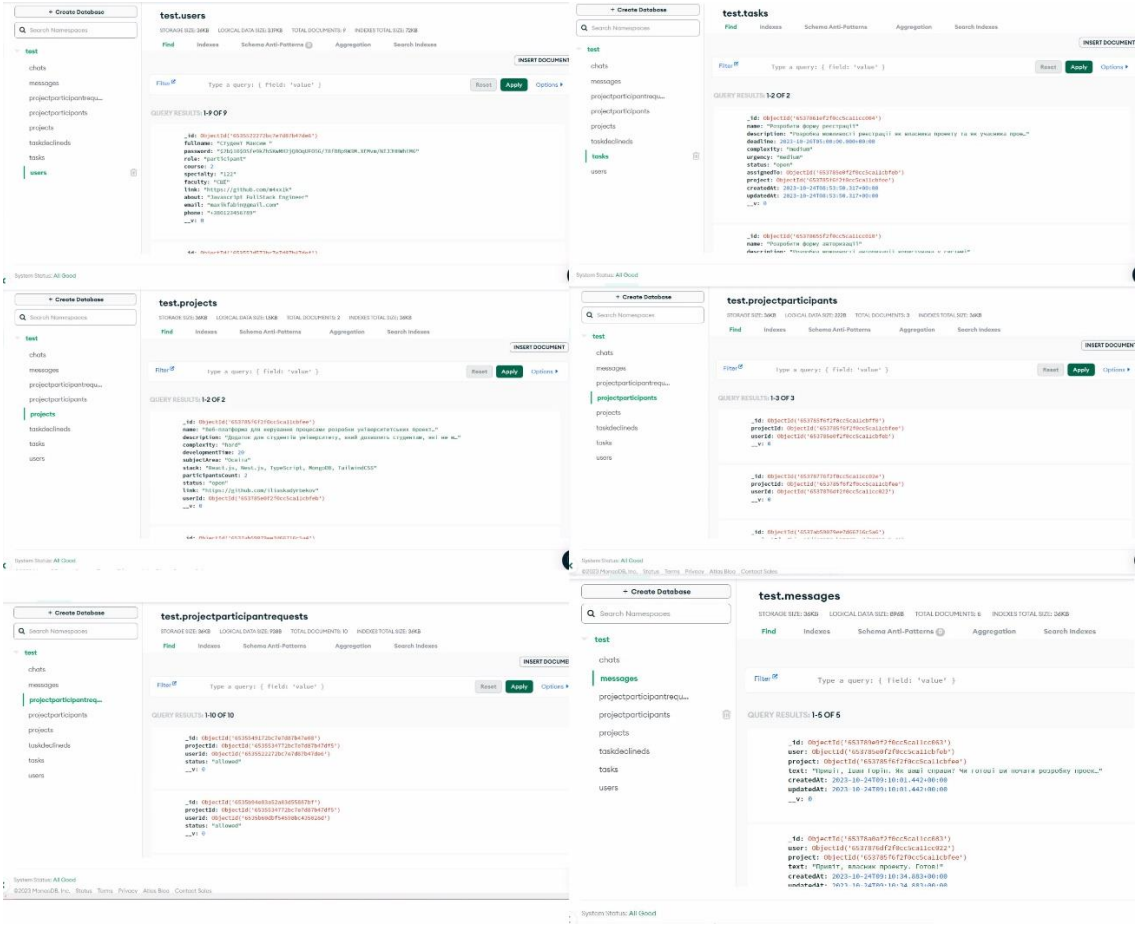


Рисунок 3.3 — Структура бази даних

Як видно з рисунку, є декілька основних сутностей — користувачі, проекти, завдання та коментарі.

Сутність користувачів містить інформацію про користувачів платформи, таку як ім'я, електронну пошту, пароль, роль (наприклад, студент, викладач, адміністратор), а також іншу відповідну особисту інформацію.

Проекти включають інформація про самі проекти, включаючи назву, опис,

дати початку та завершення, відомості про власника проекту, статус проекту тощо.

Сутність завдань містить організація завдань для кожного проекту, які розбиті на підзавдання для більшої деталізації. Кожне завдання має інформацію про статус, відповідального користувача, дату та інші характеристики.

Сутність для збереження коментарів, що відносяться до проектів, завдань чи підзавдань. Це є важливим для комунікації та обговорення певних аспектів проектів між користувачами.

Таким чином, у підрозділі, присвяченому серверній частині програми для управління університетськими проектами, розкрито безліч ключових аспектів, які визначають функціональність та надійність програмного забезпечення.

Серверна частина системи виявилася не лише центральним елементом, але й фундаментом для забезпечення безпеки, ефективності та стабільності роботи. Ця частина відповідає за обробку запитів користувачів, зберігання даних, управління бізнес-логікою та надання відповідей клієнтам.

Особлива увага приділена аутентифікації через клас User, який виконує ключову роль у взаємодії з користувачами, забезпечуючи створення та перевірку JWT токенів для ідентифікації. Ролі та обмеження доступу визначають рівні привілеїв для користувачів у системі, забезпечуючи контроль за взаємодією з ресурсами.

Інструмент Mongoose є корисним для зручної взаємодії з базою даних MongoDB, спрощуючи роботу з даними та їх валідацію. Структура бази даних надала чітку систему для збереження інформації про користувачів, проекти, завдання та коментарі, що сприяє організованому зберіганню та керуванню даними.

3.2 Клієнтська частина розробленої системи

У підрозділі, присвяченому клієнтській частині розробленої системи для управління університетськими проектами, зосереджено на ключових аспектах, що стосуються інтерфейсу користувача та його функціоналу. Цей компонент системи є прямим зв’язком між користувачем та програмним забезпеченням, тож ця частина відіграє важливу роль у забезпеченні зручності та ефективності взаємодії.

В цьому підрозділі розглянуто різноманіття аспектів, починаючи від опису інтерфейсу користувача, до аналізу функціональності та важливих особливостей клієнтської частини системи. Особлива увага приділена навігації, взаємодії з даними, можливостям користувача та іншим ключовим елементам, які формують зручне та продуктивне використання системи.

Клієнтська частина програми складається з використання фреймворку React.js, що дозволяє побудувати динамічний та ефективний інтерфейс користувача.

Окрім React.js, для оптимізації продуктивності програми використовується Next.js. Цей інструмент дозволяє виконувати початкові запити на дані на серверній частині програми при її першому завантаженні. Це дозволяє попередньо завантажувати та показувати необхідну інформацію, що сприяє швидкодії та зручності взаємодії з програмою.

Ця комбінація технологій сприяє інноваціям у функціональності і структурі програмного забезпечення, роблячи його більш адаптивним та відзивчивим. Особлива увага приділяється не лише технічним аспектам, а й дослідженню вимог та поведінки користувачів для досягнення максимальної зручності та задоволення від використання системи.

Завдяки використанню цих технологій та методів, клієнтська частина програми не лише надає інтуїтивно зрозумілий інтерфейс для користувачів, а й забезпечує швидку та ефективну роботу програми в цілому.

Розглянемо Use-Case діаграму для клієнтської частини додатку, яка має

наступний вигляд, що представлена на рисунку 3.4.

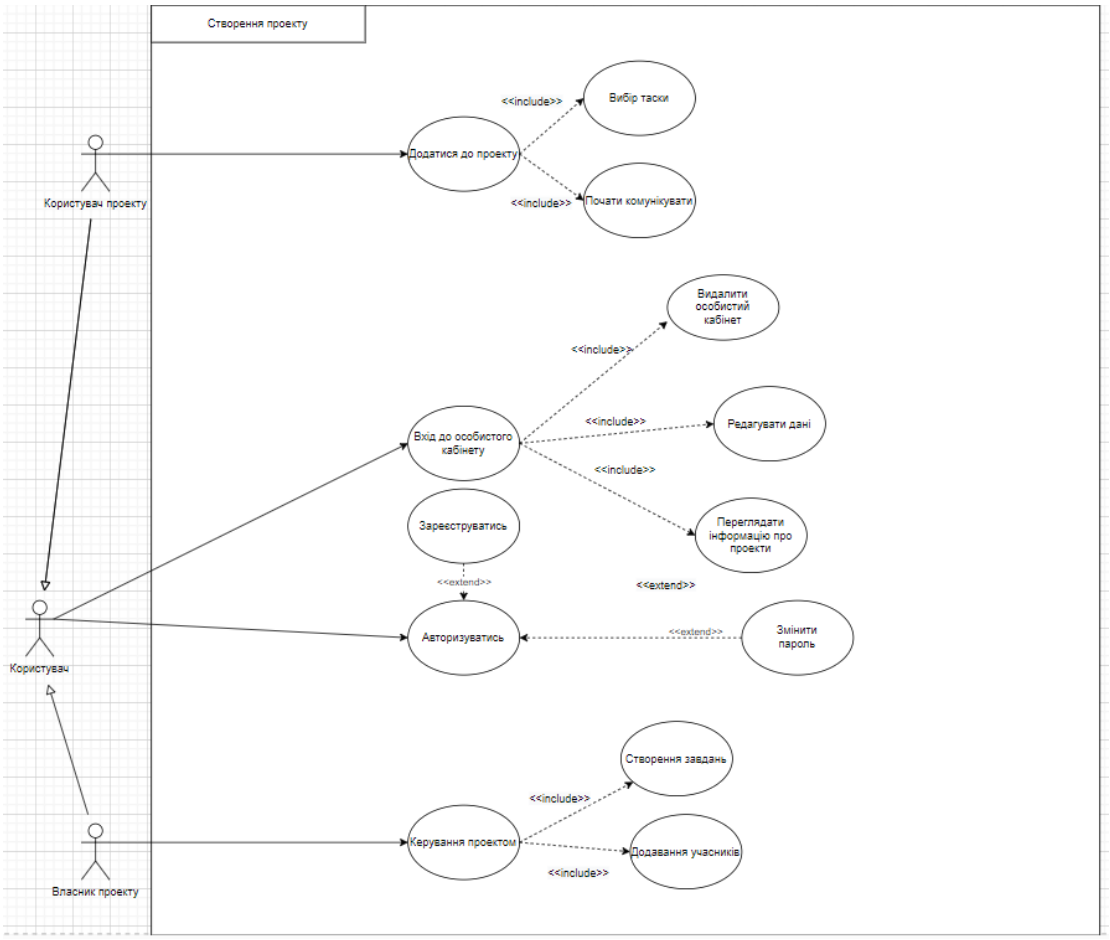


Рисунок 3.4 — Use-Case діаграма

На схемі видно, що існують дві ключові ролі у цьому застосунку: звичайний користувач проекту і власник проекту. Кожна з цих ролей успадковує властивості користувача. Таким чином, користувач має обмежений функціонал, який дозволяє створювати обліковий запис та переглядати проект, тоді як користувач проекту має доступ до всіх можливостей користувача та володіє функціями власника проекту. Власник проекту, у свою чергу, має розширені можливості, включаючи авторизацію, створення проекту, налаштування його параметрів та управління складовими проекту.

Програмний код на клієнті розбитий за компонентами, ті частини коду, які

зустрічаються у декількох місцях винесені у окремі компоненти для того, щоб перевикористати їх. На рисунку 3.5 представлена файлова стурктура розроблюваної ситеми.

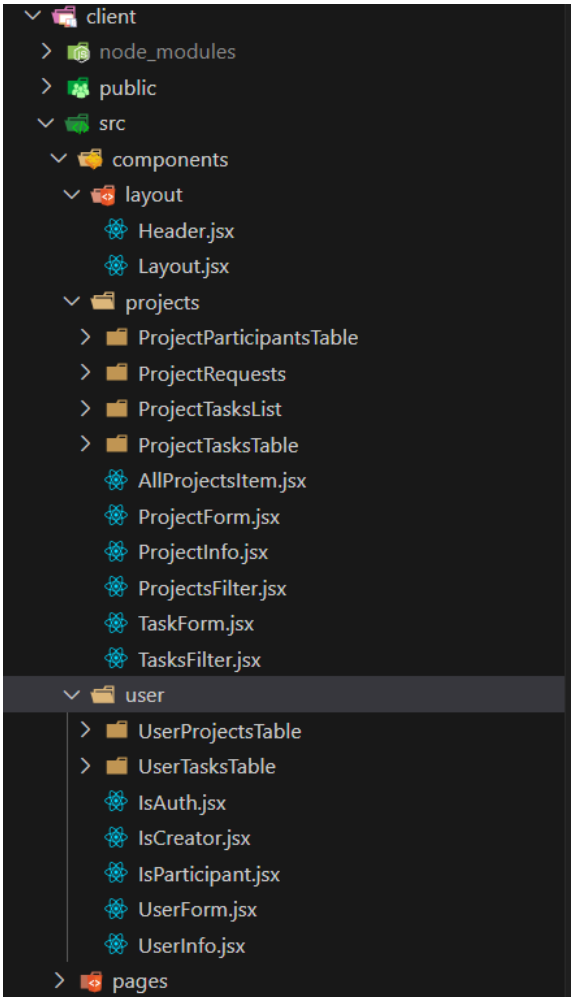


Рисунок 3.5 — Файлова структура системи

У підрозділі, присвяченому клієнтській частині системи управління університетськими проектами, детально розглянуто ключові аспекти, від інтерфейсу користувача до функціональних особливостей програми. Використання фреймворку React.js та Next.js дозволило створити динамічний та ефективний інтерфейс, а також покращити продуктивність системи завдяки початковим запитам на дані на серверній частині. Ця комбінація технологій

створює не лише зручний та інтуїтивно зрозумілий інтерфейс для користувачів, але й забезпечує ефективну та швидку роботу програми в цілому.

Особлива увага приділена не лише технічним аспектам, але й вивченню потреб користувачів для досягнення найвищої зручності та задоволення від використання системи. Це дозволило відобразити різноманітні можливості і структуру програми таким чином, щоб користувачі отримували максимальне задоволення від взаємодії з нею.

Файлова структура системи відображає чітку організацію програмного коду за допомогою компонентів, що сприяє ефективному перевикористанню коду та забезпечує легке розширення та модифікацію програмного забезпечення. В цілому, цей підрозділ проявив якісний підхід до створення клієнтської частини системи, орієнтований на зручність та високу продуктивність для користувачів.

Висновки до розділу 3

Після докладного аналізу серверної та клієнтської частини розробленої системи для управління університетськими проектами можна зробити кілька висновків.

Серверна частина системи виявилася надзвичайно важливою для забезпечення надійності, продуктивності та безпеки програмного продукту. Її структура ретельно продумана для забезпечення безпеки даних, оптимізації роботи з базою даних та впровадження бізнес-логіки програми. Використання інструменту Mongoose сприяло ефективному управлінню даними, встановленню зв'язків між колекціями документів та забезпечило стабільність та продуктивність системи.

З іншого боку, клієнтська частина системи створила зручний та інтуїтивно зрозумілий інтерфейс для користувачів. Використання фреймворку React.js та Next.js дозволило створити динамічний інтерфейс та покращити продуктивність

завдяки ефективним початковим запитам на дані. Особлива увага була приділена вивченню потреб користувачів, що дозволило відобразити різноманітні можливості програми таким чином, щоб користувачі отримували максимальне задоволення від взаємодії з нею.

У цілому, сполучення надійності та продуктивності серверної частини з зручністю та ефективністю клієнтської створює цілісну систему, яка відповідає вимогам користувачів та забезпечує стабільну та продуктивну роботу.

Такий підхід до розробки системи дозволяє не лише надати користувачам зручний інтерфейс, але й забезпечити безпеку, швидкість та ефективність управління даними всередині системи. Ця взаємодія серверної та клієнтської частин системи створює надійну основу для управління університетськими проектами, поєднуючи безпеку, ефективність та зручність в одному функціональному рішенні. Такий комплексний підхід гармонійно поєднує технічну надійність із зручністю для кінцевого користувача, створюючи інструмент, який відповідає найвищим стандартам якості та функціональності.

4 РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Взаємодія користувача з програмною системою є ключовим аспектом в її функціонуванні. Робота користувача з програмними системами включає в себе різноманітні дії, від початкового взаємодії з інтерфейсом до виконання конкретних завдань та отримання очікуваних результатів.

Користувачі взаємодіють з програмними системами через різноманітні інтерфейси, які можуть бути веб-сайтами, мобільними додатками, настільними застосунками тощо. Цей процес включає реєстрацію, авторизацію, введення даних, вибір опцій, виконання операцій та отримання результатів.

Успішність взаємодії користувача з програмними системами визначається величезною кількістю факторів, від зручності та інтуїтивності інтерфейсу до швидкості виконання операцій та якості отриманих результатів. Важливою є також підтримка та навчання користувачів, забезпечення безпеки та конфіденційності даних, а також реагування на їхні потреби та змінювані вимоги.

У цьому процесі ключовим є розуміння потреб користувачів та розробка програмної системи з урахуванням їхніх очікувань та можливостей. Наявність зручного та ефективного інтерфейсу, логічна організація функціоналу та можливість швидкої адаптації до змін у потребах користувачів визначають успішність взаємодії користувача з програмною системою.

4.1 Інсталяція програмного забезпечення

Для того, щоб виконати інсталяцію програмного забезпечення, потрібно виконати наступні кроки на локальному середовищі:

1. Перейти в папку backend: з використанням командного рядка (терміналу) перейдіть у папку “backend”. Це можна зробити за допомогою команди cd backend.

2. Створити файл .env з вмістом: потрібно створити файл з назвою .env у папці "backend" і скопіювати або вставити вміст, який міститься за посиланням https://pastebin.com/rwS5PLa2 у цей файл.

3. Виконати команду npm i: виконати команду npm i в терміналі, знаходячись у папці “backend”. Ця команда встановить всі залежності проекту згідно з файлом package.json.

4. Виконати команду npm run dev: після встановлення залежностей треба запустити команду npm run dev у папці “backend”. Ця команда запустить сервер або додаток у режимі розробки.

5. Перейти в папку client: тепер необхідно повернутися назад на рівень терміналу і перейти у папку “client” командою cd client.

6. Виконати команду npm i: так само, як у кроці 3, виконати команду npm i в папці “client”, щоб встановити залежності для клієнтської частини проекту.

7. Виконати команду npm run dev: Запустіть команду npm run dev у папці “client”. Це запустить клієнтську частину додатку в режимі розробки.

8. Після виконання цих кроків проект повинен бути доступний за адресою http://localhost:5173 у браузері.

4.2 Вимоги до обчислюваної техніки

Встановлення операційної системи є ключовою складовою для функціонування програмного забезпечення. Програмна ситема підтримує будь-яку операційну систему (Windows, macOS, Linux) та необхідні параметри її конфігурації.

Також необхідним є встановлення мінімальної кількості ОЗП для запуску

програми.

Обчислювальна техніка повинна мати можливість підключення до мережі для спільної роботи, оновлень та завантаження даних з Інтернету.

Не вимагається спеціальна графічна карта, але підтримка монітора з роздільною здатністю 1280x800 або вище буде краще.

Обчислювальна техніка повина бути сумісна для роботи з браузерами.

4.3 Демонстрація функціоналу програмної системи

У цьому підрозділі детально розглянемо, як користувачі взаємодіють з програмною системою, тобто веб-платформою для керування процесами розробки університетських проектів. Користувачі грають ключову роль у життєвому циклі проекту, і їхнє задоволення та продуктивність використання системи є важливими факторами для успіху проекту.

Розглянемо різні аспекти взаємодії користувачів з системою, включаючи створення облікових записів, авторизацію, навігацію по інтерфейсу, створення та управління проектами, взаємодію з іншими користувачами, внесення змін до проектного коду та багато інших аспектів.

Також розглянемо інструменти та функціональність, які створені для полегшення користувачам роботи з системою, забезпечення їхньої зручності та продуктивності.

Першим чином, учасник проекту повинен зареєструватися на платформі, заповниши усі необхідні поля, як показано на рисунку 4.1. Користувач вводить своє ім’я, прізвище, електронну пошту, курс, факультет, спеціальність та інші обов’язкові поля. Користувач може додати посилання на свій продукт (сайт чи репозиторій на GitHub), розповісти про себе тощо.

Після введення основної інформації, користувачеві надсилається лист з посиланням для підтвердження реєстрації.

Користувач повинен перейти за посиланням для завершення реєстрації.

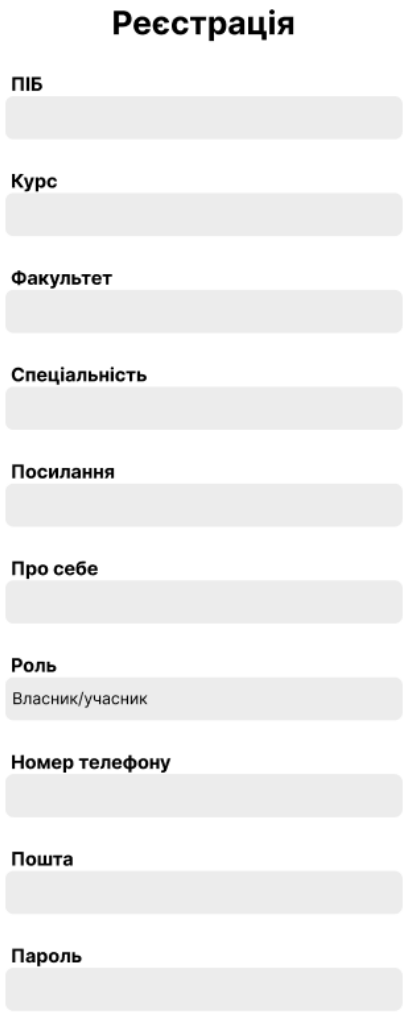


Рисунок 4.1 — Реєстрація на платформі

Після реєстрації можна авторизуватися на платформі, як показано на рисунку 4.2. Процес авторизації на платформі полягає у підтвердженні ідентифікації користувача та визначенні його прав доступу до функціоналу. Користувач вводить свою електронну адресу та пароль, щоб увійти на платформу. Платформа перевіряє введені дані на відповідність збереженим у системі обліковим даним. Якщо введені дані відповідають обліковим записам у системі, користувача авторизується. Після успішної авторизації генерується токен або

створюється сесія, які ідентифікують користувача протягом його сеансу роботи на платформі. Цей токен/сесія зберігається у системі, щоб використовувати його для підтвердження авторизації під час кожного запиту користувача до сервера. Залежно від ролі користувача (власник проекту чи учасник проекту), система надає відповідний рівень доступу до функціоналу платформи.

Рисунок 4.2 — Авторизація на платформі

Після авторизації висвітлена сторінка усіх проектів університету, що представлена на рисунку 4.3.

Сторінка усіх проектів університету є центральним місцем, де представлені всі доступні проекти для участі студентів та викладачів. Основна мета цієї сторінки полягає у зручному перегляді, пошуку та фільтрації проектів залежно від різних критеріїв.

Кожен проект представлений у вигляді картки з назвою, коротким описом, інформацією про статус, тип складності, тривалість розробки та іншими характеристиками.

На цій сторінці також представлена можливість сортування проектів за різними категоріями, такими як тип складності, стан проекту, стек технологій, час розробки та інші параметри.

Також є зручне поле для пошуку проектів за ключовими словами, назвою чи іншими критеріями.

Ця сторінка є важливим інструментом для навігації та вибору проектів для участі, тому вона має бути інтуїтивно зрозумілою, зручною та інформативною для користувачів.

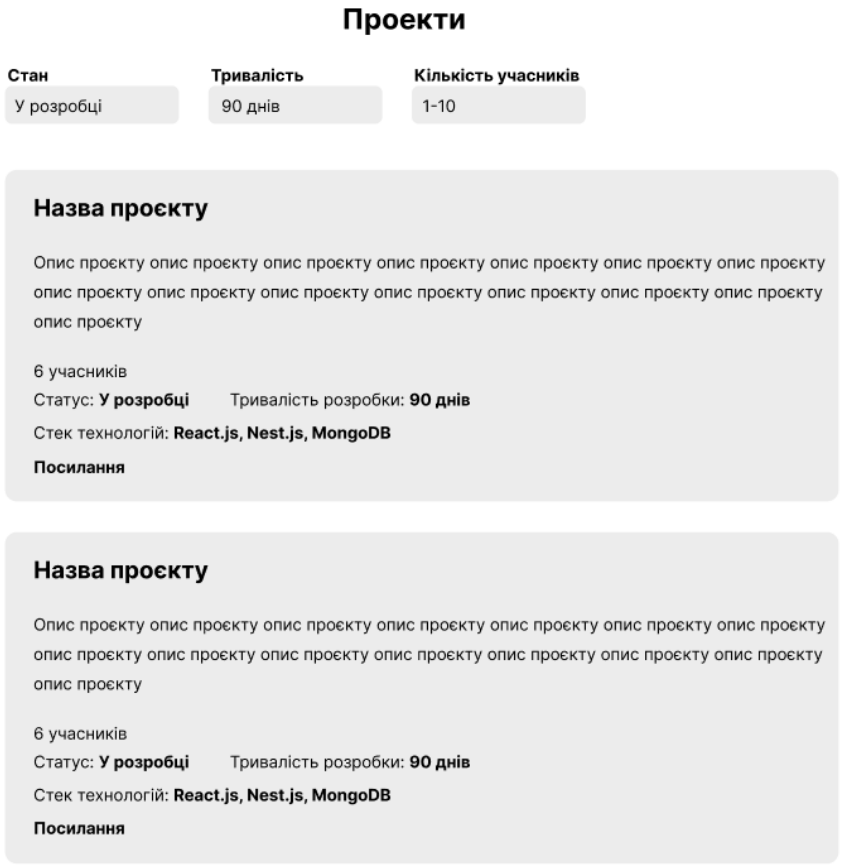


Рисунок 4.3 — Сторінка усіх університетських проектів

Після цього можна перейти до сторінки проекту, що показана на рисунку 4.4.

Сторінка проекту на платформі університету є ключовою для представлення всієї інформації, пов’язаної з конкретним проектом, та для спрощення участі студентів. Сторінка містить інформацію про проект: назва проекту, короткий опис та ключові характеристики проекту (тип складності, статус, тривалість розробки, стек технологій, предметна область тощо), фотографії. Також містить інформацію про команду проекту: список учасників, їхні ролі та відомості про кожного з них.

Учасники можуть подати заявку на участь в проекті. Трохи нижче за сторінкою є перелік завдань або задач, що пов’язані з проектом, з описом, статусом та відповідальними за виконання. На цій сторінці можна почати комунікацію та спілкування, є можливість спілкування для учасників команди проекту.

Ця сторінка створена для того, щоб учасники могли знайти всю необхідну інформацію про проект, його поточний стан та завдання, а також зручно спілкуватись та працювати над ним.

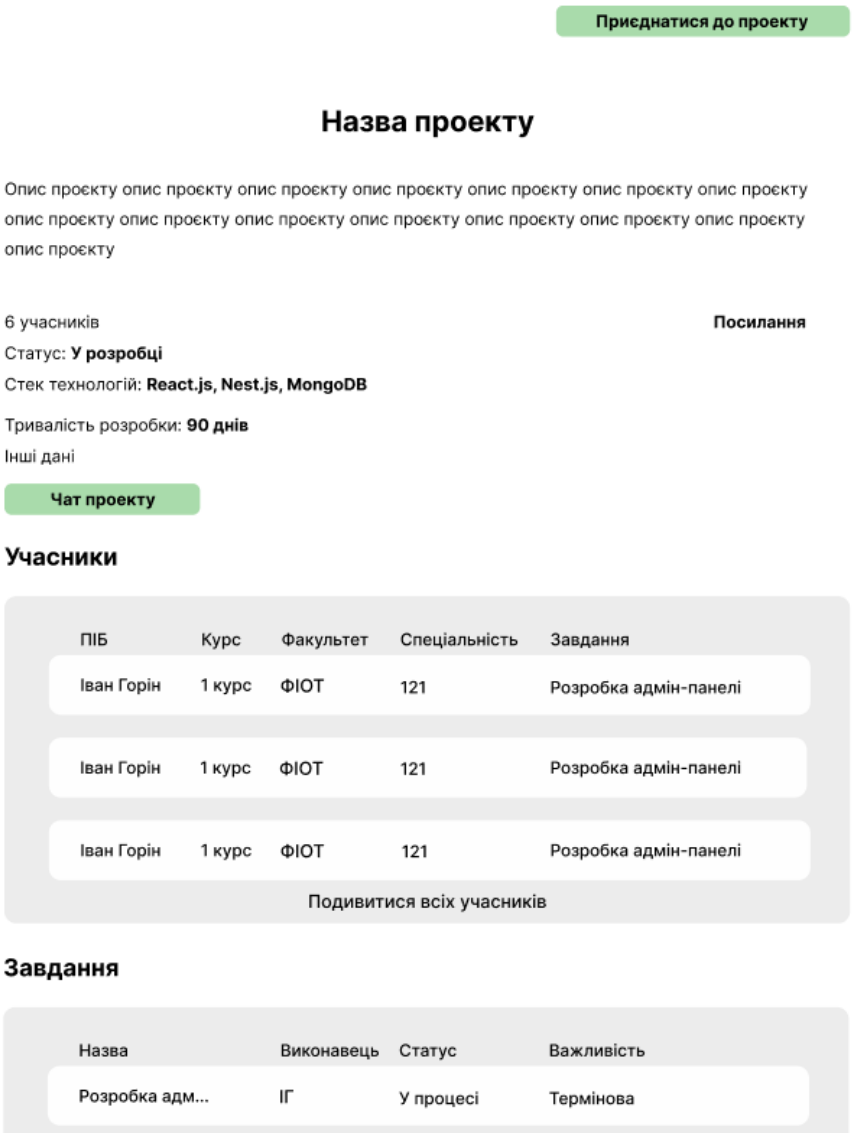


Рисунок 4.4 — Сторінка проекту

Через сторінку проекту можна перейти на сторінку учасників, що показана на рисунку 4.5.

Сторінка учасників проекту в системі управління проектами університету — це важлива складова, яка забезпечує перегляд і управління інформацією про всіх учасників, що беруть участь у конкретному проекті.

Ця сторінка створена для того, щоб учасники могли знайти всю необхідну інформацію про проект, його поточний стан та завдання, а також зручно спілкуватись та працювати над ним. Також на цій сторінці представлені статуси завдань учасників та нформація про завдання, які призначені кожному учаснику. Статус завдань (в роботі, відкладено, завершено тощо) відображений для кожного учасника.

Рисунок 4.5 — Сторінка учасників проекту

Розглянемо далі сторінку завдань проекту, яка представлена на рисунку 4.6. Сторінка завдань проекту в системі управління університетськими проектами — це місце, де можна переглянути та керувати усіма завданнями, пов’язаними з

конкретним проектом.

На сторінці зображен список усіх завдань, які пов’язані з проектом. Кожне завдання може мати назву, опис, призначеного виконавця, статус (виконується, відкладено, завершено тощо) та дедлайн.

Також на цій сторінці є можливість фільтрувати завдання за різними критеріями, такими як стан, призначений виконавець, строк виконання та інші параметри.

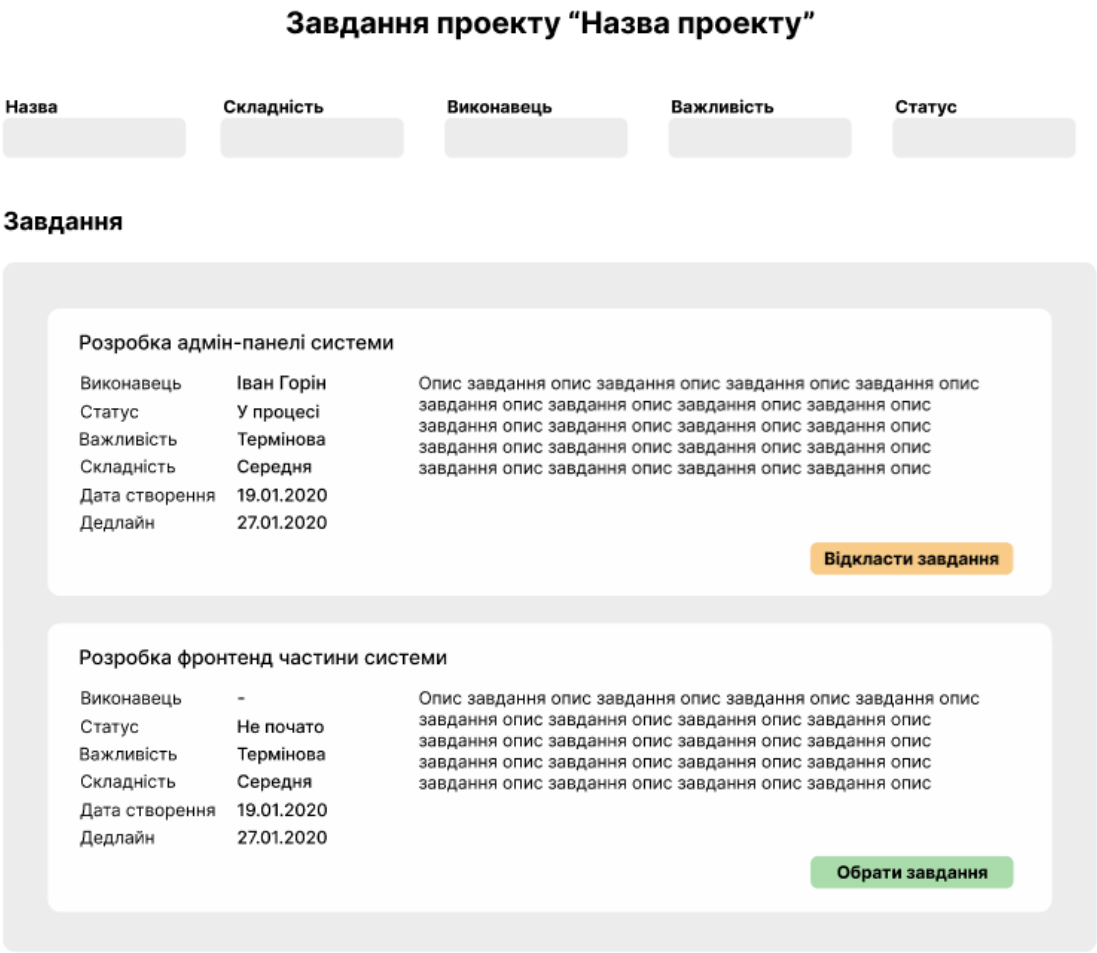


Рисунок 4.6 — Сторінка завдань проекту

Наступною сторінкою розглянемо сторінку профілю користувача, яка представлена на рисунку 4.7. Сторінка профілю користувача є особистим кабінетом, який містить інформацію про кожного окремого користувача

платформи.

На цій сторінці відображено основні дані користувача, такі як ім’я, прізвище, контактна інформація (email, телефон тощо). Представлена можливість внесення або редагування особистих даних, таких як місце проживання, дата народження, освіта, досвід тощо, також можна видалити профіль.

Трохи нижче представлена активність на платформі та інформація про активність користувача, така як дата реєстрації, останні входи, кількість участі в проектах, завершені завдання тощо.

Ще нижче представлена інформація про перелік усіх проектів, участь в яких приймає користувач, а також список призначених завдань та їх статуси.

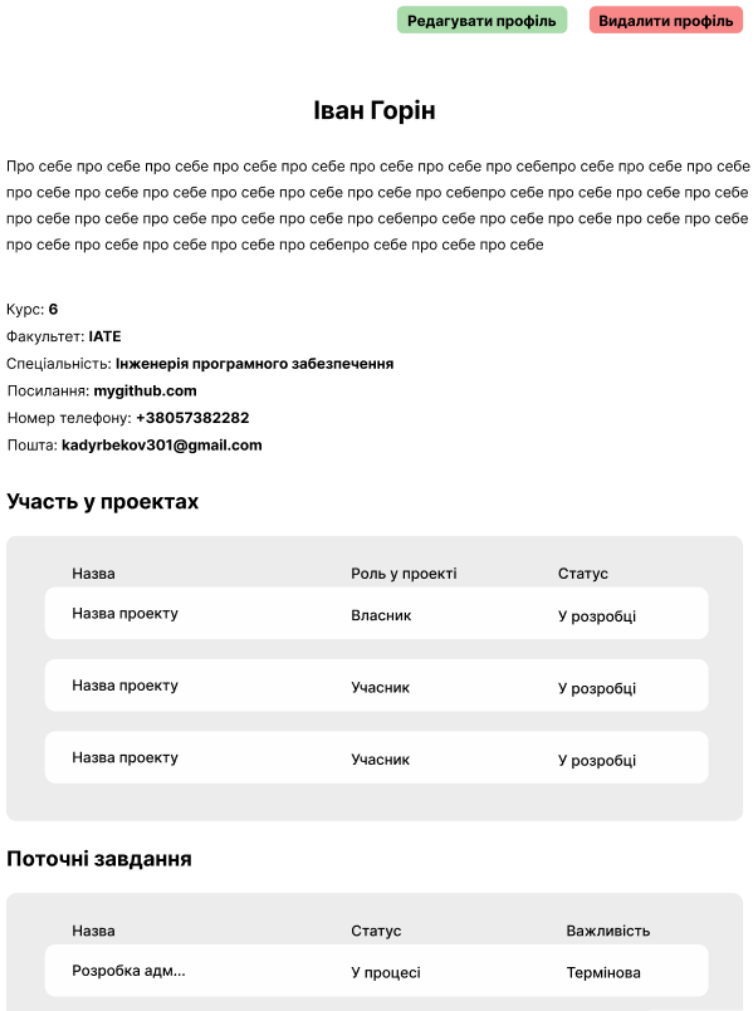


Рисунок 4.7 — Сторінка профілю

Редагування сторінки профілю можна побачити на рисунку 4.8.

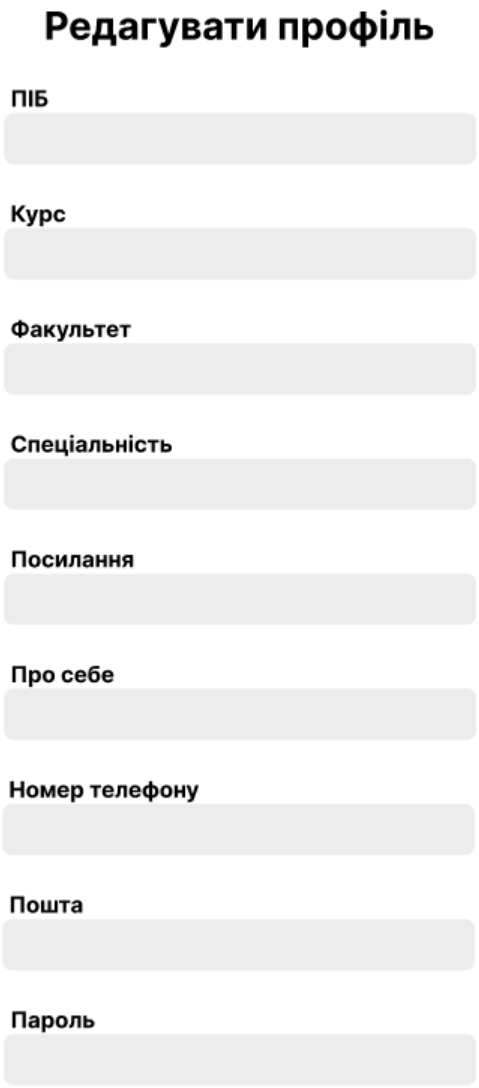


Рисунок 4.8 — Сторінка редагування профілю

Розглянемо сторінку комунікації, що представлена на рисунку 4.9. Сторінка комунікації — це інтерфейс, який надає можливість учасникам проекту спілкуватися, обмінюватися інформацією та координувати роботу в рамках проекту. Сторінка комунікації представлена загальним чатом для всієї команди або приватними повідомленнями між окремими учасниками. Представлена можливість обмінюватися файлами, документами чи іншою інформацією в

рамках проекту, обговорювати конкретні завдання, задачі або процеси прямо на сторінці цих завдань.

Ця сторінка спрямована на полегшення комунікації та співпраці між учасниками проекту, щоб ефективно керувати та координувати роботу над завданнями.

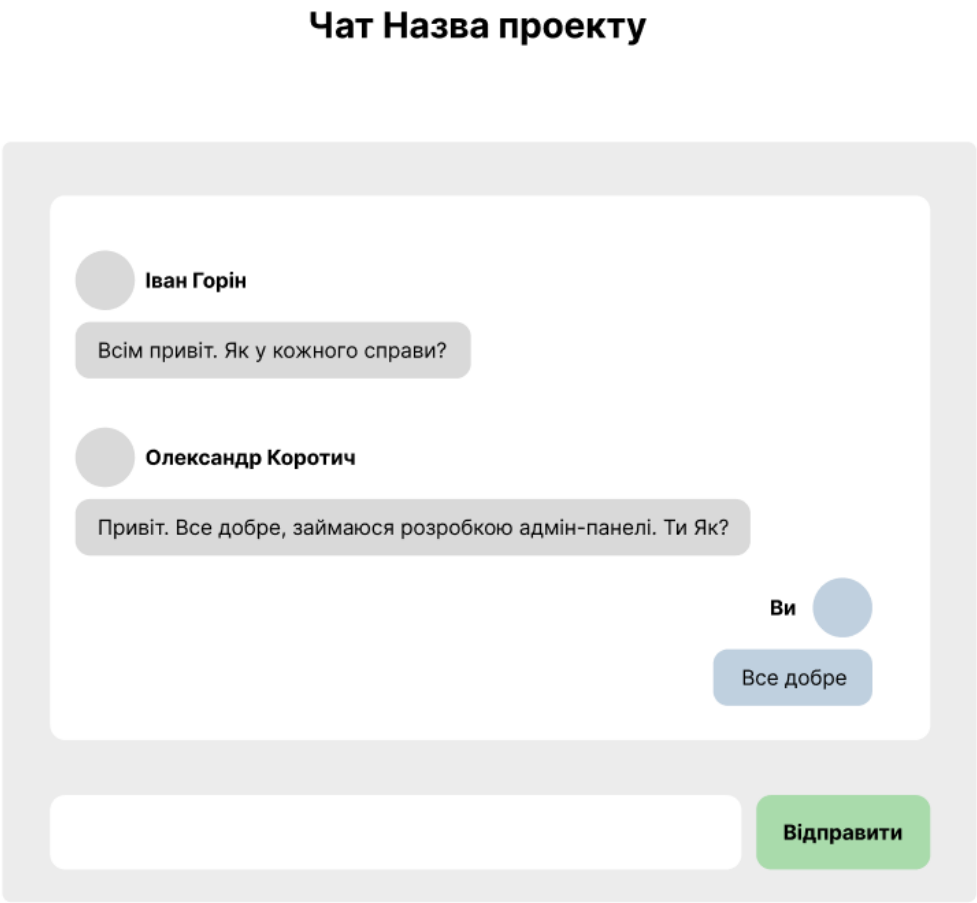


Рисунок 4.9 — Сторінка комунікації

Висновки до розділу 4

У розділі “Робота користувача з програмною системою” детально розглянуто ключовий аспект взаємодії між користувачем та програмною

системою університетського проекту. Взаємодія з системою розглянута на різних рівнях, від реєстрації та авторизації до навігації, співпраці над проектами та комунікації.

Засвідчено, що успішність взаємодії з програмною системою визначається рядом факторів, таких як зручність інтерфейсу, швидкість виконання операцій, якість результатів та підтримка користувачів. Ключовими моментами взаємодії є розуміння потреб користувачів, їхніх очікувань та розробка системи з урахуванням цих факторів.

Детально описані різні етапи роботи з системою, починаючи від реєстрації, авторизації та навігації до участі в проектах, управління завданнями та комунікації. Ці сторінки системи важливі для ефективного співробітництва та координації роботи.

Такий докладний опис підкреслює значення зручності, доступності та функціональності для користувачів, що стає ключем до успіху взаємодії користувачів з програмною системою.

Досягнення взаємодії користувача з програмною системою є важливим етапом у забезпеченні плавного та продуктивного функціонування проектів. Постійне вдосконалення і адаптація до потреб користувачів є ключовим для забезпечення успішної співпраці та досягнення максимального результату.

5 РОЗРОБЛЕННЯ СТАРТАПУ ДЛЯ КЕРУВАННЯ ПРОЦЕСАМИ РОЗРОБКИ УНІВЕРСИТЕТСЬКИХ ПРОЕКТІВ

Зараз, в епоху стрімкого розвитку технологій та наукових досліджень, створення та впровадження стартапів стає важливою складовою для перетворення наукових ідей у практичні рішення, що вирішують актуальні проблеми. Розділ “Розроблення стартапу для керування процесами розробки університетських проектів” у магістерській дисертації є не лише підсумком отриманих знань, але й відкриттям нових горизонтів у зрозумінні, як маркетингові аспекти впливають на успішне втілення наукових проектів у реальність.

Цей розділ спрямований на глибоке вивчення процесу створення стартап-проектів, починаючи від формування концепції продукту до аналізу його потенціалу на ринку та визначення стратегій його впровадження. Цей розділ є ключовим компонентом для студентів-магістрантів у розвитку їх підприємницьких навичок, розумінні принципів комерціалізації та реалізації наукових ідей.

Розділ не тільки забезпечує можливість оцінити потенціал створення та впровадження інноваційних продуктів, але й спонукає до аналізу ринкових можливостей, виявлення переваг та недоліків пропонованих ідей у порівнянні з конкурентами.

Інтегруючи теоретичні знання з практичним досвідом, цей розділ сприяє розвитку аналітичних та стратегічних навичок, необхідних для ефективного ринкового впровадження наукових розробок у формі стартап-проектів. Його цінність полягає у поглибленні розуміння студентами процесу комерціалізації інновацій та його ключової ролі у сучасній економіці.

5.1 Опис ідеї стартапу

Розглянемо таблицю 5.1, у якій представлено опис ідеї стартапу, а саме зміст ідеї, напрямки застосування та вигоди для користувача.

Таблиця 5.1 — Опис ідеї стартапу

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Програмний засіб для створення університетських проектів	1. Освітній сектор	Здобуття практичного досвіду у реальних проектах, підвищення професійних навичок, можливість знаходити та приєднуватися до цікавих проектів. Також збільшення доступу студентів до практичного навчання, можливість створення та керування університетськими проектами.
	2. Технологічні стартапи	Можливість залучати талановитих студентів до своїх проектів. Збільшення шансів знайти співробітників для реалізації ідей.

Продовження таблиці 5.1.

	3. Програмістські спільноти	Можливість знаходити та долучатися до спільних проектів. Обмін знаннями та досвідом з програмування з колегами.
	4. Дослідницькі програми	Забезпечення співпраці між студентами та ученими для реалізації наукових досліджень. Підвищення можливостей для студентських наукових робіт.
	5. Відкриті інновації	Створення відкритих платформ для обміну ідеями та співпраці над інноваційними проектами. Розвиток спільних ініціатив та новаторських ідей.

У розвитку сучасної освіти та інноваційного потенціалу студентів значну роль відіграють університетські проекти. Розробка програмного засобу, спрямованого на управління та співпрацю над такими ініціативами, відкриває широкі можливості для розвитку в освітній сфері, технологічних стартапах, програмістських спільнотах, дослідницьких програмах та відкритих інноваціях.

Цей програмний засіб створений для стимулювання практичного досвіду

студентів, сприяння їхній активній участі у проектах та розвитку інноваційних ідей. Засіб надає можливість знаходити цікаві проекти, залучати талановитих студентів до реалізації ідей у технологічній сфері, співпрацювати та обмінюватися знаннями у програмістських групах, проводити дослідницьку роботу університетського рівня та впроваджувати інноваційні проекти.

Цей інструмент створений з метою сприяти зростанню активності та розвитку у сфері освіти та технологій, надаючи користувачам широкий спектр можливостей для співпраці, навчання та реалізації ідей у вибраних напрямках застосування.

Для визначення характеристик ідеї проекту розглянемо таблицю 5.2, що представлена нижче.

Таблиця 5.2 — Визначення характеристик ідеї проекту

№ з/п	Економічні характеристики	Концепція (стратегія) конкурентів			W	N	S
		Власний проект	GitHub Classroom	Moodle			
1.	Продуктивність	Можливість оптимізувати продуктивність для відповідності унікальним потребам організації.	Зависока продуктивність через використання готової інфраструктури GitHub.	Продуктивність залежить від обсягу та складності встановленої системи.			+

Продовження таблиці 5.2.

2.	Гнучкість та налаштовуваність	Максимальна гнучкість і можливість налаштовування під конкретні потреби.	Має обмежені можливості налаштування порівняно з власним проектом.	Забезпечує певну гнучкість у налаштуванні та можливості розширення, але не на тому рівні, що у власного проекту.			+
3.	Ризики та надійність	Може мати ризики, пов'язані з нестабільністю або помилками, але може бути більш надійним у разі внутрішньої підтримки.	Зазвичай надійний, оскільки базується на великій та стабільній інфраструктурі GitHub.	Надійність може залежати від правильності налаштування та обслуговування, може бути менш стійким у порівнянні зі стандартизованими рішеннями.		+	

Продовження таблиці 5.2.

4.	Вартість впровадження	Вимагає значних витрат на розробку та впровадження власної платформи для управління проектами.	Має низькі витрати впровадження, оскільки це хмарний сервіс із встановленою інфраструктурою та платформою.	Витрати на впровадження також невеликі, оскільки це відкрита система з вільним доступом для встановлення.	+		
5.	Вартість підтримки	Потребує значних витрат на постійну підтримку, розширення функціоналу та виправлення помилок.	Зазвичай має стабільні щомісячні або річні витрати, пов'язані з підпискою на сервіс та можливими додатковими витратами на інтеграцію.	Може мати витрати на технічну підтримку та оновлення для забезпечення роботи платформи відповідно до вимог користувачів.	+		

Продовження таблиці 5.2.

6.	Оновлення та розвиток	Вимагає постійних витрат на розвиток та модернізацію.	Оновлення та нові функції включені у підписку, але додаткові функції можуть вимагати додаткових витрат.	Витрати на оновлення та адаптацію до нових стандартів можуть варіюватися в залежності від потреб користувачів.	+		

Порівнюючи економічні характеристики між власним проектом, GitHub Classroom та Moodle, можна зробити кілька висновків. GitHub Classroom вибивається високою продуктивністю та готовою інфраструктурою, тоді як Moodle, як відкрита система, має певну гнучкість, але залежить від складності встановлення та конфігурації. Однак, власний проект може бути найбільш ефективним, оскільки його гнучкість налаштування дозволяє пристосувати платформу під унікальні потреби організації. Щодо вартості, GitHub Classroom та Moodle виграють над власним проектом через менші витрати на впровадження та підтримку. Проте, обидва вимагають витрат на оновлення та адаптацію до потреб користувачів. У виборі між ними важливо враховувати конкретні потреби, фінансові можливості та стратегію організації.

5.2 Технологічний аудит ідеї проекту

Відділ “Технологічний аудит ідеї проекту” включає в себе глибокий огляд та аналіз технічних аспектів ініціативи чи стартапу. Цей підрозділ створений з

метою ретельного дослідження технічної складової проекту, виявлення сильних та слабких сторін, а також визначення можливостей для його подальшого розвитку та вдосконалення.

В рамках даного підрозділу розглядаються ключові технічні аспекти проекту, включаючи аналіз існуючих технологій, програмних рішень, архітектурних особливостей, безпеки даних та інфраструктури. Такий аудит спрямований на забезпечення ефективності, надійності та масштабованості проекту, а також виявлення можливих ризиків та шляхів їх усунення.

Відповідно до завдань цього підрозділу, проведемо глибокий розбір технічних компонентів проекту, оцінимо їхню відповідність поставленим цілям та завданням, і запропонуємо конструктивні рекомендації щодо покращення та оптимізації технічного боку проекту.

Розглянемо таблицю 5.3, на якій представлено технологічний аудит ідеї проекту.

Таблиця 5.3 — Визначення технологічної здійсненності ідеї проекту

№ з/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1.	Інтерфейс користувача	JavaScript, React.js, TailwindCSS	Наявна	Доступна безкоштовно
2.	Веб-сервер	Node.js, Nest.js	Наявна	Доступна безкоштовно
3.	База даних	MongoDB	Наявна	Доступна безкоштовно
4.	Інструмент для комунікації	WebSocket	Наявна	Доступна безкоштовно

Продовження таблиці 5.3.

5.	Хостінг	AWS	Наявна	Доступна безкоштовно
Висновок: проект реалізувати можливо. Обрана технологія реалізації: інтерфейс користувача, веб-сервер, база даних, інструмент для комунікації, хостінг.				

Використані технології доступні безкоштовно і володіють функціоналом, необхідним для виконання завдань проекту.

5.3 Аналіз ринкових можливостей запуску стартап-проекту

Цей розділ присвячений глибокому вивченню та оцінці потенційних ринків, на яких планується запускати новий стартап. Аналіз ринкових можливостей дозволяє зрозуміти сферу діяльності, ідентифікувати цільову аудиторію та конкурентні переваги, а також визначити можливі ризики та шляхи їх уникнення.

У цьому розділі вивчимо поточний стан ринку, його розмір, динаміку та тенденції розвитку. Також віддамо увагу аналізу конкурентного оточення, виявленню основних гравців на ринку, їхнім стратегіям та слабким сторонам. Крім того, проаналізуємо потреби та очікування майбутніх користувачів чи клієнтів, а також можливості врегулювання можливих проблем через новаційні підходи.

Висновки, отримані в результаті аналізу ринкових можливостей, стануть важливим джерелом для прийняття обґрунтованих рішень щодо стратегії запуску стартап-проекту та його подальшого розвитку.

Розглянемо таблиці 5.4 та 5.5, у яких звітується про труднощі та підтримку, які впливають на введення проекту на ринок.

Таблиця 5.4 — Фактори загроз

№ з/п	Фактор	Зміст загрози	Можлива реакція компанії
1.	Сильна конкуренція	Поява або наявність інших компаній, які пропонують схожі продукти чи послуги, може ускладнити проникнення на ринок та залучення клієнтів.	Ретельне дослідження конкурентів дозволить зрозуміти їхні стратегії та знайти унікальні переваги для власного продукту чи послуги.
2.	Зміни в попиті	Неочікувані зміни у попиті або уподобаннях споживачів можуть змінити відношення до продукту або послуги, що може вплинути на їхню популярність.	Слід постійно моніторити зміни в уподобаннях споживачів та швидко реагувати на їхні вимоги, враховуючи їхні потреби в розвитку продукту чи удосконаленні послуги.
3.	Технологічні зміни	Швидкий розвиток технологій може зробити поточні рішення застарілими або менш конкурентоспроможними, якщо компанія не може відповісти на ці зміни.	Швидке реагування на технологічні зміни може забезпечити компанії конкурентні переваги. Інвестування у дослідження та розробку нових технологій дозволить підтримувати актуальність продукту.

Продовження таблиці 5.4.

4.	Політичні чи регуляторні обмеження	Зміни у законодавстві, правилах або стандартах можуть створити додаткові вимоги чи обмеження для ведення бізнесу.	Активна участь у формуванні регуляторної політики або співпраця з урядовими органами може допомогти у формуванні законодавчих рамок, сприятливих для бізнесу.
5.	Нестабільність економічної ситуації	Негативні зміни на ринку або в економіці загалом, такі як рецесія, можуть призвести до скорочення бюджетів на закупівлю нових продуктів чи послуг.	Розгляд можливостей диверсифікації бізнесу або розширення продуктової лінійки допоможе зменшити вплив негативних змін у конкретному сегменті ринку.

Таблиця 5.5 — Фактори можливостей

№ з/п	Фактор	Зміст можливості	Можлива реакція компанії
1.	Зростання попиту	Виявлення та використання зростання попиту на певні продукти чи послуги у відповідності до напрямків розвитку ринку.	Зорієнтуватися на потреби ринку та розробити продукт, що відповідає актуальним вимогам споживачів.

Продовження таблиці 5.5.

2.	Нішева спеціалізація	Виділення власного проекту у вузькій, але перспективній ніші ринку, що може забезпечити більшу увагу від потенційних клієнтів.	Виділити унікальні особливості продукту чи послуги та акцентувати увагу споживачів на цих позитивних аспектах.
3.	Технологічні інновації	Використання передових технологій чи інновацій для створення конкурентних переваг та привертання уваги клієнтів.	Зосередити увагу на постійному покращенні продукту, впровадженні нових технологій та інновацій для забезпечення конкурентоспроможності.
4.	Тенденції ринку	Аналіз та використання актуальних тенденцій у споживчому поведінці, уподобаннях клієнтів, змінах у ринковій динаміці.	Розширити присутність на нові ринки або розглянути можливість експорту продукту за межі національного ринку.
5.	Глобальна доступність	Використання можливостей глобального ринку, якщо проект має потенціал для міжнародного розширення.	Укладення партнерських угод чи співпраця з іншими компаніями для розвитку та підтримки нових ринкових можливостей.

Аналіз ринкових можливостей є важливим кроком у стратегічному плануванні для будь-якого стартапу. Після ретельного вивчення факторів, що

впливають на ринок, та оцінки загроз і можливостей, стає очевидним, що успішний запуск проекту потребує не лише інноваційних ідей, але й гнучкості у реагуванні на зміни.

Загрози, такі як сильна конкуренція або технологічні зміни, потребують не лише реакції, але й постійного моніторингу. З іншого боку, розуміння ринкових можливостей, таких як зростання попиту чи технологічні інновації, відкриває шлях для розвитку та зайняття нішевих позицій.

Важливо поєднувати стратегічне планування з гнучкістю та реагувати на ринкові тенденції. Отримана інформація про ринкові умови допоможе створити стратегію введення на ринок, що буде сприяти успішному старту та подальшому розвитку проекту.

Висновки до розділу 5

Розділ “Основні положення щодо стартапу” зводиться до ключових аспектів, які визначають успішний шлях розвитку нового проекту. Розділ включає аналіз ризиків, можливостей, технічних аспектів та ринкових умов, які суттєво впливають на його успіх.

Ретельний технологічний аудит та оцінка ринкових можливостей створюють фундаментальні знання для прийняття обґрунтованих рішень у стратегічному плануванні. Аналіз ризиків, виявлення сильних та слабких сторін, а також розуміння конкурентного середовища є ключовими для успішного введення стартапу на ринок та його подальшого зростання.

Успіх стартапу вимагає не лише інноваційних ідей, але й гнучкості у відповіді на зміни. Реагування на ринкові тенденції та швидке адаптування до змін є ключовими елементами стратегії успіху. Тільки поєднуючи стратегічне планування з гнучкістю реагування на зміни в ринкових умовах можна досягти стабільного розвитку та успішного введення стартапу на ринок.

ВИСНОВКИ

Мета проекту — створення інноваційної веб-платформи для керування університетськими проектами, спрямованої на покращення співпраці між студентами та викладачами, а також надання студентам можливості отримати практичний досвід у розробці програмного забезпечення.

Технологічний стек проекту включає такі компоненти, як React.js для фронтенду, Nest.js для бекенду, MongoDB для бази даних, GitHub для CI/CD та контролю версій, Docker для контейнеризації, Heroku для розміщення та хмарна платформа для розгортання веб-додатку.

Успішне виконання проекту передбачає реалізацію усіх вищезазначених завдань, забезпечення безпеки та конфіденційності даних користувачів, публікацію платформи та її впровадження в університетську спільноту, а також подальший моніторинг та підтримку.

Аналіз аналогічних систем виявився ключовим етапом у процесі розробки. Це дозволило не лише зрозуміти, які підходи вже існують на ринку, а й вивчити їхні сильні та слабкі сторони. Виокремлення різних систем дало можливість зрозуміти, які особливості варто взяти на увагу, а що можна уникнути чи покращити власними зусиллями.

Опис програмних засобів приніс важливість вибору правильних інструментів для власного проекту. Оцінка їхньої ефективності, сумісності та можливостей покращила шлях до реалізації проекту. Крім цього, вірний вибір програмних засобів впливає на продуктивність та успішність розробки.

Принципи розробки стали фундаментом системи. ООП та SOLID принципи допомогли побудувати чітку архітектуру та забезпечити модульність коду. Ці принципи також вплинули на розподіл відповідальностей та взаємодію компонентів системи, що сприяє легкості підтримки та розвитку програми у майбутньому.

Опис методів та алгоритмів надав необхідний інструментарій для вибору

найефективніших підходів до обробки даних та виконання операцій. Розуміння алгоритмів сортування, шифрування та інших методів вирішення завдань стало ключовим елементом в розробці оптимальних стратегій.

У розділі “Робота користувача з програмною системою” детально розглянуто ключовий аспект взаємодії між користувачем та програмною системою університетського проекту. Взаємодія з системою розглянута на різних рівнях, від реєстрації та авторизації до навігації, співпраці над проектами та комунікації.

Розділ “Основні положення щодо стартапу” зводиться до ключових аспектів, які визначають успішний шлях розвитку нового проекту. Розділ включає аналіз ризиків, можливостей, технічних аспектів та ринкових умов, які суттєво впливають на його успіх.

Проект має великий потенціал для поліпшення університетського середовища, сприяючи співпраці та навчанню студентів в галузі розробки програмного забезпечення. Наявність чіткої постановки завдання та визначеного технологічного стеку допоможе здійснити ефективну розробку та розгортання платформи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hughes-Croucher T. Node: Up and Running: Scalable Server-Side Code with JavaScript 1st Edition. 2012, p. 204.

2. Офіційна документація бібліотеки React.js. URL: https://React.js.org/.

3. Офіційна документація фреймворку Nest.js. URL: https://docs.Nest.js.com/.

4. Офіційна документація мови программування TypeScript. URL: https://www.types criptlang.org/docs/.

5. Nadareishvili I. Microservice Architecture: Aligning Principles, Practices, and Culture. 2016 p. 146.

6. Bell J., Magolan G. Nest.js: A Progressive Node.js Framework Kindle Edition. 2018 p. 350.

7. Hohpe G., Woolf B. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. 2004 p. 736.

8. Zhmutska N. Construction of web-applications based on microservice architecture. 2019 p. 5.

9. Bass L. Software architecture in practice (3rd ed.). Addison-Wesley Professional, 2021 p.102.

10. Мікросервісна архітектура. URL: https://medium.com/@IvanZmerzlyi/microservices-architecture-461687045b3d.

11. Мікросервісна архітектура для початківців. URL: https://www.globallogic.com/ua/insights/blogs/microservices-architecture-for-beginners-part-one/.

12. Porcello E., Banks A. Learning GraphQL: Declarative Data Fetching for Modern Web Apps. 2018, p. 175.

13. Wieruch R. The Road to GraphQL: Your journey to master pragmatic GraphQL in JavaScript with React.js and Node.js. 2018, p. 352.

14. Mardan A. React Quickly: Painless Web Apps with React, Redux, and Graphql. 2017, p. 528.

15. Jansen H. Learning TypeScript 2.x: Develop and maintain captivating web

applications with ease, 2nd Edition. 2018, p. 536.

16. Powers S. Learning Node: Moving to the Server-Side 2nd Edition. 2016, p. 288.

17. Breseman K. Node.js for Embedded Systems: Using Web Technologies to Build Connected Devices 1st Edition. 2016, p. 266.

18. Зеленін В.А. Основні принципи побудови мікросервісних систем. Київ, 2017. 18 с.

19. Pasquali S. Mastering Node.js: Expert techniques for building fast servers and scalable, real-time network applications with minimal effort (Community Experience Distilled). 2013, p. 346.

20. Офіційна документація платформи Node.js. URL: https://Node.js.org/en.

21. Офіційна документація протоколу WebSocket. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API.

22. Офіційна документація бібліотеки Next.js. URL: https://Next.js.org/.

23. Вихідний код бібліотеки Next.js. URL: https://github.com/vercel/next.js.

24. Next.js — A brief overview. URL: https://levelup.gitconnected.com/next-js-a-brief-overview-78ede74a22b9?gi=63ac4d65bfab.

25. Making Sense of React Server Components. URL: https://www.joshwcomeau.com/react/server-components/.

26. These Are the Concepts You Should Know in React.js (After You Learn the Basics) — 2023 Edition. URL: https://betterprogramming.pub/these-are-the-concepts-you-should-know-in-react-js-after-you-learn-the-basics-2023-edition-9594f3d0f6d3.

27. What Is Nest.js? A Look at the Lightweight JavaScript Framework. URL: https://kinsta.com/knowledgebase/nestjs/.

28. NestJS: The Good, The Bad, and The Ugly. URL: https://betterprogramming.pub/nestjs-the-good-the-bad-and-the-ugly-d51aea04f267.

29. Why Use Nest.js. URL: https://devcycle.com/blog/why-use-nest-js.

30. Nest JS: The Magic Wand for Effortless Backend Development. URL: https://javascript.plainenglish.io/nest-js-the-magic-wand-for-effortless-backend-development-d2b1465d3b17.

ДОДАТОК А

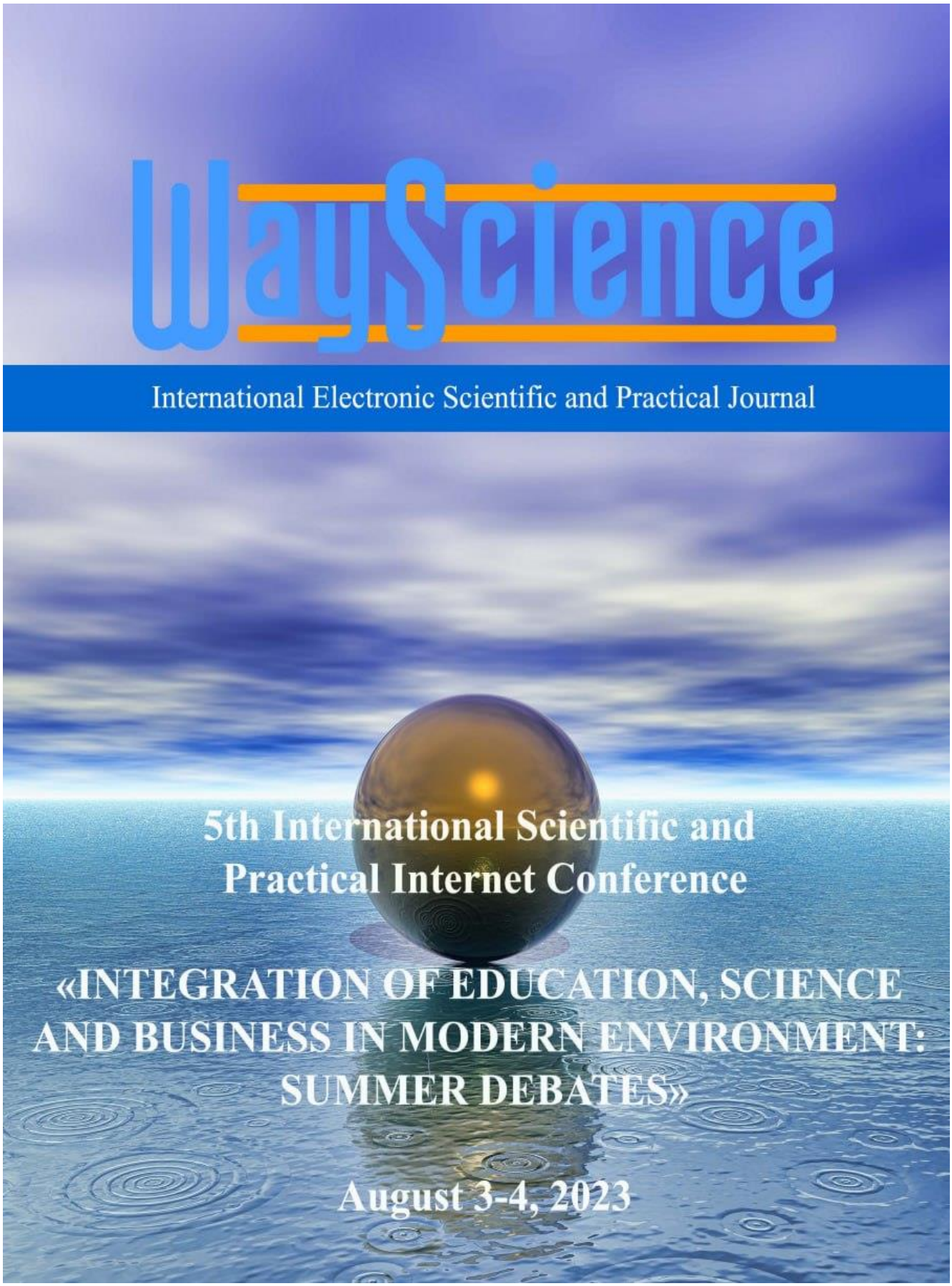
Веб-платформа для керування процесами розробки університетських проектів

Апробація результатів дисертації

УКР.НТУУ«КПІ ім. Ігоря Сікорського» ТР-з2118мп

Аркушів 3

Київ — 2023


WayScience
International Electronic Scientific and Practical Journal
5th International Scientific and
Practical Internet Conference
«INTEGRATION OF EDUCATION, SCIENCE
AND BUSINESS IN MODERN ENVIRONMENT:
SUMMER DEBATES»
August 3-4, 2023

ВЕБ-ПЛАТФОРМА ДЛЯ КЕРУВАННЯ ПРОЦЕСАМИ РОЗРОБКИ УНІВЕРСИТЕТСЬКИХ ПРОЕКТІВ

Кадирбеков І.Ю.
магістрант 2 курсу НТУУ «КПІ»

Університетське середовище постійно еволюціонує, ставши місцем активного наукового дослідження, розвитку та інновацій. У цьому контексті ефективне управління проектами є не просто важливим елементом, а справжньою ключовою складовою успіху. Швидкість змін, якість роботи, спроможність пристосуватися до нових викликів - усе це стає вирішальними факторами. Університети мають унікальний шанс стати лідерами в інноваціях, і здійснення цієї мети значною мірою залежить від ефективного керування проектами.

Нинішня доповідь має на меті пролити світло на те, як веб-платформи стають невід'ємною частиною університетських проектів. Ці платформи - це не просто інструменти. Вони є каталізаторами співпраці, обміну ідеями та спільної роботи між студентами, викладачами та зовнішніми партнерами. Вони створюють віртуальне середовище, де кожен учасник може долучитися до спільної роботи, внести свій внесок та сприяти досягненню спільної мети.

Ці веб-платформи дозволяють легко створювати завдання, призначати терміни виконання, відстежувати прогрес та координувати роботу. Вони полегшують комунікацію, сприяють обміну документами та дозволяють взаємодіяти з будь-якого місця і в будь-який час. Такий підхід до управління проектами відкриває нові можливості для швидкого реагування на зміни та ефективного ресурсного планування.

Однак важливо не тільки визнавати переваги цих платформ, але й розуміти їхні можливі ризики та виклики. Безпека даних, захист конфіденційної інформації та забезпечення надійності цифрових інструментів стають докорінними аспектами при роботі з веб-платформами університетських проектів.

Таким чином, доповідь спрямована на розкриття того, як веб-платформи не лише спрощують управління університетськими проектами, але й стимулюють інновації, сприяючи колаборації та кращому використанню ресурсів. Їхній потенціал для трансформації університетської сфери робить їх необхідними інструментами для сучасних навчальних установ.

Процес розробки проектів університетського середовища потребує постійної обміну інформацією між різними учасниками. Веб-платформи стають мостом для цієї комунікації, створюючи централізовані канали для обміну документами, ідеями та оновленнями щодо проектів. Це сприяє зменшенню часу, який потрібен на передачу інформації, та сприяє більш ефективній роботі команди. Така система дозволяє кожному учаснику бути на одній хвилі з усією командою, не обмежуючи просторових або часових рамок. Вона забезпечує консистентність інформації та створює сприятливу атмосферу для співпраці, що полегшує процес прийняття рішень та розв'язання можливих конфліктів.

Веб-платформи для університетських проектів відрізняються своєю гнучкістю. Вони не є статичними інструментами, а скоріше - адаптивними системами, які можуть налаштовуватися під конкретні потреби кожного проекту. Ця адаптивність дозволяє легко змінювати параметри проекту: встановлювати терміни, призначати відповідальних, модифікувати завдання та контролювати прогрес. Такий підхід робить платформи гнучкими і готовими до швидкого відгуку на будь-які зміни у процесі розробки. Це дозволяє ефективно використовувати ресурси, зокрема, розподіляти завдання між членами команди відповідно до їхніх навичок та можливостей. Крім того, ця адаптивність забезпечує відповідність проекту змінам у вимогах та умовах, що з'являються протягом процесу роботи. Такий підхід до управління проектами робить веб-платформи не тільки зручними, але й потужними інструментами, які допомагають оптимізувати робочі процеси та забезпечувати успішне завершення університетських проектів.

Інтелектуальний аналіз даних стає ключовим компонентом сучасних веб-платформ для університетських проектів, надаючи можливість систематично аналізувати великі обсяги інформації та витягати цінні інсайти. Цей аналіз не лише надає звітливі дані, але й перетворює ці дані у корисну інформацію для прийняття обґрунтованих рішень на кожному етапі проекту.

Однією з ключових можливостей інтелектуального аналізу є виявлення тенденцій. Він дозволяє виявити шаблони, що виникають у процесі роботи над проектом, показуючи, які аспекти працюють добре та які можуть потребувати додаткової уваги. Це дозволяє команді працювати на основі фактичних даних та пристосовувати свою стратегію до змін у реальному часі.

Прогностичні можливості інтелектуального аналізу даних є також надзвичайно важливими. Вони допомагають передбачати можливі результати, виявляти потенційні ризики та перешкоди, які можуть виникнути у процесі проектування чи реалізації. Це дає команді можливість приймати заходи заздалегідь для уникнення проблем та мінімізації ризиків.

Крім того, інтелектуальний аналіз даних допомагає узагальнювати отримані інформацію, перетворюючи масиви даних у зрозумілі та корисні висновки. Це дозволяє команді краще розуміти тенденції, спрямовувати свої зусилля та ресурси туди, де вони найбільш потрібні, і приймати обґрунтовані рішення, підкріплені фактами. Використання інтелектуального аналізу даних на кожному етапі проекту дозволяє не лише реагувати на поточні потреби, але й активно керувати майбутніми напрямками діяльності. Він стає могутнім інструментом для уникнення можливих проблем, забезпечуючи успішне завершення університетських проектів.

Забезпечення високого рівня безпеки є надзвичайно важливою складовою для будь-якої веб-платформи, особливо в контексті університетських проектів, які можуть містити конфіденційну та чутливу інформацію. Враховуючи це, важливо мати комплексний підхід до захисту даних на цих платформах.

Технічні засоби шифрування стають першим рівнем захисту. Вони дозволяють перетворити дані в незрозумілу для посторонніх осіб форму під час передачі чи зберігання. Шифрування забезпечує конфіденційність даних, перетворюючи їх у код, що може бути розшифровано тільки з допомогою відповідного ключа.

Підвищення безпеки також включає в себе процеси автентифікації та контролю доступу. Механізми автентифікації перевіряють ідентичність користувачів перед наданням доступу до системи, що дозволяє уникнути несанкціонованого входу. Контроль доступу встановлює правила, які визначають, хто, як і коли може отримувати доступ до певних даних або функцій платформи.

Управління безпекою також включає постійний моніторинг, виявлення та відповідь на потенційні загрози. Це включає в себе регулярне оновлення систем безпеки, аналіз журналів подій для виявлення незвичних або підозрілих активностей та негайну реакцію на виявлені потенційні загрози.

Загалом, веб-платформи для університетських проектів мають не лише забезпечувати ефективність та зручність, але й гарантувати найвищий рівень захисту конфіденційності та цілісності даних. Безпека є пріоритетом, і технічні засоби шифрування, автентифікації та системи контролю доступу є лише деякими з інструментів, які використовуються для досягнення цієї мети.

Висновок. Веб-платформи для управління університетськими проектами стають кардинальною частиною інфраструктури сучасного навчального середовища. Вони не просто реагують на потреби ефективного управління, але й створюють фундамент для сприяння співпраці, інновацій та стабільного розвитку. Ці платформи виступають у ролі каталізатора для співпраці між різними учасниками університетських проектів. Вони забезпечують зручні та централізовані засоби комунікації, обміну інформацією та спільної роботи, що розширює можливості колаборації.