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

Приладобудівний факультет
Кафедра комп'ютерно-інтегрованих оптичних та навігаційних систем

«На правах рукопису»
УДК 004.5

«До захисту допущено»
Завідувач кафедри

_____ Надія Бурау
(підпис)
«____» ____січня_____ 2024 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою
«Комп'ютерно-інтегровані технології та системи в приладобудуванні»
зі спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»
на тему:
«Автоматизована система створення супровідної документації»

Виконав:
студент ІІ курсу, групи ПГ-21мп
Білич Максим Валерійович _____
(підпис)

Науковий керівник:
к.т.н., доц., доцент кафедри комп'ютерно-інтегрованих оптичних та навігаційних систем
Півторак Діана Олександрівна _____
(підпис)

Консультант з розроблення стартап-проекту:
д.е.н., проф., завідувач кафедри економічної кібернетики
Бояринова Катерина Олександрівна _____
(підпис)

Рецензент:
д.т.н., проф. в.о. завідувача кафедри виробництва приладів
Антонюк Віктор Степанович _____
(підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2024 р.

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Приладобудівний факультет

Кафедра комп'ютерно-інтегрованих оптичних та навігаційних систем

Рівень вищої освіти – другий (магістерський)

Спеціальність – 151«Автоматизація та комп'ютерно-інтегровані технології»

Освітньо-професійна програма «Комп'ютерно-інтегровані системи та технології в приладобудуванні»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Н.І. Бурау

«___»_____2023 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Біличу Максиму

1. Тема дисертації «Автоматизована система створення супровідної документації», науковий керівник дисертації Півторак Діана Олександрівна, к.т.н., доцент, затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом дисертації: 29 грудня 2023 року

3. Об'єкт дослідження: є процес створення супровідної документації навчального процесу.

4. Предмет дослідження: автоматизація процесу створення супровідної документації навчального процесу, а саме індивідуальних навчальних планів студентів.

5. Перелік завдань, які потрібно розробити:

5.1 Провести огляд стану проблеми.

5.2 Провести огляд існуючих рішень у сфері автоматизованих систем документообігу.

5.3 Провести огляд існуючих засобів розроблення автоматизованої системи створення супровідної документації.

5.4 Розробити основні модулі та підсистеми.

5.5 Перевірка працездатності системи.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: таблиці, графіки, рисунки, схеми, діаграми тощо

7. Орієнтовний перелік публікацій: 1 стаття в матеріалах конференції

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розроблення стартап-проекту	Бояринова К.О., д.е.н., проф., завідувач кафедри економічної кібернетики		

9. Дата видачі завдання 15 вересня 2023 року

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Провести огляд стану проблеми	05.10.2023	
2.	Провести огляд існуючих рішень у сфері автоматизованих систем документообігу	19.10.2023	
3.	Провести огляд існуючих засобів розроблення автоматизованої системи створення супровідної документації	26.10.2023	
4.	Розробити основні модулі та підсистеми	23.11.2023	
5.	Перевірка працездатності системи	30.11.2023	
6.	Оформлення рукопису дисертації	14.12.2023	

Студент Максим БІЛИЧ

Науковий керівник дисертації Діана ПІВТОРАК

РЕФЕРАТ

Магістерська дисертація складається з 120 сторінок, в ній міститься 35 ілюстрації, 27 таблиці та використано 52 джерел.

У даній роботі розроблено автоматизовану систему створення супровідних документів в освітньому процесі, яка дозволяє формувати індивідуальні навчальні плани. Автоматизована система представлена у вигляді веб-додатку.

Мета і завдання дослідження. Метою роботи є розроблення автоматизованої системи створення супровідної документації освітнього процесу, а саме індивідуальних навчальних планів студентів.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

1. Провести огляд сучасних рішень у сфері створення супровідної документації для навчального процесу.

2. Визначити вимоги до автоматизованої системи створення супровідної документації освітнього процесу.

3. Розробити початковий шаблон індивідуальних навчальних планів студентів.

4. Розробити основні підсистеми, які реалізовують бізнес-логіку.

5. Провести тестування автоматизованої системи на прикладі створення індивідуальних навчальних планів студентів

6. Розробити стартап проект.

Об’єкт дослідження – процес створення супровідної документації освітнього процесу.

Предмет дослідження – автоматизація процесу створення супровідної документації освітнього процесу, а саме індивідуальних навчальних планів студентів з урахуванням динамічних вимог.

Апробація результатів дисертації

Результати роботи магістерської дисертації були оприлюднені на V Міжнародна науково-практична конференція «Modern research in science and education»

Публікації:

Основні результати досліджень викладені в 1 статті у збірнику праць конференцій:

Білич М. Використання автоматизованої системи супровідної документації освітнього процесу для створення індивідуального навчального плану студента / М. Білич, Д. Півторак // V Міжнародна науково-практична конференція «Modern research in science and education», 11-13.01.2024. - Чикаго, США. Архів - с. 284-289.

Ключові слова: АВТОМАТИЗОВАНА СИСТЕМА, ІНДИВІДУАЛЬНИЙ НАВЧАЛЬНИЙ ПЛАН, ГРАФІЧНИЙ ІНТЕРФЕЙС, ВЕБ-ДОДАТОК, ІНТЕРАКТИВНА ТАБЛИЦЯ, JAVASCRIPT, REACT.

ABSTRACT

The master's thesis consists of 120 pages, including 35 illustrations, 27 tables, and references to 52 sources.

This work presents the development of an automated system for generating accompanying documents in the educational process, enabling the creation of individual study plans. The automated system is implemented as a web application.

Objective and Tasks of the Research. The aim of the study is to develop an automated system for creating supporting documentation for the educational process, specifically individual study plans for students.

To achieve the set goal, the following tasks need to be addressed:

1. Conduct a review of contemporary solutions in the field of creating supporting documentation for the educational process.

2. Define the requirements for the automated system for creating supporting documentation for the educational process.

3. Develop an initial template for individual study plans for students.

4. Design the core subsystems implementing the business logic.

5. Conduct testing of the automated system using the example of creating individual study plans for students.

6. Develop a startup project.

Research Object. The object of the research is the process of creating supporting documentation for the educational process.

Research Subject. The subject of the research is the automation of the process of creating supporting documentation for the educational process, specifically individual study plans for students, taking into account dynamic requirements.

Validation of Dissertation Results:

The results of the master's thesis were presented and published at the V International Scientific-Practical Conference "Modern Research in Science and Education."

Publications:

The primary research findings are presented in one article in the conference proceedings:

Bilych M. Utilization of an Automated System for Supporting Documentation in the Educational Process to Generate an Individual Study Plan for a Student / M. Bilcyh, D. Pivtorak // V International Scientific-Practical Conference "Modern Research in Science and Education," January 11-13, 2024. Chicago, USA. Archive - p. 284-289.

Keywords: AUTOMATED SYSTEM, INDIVIDUAL STUDY PLAN, GRAPHIC INTERFACE, WEB APPLICATION, INTERACTIVE TABLE, JAVASCRIPT, REACT

ЗМІСТ

ВСТУП 11
РОЗДІЛ 1 14
ОГЛЯД СТАНУ ПРОБЛЕМИ 14
1.1. Документація в сфері освіти 14
1.2 Індивідуальний навчальний план студента 17
1.3 Огляд автоматизованих систем 20
1.4 Сучасні рішення по темі дисертації 22
1.4.1 Microsoft Excel 22
1.4.2 Microsoft 365 24
1.4.3. Сервіс «my.kpi.ua» 25
1.4.4 Формування індивідуальних планів у Яґеллонському університеті в Кракові 27
1.4.5 Огляд наукових робіт за темою магістерської дисертації 31
1.5 Мета та завдання досліджень 36
РОЗДІЛ 2 38
ВИБІР ТЕХНОЛОГІЙ ДЛЯ РОЗРОБЛЕННЯ 38
АВТОМАТИЗОВАНОЇ СИСТЕМИ 38
2.1 Вибір типу додатка 38
2.2 Вибір мови програмування 51
2.3. Вибір фреймворку 60
2.4 Вибір інтегрованого середовища розробки 65
2.5 Вибір бази даних 66
РОЗДІЛ 3 70
РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ 70
3.1 Розробка баз даних 70
3.2 Розроблення графічного інтерфейсу проектованого програмного забезпечення 75
3.3 Взаємодія з базою даних силабуса 80
3.4 Інструкція користувача 84
3.4.1. Початок роботи з веб-додатком 84
3.4.2. Головна сторінка веб-додатку 85
РОЗДІЛ 4 РОЗРОБКА СТАРТАП ПРОЕКТУ «АВТОМАТИЗОВАНА СИСТЕМА СТВОРЕННЯ СУПРОВІДНИХ ДОКУМЕНТІВ ОСВІТНЬОГО ПРОЦЕСУ» 91

4.1 Опис та технологічний аудит ідеї стартап-проекту 91
4.2 Аналіз ринкових можливостей запуску стартаппроекту 95
4.3 Розроблення ринкової стратегії проекту 105
4.4 Розроблення маркетингової програми та бізнес-моделі стартап-проекту . 108
ВИСНОВКИ 116
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 117

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

БД – бази даних

СКБД - система керування базами даних

API - application programming interface або прикладний програмний інтерфейс

ПЗ – програмне забезпечення

РСО - рейтингова система оцінювання

ОС - операційна система

GUI - графічний інтерфейс

UI - інтерфейс користувача

ООП - об'єктно-орієнтоване програмування

SPA - single-page application або односторінковий застосунок

CSV - Comma Separated Values або значення, розділені комами

ЄКТС - європейська кредитно трансферна-накопичувальна система

МКР - модульна контрольна робота

РГР - розрахунково-графічна робота

РР - розрахункова робота

ГР - графічна робота

ІТ – інформаційні технології

URL - Uniform Resource Loator або уніфікований локатор ресурсів

ER-діаграма - Entity-Relationship diagram

ВСТУП

Актуальність. Зі зростанням обсягів інформації та вимог до персоналізації навчального процесу, необхідність вдосконалення систем керування освітою стає важливішою. З урахуванням стрімкого розвитку технологій, електронний документообіг стає необхідністю для забезпечення оперативності та зручності у веденні освітньої діяльності.

Усунення недоліків існуючих автоматизованих систем документообігу сприятиме покращенню їхньої функціональності та ефективності. На сьогоднішній день існують певні проблеми, які обмежують користувачів у зручному та ефективному використанні цих систем.

Недостатня інформативність наявних систем призводить до обмеженої кількості інформації щодо індивідуальних навчальних планів та дисциплін. Це ускладнює студентам зрозуміти аспекти їхнього навчання та може впливати на якість їх освітнього процесу.

Також важливо вирішити проблему недостатньої інтеграції існуючих систем з іншими елементами університетської інфраструктури, такими як бібліотеки чи формування силабусів. Не ефективна взаємодія може призвести до труднощів у доступі до різних ресурсів та управлінні ними.

Брак інтерактивності в існуючих системах, особливо в навчальних, обмежує можливості взаємодії студента з індивідуальними навчальними планами. Відсутність інтерактивних інструментів може призвести до зниження зацікавленості студентів у навчанні та несприятливо вплинути на їхню активність.

Тому розроблення автоматизованої системи створення супровідної документації освітнього процесу, а саме індивідуальних навчальних планів студентів є актуальною задачею

Мета і завдання дослідження. Метою роботи є розроблення автоматизованої системи створення супровідної документації освітнього процесу, а саме індивідуальних навчальних планів студентів.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

1. Провести огляд сучасних рішень у сфері створення супровідної документації для навчального процесу.

2. Визначити вимоги до автоматизованої системи створення супровідної документації освітнього процесу.

3. Розробити початковий шаблон індивідуальних навчальних планів студентів.

4. Розробити основні підсистеми, які реалізовують бізнес-логіку.

5. Провести тестування автоматизованої системи на прикладі створення індивідуальних навчальних планів студентів

6. Розробити стартап проект.

Об’єкт дослідження – процес створення супровідної документації освітнього процесу.

Предмет дослідження – автоматизація процесу створення супровідної документації освітнього процесу, а саме індивідуальних навчальних планів студентів з урахуванням динамічних вимог.

Наукова новизна магістерської дисертації полягає у покращенні користувацького досвіду шляхом вдосконалення графічного інтерфейсу користувача за допомогою введення інтерактивних дизайну та елементів відображення індивідуального навчального плану студента.

Практична значущість магістерської дисертації полягає в тому, що розроблено програмне забезпечення для автоматизованої системи створення супровідних документів освітнього процесу, використовуючи для цього шаблон індивідуального навчального плану студента.

Апробація результатів дисертації

Результати роботи магістерської дисертації були оприлюднені на V Міжнародна науково-практична конференція «Modern research in science and education»

Публікації:

Основні результати досліджень викладені в 1 статті у збірнику праць конференцій:

Білич М. Використання автоматизованої системи супровідної документації

освітнього процесу для створення індивідуального навчального плану студента / М. Білич, Д. Півторак // V Міжнародна науково-практична конференція «Modern research in science and education», 11-13.01.2024. - Чикаго, США. Архів - с. 284-289.

РОЗДІЛ 1

ОГЛЯД СТАНУ ПРОБЛЕМИ

Освіта - це процес набуття знань, навичок, цінностей та розуміння, який сприяє розвитку людини і підготовці її до участі в суспільстві. Освіта може відбуватися в різних формах і на різних рівнях, включаючи формальну, неформальну і інформальну освіту [1].

Основні аспекти освіти включають в себе [1]:

- Знання: Освіта передбачає засвоєння інформації про різні аспекти світу, включаючи природні науки, гуманітарні науки, технічні знання та інше.
- Навички: Освіта розвиває практичні навички, які можуть бути застосовані в різних сферах життя, такі як навички роботи з інформаційними технологіями, мови, технічні навички і багато інших.
- Цінності: Освіта також передає цінності і моральні норми суспільства, допомагаючи індивідам розуміти, що є важливим та морально прийнятним.
- Розвиток особистості: Освіта сприяє розвитку особистості, включаючи розумовий, емоційний та соціальний розвиток.
- Сприяння суспільству: Освіта має важливий вплив на розвиток суспільства. Вона допомагає підготувати населення до різних ролей у суспільстві і сприяє економічному, культурному та науковому розвитку.

Освіта може здійснюватися через навчання в школах, університетах, професійних курсах, а також через самонавчання та навчання в життєвому процесі. Вона є фундаментальним чинником розвитку і успішності суспільства та людини.

1.1. Документація в сфері освіти

Ведення ділової документації є важливим і відповідальним напрямом діяльності закладу освіти. Адже будь-який аспект роботи має бути правильно задокументованим у відповідності з чинними нормами законодавства, а саме Законів України «Про освіту», «Про вищу освіту», наказів Міністерства освіти і науки України. Тож ця справа трудомістка та доволі складна. Є чіткі вимоги до

оформлення документів, які необхідно дотримуватися, а саме постанова Кабінету Міністрів України від 22.07.2015 №645 «Про документи про професійну (професійно-технічну) освіту державного зразка і додатки до них», постанова Кабінету Міністрів України від 09.09.2020 №811 «Про документи про вищу освіту (наукові ступені)» (чинна з 1 січня 2021 року), наказ Міністерства освіти і науки України від 10.02.2021 №164 «Про затвердження Методичних рекомендацій щодо опису документів про вищу освіту (наукові ступені) та додатка до них, академічної довідки та Методичних рекомендацій щодо заповнення додатка до диплома європейського зразка».

Прикладами документації у сфері освіти може бути: силабус, щоденник практики, графік навчального процесу, розклад занять, індивідуальний навчальний план студента, тощо.

Силабус - це офіційний документ, що містить деталі та інформацію про освітній компонент (дисципліну). Силабус допомагає структурувати та організувати процес навчання, надаючи студентам і викладачам чіткий план, що включає в себе опис навчальної дисципліни, її мета, предмет вивчання та результати навчання, пререквізити та постреквізити дисципліни (місце в структурно-логічній схемі навчання за відповідною освітньою програмою), зміст навчальної дисципліни, навчальні матеріали та ресурси, методика опанування навчальної дисципліни (освітнього компонента), самостійну роботу студента, політику навчальної дисципліни (освітнього компонента), види контролю та рейтингова система оцінювання результатів навчання (РСО). Силабус розробляється викладачем чи навчальним закладом і може бути докладною інструкцією для всіх учасників навчального процесу.

Силабус допомагає уникнути непорозумінь і забезпечує послідовність і якість навчання, роблячи навчальний процес більш структурованим та ефективним [2].

Основним призначенням Силабусу є [2]:

- ознайомлення здобувачів та інших учасників освітнього процесу зі змістом дисципліни, політикою викладача, дедлайнами, критеріями та засобами оцінювання результатів навчання тощо;

- встановлення відповідності змісту освіти освітній програмі та стандартам вищої освіти під час акредитації;
- встановлення відповідності при зарахуванні результатів навчання, отриманих в інших закладах освіти (академічна мобільність), за іншими освітніми програмами, у попередні роки, а також у неформальній та інформальній освіті.

Силабус складається на всю навчальну дисципліну, якщо дисципліна односеместрова, або на освітній компонент – складову дисципліни, за яким передбачено семестровий контроль. Силабус може складатися на багатосеместрову дисципліну в рамках одного року навчання та за умови представлення окремих РСО для кожного освітнього компонента [2].

Для загальноуніверситетських дисциплін рекомендовано для різних груп спеціальностей складати окремі Силабуси, в яких буде відображена специфіка змістовної частини курсу, але РСО має бути уніфікованою [2].

Щоденник практики - це документ, який використовується для фіксації подій, досвіду, спостережень і вражень, які виникають під час проведення практики або стажування в певній галузі або професії. Цей інструмент дозволяє особам відстежувати та аналізувати свій прогрес, вивчати здобуті знання та вміння [3].

Основні характеристики щоденника практики включають в себе [3]:

- Записи про події: У щоденнику фіксуються всі важливі події та активності, які відбулися під час практики. Це може включати в себе зустрічі, консультації з керівником практики, завдання та інше.
- Рефлексія: Однією з ключових функцій щоденника практики є можливість відображення над подіями і враженнями.
- Вивчення та навчання: Щоденник може містити записи про нові знання, які було набуто, та рефлексію над тим, як вони можуть бути застосовані в майбутньому.
- Планування: Окрім фіксації минулих подій, щоденник практики також може містити плани на майбутні дії та цілі, які потрібно досягти.
- Досягнення: Цей документ може використовуватися для фіксації досягнень, важливих моментів.

Щоденник практики може бути корисним інструментом для студентів, стажерів, фахівців у різних галузях, а також для тих, хто бажає активно відстежувати свій розвиток та вдосконалення в освоєнні нових навичок та професійній діяльності.

Графік навчального процесу – це документ, який визначає коли та які освітні події відбуваються протягом певного навчального або академічного періоду. Цей графік включає в себе розподіл різних освітніх заходів, коли проводиться навчання, практика, канікули, захисти кваліфікаційних робіт, які відбуваються в рамках освітнього процесу в навчальному закладі [4].

Розклад занять - це документ або графічне представлення, яке визначає розподіл часу та послідовність занять, які проводяться в навчальному закладі, організації або курсі. Він включає інформацію про дні тижня, години та місце проведення різних видів навчальних або робочих активностей, таких як: лекційних, практичних та лабораторних занять, семінарів та інше. Розклад занять є важливим інструментом для організації навчання та робочого процесу.

Індивідуальний навчальний план студента – це документ, який розробляється для студента з урахуванням його особистих потреб, інтересів та можливостей в навчанні. Цей план визначає конкретні навчальні цілі, завдання та методи, які допомагають студентові досягти найкращого успіху в навчанні

1.2 Індивідуальний навчальний план студента

Індивідуальний навчальний план — документ, що визначає послідовність, форму і темп засвоєння здобувачем освіти освітніх компонентів освітньої програми з метою реалізації його індивідуальної освітньої траєкторії та розробляється закладом освіти у взаємодії із здобувачем освіти за наявності необхідних для цього ресурсів. Складається відповідальною особою від випускаючої кафедри на кожний наступний навчальний рік наприкінці попереднього навчального року [5].

Індивідуальний навчальний план студента містить перелік нормативних та вибіркових дисциплін (освітніх компонентів) в межах визначених кредитів ЄКТС (Європейська кредитно трансферна-накопичувальна система) на навчальний рік.

Вибір студентом освітніх компонентів здійснюється з каталогів вибіркових дисциплін передбачених відповідними освітньою програмою та навчальним планом. Каталоги вибіркових дисциплін формуються відповідно до порядку встановленого в КПІ ім. Ігоря Сікорського. Всі освітні компоненти, внесені до індивідуального навчального плану студента, є обов'язковими для вивчення [5].

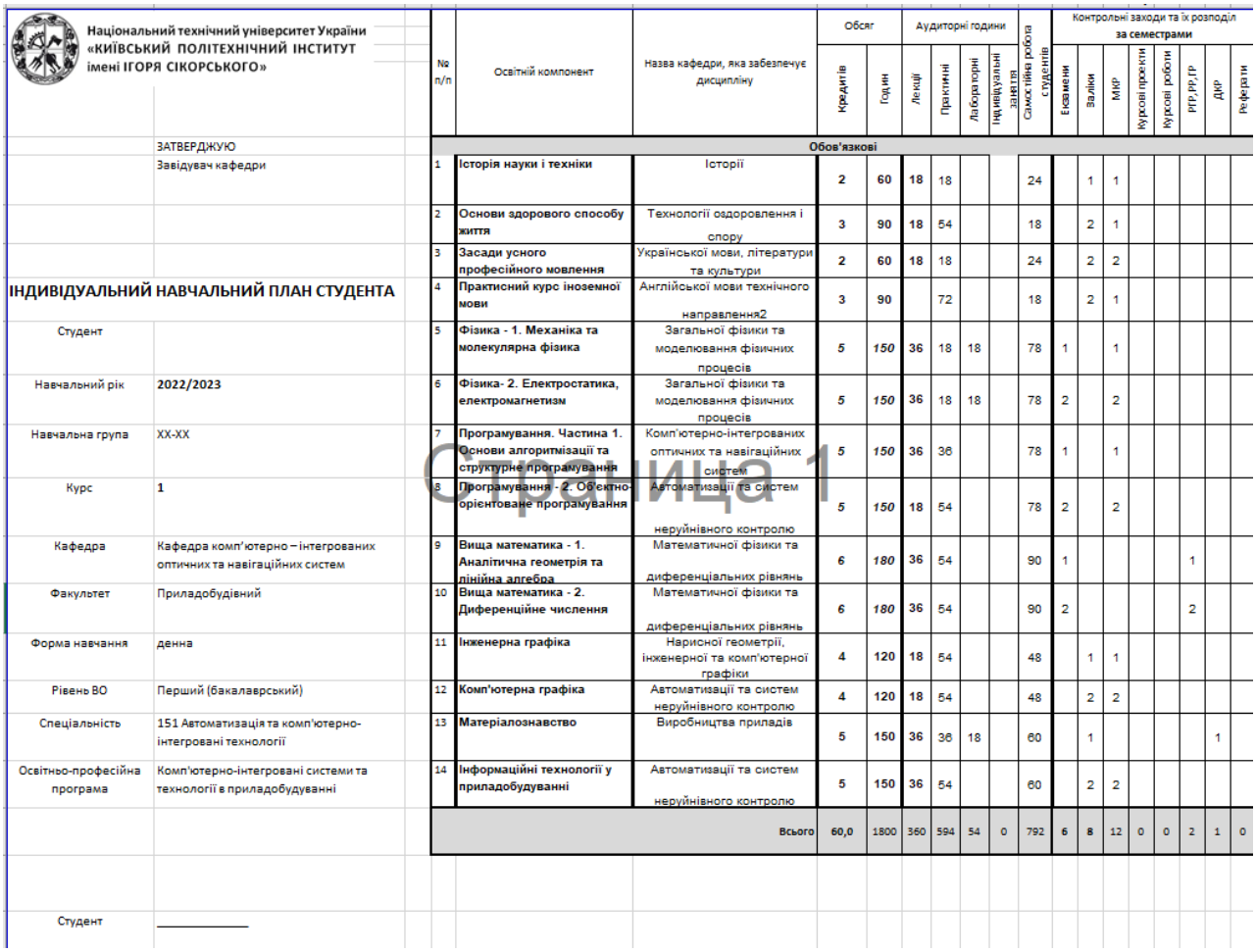
Університет з метою сприяння виконанню індивідуального навчального плану надає студентам можливість користуватися навчальними приміщеннями, бібліотекою, навчальною, навчально-методичною і науковою літературою, обладнанням, устаткуванням та іншими засобами навчання на умовах, визначених правилами внутрішнього розпорядку [5].

Виконання індивідуального навчального плану здійснюється відповідно до графіку навчального процесу, затвердженого в КПІ ім. Ігоря Сікорського на навчальний рік, та складеного за ним розкладу занять і заходів поточного, календарного та семестрового контролю. За виконання індивідуального навчального плану персональну відповідальність несе студент.

Невиконання студентом індивідуального навчального плану у встановлені терміни є підставою для його відрахування з Університету. Виконання вимог індивідуального навчального плану є умовою переведення студента на наступний курс навчання, допуску до випускної атестації [5].

Індивідуальні плани студентів роздруковується у двох примірниках, один з яких знаходяться у самого студента, а другий – в деканаті.

Приклад індивідуального навчального плану студента ~~в~~ КПІ ім. Ігоря Сікорського, наведено на рис. 1.1.



Національний технічний університет України
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ЗАТВЕРДЖУЮ
Завідувач кафедри

ІНДИВІДУАЛЬНИЙ НАВЧАЛЬНИЙ ПЛАН СТУДЕНТА

Студент	
Навчальний рік	2022/2023
Навчальна група	ХХ-ХХ
Курс	1
Кафедра	Кафедра комп'ютерно – інтегрованих оптичних та навігаційних систем
Факультет	Приладобудівний
Форма навчання	денна
Рівень ВО	Перший (бакалаврський)
Спеціальність	151 Автоматизація та комп'ютерно-інтегровані технології
Освітньо-професійна програма	Комп'ютерно-інтегровані системи та технології в приладобудуванні
Студент	_____

№ п/п	Освітній компонент	Назва кафедри, яка забезпечує дисципліну	Обсяг		Аудиторні години				Самостійна робота студентів	Контрольні заходи та їх розподіл за семестрами							
			Кредитів	Годин	Лекції	Практичні	Лабораторні	Індивідуальні заняття		Екзамени	Заліки	МКР	Курсові проекти	Курсові роботи	РГР, РР, ГР	ДКР	Реферати
		Обов'язкові															
1	Історія науки і техніки	Історії	2	60	18	18			24		1	1					
2	Основи здорового способу життя	Технології оздоровлення і спору	3	90	18	54			18		2	1					
3	Засади усного професійного мовлення	Української мови, літератури та культури	2	60	18	18			24		2	2					
4	Практисний курс іноземної мови	Англійської мови технічного направлення2	3	90		72			18		2	1					
5	Фізика - 1. Механіка та молекулярна фізика	Загальної фізики та моделювання фізичних процесів	5	150	36	18	18		78	1		1					
6	Фізика- 2. Електростатика, електромагнетизм	Загальної фізики та моделювання фізичних процесів	5	150	36	18	18		78	2		2					
7	Програмування. Частина 1. Основи алгоритмізації та структурне програмування	Комп'ютерно-інтегрованих оптичних та навігаційних систем	5	150	36	36			78	1		1					
8	Програмування - 2. Об'єктно-орієнтоване програмування	Автоматизації та систем неруйнівного контролю	5	150	18	54			78	2		2					
9	Вища математика - 1. Аналітична геометрія та лінійна алгебра	Математичної фізики та диференціальних рівнянь	6	180	36	54			90	1					1		
10	Вища математика - 2. Диференційне числення	Математичної фізики та диференціальних рівнянь	6	180	36	54			90	2					2		
11	Інженерна графіка	Нарисної геометрії, інженерної та комп'ютерної графіки	4	120	18	54			48		1	1					
12	Комп'ютерна графіка	Автоматизації та систем неруйнівного контролю	4	120	18	54			48		2	2					
13	Матеріалознавство	Виробництва приладів	5	150	36	36	18		60		1					1	
14	Інформаційні технології у приладобудуванні	Автоматизації та систем неруйнівного контролю	5	150	36	54			60		2	2					
		Всього	60,0	1800	360	594	54	0	792	6	8	12	0	0	2	1	0

Рис 1.1 – Приклад індивідуального навчального плану студента

В таблиці вказується наступна інформація:

- назва освітніх компонентів;
- назва кафедри, яка забезпечує дисципліну;
- загальний обсяг кредитів та годин;
- кількість аудиторних годин (лекційних, практичних та лабораторних занять , індивідуальні заняття);
- кількість годин, ~~які~~ відведених на самостійні роботи студентів;
- контрольні заходи, а саме екзамени, заліки, модульна контрольна робота (МКР), курсові проекти, курсові роботи, розрахунково-графічна робота (РГР), розрахункова робота (РР), графічна робота (ГР), реферати.

У студентів однієї групи список обов'язкових освітніх компонентів, які вказані в індивідуальних навчальних планах співпадають, відрізняються лише дисципліни, які обирали студенти. Індивідуальний план допомагає студентам або учням більше активно приймати участь у власному навчанні та здобуванні знань,

а також сприяє розвитку їхнього самореґулюючого підходу до освіти. Він є важливим інструментом для підтримки індивідуального розвитку кожного здобувача.

1.3 Огляд автоматизованих систем

Автоматизація торкнулася всіх сфер нашого життя. У сучасному університеті її елементом є системи електронного документообігу. Адже робота з документами: як внутрішніми, так і зовнішніми – вважається найбільш трудомісткою. Оптимізувати, а також значно спростити процес допомагає спеціальні сервіси.

Спеціалізовані сервіси, які допомагають організувати електронний документообіг, впроваджуються в державних установах. У комерційному секторі вони допомагають знизити трудовитрати і прискорити процес оформлення документів.

Автоматизація — це використання технологій виконання завдань, у яких участь людини зводиться до мінімуму. Сюди входять корпоративні програми, такі як автоматизація бізнес-процесів, автоматизація мережі, автоматизація інтеграції між системами, промислова автоматизація, така як робототехніка, а також споживчі програми та багато іншого [6].

Існують такі види автоматизації [6]:

- Базова автоматизація або автоматизація задач дозволяє автоматизувати прості завдання, що повторюються. Цей рівень автоматизації передбачає оцифрування роботи за рахунок використання автоматизації для оптимізації та централізації рутинних завдань. Це допомагає усунути помилки, прискорити темпи роботи та звільнити час людей до виконання більш цінної і значимої роботи.

- Автоматизація процесів вимагає складніших і повторюваних багатоетапних процесів за допомогою інтеграції з кількома системами та його автоматизації. Цей рівень автоматизації забезпечує однаковість та прозорість керування бізнес- та IT-процесами (IT – інформаційні технології). Використання автоматизації процесів може підвищити продуктивність та ефективність

освітнього процесу. Він також може дати нове розуміння проблем бізнесу та IT та запропонувати рішення, засновані на правилах. Аналіз процесів та автоматизація робочих процесів, а також керування бізнес-процесами є прикладами автоматизації процесів.

- Найсучасніший рівень автоматизації – це інтелектуальна автоматизація. Вона поєднує в собі автоматизацію з можливостями штучного інтелекту та машинного навчання. Це означає, що машини, які автоматизація може постійно «вчитися» та створювати, дозволяють краще приймати рішення та діяти на основі даних з минулих ситуацій, з якими вони стикалися та аналізували. Наприклад, у сфері обслуговування клієнтів віртуальні помічники на базі штучного інтелекту та машинного навчання можуть зменшити витрати, одночасно розширюючи можливості як клієнтів, так і агентів, створюючи оптимальний досвід обслуговування клієнтів.

Для максимальної ефективності та вдосконалення робочого процесу, важливим кроком є не лише автоматизація, але й впровадження системи автоматизованого електронного документообігу.

Автоматизований електронний документообіг розширює можливості бізнесу, навчального закладу, щодо обробки та обміну інформацією. Замість традиційного паперового документообігу, ця автоматизована система дозволяє ефективно керувати документами, вносити зміни в робочі процеси та підвищує швидкість прийняття рішень

Практично кожна система автоматизованого електронного документообігу виконує ряд стандартних функцій [7]:

- Створювати нові документи, як по підготовленим шаблонах, так і без них;
- Завантажувати документи, завантажені з різних систем (сканер електронної пошти, файлової системи тощо);
- Документообіг на підприємстві дозволяє порівнювати версії редагування і стежити за оновленнями версій;
- Систематизувати каталоги документів і зберігати їх;
- Формувати завдання і контролювати їх виконання;

- Колективно працювати з документами, використовуючи внутрішню мережу;
- Вести журнал поширення.

1.4 Сучасні рішення по темі дисертації

1.4.1 Microsoft Excel

Microsoft Excel може бути корисним інструментом для формування індивідуальних навчальних планів студентів, але він також має свої недоліки та переваги. Переваги використання Microsoft Excel для формування індивідуальних навчальних планів студентів [8]:

- Спрощена структура: Excel надає можливість легко створювати таблиці та графіки, що дозволяє структурувати інформацію та індивідуальний навчальний план студента в зручному форматі.

Розрахунки та формули: Excel дозволяє автоматизувати розрахунки та створювати формули для обчислення загальної кількості кредитів, загальної кількості аудиторних годин (лекційні, практичні та лабораторні заняття), загальної кількість годин відведених для самостійної роботи студента, загальної кількості контрольні заходи (екзаменів, заліків, МКР, РГР, РР, ГР, рефератів), та інших важливих показників.

- Гнучкість: Excel дозволяє змінювати та адаптувати індивідуальні навчальні плани студентів в разі зміни цілей, курсів або інших факторів.
- Візуалізація даних: Excel дозволяє створювати графіки та діаграми для візуалізації прогресу та результатів студента, що може полегшити аналіз і відстеження просування.

Недоліки використання Microsoft Excel для формування індивідуальних навчальних планів студентів [8]:

- Складність для початківців: Для тих, хто не знайомий з Excel, програма може виявитися складною та вимагати додаткового часу на вивчення.
- Одночасний доступ: Робота над індивідуальним навчальним планом у форматі Excel може бути ускладнена, коли потрібно спільно працювати над ним,

так як потрібен доступ до одного документу одночасно для декількох користувачів.

- Обмежені можливості спільної роботи: Якщо студенти або викладачі хочуть спільно працювати над планами, Excel може бути менш зручним для спільної роботи в режимі реального часу порівняно з хмаровими платформами, такими як Google Документи.
- Можливість помилок: При ручному введенні даних і формул можуть виникати помилки, які можуть вплинути на точність розрахунків та аналізу.

Загалом, використання Microsoft Excel для створення індивідуальних навчальних планів студентів має свої переваги, особливо для тих, хто вже володіє базовими навичками роботи з цією програмою. Однак важливо враховувати індивідуальні потреби та можливості кожного користувача та враховувати альтернативи, такі як онлайн-платформи для створення та спільного використання індивідуальних навчальних планів.

Приклад створеного індивідуального навчального плану, розробленого за допомогою програмного забезпечення Excel наведено на рис. 1.2.

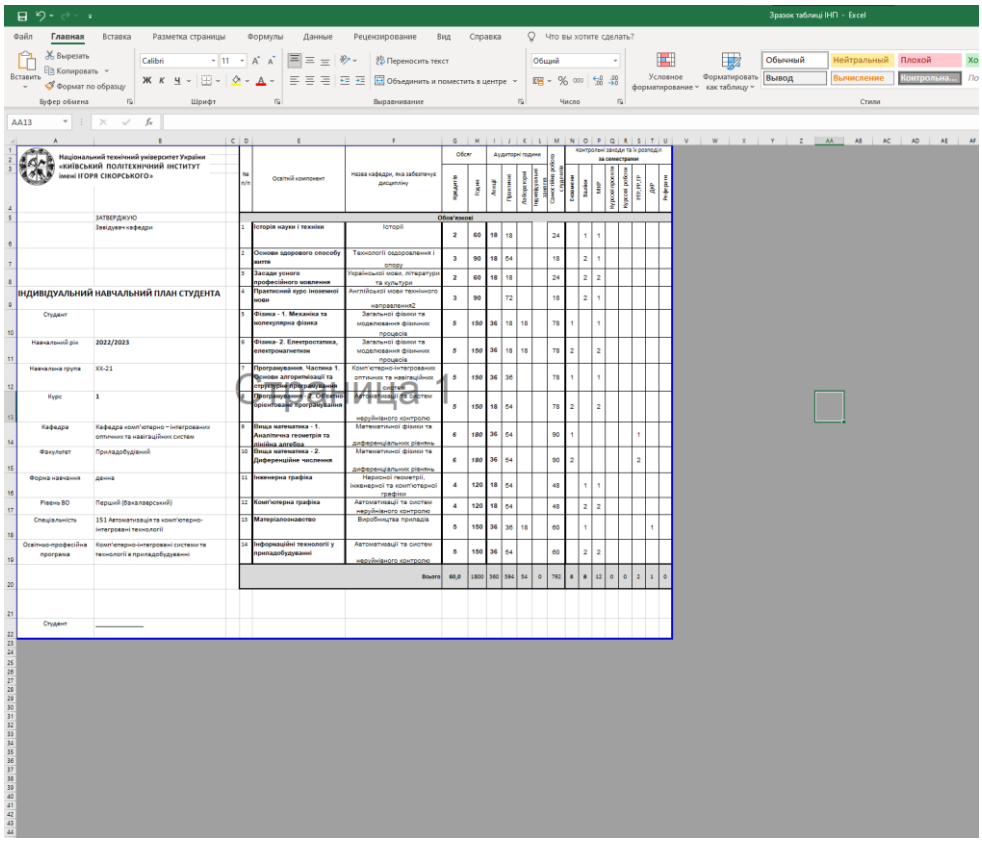


Рис 1.2. – Створення індивідуального навчального плану в Microsoft Excel

1.4.2 Microsoft 365

Microsoft 365 - це платний хмарний інтернет-сервіс і програмне забезпечення компанії Microsoft. Хмарний формат означає, що дані зберігаються в центрі обробки даних, а не на комп'ютері, що забезпечить користувачам доступ до документів і даних через браузер з різних пристроїв з можливістю виходу в Інтернет [9].

Використання пакету Microsoft 365 для формування індивідуального навчального плану студента має свої переваги та недоліки.

До основних переваг використання Microsoft 365 для формування індивідуальних навчальних планів студента можна віднести [9]:

- Інтеграція з Microsoft Office: Microsoft 365 надає доступ до різноманітних інструментів, таких як Word, Excel, PowerPoint та OneNote, які можуть бути використані для створення документів, таблиць, презентацій і записів, необхідних для індивідуальних навчальних планів.
- Зручність у спільній роботі: Microsoft 365 дозволяє спільно працювати над документами та планами в режимі реального часу, використовуючи хмарові послуги, що полегшує спільну роботу студентів та викладачів.
- Зберігання та доступність: Документи можуть бути збережені в хмарному сховищі (OneDrive або SharePoint), що дозволяє зручно зберігати та отримувати доступ до них з будь-якого пристрою з підключенням до Інтернету.
- Безпека даних: Microsoft 365 має високі стандарти безпеки, включаючи можливості захисту даних та керування доступом до документів.

До недоліків використання Microsoft 365 для формування індивідуальних навчальних планів студента відносяться [9]:

- Вартість: Microsoft 365 - це комерційний продукт, тому користування ним може вимагати витрат на ліцензії, що може бути обмежено для студентів з фінансовими обмеженнями.
- Вимоги до технічних засобів: Для повного використання Microsoft 365 потрібен доступ до Інтернету та сучасний комп'ютер або інший пристрій.
- Вивчення інтерфейсу: Деяким користувачам може знадобитися час для вивчення інтерфейсу та функціоналу Microsoft 365.

- Залежність від Інтернету: Робота з хмарними послугами вимагає постійного підключення до Інтернету, що може бути проблематичним у випадку обмеженого доступу до мережі.

Загалом, Microsoft 365 може бути потужним інструментом для створення та керування індивідуальними навчальними планами студентів завдяки своїм функціональним можливостям та інтеграції з іншими інструментами Office. Однак варто розглядати всі переваги та недоліки та розглядати альтернативи, зокрема безкоштовні або відкриті інструменти для створення та керування індивідуальними навчальними планами.

Приклад інструментів, які є у Microsoft 365 наведено на рис. 1.3.

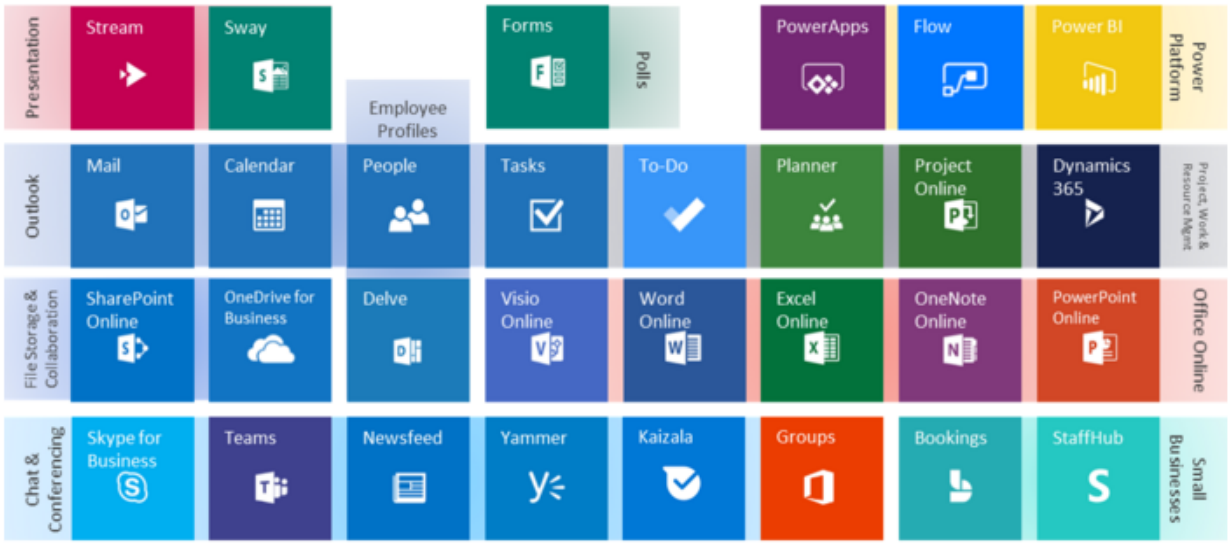


Рис 1.3. – Інструменти Microsoft 365

1.4.3. Сервіс «my.kpi.ua»

MyKPI (My Key Performance Indicators) - це інформаційна платформа, яка надає студентам можливість створювати та відстежувати індивідуальні навчальні плани. Головну сторінку сайту наведено на рис 1.4.

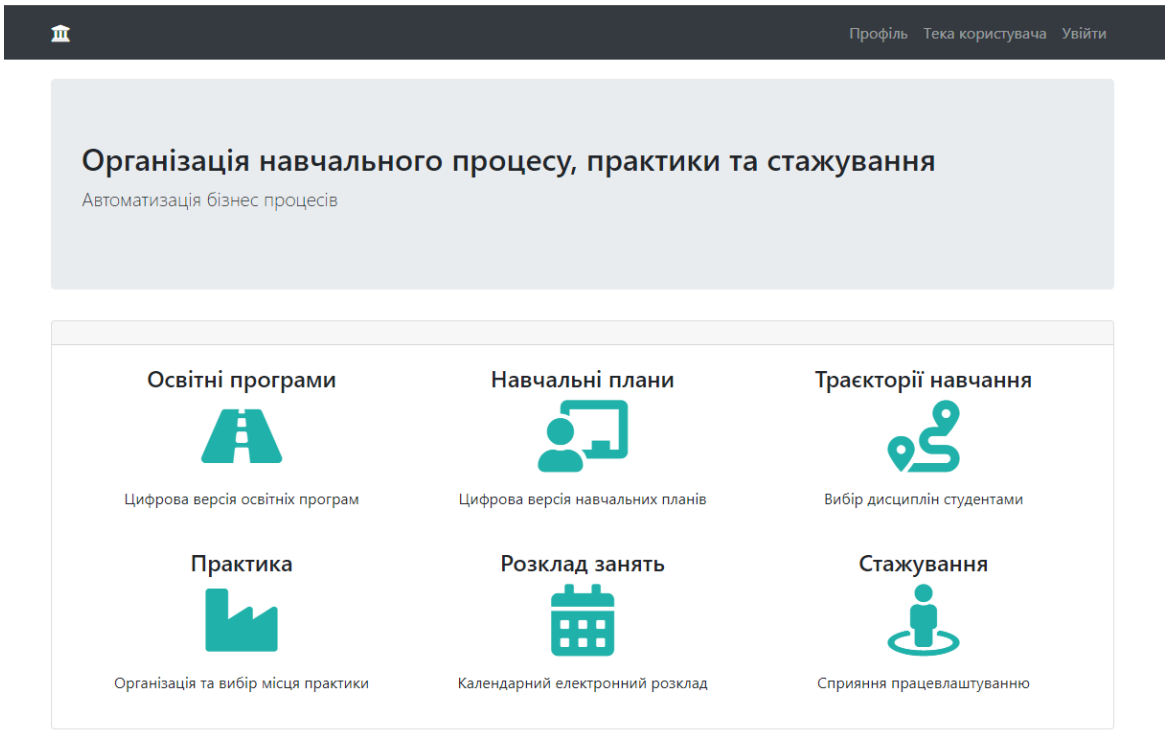


Рис 1.4. – Головна сторінка сайту

Ось деякі переваги та недоліки використання MyKPI для індивідуальних навчальних планів студентів:

- Зручний доступ до даних: MyKPI надає студентам зручний спосіб отримання доступу до своїх навчальних даних та інформації про академічний прогрес. Вони можуть переглядати цю інформацію в будь-який зручний для них момент, що полегшує самостійне відстеження навчальних цілей.
- Персоналізація: Система дозволяє створювати індивідуальні навчальні плани, які враховують їхні особисті цілі, інтереси та потреби. Це сприяє більш ефективному навчанню та розвитку.
- Зручність для адміністрації: Для адміністрації навчальних закладів, використання MyKPI спрощує процес керування та моніторингу навчальних планів. Вони можуть легко отримувати доступ до даних та аналізувати їх.
- Авторизація та захист даних: Система MyKPI надає можливість авторизуватися, що допомагає захищати особисті дані студентів та їхні навчальні досягнення. Це важливо для забезпечення конфіденційності та безпеки інформації. Приклад авторизації наведено на рис 1.5.

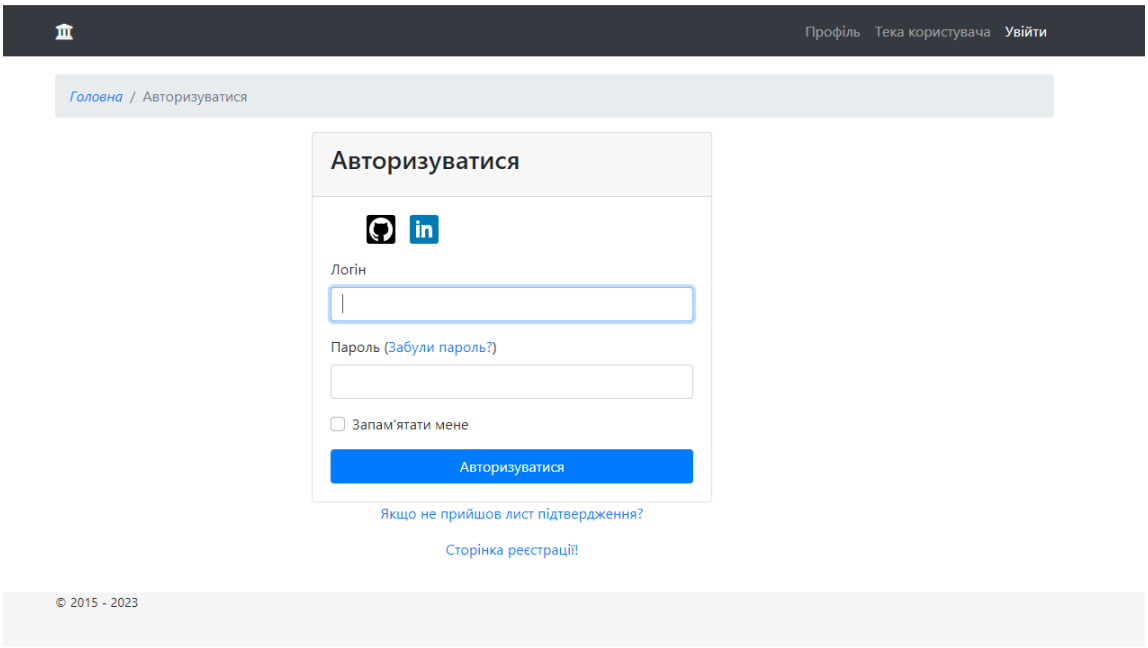


Рис 1.5. – Процес авторизації на сайті

Недоліки використання MyKPI для створення індивідуальних навчальних планів студента:

- Навчання та адаптація: Використання платформи може вимагати часу та навчання, особливо для студентів, які не володіють необхідними навичками. Це може займати додатковий час та зусилля.
- Залежність від Інтернету: Робота з хмарними послугами вимагає постійного підключення до Інтернету, що може бути проблематичним у випадку обмеженого доступу до мережі.

1.4.4 Формування індивідуальних планів у Яґеллонському університеті в Кракові

Яґеллонський університет є одним з найкращих закладів вищої освіти у Польщі, а у світовому рейтингу входить до 300х найкращих університетів. На рис 1.6 показано список дисциплін, які будуть вивчатися студентом, а на рис 1.7 продемонстровано, що при натисканні на назву дисципліни, отримується більш детальна інформація про дисципліну.

Student w ciągu całego I roku ma łącznie zrealizować 2 przedmioty fakultatywne z puli fakultetów kierunkowych, po jednym w każdym semestrze.
Student w ciągu całego II roku ma zrealizować następującą liczbę przedmiotów fakultatywnych: 4 przedmioty fakultatywne w ramach specjalności (po dwa w semestrze) oraz 2 kursy ogólnoinstytutowe (po jednym w semestrze).
Student w ciągu całego III roku ma zrealizować następującą liczbę przedmiotów fakultatywnych: 4 przedmioty fakultatywne w ramach specjalności (po dwa w semestrze) oraz dwa

Zobacz więcej

Przedmiot	Liczba godzin	Punkty ECTS	Forma weryfikacji	
Filozofia	30	3	egzamin	O
Ochrona własności intelektualnej	15	2	zaliczenie	O
Historia ładu międzynarodowego XX wieku	45	5	egzamin	O
Podstawy prawa konstytucyjnego	45	5	egzamin	O
Systemy polityczne	60	6	egzamin	O
Nauka o polityce	30	3	egzamin	O
Wprowadzenie do metodologii	15	2	zaliczenie	O
Język obcy	30	2	zaliczenie	O
WF	30	-	zaliczenie	O
Szkolenie BHK	4	-	zaliczenie	O
Szkolenie uniwersyteckie	4	-	zaliczenie	O
Grupa przedmiotów fakultatywnych kierunkowych				O

Рис 1.6 – Список навчальних дисциплін

Формування індивідуальних планів навчання здійснюється наступним чином [4,10,11]:

1. Консультація з академічним радником: Студенти можуть звернутися до академічного радника або консультанта для обговорення своїх навчальних цілей та інтересів. Під час цих консультацій обговорюється вибір дисциплін, які відповідають планам студента.

2. Вибір предметів: Студенти вибирають дисципліни для включення в свій індивідуальний навчальний план на підставі своїх цілей і потреб. Вони можуть обирати з різних курсів та дисциплін, які пропонуються університетом.

3. Складання навчального плану: На основі обговорення і вибору дисциплін студенти складають свій індивідуальний навчальний план. Цей план включає в себе вибрані курси, семестри, в які їх буде вивчено, і години, які будуть призначені для кожної дисципліни.

4. Реєстрація: Після складання індивідуального навчального плану студенти реєструються на вибрані курси та семестри відповідно до розкладу реєстрації, встановленого університетом.

5. Відстеження прогресу: Під час навчання студенти можуть відстежувати свій прогрес у виконанні індивідуального навчального плану. Якщо виникають питання або потрібна додаткова допомога, вони можуть звертатися до академічного радника.

6. Коригування плану: Якщо студент виявляє, що його цілі чи інтереси змінилися, він може внести коригування до свого індивідуального навчального плану в співпраці з академічним радником.

Деталі та процедури можуть відрізнятися від факультету до факультету та від спеціальності до спеціальності.

Недоліки системи формування індивідуальних планів у Яґеллонському університеті є наступні:

1. Недостатня інтеграція з іншими системами: Поточна система не ефективно взаємодіє з іншими елементами університетської інфраструктури (наприклад, бібліотечною або формуванням силабусів), це може вплинути на повноту та точність наданої інформації.

2. Недостатня персоналізація: Існуюча система може бути обмеженою в здатності надавати індивідуальні навчальні плани для студентів з різними потребами та інтересами.

3. Відсутність механізмів забезпечення конфіденційності: Система недостатньо захищає конфіденційні дані викладачів у процесі формування індивідуальних планів, це може порушувати приватність і призводити до можливих негативних наслідків.

UNIWERSYTET JAGIELLOŃSKI W KRAKOWIE

Filozofia

Karta opisu przedmiotu

Informacje podstawowe

Kierunek studiów politologia	**Cykl kształcenia** 2023/24
Ścieżka -	**Kod przedmiotu** UJ.WSMPOLS.110.5cac67d9e452a.23
Jednostka organizacyjna Wydział Studiów Międzynarodowych i Politycznych	**Języki wykładowe** Polski
Poziom kształcenia pierwszego stopnia	**Przedmiot powiązany z badaniami naukowymi** Tak
Forma studiów studia stacjonarne	**Dyscypliny** Nauki o polityce i administracji, Filozofia
Profil studiów ogólnoakademicki	**Klasyfikacja ISCED** 0388 Interdyscyplinarne programy i kwalifikacje związane z naukami społecznymi, dziennikarstwem i informacjami
Obligatoryjność obowiązkowy	**Kod USOS** WSM.INP-PDL-11
Koordynator przedmiotu	[redacted]
Prowadzący zajęcia	[redacted]

Okres	**Forma weryfikacji uzyskanych efektów uczenia się**	**Liczba punktów ECTS**
Semestr 1	egzamin **Sposób realizacji i godziny zajęć** wykład: 30	3.0

Cele kształcenia dla przedmiotu

C1	Zapoznanie studentów z podstawowymi pojęciami ontologii, epistemologii i etyki oraz z głównymi koncepcjami przedstawicieli najważniejszych kierunków zachodniej filozofii. Uświadomienie studentom związków między koncepcjami filozoficznymi a rozwiązaniami społecznymi i politycznymi.

Efekty uczenia się dla przedmiotu

Рис 1.7. – Детальна інформація про навчальну дисципліну

1.4.5 Огляд наукових робіт за темою магістерської дисертації

Аналіз існуючих методологій та підходів до розроблення та впровадження автоматизованих систем є важливим для аналізу перспектив розробки проекту в даному напрямку.

Робота [12] зосереджена на проблемі ефективного керування часом та ресурсами в магістратурі за допомогою автоматизованої системи формування розкладу. Автори розробили веб-додаток за допомогою HTML та CSS для розроблення автоматизованої системи. Головна сторінка середовища системи формування розкладу зображено на рис 1.8.

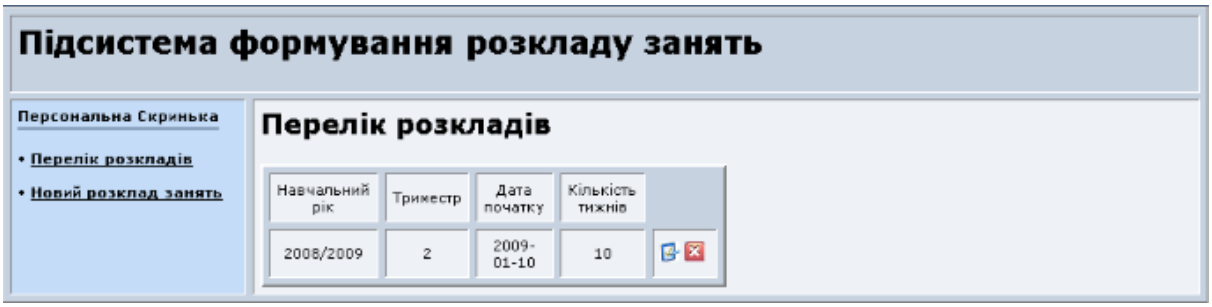


Рис 1.8 – Головна сторінка середовища

Слід зазначити переваги й недоліки розробленої системи. Серед переваг можна відзначити, що система може легко адаптуватися до змін в розкладі або умовах та автоматично перераховувати графік. Також ще одним плюсом є зменшення трудомісткості та помилок при формуванні розкладу, що полегшує адміністративні обов'язки. Серед недоліків можна відзначити, що система може бути незручною у використанні має неінтуїтивний інтерфейс, що впливає на користувацький комфорт та продуктивність, також система має потребу у постійних оновленнях та технічній підтримці для забезпечення актуальності [12].

Авторами в роботі [13] були розроблені моделі автоматизованого ведення документообігу вчених, методичних та виховних рад, які реалізовані в інтерактивному програмному комплексі. Автори використовують для розроблення загальної інтерактивної системи документообігу ІнМАД. Інтерфейс меню «Користувачі» системи ІнМАД зображено на рис 1.9.

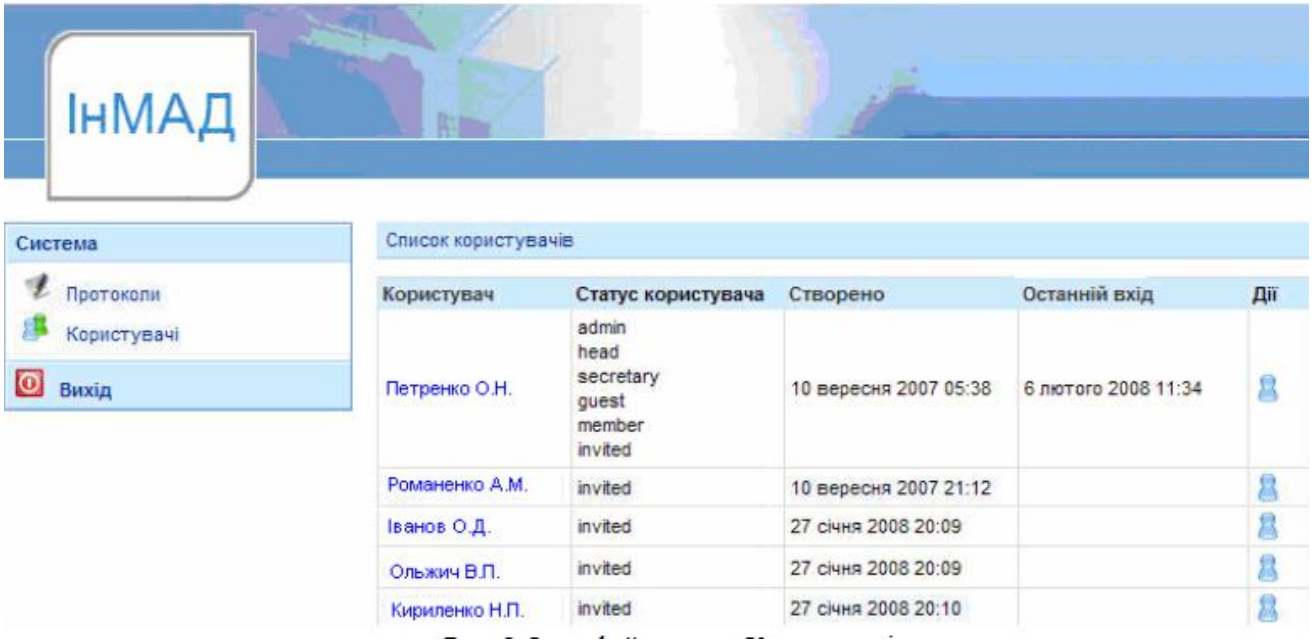


Рис 1.9 – Інтерфейс меню «Користувачі»

Серед переваг треба відзначити розширений функціонал, що включає в себе не лише базові операції обігу документів, але й різноманітні інструменти аналізу, звітності та спільної роботи. Інтуїтивно зрозумілий та зручний інтерфейс, що спрощує користування системою та підвищує продуктивність користувачів. Легка навігація, можливість персоналізації та інші елементи, спрямовані на комфорт користувача [13].

Система має наступні недоліки оскільки система розробляється з використанням системи ІнМАД, це може призвести до наступних проблем, а саме до ризику стороннього доступу, що надасть можливість неправомірного доступу сторонніх осіб до конфіденційної інформації, яка може призвести до ускладнення або порушення безпеки системи. Потреба у регулярних аудиторіях безпеки для визначення та вирішення можливих проблем безпеки [13].

В роботі [14] автори ознайомлюють з автоматизованою системою, яка може використовуватись у школах або вищих освітніх заклад на одному комп’ютери або на системі комп’ютерів, пов’язаних локальною мережею. Вікно з формою «Документи таблиці» наведено на рис 1.10.

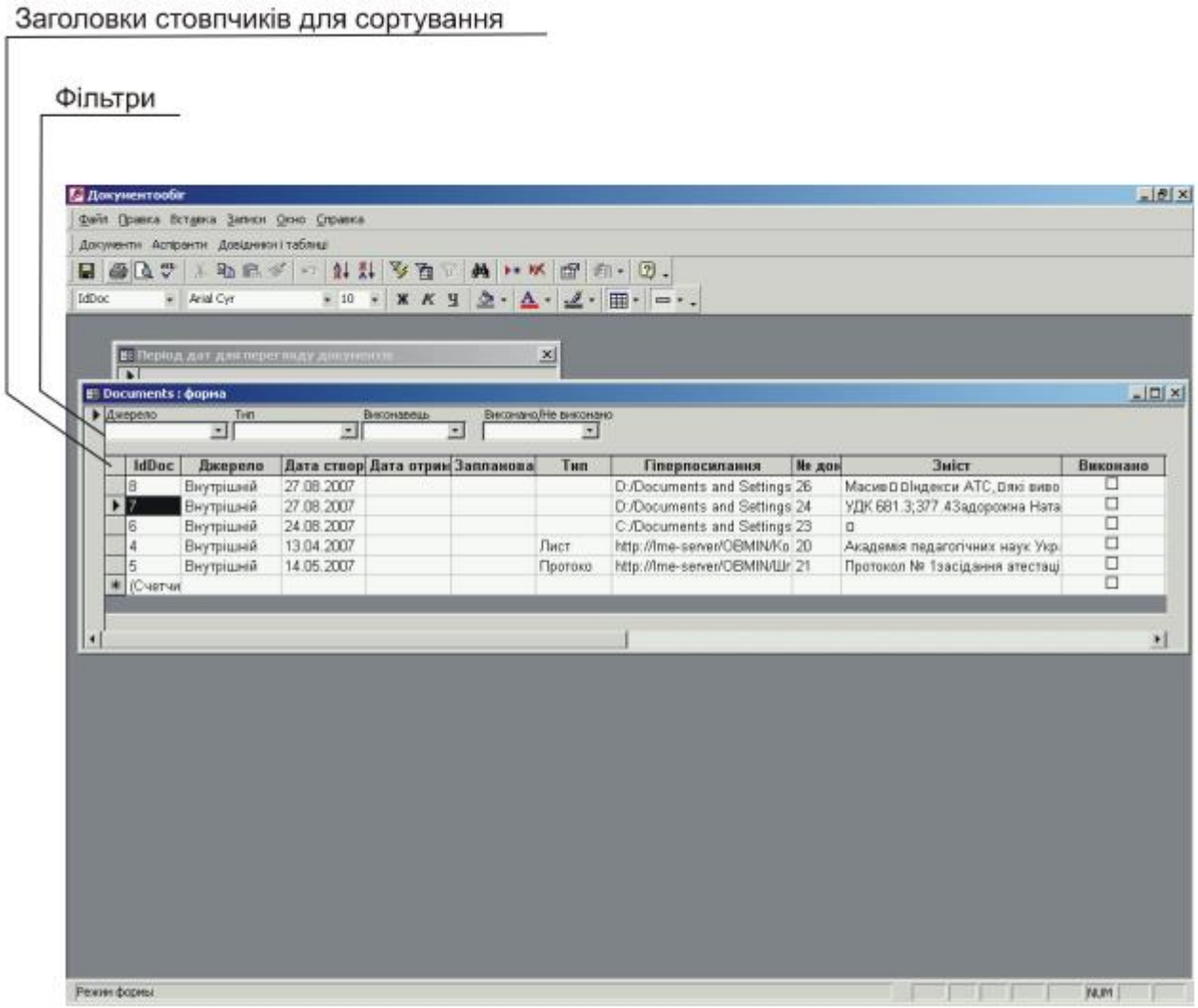


Рис 1.10 – Форма «Документи таблиці»

Серед переваг слід відзначити легкість використання, Microsoft Access може забезпечити простий та інтуїтивно зрозумілий інтерфейс для користувачів. Ще одною перевагою є локальний та мережевий запуск, можливість використання системи як на одному комп'ютері, так і на системі комп'ютерів, пов'язаних локальною мережею [14].

Треба зазначити, що як і у роботі [13], для розробки використовується стороннє програмне забезпечення (ПЗ), що може призвести до аналогічних проблем, та необхідність наявності Microsoft Access, також є недоліком, бо користувачам, які не мають доступу до Microsoft Access, не можуть взаємодіяти з системою, а використання Access можливо лише за умови підписки або покупки продукту. Також серед недоліків виділяється обмеженні можливості масштабування, Microsoft Access може стати обмеженням у разі потреби

масштабування системи, бо система підтримує лише використання до 15 комп'ютерів, також слід зазначити відсутність розширених функцій, у порівнянні з більш потужними системами керування базами даних, Microsoft Access може бути обмеженим щодо функціоналу та можливостей.

В роботі [15] автори проаналізували особливості функціонування локальної автоматизованої системи розподілу навантаження в процесі ведення документообігу та запропонували модель та метод реалізації автоматизованої системи.

Серед переваг слід відзначити ефективність системи, автоматизація дозволяє виконувати операції значно швидше, що сприяє прискоренню обробки документів та розподілу навантаження, зменшується ймовірність помилок та неправильного розподілу завдань через автоматичну обробку даних. Також система дозволяє спрямовувати зусилля персоналу на більш важливі завдання, а рутинні операції виконує автоматизована система.

У системі аналогічно до роботи [13] використовується ІнМАД, що призводить до таким самих недоліків, як у попередній роботі, а саме збільшується ризик кібератак та витоку конфіденційної інформації.

Авторами в роботі [16] розглянуто питання впровадження автоматизованої системи обліку на підприємстві. Були надані основні переваги такої системи, та наведені результати основні результати від використання системи.

Серед переваг системи з використанням «1С» автор визначає, розширення функціональності, зручний інтерфейс, нові інструменти аналізу, створення різноманітних аналітичних звітів, високу продуктивність та розмежування прав доступу.

Але треба зазначити й недоліки запропонованої системи. Ситуація аналогічна з роботами [13,15] використання стороннього ПЗ для впровадження автоматизованої системи, вимагає витрат на ліцензію для використання ПЗ, а для невеликих підприємств, вартість може бути доволі високою. Також для належної настройки та конфігурації системи часто потрібні спеціалісти з високим рівнем кваліфікації, через що «поріг входження» є доволі високим. Деякі користувачі можуть зіткнутися з обмеженнями в адаптації системи під вимоги свого бізнесу.

В роботі [17] автори аналізують проблему програмної підтримки автоматизації генерації електронних документів, розглядаються основні підходи до автоматизації генерації електронних документів в системах організаційного керування, пропонують метод математичних перетворень. Приклад шаблону вихідного документа наведений на рис 1.11.

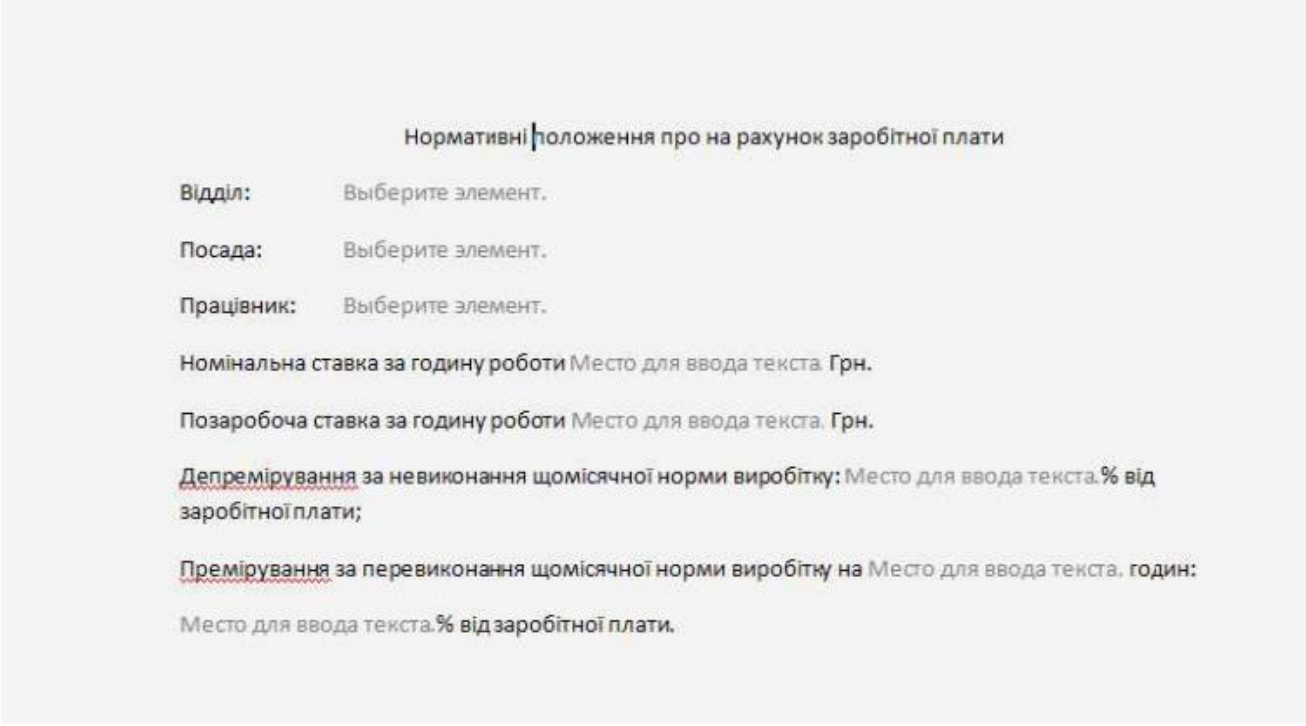


Рис 1.11 - Приклад шаблону вихідного документа

Серед переваг слід зазначити, що виконується повна розробка самостійного ПЗ, без використання сторонніх додатків. Також система надає можливість створення стандартних шаблонів документів, що забезпечує консистентність у вигляді та структурі. Автоматизація звільняє робочий час, який можна використовувати для інших завдань.

Серед недоліків, можна відзначити: складність використання; інтерфейс є менш інтуїтивним для користувачів, що може викликати труднощі у використанні; обмежена можливість змінювати і адаптувати інтерфейс до особистих потреб користувача. Ще в системі існує недостатня кількість інструментів для редагування, системи з недостатнім функціоналом можуть не забезпечувати можливості ефективної інтеграції, застарілі технології можуть призвести до труднощів у підтримці сучасних стандартів та технічних вимог.

В роботі [18] автори розробляють структуру автоматизованої системи формування індивідуальних планів магістрантів із забезпеченням вимог Болонського процесу. Запропоновані моделі реалізовано у середовищі автоматизованої системи ведення документообігу.

Проаналізувавши роботу [18], одразу необхідно відзначити недоліки, які аналогічні роботами [13,15], а саме використання стороннього ПЗ для реалізації автоматизованої системи, а саме системи ІнМАД, що може призвести до загрози витоку конференційних даних, також використання ІнМАД обмежує додавання нових функцій та інструментів, котрі потрібні користувачам, що призводить до обмеження в адаптації системи під унікальні вимоги. Також система має застарілий інтерфейс, що може бути складним для використання через відсутність сучасних елементів та інтуїтивних рішень

1.5 Мета та завдання досліджень

У сучасному освітньому середовищі ключову роль відіграє ефективна організація та управління освітнім процесом. З урахуванням стрімкого розвитку технологій, електронний документообіг стає необхідністю для забезпечення оперативності та зручності у веденні освітньої діяльності. Оглядаючи і аналізуючи існуючі автоматизовані системи, виявлено потребу у вдосконаленні процесу створення супровідної документації освітнього процесу. З цієї причини, розробка нової автоматизованої системи є необхідністю. Ця система буде спрямована на створення індивідуальних навчальних планів студентів, враховуючи проведений огляд та аналіз існуючих систем. Спроектована система вирішить такі проблеми, як гнучкість, інформативність та інтерактивність системи, безпека даних, недостатню інтеграцію з іншими системами.

Мета і завдання дослідження. Метою роботи є розроблення автоматизованої системи створення супровідної документації освітнього процесу, а саме індивідуальних навчальних планів студентів.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

1. Провести огляд сучасних рішень у сфері створення супровідної документації для навчального процесу.

2. Визначити вимоги до автоматизованої системи створення супровідної документації освітнього процесу.

3. Розробити початковий шаблон індивідуальних навчальних планів студентів.

4. Розробити основні підсистеми, які реалізовують бізнес-логіку.

5. Провести тестування автоматизованої системи на прикладі створення індивідуальних навчальних планів студентів

6. Розробити стартап проект.

РОЗДІЛ 2
ВИБІР ТЕХНОЛОГІЙ ДЛЯ РОЗРОБЛЕННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ

2.1 Вибір типу додатка

Завданням сучасного освітнього середовища є надання студентам швидкого та легкого доступу до інформації про навчальний процес. Звичайно, можуть існувати різні варіанти типів додатків, що виконуватимуть подібні функції.

Одним з варіантів реалізації є мобільний додаток. Проте, одним з недоліком цього варіанту є необхідність спеціального встановлення на смартфон, що веде до використання ресурсів пристрою та постійного перебування в пам'яті.

Найголовніший недолік мобільного додатка, пов'язаний з операційною системою (ОС), полягає в його залежності від конкретної ОС пристрою. Наприклад, якщо додаток спроектований для операційної системи Android, він може бути неповноцінно або навіть несумісно працювати на пристроях з iOS та навпаки. Це обмежує кількість потенційних користувачів, які можуть використовувати додаток, і вимагає подвійного внесення зусиль при розробці, оновленні та підтримці для кожної платформи окремо. Також, виникає проблема синхронізації та узгодженості функцій між різними ОС, що може призвести до відмінностей в роботі додатка на різних платформах [19].

Ще однією можливістю є створення веб-додатка, який має наступні переваги:

- доступність з будь-якого пристрою з Інтернет-підключенням засіб, що робить його універсальним і зручним для користувачів. Користувач та фахівці можуть отримати доступ до інформації, не обмежуючи себе конкретним пристроєм чи місцем перебування.
- простий та зрозумілий інтерфейс для користувачів будь-якого рівня навичок. Легкість навігації та інтуїтивно зрозумілі функції користувачам.
- веб-сайт легко масштабується та оновлюється без необхідності переінсталяції чи складних процедур встановлення для кінцевого користувача. Це надає можливість швидко реагувати на зміни в освітній програмі та інші вимоги.

- легка взаємодія з користувачами через елементи інтерфейсу, такі як, наприклад, модальні вікна, які відображають деталі про предмети навчального плану. Це створює зручний інструмент для отримання докладної інформації без зайвих зусиль.
- можливість легко інтегрувати різноманітні розширення та додатки, які розширюють функціональність системи. Це важливо для того, щоб забезпечити найбільш ефективне використання та адаптацію системи до змін у вимогах [20,21].

Суттєві недоліки у такого рішення практично відсутні, а переваги вказують на те, що створення веб-сайту є належним вибором для автоматизованої системи інформаційних документів. Веб-сайт має таки переваги, як зручність взаємодії, доступність та можливість розширення допомагають полегшити доступ до навчальної інформації та підтримки освітнього процесу.

Отже, враховуючи основний напрямок розроблення та обґрунтований вибір, дана робота націлена на створення веб-додатку. Архітектура розроблюваного веб-додатку ґрунтується на принципах «клієнт-сервер», оскільки при взаємодії користувача з елементами інтерфейсу, посилається запит на серверну частину (бек-енд), яка має знаходитися на комп’ютері-сервері, що має постійне підключення до мережі Інтернет.

Трьохрівнева архітектура є концепцією проектування програмного забезпечення, яка розділяє систему на три основних рівня функціональності. Ця архітектурна модель допомагає покращити розподіл обов'язків, забезпечити легку модифікацію та розвиток системи, а також сприяє підтримці високого рівня абстракції та незалежності між різними компонентами системи. Трьохрівнева архітектура зазвичай використовується в сферах розроблення програмного забезпечення, веб-додатків та баз даних [22].

1. Представлення (Presentation Layer) [23]:

Перший рівень відповідає за відображення інформації користувачам та обробку їхніх введених даних. Цей рівень включає інтерфейс користувача, який може бути графічним або текстовим. Завданням представлення є надання зручного та ефективного способу взаємодії користувача з системою.

2. Логіка додатку (Application Layer) [23]:

Другий рівень відповідає за обробку бізнес-логіки та виконання операцій, пов'язаних з обробкою даних. Тут знаходяться всі необхідні алгоритми, операції та правила бізнес-процесів. Цей рівень забезпечує взаємодію між рівнем представлення та рівнем доступу до даних.

3. Доступ до даних (Data Access Layer) [23]:

Третій рівень відповідає за зберігання та отримання даних. Це може включати в себе бази даних, файлові системи або будь-які інші механізми зберігання. Рівень доступу до даних керує запитами до джерел даних, забезпечуючи їхню ефективну обробку та передачу на рівень логіки додатку.

Представлення або інтерфейс користувача, є першим рівнем трьохрівневої архітектури, і воно відповідає за те, як інформація відображається та взаємодіє з користувачем системи. Основна мета цього рівня - забезпечити зручність та ефективність взаємодії користувача з програмою чи додатком. Давайте розглянемо основні аспекти цього рівня [22,23,24]:

1. Графічний інтерфейс (GUI):

В графічний інтерфейс входять елементи, такі як кнопки, текстові поля, меню, вікна та інші графічні компоненти, які користувач використовує для взаємодії з системою.

Графічний інтерфейс повинен бути не тільки функціональним, але й зручним та інтуїтивно зрозумілим для користувача.

2. Керування подіями:

Цей компонент відповідає за обробку подій, що виникають при взаємодії користувача з інтерфейсом (наприклад, натискання кнопок, введення тексту). Він ініціює виклики до рівня логіки додатку для відповідної обробки.

3. Валідація та форматування даних:

Цей аспект відповідає за перевірку введених користувачем даних на відповідність вимогам та форматування їх для подальшого використання в системі.

4. Керування станом інтерфейсу:

Система повинна відслідковувати стан інтерфейсу з метою відображення

поточного контексту або стану додатку. Наприклад, зміна стану кнопок або полів відповідно до виконаного користувачем дії.

5. Клієнт-серверна взаємодія:

Якщо додаток використовує клієнт-серверну архітектуру, то на рівні представлення здійснюється взаємодія з сервером, надсилання запитів та обробка отриманих відповідей.

6. Забезпечення безпеки:

Представлення включає механізми для захисту від несанкціонованого доступу та атак, таких як аутентифікація та авторизація.

Чітка відокремленість цього рівня дозволяє змінювати або модифікувати інтерфейс, не впливаючи на інші частини системи.

Логіка додатку – це другий рівень трьохрівневої архітектури, який відповідає за обробку бізнес-логіки, виконання операцій та взаємодію з даними. Розглянемо ключові аспекти цього рівня [22,25,26]:

1. Обробка бізнес-логіки:

Цей аспект включає в себе виконання операцій та взаємодію з даними відповідно до бізнес-правил та вимог.

2. Виконання операцій:

Логіка додатку містить реалізацію функціональності, яка визначає, як система повинна реагувати на конкретні дії користувачів чи інші події.

3. Бізнес-правила:

Визначення та реалізація бізнес-правил, які визначають логіку взаємодії та операції, які система виконує для досягнення бізнес-цілей.

4. Керування життєвим циклом об'єктів:

У випадку об'єктно-орієнтованої розробки, цей аспект включає в себе створення, модифікацію та видалення об'єктів відповідно до вимог бізнес-логіки.

5. Інтеграція з іншими системами:

Логіка додатку може взаємодіяти з іншими додатками чи сервісами, включаючи обмін даними та виклик методів інших систем.

6. Керування виключеннями та помилками:

Реалізація механізмів обробки виключень та помилок для забезпечення

надійності та стабільності роботи системи.

7. Спостереження за даними:

У випадку, коли система має великий обсяг даних, логіка додатку може включати в себе механізми для спостереження за даними та виконання аналізу для прийняття рішень.

8. Валідація та перевірка даних:

Логіка додатку відповідає за перевірку та валідацію даних, що надходять від користувачів або інших джерел, для забезпечення їхньої правильності та цілісності.

9. Безпека додатку:

Реалізація механізмів безпеки, таких як автентифікація, авторизація та захист від атак, для забезпечення конфіденційності та цілісності даних.

10. Керування транзакціями:

У випадку використання баз даних, логіка додатку може включати керування транзакціями для забезпечення атомарності та консистентності операцій.

11. Оптимізація продуктивності:

Розробка оптимізованих алгоритмів та структур даних для забезпечення швидкодії виконання операцій.

12. Керування сесіями:

Якщо система має концепцію сесій, то логіка додатку відповідає за керування сесіями користувачів.

Така архітектурна модель дозволяє легше розподілити функціональність системи, що сприяє зменшенню складності коду, полегшує роботу розробників та підвищує можливість змін у системі без необхідності редагування всіх її компонентів. Також вона сприяє відокремленню інтерфейсу користувача від бізнес-логіки та забезпечує підтримку множини інтерфейсів або пристроїв для взаємодії з однією і тією ж системою.

Доступ до даних є третім рівнем трьохрівневої архітектури, і його основна відповідальність полягає в керуванні доступом до даних та їх зберіганням. Цей рівень взаємодіє як із рівнем представлення, так із рівнем логіки додатку, надаючи

необхідний механізм для отримання та зберігання даних. Ключові аспекти цього рівня [22,24,25,26]:

1. З'єднання з джерелами даних:

Цей аспект включає в себе встановлення з'єднань з різними джерелами даних, такими як бази даних, файлові системи, веб-сервіси та інші.

2. Виконання операцій CRUD:

Доступ до даних надає інтерфейс для виконання операцій створення (Create), читання (Read), оновлення (Update) та видалення (Delete) даних у джерелі даних.

3. Об'єктно-реляційне відображення (ORM):

Використання ORM для спрощення взаємодії з реляційними базами даних, що дозволяє використовувати об'єктно-орієнтований підхід при роботі з даними.

4. Оптимізація запитів:

Розробка ефективних запитів до джерел даних та оптимізація їх виконання для забезпечення продуктивності системи.

5. Трансакційний контроль:

Забезпечення підтримки транзакцій для забезпечення атомарності та консистентності виконання операцій над даними.

6. Кешування даних:

Реалізація механізмів кешування для збереження часто використовуваних даних та підвищення швидкодії запитів.

7. Моделювання даних:

Створення моделей даних, які відображають структуру та взаємозв'язки даних, що забезпечує їхню правильну обробку та зберігання.

8. Індексація та пошук:

Використання індексації для швидкого доступу до даних та реалізація механізмів пошуку для ефективного виконання пошукових операцій.

9. Забезпечення цілісності даних:

Впровадження засобів для забезпечення цілісності даних, включаючи обмеження, валідацію та контроль доступу.

10. Резервне копіювання та відновлення:

Забезпечення можливості резервного копіювання даних та їх відновлення для запобігання втраті важливої інформації.

11. Шифрування даних:

Застосування механізмів шифрування для забезпечення конфіденційності та безпеки даних.

12. Підтримка множинних джерел даних:

Якщо система використовує різні джерела даних, наприклад, бази даних та зовнішні API (Application Programming Interface або прикладний програмний інтерфейс), розробка механізмів для їхньої взаємодії та синхронізації.

Доступ до даних відіграє ключову роль у забезпеченні ефективного та надійного керування даними. Він дозволяє іншим рівням взаємодіяти з джерелами даних без прямого зазначення деталей їхньої реалізації, що дозволяє легше внести зміни в систему та забезпечує високий рівень абстракції.

Для розроблення автоматизованої системи створення супровідних документів освітнього процесу було розроблено багаторівневу архітектуру, що складається із 4 шарів (рис 2.1):

1. Шар відображення.
2. Шар бізнес-логіки.
3. Шар доступу до даних.
4. Шар сховища даних.

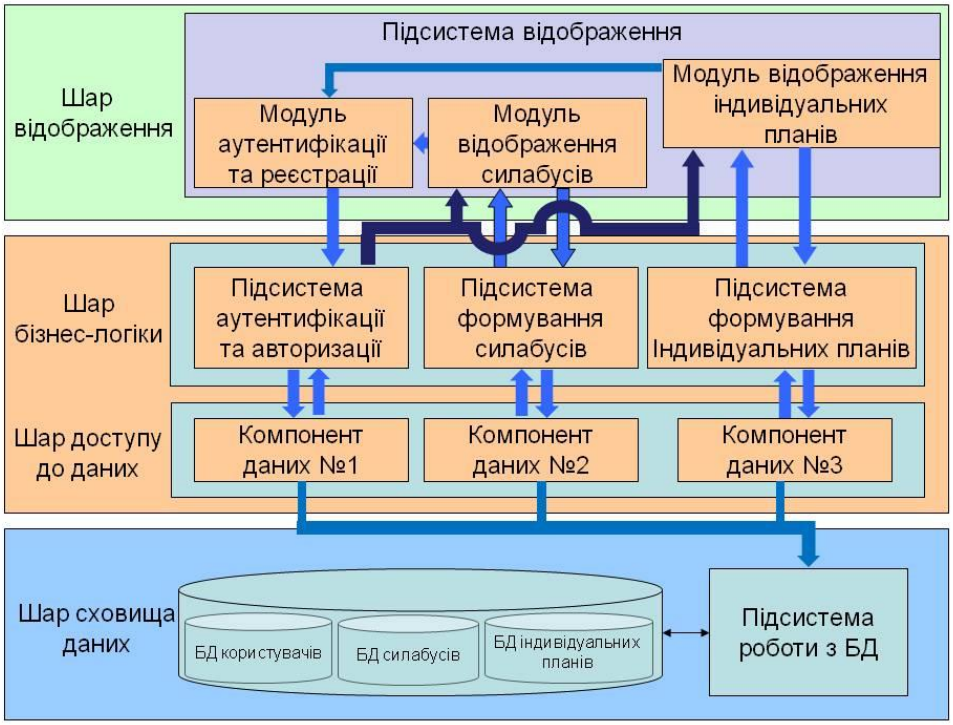


Рис.2.1 - Трьохрівнева архітектура автоматизованої системи створення супровідних документів освітнього процесу

Шар відображення складається з компонентів аутентифікації, реєстрації, роботи із силабусами та роботи з індивідуальними навчальними планами. Важливо відзначити, що компоненти, відповідальні за роботу з силабусами та індивідуальними планами, тісно пов'язані з компонентами аутентифікації та реєстрації, що визначає взаємодію між шарами відображення та бізнес-логіки.

Шар бізнес-логіки включає в себе підсистему аутентифікації та реєстрації, а також підсистеми формування силабусів і індивідуальних планів. У кожній підсистемі містяться методи, які відповідають за конкретну бізнес-логіку. Важливо відзначити, що кожна підсистема взаємопов'язана з відповідними компонентами з шару відображення, забезпечуючи цілісність системи.

Шар доступу до даних складається з компонентів даних для кожної підсистеми з шару бізнес-логіки. Кожен компонент даних взаємодіє з відповідною базою даних з шару сховища даних за допомогою підсистеми роботи із базою даних.

Шар сховища даних включає в себе базу даних користувачів, базу даних силабусів та базу даних індивідуальних планів. Робота із базою даних в цьому

шарі відбувається за допомогою підсистеми роботи із базою даних, яка взаємопов'язана з шаром доступу до даних.

Проаналізувавши вище зазначену інформацію була розроблена трьох-рівнева архітектура для розроблюваної системи створення супровідної документації освітнього процесу, а саме індивідуальних навчальних планів студентів (рис 2.2).

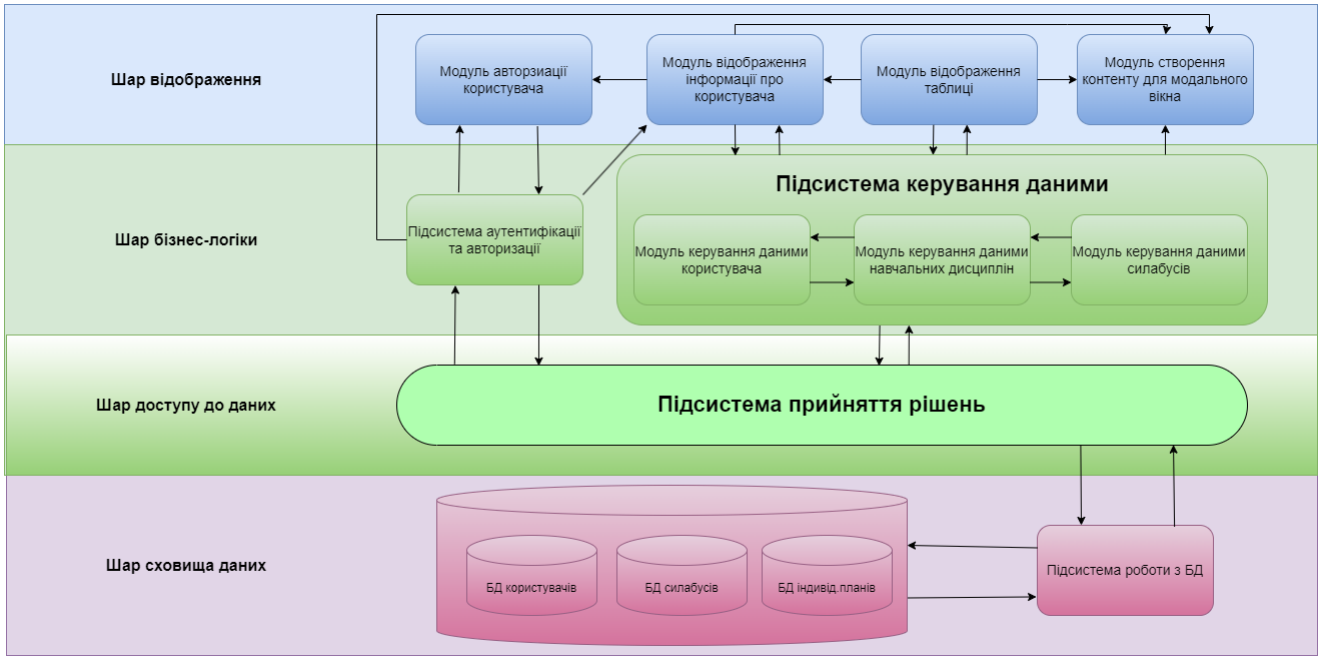


Рис 2.2 - Розроблена трьохрівнева архітектура

Модуль авторизації користувача відповідає за процес авторизації користувачів у системі. Містить форму входу, обробники для перевірки ідентифікаційних даних користувача та логіку керування сесією.

Модуль відображення інформації про користувача відповідає за відображення основної інформації про авторизованого користувача. Включає ім’я, прізвище, профіль користувача, статус тощо.

Модуль відображення таблиці відповідає за відображенням табличних даних. Пов'язаний із конкретними операціями, такими як сортування, фільтрація, або додавання нових записів.

Модуль створення контенту для модального вікна відповідає за генерацію або підготовку контенту, який буде відображатися в модальному вікні. Містить інтерфейс для створення даних, які будуть відображатися в модальному вікні.

Підсистема аутентифікації та авторизації забезпечує безпеку системи шляхом ідентифікації та авторизації користувачів. Включає механізми перевірки прав доступу, керування профілями користувачів та інші засоби для забезпечення конфіденційності та безпеки даних.

Модуль керування даними користувача забезпечує функціонал для додавання, редагування та видалення даних про користувачів. Включає операції зі створення та керування профілями користувачів, а також керування їхніми особистими налаштуваннями.

Модуль керування даними навчальних дисциплін відповідає за керування інформацією про різні навчальні дисципліни. Містить функції для додавання нових дисциплін, редагування їх властивостей та видалення.

Модуль керування даними силабусів відповідає за керування інформацією про силабуси (плани навчання) для різних навчальних дисциплін.

Третій рівень архітектури включає в себе підсистему прийняття рішень, яка відповідає за обробку та аналіз даних для прийняття важливих рішень. Ця підсистема може складатися з ряду модулів та функціональних блоків.

Підсистема роботи з базами даних (БД) забезпечує інтерфейс між програмним забезпеченням та самою базою даних, включає в себе функції для взаємодії з даними, такі як додавання, зчитування, редагування та видалення записів

В базі даних ці запити обробляються, і відповідь пересилається на клієнтську сторону для відображення користувачеві. Ліміт взаємодії між сервером і клієнтами (користувачами веб-додатку) обмежується лише апаратною продуктивністю сервера і може включати тисячі користувачів одночасно.

Серверна частина в архітектурі «клієнт-сервер» відповідає за обробку та керування ресурсами, надання сервісів, взаємодію з базою даних, надання необхідної функціональності клієнтській частині.

Основні завдання та відповідальності сервера [27,28]:

1. Обробка запитів та відправлення відповідей:

- Сервер отримує запити від клієнта та відправляє відповіді, забезпечуючи взаємодію між клієнтом та сервером.

- Інструменти: HTTP-протокол, різні методи обробки запитів (GET, POST, PUT, DELETE) для взаємодії з веб-додатками.

2. Логіка бізнес-процесів:

- Сервер виконує логіку бізнес-процесів, обробляє дані та виконує різноманітні операції, необхідні для функціонування системи.

- Інструменти: Мови програмування, фреймворки для реалізації бізнес-логіки.

3. Керування даними:

- Сервер взаємодіє з базою даних, проводить операції зберігання, оновлення, видалення та читання даних.

- Інструменти: SQL або NoSQL бази даних, мови запитів (наприклад, SQL), ORM (Object-Relational Mapping) для мапінгу об'єктів на дані бази даних.

4. Аутентифікація та авторизація:

- Забезпечення безпеки шляхом перевірки ідентифікації користувача (аутентифікація) та визначення його прав доступу (авторизація).

- Інструменти: Механізми аутентифікації (логін/пароль, токени), системи контролю доступу.

5. Робота з мережевими протоколами:

- Сервер використовує різні мережеві протоколи для обміну даними з клієнтами (наприклад, HTTP, WebSocket).

- Інструменти: Різні бібліотеки та фреймворки для роботи з протоколами.

6. Обробка помилок та створення лог-файлів:

- Сервер обробляє помилки та здійснює створення лог-файлів під час подій для відстеження та аналізу проблем.

- Інструменти: Системи процес створення лог-файлів—обробники винятків.

Характеристики серверної частини:

- Забезпечення ефективної обробки великої кількості запитів та оптимізація роботи сервера.

- Розробка сервера так, щоб він був готовий до масштабування у випадку зростання обсягів обробки даних та запитів.
- Захист від різноманітних атак та забезпечення конфіденційності даних.
- Забезпечення стабільності роботи сервера та доступності сервісів у різних сценаріях.

Серверна частина в архітектурі «клієнт-сервер» є центральним елементом, який забезпечує обробку запитів, логіку бізнес-процесів та керування даними. Її ефективність та надійність визначають стабільність та функціональність всієї системи.

В архітектурі «клієнт-сервер», клієнтська частина відповідає за взаємодію з користувачем та відображення інформації на його пристрої. Клієнтська частина може бути представлена в різних формах, таких як веб-браузер, мобільний додаток або десктоп-клієнт.

Основні завдання та відповідальності клієнта:

1. Інтерфейс користувача (UI):
- Клієнт відповідає за відображення інтерфейсу користувача, який включає в себе всі елементи та функції, що взаємодіють з користувачем.
- Інструменти: HTML, CSS, JavaScript для веб-браузерів; різні фреймворки та мови для мобільних та десктоп додатків.

Інтерфейс користувача (UI) складається з різноманітних елементів, які взаємодіють з користувачем і дозволяють йому взаємодіяти з програмами або системами. Основні елементи UI включають:

1) Вікна (Windows): області на екрані, які містять інші елементи інтерфейсу. Вони можуть містити текст, зображення, кнопки і інші візуальні компоненти.

2) Меню (Menus): візуальні списки опцій або команд, які користувач може вибирати для виклику певної дії чи навігації.

3) Панелі інструментів (Toolbars): рядки або блоки кнопок і ікон, які представляють швидкий доступ до часто використовуваних команд чи функцій.

4) Кнопки (Buttons): елементи, які користувач може натискати для виклику певної дії чи команди.

5) Текстові поля (Text Fields): області для введення тексту, такі як поля для введення імені користувача, пароля або інших даних.

6) Чекбокси та радіокнопки (Checkboxes and Radio Buttons): елементи для вибору опцій чи варіантів з можливістю вибору одного чи більше варіантів.

7) Віджети (Widgets): різні елементи інтерфейсу, такі як ползунки, випадаючі списки, календарі, які дозволяють користувачеві вибирати значення чи налаштувати параметри.

8) Модальні вікна (Dialog Boxes): використовуються для виведення важливої інформації, збору даних або виклику користувачевих дій, які вимагають негайної уваги

9) Вказівники (Pointers): Зображення на екрані, яке вказує на місце, де знаходиться вказівник миші чи іншого пристрою введення.

2. Логіка клієнта:

- Клієнт може виконувати певну логіку на стороні клієнта для покращення відзивчивості та ефективності, таку як валідація введених даних, обробка подій тощо.

- Інструменти: JavaScript, фреймворки для розроблення веб-додатків, мови програмування для десктоп-та мобільних платформ.

3. Зберігання даних клієнта:

- Клієнт може зберігати та керувати даними локально, наприклад, кешувати деякі дані для швидшого доступу або для роботи в режимі офлайн.

- Інструменти: LocalStorage, IndexedDB для веб-додатків; локальні бази даних для мобільних та десктоп-додатків.

4. Взаємодія з сервером:

- Клієнтська частина забезпечує взаємодію з сервером шляхом відправлення запитів та обробки отриманих відповідей. Це може бути здійснено за допомогою HTTP-запитів, WebSocket тощо.

- Інструменти: AJAX, Fetch API для веб-додатків; різні бібліотеки та SDK для мобільних та десктоп-додатків.

5. Безпека:

- Забезпечує безпеку та конфіденційність даних, що обробляються та передаються клієнту. Включає в себе валідацію даних, обробку аутентифікації та авторизації.

- Інструменти: HTTPS для забезпечення захищеної передачі даних; механізми аутентифікації, токени безпеки тощо.

Характеристики клієнтської частини:

- Забезпечення інтуїтивно зрозумілого та ефективного інтерфейсу для користувача.

- Забезпечення оптимізації та ефективності в роботі.

- Адаптація до різних пристроїв та екранів, зокрема мобільних пристроїв

- Забезпечення стабільності та надійності роботи в різних умовах.

Клієнтська частина в архітектурі «клієнт-сервер» є ключовим елементом, який взаємодіє з користувачем та відображає інформацію. Її функціонал включає створення інтерфейсу користувача, обробку подій, взаємодію з сервером та зберігання даних. Успішна реалізація клієнтської частини визначає зручність взаємодії.

Таким чином, трьохрівнева архітектура, яка містить модуль авторизації користувача, модуль відображення інформації про користувача, модуль відображення таблиці, модуль створення контенту для модального вікна, підсистему аутентифікації та авторизації, модуль керування даними користувача, модуль керування даними навчальних дисциплін, модуль керування даними силабусів, підсистему прийняття рішень та базу даних, проектованого програмного продукту є визначеною, далі необхідно обрати засоби розробки та подальшої програмної реалізації.

2.2 Вибір мови програмування

Для програмування веб-додатків можна використовувати багато мов програмування. Аналіз публікацій [29,30,31,32,33,34,35,36,37,38,39,40] з даного напрямку свідчить про те, що часто для даних цілей використовуються мови

програмування, які наведено на рис 2.3.

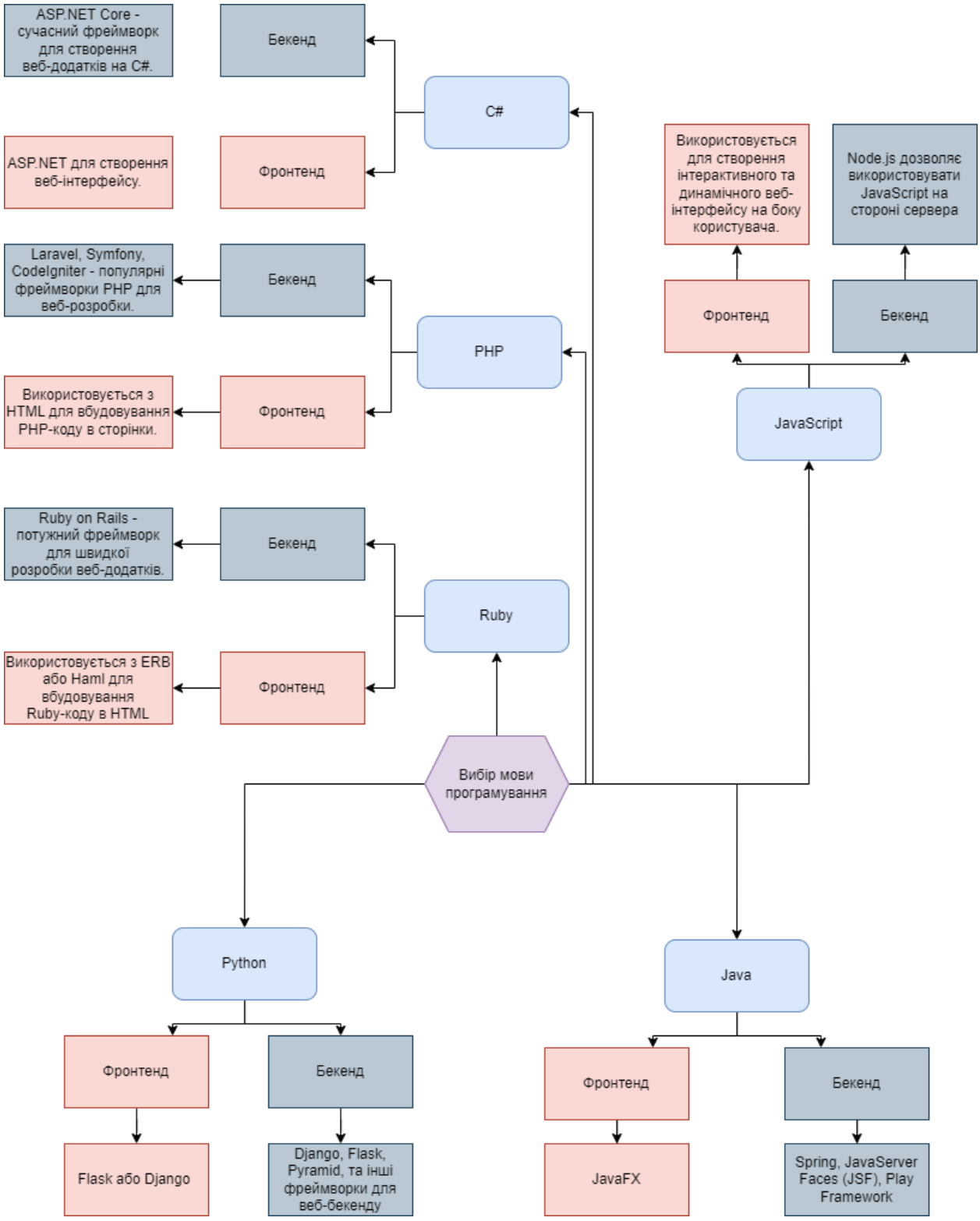


Рис 2.3 – Види мов програмування

Python – мова програмування, яка завдяки своїй простоті та гнучкості стала популярною серед розробників. Динамічною типізацією є ключовою особливістю мови програмування Python. Це означає, що типи змінних визначаються не на

етапі компіляції, а на етапі виконання програми. Це робить мову дуже гнучкою та легкою для використання, але може призвести до неочікуваних помилок у великих проектах. Є можливість визначити змінну та привласнити їй значення без явного вказання типу. Типи змінних можуть змінюватися під час виконання програми. Це може призвести до несподіваних результатів, якщо не враховувати типи даних у великих програмах. Динамічна типізація також означає, що є можливість використовувати одну змінну для різних типів даних у різних частинах програми. Навіть з введенням анотацій типів (PEP 484), інтерпретатор не виконує строгу перевірку типів. Це може спричинити помилки під час виконання, що можна було б уникнути за більш строгих умов [29].

Python є об’єктно-орієнтованою мовою, де всі дані та функції розглядаються як об’єкти. Це полегшує організацію коду та взаємодію з об’єктами. Наслідування, інкапсуляція та поліморфізм – основні концепції об'єктно-орієнтоване програмування (ООП), які реалізовані.

Хоча Python потужна мова, але вибір для розроблення веб-додатків повинен враховувати конкретні потреби проекту та експертизу розробників, не є найкращим вибором для розроблення веб-додатків через свої сильні та слабкі сторони.

Java є мовою програмування, яка завоювала визнання завдяки своїй універсальності та ефективності у великих та складних системах. Java відома своєю строгою системою контролю типів, що дозволяє визначати, які дані можна зберігати в конкретних змінних та об'єктах. Всі типи даних в Java визначаються перед використанням, надаючи компілятору можливість виявити помилки ще до виконання програми. Це забезпечує безпеку та надійність програм, зменшуючи ймовірність помилок, пов'язаних з типами даних. Компілятор визначає, чи вірно використані дані в програмі та виводить помилку, якщо знаходить несумісність типів.

Об'єктно-орієнтоване програмування в Java базується на концепції «об'єктів» - екземплярів класів, які описують дані та методи, що їх обробляють. Все в Java є об'єктом, від примітивних типів даних до складних структур. Цей підхід дозволяє розділити складність програм на менші, самодостатні модулі -

об'єкти, які можуть взаємодіяти між собою. Кожен об'єкт відповідає за свої власні дані та поведінку, і взаємодіє лише через визначені інтерфейси. Також слід зазначити у Java, взаємодія з об'єктами відбувається через виклик методів та передачу параметрів. Методи визначають поведінку об'єкта, а параметри - дані, які можуть впливати на цю поведінку. Сценарії взаємодії визначають, як об'єкти взаємодіють між собою для досягнення певних цілей. Це може бути обмін інформацією, передача контролю від одного об'єкта до іншого, або будь-який інший спосіб спільної дії [30].

Загалом, об'єктно-орієнтований підхід у Java спрощує розроблення та підтримку коду, дозволяючи розділити програму на логічні компоненти та визначати взаємодію між ними. Але ця мова має свої недоліки, в порівнянні з іншими мовами, наприклад, Python або JavaScript, Java вимагає більше коду для досягнення того ж самого результату, через процес розроблення на Java може бути менш гнучким та вимагати більше часу порівняно з іншими мовами, особливо для невеликих проектів [31].

Ruby – це мова програмування, яка отримала популярність завдяки своєму простому та читабельному синтаксису, а також фреймворку Ruby on Rails, який робить розроблення веб-додатків ефективною та приємною. Ця мова майже виключно використовується з метою веб-розроблення. Ruby, як й Python володіє динамічною типізацією, що дозволяє змінювати типи змінних під час виконання програми. Це надає розробникам велику гнучкість, але також може стати причиною помилок під час виконання. Через що має схожі недоліки з Python. Мова є повністю об'єктно-орієнтованою, де навіть базові типи даних представлені об'єктами.

Веб-додатки на Ruby зазвичай будуються за допомогою фреймворка Ruby on Rails - це високорівневий веб-фреймворк, заснований на мові програмування Ruby. Завдяки своїй конвенції перед конфігурацією та активній спільноті розробників, Rails став одним з найпопулярніших інструментів для швидкої та ефективної розроблення веб-додатків. Контролери у Rails відповідають за обробку HTTP-запитів та виклик методів моделей для збереження чи отримання даних з бази даних. Взаємодія з об'єктами відбувається через ActiveRecord, який

надає зручний спосіб роботи з базою даних. Rails використовує архітектурний патерн Модель-Вид-Контролер (MVC). Це розділяє додаток на три основні компоненти, що полегшує розроблення та підтримку. Також Rails має вбудовані засоби захисту від атак, такі як CSRF (Cross-Site Request Forgery) та SQL Injection захист [32,33].

Незваючи на те, що Ruby є гарним вибором коли необхідно розробити веб-додаток, вона має свої недоліки. Ruby відомий своїм споживанням пам'яті, особливо порівняно з деякими іншими мовами програмування. Це може обмежити її застосовність у сферах, де ефективне керування ресурсами пам'яті є критично важливим, наприклад, у розробленні вбудованих систем з обмеженими ресурсами або у високонавантажених додатках, де оптимізація споживання пам'яті має першорядне значення. Також, як й Python, Ruby має динамічну типізацію, а це може спричинити помилки під час виконання, що можна було б уникнути за більш строгих умов [33].

PHP - це серверна мова програмування, яка широко використовується для розроблення веб-додатків. Враховуючи свою простоту та розповсюдженість, PHP став однією з найпопулярніших мов для веб-розроблення. Вона визначається своєю простотою, розповсюдженістю та високою швидкодією.

PHP має динамічну типізацію, що надає гнучкість у роботі з даними. Його синтаксис схожий на мову програмування «C», що полегшує адаптацію для багатьох розробників, але як вже було розглянуто раніше має доволі суттєвий недолік, але PHP має механізм обробки помилок та винятків, що дозволяє гнучко реагувати на непередбачені ситуації. PHP підтримує об'єктно-орієнтований підхід до програмування, проте він ґрунтується на прототипах, а не класах, що робить його унікальним у своєму роді. Мова має чотири типи об'єктів: вбудовані, об'єкти браузера, об'єкти документа і об'єкти користувача.

В PHP введення-виведення в основному обмежене взаємодією з документами та користувачами. Доступ до локальної файлової системи за замовчуванням може бути недозволеним, але браузери можуть надавати об'єкти для роботи з файловою системою користувача. PHP виконується на стороні сервера і обробляє запити від клієнтів. Для взаємодії із зовнішніми API та

інтеграції з різними сервісами, PHP може використовувати HTTP-запити. Сценарії та сервери в PHP виконується на стороні сервера, тому він ініціюється та обробляє запити від клієнтів, наприклад, через веб-сервер Apache чи Nginx [34,35].

Незважаючи на те, що PHP має багатий інструментарій, якісні фреймворки Laravel та Symfony, які надають структуру та інструменти для швидкого та ефективного розроблення, треба відмити й його недоліки. PHP є інтерпретованою мовою, що може призводити до менш ефективної роботи порівняно з компільованими мовами, такими як Java або C#. Це особливо важливо для великих та обчислювально інтенсивних додатках. Також у певних випадках оновлення PHP може призводити до змін у API, що може вплинути на існуючий код. Це особливо важливо при роботі зі старішими версіями PHP або при плануванні майбутніх оновлень. Незважаючи на багато покращень у нових версіях, PHP залишається менш продуктивним у порівнянні з деякими іншими мовами, особливо в великих та високозавантажених додатках. Необхідно зазначити, що PHP має історію проблем з безпекою. Незахищений код може призвести до вразливостей, таких як SQL-ін'єкції чи вразливості у сесіях, і вимагати додаткових зусиль для забезпечення надійності додатка [36].

Мова C# є високопродуктивною та об'єктно-орієнтованою мовою програмування, часто використовуваною для створення веб-додатків за допомогою технології ASP.NET.

C# має жорсткий (статичний) контроль типів, що дозволяє виявити помилки під час компіляції. Це підвищує стабільність та ефективність коду, а також полегшує його обслуговування. Мова є повністю об'єктно-орієнтованою, це сприяє модульності та розширюваності програм. Об'єктно-орієнтований підхід полегшує роботу з кодом, його розуміння та модифікацію. У мові C# взаємодія з об'єктами ґрунтується на концепції класів та об'єктів. Класи визначають властивості та методи, а об'єкти є їх екземплярами. Робота з об'єктами включає їх створення, зміну та використання в коді. Мова дозволяє створювати сценарії та обробляти події в середовищі ASP.NET. Сценарії використовуються для виконання конкретних завдань, а обробка подій відслідковує та реагує на користувацькі дії. Через особливості треба відзначити інструмент LINQ в C# є

потужним механізмом для маніпулювання та обробки даних. Цей інструмент дозволяє виконувати операції фільтрації, сортування та об'єднання даних безпосередньо в мові програмування [37,38,39].

ASP.NET — це високопродуктивний фреймворк для створення веб-додатків. Він надає широкий набір інструментів для швидкого розроблення та ефективного розгортання веб-рішень. ASP.NET використовує модель подій та code-behind для розділення логіки від представлення. Code-behind дозволяє відокремити серверний код від коду HTML, що полегшує розроблення та обслуговування великих веб-проектів. Дуже зручно, що фреймворк надає вбудовані засоби відстеження стану, такі як ViewState, що полегшує збереження стану сторінки між запитами. Механізми керування сесіями спрощують збереження та обробку інформації про користувача. ASP.NET є розширюваним та підтримує ряд сучасних технологій, таких як Web API для створення веб-служб, SignalR для реального часу, та вбудовані можливості для роботи з базами даних [38].

Також треба відзначити недоліки мови C#. Мова тісно пов'язана з екосистемою Microsoft. Це може обмежувати переносимість коду між різними платформами, оскільки .NET Framework і .NET Core підтримуються переважно на платформах Microsoft. Закритість Entity Framework, хоча й Entity Framework дозволяє легко працювати з базами даних, він є в основному прив'язаним до платформи Microsoft SQL Server, що може обмежити вибір для розробників. В C# використовується система автоматичного збору сміття, що може впливати на продуктивність у випадках, коли важливо максимально ефективно керувати використанням ресурсів [37].

JavaScript - це високорівнева мова програмування, яка використовується для розроблення веб-додатків. Вона була створена компанією Netscape і у 1995 році. JavaScript є однією з ключових технологій для реалізації інтерактивності на веб-сайтах і дозволяє динамічно змінювати зміст сторінок [40].

JavaScript базується на об'єктно-орієнтованому програмуванні, де дані та функції групуються в об'єкти. Мова JavaScript є об’єктно-орієнтована, проте вона заснована на прототипах, а не на класах. Є чотири типи об’єктів: вбудовані,

об’єкти браузера, об’єкти документа і об’єкти користувача (програміста), JavaScript застосовується не тільки на клієнтській стороні браузера, але і на сервері, використовуючи популярні середовища виконання, такі як Node.js.

До даних застосовується слабкий (динамічний) контроль типів. В операторах з різнотипними даними останні автоматично приводяться до необхідного типу. Типи даних можуть бути примітивними і складеними. Примітивні типи містять прості однорідні значення, такі дані можна передавати функціям як параметри за значенням, а не за посиланням. Складені типи містять різнорідні дані (в тому числі і складені), їх передають у функції тільки за посиланням [40].

Важливою особливістю мови JavaScript є те, що її вихідні програмні коди інтерпретуються, це досить гнучке, але повільне рішення. Мова чутлива до регістру, що часто випускають з виду початківці, ймовірно тому, що HTML, застосовуваний зазвичай з JavaScript спільно, не залежить від регістру (імена тегів і атрибутів HTML можна писати як малими, так і великими літерами) [20].

Мова дозволяє реагувати на події користувача, такі як кліки мишею або введення з клавіатури, що робить її ідеальною для створення інтерактивних веб-сайтів. JavaScript базується на тому, що скрипт може реагувати на події, які відбуваються на веб-сторінці, такі як клік мишею, натискання клавіш, завантаження сторінки, зміна значення поля введення та багато інших. Це дозволяє розробникам створювати інтерактивні та динамічні веб-сайти, які реагують на взаємодію користувача [19].

JavaScript може динамічно змінювати вміст сторінок, маніпулювати DOM (Document Object Model) та змінювати стилі за допомогою CSS. Мова має динамічну типізацію, змінні в JavaScript можуть містити значення різних типів, і їх тип може змінюватися під час виконання програми. Функції в JavaScript є об'єктами першого класу, і мова підтримує функціональні конструкції, такі як замикання та функції вищого порядку [40].

Мова призначена для використання в різних браузерах, що робить її універсальною для розроблення веб-додатків. JavaScript підтримує асинхронний код, що дозволяє взаємодіяти з сервером без перезавантаження сторінки, що

поліпшує швидкість та ефективність веб-додатків [40].

JavaScript має широку екосистему бібліотек та фреймворків, таких як React, Angular, Vue.js, які полегшують розробку великих та складних веб-додатків.

Введення-виведення в основному обмежене взаємодією з документами і користувачами. За замовчуванням очікується, що доступ до локальної файлової системи недозволений. Однак браузери можуть надавати спеціальні об’єкти, за допомогою яких забезпечується робота з файловою системою користувача, хоча і з видачею попереджень про небезпеку здійснення файлових операцій [41].

Node.js — це серверна технологія, яка базується на JavaScript та дозволяє виконувати код на стороні сервера. Вона побудована на движку V8 від Google, який також використовується в браузері Google Chrome [41].

Одна з особливостей Node.js — це використання асинхронної моделі обробки подій, що дозволяє обробляти багато запитів одночасно без блокування виконання коду.

Node.js побудована так, щоб бути кросплатформенною. Вона може працювати на різних операційних системах, таких як Windows, macOS, Linux тощо. Технологія підтримує модульність за допомогою системи модулів CommonJS. Пакетний менеджер npm (Node Package Manager) дозволяє легко встановлювати та керувати залежностями проекту [41].

Node.js підтримує роботу з різними базами даних, надаючи розробникам можливість здійснювати взаємодію з базами даних за допомогою драйверів та ORM (Object-Relational Mapping) бібліотек.

Node.js може виступати в ролі веб-сервера. За допомогою фреймворків, таких як Express, можна швидко створювати веб-сервери та API для обробки запитів [41].

Node.js — це потужний інструмент у веб-розробці, який надає можливості для створення ефективних серверних додатків[41].

Серед недоліків можна виділити те, що деякі особливості JavaScript можуть призводити до неочікуваної поведінки, наприклад, в концепції "hoisting" або в некоректної роботи з типами даних. З погляду безпеки, важливо пам'ятати, що JavaScript виконується на стороні клієнта, і неправильно написаний код може

призводити до проблем безпеки. Виконання коду на різних пристроях може призводити до різниці в швидкості, що важко передбачити.

Проаналізувавши мову JavaScript зваживши всі переваги й недоліки мови, недоліки виявились несуттєвими, а переваги які надає мова, стали важливим чинником, чому саме її було обрано для створення проекту.

JavaScript є ключовим компонентом веб-розроблення, відповідаючи за динаміку та взаємодію на стороні клієнта. Його зручний синтаксис та багатий функціонал, призначеня для використання в різних браузерах, можливість реагувати на події користувача,, що робить його найкращим вибором при розробленні інтерактивного веб-додатку.

2.3. Вибір фреймворку

JavaScript-фреймворки — це інструменти, що спрощують та систематизують розроблення великих, складних веб-додатків. Вони надають готовий фундамент для створення і взаємодії з елементами інтерфейсу, вирішують типові завдання та забезпечують кращу організацію коду.

Для розроблення веб-додатку найчастіше використовуються наступні фреймворки для JavaScript, це React, Angular та Vue.js.

React - це бібліотека JavaScript, розроблена Facebook, призначена для створення інтерфейсів користувача, особливо для односторінкових додатків, де динамічність та ефективність важливі. З React можна побудувати компоненти UI, які реагують на зміни стану, що робить процес розробки ефективнішим та більш простим [42].

React базується на концепції компонентів. Компонентний підхід полягає в розбитті інтерфейсу користувача на невеликі ізольовані блоки, які називаються компонентами. Кожен компонент відповідає за певну частину функціоналу чи відображення та може бути використаний багаторазово в інших частинах додатку. Цей підхід дозволяє розбити складний інтерфейс на більш прості та керовані частини [21].

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
```

}

React використовує віртуальний DOM. Віртуальний DOM (Document Object Model) — це концепція, яку використовує React для оптимізації оновлень інтерфейсу. Замість безпосередньої модифікації реального DOM при кожній зміні стану, React використовує внутрішню віртуальну копію DOM [43].

1. Зміна стану:

- Коли стан компонента змінюється, React створює новий віртуальний DOM для всього дерева компонентів.

2. Порівняння з попереднім станом:

- Він порівнює цей новий віртуальний DOM з попереднім (попереднім станом) і визначає різницю між ними.

3. Оптимізовані оновлення:

- Лише змінені частини віртуального DOM та реального DOM оновлюються, що зменшує навантаження та робить оновлення більш ефективним.

Віртуальний DOM сприяє великою швидкодією реагування на зміни стану, зменшує кількість маніпуляцій з реальним DOM та покращує продуктивність додатку [43].

JSX (JavaScript XML) — це розширення JavaScript, яке дозволяє вписувати HTML-подібний код безпосередньо в JavaScript. Це полегшує роботу з React-елементами та робить код більш зрозумілим [42].

const element = <h1>Hello, React!</h1>;

JSX дозволяє використовувати вирази JavaScript всередині фігурних дужок, що робить його потужним інструментом для динамічного створення інтерфейсу [43].

JSX, разом із зазначеними вище концепціями компонентного підходу та віртуального DOM, утворює потужний інструментарій для розробки динамічних та ефективних інтерфейсів користувача з використанням React.

React ідеально підходить для розробки SPA, де сторінка не перезавантажується під час взаємодії користувача, а лише частково оновлюється. Фреймворк дозволяє ефективно керувати станом додатку. Зазвичай, стан зберігається в верхніх рівнях ієрархії компонентів та передається вниз за

допомогою властивостей (props).

React підтримує два типи компонентів — функціональні та класові. Функціональні компоненти — це прості функції, а класові компоненти можуть мати внутрішній стан та методи життєвого циклу.

Фреймворк забезпечує ефективний механізм обробки подій та взаємодії з користувачем. Дії користувача та події обробляються за допомогою обробників подій.

Angular — це великий фреймворк для створення односторінкових веб-додатків (SPA) і динамічних інтерфейсів користувача. Розроблений та підтримуваний Google, Angular пропонує повноцінний набір інструментів для розробки фронтенду, включаючи обробку стану, маршрутизацію, HTTP-запити та багато іншого [44].

Angular використовує компонентний підхід, розбиваючи великі додатки на малі, незалежні компоненти. Кожен компонент відповідає за певну частину інтерфейсу користувача та має свій власний стан та логіку.

Одна з ключових особливостей Angular - це двостороннє зв'язування даних - це концепція, яка дозволяє автоматично синхронізувати дані між моделлю та відображенням без необхідності вручну написаного коду для оновлення інтерфейсу користувача [44].

При двосторонньому зв'язуванні зміни в моделі автоматично оновлюють відображення, і навпаки. Це означає, що якщо дані змінюються в моделі (наприклад, через користувацьку дію), то ці зміни автоматично відобразяться в інтерфейсі користувача, і навпаки.

Angular використовує двостороннє зв'язування за допомогою директиви ngModel. Це дозволяє легко пов'язувати дані форми з властивостями моделі та автоматично синхронізувати їх [44].

Фреймворк розвиває концепцію модульності та ін'єкції залежностей (Dependency Injection). Модульність - це спосіб організації коду в Angular, де функціонал розбивається на невеликі та самостійні модулі. Кожен модуль може містити компоненти, сервіси, директиви та інші ресурси, що стосуються конкретного функціоналу додатка [45].

Dependency Injection - це концепція, яка полягає в тому, щоб впроваджувати залежності (наприклад, сервіси чи інші об'єкти) в компоненти чи інші частини додатка, замість того, щоб їх створювати всередині самого компонента. В Angular, модуль використовується для організації функціоналу, а Dependency Injection дозволяє вводити та використовувати сервіси чи інші об'єкти в компонентах [45].

Angular включає вбудований модуль роутингу, який дозволяє визначати маршрути та завантажувати відповідні компоненти залежно від URL-адреси (уніфікований локатор ресурсів або Uniform Resource Locator). Це робить навігацію між різними частинами додатка зручною та ефективною.

Фреймворк надає `HttpClient`, що спрощує взаємодію з сервером за допомогою HTTP-запитів. Результати HTTP-запитів можна обробляти асинхронно за допомогою Observable та використовувати їх в компонентах або сервісах. Angular також дозволяє використовувати інтерцептори для обробки запитів та відповідей, що дозволяє вставляти код для обробки перед відправленням запиту чи після отримання відповіді. Це важливий аспект для отримання та відправлення даних на сервер [45].

Angular підтримує високий рівень тестування. Розробники можуть легко написати юніт-тести та тести «енд-ту-енд» для перевірки функціональності додатків.

Vue.js — це прогресивний JavaScript фреймворк для створення користувацьких інтерфейсів. Завдяки своїй простоті та гнучкості, Vue широко використовується для розроблення веб-додатків та односторінкових додатків (SPA). Основними концепціями Vue є декларативність, компонентна архітектура та легкість інтеграції [46].

У Vue.js великий акцент робиться на декларативному підході до розроблення інтерфейсів. Це означає, що розробник описує, як повинен виглядати результат (інтерфейс) і Vue.js відстежує зміни та забезпечує відображення згідно з оновленими даними. Декларативність дозволяє простіше розуміти та зберігати структуру коду. Також Vue.js базується на концепції реактивності. Реактивність полягає у здатності відслідковувати зміни в даних та автоматично оновлювати відображення, яке залежить від цих даних. Використовуючи систему реактивних

об'єктів та властивостей, Vue.js робить код більш прозорим та ефективним.

Vue.js базується на компонентній архітектурі, де весь інтерфейс розбивається на невеликі та самостійні компоненти. Кожен компонент має свою логіку, свої дані та власне відображення. Це полегшує розробку, організацію та підтримку коду [47].

Використання директив, таких як v-if, v-for, v-bind, дозволяє легко маніпулювати DOM та динамічно змінювати властивості елементів.

Vue CLI (Command Line Interface) — це інструмент командного рядка, який допомагає швидко створювати та розгортати проекти Vue.js. Він включає в себе багато зручних функцій для автоматизації рутинних завдань розробки. Vue CLI дозволяє швидко створити новий проект Vue з налаштуванням за замовчуванням або використовуючи популярні стартові шаблони. Є можливість налаштовувати та змінювати конфігурацію проекту за допомогою Vue CLI. Підтримка гарячого перезавантаження дозволяє вам бачити зміни в реальному часі під час розробки. З Vue CLI можливо легко зібрати та оптимізувати свій проект для розгортання на виробничому сервері [47].

Vue Router — офіційний маршрутизатор для Vue.js, який дозволяє легко налаштовувати навігацію в односторінкових додатках.

Vuex — бібліотека для керування станом додатка в Vue. Вона дозволяє ефективно управляти станом за допомогою великої кількості зручних інструментів.

Кожен Vue-компонент має свій власний життєвий цикл, який включає в себе етапи від створення до знищення компонента. Це дозволяє виконувати різні дії на різних етапах життєвого циклу.

Vue.js може легко інтегруватися з іншими бібліотеками та фреймворками, такими як Vuex для керування станом, або з іншими інструментами для розширення його функціональності.

Після докладного розгляду трьох фреймворків (React.js, Angular, та Vue.js), був обран React.js, на це вплинули його переваги, та технічне завдання магістерської дисертації, яка полягає саме в розробленні веб-додатку, який буде односторінковим та використання JSX, який дає можливість описувати інтерфейс

за допомогою розширеного синтаксису, віртуальний DOM, який підвищує ефективність рендерингу .

2.4 Вибір інтегрованого середовища розробки

На основі проведеного огляду та аналізу мов програмування у п.2.2 було обрано мову програмування JavaScript, тому слід вибрати середовище розроблення (IDE – Integrated Development Environment) і, у випадку з JavaScript існує досить багато варіантів таких продуктів. Деякі розробники цілком задовольняються можливостями консольного інтерфейсу та пишуть код у текстових редакторах типу Sublime Text Editor (рис. 2.4).

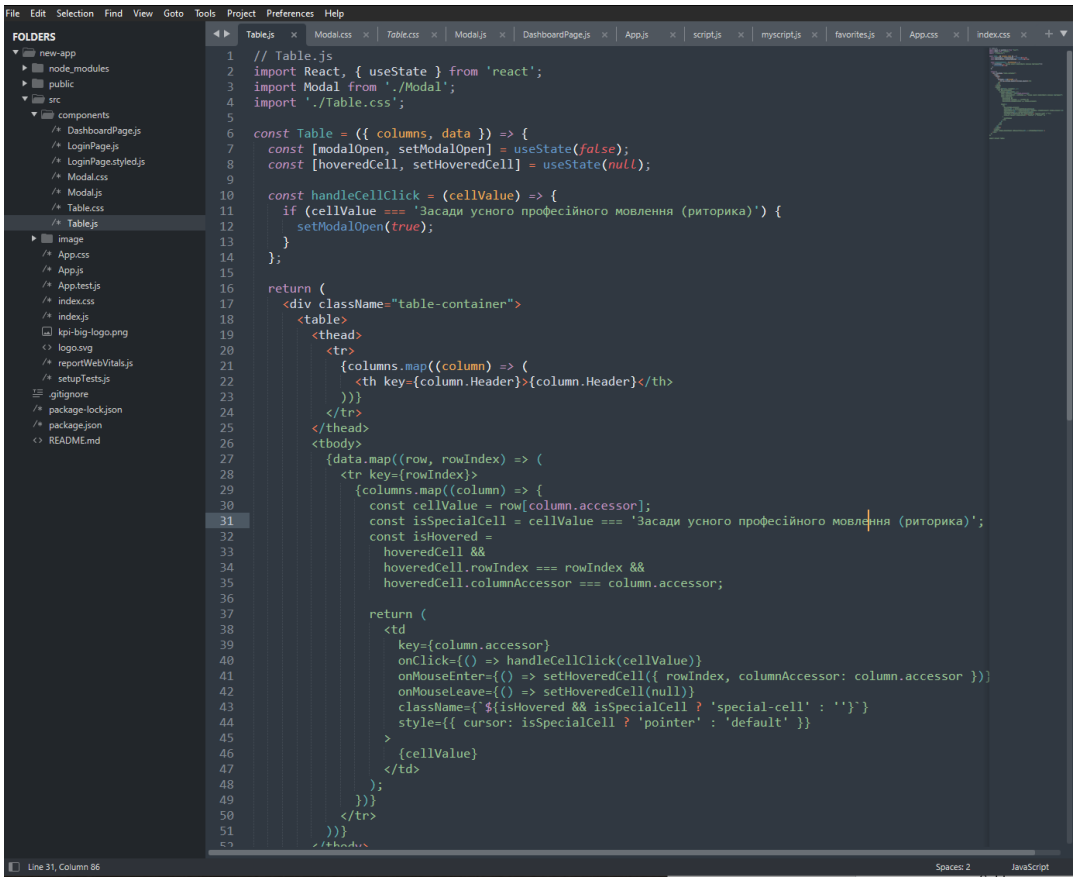


Рис. 2.4 - Інтерфейс текстового редактору Sublime Text Editor

Sublime має підсвічування синтаксису та зручні автодоповнення коду. У лівій частині вікна може відображуватися дерево файлів певного каталогу (папки проекту), що доступно, якщо у головному меню «File» вибрати пункт «Open folder». Важливою рисою даного продукту є його малі системні вимоги, тому він дозволяє комфортно вести розробку навіть на дуже простих ПК [48].

Ще одним варіантом для розробленя на JavaScript є використання ~~«легкого»~~ середовища Visual Studio Code, яке є досить популярним для даної мови програмування. Слід відмітити, що даний продукт має вищі системні вимоги та у базовому варіанті є ненаповненим: усі можливості слід довстановлювати, що з одного боку не є проблемою, а з іншого – може бути неприємним фактором при необхідності активно попрацювати певний час без доступу до Інтернет [49]

Нарешті, можливим є використання одного з потужних середовищ розроблення, наприклад, побудованих на базі движка JetBrains або Microsoft Visual Studio. Слід відмітити, що компанія-розробник JetBrains створила чимало варіацій такого середовища, що підходять для використання з різними мовами програмування. Зокрема, для розроблення на JavaScript зручно використати основний продукт IntelliJ IDEA, яке ефективно підтримує розроблення на JavaScript. Дане інтегроване середовище розроблення має велику кількість переваг, але й великим його недоліком є дуже високі системні вимоги [50].

Було обрано Sublime Text зважаючи на його переваги, а саме через швидкодію середовища, Sublime Text відомий своєю високою швидкодією завантаження та використання. Він має легкий та ефективний двигун, що дозволяє швидко реагувати на дії користувача. Редактор дозволяє відкривати багато файлів одночасно та швидко перемикатися між ними, що особливо корисно при роботі над складними проектами. Він має велику кількість розширень (плагінів) в спільноті розробників, що дозволяє розширити його функціональність та адаптувати під конкретні потреби користувача.

2.5 Вибір бази даних

Слід відмітити, що програмний продукт, який розробляється у даній магістерській дисертації, буде взаємодіяти з користувачем (студентом). Відмітимо, що фізично переважна більшість вказаної інформації являється текстовими рядками, які можуть зберігатися як в одному, так і окремих файлах. Найбільш логічним для таких цілей є використання текстових форматів (доцільніше всього – простого текстового формату *.TXT) або файлів електронних таблиць (у цьому випадку найбільш доцільно використовувати

також досить прості формати типу *.CSV – Comma Separated Values, тобто значення, розділені комами), а також систем керування базами даних (реляційних баз даних (БД) та документо-орієнтованих).

Окремого розгляду потребує питання доцільності використання для окресленої задачі програм для роботи з реляційними базами даних. Цей дуже потужний засіб підлягає використанню у наступних випадках [28]:

- дані, що використовуються, мають досить складну структуру із багатьма взаємозв’язками;

- дані, що підлягають збереженню, прості, але їх дуже багато (мається на увазі самих записів, кожен з яких є сам по собі досить простим);

- для забезпечення цілісності даних (оскільки практично будь-яка система керування базами даних (СКБД) не дозволяє видаляти записи, на які посилаються інші таблиці бази, а за рахунок нормалізації таблиць БД виключаються ситуації опису одного і того ж об’єкту реального світу кількома різними однотипними записами у базі даних – за всім цим слідкує процесор баз даних);

- для забезпечення швидкості доступу до даних при виконанні інтенсивних пошуків записів, що відповідають заданій умові (дуже поширений вид прикладної задачі), а також при виконанні сортувань різних типів (менш поширена операція, але така, що також досить часто зустрічається на практиці).

Було проаналізовано ці умови, при яких доцільно використовувати збереження даних у СКБД, стосовно до задачі, що розглядається. Усі відомості відносяться до індивідуального навчального плану, даних користувачів.

Використання реляційних СКБД для даної роботи є не доцільним, через те, що система має практично повну незалежність окремих компонентів інформації (відсутність зв’язку між ними).

Отже, всі ці факти свідчать про те, що у даному конкретному випадку застосування реляційних СКБД, наповнених обширними табличними даними, не є обґрунтованим, і дозволяє прийняти наступне рішення: цілком достатньо для даної розробки застосувати так звану текстову базу даних.

Текстовою базою даних називають звичайний текстовий файл (часто просто формату *.TXT), у який інформація записується за певними правилами, у форматі,

визначеному розробниками даної конкретної прикладної програми. Розроблення формату збереження даних частково визначається вимогами завдання, а частково будуть визначені технічним завданням (наприклад, порядок слідування окремих компонентів інформації) нижче, у відповідному підрозділі [28].

Сам файл текстової бази даних буде розміщуватись у тому ж каталозі, що й прикладна програма типу *.EXE. Доступ до файлу, відповідно до його суті, можна здійснювати як засобами прикладної програми, що розробляється, так і за допомогою звичайних текстових редакторів типу Блокнот.

Слід відмітити, що ще однією технологією доступу до даних, яку можна застосувати у даній магістерській дисертації, є документо-орієнтована СКБД Mongo. Одним із тривіальних варіантів використання саме MongoDB є дуже простий випадок, коли потрібно десь зберігати налаштування програми (які неефективно розміщувати в таблицях, оскільки практично всі вони є унікальними і не повторюються). В загальному випадку ця система використовується, коли у сховищі даних слід зберігати обширні дані, що не є структурованими, наприклад, являють собою окремі документи (і тому сама СКБД називається документоорієнтованою) [51].

СКБД MongoDB слід завантажити з офіційного серверу mongodb.com, обираючи безкоштовну версію MongoDB Community Server (рис. 2.5).

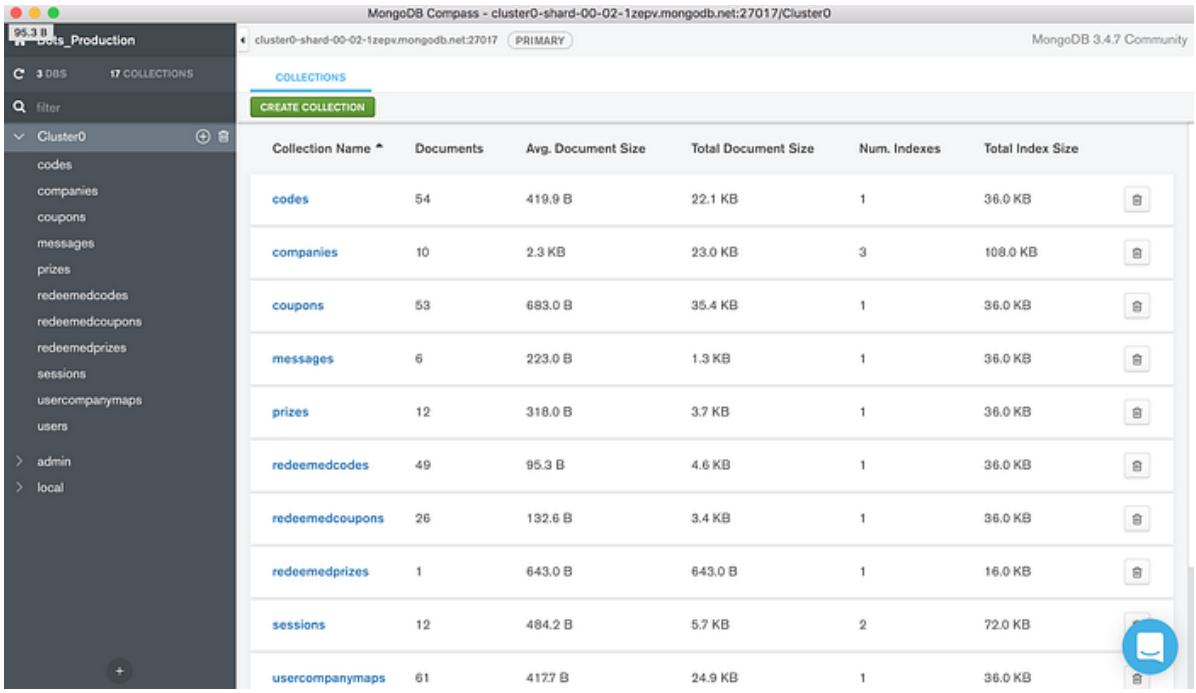


Рис.2.5 - MongoDB Compass

Для роботи з базою даних MongoDB зручно використати спеціальну бібліотеку для мови JavaScript, яка називається mongoose. Mongoose зображає собою спеціальну ODM-бібліотеку (Object Data Modelling) для роботи з MongoDB, вона дозволяє зіставляти об’єкти класів та документи колекцій із бази даних. Дана бібліотека встановлюється у командному рядку в каталозі проекту рядком виду: «npm install mongoose».

Висновки до розділу 2

На основі проведеного огляду та аналізу літературних джерел було прийняте рішення розроблення веб-додатку з використанням мови програмування JavaScript.

Спроектована трьохрівнева архітектура, яка містить модуль авторизації користувача, модуль відображення інформації про користувача, модуль відображення таблиці, модуль створення контенту для модального вікна, підсистему аутентифікації та авторизації, модуль керування даними користувача, модуль керування даними навчальних дисциплін, модуль керування даними силабусів, підсистему прийняття рішень та базу даних. В роботі детально розписано призначення кожного рівня архітектури.

РОЗДІЛ 3

РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ

У даному розділі розглядаються кроки розроблення сучасного програмного забезпечення для веб-додатку та описано детальний огляд інтерфейсу користувача веб-додатку.

3.1 Розробка баз даних

При реалізації бази даних рядок із її налаштуванням вбудовуємо у файл config.js:

const MongodbURI = "mongodb://localhost:27017"

Далі, інформація, що є налаштуваннями веб-сайту, має зберігатися у базі даних MongoDB. У такому випадку створюється схема даних сховища, яка вміщується у каталог models (як відомо, відповідно до шаблону Model-View-Controller дані, що обробляються програмою, відносяться до сутності «модель») та являє собою файл User.js, який наведено на рис.3.1.

```
const mongoose = require('mongoose');

// Connect to the database
mongoose.connect('mongodb://localhost:27017/3000', { useNewUrlParser: true, useUnifiedTopology: true });

// Define the user schema
const userSchema = new mongoose.Schema({
  username: {
    type: String,
    required: true,
    unique: true,
  },
  password: {
    type: String,
    required: true,
  },
  fullName: {
    type: String,
    required: true,
  },
  // Other user fields, if any
});

// Create the user model
const User = mongoose.model('User', userSchema);

module.exports = User;
```

Рис 3.1 - Налаштування бази даних

В базі даних зберігається інформація про користувача, а саме логін, пароль, ім’я та прізвище користувача - рис.3.2. При додаванні нового користувача у БД

створюється новий запис наступного виду – рис. 3.3.

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const User = require('./userModel');

const app = express();
const port = 3000;

app.use(bodyParser.json());

// Connect to the database
mongoose.connect('mongodb://localhost:27017/your_database', { useNewUrlParser: true, useUnifiedTopology: true });

// Endpoint to create and save two users
app.post('/createUsers', async (req, res) => {
  try {
    // Creating user 1
    const user1 = new User({
      username: 'Maksym1',
      password: 'Password1',
      fullName: 'Максим Білич',
    });
    await user1.save();

    // Creating user 2
    const user2 = new User({
      username: 'Maksym2',
      password: 'Password2',
      fullName: 'Вікторія Накорик',
    });
    await user2.save();

    res.send('Users created successfully.');
  } catch (error) {
    console.error(error);
    res.status(500).send('Internal Server Error');
  }
});

app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

Рис 3.2 - Частина база даних, де зберігається інформація про користувача

```
const user2 = new User({
  username: 'Maksym2',
  password: 'Password2',
  fullName: 'Вікторія Накорик',
});
await user2.save();
```

Рис 3.3 - Додавання нового користувача

На етапі створення моделі даних формується семантична модель, що є високорівневим абстрактним відображенням. При розробленні цієї моделі необхідно відокремитися від конкретних систем керування базами даних (СКБД) та особливостей фізичного зберігання. Процес моделювання розпочинається з

аналізу предметної області та визначення вимог користувачів. При розгляді бази даних розроблювальної системи виявляється, що різні користувачі ставлять однакові вимоги до цієї бази даних. Студентам потрібно переглядати індивідуальний навчальний план навчання та отримати детальну інформацію по дисципліні, які вивчається.

ER-діаграма (Entity-Relationship diagram) - це графічний інструмент, який використовується для моделювання взаємозв'язків між сутностями в базах даних. Ця діаграма відображає структуру бази даних, визначає, які сутності існують, та як вони взаємодіють між собою [52].

Основною метою ER-діаграми є візуалізація структури інформації та зв'язків між різними об'єктами в системі. Основні компоненти ER-діаграми включають сутності, атрибути і зв'язки. Сутності представляють основні об'єкти або концепції, що зберігаються в базі даних. Наприклад, може бути сутність «Студент», «Курс», «Викладач» тощо. Атрибути це властивості чи характеристики сутностей. Наприклад, атрибути для сутності «Студент» можуть включати ім'я, прізвище, номер студентського квитка тощо. Зв'язки показують взаємозв'язки між різними сутностями. Наприклад, зв'язок «Студент бере участь у Курс» визначає взаємодію між сутностями «Студент» і «Курс» [52].

Зв'язок один до одного (1:1): Наприклад, один студент має один номер студентського квитка і кожен номер студентського квитка призначений лише одному студентові.

Зв'язок Один до багатьох (1:M): Наприклад, один викладач може мати багато студентів, але кожен студент може мати лише одного викладача.

Зв'язок Багато до багатьох (M:N): Наприклад, багато студентів може брати участь у багатьох курсах і багато курсів може мати багато студентів.

ER-діаграми є важливим інструментом при проектуванні баз даних, оскільки вони допомагають розуміти логічну структуру даних та взаємодії між ними. Вони дозволяють визначити, як дані пов'язані одне з одним, і це допомагає при створенні ефективних та оптимізованих баз даних. ER-діаграми є основою для подальшої реалізації баз даних на рівні програмування [52]

Для розроблюваної бази даних використовується «зв’язок один до одного»

та «один-до-багатьох» [52]. Взаємозв'язок «один до одного» означає, що кожному конкретному представнику однієї сутності відповідає лише один представник іншої сутності (рис 3.4.).

Рис 3.4. – «Модель один-до-одного»

Наступний зв’язок «один до багатьох», в якому об’єкт одного виду може відповідати декілька об’єктів іншого виду, але не навпаки (рис 3.5.).

Рис 3.5. – «Модель один-до-багатьох»

Розглянемо зв’язки, які зображені на ER-моделі (рис 3.6). Зв’язок «є» пов’язує сутність «користувач» та сутності «студент». Цей зв’язок один до одного, так як один об’єкт сутності «користувач» відповідає одному і лише одному об’єкту сутності «студент», і навпаки. Зв’язок «Використовує» поєднує сутності «Модальне вікно» та «Таблиця». Такий зв’язок має потужність один до багатьох, тому що студент може просто переглянути таблицю, або скористатись модальним вікном.

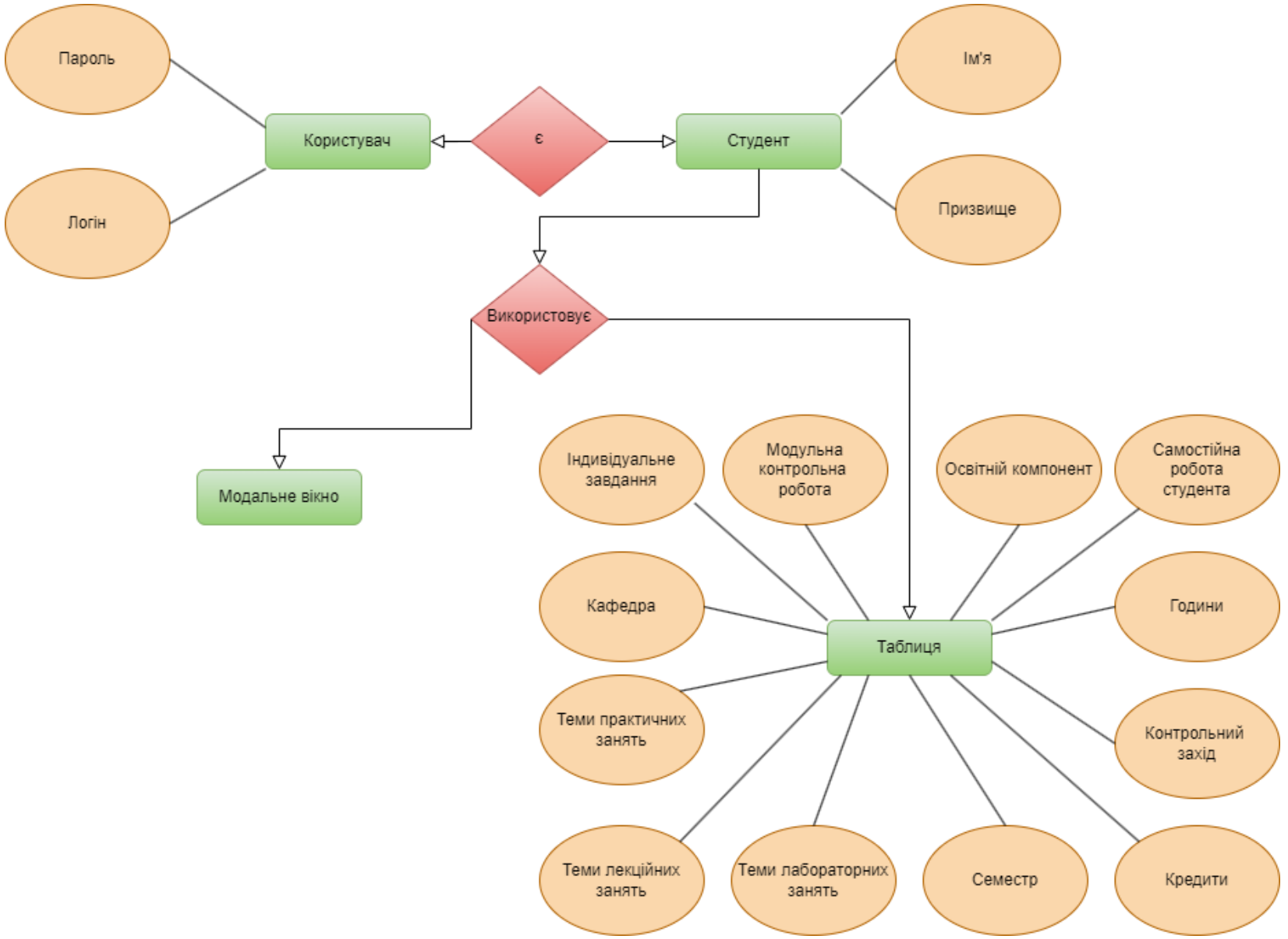


Рис 3.6 - ER-модель бази даних

3.2 Розроблення графічного інтерфейсу проектованого програмного забезпечення

Згідно розробленій структурі, яка представлена на рис. 2.2, була спроектована сторінка авторизації користувачів та головна сторінка, яка включає інформацію про студента та інтерактивну таблицю з індивідуальним навчальним планом.

У розроблення веб-додатка включено створення макету веб-сторінок, на якому в подальшому будуть розташовані інші елементи. При цьому формується так званий структурний блок сайту - ізольовані модулі, кожен з яких відповідає за конкретний функціонал додатка.

Веб-додаток складається з елементів, які наведено на рис 3.7.

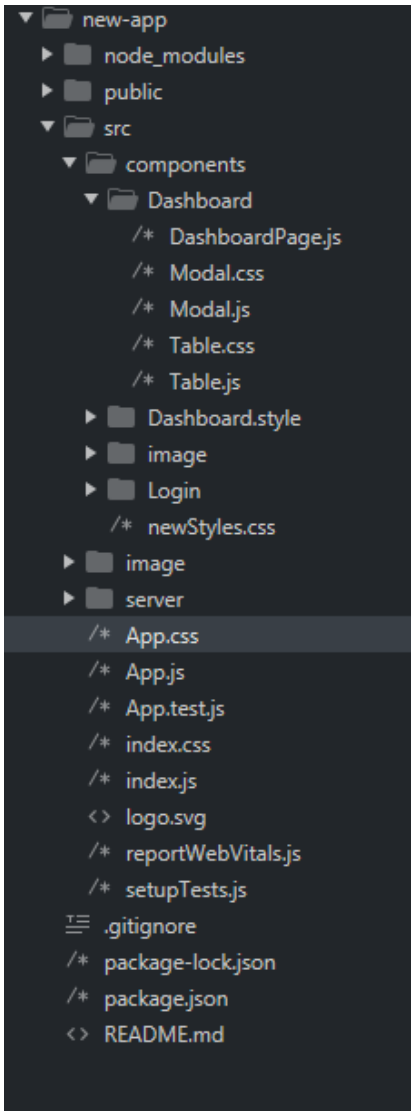


Рис.3.7 - Структура компонентів веб-додатку

- Сторінка авторизації користувачів

Для взаємодії з веб-додатком користувачеві потрібно буде авторизуватися, оскільки кожен обліковий запис має індивідуальну інформацію про користувача та сформований індивідуальним навчальний план. Без активного облікового запису розроблювальний веб-додаток буде непрацездатним. Реєстрація відсутня, дані для авторизації генеруються для кожного користувача окремо так, як інформація для кожного користувача є індивідуальною, й наперед необхідно знати, хто логіниться.

Аутентифікація виконується за допомогою POST запиту. Компонент використовує fetch для відправлення POST-запиту на URL 'http://localhost:3000/login'. У запиті вказується метод 'POST', а також встановлюються заголовки, зокрема 'Content-Type', що вказує серверу на прийняття даних у форматі JSON. Дані для аутентифікації (ім'я користувача та пароль) передаються у тілі запиту, яке містить JSON-об'єкт з цими даними:

```
{
  "username": "введене_ім'я_користувача",
  "password": "введений_пароль"
}
```

- Головна сторінка

Користувач бачить основний контент сторінки, який включає інформацію про назву університету, навчальний рік, групу, курс, назви кафедри та факультету, форму навчання, рівень вищої освіти, назви спеціальності та освітньо-професійної програми. Користувач бачить таблицю з даними, яка відображає різні освітні компоненти разом із зазначеними атрибутами, наприклад, такими як кількість аудиторних годин (лекційні, практичні та лабораторні заняття) та інші.

Користувач може клікати на різні комірки таблиці. Якщо користувач клікає на певну комірку, то відкривається модальне вікно із зазначеним вмістом. Користувач бачить модальне вікно з текстовим вмістом в залежності від того, на яку комірку він клікнув. Модальне вікно може містити текстові повідомлення, які пояснюють певні аспекти, такі як інформація про теми лекцій, опис предмету та

інші.

Для створення таблиці використовується бібліотека react-table.

Хук React.useMemo() використовується в бібліотеці React для мемоізації значень. Він дозволяє кешувати результат обчислень і повертати закешоване значення, коли компонент повторно рендериться, але зберігає його незмінним, якщо змінні, вказані в залежностях, залишаються незмінними.

Columns масив об'єктів, який представляє стовпці таблиці. Кожен об'єкт має властивості Header (назва стовпця, яка відображається в заголовку) та accessor (ключ, який вказує на поле в об'єктах даних для отримання значення).

На рис. 3.8 наведено створення таблиці за допомогою react-table.

```
const columns = React.useMemo(
  () => [
    { Header: '№ п/п', accessor: '№ п/п' },
    { Header: 'Освітній компонент', accessor: 'Освітній компонент' },
    { Header: 'Кафедра', accessor: 'Кафедра' },
    { Header: 'Сем', accessor: 'Сем' },
    { Header: 'Кред', accessor: 'Кред' },
    { Header: 'Год', accessor: 'Год' },
    { Header: 'Лек', accessor: 'Лек' },
    { Header: 'Прак', accessor: 'Прак' },
    { Header: 'Лаб', accessor: 'Лаб' },
    { Header: 'Інд.звд', accessor: 'Інд.звд' },
    { Header: 'СРС', accessor: 'СРС' },
    { Header: 'Контр', accessor: 'Контр' },
    { Header: 'МКР', accessor: 'МКР' },
  ],
);
```

Рис 3.8 - Створення таблиці за допомогою react-table

Для реалізації модального вікна використовується бібліотека react-modal.

const Modal = ({ isOpen, onRequestClose, content }) => {

Компонент отримує три пропси: isOpen (чи відкрите модальне вікно), onRequestClose (функція для закриття модального вікна) і content (вміст модального вікна).

<ReactModal

isOpen={isOpen}

onRequestClose={onRequestClose}

```
    className="custom-modal"
    overlayClassName="custom-modal-overlay"
    contentLabel="Модальне вікно"
    ariaHideApp={false}
>
```

У вище наведеному коді використовується компонент ReactModal та конфігурується за допомогою пропсів, таких як isOpen, onRequestClose, className, overlayClassName, contentLabel, та ariaHideApp. Також була додана кнопка для закриття модального вікна, яка викликає функцію onRequestClose при натисканні. На рис.3.9 представлено реалізація функції закриття вікна.

```
    <div className="modal-header">
      <button className="close-button" onClick={onRequestClose}>
        &times;
      </button>
    </div>
    <div className="modal-content">{content}</div>
  </ReactModal>
```

Рис 3.9 - Реалізація функції закриття вікна

Для реалізації виведення інформації використовуються хуки useState, щоб створити стани для визначення того, чи відкрите модальне вікно (isModalOpen) і для збереження вмісту модального вікна (modalContent). На рис. 3.10 приведена реалізація виведення інформації.

```
const [isModalOpen, setModalOpen] = useState(false);
const [modalContent, setModalContent] = useState(null);
```

Рис 3.10 - Реалізація виведення інформації

Функції обробки подій виконуються за допомогою handleCellClick, яка відповідає за обробку кліку на комірці таблиці. Якщо клікнуто на певну комірку, то встановлюється стан modalContent згідно логіки у функції getModalContent, і встановлюється isModalOpen в true. Стан closeModal викликається для закриття модального вікна. Встановлює isModalOpen в false і очищує modalContent.

Функція handleCellClick викликається при кліку на комірку в таблиці.

Перевіряється, чи rowIndex дорівнює заданому номеру в рядку таблиці. Також перевіряється, чи columnIndex входить до масиву заданих стовпців таблиці. Якщо обидві умови виконуються, то викликаються наступні дії:

1. Визивається функція getModalContent з передачею rowIndex та columnIndex.
2. Отриманий контент встановлюється за допомогою setModalContent.
3. Встановлюється стан isModalOpen в true за допомогою setModalOpen, відкриваючи модальне вікно

Функція getModalContent призначена для отримання контенту модального вікна в залежності від індексів рядка і стовпця в таблиці. Функція отримує параметри rowIndex (рядок) і columnIndex (стовпчик), які вказують на конкретну комірку в таблиці. Створюється унікальний ключ комірки за допомогою рядка ${rowIndex}.${columnIndex}. Цей ключ унікально ідентифікує комірку в таблиці. Здійснюється перевірка ключа комірки у виразі switch. Якщо ключ відповідає 'X1.Y1', то повертається . Користувач бачить всю необхідну інформацію , якщо комірка 'X2.Y2', то буде виводитись вся інша інформація. Отже, ця функція визначає контент модального вікна на основі конкретної комірки в таблиці, забезпечуючи різний вміст для різних комірок.

Також слід зазначити, що інформація, яка виводиться у модальному вікні надходить з сервера, для цього використовує хук useEffect для асинхронного завантаження даних з файлу Silabus.json при монтуванні компонента. Хук useEffect викликається після того, як компонент змонтовано. У функції fetchSilabusData виконується асинхронний запит до сервера за допомогою fetch.

Для запиту на сервер використовується fetch для виконання запиту GET за адресою файлу Silabus.json на сервері. Перевіряється статус відповіді, і якщо він не є успішним (не 200), видається помилка. Якщо запит успішний, отримані дані парсяться з JSON формату за допомогою response.json() - це метод об'єкта Response у JavaScript, який дозволяє асинхронно отримати тіло відповіді в форматі JSON. Встановлюється стан silabusData за допомогою отриманих та, при необхідності, парсених даних. Помилки обробляються та виводяться в консоль

для налагодження.

3.3 Взаємодія з базою даних силабуса

Для виведення інформації про навчальну дисципліну (освітній компонент) засовуються дані з бази даних силабусів.

Для цього використовуються різні модулі для різних функціональностей:

- express для створення веб-сервера та обробки веб-запитів;
- mongoose для взаємодії з базою даних MongoDB;
- multer для обробки завантаження файлів на сервер;
- path для роботи з шляхами до файлів та каталогів;
- body-parser для розбору тіла запиту.

Конфігурація Multer для завантаження файлів. Multer налаштовується для зберігання завантажених файлів у заданому каталозі із збереженням їхніх оригінальних імен файлів.

Далі необхідно з’єднатись з базою даних MongoDB за допомогою бібліотеки mongoose. Опис кожної частини наведено у табл.3.1.

Таблиця 3.1

Приклад з коду з’єднання з базою даних

Частина коду	Пояснення
mongoose.connect:	Цей метод використовується для встановлення з'єднання з базою даних
{ useNewUrlParser: true }:	Ця опція вказує Mongoose використовувати новий парсер URL для з'єднання з MongoDB. Це обов'язково для усунення попереджень про застарілість та коректне підключення.
.then(() => console.log('connected to db')):	Це метод об’єкта Promise на успішне встановлення з'єднання з базою даних. Коли з'єднання успішно встановлено, виводиться повідомлення "connected to db".
.catch((err) => console.log(err)):	Якщо виникає помилка при встановленні з'єднання, вона обробляється у блоку catch, і повідомлення про помилку виводиться у консоль.

Налаштування основних параметрів веб-додатку за допомогою фреймворку Express наведено у табл. 3.2.

Таблиця 3.2

Налаштування веб-додатку за допомогою фреймворку Express

Частина коду	Пояснення
var app = express(). express()	Це функція, яка створює екземпляр застосунку Express. Цей екземпляр буде використовуватися для налаштування та визначення обробників маршрутів.
app.set('view engine', 'ejs')	Ця функція викликає та встановлює двигун шаблонів для Express, і в даному випадку, це EJS (Embedded JavaScript). Він вказує, що ваші шаблони відображення будуть написані за допомогою EJS.
app.use (bodyParser.urlencoded({ extended: false }))	Це middleware, яке використовується для обробки запитів з формами, передаючи дані форми в об'єкт req.body. В даному випадку, використовується body-parser, який дозволяє розбирати дані форми з urlencoded типом.
pp.use(express.static(path.resolve(__dirname, 'public'))):	Це middleware для вказаної статичної директорії, в даному випадку, ./public. Всі статичні файли (наприклад, CSS, JavaScript) з цієї директорії будуть доступні з публічної адреси сервера.

При запиті на головну сторінку відбувається витягування даних з бази даних та їх відображення на сторінці. Код визначає обробник маршруту для HTTP GET-запитів до кореневого шляху ('/') веб-додатку за допомогою фреймворку Express, отримання GET-запиту до кореневого шляху веб-додатку наведено у табл. 3.3.

Таблиця 3.3

Отримання GET-запиту до кореневого шляху веб-додатку

Частина коду	Пояснення
app.get('/')	Цей обробник маршруту викликається при отриманні HTTP GET-запиту на кореневий шлях.
(req, res) => { ... }	Ця частина представляє собою функцію, яка викликається при отриманні запиту. req - це об'єкт запиту (request), res - об'єкт відповіді (response).
(err, data) => { ... }	Це зворотний виклик (callback), який обробляє результат find. Якщо виникає помилка (err не пустий), вона виводиться у консоль (console.log(err)). Якщо дані успішно знайдено, вони передаються до шаблону.
res.render('demo', { data: data })	Цей виклик відправляє відповідь клієнту у вигляді рендерингу шаблону.

Отже, при отриманні GET-запиту до кореневого шляху веб-додатку, дані з бази даних витягуються, і, якщо вони існують, вони передаються для рендерингу в шаблон.

```
app.post('/', uploads.single('csv'), (req, res) => {
    csv().fromFile(req.file.path).then((jsonObj) => {
        // Конвертація числових значень
        for (var x = 0; x < jsonObj.length; x++) {
            // ...
        }
        // Збереження даних у  БД
        csvModel.insertMany(jsonObj, (err, data) => {
            if (err) {
                console.log(err);
            } else {
                res.redirect('/');
            }
```

```
        });
    });
});
```

Цей код визначає обробник маршруту для HTTP POST-запитів на кореневий шлях веб-додатку за допомогою фреймворку Express. У табл. 3.4 наведено HTTP POST-запитів.

Таблиця 3.4

Обробник маршруту для HTTP POST-запитів

Частина коду	Пояснення
app.post('/')	Цей обробник маршруту викликається при отриманні HTTP POST-запиту на кореневий шлях ('/'). Тобто, коли дані надсилаються на сервер через форму або інший метод POST.
uploads.single('csv')	Це middleware, яке використовується для обробки одного файлу з назвою 'csv'. Це передає дані файлу у властивість req.file.
csv().fromFile(req.file.path).then((jsonObj) => { ... })	Використовується бібліотека csvtojson, яка конвертує дані з CSV-файлу в об'єкт JavaScript (JSON). req.file.path містить шлях до завантаженого файлу.
for (var x = 0; x < jsonObj.length; x++) { ... }	Цикл, який використовується для конвертації числових значень у JSON-об'єкті.
csvModel.insertMany(jsonObj, (err, data) => { ... })	використовує модель csvModel для вставки багатьох записів у базу даних MongoDB. Зворотній виклик обробляє помилку чи успіх операції.
res.redirect('/')	Якщо дані успішно вставлені у базу даних, відбувається перенаправлення на кореневий шлях (відображення головної сторінки).

3.4 Інструкція користувача

3.4.1. Початок роботи з веб-додатком

Для взаємодії з веб-додатком користувач спочатку відкриває головний екран програми, де відображено інформацію про функції додатку. На цій сторінці користувач може обрати наступний крок. На рис.3.11 наведена сторінка авторизації

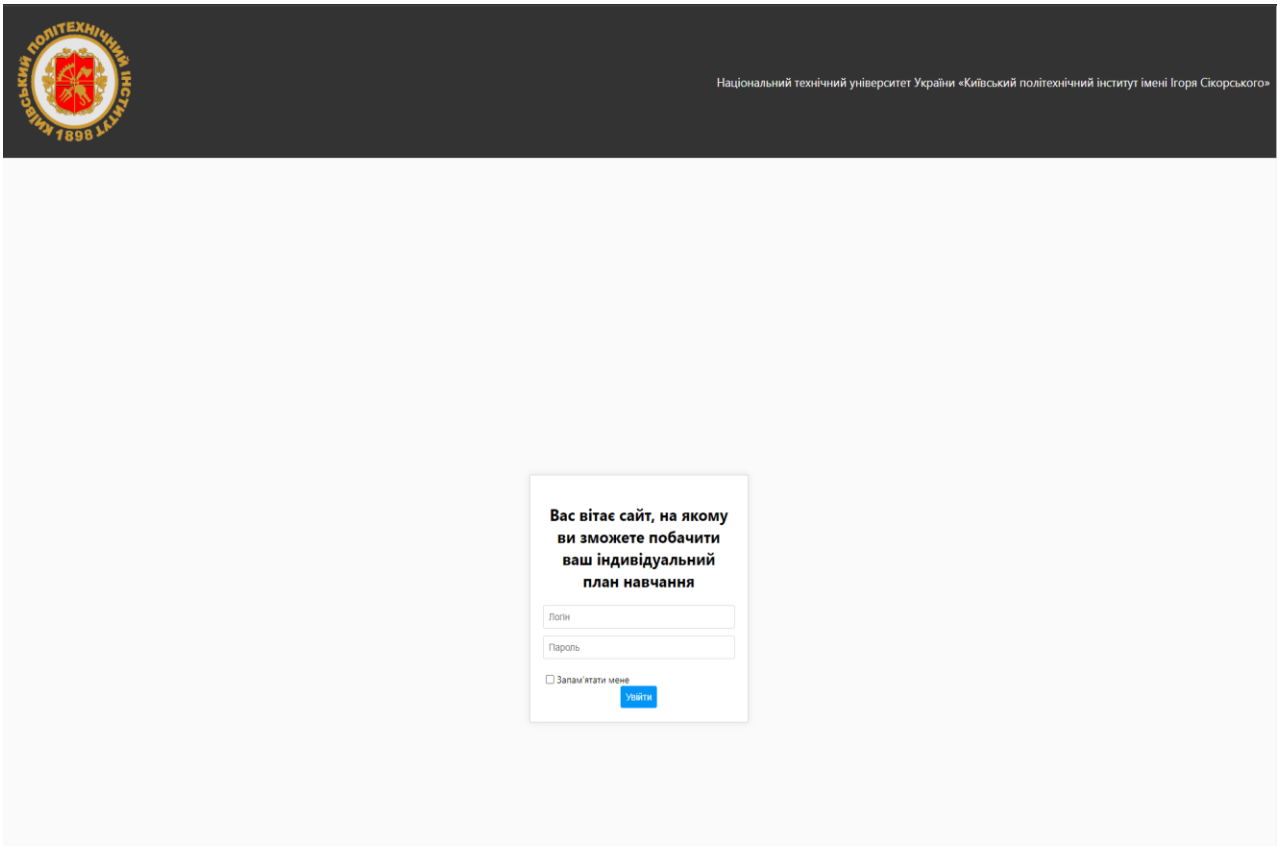


Рис 3.11 - Сторінка авторизації користувача

Для початку користування веб-додатком, обов'язковою є авторизація користувача в системі. Без цього кроку доступ та використання додатку неможливі. Під час авторизації необхідно ввести дані, такі як логін та пароль, та поставити «галочку» біля «Запам'ятати мене» за бажанням, для збереження даних у локальному сховищі браузера. Поля логін та пароль обов'язкові до заповнення, і в самому додатку вбудовано автоматичну валідацію введених даних. Якщо користувач неправильно вказує дані, додаток повертає йому повідомлення про помилку (рис. 3.12).

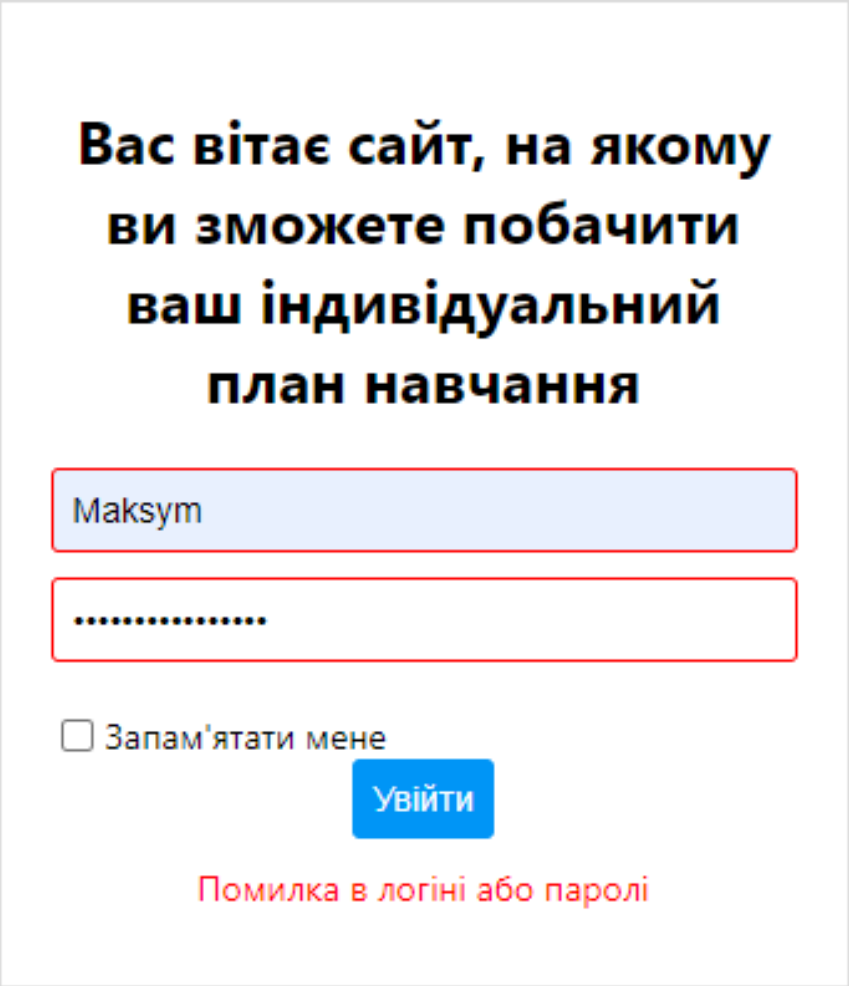
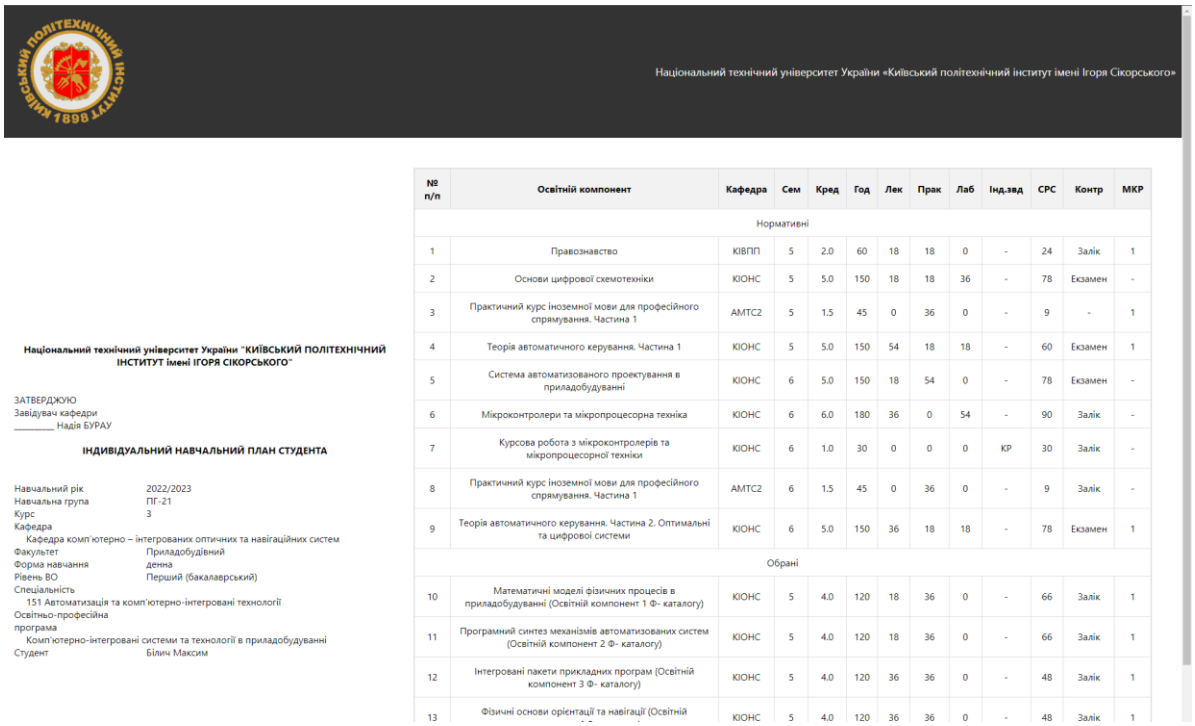


Рис 3.12 - Приклад помилки, при введенні некоректних даних

3.4.2. Головна сторінка веб-додатку

Після успішної авторизації за даними логіну й паролю, які надаються, користувач потрапляє на головну сторінку додатку (рис 3.13). У лівій частині сторінки користувач бачить загальну інформацію про себе та навчальний заклад (рис 3.14), а саме інформацію про назву університету, навчальний рік, групу, курс, назви кафедри та факультету, форму навчання, рівень вищої освіти, назви спеціальності та освітньо-професійної програми.



Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

Національний технічний університет України "КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО"

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Надія БУРАУ

ІНДИВІДУАЛЬНИЙ НАВЧАЛЬНИЙ ПЛАН СТУДЕНТА

Навчальний рік 2022/2023
Навчальна група ПГ-21
Курс 3
Кафедра Кафедра комп'ютерно – інтегрованих оптичних та навігаційних систем
Факультет Приладобудівний
Форма навчання денна
Рівень ВО Перший (бакалаврський)
Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
Освітньо-професійна програма Комп'ютерно-інтегровані системи та технології в приладобудуванні
Студент Білич Максим

№ п/п	Освітній компонент	Кафедра	Сем	Кред	Год	Лек	Прак	Лаб	Інд.зад	СРС	Контр	МКР
Нормативні												
1	Правознавство	КІВПП	5	2.0	60	18	18	0	-	24	Залік	1
2	Основи цифрової схемотехніки	КІОНС	5	5.0	150	18	18	36	-	78	Екзамен	-
3	Практичний курс іноземної мови для професійного спрямування. Частина 1	АМТС2	5	1.5	45	0	36	0	-	9	-	1
4	Теорія автоматичного керування. Частина 1	КІОНС	5	5.0	150	54	18	18	-	60	Екзамен	1
5	Система автоматизованого проектування в приладобудуванні	КІОНС	6	5.0	150	18	54	0	-	78	Екзамен	-
6	Мікроконтролери та мікропроцесорна техніка	КІОНС	6	6.0	180	36	0	54	-	90	Залік	-
7	Курсова робота з мікроконтролерів та мікропроцесорної техніки	КІОНС	6	1.0	30	0	0	0	КР	30	Залік	-
8	Практичний курс іноземної мови для професійного спрямування. Частина 1	АМТС2	6	1.5	45	0	36	0	-	9	Залік	-
9	Теорія автоматичного керування. Частина 2. Оптимальні та цифрові системи	КІОНС	6	5.0	150	36	18	18	-	78	Екзамен	1
Обрані												
10	Математичні моделі фізичних процесів в приладобудуванні (Освітній компонент 1 Ф- каталогу)	КІОНС	5	4.0	120	18	36	0	-	66	Залік	1
11	Програмний синтез механізмів автоматизованих систем (Освітній компонент 2 Ф- каталогу)	КІОНС	5	4.0	120	18	36	0	-	66	Залік	1
12	Інтегровані пакети прикладних програм (Освітній компонент 3 Ф- каталогу)	КІОНС	5	4.0	120	36	36	0	-	48	Залік	1
13	Фізичні основи орієнтації та навігації (Освітній	КІОНС	5	4.0	120	36	36	0	-	48	Залік	1

Рис 3.13 - Головна сторінка додатку

Національний технічний університет України "КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО"

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Надія БУРАУ

ІНДИВІДУАЛЬНИЙ НАВЧАЛЬНИЙ ПЛАН СТУДЕНТА

Навчальний рік	2022/2023
Навчальна група	ПГ-21
Курс	3
Кафедра	Кафедра комп'ютерно – інтегрованих оптичних та навігаційних систем
Факультет	Приладобудівний
Форма навчання	денна
Рівень ВО	Перший (бакалаврський)
Спеціальність	151 Автоматизація та комп'ютерно-інтегровані технології
Освітньо-професійна програма	Комп'ютерно-інтегровані системи та технології в приладобудуванні
Студент	Білич Максим

Рис 3.14 - Загальна інформація для користувача

У правій частині сторінки користувач бачить інтерактивну таблицю (рис

3.15), в якій надається інформація про назву освітнього компоненту, кафедри, семестр, загальний обсяг дисципліни (кількість кредитів ECTS та годин), кількість аудиторних годин (лекційні, практичні та лабораторні заняття), кількість годин відведених для самостійної роботи студента, контрольні заходи (екзамен, залік, модульна контрольна робота (МКР), розрахунково-графічна робота (РГР), розрахункова робота (РР), графічна робота (ГР), реферати).

№ п/п	Освітній компонент	Кафедра	Сем	Кред	Год	Лек	Прак	Лаб	Інд.звд	СРС	Контр	МКР
Нормативні												
1	Правознавство	КІВПП	5	2.0	60	18	18	0	-	24	Залік	1
2	Основи цифрової схемотехніки	КІОНС	5	5.0	150	18	18	36	-	78	Екзамен	-
3	Практичний курс іноземної мови для професійного спрямування. Частина 1	АМТС2	5	1.5	45	0	36	0	-	9	-	1
4	Теорія автоматичного керування. Частина 1	КІОНС	5	5.0	150	54	18	18	-	60	Екзамен	1
5	Система автоматизованого проектування в приладобудуванні	КІОНС	6	5.0	150	18	54	0	-	78	Екзамен	-
6	Мікроконтролери та мікропроцесорна техніка	КІОНС	6	6.0	180	36	0	54	-	90	Залік	-
7	Курсова робота з мікроконтролерів та мікропроцесорної техніки	КІОНС	6	1.0	30	0	0	0	КР	30	Залік	-
8	Практичний курс іноземної мови для професійного спрямування. Частина 1	АМТС2	6	1.5	45	0	36	0	-	9	Залік	-
9	Теорія автоматичного керування. Частина 2. Оптимальні та цифрові системи	КІОНС	6	5.0	150	36	18	18	-	78	Екзамен	1
Обрані												
10	Математичні моделі фізичних процесів в приладобудуванні (Освітній компонент 1 Ф- каталогу)	КІОНС	5	4.0	120	18	36	0	-	66	Залік	1
11	Програмний синтез механізмів автоматизованих систем (Освітній компонент 2 Ф- каталогу)	КІОНС	5	4.0	120	18	36	0	-	66	Залік	1
12	Інтегровані пакети прикладних програм (Освітній компонент 3 Ф- каталогу)	КІОНС	5	4.0	120	36	36	0	-	48	Залік	1
13	Фізичні основи орієнтації та навігації (Освітній компонент 4 Ф- каталогу)	КІОНС	5	4.0	120	36	36	0	-	48	Залік	1
14	Теорія чутливих елементів системи навігації і орієнтації (Освітній компонент 5 Ф- каталогу)	КІОНС	6	4.0	120	18	36	0	-	66	Залік	1
15	Дифракція і поляризація світла (Освітній компонент 6 Ф- каталогу)	КІОНС	6	4.0	120	36	36	0	-	48	Залік	1
16	Програмні алгоритми та структури даних (Освітній компонент 7 Ф- каталогу)	КІОНС	6	4.0	120	36	36	0	-	48	Залік	1

Рис 3.15 - Таблиця з навчальними дисциплінами (освітніми компонентами)

Коли користувач наводить курсор на комірку з рядка освітнього компоненту, де є інформація по дисципліну вона виділяються. Це дозволяє користувачеві швидко визначати активні області та здійснювати точний вибір. Це продемонстровано на рис. 3.16.

6	Мікроконтролери та мікропроцесорна техніка	КІОНС	6
7	Курсова робота з мікроконтролерів та мікропроцесорної техніки	КІОНС	6
8	Практичний курс іноземної мови для професійного спрямування. Частина 1	АМТС2	6
9	Теорія автоматичного керування. Частина 2. Оптимальні та цифрові системи	КІОНС	6
	Обрані		
10	Математичні моделі фізичних процесів в приладобудуванні (Освітній компонент 1 Ф- каталогу)	КІОНС	5
11	Програмний синтез механізмів автоматизованих систем (Освітній компонент 2 Ф- каталогу)	КІОНС	5
12	Інтегровані пакети прикладних програм (Освітній компонент 3 Ф- каталогу)	КІОНС	5
13	Фізичні основи орієнтації та навігації (Освітній компонент 4 Ф- каталогу)	КІОНС	5
14	Теорія чутливих елементів системи навігації і орієнтації (Освітній компонент 5 Ф- каталогу)	КІОНС	6
15	Дифракція і поляризація світла (Освітній компонент 6 Ф- каталогу)	КІОНС	6
1(	Програмні алгоритми та структури даних (Освітній компонент 7 Ф- каталогу)	ОНС	6

Рис 3.16 - Приклад активної комірки при наведені курсором миші

Якщо користувач наводить курсор на комірку з рядка освітнього компоненту, де немає інформація по дисципліні, то вона не виділяється й ніяк взаємодіяти не може, лише підсвічується рядок для зручної навігації (рис 3.17).

10	Математичні моделі фізичних процесів в приладобудуванні (Освітній компонент 1 Ф- каталогу)
11	Програмний синтез механізмів автоматизованих систем (Освітній компонент 2 Ф- каталогу)
12	Інтегровані пакети прикладних програм (Освітній компонент 3 Ф- каталогу)
13	Фізичні основи орієнтації та навігації (Освітній компонент 4 Ф- каталогу)
14	Теорія чутливих елементів системи навігації і орієнтації (Освітній компонент 5 Ф- каталогу)
15	Дифракція і поляризація світла (Освітній компонент 6 Ф- каталогу)
16	Програмні алгоритми та структури даних (Освітній компонент 7 Ф- каталогу)

Рис 3.17 - Приклад неактивної комірки при наведені курсором миші

Як було зазначено вище при наведені курсору на комірку з рядка освітнього компоненту, де є інформація по дисципліні вона виділяються, це означає що при натисканні на неї користувачеві відкриється модальне вікно, де буде зазначена інформація в залежності від того, яку комірку обрав користувач (рис 3.18).

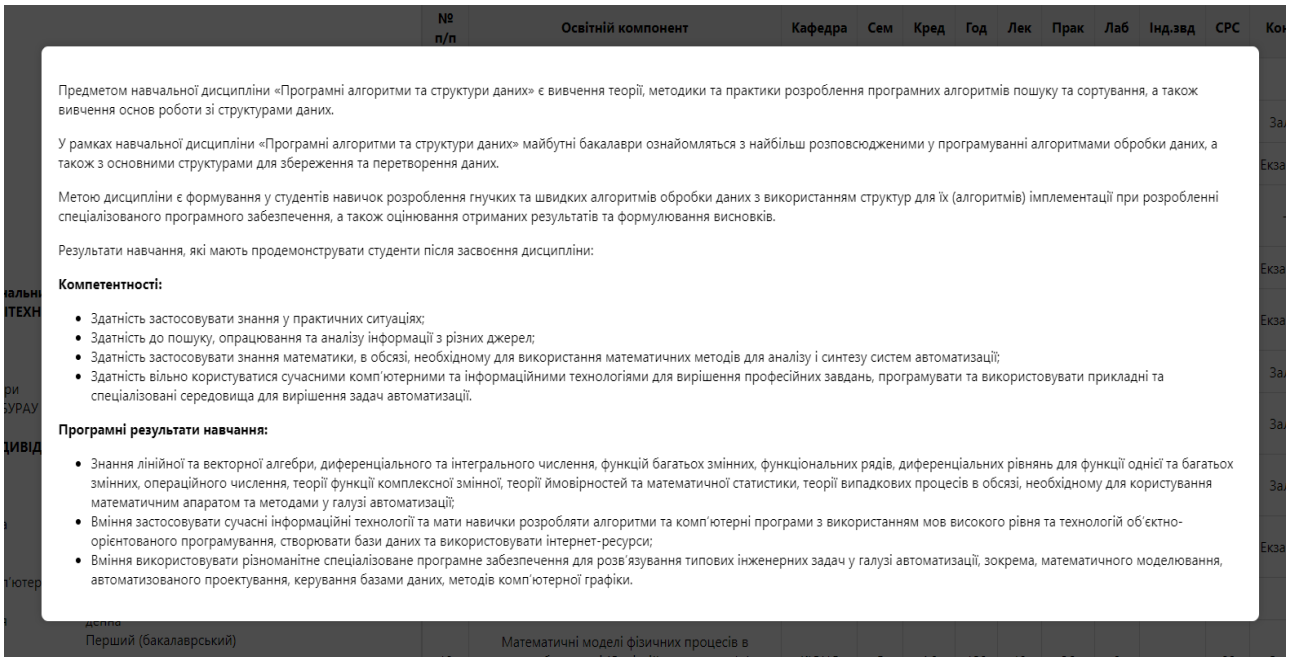


Рис 3.18 - Інформація з модального вікна при натисканні на освітній компонент

Якщо користувач обирає іншу комірку в таблиці, інформація в модальному вікні, буде відповідно до комірки, з якої взаємодіяли. Приклад наведено на рис. 3.19.

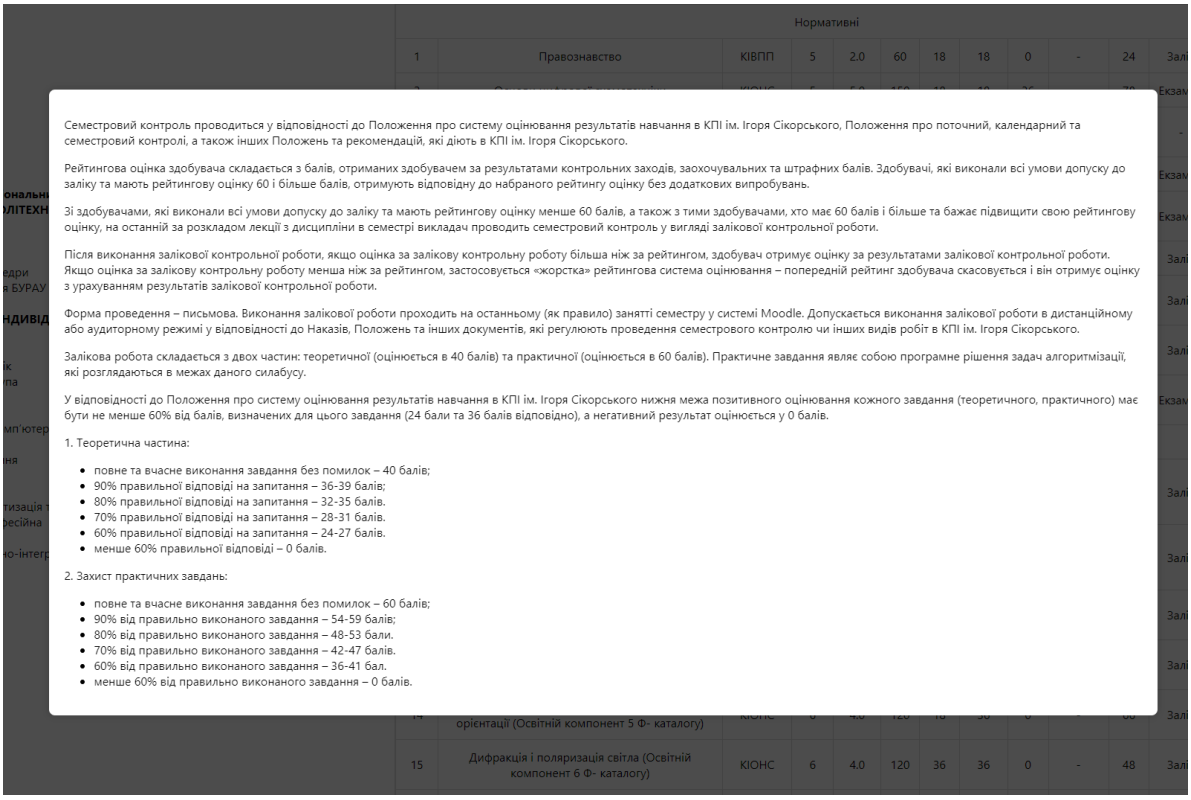


Рис 3.19 - Інформація з модального вікна при натисканні на контрольний захід

Висновки до розділу 3

В даному розділ проведено розроблення баз даних, графічного інтерфейсу проектованого програмного забезпечення та описано огляд інтерфейсу користувача.

Розроблена база даних на основі MongoDB, графічний інтерфейс програмного забезпечення включає в себе таблицю з інформацією про загальну кількість аудиторних годин (лекційні, практичні та лабораторні заняття), загальну кількість годин відведених для самостійної роботи студента, загальну кількість контрольні заходи (екзаменів, заліків, МКР, РГР, РР, ГР, рефератів та загальну інформацію про користувача. Докладно розписано, як користувач може взаємодіяти з розробленою системою.

РОЗДІЛ 4
РОЗРОБКА СТАРТАП ПРОЕКТУ «АВТОМАТИЗОВАНА СИСТЕМА СТВОРЕННЯ СУПРОВІДНИХ ДОКУМЕНТІВ ОСВІТНЬОГО ПРОЦЕСУ»

4.1 Опис та технологічний аудит ідеї стартап-проекту

Після вивчення попередніх розділів, де розглядалися потенційні проблеми у процесі створення освітніх документів, дана робота пропонує інноваційну автоматизовану систему для формування індивідуальних навчальних планів. У цьому розділі проводиться аналіз стартап-проекту, який представляється у вигляді веб-додатку. Ця система поєднує сучасні технології і освітній процес, спрямовуючись на спрощення процесу використання індивідуальних навчальних планів та роблячи їх доступними для всіх зацікавлених сторін.

Реалізація даного проекту полягає у створенні веб-додатка, що дозволить зручно переглядати та взаємодіяти з індивідуальним навчальним планом. Розробка даної системи націлена на підвищення ефективності та якості навчального процесу, забезпечення зручного та інноваційного інструменту для використання індивідуальних навчальних планів, що враховують потреби кожного студента та вимоги освітнього процесу.

Ключові аспекти даної системи включає легкий та зрозумілий веб-інтерфейс. Це сприяє зручності взаємодії користувача з системою, дозволяючи швидко та ефективно користуватися всіма її можливостями. інтерактивність індивідуального плана навчання, що дозволяє користувачам взаємодіяти з ним та отримувати додаткову інформацію Система дозволяє користувачам отримувати актуальну інформацію з будь-якого пристрою, який має Інтернет-підключення. Це розширює зручність користування та забезпечує доступність даних навіть у рухливому середовищі.

У таблиці 4.1 зображено зміст ідеї та можливі базові потенційні ринки.

Таблиця 4.1.

Опис ідеї стартап проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Автоматизована система дозволяє створювати та взаємодіяти з індивідуальним планом навчання, тобто окрім загальної інформацію, користувач отримує усю	Заклади вищої освіти	Студенти в університетах та коледжах можуть використовувати індивідуальні плани для оптимізації свого навчання
докладну інформацію, не переходячи на додаткові ресурси	Онлайн-навчання	Студенти, що обирають навчання в мережі, можуть скористатися індивідуальними планами для адаптації навчального процесу до свого робочого графіку та темпу вивчення
	Корпоративна освіта	Компанії можуть використовувати індивідуальні плани для навчання своїх працівників, сприяючи їхньому професійному розвитку

Отже, пропонується новий спосіб автоматизованої системи для створення та використання індивідуальних навчальних планів, замість використання текстових редакторів. У розробленому додатку є створена таблиця з навчальними дисциплінами та можливістю взаємодіяти з нею, й отримувати усю інформацію в «один клік».

Аналіз потенційних техніко-економічних переваг ідеї порівняно із пропозиціями конкурентів наведено у табл. 4.2.

З нижче наведеної таблиці видно, що перевагами розробленого проекту є вартість програмного забезпечення, Вона є найнижчою на ринку порівняно з аналогічними продуктами конкурентів. Це може привертати увагу нових клієнтів і забезпечувати конкурентну перевагу. Також перевагою є простота використання інтерфейсу користувача, інтуїтивно зрозумілий дизайн сприяє зручному взаємодії

користувача з програмою. Відсутність зайвої інформації допомагає уникнути замішань та забезпечує ефективне використання продукту. Нейтральними сторонами проекту є зручність інтерфейсу, адже в даному проекті, що у конкурентів він є доволі зручним для користувача.

Таблиця 4.2.

Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W (слабка сторона)	N (нейтральна сторона	S (сильна сторона)
		Мій проект	Конкурент1 Microsoft 365	Конкурент2 Microsoft Excel	Конкурент3 Mykpi.ua			
1.	Вартість програмного забезпечення	1500	5000	4000	2000	-	-	+
2.	Зручність використання	Зручний	Зручний	Зручний	Зручний	-	+	-
3.	Інформаційність	Інформаційний	Інформаційний	Неінформаційний	Неінформаційний	-	-	+
4.	Інтерфейс користувача	Простий	Змішаний	Перевантажений	Простий	-	-	+

З вище наведеної таблиці видно, що перевагами розробленого проекту є вартість програмного забезпечення, вона є найнижчою на ринку порівнюючи з конкурентами. Також перевагою є простота використання інтерфейсу користувача, все інтуїтивно зрозуміло, та не має зайвої інформації. Нейтральними сторонами проекту є зручність інтерфейсу, адже в даному проекті, що у конкурентів він є доволі зручним для користувача.

Аудит технологічних рішень представляє собою важлий етап у забезпеченні високої продуктивності та ефективності проєкту створення системи автоматизованої системи супровідної документації. У цьому контексті аудит

включає вибір оптимальних мов програмування та інструментальних засобів, необхідних для реалізації потрібної функціональності.

Також, процес аудиту технологій враховує оцінку доступних бібліотек та фреймворків, які можуть сприяти втіленню ключових аспектів системи. Вибір інструментів, які найкраще відповідають специфіці проєкту, є ключовим завданням для гарантування стабільності, адаптивності та ефективності системи.

Важливими етапами в аудиті технологічних рішень є аналіз можливостей мов програмування, вибір оптимальних інструментів розробки, оцінка відповідних бібліотек та фреймворків. Орієнтація на інструментарій, який найкраще взаємодіє з потребами проєкту, забезпечить не лише технічну ефективність, але й стабільність та гнучкість системи.

Аналіз ключових компонентів для забезпечення технічної відповідності проєкту подано у таблиці 4.3

Проаналізувавши таблицю можна зробити висновок що розроблений проект використовує такі технології, як Node.js, JavaScript, HTML, CSS, React, MongoDB.

MongoDB вказує на використання бази даних NoSQL для зберігання та управління даними, це корисно для проєктів, які вимагають гнучкості та швидкості обробки даних. Використання React дозволяє створювати ефективні та динамічні веб-додатки.

Таблиця 4.3.

Технологічна здійсненність ідеї проекту

№ п/п	Ідея проек-ту	Технології її реалізації	Наявність технологій	Доступність техно-логій
1.	Автоматизована система створення супровідних документів освітнього процесу	Node.js	Наявна	Доступність
		JavaScript	Наявна	Доступність
		HTML	Наявна	Доступність
		CSS	Наявна	Доступність
		React	Наявна	Доступність
		MongoDB	Наявна	Доступність
Обрана технологія реалізації ідеї проекту: Node.js, JavaScript, HTML, CSS, React, MongoDB				

Проаналізувавши таблицю можна зробити висновок що розроблений проект використовує такі технології, як Node.js, JavaScript, HTML, CSS, React, MongoDB.

MongoDB вказує на використання бази даних NoSQL для зберігання та управління даними, це корисно для проєктів, які вимагають гнучкості та швидкості обробки даних. Використання React дозволяє створювати ефективні та динамічні веб-додатки.

4.2 Аналіз ринкових можливостей запуску стартаппроекту

Визначення ринкових можливостей та аналіз ринкових загроз є ключовим етапом при плануванні впровадження проекту на ринок. Цей процес дозволяє стратегічно розробити напрями розвитку проекту, враховуючи особливості ринкового середовища, потреби потенційних клієнтів та конкурентні пропозиції.

Аналіз ринкових можливостей дозволяє ідентифікувати перспективні сегменти ринку, де може бути висока попит на продукт чи послугу проекту. Зрозуміння потреб і бажань цільової аудиторії сприяє налаштуванню проекту для максимального відповідання їхнім очікуванням. З іншого боку, аналіз ринкових загроз дозволяє визначити фактори, які можуть стати для компанії

Спочатку проведемо аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку (табл. 4.4).

Таблиця 4.1.

Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	10
2	Загальний обсяг продаж, грн/ум.од	400 000 грн/ум.од
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Необхідність доступу до мережі інтернет
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	80%

Після ретельного аналізу попередньої характеристики можна визначити, що сучасне ринкове середовище є вельми привабливим для успішної реалізації стартап-проекту. Важливо відзначити низьку кількість суттєвих конкурентів, що створює сприятливі умови для входження на ринок та визначення власної ніші.

Динаміка ринку свідчить про його поступовий ріст, що додатково підсилює потенціал успіху проєкту. На додаток до цього, зафіксована хороша рентабельність свідчить про можливість отримання прибуткових результатів у цьому секторі.

Загалом, наявні умови створюють перспективний контекст для впровадження стартапу, а врахування усіх цих факторів може сприяти успішному розвитку та конкурентоспроможності проєкту на ринковій арені.

Надалі визначаємо потенційні групи клієнтів, їх характеристики, та формуємо орієнтовний перелік вимог до товару для кожної групи (табл. 4.5).

Таблиця 4.2.

Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Зручний спосіб представлення інформації про навчальні програми та індивідуальні плани для студентів.	Освітні заклади	Розгалуженість у потребах: від адміністрації, яка шукає ефективний інструмент управління, до вчителів, що фокусуються на навчальному процесі.	Легкість використання, надійність системи, можливість адаптації під конкретні вимоги закладу.

		Студенти	Студенти шукають зручний інтерфейс та інформативний додаток	Інтуїтивне користування, зрозуміла візуалізація прогресу, доступність для студентів.
		Онлайн-курси	Вимоги до аналітичних можливостей та звітності системи	Зручний доступ до інформації, інструменти для взаємодії з учнем онлайн-курсів

Отже проаналізувавши таблицю, можна зазначити, що цільова аудиторія включає освітні заклади, онлайн курси та студентів. Основні потреби цих груп клієнтів включають вимоги до інноваційних навчальних технологій, ефективного формування індивідуальних планів навчання, а також забезпечення доступності та прозорості у формуванні програм навчання.

Стратегія позиціонування стартап-проекту спрямована на надання продукту, який відповідає вимогам цільової аудиторії. Основні принципи цієї стратегії включають інноваційні технології, персоналізовані рішення, аналітичні інструменти та забезпечення легкості використання.

Аналіз факторів загроз є ключовим етапом в плануванні проекту, особливо в сфері освітньої документації. При використанні нової технології фактори ризики впливають на прийняття та успішність проекту в галузі освіти. Фактори загроз наведені у табл. 4.6.

Таблиця 4.3.

Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1.	Обменежене або відсутнє інтернет - з'єднання	Відсутність доступу до мережі може призвести до втрати зв'язку з користувачами та перерв у роботі онлайн-сервісів.	Встановлення альтернативних інтернет-з'єднань, розробка систем резервного копіювання для збереження даних.

2.	Кіберзагроза	Атаки з боку хакерів, віруси чи інші кіберзагрози можуть викликати витоки даних, порушення безпеки користувачів та завдати шкоди репутації компанії.	Впровадження сучасних засобів кібербезпеки, регулярне оновлення програмного забезпечення, навчання персоналу з питань кібербезпеки.
3.	Конкуренція	Зростання конкуренції може призвести до втрати ринкової частки та зменшення прибутків.	Розробка стратегій маркетингу та інновацій, збільшення якості продукту, підтримка лояльності клієнтів.
4.	Збільшення навантаження на систему	Різке зростання обсягів користування сервісом може спричинити перевантаження серверів та зниження швидкості обробки запитань.	Масштабування інфраструктури, використання хмарних рішень для забезпечення гнучкості та ефективності.
5.	Виникнення системних помилок	Непередбачувані помилки та збої можуть вплинути на надійність системи	Розробка систем моніторингу та автоматичного виявлення помилок, проведення регулярних аудитів

Аналіз таблиці вказує на різноманітні аспекти, які можуть стати потенційними загрозами для проекту. Розглядаючи ці фактори, можна зробити кілька ключових висновків.

Забезпечення високого рівня кібербезпеки є важливим завданням для захисту конфіденційності та цілісності даних. Врахування конкурентного середовища дозволяє адекватно реагувати на зміни на ринку. Фактора інтернет - з'єднання вказує на необхідність врахування альтернативних методів доступу до сервісу для користувачів

Уважне аналізування та врахування зазначених факторів під час розробки системи допоможе знизити ризики та підвищити готовність системи для широкого кола користувачів.

Але поряд із колом загроз існують і певні можливості, використання цих можливостей може виявитися ключовим елементом для успішного розвитку та конкурентоспроможності проекту. Ретельний аналіз цих можливостей стане основою для розробки стратегій, спрямованих на максимізацію переваг (табл. 4.7).

Таблиця 4.4.

Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1.	Технологічний прогрес	Розвиток нових технологій у галузі освіти	Інвестування в дослідження та розробку для вдосконалення продукту, адаптація до нових технологій.
2.	Зростання запиту на освіту	Збільшення кількості освітніх програм та студентів.	Розширення функціональності продукту, збільшення ресурсів для обслуговування зростаючого попиту.
3.	Розширення функціоналу	Можливість впровадження нових функціональних можливостей та інструментів у систему	Постійне оновлення та вдосконалення системи, аналіз запитів користувачів для додавання необхідних функцій
4.	Міжнародна співпраця	Зможуть залучити аудиторію з інших країн та ринків.	Розширення мовних опцій та адаптація до культурних особливостей, реклама
5.	Зміни в споживчих уподобаннях	Зростання інтересу до онлайн-освіти та інтерактивних платформ.	Розробка інноваційних освітніх інструментів, акцент на користувальницькому досвіді та взаємодії.

Виявлені можливості створюють проекту потенціал для успішного розвитку. Зокрема, технологічний прогрес відкриває широкі можливості для вдосконалення та розширення функціоналу системи. Збільшення попиту на

електронні сервіси та можливість глобалізації відкривають нові ринки та забезпечують можливість залучення більшої аудиторії.

Встановлення партнерських відносин може стати стратегічним кроком у розширенні спектру послуг та забезпеченні конкурентних переваг. Правильне використання цих можливостей дозволить ефективно конкурувати на ринку.

Ступеневий аналіз конкуренції на ринку є ключовим етапом стратегічного планування, де проводиться глибоке вивчення особливостей конкурентного середовища. Цей аналіз дозволяє зрозуміти структуру ринку, визначити конкурентів та їхні стратегії та вплив діяльності на компанії . Ступеневий аналіз зазначено у таблиці 4.8.

Таблиця 4.5.

Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Тип конкуренції - чиста конкуренція	Існує величезна кількість конкурентів на ринку.	Пропонування унікальних функціональних можливостей
За рівнем конкурентної боротьби - міжнародний	Існуючі аналоги розроблялися різними країнами	Впровадження локалізації відповідно до мови країни, адаптація до різних культур на ринків
За галузевою ознакою - міжгалузева	Різні галузі представлені на ринку.	Розробка стратегій, які враховують взаємодію між галузями, партнерства для зростання в різних напрямках.
Конкуренція за видами товарів: - товарно-видова	Фокус на конкретних властивостях продукту.	Розробка продукта з високим ступенем відрізненості, позиціонування як унікального рішення на ринку.

За характером конкурентних переваг - цінова	Акцент на конкуренції за цінами.	Управління ціноутворенням, пошук вартісних пропозицій, ефективне управління витратами.
За інтенсивністю - марочна	Акцент на створенні та управлінні брендом.	Інвестування в брендінг, створення позитивного сприйняття та узнаваності бренду.

Ступеневий аналіз конкуренції на ринку надав загальне уявлення про конкурентне середовище та його вплив на розроблювану систему. Аналізуючи особливості конкурентів та їхні стратегії, було виявлено ключові фактори, які впливають на успіх проекту. На основі зібраної інформації можна визначити оптимальні стратегії для ефективної конкуренції та стабільного розвитку.

Після аналізу конкуренції проведемо більш детальний аналіз умов конкуренції в галузі. Аналіз конкуренції визначено у таблиці 4.9.

Таблиця 4.6.

Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Складові аналізу	Miscrosoft 365, Excel, Mykpi.ua	Спеціалізовані розробники та агенції, що пропонують нові функції	Витрати на технології для розробки, а також їхню особливість чи доступність	Очікування замовників від рівня якості продукту або послуги. Рівень впливу бренду на рішення споживачів.	Інші платформи для створення освітніх веб-сайтів, різні додатки для управління навчальними документами
Висновки:	Зазвичай конкуренція в галузі заснована на вартості,	Висока ймовірність появи нових учасників у галузі, які	Постачальники не визначають ключові умови.	Задоволення потреб та очікувань клієнтів є ключовим для	Наявність альтернативних засобів може

Висновки:	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	інноваціях та якості сервісу. Проект повинен постійно удосконалюватися для підтримання конкурентоспроможності	пропонують новаторські рішення. Зростання конкуренції може вимагати акценту на унікальних особливостях та якості продукту.	Вартість технологій є значущою, але вона доступна для використання відносно багатьма учасниками ринку.	забезпечення сталого успіху.	змушувати нас постійно вдосконалювати наш продукт та надавати унікальні переваги для залучення та утримання клієнтів.

Аналіз конкуренції в галузі за М. Портером дозволив глибше зрозуміти структуру та динаміку ринкового сегменту. Виокремлені основні аспекти, такі як ступінь конкурентної боротьби, потенційна загроза нових учасників та вплив постачальників і покупців, надали чіткий образ конкурентного середовища.

На підставі цього аналізу ми зможемо визначити оптимальні стратегії для підтримання конкурентоспроможності. Залучення нових клієнтів, розробка унікальних функцій може стати ключовими напрямками для успішного ведення бізнесу у цьому конкурентному середовищі.

В процесі узагальнення та аналізу отриманих даних визначено ключові фактори, що визначатимуть конкурентоспроможність проекту. Ці фактори охоплюють технічні, стратегічні та репутаційні аспекти, які впливають на ефективність та розвиток продукту. Ці аспекти взаємодіють між собою, утворюючи комплексну основу для конкурентоспроможності проекту Детальне обґрунтування цих факторів буде представлено в таблиці 4.10.

Таблиця 4.7.

Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Інноваційні технології	Застосування передових та ефективних технологій для створення продукту з вищою функціональністю та якістю.
2	Вартість розробки	Здатність здержувати витрати на розробку на конкурентоспроможному рівні, що робить продукт більш доступним.
3	Висока якість продукту	Забезпечення високої якості у всіх аспектах продукту, що привертає клієнтів та збільшує їхню лояльність.
4	Схвалення ринку та репутація	Позитивні відгуки та рейтинги користувачів, а також добра репутація на ринку, що впливає на сприйняття продукту.
5.	Гнучкість та адаптивність	Здатність швидко реагувати на зміни у вимогах ринку та адаптувати продукт до нових тенденцій та потреб клієнтів.

Обґрунтування факторів конкурентоспроможності підкреслило важливість розгляду різноманітних аспектів нашого проекту. Зазначені фактори, такі як інноваційні технології, висока якість продукту вартість розробки, та гнучкість та адаптивність, обрані не випадково, адже вони становлять основні підстави для створення конкурентоздатної системи. В таблиці 4.11 проведено порівняльний аналіз сильних та слабких сторін.

З таблиць 4.10 та 4.11 бачимо, що фактори конкурентоспроможності суттєві та мають великий позитивний внесок при впровадженні нового програмного забезпечення для створення автоматизованої системи супровідної документації, а саме індивідуальних навчальних планів.

Таблиця 4.8.

Порівняльний аналіз сильних та слабких сторін «DUST_METER»

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з METER_Kompany						
			–3	–2	–1	0	+1	+2	+3
1	Інноваційні технології	16					+		

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з METER_Kompany						
			–3	–2	–1	0	+1	+2	+3
2	Вартість розробки	18						+	
3	Висока якість продукту	17						+	
4	Схвалення ринку та репутація	15				+			
5.	Гнучкість та адаптивність	15						+	

Основною перевагою та головним досягненням є висока якість продукту, гнучкість та адаптивність. В таблиці 4.12 визначне SWOT- аналіз стартап-проекту

Таблиця 4.9.

SWOT- аналіз стартап-проекту

Сильні сторони: 1. Вартість програмного забезпечення. 2. Інноваційні технології для створення продукту 3. Попит на зручний інструмент для взаємодії з здобувачами освіти 4. Можливість отримувати інформацію в «один клік»	Слабкі сторони: 1. Залежність від специфічних технологічних рішень, які можуть застаріти. 2. Потреба в постійному оновленні 3. Відсутність реклами
Можливості: 1. Розвиток нових технологій у галузі освіти; 2. Збільшення кількості освітніх програм та студентів; 3. Можливість впровадження нових функціональних можливостей та інструментів у систему; 4. Зможуть залучити аудиторію з інших країн та ринків; 5. Зростання інтересу до онлайн-освіти та інтерактивних платформ.	Загрози: 1. Відсутність доступу до мережі може призвести до втрати зв'язку з користувачами та перерв у роботі онлайн-сервісів; 2. Атаки з боку хакерів, віруси чи інші кіберзагрози можуть викликати витоки даних, порушення безпеки користувачів та завдати шкоди репутації компанії; 3. Непередбачувані помилки та збої можуть вплинути на надійність та продуктивність системи; 4. Різке зростання обсягів користування сервісом може спричинити перевантаження серверів 5. Зростання конкуренції може призвести до втрати ринкової частки та зменшення прибутків;

SWOT-аналіз стартап-проекту дав чіткий огляд внутрішніх сильних і слабких сторін, а також зовнішніх можливостей та загроз. Сильні сторони, такі як інноваційні технології та висока якість продукту, надають конкурентну перевагу на ринку. З іншого боку, слабкі сторони вимагають уваги та вдосконалення для досягнення максимального потенціалу. Можливості, які виявлені в аналізі, відкривають перспективи для розвитку та розширення. Однак потрібно бути уважними до потенційних загроз. Альтернативи ринкового впровадження стартап-проекту визначено у таблиці 4.13.

Таблиця 4.10.

Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Стратегія нейтралізації ринкових загроз сильними сторонами стартапу	Висока	8-12міс
2	Стратегія компенсації слабких сторін стартапу наявними ринковими можливостями	середня	6-8міс
3	Стратегія виходу з ринку	Низька	4міс

З зазначених альтернатив обираємо стратегію компенсації слабких сторін стартапу наявними ринковими можливостями.

4.3 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів(табл. 4.14).

Таблиця 4.11.

Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Освітні заклади	+	+	середня	легка
2	Онлайн-курси	+	+	середня	легка
Які цільові групи обрано: Під час аналізу потенційних груп споживачів було прийнято рішення що компанія буде працювати із освітніми компаніями та закладами.					

За результатами аналізу потенційних груп споживачів було обрано цільові групи для стартап-проекту. Орієнтування на навчальні установи створить попит на продукт серед викладачів та студентів, що сприятиме ширшій імплементації в освітньому секторі. Забезпечення інструментів для ефективного створення та впровадження індивідуальних навчальних планів для провайдерів онлайн-курсів, сприятиме залученню цього ринкового сегменту.

Вибір цільових груп споживачів відповідає стратегії широкого охоплення ринку та сприяє позиціонуванню нашого продукту як універсального рішення для різних сфер освіти та навчання.

Для роботи в обраному сегменті необхідно сформувати базову стратегію розвитку. Визначення базової стратегії розвитку зазначено у таблиці 4.15.

Таблиця 4.12.

Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
	Підсилення сильних сторін стартапу за рахунок	Диференційований маркетинг	Якісний продукт, до якого прихильні споживачі,	Стратегія диференціації

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
	ринкових можливостей		постійний зворотній зв'язок з клієнтами.	

Визначення базової стратегії розвитку є ключовим етапом формування успішного стартап-проекту. Стратегії розвитку продукту та маркетингу націлені на вирізнення унікальних характеристик системи та привертання уваги цільової аудиторії. Вказані стратегія диференціації акцентує на відмінностях та інноваціях, сприяючи унікальності стартапу та його стійкій позиції на ринку. Наступним кроком є вибір стратегії конкурентної поведінки (табл. 4.16).

Таблиця 4.13.

Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
	Так	Компанія буде шукати нових споживачів та працювати над брендом, щоб забрати існуючих споживачів	Копіювати стратегії маркетингу конкурента	Цінова стратегія та стратегія заняття окремої ніші серед конкурентів

Обрана базова стратегія конкурентної поведінки визначає ключові напрямки дій стартапу в умовах конкурентного середовища. Враховуючи фактори першопрохідця на ринку та акцентуючи увагу на розширенні аудиторії, а також інноваціях та унікальних характеристиках продукту, стартап прагне забезпечити

собі конкурентні переваги та стійке позиціонування у вибраному сегменті ринку.

На основі вимог споживачів у вибраному сегменті, їхніх очікувань від постачальника та продукту, а також ураховуючи обрану стратегію розвитку та стратегію конкурентної поведінки, формується стратегія позиціювання. Ця стратегія визначає, яким чином проект буде сприйматися на ринку, намагаючись створити у споживачів конкретне уявлення та ідентифікацію з ним. Визначення стратегії позиціонування було зроблено у таблиці 4.17.

Таблиця 4.14.

Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
	Інформативність та інтуїтивно зрозумілий інтерфейс	Стратегія диференціації	Простота використання та інтуїтивно зрозумілий дизайн	Легкість використання. Зворотній зв'язок із виробником. Індивідуалізація

Результатом даного підрозділу є система рішень щодо ринкової поведінки компанії, вона визначає в якому напрямі буде працювати компанія на ринку.

4.4 Розроблення маркетингової програми та бізнес-моделі стартап-проекту

Під час розроблення маркетингової програми першим кроком є розробка маркетингової концепції товару, який отримає споживач. Маркетингова концепція - це загальна ідея та стратегія, яка лежить в основі позиціонування товару чи послуги на ринку У таблиці 4.18 підсумовуємо результати аналізу конкурентоспроможності товару.

Отже таблиця відображає важливі аспекти, які визначають конкурентні переваги проекту. Враховуючи потреби споживачів, вигоди, які пропонує товар, та ключові переваги перед конкурентами, стає очевидним, що система

інтерактивного створення індивідуальних навчальних планів має унікальність та значущі переваги.

Таблиця 4.15.

Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Оптимізація створення індивідуальних планів	Ефективність та швидкість формування планів	Використання інноваційних технологій, автоматизація процесу, гнучкість та персоналізація.
2	Гнучкість і можливість масштабування системи	Адаптація до різних освітніх потреб, масштабованість	Розширення функціоналу для задоволення різних освітніх потреб, можливість легкого масштабування системи при збільшенні обсягів користувачів та запитань.

Зручність та персоналізація в навчанні, гнучкість системи та можливість масштабування роблять проект привабливим для різних освітніх сфер. Інноваційний підхід та використання сучасних технологій створюють потенціал для успішного виходу на ринок і задоволення потреб користувачів. У таблиці 4.19 було описано трьох рівнів моделі товару.

Таблиця 4.16.

Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Автоматизована система створення супровідних документів освітнього процесу, а саме індивідуальний план навчання.		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1.Довговічність 2. Вартість 3. Якість 4. Інформативність 5. Зручність	Нм М Нм Нм Нм	Тх Е Вр Тл Тл
	Якість: відповідає нормам ДСТУ2499:2017		
	Пакування: -		
	Марка: -		
III. Товар із підкріпленням	До продажу с		
	Після продажу		

Основні властивості продукту, такі як довговічність, вартість, якість, інформативність, зручність відповідають вимогам сучасного ринку. додатковий функціонал системи, такий як гнучкість та можливість масштабування для різних освітніх потреб, а також інтеграція з різними освітніми платформами. Ця багаторівнева модель враховує різні аспекти розробки та функціональності продукту, що сприяє його широкому впровадженню та успішному використанню в освітньому середовищі. Продукт не використовує пакування та маркування бо є електронним продуктом, а фізичні носії наданий час вже є застарілими.

Наступним кроком є визначення цінових меж, якими необхідно керуватися при встановленні ціни на потенційний товар, це передбачає аналіз цін товарів конкурентів, та доходів споживачів продукту (табл. 4.20).

Таблиця 4.17.

Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
	200-500 у.о./рік	250-400 у.о./рік	70000у.о./рік	100-200 у.о./рік

Отже виходячи з таблиці, було визначено рівень цін на товари-аналоги та рівень цін на товари-замінники, також було визначено рівень доходів цільової групи споживачів. Проаналізувавши аспект встановлення ціни, були визначені. Верхня та нижня межі встановлення ціни на товар/послугу 100-200 у.о./рік.

В умовах сучасного бізнес-середовища ефективна система збуту визначає успіх продукту. Аналіз та розуміння потреб цільових клієнтів дозволяє визначити, яким чином їм можна найкраще задовольнити. Вивчення процесу прийняття рішень в закупівельному циклі клієнтів допомагає визначити оптимальні точки втручання та впливу. Також необхідно визначити функції збуту, які має виконувати постачальник товару, зазвичай це посилення зусиль у продажах та рекламі для залучення нових клієнтів, забезпечення високого рівня обслуговування та вирішення проблем клієнтів, Ефективне управління запасами для забезпечення належного наявності товарів. Також необхідно визначити канали збуту, прямий канал збуту це безпосередній контакт з клієнтами, що може

бути важливим для продуктів, де необхідна особиста демонстрація чи консультація. Непрямий збут це використання посередників, таких як роздрібні торговці чи дистриб'ютори, для зростання обсягів продажів. Оптимальна система збуту повинна враховувати унікальність продукту, особливості цільової аудиторії та конкурентне середовище. Система збуту визначена в таблиці 4.21.

Таблиця 4.18.

Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
	Підписки на регулярне отримання програмного продукту	Забезпечення постійного оновлення та підтримки програмного продукту, надання технічної підтримки	Електронний канал збуту, онлайн-магазин	Продаж через власний онлайн-магазин та підписка на регулярне оновлення програм через систему підписок

Враховуючи специфіку закупівельної поведінки цільових клієнтів, що мають великий інтерес до продуктів з категорії програмного забезпечення, а саме автоматизована система створення супровідних документів освітнього процесу та віддають перевагу підпискам на регулярне отримання оновлень, система підписок є ефективним каналом збуту. Постачальник програмного продукту повинен забезпечити стійку технічну підтримку та регулярні оновлення, а електронний канал збуту, такий як онлайн-магазин, є оптимальним для забезпечення доступу до продукту. Така система сприяє зручності та надійності збуту програмного продукту. Концепція маркетингових комунікацій зазначена у таблиці 4.22.

Таблиця 4.19.

Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
	Потреба в зручному та інформативному спосібі представлення інформації про навчальні програми та індивідуальні плани для студентів.	Таргетована реклама в інтернеті, рекомендації від існуючих користувачів	Інформативність Зручність Якість	Показати переваги продукту над конкурентами	Показана інформативність та зручність продукту

Аналіз концепції маркетингових комунікацій, зокрема, специфіка поведінки цільових клієнтів вказує на їхні основні потреби у зручному та інформативному представленні інформації про навчальні програми та індивідуальні плани для студентів.

У контексті цього аналізу обрано ефективні канали комунікацій, такі як таргетована реклама в інтернеті та рекомендації від існуючих користувачів. Обрані ключові позиції для позиціонування — інформативність, зручність та якість, що відображає визначені потреби цільової аудиторії.

Завдання рекламного повідомлення направлені на відзначення переваг продукту над конкурентами, а концепція рекламного звернення підкреслює інформативність та зручність продукту.

Цей комплексний підхід до розробки маркетингових стратегій створює впевненість у тому, що ваш стартап має всі шанси вразити свою цільову аудиторію та стати успішним на ринку.

Розробка бізнес-моделі проекту є критичним етапом в створенні та успішному впровадженні будь-якого нового підприємства чи стартапу. Цей

процес вимагає ретельного аналізу та стратегічного планування для визначення того, як компанія буде створювати, доставляти та забезпечувати цінність для своїх клієнтів, а також як буде генеруватися прибуток.

Бізнес-модель є фундаментальним каркасом, на якому ґрунтується весь бізнес-процес. Вона визначає стратегічний напрямок компанії, визначає її конкурентні переваги та визначає, як вона буде взаємодіяти з ринком та іншими учасниками бізнес-середовища.

Бізнес-модель проекту визначена в таблиці 4.23, та за цією концепцією включає наступні пункти:

1. Ключові партнери. Розглядаються партнерства, які можуть допомогти у виконанні ключових завдань. Це можуть бути постачальники, дистриб'ютори, технологічні партнери та інші стейкхолдери. Також можлива відсутність партнерів.
2. Ключові види діяльності. Основні операційні процеси, які необхідні для виконання бізнес-плану.
3. Ціннісна пропозиція. Визначає, що саме продукт чи послуга пропонує клієнтам. Які проблеми вирішує чи які потреби задовольняє?
4. Відносини з клієнтами. Визначає, як компанія буде взаємодіяти з клієнтами. Це може бути обслуговування клієнтів, підтримка, програми лояльності тощо.
5. Канали збуту. Визначає шляхи, якими продукт чи послуга буде досягати клієнтів. Це може бути продаж через інтернет-магазин, роздрібні точки, дистриб'ютори тощо.
6. Сегменти користувачів. Важливим елементом бізнес-моделі є групи людей або компаній, які будуть клієнтами. Необхідно визначити їх потреби, характеристики та інші аспекти, що впливають на взаємодію.
7. Структура витрат. Оцінюються основні витрати, пов'язані з запуском та утриманням бізнесу. Це включає зарплати, оренду, обладнання, маркетинг, дослідження та інші витрати.

8. Джерела доходів. Визначає, як компанія буде заробляти гроші. Це може бути продаж продукту, підписки, реклама, ліцензії, партнерські угоди тощо.

Таблиця 4.23

Бізнес-модель проекту

<table>
<tr>
<td rowspan="2">Ключові партнери:
Проект розробляється однією людиною</td>
<td rowspan="2">Ключові види діяльності:
Створення автоматизиваної системи індивідуальних планів навчання</td>
<td rowspan="2">Ціннісна пропозиція
Актуальна система з зручним та інтуїтивно зрозумілим інтерфейсом,що забезпечує іноформативність документації</td>
<td>Відносини з клієнтами
Індивідуальний підхід: Постійний контакт з клієнтами для врахування їхніх потреб та адаптації продукту.</td>
<td rowspan="2">Сегменти користувачів:
-Освітні заклади
-Онлайн-курси
Ринок
Освіта</td>
</tr>
<tr>
<td>Канали збуту
Прямий продаж:
Власний інтернет-магазин з можливісттю оформити підписку на продукт</td>
</tr>
<tr>
<td colspan="2">Структура витрат:
- Витрати на сервери, хостинг
- Рекламні кампанії в Інтернеті (Google Ads, Facebook Ads тощо).
- Витрати на інструменти аналітики та відстеження користувачів
- Витрати на інструменти для управління проектом
- Витрати ПО та середовище розробки</td>
<td colspan="3">Джерела доходів:
- Щомісячна підписка на розроблений продукт, яка включає технічну підтримку у робочі дні
- Щорічна підписка на розроблений продукт, яка включає технічну підтримку 24/7.</td>
</tr>
</table>

З розгляду ключових елементів бізнес-моделі, включаючи ключових партнерів, види діяльності, ціннісну пропозицію, відносини з клієнтами, канали збуту, сегменти користувачів, структуру витрат та джерела доходів, стає

зрозумілим, що визначення чітких та збалансованих аспектів кожного з них є важливим для успішної реалізації бізнес-стратегії.

Ефективна бізнес-модель враховує потреби та очікування клієнтів, реалізує конкурентні переваги та максимізує ефективність витрат. Зрозуміння ринкових умов, управління партнерськими відносинами, виокремлення цінностей для клієнтів та здатність адаптуватися до змін у бізнес-середовищі - це основні складові успішної бізнес-моделі.

Висновки до розділу 4

Розділ присвячений розробці стартап-проекту "Автоматизована система створення супровідних документів освітнього процесу". Проаналізувавши всі аспекти, визначено, що проект спрямований на вирішення актуальної проблеми — оптимізації процесу створення супровідних документів у сфері освіти.

Описано цільову аудиторію, яка включає освітні заклади, онлайн курси та корпоративне навчання. Виділені пріоритетні потреби клієнтів, такі як необхідність зручного та інформативного представлення інформації про навчальні програми та індивідуальні плани для студентів.

Проведено аналіз конкурентоспроможності, визначивши прямих та потенційних конкурентів, постачальників, клієнтів та товарів-замінників. Також враховано вимоги та очікування споживачів щодо якості продукту чи послуги.

Ступінь впливу бренду на рішення клієнтів, вартість технологій та їхню унікальність розглянуті як фактори, які можуть визначати конкурентну перевагу проекту.

Детально висвітлені можливості та загрози, які можуть виникнути в ході реалізації стартапу. Визначено стратегії розвитку та конкурентної поведінки, а також варіанти цільових груп потенційних споживачів та їхні вимоги.

На основі цього розділу розроблено інтегровану маркетингову стратегію, яка враховує особливості цільової аудиторії, конкурентний ландшафт та ключові переваги продукту. Зазначено, що проект має потенціал для успіху, якщо будуть враховані визначені чинники та відпрацьовані стратегії.

ВИСНОВКИ

У даній магістерській дисертації було проведено розроблення автоматизованої системи створення супровідної документації, а саме індивідуальних планів навчання, система була реалізована, як веб-додаток.

Було проведено огляд та аналіз сучасних рішень по темі дисертації, визначені переваги й недоліки існуючих систем, також було проведено аналіз наукових робіт, які були проаналізовані, щоб усунути подібні недоліки у розроблювальній системи.

Проаналізувавши літературні джерела та наукові статті була спроектована трьохрівнева архітектура для розробленої системи, також була розроблена багаторівнева архітектура комплексної системи створення супровідної документації.

Після огляду та аналізу літературних джерел було обрано тип додатку, мову програмування, фреймворки, тип бази даних та середовище для розроблення. Прийнято рішення розроблення веб-додатку з використанням мови програмування JavaScript з використанням фреймворку React та бази даних MongoDB сумісно з Node.js.

При розробленні веб-додатку було проведено реалізацію бази даних системи, побудована ER-діаграма взаємозв'язків між сутностями в базах даних. Розроблений графічний інтерфейс додатку та реалізована взаємодія з базою даних силабусів. Детально описана інструкція користувача, а саме як почати роботу з веб-додатком та як взаємодіяти з його інтерфейсом для отримання необхідної інформації.

У перспективі можливим є розширення функціоналу взаємодії користувача з веб-додатком, покращення дизайну та впровадження більшої інтеграції з інфраструктурою університету.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сисоєва С. Вища освіта України: реалії сучасного розвитку / Сисоєва С., Батчеко Н. // Київський університет ім. Бориса Грінченка – 2011. – с. 368.

2. Чепуріна С. Силубас як засіб організаціїї самостійної роботи студентів під час педагогічної практики / С. Чепуріна // Актуальні питання гуманітарних наук: Міжвузівський збірник наукових праць молодих вчених Дрогобицького державного педагогічного університету– 2018. – с. 280.

3. Заколодяжний В.В. Науково-дослідна практика здобувачів ступеня магістра: Організація, завдання та звітування / В.В. Заколодяжний // Навчальний посібник. – Київ: КПІ ім. Ігоря Сікорського, 2023. – с. 14.

4. Третяк О. С. Розробка програмного забезпечення «Графік навчального процесу» на платформі 1С / Вісник студентського наукового товариства / О.С. Третяк, В.С. Фетісов // Ніжинського державного університету імені Миколи Гоголя. – 2012. – С. 70-72

5. Положення про індивідуальний навчальний план здобувача вищої освіти в КПІ ім. Ігоря Сікорського від 01.10.2022 [Електронний ресурс]. – Режим доступу: https://osvita.kpi.ua/sites/default/files/downloads/Pologennia_INP.pdf

6. IBM Automation [Електронний ресурс]. – Режим доступу: https://www.ibm.com/products/document-processing

7. Dealssign - Електронний документообіг: види систем та їхні функції [Електронний ресурс]. – Режим доступу: https://dealssign.com/blog/elektronnij-dokumentoobig-vidi-sistem-ta-yixni-funkciyi/

8. Microsoft – Microsoft Excel [Електронний ресурс]. Режим доступу: https://support.microsoft.com/uk-ua/office/%D0%B7%D0%B0%D0%B3%D0%B0%D0%BB%D1%8C%D0%BD%D1%96-%D0%B2%D1%96%D0%B4%D0%BE%D0%BC%D0%BE%D1%81%D1%82%D1%96-%D0%BF%D1%80%D0%BE-excel-starter-601794a9-b73d-4d04-b2d4-eed4c40f98be

9. Microsoft - Microsoft 365 [Електронний ресурс]. – Режим доступу: https://www.microsoft.com/uk-ua/microsoft-365

10. Освітній процес в КПІ ім.Ігоря Сікорського [Електронний ресурс]. – Режим доступу: https://osvita.kpi.ua/docs

11. Ягелонський університет [Електронний ресурс]. – Режим доступу: https://www.uj.edu.pl/uk_UA

12. Бевз С. В. Розробка автоматизованої системи формування розкладу магістратури / С. В. Бевз, В. В. Войтко, С. М. Бурбело, А. М. Шоботенко // Інформаційні технології та комп'ютерна техніка: Наукові праці ВНТУ. - 2009. - № 4. [Електронний ресурс]. – Режим доступу: https://praci.vntu.edu.ua/index.php/praci/article/view/169/168

13. Мокін В.Б. Система автоматизованого ведення документообігу рад вищих навчальних закладів / В.Б. Мокін, С.М. Бурбело, С.В. Бевз, В.В. Войтко, Н.Ф. Кузьміна. // Інформаційні технології та комп'ютерна техніка: Наукові праці ВНТУ. – 2008. – № 3. – С. 1-6. [Електронниы ресурс]. – Режим доступу: https://praci.vntu.edu.ua/index.php/praci/article/view/75/74

14. Биков В.Ю. Система автоматизованого документообігу в оболонці базового програмного забезпечення / В.Ю. Биков, Т.О. Дерба // Інформаційні технології і засоби навчання: електронне наукове фахове видання [Електронний ресурс] / Гол. ред.: В.Ю. Биков; Ін-т інформ. технологій і засобів навчання АПН України, Центр. ін-т післядиплом. пед. освіти АПН України. – 2007. – № 2(3). – Режим доступу: https://lib.iitta.gov.ua/492/1/Byk+Derb_2_3_2007.pdf

15. Юхимчук С.В. Розробка локальної автоматизованої системи розподілу навантаження в процесі ведення документообігу / С.В. Юхимчук, С. В. Бевз, С. М. Бурбело, О. В. Дмитришин, І.О. Чернова // Вісник Вінницького політехнічного інституту. – 2009. - № 1. – С. 107-111.

16. Гаркуша С.А. Автоматизація облікових процесів: впровадження та переваги роботи системи / С.А. Гаркуша // Вісник Сумського національного аграрного університету. – 2012. – випуск 4. – С. 60-65 [Електронниы ресурс]. – Режим доступу: https://repo.snau.edu.ua/bitstream/123456789/537/3/1074.pdf

17. Хмелюк В.С. Засоби автоматизації генерації електронних документів в системах організаційного управління / В.С. Хмелюк, О.А. Амонс // Проблеми програмування. – 2008. - № 2-3. – С. 641-649 [Електронниы ресурс]. – Режим доступу: http://dspace.nbuv.gov.ua/bitstream/handle/123456789/1497/%e2%84%962-3_2008_Hmeluk.pdf?sequence=1

18. Юхимчук С.В. Комп'ютеризована система управління індивідуальними навчальними планами магістратури / С.В. Юхимчук, С. В. Бевз, С. М. Бурбело, Н.Ф. Кузьміна, С.В. Хрущак // Оптико-електронні інформаційно-енергетичні технології. - 2008. - № 2. - С. 5-8. [Електронниы ресурс]. – Режим доступу: http://dspace.nbuv.gov.ua/bitstream/handle/123456789/32171/01-Yukhimchuk.pdf?sequence=1

19. Haverbeke Marijn Eloquent JavaScript, 3rd Edition / Marijn Haverbeke - San Francisco: No Starch Press, 2012. - p.472.

20. Flanagan David JavaScript: The Definitive Guide / David Flanagan. – Sebastopol: O'Reilly Media, Incorporated, 2011. – p.1078.

21. Banks Alex. Learning React: Functional Web Development with React and Redux / Alex Banks, Eve Porcello. – Sebastopol: O'Reilly Media, 2017. – p.350.

22. Techopedia – A Detailed Look at 3-Tier Sowftware Archirecture [Електронний ресурс]. - Режим доступу: https://www.techopedia.com/2/32100/software/a-detailed-look-at-3-tier-software-architecture

23. Mircosoft Learn – N-tier architecture style [Електронний ресурс]. Режим доступу: https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/n-tier

24. STM - Software Architecture: One-Tier, Two-Tier, Three Tier, N Tier [Електронний ресурс]. - Режим доступу: https://www.softwaretestingmaterial.com/software-architecture/

25. Fowler Martin. Patterns of Enterprise Application Architecture / Martin Fowler. - London: Pearson Education, 2012. – p.560.

26. Martin Robert C. Clean Architecture: A Craftsman's Guide to Software Structure and Design / Robert C. Martin. – London: Pearson Education, 2017. – p.432.

27. Foxminded – «Реляційні бази даних усе, що необхідно про них знати» [Електронний ресурс]. – Режим доступу: https://foxminded.ua/reliatsiini-bazy-danykh/

28. Termin.in.ua – «Сервер – що це таке, як працює і навіщо потрібен. Види, функції та приклади.»[Електронний ресурс]. – Режим доступу: https://termin.in.ua/server/

29. Python Documentation [Електронний ресурс]. – Режим доступу: https://www.python.org/doc/

30. Васильєв О.М. Програмування мовою Java / О.М. Васильєв. – Тернопіль: Bohdan Books – 2022. – с. 696.

31. Java Documentation [Електронний ресурс]. – Режим доступу: https://docs.oracle.com/en/java/

32. Ruby Documentation [Електронний ресурс]. – Режим доступу: https://www.ruby-lang.org/en/documentation/

33. LemonSchool – «Навіщо потрібна мова програмування Ruby?» [Електронний ресурс]. – Режим доступу: https://lemon.school/blog/navishho-potribna-mova-programuvannya-ruby

34. PHP Documentation [Електронний ресурс]. – Режим доступу: https://www.php.net/manual/php3.php

35. Hostinger Tutorials – «What Is PHP? Learning All About the Scripting Language» [Електронний ресурс]. – Режим доступу: https://www.hostinger.com/tutorials/what-is-php/

36. Wiki TNEU – «Недоліки PHP» [Електронний ресурс]. – Режим доступу: http://wiki.tneu.edu.ua/index.php?title=11_%D0%9D%D0%B5%D0%B4%D0%BE%D0%BB%D1%96%D0%BA%D0%B8_PHP#.D0.9D.D0.B5.D0.BC.D0.B0.D1.94_.D1.96.D0.BC.D0.B5.D0.BD

37. C# Documentation [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/csharp/

38. ASP.NET Documentation [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-8.0

39. TutorialsTeacher – Learn LINQ [Електронний ресурс]. – Режим доступу: https://www.tutorialsteacher.com/linq

40. JAVASCRIPT Programming / Neos Thanh // Neos Thanh Бибо – 2021. - 184с.

41. TechTarget – «Node.js» [Електронний ресурс]. – Режим доступу: https://www.techtarget.com/whatis/definition/Nodejs

42. Gackenheimer Cory. Introduction to React / Cory Gackenheimer. – Heidelberg: Apress, 2015. – p.129.

43. Fedosejev Artemij. React.js Essentials / Artemij Fedosejev. – Birmingham: Packt Publishing, 2015. – p. 208.

44. Bampakos Aristeidis. Aristeidis Bampakos. Angular Projects / Aristeidis Bampakos, Mark Thompson. - Birmingham: Packt Publishing, 2023. – p.312.

45. Angular Documentation [Електронний ресурс]. – Режим доступу: https://angular.io/guide/architecture-services

46. Vue.js [Електронний ресурс]. – Режим доступу: https://vuejs.org/

47. Hanchett Erik Vue.js in Action / Erik Hanchett, Ben Listwon. – New York: Manning, 2018. – p.304.

48. Sublime Text Documentation [Електронний ресурс]. – Режим доступу: https://www.sublimetext.com/docs/

49. Visual Studio Code Documentation [Електронний ресурс]. – Режим доступу: https://code.visualstudio.com/docs

50. JetBrains - JavaScript [Електронний ресурс]. – Режим доступу: https://www.jetbrains.com/webstorm/

51. MobgoDB Documentation [Електронний ресурс]. – Режим доступу: https://www.mongodb.com/docs/drivers/node/current/

52. Microsoft – «Посібник зі зв’язків між таблицями» [Електронний ресурс]. – Режим доступу: https://support.microsoft.com/uk-ua/topic/%D0%BF%D0%BE%D1%81%D1%96%D0%B1%D0%BD%D0%B8%D0%

BA-%D0%B7%D1%96-%D0%B7%D0%B2-%D1%8F%D0%B7%D0%BA%D1%96%D0%B2-%D0%BC%D1%96%D0%B6-%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D1%8F%D0%BC%D0%B8-30446197-4fbe-457b-b992-2f6fb812b58f