

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ВЕБСАЙТУ ДЛЯ ФОРМУВАННЯ
ІНДИВІДУАЛЬНОГО НАВЧАЛЬНОГО ПЛАНУ СТУДЕНТА
WEBSITE DEVELOPMENT FOR AN INDIVIDUAL
STUDENT CURRICULUM FORMATION

Виконав: студент(ка) 4 курсу, групи КНТ-119сп
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Щербакова О.С.

(прізвище та ініціали)

Керівник Каплієнко ТІ.

(прізвище та ініціали)

Рецензент Шитікова О.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Щербакової Ольги Сергіївни

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка вебсайту для формування індивідуального навчального плану студента. Website Development for an Individual Student Curriculum Formation

керівник проєкту (роботи) Каплієнко Тетяна Ігорівна, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “ 27 ” квітня 2022 року № 102

2. Строк подання студентом проєкту (роботи) 7 червня 2022 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Каплієнко Т.І., доцент		
Нормоконтроль	Камінська Ж.К., асистент		

7. Дата видачі завдання “ 20 ” лютого 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Щербакова О.С.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Каплієнко Т.І.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
88 с., 50 рис., 15 табл., 4 дод., 24 джерел.

КАБІНЕТ СТУДЕНТА, НАВЧАЛЬНИЙ ПЛАН, ІНДИВІДУАЛЬНИЙ ПЛАН, ВИБІР ДИСЦИПЛІН, ВЕБСАЙТ, СТУДЕНТ, НОРМАТИВНА ЧАСТИНА, ВИБІРКОВА ЧАСТИНА.

Об'єкт дослідження – процес формування індивідуального навчального плану студента та процес обрання вибіркового дисциплін.

Предмет дослідження – програмне забезпечення для формування навчального плану.

Метою роботи є розробка програмного забезпечення у вигляді вебсайту для формування навчального плану.

Матеріали, методи та технічні засоби: мова програмування JavaScript, бібліотека React, Node.JS Express, система управління базами даних MySQL.

Результати. Розроблено програмне забезпечення у вигляді вебсайту для формування навчального плану.

Практична цінність роботи полягає у розробці програмного забезпечення для формування навчального плану.

Висновки. Розроблено програмне забезпечення у вигляді вебсайту для формування навчального плану.

Галузь використання – навчальні організації, студенти, викладачі.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 88 pages, 50 figures, 15 tables, 4 appendixes, 24 sources.

STUDENT ROOM, DISCIPLINES, WEBSITE, STUDENT, CURRICULUM, SELECTIVE DISCIPLINE, NORMATIVE DISCIPLINE.

The object of research is the process of forming the student's individual curriculum and selecting elective courses.

The subject of the research is software for forming the individual educational plan.

The purpose of this work is to develop software in the form of a website for forming the individual educational plan.

Materials, methods and technical tools: JavaScript programming language, React library, Node.JS Express, MySQL database management system.

Results. The software tools in the form of a website for students and teachers have been developed.

The practical value of the work lies in the development of software to inform users about the individual educational plan.

Conclusions. The software in the form of a website for educational institutions is developed.

Scope of use – educational organization, students, teachers.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ	7
ВСТУП.....	9
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	10
1.1 Предметна область та основні поняття.....	10
1.2 Актуальність роботи	11
1.3 Дослідження існуючих аналогів.....	11
1.4 Постановка завдання.....	16
1.5 Висновки за розділом 1	18
2 МАТЕРІАЛИ І МЕТОДИ	19
2.1 Вибір мов програмування для розробки програмного забезпечення	19
2.2 Обґрунтування вибору програмної надбудови на стороні клієнта.....	19
2.3 Обґрунтування вибору програмної надбудови на стороні сервера	21
2.4 Вибір застосування рішень для виконання серверних запитів	22
2.5 Вибір середовища розробки.....	23
2.6 Система керування базами даних.....	25
2.7 Допоміжні засоби у розробці проєкту	25
2.8 Архітектура проєкту	27
2.9 Висновки за другим розділом	27
3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ	29
3.1 Структура програмного забезпечення	29
3.2 Функціонування програмного забезпечення.....	30
3.3 Алгоритм функціонування розробленої програми.....	33
3.3.1 Алгоритм авторизації	33
3.3.2 Алгоритм запиту на вибір вибіркового дисциплін	33

3.4	Проектування бази даних	34
3.5	Проектування дизайну	37
3.6	Програмна реалізація	41
3.7	Висновки за розділом 3	47
4	ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ	48
4.1	Опис застосування програми	48
4.2	Характеристики програми	49
4.3	Інструкція по експлуатації програми	49
4.4	Тестування та результати роботи програмного забезпечення	50
4.5	Висновки за розділом 4	53
	ВИСНОВКИ	54
	ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	55
	ДОДАТОК А Технічне завдання	58
	ДОДАТОК Б Опис програми	62
	ДОДАТОК В Текст програми	65
	ДОДАТОК Д Слайди презентації	80

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

JS – Javascript;

npm – node package manager

БД – база даних;

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп'ютер;

СКБД – система керування базами даних;

ВСТУП

За останні роки популярність, розвиток та значимість онлайн навчання стала як ніколи дуже впливовим, поширеним та доступним чинником в нашому житті. Користування інтернетом надає необмежені можливості для саморозвитку, удосконалення умінь та навичок.

Також онлайн система надає певних переваг для викладачів, оскільки завдяки користуванню інтернету, надається змога оптимізувати систему та відслідковувати усе автоматично.

Через це було вирішено дослідити можливість створення вебресурсу для організації навчального плану, зокрема самостійний вибір навчальних дисциплін.

Програма буде працювати наступним чином:

- студент заходить до свого профілю;
- передивляється перелік дисципліни, які діляться на дві категорії: нормативні, які є обов'язковими, і вибіркові;
- студент може вибрати зі списку вибіркові дисципліни таким чином, щоб максимальна кількість кредитів складала 30 на семестр;
- після того як дисципліни обрані, треба підтвердити свій вибір та надіслати запит на підтвердження;
- авторизований користувач з роллю адміністратор має всі права на перегляд дисциплін, груп, студентів. Адміністратор відповідає за запити вибірових дисциплін.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Предметна область та основні поняття

Студент – особа, яка навчається у закладі вищої освіти з метою здобуття певних освітнього та освітньо-кваліфікаційного рівнів.

Навчальна дисципліна – це система знань, умінь і навичок, відібраних з певної галузі наукової, технічної, художньої, виробничої діяльності для навчання в навчальному закладі.

Навчальні дисципліни за вибором студента – це дисципліни, які вводяться закладом вищої освіти для більш повного задоволення освітніх і кваліфікаційних вимог особи для потреб суспільства, ефективного використання можливостей закладу освіти, врахування регіональних особливостей тощо. Вибіркові навчальні дисципліни професійної та практичної підготовки надають можливість здійснення поглибленої підготовки за спеціальностями і спеціалізаціями, що визначають характер майбутньої діяльності; сприяють академічній мобільності студента та його особистим інтересам, дозволяють здійснювати впровадження спеціалізацій у межах базової спеціальності з метою формування компетентності здобувача відповідно до вимог ринку праці.

Навчальний семестр – це половина навчального плану на рік. Завершується семестр іспитами. Між семестрами встановлюються канікули. Якщо предмет вивчається більше одного семестру, семестр ділиться на завершені частини.

Екзаменаційна сесія – це період, за який підбиваються підсумки навчальних досягнень студентів за семестр.

Залік – проводиться під час залікового тижня, ставиться оцінка у разі виконання певних видів робіт, таких як, практичні, лабораторні, семінари.

Контрольні роботи – це один із видів перевірки набутих знань.

Практичні роботи – це вид заняття, який спрямований на закріплення набутих знань та для формування нових умінь [1].

1.2 Актуальність роботи

Основною метою дипломної роботи є створення вебресурсу, який буде допомагати студентам в керуванні навчальним планом, також він допоможе спростити процес обрання вибіркових навчальних дисциплін.

На сьогоднішній день дистанційне навчання є дуже поширеним і саме тому створення ресурсу для формування індивідуального навчального плану є дуже затребуваним. Це вирішить проблему подачі запиту на вивчення дисципліни, для цього не потрібно бути присутнім у закладі вищої освіти та писати заяву. Необхідно мати з'єднання з Інтернетом та бути студентом навчального закладу.

1.3 Дослідження існуючих аналогів

Першим аналогом є Google Classroom (рис. 1.1) – це безкоштовна онлайн платформа, яка допомагає керувати навчанням, як для вчителів, так і для студентів. Для викладачів це зручний спосіб створити унікальний клас для кожної групи, додати завдання, виставити оцінки та вказати строки виконання робіт. Зручний спосіб перегляду всіх студентів групи. Долучити студентів до онлайн класу можна вручну або за допомогою облікового запису Google, при створенні класу створюється код, за допомогою якого студенти зможуть приєднатися до класу.

Classroom є зручним для створення завдання, де можна налаштувати терміни здачі завдання, вказати назву та описати завдання, завантажити файли будь-якого типу, то може бути документ (doc, docx), презентація (pptx, pptm), pdf-файли, відео, тощо.

Для студентів це зручно тим, що відкривши завдання можна одразу побачити текст до завдання, термін на виконання роботи, кількість балів, місце завантаження файлів та коментарі.

Студенти можуть завантажувати свою роботу, як з персонального комп'ютера (ПК), так і з Google Drive, і так само файли можуть бути різних форматів, документи, таблиці, презентації.

Також у разі чого є коментарі, є власні, де повідомлення бачать тільки студент та викладач, та є загальні, де одразу всі студенти можуть писати та ставити свої запитання.

Google Classroom підтримується різними системами, для того щоб отримати доступ можна увійти за допомогою ПК через браузер чи через мобільний телефон, встановивши собі додаток для Android або для iOS [2].

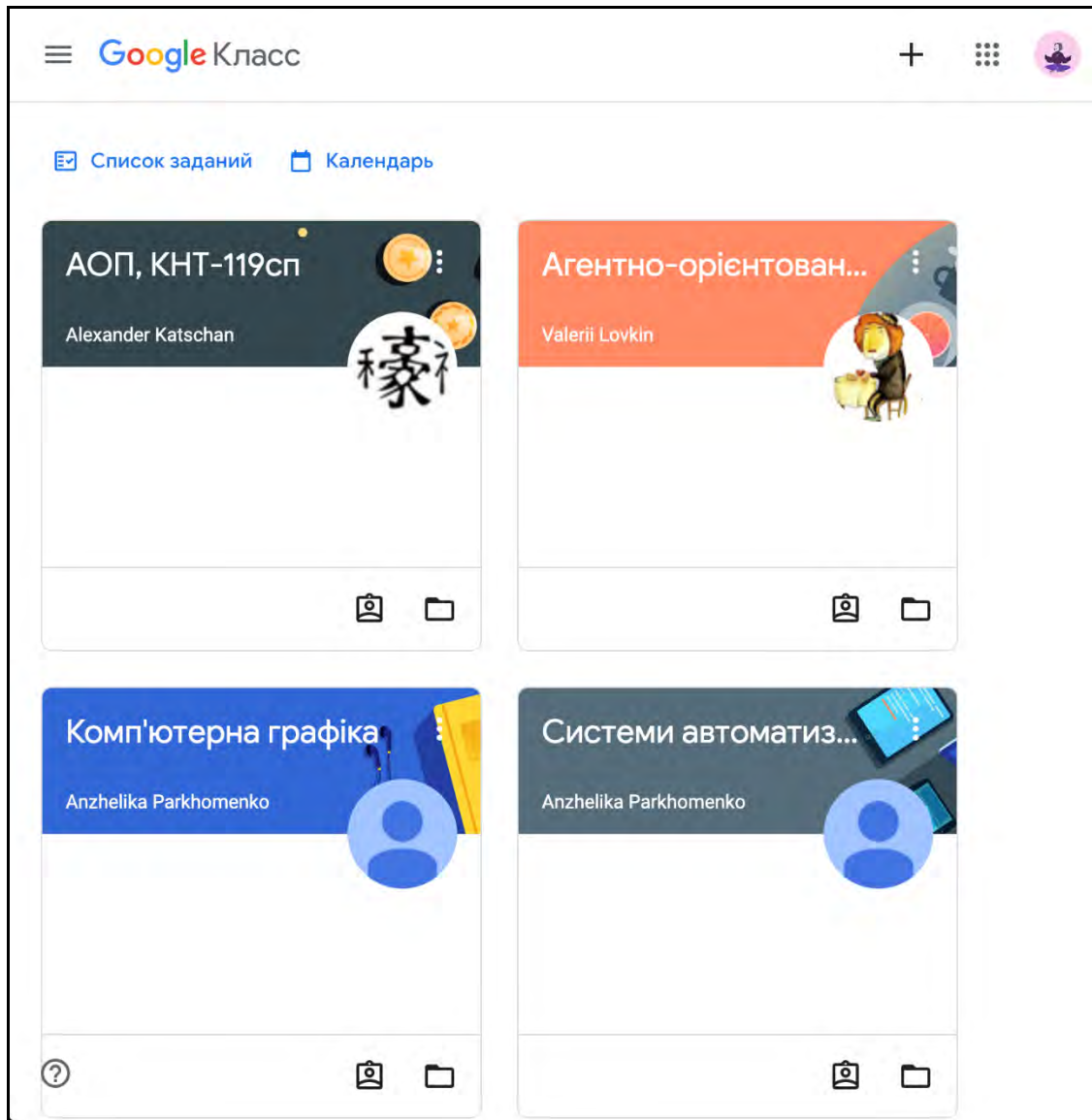


Рисунок 1.1 – Головний екран Classroom

Наступний аналог це Moodle (Modular Object-Oriented Dynamic Learning Environment) – це модульне об'єктно-орієнтоване динамічне навчальне середовище. Це саме платформа для навчання, орієнтована на навчальні заклади.

Сервіс надає можливості створювати користувачів за ролями та правами доступу, формувати навчальні середовища. Викладач може контролювати доступ до курсів, створювати власні системи оцінювання, створювати тести.

Тести можуть бути:

- з однією відповіддю;
- з множинним вибором відповідей;

- встановлення відповідності;
- інші питання.

Тести перевіряються автоматично та після перевірки виставляються до журналу оцінок [3].

Головна сторінка Moodle представлена на рис 1.2.

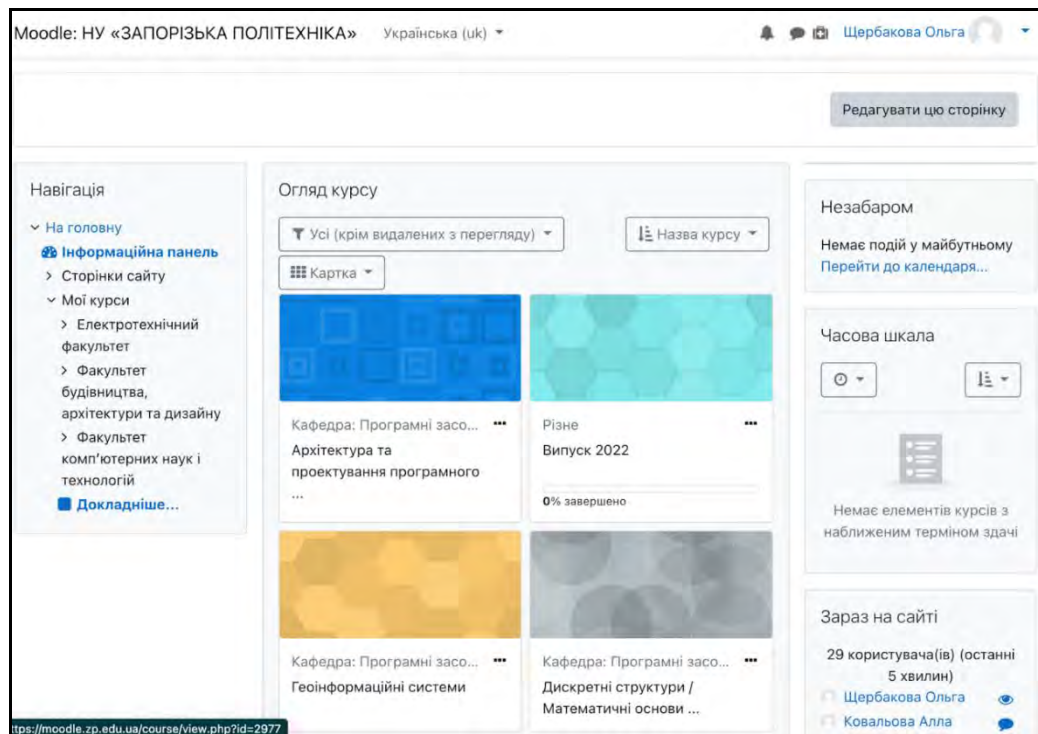


Рисунок 1.2 – Головний екран Moodle

Третім аналогом є SMLS (School Management & Learning System) – це система управління школою та навчанням для шкіл. Цей додаток є платним та він використовується для повноцінного ведення навчання (рис 1.3).

Вся інформація про навчання відображається у “Щоденник”, учень може отримати інформацію про навчальний план:

- розклад дисциплін;
- навчальний графік;
- викладач предмету;
- навчальні та допоміжні ресурси;
- домашнє завдання;

- оцінки;
- медична карта;
- меню їжі.

У ролі викладача надаються такі можливості:

- будувати навчальний план;
- створювати завдання та оцінювати їх;
- робити необхідні оголошення;
- вести журнал оцінок та відмічати присутніх на уроці;
- передивлятися звіти за оцінками та за відвідуваннями;
- створювати онлайн тести.

Також у цьому ресурсі є роль - “Батьки”. Вони мають змогу передивлятися всю інформацію про свою дитину. Батьки інформовані про домашні завдання, про оцінки, мають доступ до шкільної медичної карти дитини, можуть отримувати інформацію про маршрут шкільного автобуса та слідкувати за ним [4-5].

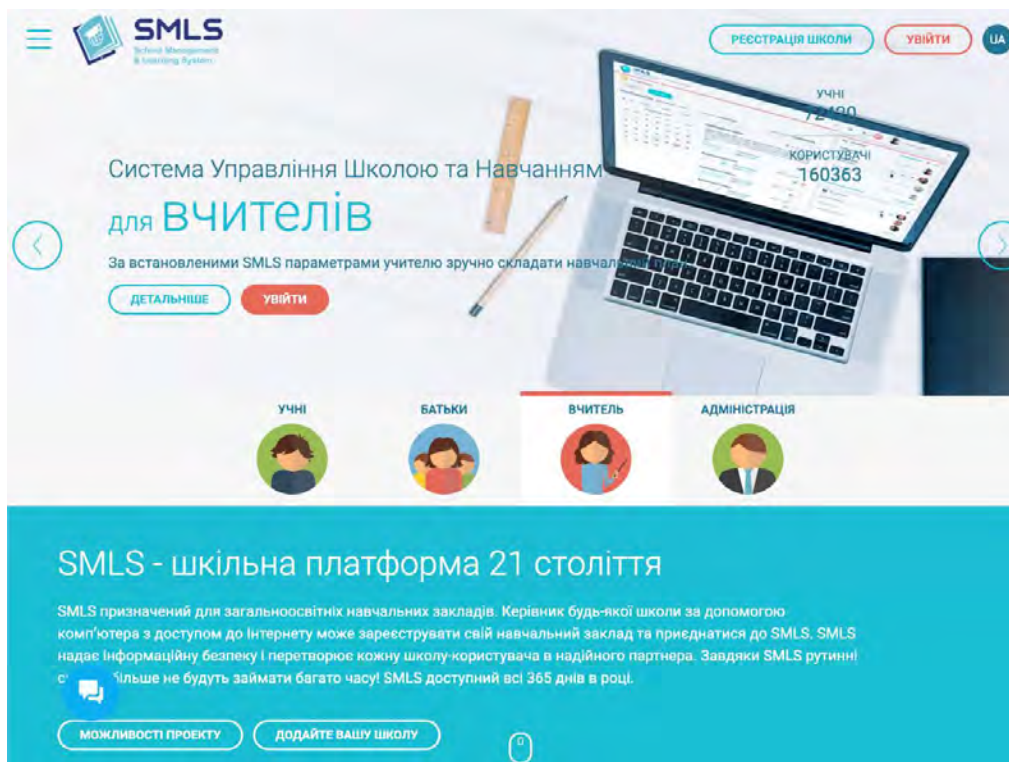


Рисунок 1.3 – Головний екран SMLS

У таблиці 1.1 наведено порівняльну характеристику існуючих аналогів.

Таблиця 1.1 – Порівняльна характеристика аналогів

Критерій порівняння	Google Classroom	Moodle	SMLS
Безкоштовно	+	+	–
Журнал оцінок	+	+	+
Формування звітів	–	–	+
Формування індивідуального навчального плану	–	–	–

З порівняння видно, що ці вебресурси є корисними, як для студентів, так і для викладачів. Але ці програмні ресурси не надають можливості обирати дисципліни та формувати власний навчальний план для студента.

Виходячи з цього було вирішено створити свій власний вебресурс, який надасть можливості для створення навчального плану.

1.4 Постановка завдання

Після проведення аналізу предметної області та існуючих аналогів, для вирішення проблеми, сформулюємо постановку задачі.

Основною метою розробки вебсайту є створення електронного кабінету для студента та адміністрування сайту для оновлення інформації та отримання зведеної статистики. Система повинна здійснювати певні вимоги. До ключових функціональних вимог відносяться:

- автентифікація в системі;

- для студентів перегляд списків з дисциплінами, навчального плану, обрання вибіркового дисциплін;
- для адміністратора перегляд списків з дисциплінами, з студентами, з групами, з викладачами;
- виведення даних за певними критеріями;

Якщо користувач не зареєстрований у системі, то він не має доступу до будь-якої інформації про дисципліни, студентів та викладачів.

Авторизований користувач у ролі Студент повинен мати наступні можливості:

- авторизація у системі;
- перегляд дисциплін;
- фільтрація дисциплін;
- формування навчальних вибіркового дисциплін;
- запит на прийняття вибору дисциплін;
- перегляд навчального плану, як нормативні дисципліни так і вибірково.

Авторизований користувач у ролі Адміністратор повинен мати наступні можливості:

- авторизація у системі;
- перегляд списків усіх дисциплін, студентів;
- фільтр по спискам;
- прийняття або відхилення запитів від студентів.

Авторизований користувач у ролі Декан може бачити списки лише за фільтром власного факультету.

Вимоги до інтерфейсу вебресурсу:

- простота та зрозумілість;
- кольори в одному стилі, без яскравих відтінків;
- гнучкість інтерфейсу;
- навігація по вебсайту.

1.5 Висновки за розділом 1

У першому розділі було розглянуто предметну область, визначені основні поняття щодо навчального плану, визначено актуальність роботи.

Розглянуто три поширених аналоги Moodle, Google Classroom, SMLS. Проведено порівняльну характеристику аналогів системи.

Було зазначено постановку задачі, сформульовані функціональні вимоги, вимоги до інтерфейсу та зазначено системні вимоги.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мов програмування для розробки програмного забезпечення

Для розробки застосунку було обрано використовувати мову програмування JavaScript.

JavaScript – являє собою динамічну та об’єктно-орієнтовану прототипну мову програмування. JS може бути застосовано для програмування, як на клієнті, так і на сервері, для комп’ютерних та мобільних застосунків, для сценаріїв в прикладних програмах, PDF-документів та для інтерактивності вебсторінок.

Через те що, всі основні браузері підтримують компіляцію JIT – це означає, що немає необхідності компілювати код перед його запуском.

JavaScript має дуже високу популярність, оскільки можна застосовувати, як на стороні сервера так і на стороні клієнта. І саме тому JS дуже поширений і буде легко знайти різні ресурси для вивчення цієї мови або для знаходження відповіді на будь-яке питання чи проблему.

Також перевагою є те, що JS працює на стороні клієнта, і саме цим він розвантажує навантаження на сервер, зменшуючи попити на сервер [6].

2.2 Обґрунтування вибору програмної надбудови на стороні клієнта

Для створення клієнтської частини був вибір між бібліотекою React JS та фреймворком Vue.

Першим буде розглянуто бібліотеку React JS, яка використовується для рендерингу візуальної складової програми, та яка була обрана для розробки інтерфейсу користувача.

React використовує розширення синтаксису JS і це – JSX, він дозволяє синтаксису HTML та інших тегів у коді. Вирази JSX після компіляції

становляться звичайними викликами функцій JS і обчислюються в об'єкт JS. Це означає, що є можливість використовувати оператори `if` і `for` в JSX, щоб призначати змінним, приймати як аргументи та повертати з функцій.

React використовує компонентний підхід, віртуальний DOM, модульність.

Ізольованою частиною інтерфейсу називають компонент, користувацький інтерфейс розділений на частини які є різними компонентами. React компоненти мають наступний життєвий цикл - ініціалізація, оновлення стану і знищення.

Віртуальний DOM (Document Object Model). DOM складається з двох частин — об'єктне представлення HTML-документа та API для керування цим об'єктом. Сам документ можна уявити у вигляді дерева DOM.

Virtual DOM створений, щоб постійне оновлення DOM стало більш продуктивним. Сам по собі Virtual DOM – це копія звичайного DOM, він має такі самі властивості.

Використання віртуального DOM в React застосовується для того, аби при повторному відображенні компонентів, оновлювати лише ті частини, які змінилися [7]-[8].

Наступним буде розглянуто Vue, який являє собою JS-фреймворк, та для створення інтерфейсів користувача застосовує шаблон MVVM. Vue використовує реактивне програмування – це означає, що програмування побудоване на потоках даних та є здатність виразити динамічні і статичні потоки даних.

Архітектура Vue побудована на основі компонентів, це означає, що код можна розбити на незалежні частини та повторно використовувати у подальшій розробці, тим самим уникаючи дублювання коду.

Vue відображає дані в об'єктній моделі документа, застосовуючи синтаксис шаблонів, які можна писати використовуючи JS, HTML або розширення синтаксису на основі XML – JSX [9].

У табл. 2.1 буде наведено порівняльну характеристику надбудов.

Таблиця 2.1 – Порівняльна таблиця надбудови на стороні клієнта

Критерії	React	Vue
Virtual DOM	+	+
TypeScript підтримка	++	+
Проста міграція версій	+	+
Легка масштабованість	+	–
Маршрутизація	+	+
Повторне використання компонентів	+	+
Спільнота та підтримка	++	+
Документація	+	++
Вибір гнучких бібліотек	+	–

2.3 Обґрунтування вибору програмної надбудови на стороні сервера

Node.js – це серверна платформа, саме середовище виконання коду, використовує мову JS, як інтерпретаційну мову для обробки запиту або відповіді на стороні сервера. Це прискорює завантаження та зменшує навантаження на центральний процесор.

Також Node.js доступний для Windows, Linux та MacOS і підтримується багатьма провайдерами хостингу. Node.js є легковажним і ефективним завдяки тому, що він використовує подійно-орієнтовану модель і неблокуючу введення/виведення архітектуру, це означає що головний потік не буде блокуватися операціями введення/виведення і сервер буде продовжувати обробляти запити. Оскільки, JS являється однопоточним, то з цього виходить, що код буде асинхронний [10]-[11].

Переваги використання Node.js

- підтримка JavaScript;
- створення швидких, масштабованих сервісів;

- рушій V8;
- використання event loop;
- використання node package manager.

Для розгляду та порівняння вибору фреймворку для створення серверної частини, було вирішено використовувати два фреймворки Express and NestJS. У таблиці 2.2 наведено порівняльну характеристику.

Express – це мінімальний і гнучкий фреймворк, який надає потужний набір функцій для додатків. Запуск коду є простим і швидким. Він надає набір функцій для веб та мобільних додатків та інструменти, які є необхідними для створення застосунку, та також дозволяє повторне використання коду [12].

NestJS є прогресивним фреймворком Node.js. Його використовують для масштабованих додатків на серверній стороні. Через реалізацію шаблону MVC (Model-View-Controller) забезпечується розширюваність проєкту. Особливістю фреймворку є вбудована підтримка TypeScript, яка забезпечує статичну перевірку типів даних [13].

Таблиця 2.2 – Порівняльна характеристика надбудови на стороні сервера

Критерії	Express JS	NestJS
Гнучкий	+	+
Складна структура	–	+
Легкість використання	+	–
Високошвидкісні операції введення-виводу	+	+

2.4 Вибір застосування рішень для виконання серверних запитів

Для виконання запитів на клієнті є два рішення, які можуть застосовуватися, це Axios і Fetch. У таблиці 2.3 наведена порівняльна характеристика.

Першим буде розглянуто Axios.

Axios – це бібліотека JS для з'язку із сервером та для створення HTTP-запитів (GET, POST, PUT, DELETE). Бібліотека підтримує API Promise, який застосовують для гарантування захисту на стороні клієнта від XSRF та для перехватів HTTP-запитів і відповідей. Також може бути використаний для скасування запиту [14].

Axios має такі переваги:

- автоматичне перетворення даних JSON;
- приймає URL-адреси в об'єкті запиту;
- сумісність з браузером;
- простота встановлення таймауту;
- індикатор прогресу завантаження.

Далі буде розглянуто Fetch.

Fetch дозволяє робити HTTP-запити до серверів із браузерів і оброблювати відповіді від серверу. Метод `fetch()`, як тільки починає процес отримання ресурсу з мережі, він повертає `promise` (обіцянка), яка може бути виконана тільки після того, як результат буде доступний [15].

Таблиця 2.3 – Порівняльна характеристика застосувань для запитів

Критерії	Axios	Fetch
Захист XSRF	+	–
Можливість скасувати запит	+	–
Підтримка браузерів	+	–
Автоматичне перетворення даних у JSON	+	–

2.5 Вибір середовища розробки

На сьогодні існує достатня кількість різноманітних редакторів коду. Далі буде опис та порівняння трьох відомих редакторів, це Visual Studio Code (VS Code), PhpStorm, Atom.

VS Code – це стильний та сучасний редактор вихідного коду, який саме застосовується для написання коду. Цей текстовий редактор має такі функції, як можливість встановлення різних допоміжних плагінів для роботи, налаштування інтерфейсу, підтримувана система контролю версій [16].

WebStorm – це інтегроване середовище розробки коду, який підтримує завантаження розроблюваних програм у node.js. Ще однією підтримкою середовища є редагування проєктів на JS, як на стороні сервера, так і в браузері.

WebStorm впроваджує можливість для редагування файлів та автоматично зберігати і синхронізувати їх при зберіганні.

Також він підтримує ES6, має автоматичне оновлення програм node.js, є вдосконалення коду при написанні стилів у CSS, дозволяє автоматично визначати та вибирати конфігурації стилю коду та надає перевірку на стилістику коду [17]-[18].

Atom – це текстовий редактор, який має відкритий вихідний код. Розробником застосунку є GitHub, і саме тому він має інтеграцію з git. За допомогою вбудованого менеджера пакунків (наприклад: npm, yarn) є можливість додавати нові функції до системи.

Порівняльна характеристика редакторів коду у таблиці 2.4.

Таблиця 2.4 – Порівняльна характеристика редакторів коду

Критерії	VS Code	WebStorm	Atom
Безкоштовний	+	–	+
Багатоплатформність	+	+	+
Автодоповнення	+	+	+
Підсвітка коду	+	+	+
Продуктивність	++	–	+
Інтеграція git	+	+	+
Навігація по коду	+	+	+
Вбудований термінал	+	+	+
Легкість програми	+	–	+
Зовнішній вигляд	+	–	–

2.6 Система керування базами даних

Система керування базами даних (або СКБД) по суті є комп'ютеризована система зберігання даних. Користувачам системи надаються можливості для створення, збереження, оновлення та пошуку інформації. СКБД класифікуються відповідно до їх структур даних або типів.

У ході виконання роботи було використано MySQL, оскільки це доступна СКБД, яка підтримується різними системами. Її легко встановити та налаштувати, це є плюсом у використанні. Також перевагою є те, що СКБД працює з великими обсягами даних [19]-[21].

У таблиці 2.5 наведено основні критерії, які забезпечує СКБД.

Таблиця 2.5 – Критерії, які забезпечує MySQL

Критерії	Опис
Безпека	Mysql гарантує захищеність системи управління даними, і цьому є підтвердження, що цим користуються найпопулярніші вебдодатки такі як Youtube, Twitter, Facebook, WordPress.
Масштабованість	Перевага в тому що можна використовувати на будь якій операційній системі, це може бути як Windows, Linux або MacOS
Продуктивність	Обробка мільйонів запитів за використанням високошвидкісного процесу транзакцій
Невибагливість	Використання MySQL з низькими технічними характеристиками. Та процес налагодження є простим

2.7 Допоміжні засоби у розробці проєкту

Postman – інструмент для тестування API. Платформа надає можливість перевірити працездатність API на сервері.

За допомогою Postman, використовуючи метод GET можна перевірити як спрацьовує запит на отримання, наприклад всіх студентів, запит буде записаний наступний чином: `http://localhost:8080/api/students`, в разі успішного виконання буде виведено код статусу 200, що означає запит успішно виконано.

Далі буде наведена таблиця 2.6 з прикладами API для створення студента, для отримання усіх студентів і лише одного, для видалення та для оновлення інформації про студента.

Таблиця 2.6 – Приклад використання API з описом методів

Метод	Приклад	Опис
POST	<code>http://localhost:8080/api/students</code>	API для створення нового студента
GET	<code>http://localhost:8080/api/students</code>	Отримує усі дані з наведеного API, тобто буде інформація про всіх студентів, які містяться в цьому API
GET	<code>http://localhost:8080/api/students/id</code>	Отримує дані лише про одного студента за переданим id параметром
PUT	<code>http://localhost:8080/api/students/id</code>	Оновлює дані лише того студента, id якого було передано як параметр
DELETE	<code>http://localhost:8080/api/students/id</code>	Видаляє лише одного студента з бази по переданому id

При розробці проєкту було використано GitHub – онлайн-платформу розробки програмного забезпечення, яка надає можливості зберігати, відстежувати та взаємодіяти з проєктами. Це дозволяє розробникам завантажувати файли коду та співпрацювати з іншими розробниками над проєктами з відкритим кодом. Також можна відновити код у разі збою або перейти до попередніх версій [22].

2.8 Архітектура проєкту

Архітектура проєкту являє собою REST архітектуру – це стиль програмного забезпечення (ПЗ), який встановлює набір правил, які в подальшому будуть використовуватися для створення програми.

Більш детально, REST (Representational State Transfer), що може означати «передача репрезентативного стану». Це стиль проєктування розподілених систем за допомогою обмежень. Центральною абстракцією в REST є ресурс. Ресурс – це представлення віртуального об'єкту (може бути зображення), реального об'єкту (клієнт) чи колекції об'єктів. Дані можуть передаватися у таких форматах: HTML, XML, і розповсюджений JSON.

У проєкті процес відбувається наступним чином (рис 2.2) – Node.js Express експортує REST API та взаємодіє з базою даних MySQL за допомогою Sequelize ORM.

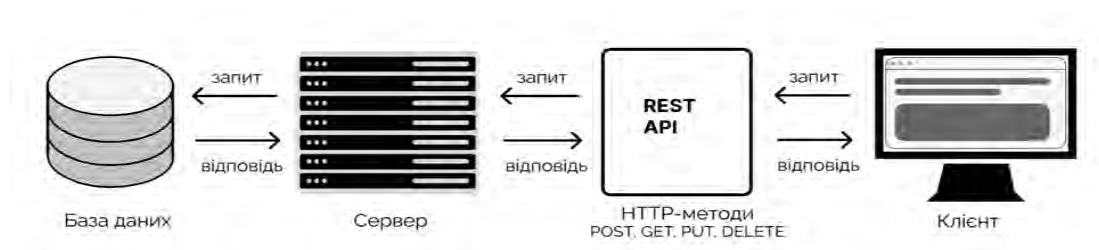


Рисунок 2.2 – Архітектура проєкту

React Client надсилає HTTP-запити та отримує HTTP-відповіді за допомогою Axios, споживає дані про компоненти. React Router використовується для переходу до сторінок [23]-[24].

2.9 Висновки за другим розділом

У другому розділі пояснювальної записки було розглянуто вибір мови програмування, а саме JS. Надано опис бібліотеки React.JS та фреймворку Vue,

та вирішено застосовувати на клієнтській частині бібліотеку React.JS через легку масштабованість, велику спільноту людей, які діляться вирішенням проблем та багатим різновидом навчальних допоміжних засобів, таких як відео, лекції, готові рішення коду. Для побудови серверної частини було розглянуто два фреймворки Express JS і NestJS. Вирішено було використовувати Express JS, завдяки легкості використання та позбавленій складності структури проєкту.

Для зв'язку між клієнтом та сервером було обрано використовувати Axios, оскільки він має більше переваг, ніж Fetch. І це завдяки захисту XSRF та підтримкою браузерів різних версій.

У якості системи керування базою даних обрано MySQL.

Наведено допоміжні засоби у розробці та для тестування – Postman, для збереження відкритого коду – GitHub.

Зазначено архітектуру проєкту, а саме REST архітектура, яка встановлює з'єднання між клієнтом і сервером, застосовуючи HTTP-методи (GET, POST, PUT, DELETE).

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Структура програмного забезпечення

Структуру програмного забезпечення для формування індивідуального навчального плану студента у вигляді структурний діаграм в нотації UML, зокрема діаграми класів, пакетів, компонентів (рис. 3.1 - 3.3). На рис 3.1 зображена загальна діаграма класів програмного забезпечення.

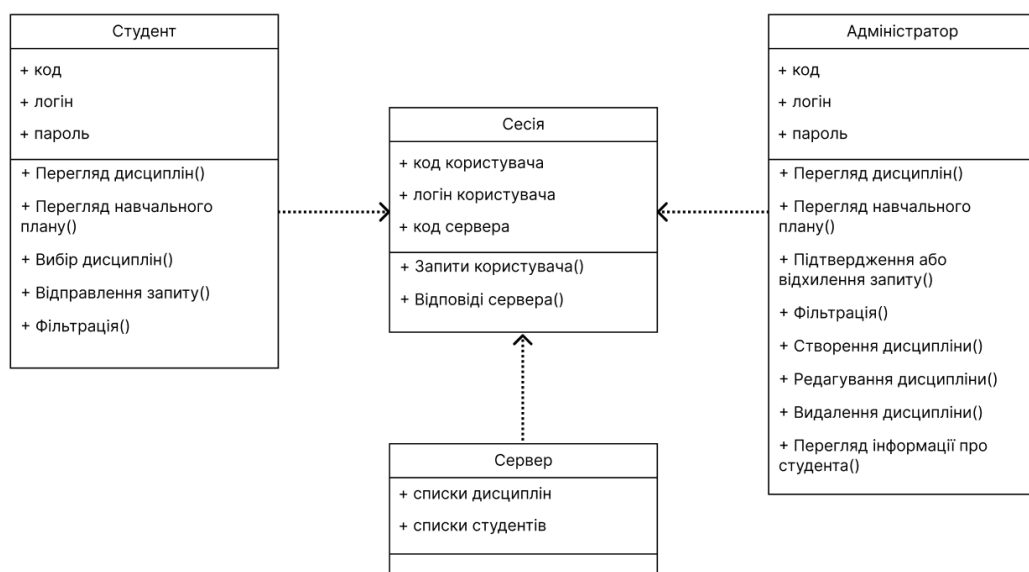


Рисунок 3.1 – Діаграма класів

На рис. 3.2 зображено загальну діаграму пакетів для надання інформації про навчальний план студента.

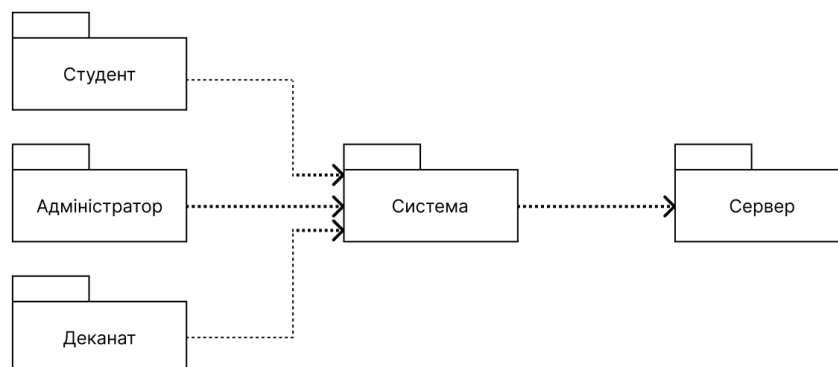


Рисунок 3.2 – Діаграма пакетів програмного забезпечення

На рис. 3.3 зображено загальну діаграму компонентів ПЗ для надання інформації про навчальний план студента.

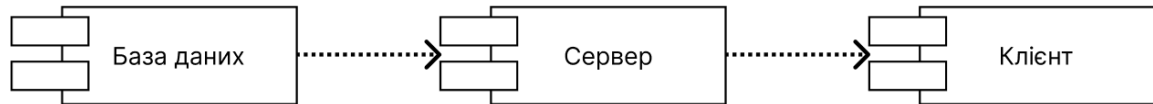


Рисунок 3.3 – Діаграма компонентів програмного забезпечення

3.2 Функціонування програмного забезпечення

За допомогою абстрактних моделей у нотації UML основні діаграми поведінки буде надано функціонування ПЗ для надання інформації про навчальний план студента.

Під час розробки ПЗ ґрунтуючись на діаграмі поведінки були створені діаграми станів, прецедентів.

Загальна діаграма прецедентів ПЗ зображена на рис. 3.4 - 3.5.

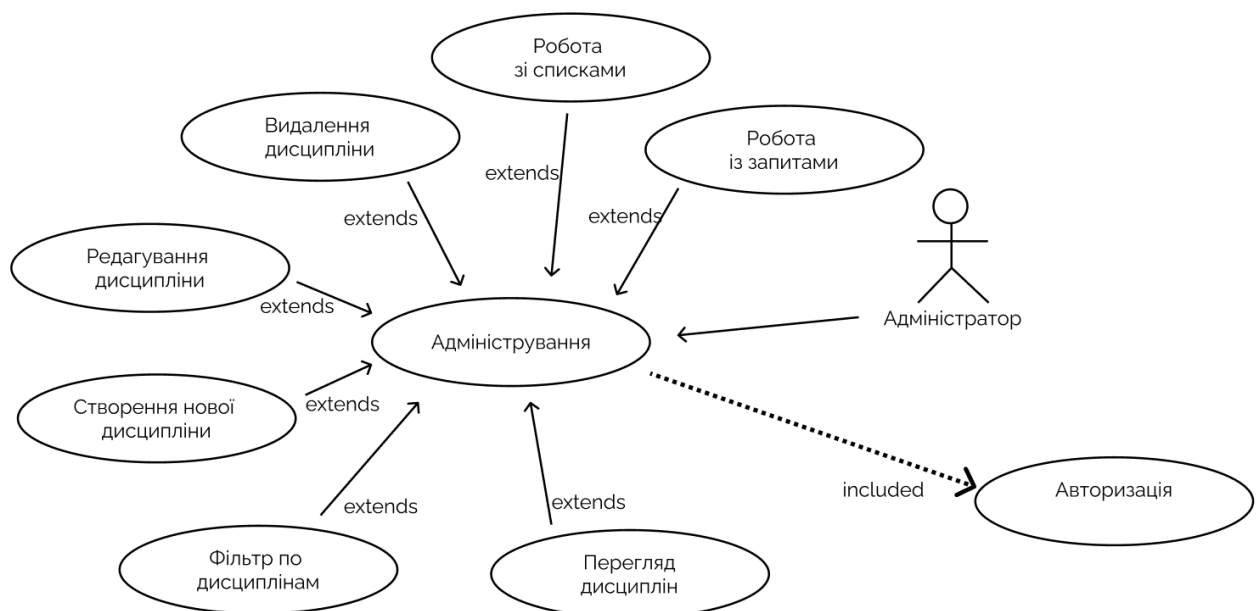


Рисунок 3.4 – Діаграма прецедентів для адміністратора

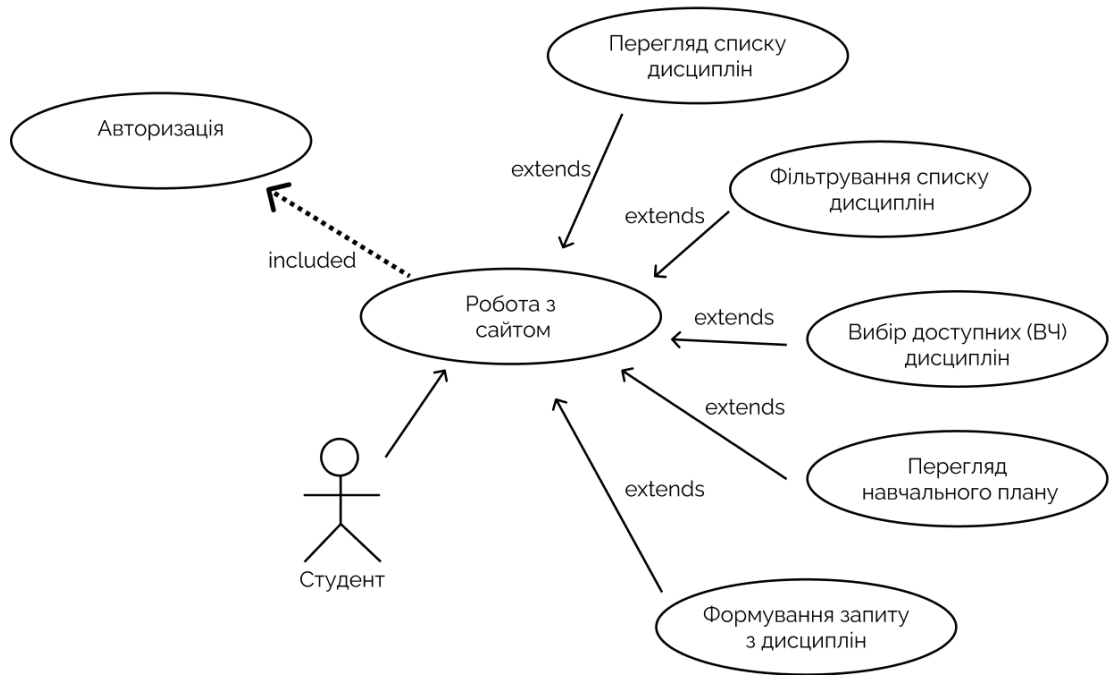


Рисунок 3.5 – Діаграма прецедентів для студента

На рис. 3.6 зображено загальну діаграму станів ПЗ.



Рисунок 3.6 – Діаграма станів програмного забезпечення

На рис. 3.7 зображено загальну діаграму послідовностей ПЗ.

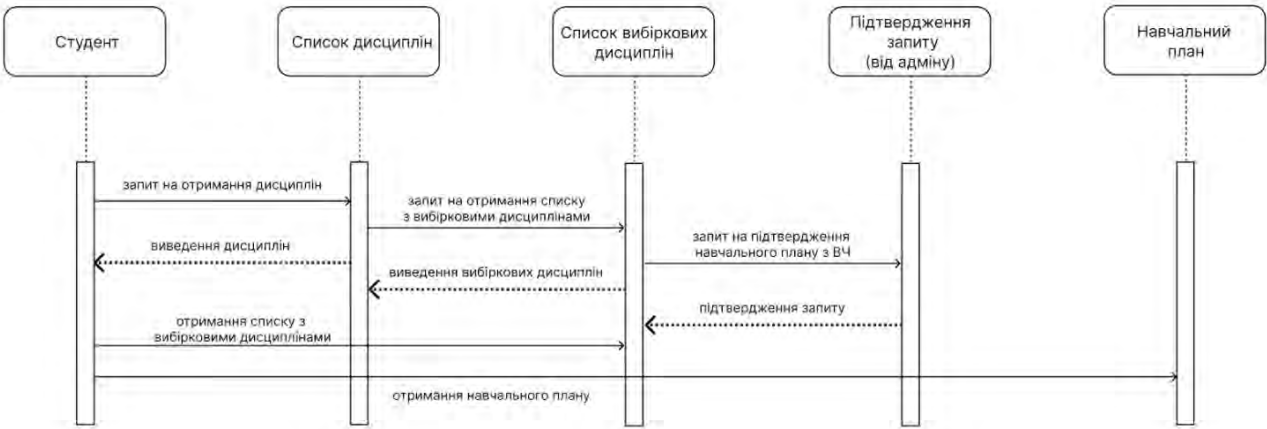


Рисунок 3.7 – Діаграма послідовностей програмного забезпечення

На рис. 3.8 зображено загальну діаграму кооперацій ПЗ.

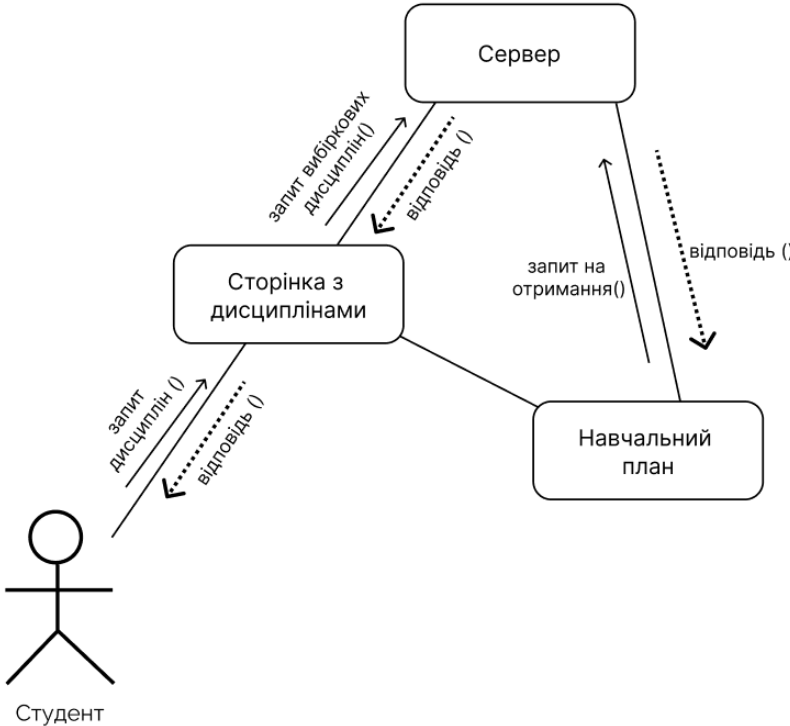


Рисунок 3.8 – Діаграма кооперацій програмного забезпечення

3.3 Алгоритм функціонування розробленої програми

3.3.1 Алгоритм авторизації

На рисунку 3.9 зображено алгоритм авторизації користувача в системі, введення даних, перевірка на вірність отриманих даних, перевірку ролі користувача, вхід до системи.

3.3.2 Алгоритм запиту на вибір вибіркового дисциплін

На рисунку 3.10 за допомогою нотації BPMN-2.0 зображено алгоритм того, як студент, переходячи на сторінку зі списком дисциплін, має можливість обрати вибірково дисципліни за певною кількістю кредитів та надіслати запит на підтвердження вибору.

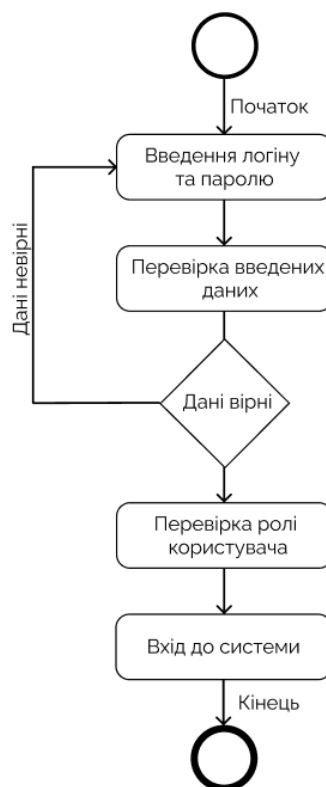


Рисунок 3.9 – Алгоритм авторизації користувача у системі

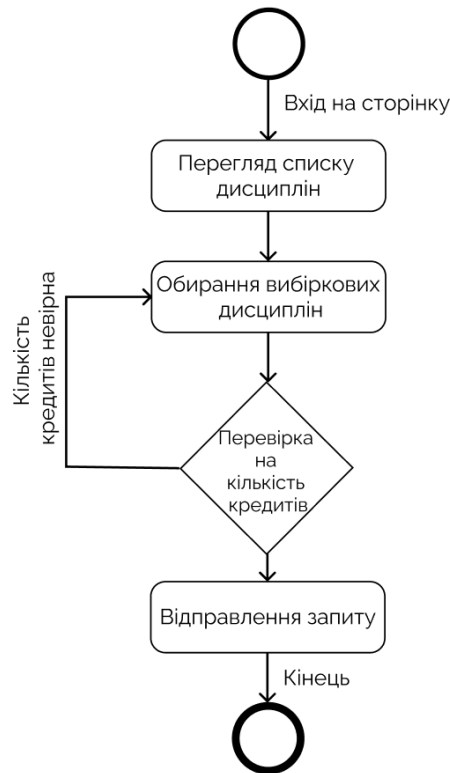


Рисунок 3.10 – Алгоритм вибору вибірових дисциплін

3.4 Проектування бази даних

При проектуванні бази даних було побудовано концептуальну модель бази даних (рис. 3.11).

Сутність «curriculumRequests» (id, user_id, discipline_id, status) – зберігає запити на підтвердження вибірових дисциплін. У таблиці 3.1 наведено опис сутності та типи даних.

Сутність «users» (id, lastname, firstname, patronymic, student_group, year_admission, type_degree, login, password, role_id) – зберігає повну інформацію про користувачів системи. У таблиці 3.2 наведено опис сутності та типи даних.

Сутність «disciplines» (id, title, student_group, part, education_cycle, hours, semester, credits, competition) – зберігає повну інформацію про дисципліни. У таблиці 3.3 наведено опис сутності та типи даних.

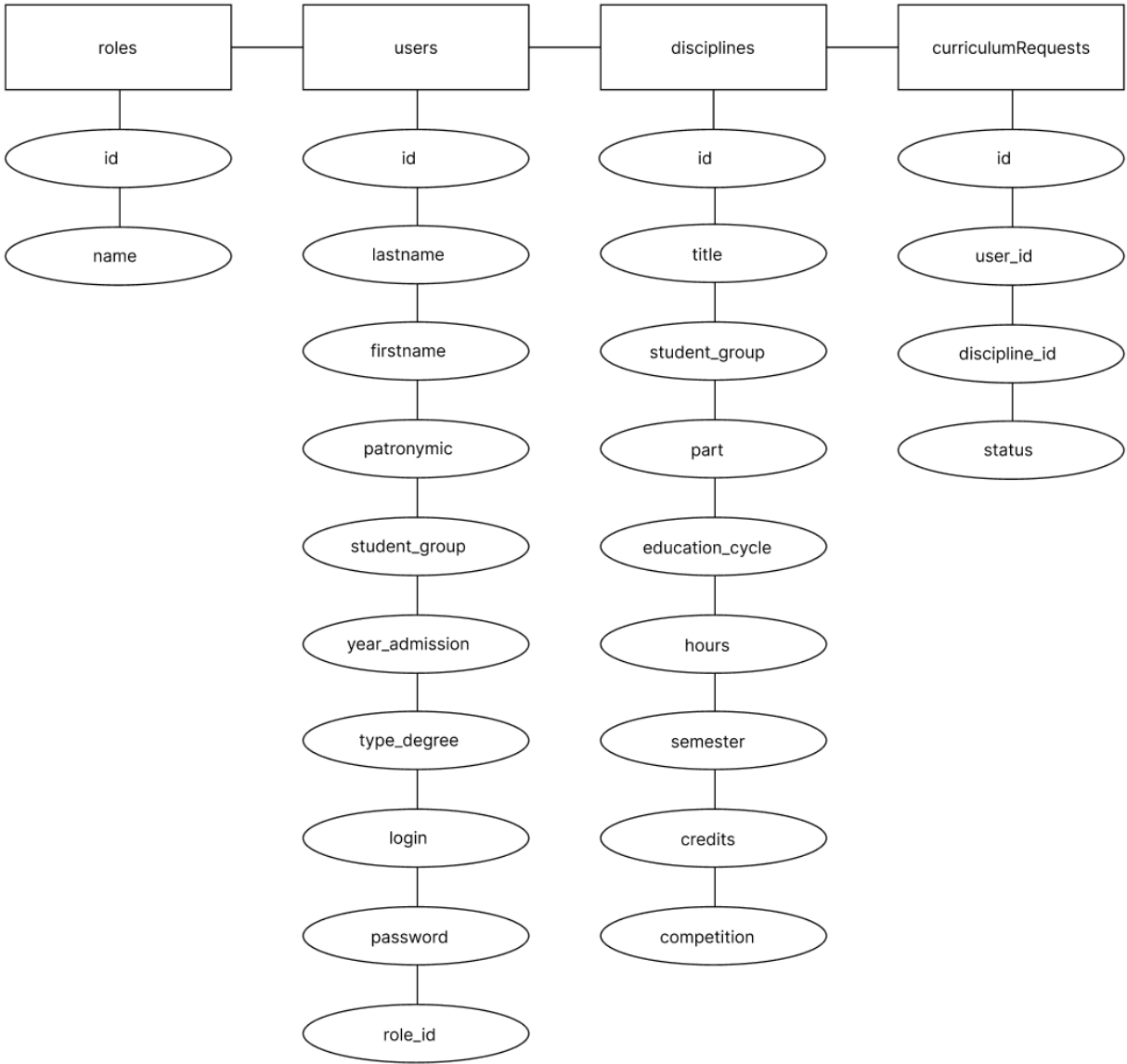


Рисунок 3.11– Концептуальна модель бази даних

Таблиця 3.1 – Опис полів сутності «curriculumRequests»

Назва поля	Тип даних	Розмір
id	int	—
user_id	int	—
discipline_id	int	—
status	varchar	20

Таблиця 3.2 – Опис полів сутності «users»

Назва поля	Тип даних	Розмір
id	int	–
lastname	varchar	45
firstname	varchar	45
patronymic	varchar	45
student_group	varchar	10
year_admission	int	–
type_degree	varchar	15
login	varchar	45
password	varchar	255
role_id	int	–

Таблиця 3.3 – Опис полів сутності «disciplines»

Назва поля	Тип даних	Розмір
id	int	–
title	varchar	255
student_group	varchar	10
part	varchar	45
education_cycle	varchar	255
hours	varchar	10
semester	int	–
credits	varchar	10
competition	varchar	10

Сутність «roles» (id, name) – зберігає список ролей. У таблиці 3.4 наведено опис сутності та типи даних.

Таблиця 3.4 – Опис полів сутності «roles»

Назва поля	Тип даних	Розмір
id	int	—
name	varchar	50

3.5 Проектування дизайну

Нижче будуть наведені приклади проектування дизайну системи для студентів на рис. 3.12 - 3.15.

На рисунках 3.12 зображено список з дисциплінами, на рисунку 3.13 відображена детальна інформація про обрану дисципліну.

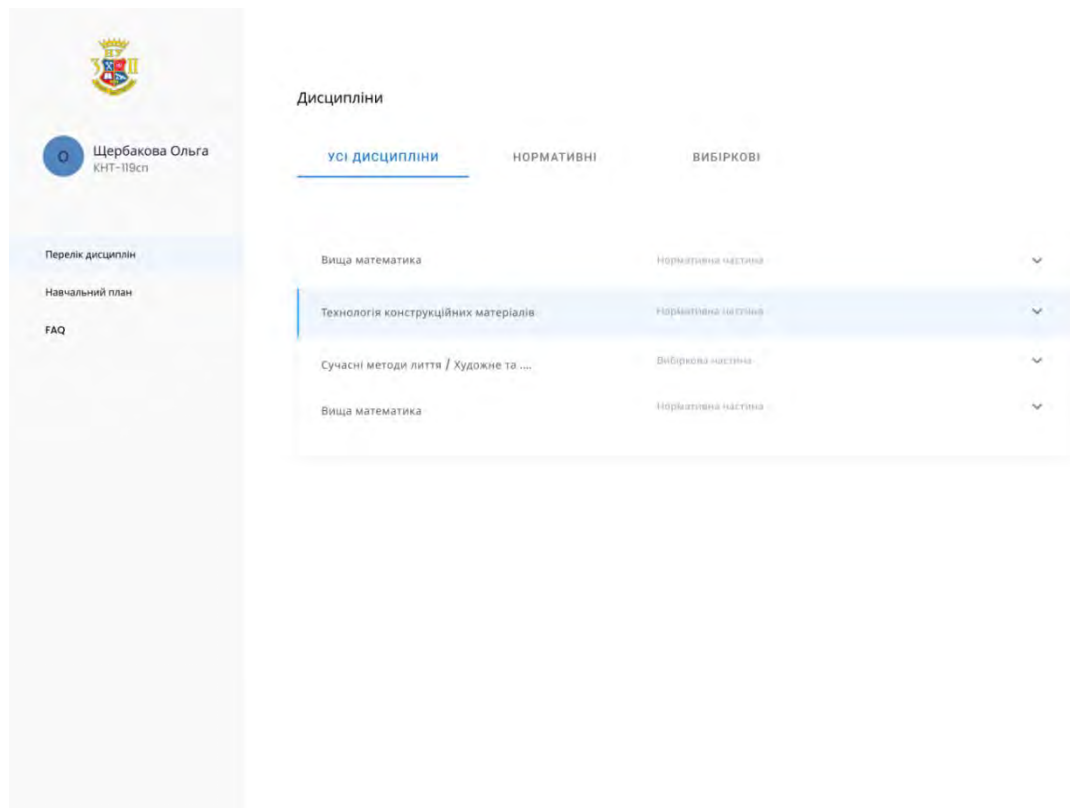


Рисунок 3.12 – Сторінка з усіма дисциплінами

На рисунку 3.13 зображено процес вибору дисциплін.

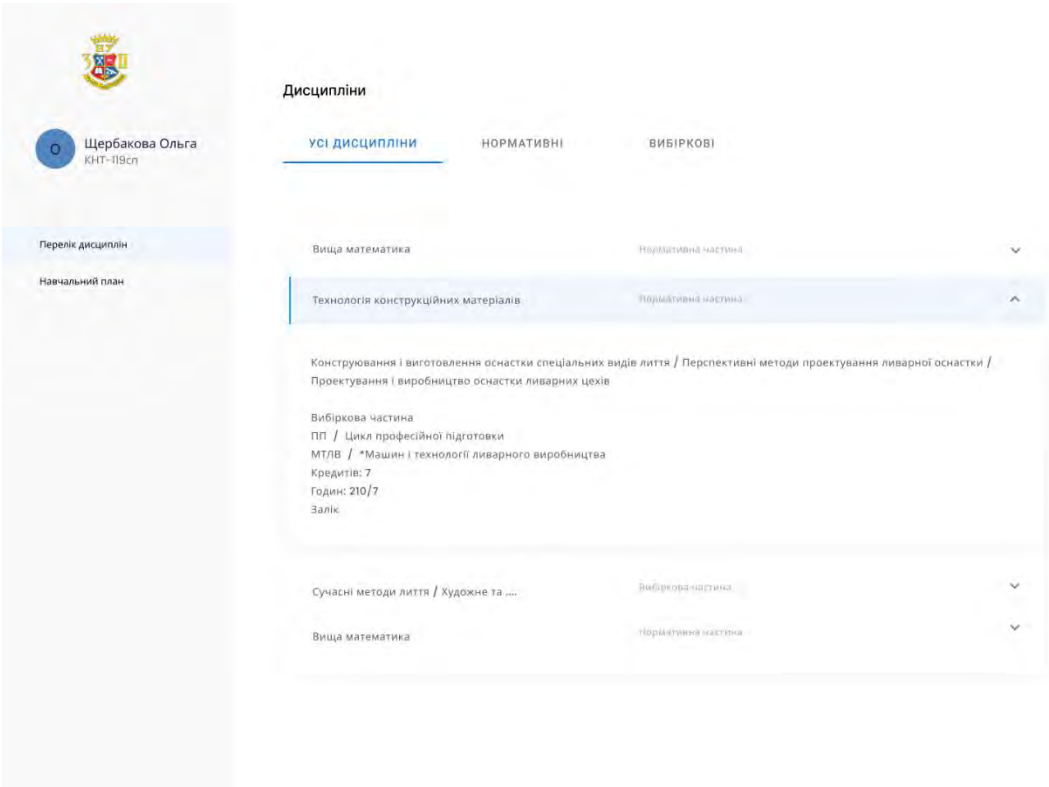


Рисунок 3.13 – Перегляд детальної інформації про дисципліну

На рисунку 3.14 зображено процес обрання вибірових дисциплін. .

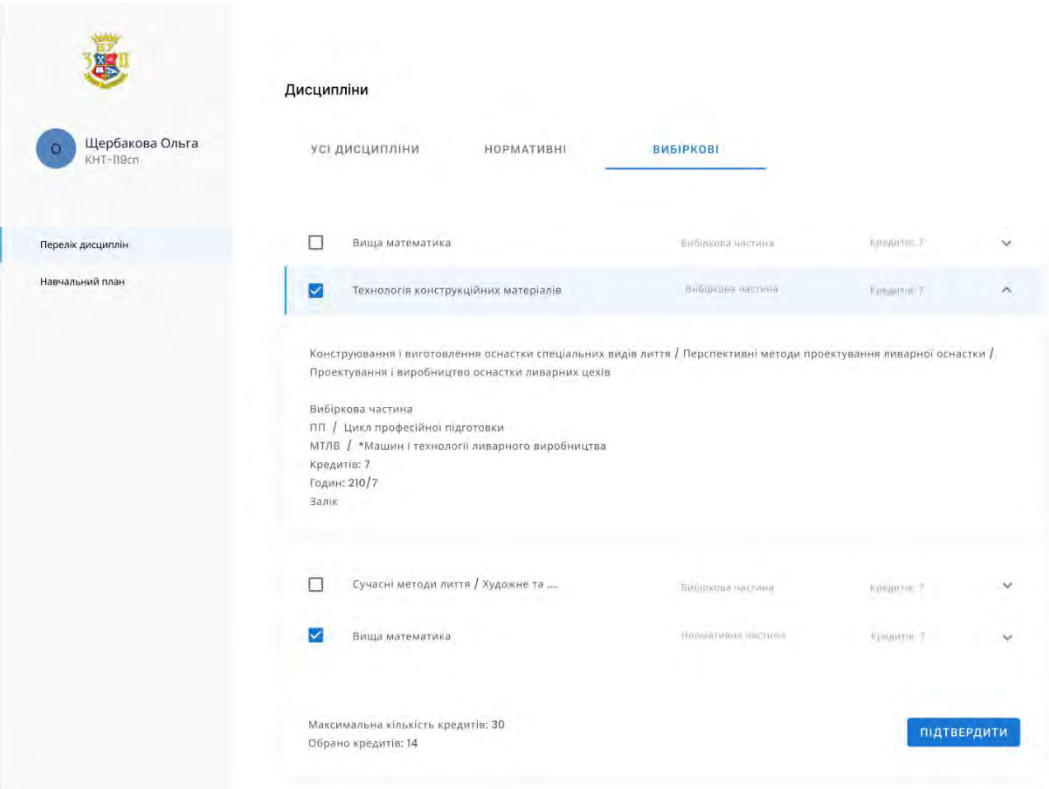


Рисунок 3.14 – Процес обрання вибірових дисциплін

На рисунку 3.15 зображена вкладинка з навчальним планом за другий курс.

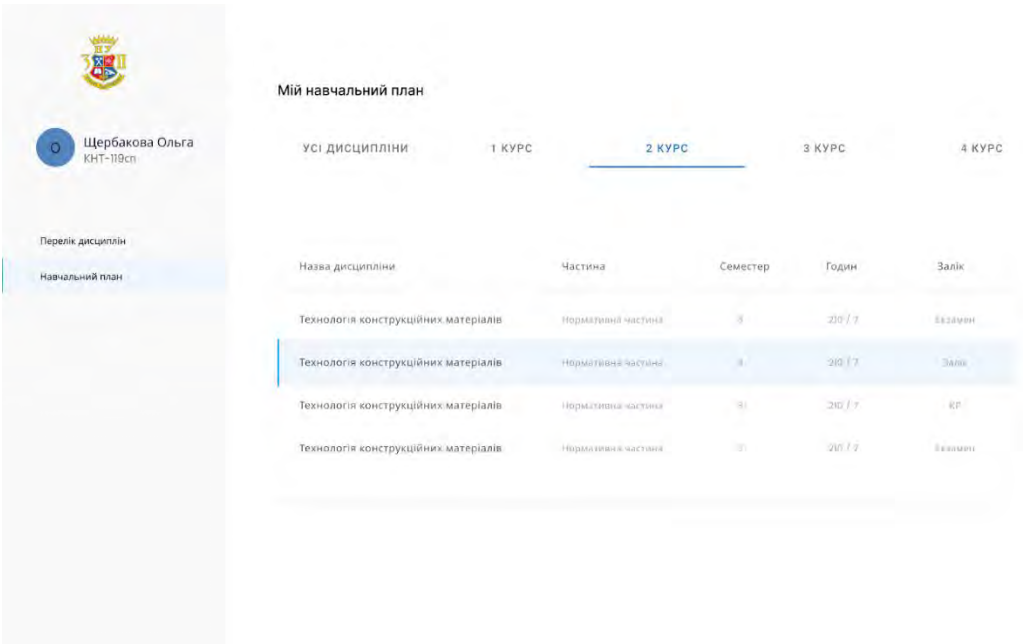


Рисунок 3.15 – Сторінка з навчальним планом за курсом

Далі будуть зображені сторінки для адміністратора (рис. 3.16 - 3.19).
На рисунку 3.16 зображена сторінка із запитом від студента.

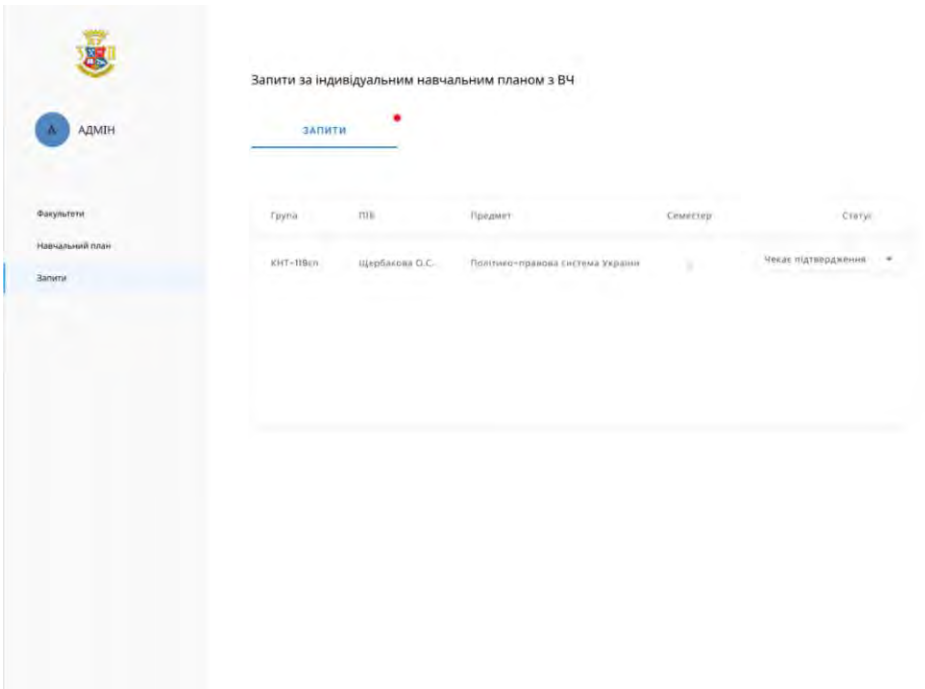


Рисунок 3.16 – Сторінка запитів з вибірковими дисциплінами

На рисунку 3.17 відображено варіанти статусу для запиту.

На рисунку 3.18 зображено модальне вікно для підтвердження дії.

На рисунку 3.19 зображено змінений статус для запиту.

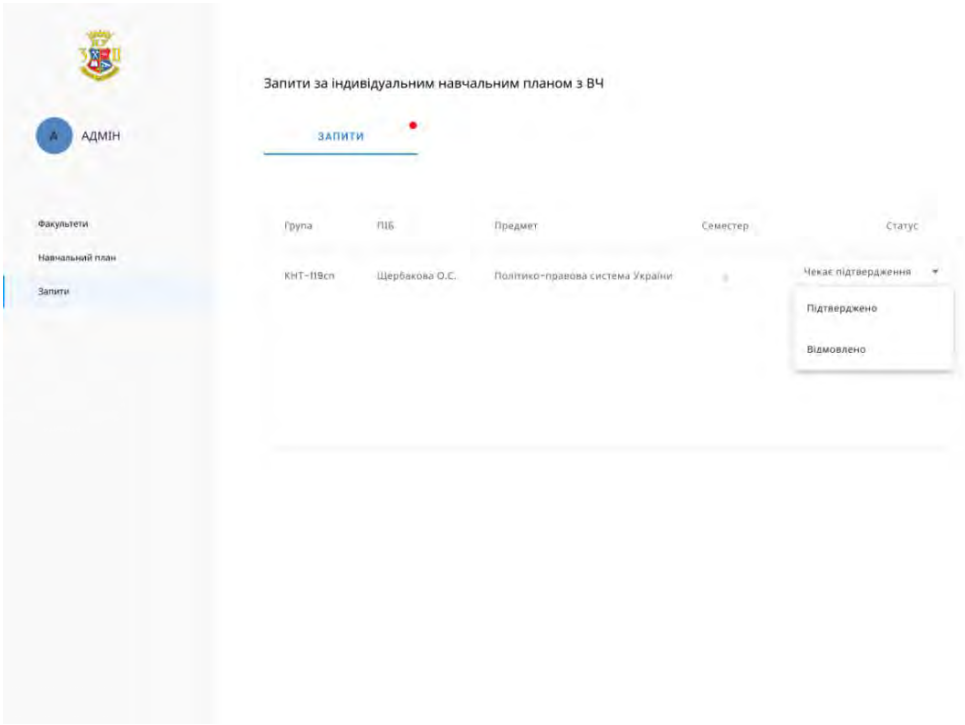


Рисунок 3.17 – Зміна статусу для запиту

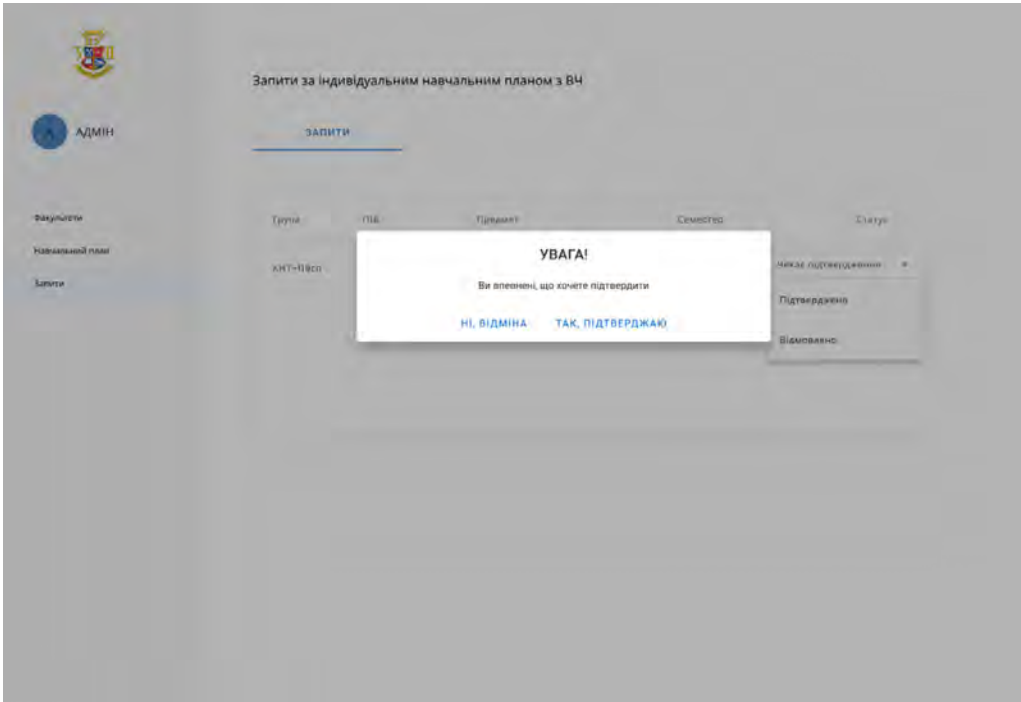


Рисунок 3.18 – Модальне вікно для підтвердження операції

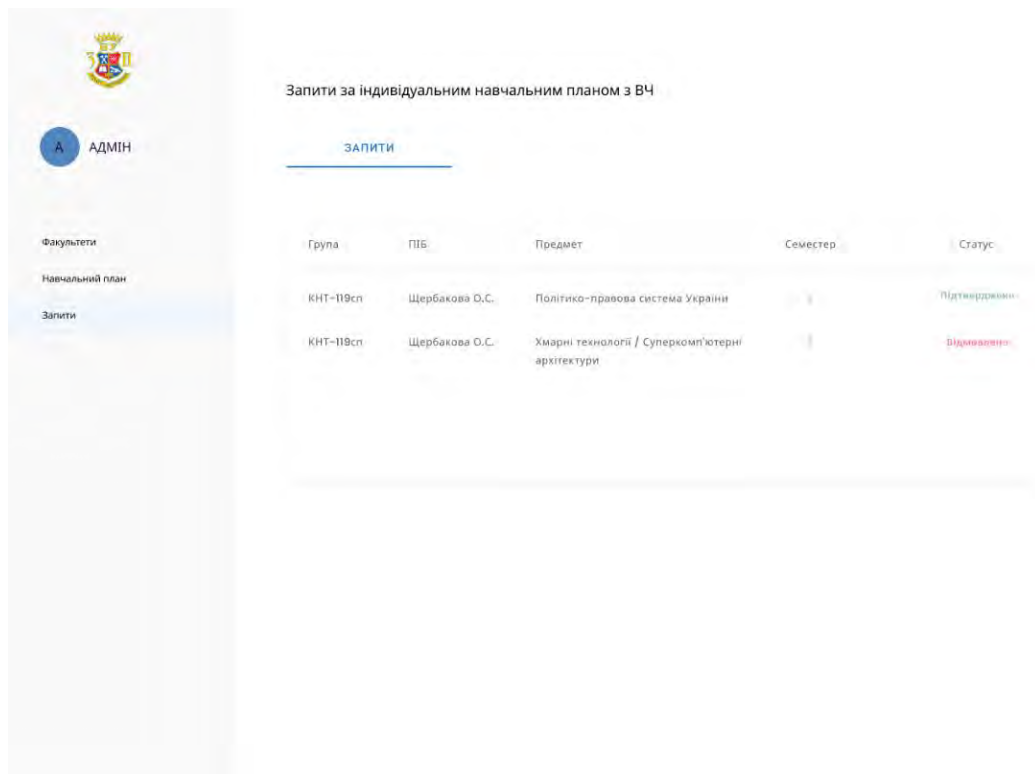


Рисунок 3.19 – Відображення статусів запитів

3.6 Програмна реалізація

Проект розділений на дві частини, які взаємодіють між собою. На рисунку 3.20 файлова структура backend.

Головним файлом є `index.js`, оскільки в ньому відбувається підключення до серверу та бази даних

`Node_modules` містить в собі пакунки, як є необхідними для запуску проекту.

`Routes` містить в собі маршрутизацію, завдяки якій сервер відповідає на запити клієнта по конкретній адресі.

Файл `package.json` зберігає інформацію про проєкт: назву проєкту, версію, залежності, скрипти.

На рис 3.21 файлова структура frontend.

В папці `public` зберігаються зображення, які використовуються на сайті.

Node_modules містить в собі пакунки, які є необхідними для запуску проекту.

Папка components містить собі розподілені компоненти, які можуть повторно використовуватися. Папка pages містить в собі сторінки

Головним файлом являє собою App.js

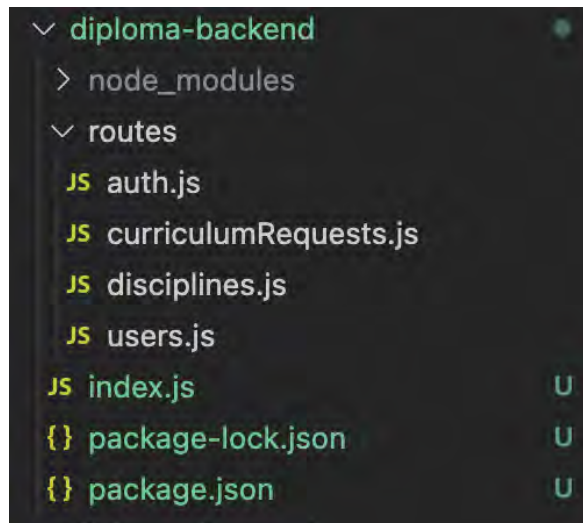


Рисунок 3.20 – Файлова структура backend

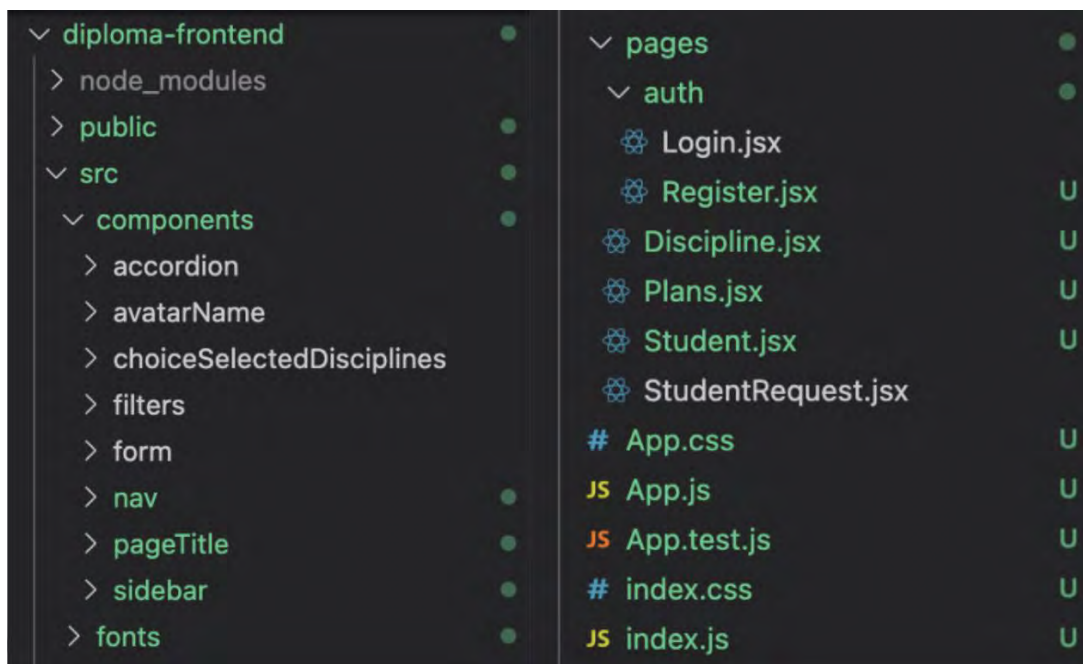


Рисунок 3.21 – Файлова структура frontend

На сервері для початку роботи потрібно зробити під'єднання до БД (рис. 3.22) та щоб перевірити, що з'єднання вдалося, для цього вказана перевірка і виведення повідомлення до консолі. На рисунку 3.23 зображено повідомлення в консолі про з'єднання з БД.

```
✓ const db = mysql.createConnection({  
  host: "127.0.0.1",  
  user: "root",  
  password: "Root",  
  database: "studentroom",  
});  
  
✓ db.connect(function (err) {  
✓   if (err) {  
     return console.error("error: " + err.message);  
   }  
   console.log("Connected to the MySQL server.");  
});
```

Рисунок 3.22 – Код налаштування та під'єднання до бази даних

```
[nodemon] restarting due to changes...  
[nodemon] starting `node index.js`  
Server is working  
Connected to the MySQL server.  
□
```

Рисунок 3.23 – Сервер працює

Для того, щоб студент отримав список дисциплін за своєю кафедрою, створюється запит із параметром, який відправляється з клієнту і обробляється на сервері. В даному випадку на рисунку 3.24 обробляється група студента і відправляється результат

```

app.get("/api/disciplinesByGroup", (req, res) => {
  let sql =
    `SELECT * FROM studyplans WHERE general_group = ${req.body.student_group} ORDER BY semester + 0 ASC`;
  db.query(sql, (error, results) => {
    res.send(results);
    if (error) {
      return console.error(error.message);
    }
  });
});

```

Рисунок 3.24 – Запит для отримання дисциплін

Для отримання даних на клієнті (рис. 3.25) робиться запит на сервер, отримуються дані та зберігаються до state. В useEffect одразу після отримання даних про всі дисципліни відбувається фільтрація даних по навчальним частинам, це Нормативна частина і Вибіркова частина.

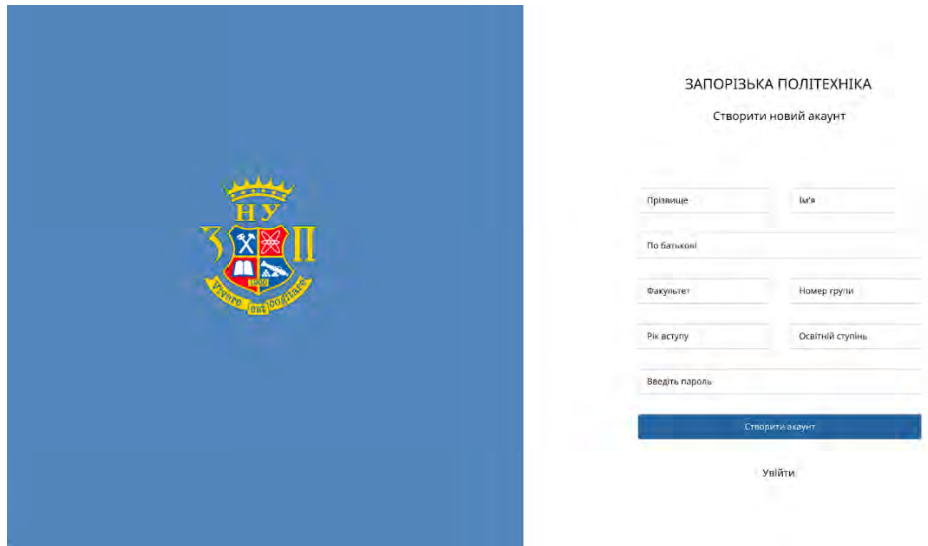
```

useEffect(() => {
  axios
    .get(`http://localhost:5001/api/disciplines?group=${student_group}`)
    .then((res) => {
      const data = res.data;
      setDisciplines(data);
      setRegularPart(
        data.filter((item) => item.part == "НЧ / НОРМАТИВНА ЧАСТИНА")
      );
      setSelectedPart(
        data.filter((item) => item.part == "ВЧ / ВИБІРКОВА ЧАСТИНА")
      );
    })
    .catch((err) => console.log(err));
}, []);

```

Рисунок 3.25 – Запит на сервер для отримання списку дисциплін

Запуск проєкту, головною сторінкою є форма реєстрації в системі (рис. 3.26). Для вдалої реєстрації потрібно ввести всі дані, які зазначено в полях.



ЗАПОРІЗЬКА ПОЛІТЕХНІКА

Створити новий акаунт

Прізвище Ім'я

По батькові

Факультет Номер групи

Рік вступу Освітній ступінь

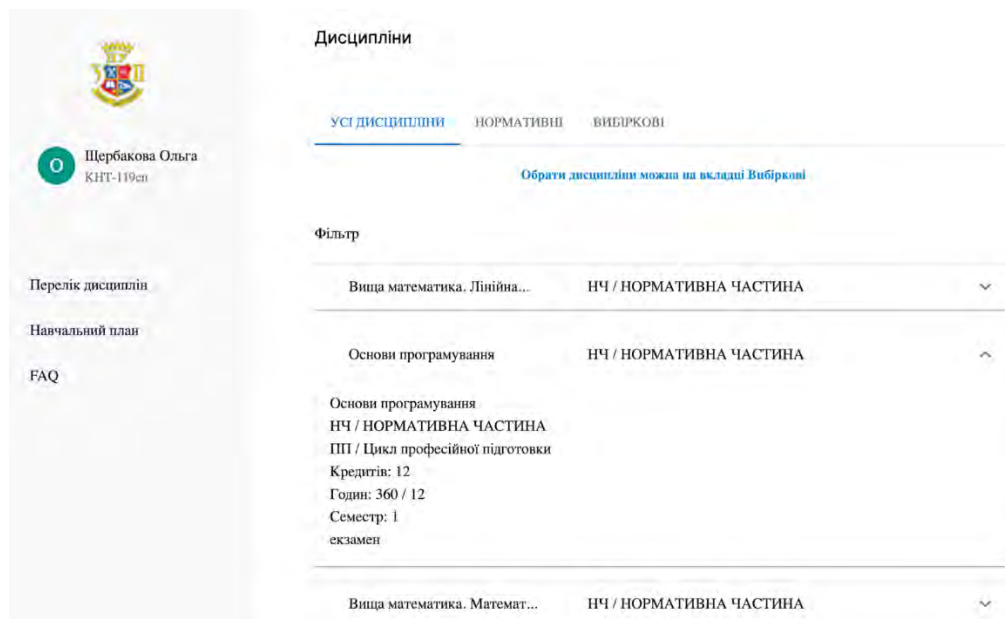
Введіть пароль

Створити акаунт

Увійти

Рисунок 3.26 – Форма

Після того, як реєстрація відбулася успішно, відбувається перехід на сторінку з дисциплінами (рис. 3.27). На сторінці з усіма дисциплінами зазначено, що обирати можна на вкладці Вибіркові.



Дисципліни

УСІ ДИСЦИПЛІНИ НОРМАТИВНІ ВИБІРКОВІ

Обрати дисципліни можна на вкладці Вибіркові

Фільтр

Вища математика. Лінійна...	НЧ / НОРМАТИВНА ЧАСТИНА	▼
Основи програмування	НЧ / НОРМАТИВНА ЧАСТИНА	▲
Основи програмування НЧ / НОРМАТИВНА ЧАСТИНА ПП / Цикл професійної підготовки Кредитів: 12 Годин: 360 / 12 Семестр: 1 екзамен		
Вища математика. Математ...	НЧ / НОРМАТИВНА ЧАСТИНА	▼

Рисунок 3.27 – Сторінка з дисципліни

Рисунок з дисциплінами, які можна обрати. На сторінці зазначена максимальна кількість кредитів на семестр та кількість кредитів обраних

дисциплін. Якщо нічого не обрано, або кількість кредитів недостатня, то кнопка відправлення недоступна.

На рисунку 3.28 показано кількість вірно обраних дисциплін за кредитами, і тому можна здійснити запит на вибір певних дисциплін.

Дисципліна	Статус
☐ Філософія	ВЧ / ВИБІРКОВА ЧАСТИНА
☐ Безпека праці на підприємстві	ВЧ / ВИБІРКОВА ЧАСТИНА
☐ Фізичне виховання / Здоров'я	ВЧ / ВИБІРКОВА ЧАСТИНА
☐ Фізика / Фізичне виховання	ВЧ / ВИБІРКОВА ЧАСТИНА
☐ Теорія ймовірності / Математика	ВЧ / ВИБІРКОВА ЧАСТИНА
☐ Спеціальні розділи вищої освіти	ВЧ / ВИБІРКОВА ЧАСТИНА
☐ Людино-машинна взаємодія	ВЧ / ВИБІРКОВА ЧАСТИНА
☐ Історія української культури	ВЧ / ВИБІРКОВА ЧАСТИНА
☐ Безпека життєдіяльності	ВЧ / ВИБІРКОВА ЧАСТИНА
☐ Фізичне виховання / Здоров'я	ВЧ / ВИБІРКОВА ЧАСТИНА
☐ Фізичне виховання / Здоров'я	ВЧ / ВИБІРКОВА ЧАСТИНА
☐ Науково-дослідницька практика	ВЧ / ВИБІРКОВА ЧАСТИНА

Максимальна кількість кредитів: 30
Обрано: 0

ПІДТВЕРДИТИ

Рисунок 3.28 – Сторінка з вибіровими дисциплінами

Дисципліна	Статус
☐ Теорія ймовірності / Математика	ВЧ / ВИБІРКОВА ЧАСТИНА
☐ Спеціальні розділи вищої освіти	ВЧ / ВИБІРКОВА ЧАСТИНА
☒ Людино-машинна взаємодія	ВЧ / ВИБІРКОВА ЧАСТИНА
☒ Історія української культури	ВЧ / ВИБІРКОВА ЧАСТИНА
☒ Безпека життєдіяльності	ВЧ / ВИБІРКОВА ЧАСТИНА
☒ Фізичне виховання / Здоров'я	ВЧ / ВИБІРКОВА ЧАСТИНА
☒ Науково-дослідницька практика	ВЧ / ВИБІРКОВА ЧАСТИНА

Науково-дослідницька практика / Науково-педагогічна практика / Науково-виробнича практика
ВЧ / ВИБІРКОВА ЧАСТИНА
ПП / Цикл професійної підготовки
Кредитів: 6
Годин: 180 / 6
Семестр: 11
екзамен

Максимальна кількість кредитів: 30
Обрано: 30

ПІДТВЕРДИТИ

Рисунок 3.29 – Вибір дисциплін

3.7 Висновки за розділом 3

Розроблене програмне забезпечення у вигляді вебресурсу для надання інформації про навчальний план студента.

У 3 розділі було запропоновано структуру ПЗ, яку подано у вигляді структурних діаграм в нотації UML, та також надано діаграми класів, пакетів і компонентів.

Функціонування ПЗ для навчального плану студенту зображено за допомогою абстрактних моделей у нотації UML, такі як, діаграма поведінки, діаграма станів, діаграма прецедентів. Також були використані діаграми взаємодії, це діаграма послідовності і діаграма кооперації.

Виконано проєктування бази даних та дизайну.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Опис застосування програми

Вебресурс призначений для полегшення формування навчального плану та для збереження часу для запитів на вибір дисциплін. Програма виконує такі функції:

- автентифікація користувача;
- фільтр по даним;
- вибір вибіркового навчального дисциплін;
- відправлення запиту на узгодження вибіркового дисциплін;
- відхилення або прийняття запиту;
- створення дисципліни;
- видалення дисципліни;
- редагування дисципліни.

Вимоги до клієнтської сторони:

- ОС: Windows Vista (SP2) / Windows 7 (SP1) / Windows 8;
- процесор: 2,66 ГГц Intel Core2Duo E6700;
- ОЗУ: 4 Гбайт;
- відеокарта: 512 Мбайт пам'яті;
- браузер: Chrome, Firefox, Opera, Safari.

Вимоги до серверної сторони:

- ОС: Windows Server;
- процесор: Intel Core i5 2400 2,5 ГГц або AMD FX 6350 3,9 ГГц;
- ОЗУ: 8 Гбайт;
- відео карта: NVIDIA GeForce GTX 660 або AMD Radeon R9 270.

Для запуску коду використовувати:

- редактор коду який підтримує мови вебпрограмування, наприклад VS Code, Atom, Sublime, WebStorm / PhpStorm;

- систему керування базами даних MySQL;
- node.js та npm;
- браузер.

Функціональні характеристики системи щодо формування навчального плану студента наведено у технічному завданні (додаток А). У технічному завданні (додаток Б) надано опис додатку та у завданні (додаток В) надано код програми.

4.2 Характеристики програми

Програмним продуктом є вебресурс, котрий був створений для оптимізації робочих навчальних процесів, для збереження часу та для формування навчального плану студентів. Забезпечує легку подачу даних на запит, щодо вибору вибіркового навчальних дисциплін та затвердження навчального плану.

4.3 Інструкція по експлуатації програми

Оскільки проєкт не є кінцевим результатом, то для того аби запустити його необхідно на своєму ПК мати node.js, оскільки це є головною платформою на проєкті.

Щоб працювати з кодом необхідно завантажити код з GitHub, у папці frontend і backend необхідно зробити встановлення залежностей, для цього у терміналі потрібно ввести `npm install`.

Для запуску серверу потрібно перейти в папку backend і ввести у терміналі `npm run devStart`.

Для запуску клієнтської частини потрібно перейти в папку frontend і ввести у терміналі `npm run start`, після чого відкривається вікно у браузері за таким посиланням `http://localhost:3000/`.

4.4 Тестування та результати роботи програмного забезпечення

На рис. 4.1 демонструється коректне відображення вебресурсу у браузері Google Chrome та на рис. 4.2 у браузері Safari.

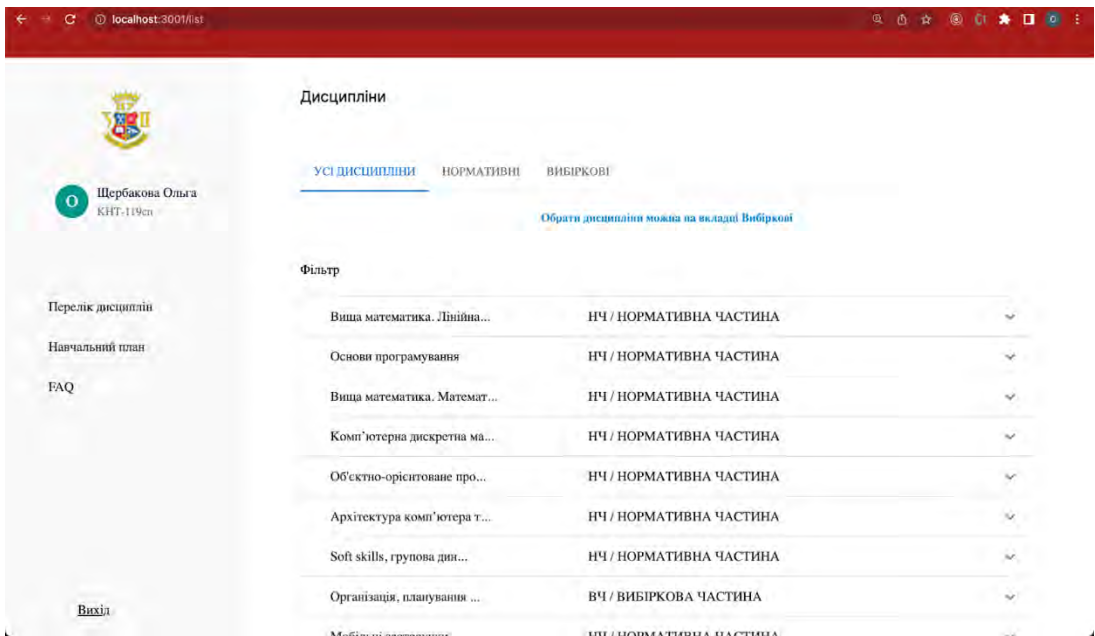


Рисунок 4.1 – Відображення у браузері Google Chrome

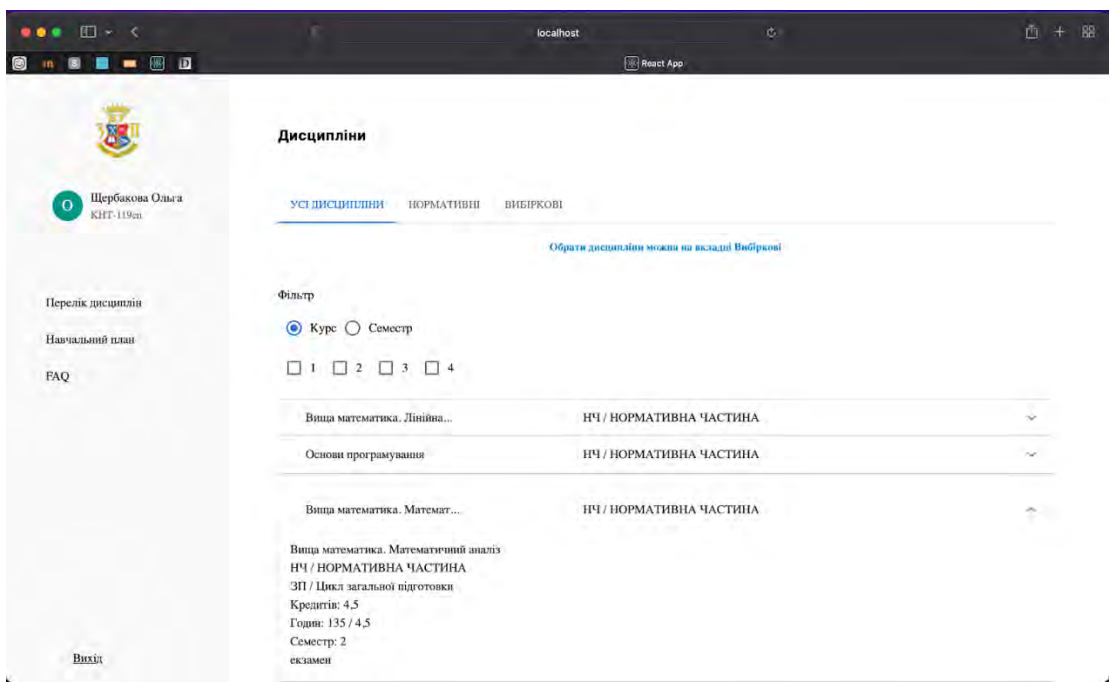


Рисунок 4.2 – Відображення у браузері Safari

Для тестування системи написані тест-кейси. Тест-кейс – це процес, у якому описується дія кроків, за певних умов та параметрів, які є необхідними для перевірки реалізації функції, що тестується.

Опис тест-кейсу функції авторизації надано у таблиці 4.1.

Таблиця 4.1 – Опис тест-кейсу функції авторизації

Мета	Перевірити роботу функції авторизації користувача		
Крок	Опис	Очікуваний результат	Результат тесту
Крок 1	Відправлення форми без введених даних	Відбувається валідація полів та виводиться повідомлення про помилку	Спрацювала валідація полів і отримано повідомлення про помилку
Крок 2	Відправлення форми з невірно вказаними даними	Відбувається валідація полів та виводиться повідомлення про помилку у відповідному полі вводу	Спрацювала валідація полів та отримано повідомлення про помилку у відповідному полі введення
Крок 3	Відправлення форми з коректно вказаними даними	Відбувається вхід до системи	Відбулася переадресація на сторінку

Опис тест-кейсу функції створення нової дисципліни надано у табл. 4.2.

Таблиця 4.2 – Опис тест-кейсу процес створення нової дисципліни

Мета	Перевірити процес створення дисципліни		
Крок	Опис	Очікуваний результат	Результат тесту
Крок 1	Дані не введені до полів	Кнопка створення неактивна	Кнопка створення неактивна
Крок 2	Не заповнені всі необхідні поля	Кнопка створення неактивна	Кнопка створення неактивна
Крок 3	Вірно заповнені всі необхідні поля	Відбувається валідація та кнопка створення активна	Кнопка створення активна

Опис тест-кейсу функції створення нової дисципліни надано у табл. 4.3.

Таблиця 4.3 – Опис тест-кейсу процес вибору навчальних дисциплін і відправлення запиту

Мета	Перевірити процес вибору навчальних дисциплін і відправлення запиту		
Крок	Опис	Очікуваний результат	Результат тесту
Крок 1	Не обрано дисципліни	Кнопка відправлення неактивна	Кнопка відправлення неактивна
Крок 2	Вибрано дисципліни з недостатньою кількістю кредитів	Виводиться інформація про кількість обраних кредитів та кнопка відправлення неактивна	Виводиться інформація про кількість обраних кредитів та кнопка відправлення неактивна
Крок 3	Вибрано дисципліни з достатньою кількістю кредитів	Виводиться інформація про кількість обраних кредитів та кнопка відправлення активна	Виводиться інформація про кількість обраних кредитів та кнопка відправлення активна

Опис тест-кейсу функції видалення дисципліни надано у табл. 4.4.

Таблиця 4.4 – Опис тест-кейсу процес видалення дисципліни

Мета	Перевірити процес видалення дисципліни		
Крок	Опис	Очікуваний результат	Результат тесту
Крок 1	Не обрано дисципліну	Нічого не відбувається	Нічого не відбулося
Крок 2	Обрано дисципліну - випадково	Виводиться модальне вікно для підтвердження дії - відхилення	Вивелося модальне вікно для підтвердження дії, при натисканні на кнопку відхилення все залишилося без змін
Крок 3	Обрано дисципліну - з наміром	Виводиться модальне вікно для підтвердження дії - підтвердження	Вивелося модальне вікно для підтвердження дії, при натисканні на кнопку підтвердження обраний елемент видалився

4.5 Висновки за розділом 4

У 4 розділі проведено опис програмного забезпечення у вигляді вебресурсу для надання інформації про формування навчального плану студента.

Визначено мінімальні системні вимоги, які є необхідними для коректної роботи системи.

Прописано інструкцію по розгортанню проєкту на ПК.

У вигляді таблиць було надано опис проведених тест-кейсів, з вказаною метою тесту, кількістю кроків, описом кроку, з очікуваним результатом та сам результат тесту. Тестування відбулося на таких функціональних вимогах:

- авторизація користувача;
- вибір навчальних дисциплін і відправлення запиту на підтвердження;
- створення нової дисципліни;
- видалення дисципліни.

ВИСНОВКИ

В ході виконання дипломної роботи у першому розділі було проведено аналіз предметної області. Визначено основні поняття навчальної дисципліни, що це система знань, умінь і навичок, відібраних з певної галузі наукової, технічної, художньої, виробничої діяльності для навчання в навчальному закладі. Визначено актуальність роботи та досліджено аналоги сайтів. Проведено порівняльну характеристику аналогів.

У другому розділі пояснювальної записки було проведено вибір мов програмування для розроблюваного забезпечення, проведено опис обраних мов програмування. Проведено опис обраної бібліотеки React.js на стороні клієнта, та фреймворку Express на стороні сервера. Обґрунтовано середовище розробки та наведено порівняльну характеристику у вигляді таблиці. Обрано середовище керування базами даних MySQL та зазначено критерії, які забезпечує СКБД. Допоміжними засобами при розробки були Postman для тестування API та git система контролю версіями. Застосунок має клієнт-серверну архітектуру, з REST архітектурним стилем.

У третьому розділі було надано структуру програмного забезпечення у вигляді діаграми класів, діаграми пакетів і діаграми компонентів. За допомогою діаграм прецедентів, діаграми станів і діаграми послідовностей надано функціонування ПЗ. Надано основні алгоритми програми. Спроектовано базу даних, описано сутності таблиць і зроблено концептуальну модель БД. Надано ілюстрації з проєктуванням дизайну вебресурсу.

У четвертому розділі проводилося тестування та дослідження програми, тестування проводилося за написанням тест-кейсів, які описують мету та кроки тестування з очікуванням та з дійсним результатом. Наголошено про вимоги до ПК, надано інструкцію по експлуатації програми. Вимоги до інтерфейсу було стримано, дизайн зроблений в одному стилі, без яскравих відтінків, простий та зрозумілий для користувача та містить навігацію по вебсайту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Положення про порядок та умови обрання студентами вибіркових навчальних дисциплін [Електрон. ресурс]. – Режим доступу: <https://vsau.org/assets/images/content/dokPDF/proekt-pro-vybirkovi-dysc.pdf>.
2. 5 Quick Tips on How To Use Google Classroom [Electronic resource]. – Access mode: <https://www.aeseducation.com/blog/2016/10/how-to-use-google-classroom>.
3. Documentation [Electronic resource]. – Access mode: https://docs.moodle.org/400/en/Main_page.
4. Програми для безпаперового менеджменту в школі [Електрон. ресурс]. – Режим доступу: <https://edpro.ua/blog/programy-dlja-bezpaperovogo-menedgmentu-v-shkoli>.
5. Система Управління Школою та Навчанням для учнів [Електронний ресурс]. – Режим доступу: <https://smls.com.ua/>.
6. JavaScript [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/javascript/>.
7. Вступ в React, якого нам не вистачало [Електрон. ресурс]. – Режим доступу: <https://devzone.org.ua/post/vstup-v-react-yakogo-nam-ne-vistachalo>.
8. Що таке Virtual DOM? [Електрон. ресурс]. – Режим доступу: <https://codeguida.com/post/1561>.
9. WHAT IS VUE JS? [Electronic resource]. – Access mode: <https://www.tatvasoft.com/outsourcing/2021/10/what-is-vue-js-and-its-benefits.html>.
10. Що таке node.js? [Електрон. ресурс]. – Режим доступу: <https://uk.theastrologypage.com/node-js>.
11. Посібник для початківців Node.js (JavaScript на сервері) [Електрон. ресурс]. – Режим доступу: <https://ua.phhsnews.com/articles/coding/beginners-guide-to-node-js-server-side-javascript.html>.

12. Working with Express, a Node.js Framework [Electronic resource]. – Access mode: <https://medium.com/@SigniorGratiano/working-with-express-a-node-js-framework-e38c0807dbf>.

13. 10 Best NodeJS Frameworks for Developers [Electronic resource]. – Access mode: <https://hackr.io/blog/nodejs-frameworks>.

14. Difference between Fetch and Axios.js for making http-requests [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/difference-between-fetch-and-axios-js-for-making-http-requests>.

15. The Fetch() API Explained in 3 Simple Steps [Electronic resource]. – Access mode: <https://javascript.plainenglish.io/fetch-api-in-three-simple-steps-explanation-276979dc8798>.

16. Getting Started Visual Studio Code [Electronic resource]. – Access mode: <https://code.visualstudio.com/docs>.

17. WebStorm [Electronic resource]. – Access mode: <https://www.jetbrains.com/webstorm>.

18. JetBrains WebStorm [Електрон. ресурс]. – Режим доступу: <https://itpro.ua/product/jetbrains-webstorm/?tab=description>.

19. What is database management system [Electronic resource]. – Access mode: <https://www.ibm.com/docs/en/zos-basic-skills?topic=zos-what-is-database-management-system>.

20. What is MySQL? [Electronic resource]. – Access mode: <https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html>

21. 15 Сильні та слабкі сторони MySQL Server, які ви повинні знати [Електрон. ресурс]. – Режим доступу: <https://altitudetvm.com/uk/komputer/1522-15-kelebihan-dan-kekurangan-mysql-server-yang-perlu-diketahui.html>.

22. What Is GitHub? (And What Is It Used For?) [Electronic resource]. – Access mode: <https://blog.hubspot.com/website/what-is-github-used-for>.

23. Що таке RESTful API? [Електрон. ресурс]. – Режим доступу: <https://codeguida.com/post/601>.

24. REST API architectural constraints [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/rest-api-architectural-constraints>.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення у вигляді вебресурсу для організації навчального плану, зокрема самостійний вибір навчальних дисциплін.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Кабінет студента для формування індивідуального навчального плану», затверджене наказом Національного університету «Запорізька політехніка» №102 від 27 квітня 2022 р.

A.2 Призначення розробки

Проект призначений для студентів та викладачів, який допомагає поліпшити подачу запиту на вивчення різних дисциплін та для формування навчальних планів.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

До основних вимог відносяться:

- автентифікація в системі;
- для студентів перегляд списків з дисциплінами, навчального плану, обрання вибіркового дисциплін;
- для адміністратора перегляд списків з дисциплінами, з студентами, з групами;
- виведення даних за певними критеріями;

Авторизований користувач у ролі Студент повинен мати наступні можливості:

- авторизація у системі;
- перегляд дисциплін;
- фільтрація дисциплін;
- формування навчальних вибіркового дисциплін;
- запит на прийняття вибору дисциплін;
- перегляд навчального плану, як нормативні дисципліни так і вибірково.

Авторизований користувач у ролі Адміністратор повинен мати наступні можливості:

- авторизація у системі;
- перегляд списків усіх дисциплін, студентів;
- фільтр по спискам;
- прийняття або відхилення запитів від студентів.

A.3.2 Вимоги до інтерфейсу програми

Вебресурс для надання інформації про навчальний план повинен мати наступні вимоги до інтерфейсу:

- простота та зрозумілість;
- кольори в одному стилі, без яскравих відтінків;
- гнучкість інтерфейсу;
- навігація по вебсайту.

A.3.3 Вимоги до надійності

Дані зберігаються в базі даних MySQL, при помилках дій на сайті в системі обробляються помилки і відображаються повідомлення про некоректність дії.

A.3.4 Умови експлуатації

Для експлуатації програми необхідно мати код програми, який розміщений на GitHub, певні вимоги до апаратного забезпечення та здійснити налаштування згідно з експлуатаційної документації.

A.3.5 Вимоги до складу та параметрів технічний засобів

Вимоги до клієнтської сторони: підключення до мережі Інтернет та браузер з використання JS.

Вимоги до серверної сторони:

- ОС: Windows Server;
- процесор: Intel Core i5 2400 2,5 ГГц або AMD FX 6350 3,9 ГГц;
- ОЗУ: 8 Гбайт;
- відео карта: NVIDIA GeForce GTX 660 або AMD Radeon R9 270.

A.3.6 Вимоги до маркування і пакування

На упаковці повинна бути назва програми – «Вебсайт для формування індивідуального навчального плану студента».

A.4 Вхідні дані до роботи

Вхідними даними до ПЗ є усі дані, які введені з клавіатури користувачем та отримані на стороні серверу.

Вихідними даними ПЗ є інформація, яка оброблена із запитів від клієнтів.

ДОДАТОК Б
Опис програми

Б.1 Загальні відомості

Вебсайт для формування індивідуального навчального плану студента та адміністрування.

Б.2 Функціональне призначення

Програмне забезпечення надає такі можливості:

- автентифікація в системі;
- для студентів перегляд списків з дисциплінами, навчального плану, обрання вибірових дисциплін, запит на прийняття вибору дисциплін, перегляд навчального плану, як нормативні дисципліни так і вибірові;
- для адміністратора перегляд списків з дисциплінами, з студентами;
- прийняття або відхилення запитів від студентів;
- виведення даних за певними критеріями.

Б.3 Використані технічні засоби

Для експлуатації програмного забезпечення у вигляді вебсайту формування індивідуального навчального плану студента необхідно таке апаратне забезпечення:

- процесор з тактовою частотою 2.6ГГц;
- ОЗУ 8Гбайт;
- відео карта: NVIDIA GeForce GTX 660 або AMD Radeon R9 270;
- наявність пристроїв для введення та виведення інформації.

Б.4 Виклик і завантаження

Для роботи з клієнтською і серверною частиною потрібно мати node.js, оскільки це є головною платформою на проєкті.

Щоб працювати з кодом необхідно завантажити код з GitHub, у папці frontend і backend необхідно зробити встановлення залежностей, для цього у терміналі потрібно ввести `npm install`.

Для запуску серверу потрібно перейти в папку backend і ввести у терміналі `npm run devStart`.

Для запуску клієнтської частини потрібно перейти в папку frontend і ввести у терміналі `npm run start`, після чого відкривається вікно у браузері з таким посиланням <http://localhost:3000/>.

Б.5 Вхідні та вихідні дані

Вхідними даними до ПЗ є усі дані, які введені з клавіатури користувачем та отримані на стороні серверу.

Вихідними даними ПЗ є інформація, яка оброблена із запитів від клієнтів.

ДОДАТОК В
Текст програми

```

const express = require("express");
const app = express();
const mysql = require("mysql");
const cors = require("cors");
const bodyParser = require("body-parser");
const db = mysql.createConnection({
  host: "127.0.0.1",
  user: "root",
  password: "Root",
  database: "studentroom",
});
db.connect(function (err) {
  if (err) {
    return console.error("error: " + err.message);
  }
  console.log("Connected to the MySQL server.");
});
app.use(cors());
app.use(express.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.get("/api/disciplines", (req, res) => {
  console.log(req.query.group);
  let sql = "SELECT * FROM disciplines";
  if (req.query.group) {
    sql = `SELECT * FROM disciplines WHERE student_group="${req.query.group}"`;
  }
  db.query(sql, (error, results) => {
    if (error) {
      return console.error(error.message);
    }
    res.send(results);
  });
});
app.post("/api/register", (req, res) => {
  const lastname = req.body.lastname;
  const firstname = req.body.firstname;
  const patronymic = req.body.patronymic;
  const studentGroup = req.body.studentGroup;
  const yearAdmission = req.body.yearAdmission;
  const typeDegree = req.body.typeDegree;
  const login = req.body.login;
  const password = req.body.password;

```

```

let sqlInsert =
  "INSERT INTO users (lastname, firstname ,patronymic, studentGroup, yearAdmission, typeDegree, login, password)
VALUES (?, ?, ?, ?, ?, ?, ?, ?)";

db.query(
  sqlInsert,
  [lastname, firstname,patronymic, studentGroup, yearAdmission, typeDegree, login, password],
  (error, result) => {
    if (error) {
      console.log(error);
    }
  }
);
});

app.get("/api/disciplinesByGroup", (req, res) => {
  let sql = `SELECT * FROM studyplans WHERE general_group =`${req.studentGroup}` ORDER BY semester + 0
ASC`;
  db.query(sql, (error, results) => {
    res.send(results);
    if (error) {
      return console.error(error.message);
    }
  });
});

app.listen(5001, ())

```

```

import React, { useState } from "react";
import { Box, Button, TextField } from "@mui/material";
import axios from "axios";
const Register = () => {
  const [authValues, setAuthValues] = useState({
    lastname: "",
    firstname: "",
    patronymic: "",
    studentGroup: "",
    yearAdmission: "",
    typeDegree: "",
    login: "",
    password: "",
  });
  const handleInputChange = (event) => {
    const { name, value } = event.target;

```

```

    setAuthValues({ ...authValues, [name]: value });
  };
const register = () => {
  axios
    .post("http://localhost:5001/api/register", {
      lastname: authValues.lastname,
      firstname: authValues.firstname,
      groupN: authValues.group,
      degree: authValues.degree,
      email: authValues.email,
      password: authValues.password,
    })
    .then(function (response) {
      console.log(response);
      setAuthValues("");
    })
    .catch(function (error) {
      console.log(error);
    });
};
return (
  <Box
    sx={{
      margin: "10px auto",
      display: "flex",
      flexDirection: "column",
      width: "50%",
    }}
  >
    <TextField
      required
      id="outlined-required"
      value={authValues.firstname}
      name="firstname"
      placeholder="firstname"
      onChange={(event) => handleInputChange(event)}
    />
    <TextField
      required
      id="outlined-required"
      value={authValues.lastname}
      name="lastname"

```

```

placeholder="lastname"
onChange={(event) => handleInputChange(event)}
/>
<TextField
  required
  id="outlined-required"
  value={authValues.patronymic}
  name="patronymic"
  placeholder="patronymic"
  onChange={(event) => handleInputChange(event)}
/>
<TextField
  required
  id="outlined-required"
  value={authValues.studentGroup}
  name="studentGroup"
  placeholder="studentGroup"
  onChange={(event) => handleInputChange(event)}
/>
<TextField
  required
  id="outlined-required"
  value={authValues.yearAdmission}
  name="yearAdmission"
  placeholder="yearAdmission"
  onChange={(event) => handleInputChange(event)}
/>
<TextField
  required
  id="outlined-required"
  value={authValues.typeDegree}
  name="typeDegree"
  placeholder="typeDegree"
  onChange={(event) => handleInputChange(event)}
/>
<TextField
  required
  id="outlined-password-input"
  value={authValues.login}
  name="login"
  placeholder="login"
  onChange={(event) => handleInputChange(event)}

```

```

    />
    <TextField
      required
      id="outlined-required"
      value={authValues.password}
      name="password"
      placeholder="password"
      onChange={(event) => handleInputChange(event)}
    />
    <Button variant="outlined" onClick={register}>
      Submit
    </Button>
  </Box>
);
};
export default Register;
import { Box, Grid, Typography } from "@mui/material";
import React from "react";
import AvatarName from "../avatarName/AvatarName";
import NavLinks from "../nav/NavLinks";
import { styles } from "../sidebar.styles";
const Sidebar = () => {
  return (
    <Grid item sx={styles.sidebar}>
      <Box sx={{ padding: "40px 0" }}>
        
      </Box>
      <AvatarName />
      <NavLinks />
    </Grid>
  );
};
export default Sidebar;
import React from "react";
import { Box, Avatar, Typography } from "@mui/material";
import { styles } from "../avatarName.styles";
const AvatarName = () => {
  return (
    <Box sx={styles.box}>
      <Avatar sx={styles.avatar}>O</Avatar>
    </Box>
    <Typography sx={styles.studentName}>Щербакова Ольга</Typography>
  );
};

```

```

    <Typography sx={styles.studentGroup}>KHT-119cn</Typography>
  </Box>
</Box>
);
};
export default AvatarName;
import React from "react";
import { Box, Checkbox, Typography } from "@mui/material";
import { styles } from "../filters.styles";
const FilterCheckbox = ({ element, state, handleCheckChange }) => {
  let a = element.course;
  return (
    <Box sx={{ ...styles.block, marginRight: "10px" }}>
      <Checkbox
        checked={state[a]}
        onChange={handleCheckChange}
        name={element.course}
      />
      <Typography>{element.numb}</Typography>
    </Box>
  );
};
export default FilterCheckbox;
import React, { useState } from "react";
import { Box, Checkbox, Typography, Radio } from "@mui/material";
import { styles } from "../filters.styles";
import FilterCheckbox from "../FilterCheckbox";
const Filters = () => {
  const [isShow, setIsShow] = useState(true);
  const [state, setState] = useState({
    first: false,
    second: false,
    third: false,
    four: false,
  });
  const handleCheckChange = (event) => {
    setState({
      ...state,
      [event.target.name]: event.target.checked,
    });
  };
};

```

```

const [selectedValue, setSelectedValue] = useState("course");
const handleChange = (event) => {
  setSelectedValue(event.target.value);
};
return (
  <Box sx={styles.section}>
    <Typography
      onClick={() => setIsShow(!isShow)}
      sx={{ cursor: "pointer", marginBottom: "10px" }}
    >
      Фільтр
    </Typography>
    {!isShow && (
      <
        <Box sx={styles.block}>
          <Radio
            checked={selectedValue === "course"}
            onChange={handleChange}
            value="course"
            name="radio-buttons"
          />
          <Typography>Курс</Typography>
          <Radio
            checked={selectedValue === "semester"}
            onChange={handleChange}
            value="semester"
            name="radio-buttons"
          />
          <Typography>Семестр</Typography>
        </Box>
        {selectedValue === "course" && (
          <Box sx={{ ...styles.block, margin: "10px 0" }}>
            {courses.map((element, index) => {
              return (
                <FilterCheckbox
                  key={index}
                  element={element}
                  state={state}
                  handleCheckChange={handleCheckChange}
                />

```

```

    );
  }}}
</Box>
  )}
  {selectedValue === "semester" && (
    <Box sx={{ ...styles.block, margin: "10px 0" }}>
      {semesters.map((element, index) => {
        return (
          <FilterCheckbox
            key={index}
            element={element}
            state={state}
            handleCheckChange={handleCheckChange}
          />
        );
      })}
    </Box>
  )}
</>
  )}
</Box>
);
};
const courses = [
  { course: "first", numb: 1 },
  { course: "second", numb: 2 },
  { course: "third", numb: 3 },
  { course: "four", numb: 4 },
];
const semesters = [
  { course: "first", numb: 1 },
  { course: "second", numb: 2 },
  { course: "third", numb: 3 },
  { course: "four", numb: 4 },
  { course: "five", numb: 5 },
  { course: "six", numb: 6 },
  { course: "seven", numb: 7 },
  { course: "eight", numb: 8 },
  { course: "nine", numb: 9 },
  { course: "ten", numb: 10 },
];
export default Filters;

```

```

import React, { useState, useEffect } from "react";
import PageTitle from "../components/pageTitle/PageTitle";
import PropTypes from "prop-types";
import SimpleAccordion from "../components/accordion/RegularAccordion";
import { Tabs, Tab, Box, Typography } from "@mui/material";
import SelectedAccordion from "../components/accordion/SelectedAccordion";
import axios from "axios";
import ChoiceSelectedDisciplines from "../components/choiceSelectedDisciplines/ChoiceSelectedDisciplines";
import Filters from "../components/filters/Filters";
function TabPanel(props) {
  const { children, value, index, ...other } = props;
  return (
    <div
      role="tabpanel"
      hidden={value !== index}
      id={`simple-tabpanel-${index}`}
      aria-labelledby={`simple-tab-${index}`}
      {...other}
    >
      {value === index && <Box sx={{ marginTop: "20px" }}>{children}</Box>}
    </div>
  );
}
TabPanel.propTypes = {
  children: PropTypes.node,
  index: PropTypes.number.isRequired,
  value: PropTypes.number.isRequired,
};
function allProps(index) {
  return {
    id: `simple-tab-${index}`,
    "aria-controls": `simple-tabpanel-${index}`,
  };
}
const InfoText = () => {
  return (
    <Typography
      sx={{
        color: "#0188d1",
        textAlign: "center",
        marginBottom: "40px",
        fontSize: "14px",
      }}
    >

```

```

        fontWeight: 600,
      }}
    >
    Обрати дисципліни можна на вкладці Вибіркові
  </Typography>
);
};
const Discipline = () => {
  const disciplineLabels = ["Усі дисципліни", "Нормативні", "Вибіркові"];
  const [value, setValue] = useState(0);
  const [disciplines, setDisciplines] = useState([]);
  const [filteredRegularPart, setRegularPart] = useState([]);
  const [filteredSelectedPart, setSelectedPart] = useState([]);
  useEffect(() => {
    axios
      .get("http://localhost:5001/api/disciplines?group=КНТ-1")
      .then((res) => {
        const data = res.data;
        setDisciplines(data);
        setRegularPart(
          data.filter((item) => item.part === "НЧ / НОРМАТИВНА ЧАСТИНА")
        );
        setSelectedPart(
          data.filter((item) => item.part === "ВЧ / ВИБІРКОВА ЧАСТИНА")
        );
      })
      .catch((err) => console.log(err));
  }, []);
  const handleChange = (event, newValue) => {
    setValue(newValue);
  };
  let [count, setCount] = useState(0);
  let [arr, setArr] = useState([]);
  return (
    <section className="section">
      <PageTitle pageTitle={"Дисципліни"} />
      <Box sx={{ borderBottom: 1, borderColor: "divider" }}>
        <Tabs
          value={value}
          onChange={handleChange}
          aria-label="basic tabs example"
        >

```

```

    {disciplineLabels.map((label, index) => (
      <Tab label={label} {...allProps(index)} key={index} />
    ))}
  </Tabs>
</Box>
<TabPanel value={value} index={0}>
  <InfoText />
  <Filters />
  {disciplines.map((item, index) => {
    return <SimpleAccordion key={index} item={item} />;
  })}
</TabPanel>
<TabPanel value={value} index={1}>
  <InfoText />
  <Filters />
  {filteredRegularPart.map((item, index) => {
    return <SimpleAccordion key={index} item={item} />;
  })}
</TabPanel>
<TabPanel value={value} index={2}>
  <Filters />
  {filteredSelectedPart.map((item, index) => {
    return (
      <SelectedAccordion
        key={index}
        item={item}
        setCount={setCount}
        count={count}
        arr={arr}
        setArr={setArr}
      />
    );
  })}
  <ChoiceSelectedDisciplines count={count} />
</TabPanel>
</section>
);
};
export default Discipline;
import React from "react";
import ExpandMoreIcon from "@mui/icons-material/ExpandMore";
import {

```

```

Typography,
Grid,
Accordion,
AccordionSummary,
AccordionDetails,
} from "@mui/material";
const SimpleAccordion = ({ item }) => {
  let competition = item.competition;
  return (
    <Accordion>
      <AccordionSummary expandIcon={<ExpandMoreIcon />}>
        <Grid sx={{ width: "40%", paddingLeft: "20px" }}>
          <Typography>
            {item.title.length > 20
              ? item.title.substr(0, 24) + "..."
              : item.title}
          </Typography>
        </Grid>
        <Typography>{item.part}</Typography>
      </AccordionSummary>
      <AccordionDetails>
        <Typography>{item.title}</Typography>
        <Typography>{item.part}</Typography>
        <Typography>{item.education_cycle}</Typography>
        <Typography>Кредитів: {item.credits}</Typography>
        <Typography>Годин: {item.hours}</Typography>
        <Typography>Семестр: {item.semester}</Typography>
        {competition === "з" ? (
          <Typography>{"залік"}</Typography>
        ) : (
          <Typography>{"екзамен"}</Typography>
        )}
      </AccordionDetails>
    </Accordion>
  );
};
export default SimpleAccordion;
import React, { useState } from "react";
import ExpandMoreIcon from "@mui/icons-material/ExpandMore";
import {
  Checkbox,
  Typography,

```

```

Grid,
Accordion,
AccordionSummary,
AccordionDetails,
} from "@mui/material";
const SelectedAccordion = ({ item, count, setCount, arr, setArr }) => {
  let competition = item.competition;
  const [checked, setChecked] = useState(false);
  function handleClickCheckbox(e) {
    e.stopPropagation();
  }
  const handleChecked = () => {
    setChecked(!checked);
    if (!checked) {
      setCount(count + 1);
    } else {
      setCount(count - 1);
    }
  };
  return (
    <Accordion>
      <AccordionSummary expandIcon={<ExpandMoreIcon />}>
        <Grid>
          <Checkbox
            value={checked}
            color="primary"
            onChange={() => handleChecked()}
            onClick={e => handleClickCheckbox(e)}
            sx={styles.checkbox}
          />
        </Grid>
        <Grid sx={{ width: "40%", paddingLeft: "20px" }}>
          <Typography>
            {item.title.length > 20
              ? item.title.substr(0, 24) + "..."
              : item.title}
          </Typography>
        </Grid>
        <Typography>{item.part}</Typography>
      </AccordionSummary>
      <AccordionDetails>
        <Typography>{item.title}</Typography>
      </AccordionDetails>
    </Accordion>
  );
};

```

```

<Typography>{item.part}</Typography>
<Typography>{item.education_cycle}</Typography>
<Typography>Кредитів: {item.credits}</Typography>
<Typography>Годин: {item.hours}</Typography>
<Typography>Семестр: {item.semester}</Typography>
{competition === "з" ? (
  <Typography>{"залік"}</Typography>
) : (
  <Typography>{"екзамен"}</Typography>
)}
</AccordionDetails>
</Accordion>
);
};
const styles = {
  checkbox: {
    "&.MuiCheckbox-root": {
      padding: "0px",
    },
  },
};
export default SelectedAccordion;

```

ДОДАТОК Д
Слайди презентації

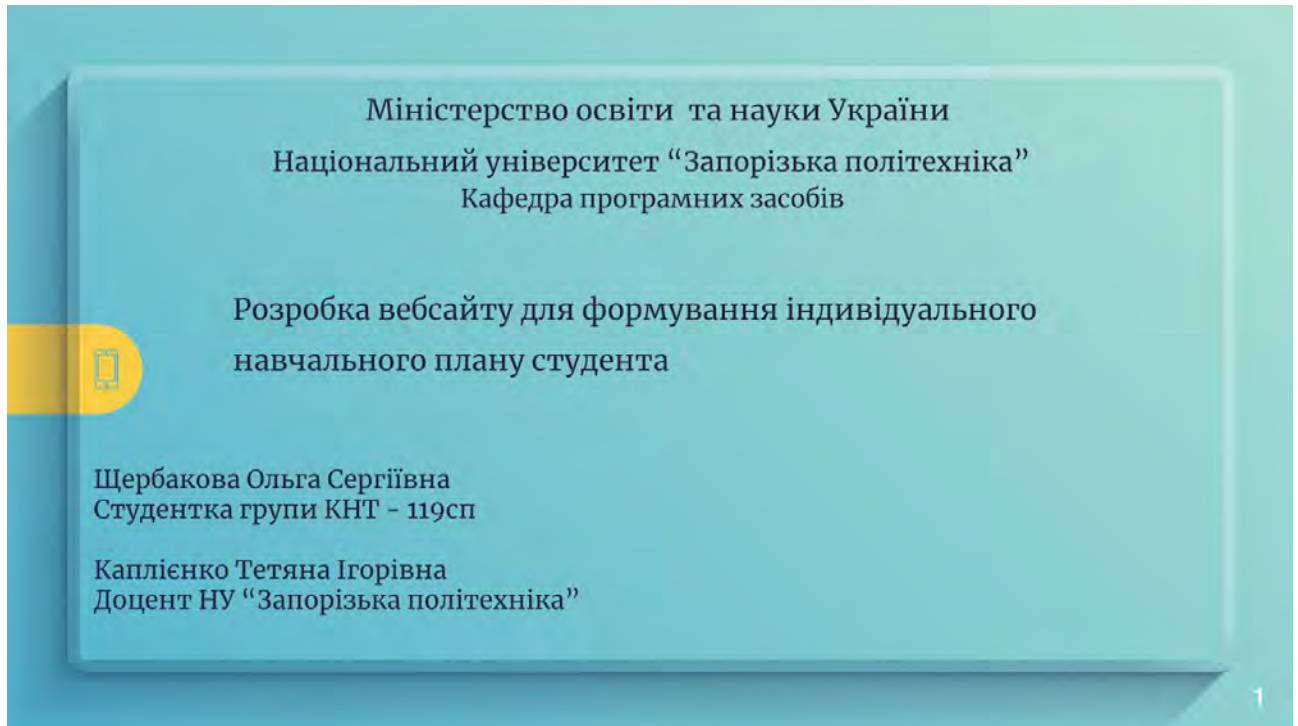


Рисунок Д.1 – Слайд 1

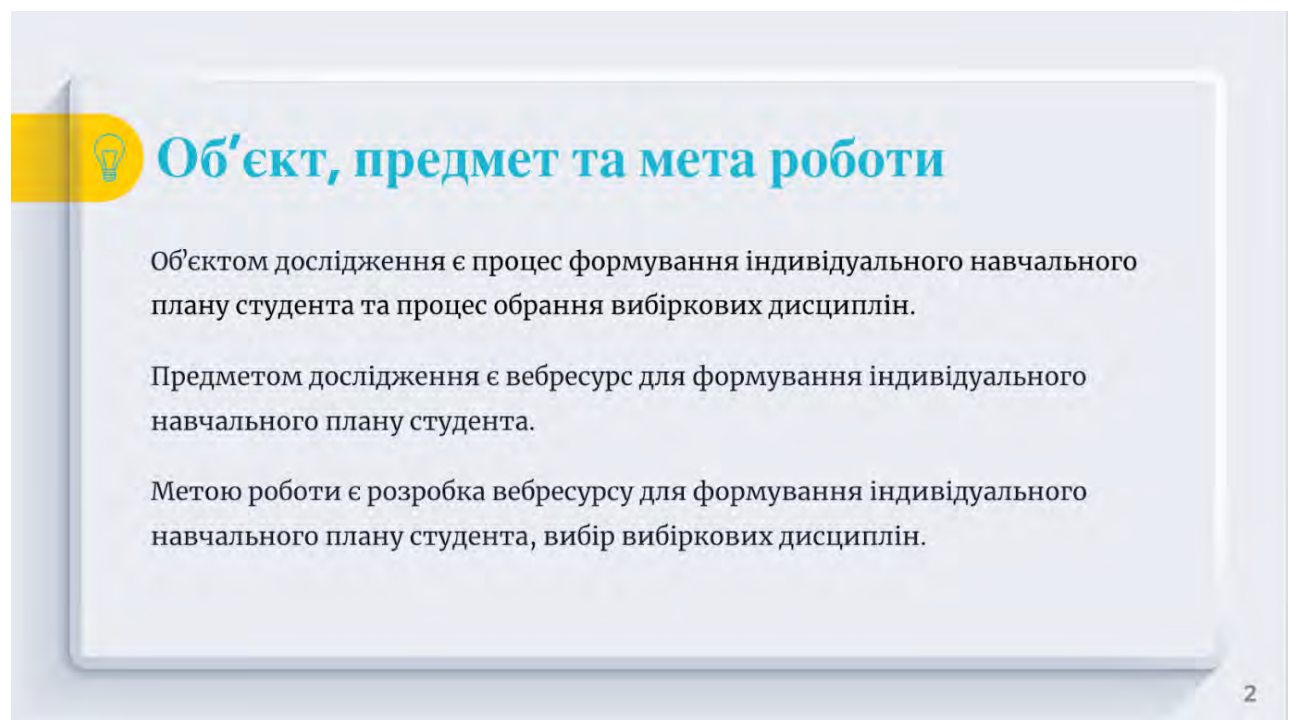


Рисунок Д.2 – Слайд 2

Завдання

Завдання до дипломної роботи:

- Аналіз предметної області та порівняння існуючих аналогів;
- Проектування програмного забезпечення у вигляді вебсайту;
- Розробка вебсайту для формування індивідуального навчального плану студента;
- Тестування розробленого програмного забезпечення.

3

Рисунок Д.3 – Слайд 3

Порівняння існуючих аналогів

Критерій порівняння	Google Classroom	Moodle	SMLS
Безкоштовно	+	+	–
Журнал оцінок	+	+	+
Формування звітів	–	–	+
Формування індивідуального навчального плану	–	–	–

4

Рисунок Д.4 – Слайд 4

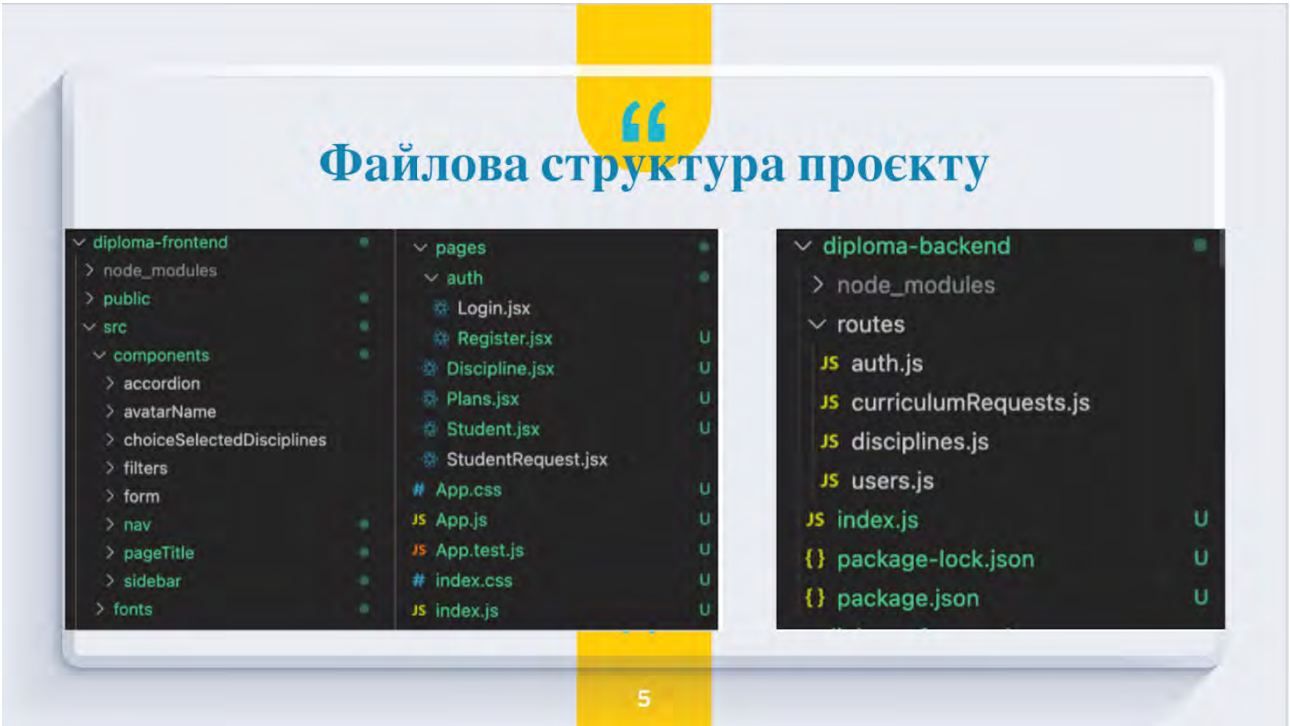


Рисунок Д.5 – Слайд 5



Рисунок Д.6 – Слайд 6

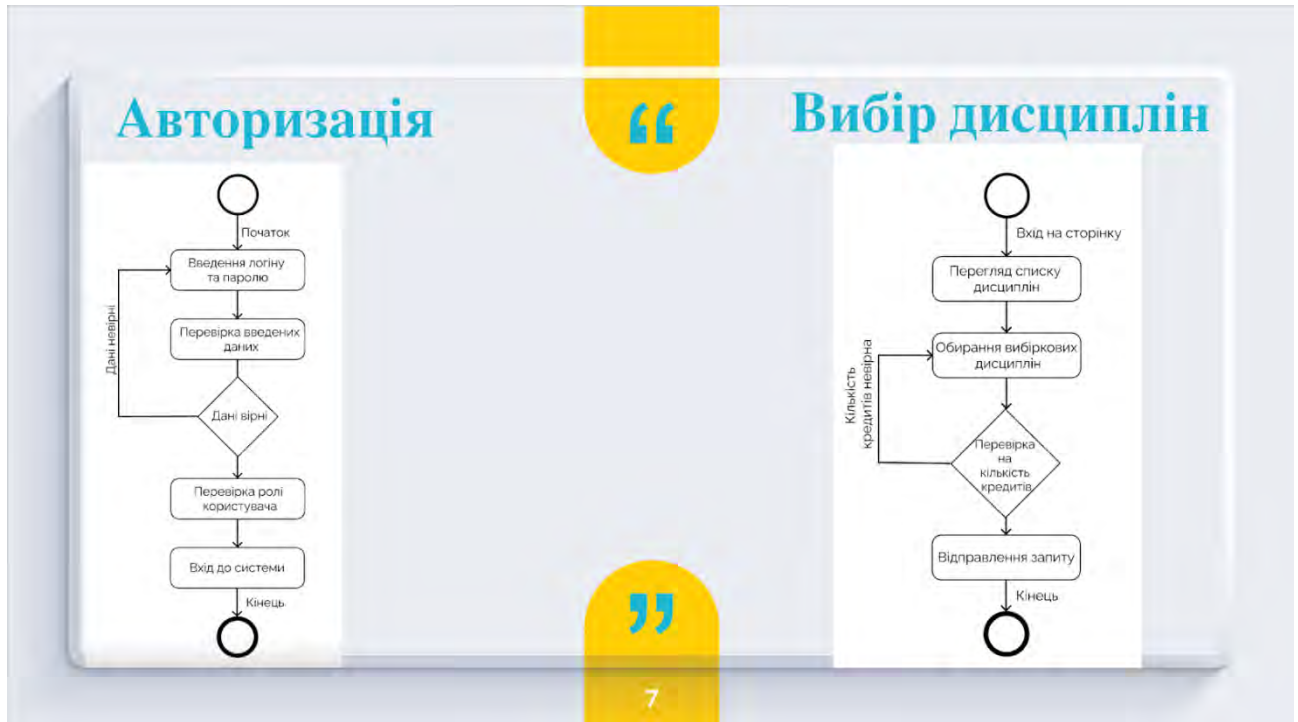


Рисунок Д.7 – Слайд 7

Вибір надбудови мови програмування frontend

Критерії	React	Vue
Virtual DOM	+	+
TypeScript підтримка	++	+
Проста міграція версій	+	+
Легка масштабованість	+	–
Маршрутизація	+	+
Повторне використання компонентів	+	+
Спільнота та підтримка	++	+
Документація	+	++
Вибір гнучких бібліотек	+	–

Рисунок Д.8 – Слайд 8

Вибір надбудови мови програмування backend

Критерії	Express JS	NestJS
Гнучкий	+	+
Складна структура	–	+
Легкість використання	+	–
Високошвидкісні операції введення-виводу	+	+

9

Рисунок Д.9 – Слайд 9

Вибір середовища розробки

Критерії	VS Code	WebStorm	Atom
Безкоштовний	+	–	+
Багатоплатформність	+	+	+
Автодоповнення	+	+	+
Підсвітка коду	+	+	+
Продуктивність	++	–	+
Інтеграція git	+	+	+
Навігація по коду	+	+	+
Вбудований термінал	+	+	+
Легкість програми	+	–	+
Зовнішній вигляд	+	–	–

10

Рисунок Д.10 – Слайд 10

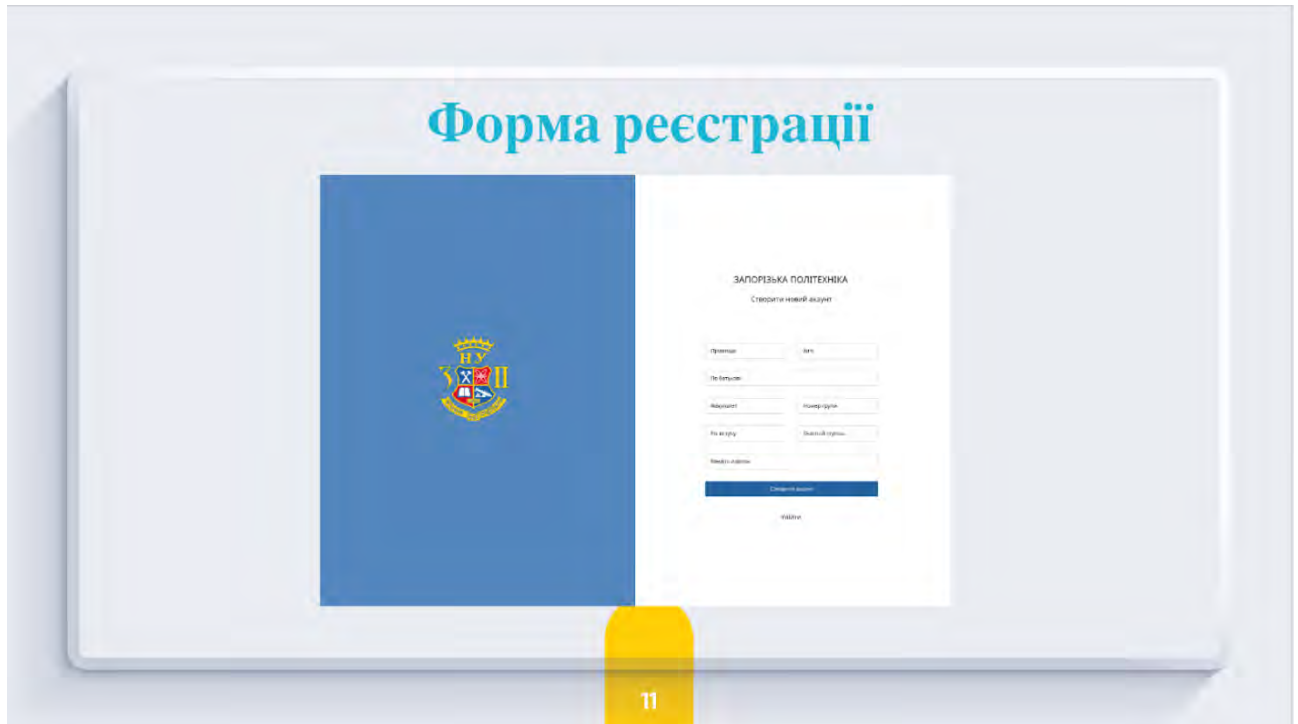


Рисунок Д.11 – Слайд 11

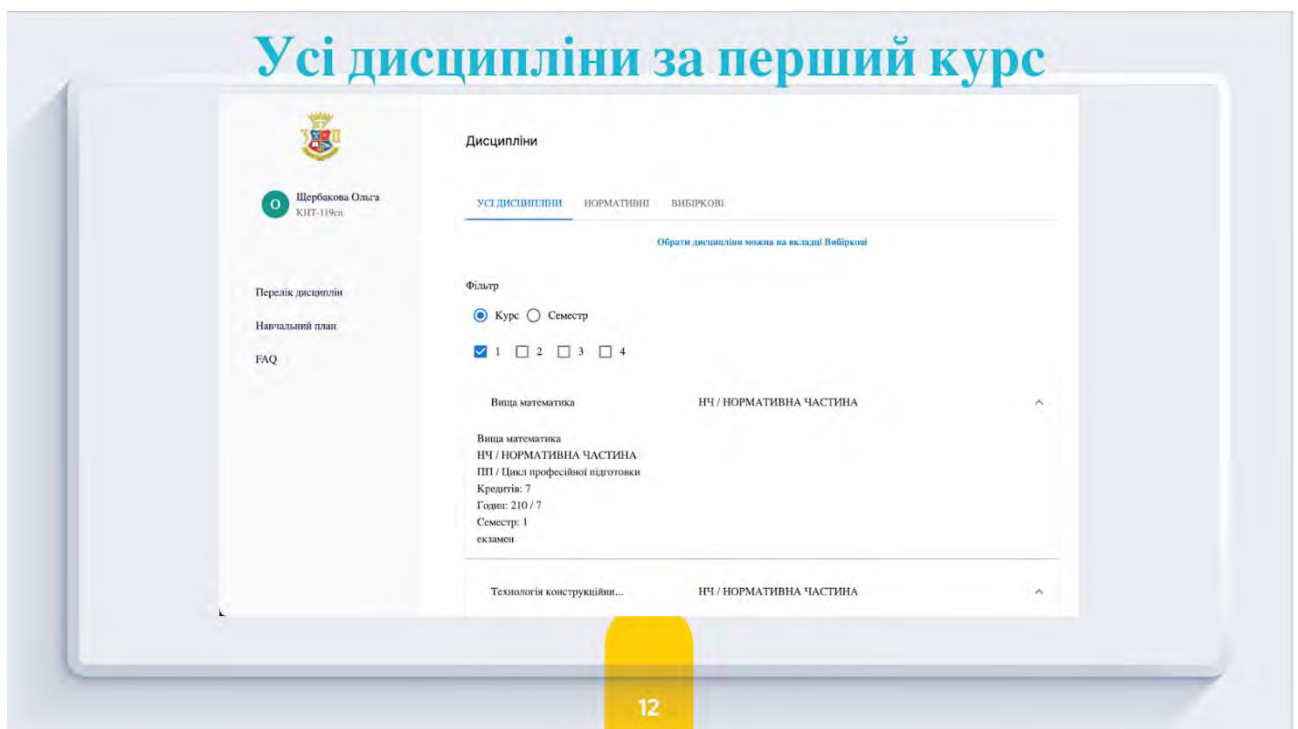


Рисунок Д.12 – Слайд 12

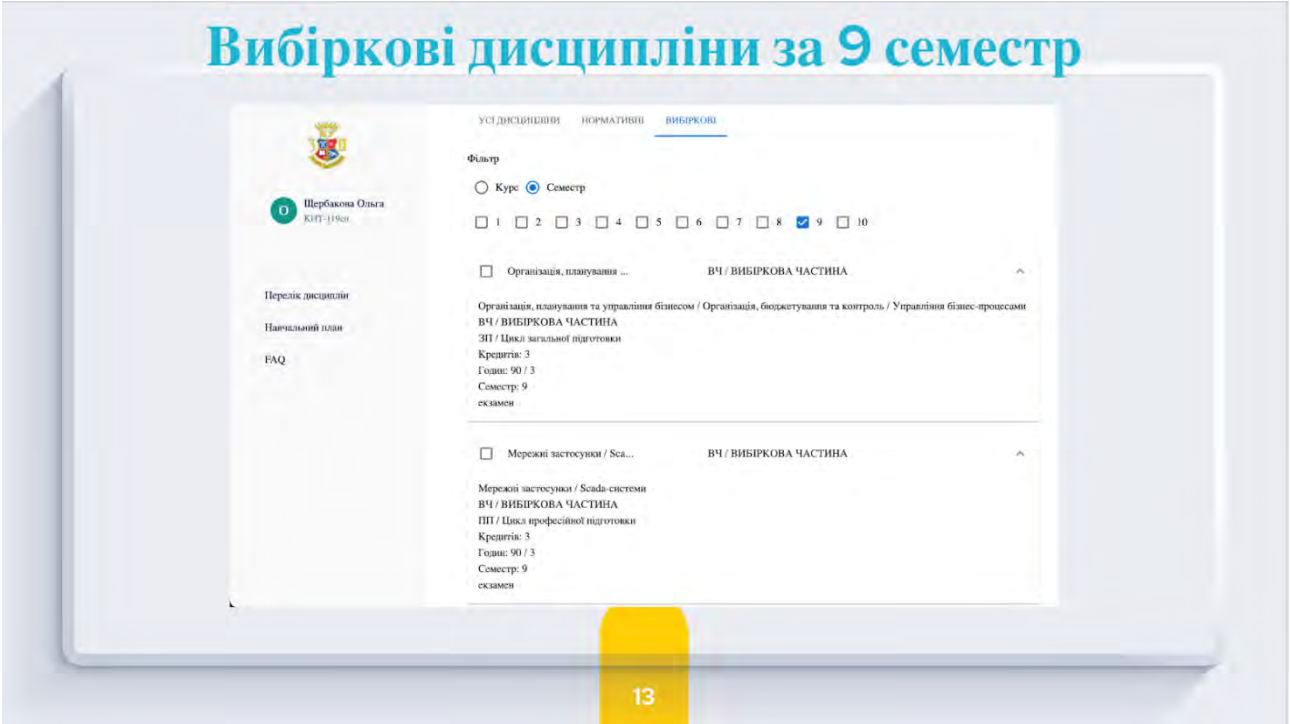


Рисунок Д.13 – Слайд 13

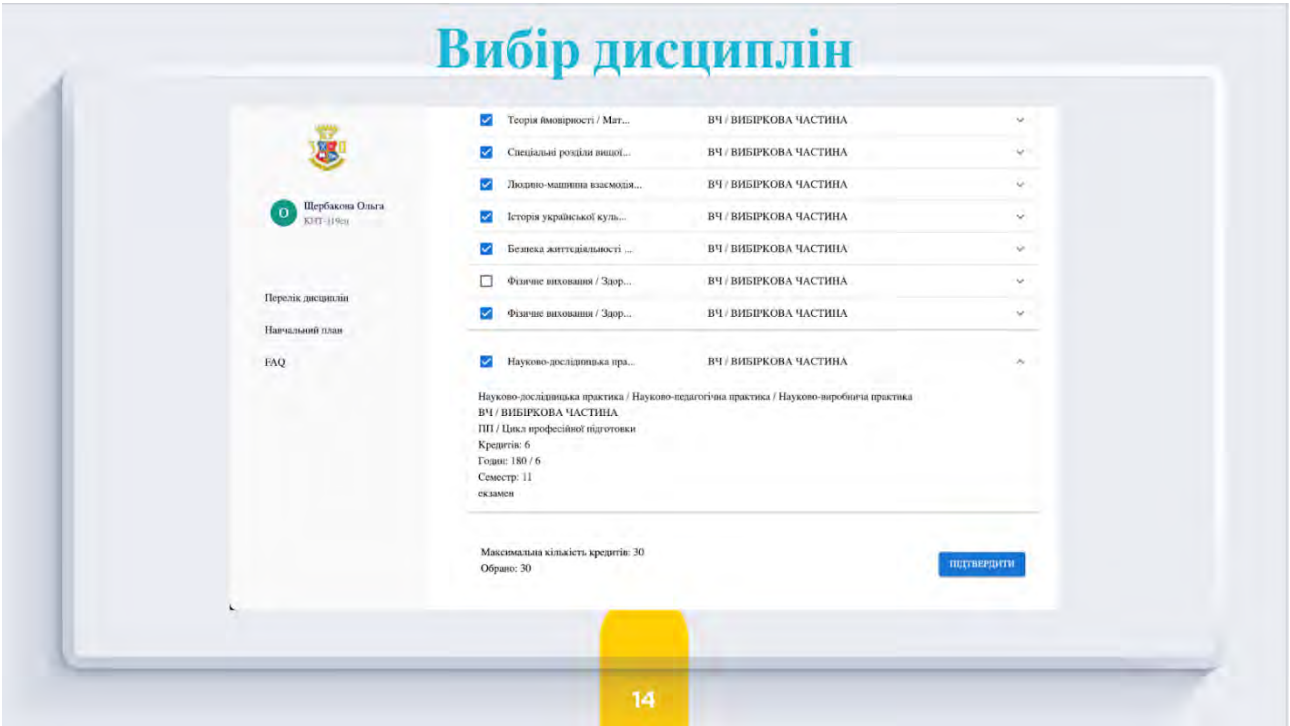


Рисунок Д.14 – Слайд 14

Висновки

В ході виконання дипломної роботи бакалавра було проаналізовано та досліджено процес розробки ПЗ для формування навчального плану студента. Розроблено ПЗ у вигляді веб ресурсу для надання інформації про навчальний план та забезпечує виконання таких функцій:



- Автентифікація користувача;
- Перегляд всіх типів дисциплін;
- Фільтрація дисциплін;
- Запит на обрання вибіркового дисциплін;
- Підтвердження або відхилення запиту.

При тестуванні системи було виявлено, що система працює коректно та може використовуватися за призначенням

Рисунок Д.15 – Слайд 15