Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота магістра

на тему: «Веб-застосунок для автоматизації проведення опитування здобувачів вищої освіти щодо якості викладання освітніх компонент»

Виконав: студент групи К23-2М

Спеціальність 122 Комп'ютерні науки

Назарян А. А.
(прізвище та ініціали)

Керівник к.т.н., доц. Ульяновська Ю. В.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рецензент Університет митної справи та фінансів
(місце роботи)

Начальник навчально-методичного відділу забезпечення якості освіти та акредитації
(посада)

к.т.н., доц. Свинаренко Д. М.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2025

АНОТАЦІЯ

Назарян А. А. Веб-застосунок для автоматизації проведення опитування здобувачів вищої освіти щодо якості викладання освітніх компонент.

Дипломна робота на здобуття освітнього ступеня магістр за спеціальністю 122 «Комп'ютерні наук». – Університет митної справи та фінансів, Дніпро, 2024.

Об'єктом дослідження є процес збору статистики за результатами опитувань студентів.

Предмет дослідження – розробка веб-застосунку для автоматизації процесу збору, обробки та аналізу статистичних даних студентських опитувань.

Метою роботи є створення веб-застосунку, який полегшує процес оцінювання викладачів студентами через автоматизацію збору та аналізу даних, забезпечуючи зручність у проведенні анонімних опитувань та перегляді детальної статистики.

Дана робота присвячена розробці веб-застосунку для організації та полегшення процесу збору даних щодо якості роботи викладачів у навчальному закладі. Дослідження акцентує увагу на створенні інструментів для зручної фільтрації даних, що дозволяє студентам обирати освітній рівень, курс, форму навчання, дисципліну та викладача. Адміністраторська панель забезпечує управління переліком дисциплін, викладачів, освітніх програм та форм опитувань.

Окремий акцент зроблено на реалізації функціоналу для перегляду детальної статистики, яка дозволяє аналізувати дані опитувань за викладачами, дисциплінами чи іншими параметрами. Система забезпечує можливість зручного пошуку викладачів за прізвищем, ім'ям та по батькові, що значно підвищує ефективність роботи та якість прийняття рішень.

Ключові слова: веб-застосунок, студентські опитування, оцінювання викладачів, збір статистики, автоматизація.

Список публікацій здобувача:

1. Лебідь О. Ю., Назарян А. А. Порівняння основних механізмів об’єктно-орієнтованого програмування в PHP та JavaScript. Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення. Випуск 55. Частина 1. Тернопіль: Міжнародна наукова інтернет-конференція, 2020, с. 67-68. URL: http://www.konferenciaonline.org.ua/data/downloads/file_1638480791.pdf

2. Ульяновська Ю. В., Назарян А. А. Розробка веб-застосунку центру інформаційних технологій Університету митної справи та фінансів. Економіко-правові та управлінсько-технологічні виміри сьогодення: молодіжний погляд: матеріали міжнародної науково-практичної конференції: у 3 т. Том 3. Дніпро: Університет митної справи та фінансів, 2023, с. 270-272. URL: https://drive.google.com/file/d/1JEG4IPGzqAcDAI1kyZ2XxE53f8FV01DQ/view

3. Лебідь О. Ю., Назарян А. А., Рябоволенко Е. А. Графові бази даних та їх роль у візуалізації та аналізі даних. Економіко-правові та управлінсько-технологічні виміри сьогодення: молодіжний погляд: матеріали міжнародної науково-практичної конференції: у 3 т. Том 3. Дніпро: Університет митної справи та фінансів, 2023, с. 272-274. URL: https://drive.google.com/file/d/1JEG4IPGzqAcDAI1kyZ2XxE53f8FV01DQ/view

4. Parshyna O. A., Nazarian A. A. Methods of cryptographic data protection. Economic-legal and managerial-technological dimensions of the present: a youth perspective: materials of the international scientific and practical conference: in 3 volumes. Vol 3. Dnipro: Dnipro: University of Customs and Finance, 2023, p. 227-228. URL: https://drive.google.com/file/d/1JEG4IPGzqAcDAI1kyZ2XxE53f8FV01DQ/view

ABSTRACT

Nazarian A. A. Development of a web application to facilitate the collection of statistics on professors.

Diploma thesis (project) for obtaining a master's degree in speciality 122 "Computer Science." - University of Customs and Finance, Dnipro, 2024.

The object of research is the process of collecting statistics based on the results of student surveys.

The subject of the study is the development of a web application for automating the process of collecting, processing and analysing statistical data from student surveys.

The purpose of the study is to create a web application that facilitates the process of student evaluation of professors by automating data collection and analysis, providing convenience in conducting anonymous surveys and viewing detailed statistics.

This paper is devoted to the development of a web application to organise and facilitate the process of collecting data on the quality of professors' work in an educational institution. The study focuses on creating tools for convenient data filtering, allowing students to choose an educational level, course, form of study, discipline and professor. The administrative panel provides management of the list of disciplines, professors, educational programmes and survey forms.

A special emphasis is placed on the implementation of functionality for viewing detailed statistics that allows you to analyse survey data by professors, disciplines or other parameters. The system provides a convenient search for professors by name, surname and patronymic, which significantly improves work efficiency and decision-making.

Keywords: web application, student surveys, professor evaluation, statistics collection, automation.

LIST OF PUBLICATIONS:

1. Lebid O. Y., Nazarian A. A. Comparison of the main mechanisms of object-oriented programming in PHP and JavaScript. Information society: technological, economic and technical aspects of formation. Issue 55. Part 1: Ternopil: International scientific Internet conference., 2020, p. 67-68. URL: http://www.konferenciaonline.org.ua/data/downloads/file_1638480791.pdf

2. Ulianovska Y. V., Nazarian A. A. Development of a web application of the Information Technology Center of the University of Customs and Finance. Economic, legal and managerial-technological dimensions of the present: a youthful view: materials of the international scientific and practical conference: in 3 volumes. Vol. 3. Dnipro: University of Customs and Finance, 2023, p. 270-272. URL: https://drive.google.com/file/d/1JEG4IPGzqAcDAI1kyZ2XxE53f8FV01DQ/view

3. Lebid O. Y., Nazarian A. A., Ryabovolenko E. A. Graph databases and their role in data visualization and analysis. Economic, legal and managerial-technological dimensions of the present: a youth perspective: materials of the international scientific and practical conference: in 3 volumes. Vol. 3. Dnipro: University of Customs and Finance, 2023, p. 272-274. URL: https://drive.google.com/file/d/1JEG4IPGzqAcDAI1kyZ2XxE53f8FV01DQ/view

4. Parshyna O. A., Nazarian A. A. Methods of cryptographic data protection. Economic-legal and managerial-technological dimensions of the present: a youth perspective: materials of the international scientific and practical conference: in 3 volumes. Vol 3. Dnipro: Dnipro: University of Customs and Finance, 2023, p. 227-228. URL: https://drive.google.com/file/d/1JEG4IPGzqAcDAI1kyZ2XxE53f8FV01DQ/view

ЗМІСТ

ВСТУП

Актуальність дослідження. У сучасних умовах розвитку інформаційних технологій важливим аспектом є створення ефективних інструментів для збору, обробки та аналізу даних, що дозволяють приймати обґрунтовані рішення. Статистичні дані, отримані за допомогою опитувань, є потужним джерелом інформації для оцінки різних процесів. Водночас універсальні платформи, такі як Google Forms, не завжди здатні забезпечити необхідний рівень адаптивності та функціональності для специфічних задач. Це створює потребу у розробці спеціалізованих рішень, які б відповідали конкретним вимогам.

Інноваційність проекту полягає у спеціалізованій інформаційній моделі, яка враховує специфічні потреби освітнього процесу. Вона дозволяє структурувати дані опитувань таким чином, щоб забезпечити адаптивність до змін у навчальних програмах, дисциплінах чи організаційних процесах. Зокрема, кожна сутність, така як викладач, курс, освітня програма чи форма навчання, має зв’язки з іншими об'єктами системи, що дозволяє легко отримувати аналітичні дані для прийняття рішень.

З точки зору алгоритмічних рішень було розроблено оптимізований підхід до обробки даних у реальному часі. Наприклад, під час проведення опитувань система автоматично агрегує результати та візуалізує їх для адміністрації. Це стало можливим завдяки інтеграції з високопродуктивним бекендом на основі Node.js та використанням RESTful API, що забезпечує швидкий обмін даними між клієнтською і серверною частинами.

Дотримання стандартів програмного забезпечення, таких як ISO/IEC 25010, стало основою для визначення вимог до функціональності, продуктивності, зручності використання та безпеки. Для клієнтської частини було обрано Vue.js завдяки його модульній структурі та підтримці реактивності, що дозволяє створювати динамічні інтерфейси для різних категорій користувачів — студентів та адміністраторів. Використання SCSS у поєднанні з

Vuetify дозволило створити сучасний адаптивний дизайн із фокусом на доступність і зручність.

Що стосується нововведень, система також забезпечує унікальну функцію автоматичного оцінювання показників задоволеності студентів за допомогою вагових коефіцієнтів, які можна налаштовувати. Наприклад, оцінки студентів можуть автоматично враховувати рівень важливості конкретних дисциплін, що дозволяє формувати об'єктивну статистику.

Вимоги до системи включали інтеграцію з базою даних, яка динамічно масштабується. Це було досягнуто завдяки використанню ORM (TypeORM), що значно спрощує розробку та забезпечує відповідність стандартам безпеки даних, таким як GDPR.

Загалом, розроблений веб-застосунок не лише вирішує існуючі проблеми універсальних платформ, але й пропонує унікальні можливості для підвищення якості освітнього процесу. Це робить його незамінним інструментом для прийняття управлінських рішень і побудови прозорих комунікацій між адміністрацією та студентами.

Мета роботи – розробка веб-застосунку, що дозволяє автоматизувати процес збору, обробки та аналізу статистики за результатами опитувань студентів, із можливістю гнучкої фільтрації та зручного доступу до даних.

Методи дослідження – методи проектування, розробки програмного забезпечення, метод теорії інформації, обробка та аналіз інформації.

У відповідності до поставленої мети в кваліфікаційній роботі ставились та вирішувались наступні завдання дослідження:

1. Проаналізувати технічні засоби, що використовуються для розробки, та обрати необхідні для створення програмного забезпечення веб-застосунку для автоматизації проведення опитування здобувачів вищої освіти щодо якості викладання освітніх компонент.

2. Розробити вимоги до програмного забезпечення для веб-застосунку для автоматизації проведення опитування здобувачів вищої освіти щодо якості

викладання освітніх компонент на основі аналізу переваг та недоліків існуючих систем.

3. Спроектувати та розробити новий застосунок на основі аналізу потреб користувачів.

4. Провести тестування.

Об'єкт дослідження – процес збору та аналізу даних, отриманих у результаті студентських опитувань.

Предмет дослідження – програмно-апаратне забезпечення для автоматизації збору статистики студентських опитувань.

Практичне значення одержаних результатів – розроблений веб-застосунок дозволить автоматизувати процес збору та аналізу статистичних даних, підвищити ефективність роботи адміністрації навчального закладу, покращити якість освітнього процесу та підвищити залученість студентів до оцінювання викладачів.

Результати роботи – створено сучасний веб-застосунок для автоматизації збору та обробки статистики опитувань із функціональністю адміністрування, фільтрації та перегляду даних.

Робота складається зі вступу, 3-х розділів, висновків, списку використаних джерел з 26 найменувань, 3-х додатків. Обсяг роботи 71 сторінок основного тексту з 91 сторінок кваліфікаційної роботи, 26 рисунків, 2 таблиці, 6 формул.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Аналіз публікацій щодо розробки веб-застосунків

У теперішній час розробка веб-застосунків набуває все більшої популярності, оскільки майже кожен аспект сучасного життя залежить від цифрових технологій. Веб-додатки використовуються в різноманітних сферах: від бізнесу до освіти, медицини та розваг. Однак, при розробці таких застосунків, особливо при проектуванні їхньої архітектури, дуже важливо враховувати вибір правильних технологій та підходів, які дозволяють не тільки забезпечити ефективність роботи програми, але й її масштабованість і безпеку.

Одним із таких важливих інструментів є фреймворки для розробки веб-застосунків, які значно полегшують роботу розробників. Фреймворки дозволяють мінімізувати необхідність написання повторюваного коду та фокусуватись на реалізації основної бізнес-логіки програми. Згідно з дослідженнями, фреймворки можуть включати різні інструменти і допоміжні програми, що значно підвищує швидкість розробки і допомагає уникнути багатьох помилок [3]. Наприклад, при створенні веб-застосунків для бізнес-процесів або електронної комерції використання фреймворків допомагає зменшити витрати часу на розробку та впровадження. У той же час, вибір неправильного фреймворку може стати однією з причин невдачі проекту. Тому так важливо ретельно підходити до вибору фреймворку, оскільки кожен з них має свої сильні та слабкі сторони.

Не менш важливою частиною проектування веб-застосунків є архітектура. Архітектура веб-застосунку має безпосередній вплив на його функціональність, безпеку і масштабованість. Це питання було детально розглянуто у роботі [4], у якій дослідники запропонували ефективний підхід до проектування архітектури веб-застосунку для імітаційного моделювання управління транспортними потоками. Вони акцентують увагу на необхідності забезпечення не тільки

ефективної роботи системи, але й на безпеці даних. Безпека веб-застосунків стає однією з найбільш актуальних проблем, особливо у світлі зростаючої кількості кібератак. Дослідження показують, що для досягнення високої безпеки системи необхідно впроваджувати сучасні методи захисту, такі як використання шифрування, механізмів автентифікації та авторизації, а також заходів проти SQL-ін'єкцій.

Важливою складовою проектування веб-застосунку є також розподілені інформаційні системи. Зінченко А.Ю. у своїй роботі пропонує використання технології слабозв'язаних компонентів для створення гнучких і масштабованих систем. Згідно з його дослідженням, цей підхід дозволяє не тільки підвищити безпеку веб-застосунків, але й зробити їх більш стійкими до помилок та збоїв, оскільки компоненти можуть працювати незалежно один від одного, що знижує ймовірність системних помилок [1].

Проектування бази даних також є важливим аспектом при створенні веб-застосунків. У роботі [2] автор вказує на необхідність ретельного підходу до проектування бази даних інтелектуальних систем діагностики захворювань, де зберігаються великі обсяги чутливих медичних даних. Розробка такої бази повинна забезпечувати не тільки ефективне зберігання та обробку інформації, але й захист даних від несанкціонованого доступу. Окрім того, проектування баз даних має забезпечувати високий рівень доступності та швидкості роботи системи, що є важливим при створенні інтелектуальних систем, які використовуються в медичних установах для діагностики захворювань.

У роботі [5] вказують на важливість проектування бази даних для модуля студента в системі підтримки вивчення дисциплін.

1.2 Аналіз методів розробки веб-застосунків

Розробка веб-застосунків є однією з основних складових сучасної програмної інженерії, що об’єднує в собі не лише різноманітні технології, але й підходи до розробки, що дозволяють створювати ефективні, масштабовані та

безпечні веб-системи. Залежно від потреб бізнесу, складності проекту та технічних вимог, вибір методів розробки може варіюватися. У цій частині розглянемо методи розробки веб-застосунків, зокрема, використовуючи популярні технології, такі як Vue.js для фронтенду і NestJS для бекенду.

1.2.1 Методології розробки веб-застосунків

Процес розробки веб-застосунків є багатоступеневим і включає кілька етапів: від збору вимог і проектування до тестування та запуску продукту в експлуатацію. На кожному з цих етапів важливо враховувати специфіку веб-технологій та потреби користувачів, що дає змогу створити ефективне і зручне програмне забезпечення. Веб-застосунки часто створюються із використанням гнучких методологій, що дозволяють швидко адаптуватися до змінних умов і вимог. Одним із основних підходів є Agile, що передбачає розбиття проекту на короткі спринти. Кожен спринт фокусується на вирішенні конкретних завдань, з прицілом на постійну ітерацію та покращення продукту. Такий підхід дозволяє команді зберігати гнучкість і швидко реагувати на нові вимоги чи змінені обставини проекту.

Однією з ключових переваг Agile є можливість постійного поліпшення продукту на кожному етапі його розробки. Це дозволяє не лише задовольняти поточні вимоги користувачів, а й оперативно реагувати на зміну цих вимог, що особливо важливо в умовах швидко змінюваних технологій. Крім того, використання спринтів дозволяє командам частіше оцінювати хід роботи, роблячи коригування, якщо це необхідно. Усі ці переваги роблять Agile надзвичайно ефективним методом для веб-розробки, адже кожна нова версія продукту може значно покращувати попередні досягнення.

Іншою важливою методологією для сучасної веб-розробки є DevOps, який об’єднує процеси розробки та експлуатації програмного забезпечення. DevOps передбачає тісну інтеграцію між розробниками та операційними командами, що дозволяє автоматизувати багато етапів розробки і впровадження, таких як

деплоймент та тестування. Це дає змогу значно зменшити час, необхідний для випуску нових версій програмного забезпечення, і забезпечити високий рівень стабільності та безпеки продукту.

Завдяки таким інструментам, як Docker, Jenkins та Kubernetes, DevOps дозволяє створювати масштабовані та надійні веб-застосунки. Docker дозволяє упаковувати застосунки в контейнери, що забезпечує їх безпечну і стабільну роботу в будь-якому середовищі. Jenkins використовується для автоматизації процесу тестування та інтеграції, що дає можливість перевіряти нові функції та виправлення помилок в реальному часі. Kubernetes, в свою чергу, допомагає автоматизувати процеси оркестрації контейнерів, забезпечуючи масштабованість і високу доступність веб-застосунків.

Особливості Vue.js і NestJS надають ще більше можливостей для реалізації цих методологій. Vue.js дозволяє створювати гнучкі та інтерактивні інтерфейси користувача, використовуючи компоненти, що легко інтегруються в більші веб-застосунки. Завдяки своїй реактивності та зручному синтаксису, Vue.js є відмінним вибором для створення динамічних користувацьких інтерфейсів. NestJS, у свою чергу, використовує модульну архітектуру для організації серверної частини, що дозволяє ефективно масштабувати застосунок і легко підтримувати його в процесі розробки. Цей підхід дозволяє застосовувати всі переваги DevOps і Agile для швидкої та безпечної розробки, тестування і деплойменту (розгортання) веб-застосунків.

1.2.2 Архітектурні патерни (шаблони) для розробки веб-застосунків

Одним з найважливіших аспектів розробки є вибір архітектури системи. У сучасному веб-розробці широко застосовуються кілька архітектурних патернів, які дозволяють забезпечити високу продуктивність і масштабованість веб-застосунків. Найбільш популярними є:

1. Монолітна архітектура. Монолітна архітектура є класичним підходом до розробки, коли всі компоненти веб-застосунку інтегровані в одну систему. Вона забезпечує простоту розробки, оскільки всі функціональні частини знаходяться в одному кодовому базисі. У випадку використання NestJS, який забезпечує побудову модульних і структурованих додатків, розробники можуть організувати свій код таким чином, щоб навіть в рамках монолітного підходу, система була розділена на логічні модулі з чітким поділом відповідальності.

2. Мікросервісна архітектура дозволяє створювати незалежні сервіси, які спілкуються між собою через API. Кожен мікросервіс має свою власну відповідальність, базу даних та логіку. Це дозволяє зменшити вплив змін у одній частині системи на інші компоненти. У випадку NestJS мікросервіси можуть бути реалізовані через використання технології GraphQL або REST API, що дозволяє гнучко налаштовувати систему, підключаючи або відключаючи певні сервіси.

3. Архітектура на основі односпрямованих потоків даних. Для фронтенду веб-застосунків застосовуються архітектури, що базуються на односпрямованих потоках даних. Це дозволяє централізовано управляти станом додатку та гарантує стабільність і масштабованість веб-застосунку. Одним з популярних інструментів для реалізації такої архітектури є Vuex, який активно використовується у Vue.js для управління станом додатків. Vuex допомагає організувати та централізувати всі зміни в даних, забезпечуючи єдиний об'єкт стану для всієї програми. Це дозволяє зберігати логіку управління даними в одному місці, що полегшує їх відладку, тестування та підтримку. Водночас, Vuex інтегрується з компонентами, даючи можливість ефективно відслідковувати зміни в даних та оновлювати інтерфейс користувача без значних затримок.

4. Архітектура на основі подій. Цей патерн є ще одним підходом до побудови ефективних веб-застосунків, де компоненти взаємодіють між собою через події. Архітектура на основі подій часто використовується в розробці складних, масштабованих застосунків, де кожен компонент може слухати події,

що надходять від інших частин системи. Це дозволяє ізолювати компоненти, зменшуючи їхню залежність один від одного та підвищуючи рівень модульності. Один із прикладів — це використання EventBus у фреймворках на кшталт Vue.js, що дозволяє передавати події між компонентами без необхідності явно передавати дані через властивості чи методи. Цей підхід ефективно спрощує взаємодію між компонентами, а також дозволяє легше керувати станом додатку.

5. Архітектура на основі серверного рендерингу (SSR). Серверний рендеринг (SSR) передбачає, що контент веб-застосунку генерується на сервері, а не в браузері користувача. Це дозволяє значно прискорити початкове завантаження сторінки та покращити SEO (пошукову оптимізацію), оскільки пошукові системи можуть сканувати вже згенеровані сторінки.

1.2.3 Використання фреймворків для розробки веб-застосунків

При розробці веб-застосунків вибір фреймворку для фронтенду та бекенду є ключовим для забезпечення ефективності, швидкості розробки, а також подальшої підтримки та масштабованості системи. У сучасній веб-розробці існують різноманітні фреймворки, і вибір між ними залежить від вимог проекту та специфіки технологій.

Vue.js є одним з найбільш популярних і широко використовуваних фреймворків для фронтенд-розробки завдяки своїй легкості в освоєнні та потужним можливостям. Основною перевагою Vue.js є його реактивна архітектура, що дозволяє автоматично оновлювати інтерфейс при зміні даних без необхідності додаткових маніпуляцій з DOM. Це робить Vue.js ідеальним для створення складних, динамічних інтерфейсів, де важлива швидка взаємодія з користувачем. Vue.js також має просту синтаксичну структуру, що дозволяє швидко інтегрувати інші бібліотеки чи інструменти. Такі бібліотеки, як Vue Router для управління маршрутизацією та Vuex для глобального управління станом, сприяють організації та структуризації коду, що полегшує

масштабування додатків і робить код зрозумілішим та зручнішим для подальшої підтримки.

Використання Vue.js у комбінації з іншими інструментами дозволяє розробникам реалізовувати складні функції, зберігаючи при цьому високу гнучкість та ефективність. Завдяки компонентному підходу, Vue.js дозволяє створювати інтерфейси, які можна легко повторно використовувати в різних частинах програми, що істотно скорочує час розробки та підвищує ефективність роботи команди.

NestJS є фреймворком для бекенд-розробки на Node.js, який використовує найкращі практики об'єктно-орієнтованого програмування (ООП) та функціонального програмування. Однією з ключових особливостей NestJS є модульна архітектура, яка дозволяє розбивати додаток на окремі частини, що полегшує масштабування та забезпечує гнучкість при роботі з великими проектами. Важливою особливістю NestJS є ін’єкція залежностей, що допомагає знижувати зв’язність компонентів і полегшує тестування коду.

NestJS підтримує як традиційні методи створення REST API, так і більш сучасні технології, такі як GraphQL. Використання GraphQL дозволяє значно зменшити кількість запитів до серверу, оскільки клієнт може вказувати, які саме дані йому потрібні, що значно знижує навантаження на сервер і підвищує ефективність передачі даних. Завдяки такій можливості NestJS стає чудовим вибором для створення масштабованих та високопродуктивних веб-застосунків.

Використання цих двох технологій у комплексі дозволяє створювати надійні, ефективні та масштабовані веб-застосунки, що забезпечують високу швидкість взаємодії з користувачем.

1.2.4 Безпека веб-застосунків

Безпека веб-застосунків є однією з найважливіших складових їхнього функціонування, оскільки з кожним роком зростає кількість кібератак,

спрямованих на витягнення конфіденційних даних або маніпулювання ними. Веб-застосунки можуть стати мішенями для різноманітних видів атак, таких як SQL-ін’єкції, міжсайтові скрипти (XSS) та міжсайтові підробки запитів (CSRF). Їх мета – використати уразливості в системах безпеки, щоб здійснити несанкціонований доступ або модифікацію даних.

SQL-ін’єкція є однією з найпоширеніших форм атак на веб-застосунки, де зловмисник вставляє шкідливі SQL-запити у вхідні дані користувачів, що передаються до серверу. Це може дозволити зловмиснику змінювати, видаляти чи навіть отримувати доступ до чутливих даних. Щоб захиститися від таких атак, важливо застосовувати параметризовані запити, що дозволяє серверу чітко визначити, які частини запиту є даними, а які – частинами SQL-структури. Таким чином, зловмисники не можуть маніпулювати запитом, що забезпечує безпеку.

XSS-атаки (міжсайтові скрипти) відбуваються, коли зловмисник вбудовує шкідливий скрипт у веб-сторінку, яка потім виконується в браузері користувача. Це може призвести до викрадення сесійних куки-файлів, крадіжки даних або навіть перенаправлення на шкідливі сайти. Для захисту від таких атак Vue.js застосовує механізми екранування небезпечних символів під час відображення динамічних даних. Більш того, система Content Security Policy (CSP) дозволяє визначити політику завантаження скриптів і ресурсів, обмежуючи джерела, з яких ці ресурси можуть бути завантажені, що значно знижує ймовірність виконання неперевірених або шкідливих скриптів.

CSRF (міжсайтові підробки запитів) є ще однією поширеною атакою, де зловмисник використовує автентифікацію користувача на одному сайті для виконання небажаних запитів на іншому сайті. Наприклад, якщо користувач увійшов у свій акаунт на веб-застосунку, зловмисник може використати цей сеанс для передачі небезпечних запитів. Для запобігання таким атакам застосовують токени CSRF, які перевіряються при кожному запиті і дозволяють визначити, чи є запит легітимним.

У середовищі NestJS, для забезпечення безпеки використовуються різноманітні методи. Одним із найпоширеніших є використання JWT (JSON Web

Token), що дозволяє забезпечити безпечну передачу даних між сервером і клієнтом. JWT дозволяє створити захищену аутентифікацію, де кожен запит містить підписаний токен, який підтверджує, що запит походить від авторизованого користувача. Завдяки цьому, навіть при використанні відкритих каналів зв’язку, дані залишаються захищеними. Для додаткового контролю доступу NestJS використовує middleware, які можуть перевіряти права доступу користувачів та їх роль перед обробкою запиту, що додає ще один рівень безпеки.

На стороні фронтенду, захист від XSS-атак досягається через автоматичне екранування небезпечних символів у шаблонах Vue.js. Це означає, що навіть якщо вхідні дані містять шкідливі скрипти, вони не виконуються. Крім того, Content Security Policy (CSP) дозволяє встановлювати строгі правила для завантаження і виконання скриптів, що знижує ризик запуску неперевіреного коду на веб-сторінці.

У результаті, ефективний захист веб-застосунків потребує поєднання різних технологій та методів на всіх етапах розробки, від бекенду до фронтенду, щоб мінімізувати ризик атак і забезпечити безпеку даних та користувачів.

1.2.5 Використання баз даних

Використання баз даних є основною складовою більшості веб-застосунків, оскільки вони забезпечують збереження, обробку та доступ до даних, необхідних для роботи програми. Бази даних дозволяють ефективно організовувати великі об'єми інформації, а також забезпечувати швидкий доступ до неї при необхідності. Вибір типу бази даних (реляційна чи нереляційна), її архітектура, а також налаштування безпеки мають вирішальне значення для ефективності та безпеки веб-застосунку.

Основним розрізненням між типами баз даних є їх структура. Реляційні бази даних (RDBMS), такі як PostgreSQL, MySQL або Microsoft SQL Server,

використовують таблиці для зберігання даних і забезпечують можливість виконання складних запитів за допомогою SQL (Structured Query Language). Реляційні бази ідеально підходять для застосунків, які потребують збереження зв'язків між різними наборами даних (наприклад, для управління інформацією про користувачів та їх замовлення).

Нереляційні бази даних, такі як MongoDB або Redis, використовують інші методи збереження даних, зокрема документи або ключ-значення. Вони є більш гнучкими у зберіганні неструктурованих або напівструктурованих даних, що робить їх ідеальними для швидкого масштабування та роботи з великими обсягами даних, які не вимагають складних взаємозв'язків.

Одним з основних аспектів роботи з базами даних є забезпечення їх безпеки. Основні загрози для баз даних включають SQL-ін’єкції, доступ до чутливих даних без належної авторизації, а також витік даних через незахищені канали зв'язку. Для захисту від SQL-ін’єкцій важливо використовувати параметризовані запити та підготовлені вирази, що дозволяють уникнути введення шкідливих SQL-команд через форму або інтерфейс користувача.

Більше того, для забезпечення безпеки даних слід використовувати шифрування даних як під час збереження, так і при їх передачі через мережу. Шифрування дозволяє захистити конфіденційні дані від несанкціонованого доступу навіть у разі витоку чи крадіжки даних. Крім того, важливо організувати автентифікацію та авторизацію на рівні бази даних, щоб гарантувати, що лише користувачі з відповідними правами можуть виконувати операції, що змінюють або видаляють дані.

Якщо необхідно зберігати структуру даних з чіткими відносинами між таблицями (наприклад, дані про замовлення, користувачів та продукти), то реляційні бази даних будуть оптимальним вибором. Якщо ж потрібна висока гнучкість і можливість обробляти великі об'єми даних, не зв'язаних між собою, краще використовувати нереляційні бази даних.

Також важливо враховувати масштабованість бази даних. Для масштабованих застосунків, які повинні обробляти велику кількість запитів або

працювати з великими наборами даних, варто обирати бази даних, які підтримують розподілене зберігання даних, такі як Cassandra або CouchDB, що дозволяють ефективно працювати з даними у великому масштабі.

1.3 Висновки до першого розділу

Метою даної роботи є полегшення процесу оцінювання викладачів студентами через автоматизацію збору та аналізу даних, забезпечуючи зручність у проведенні анонімних опитувань та перегляді детальної статистики про викладачів.

Для досягнення поставленої мети в кваліфікаційній роботі ставились та вирішувались наступні завдання дослідження:

1. Проведено аналіз технічних засобів, які застосовуються при розробці веб-застосунків, а також обрано необхідні інструменти для створення програмного забезпечення веб-застосунку для автоматизації проведення опитування здобувачів вищої освіти щодо якості викладання освітніх компонент.

2. Розроблено вимоги до програмного забезпечення, враховуючи як переваги, так і недоліки наявних систем, що дозволило сформулювати необхідні функціональні та нефункціональні вимоги для майбутнього застосунку.

3. Спроектовано та розроблено новий веб-застосунок, спираючись на детальний аналіз потреб цільових користувачів та врахування особливостей їх взаємодії з системою.

4. Проведено тестування розробленого застосунку, що дозволило перевірити його працездатність та виявити можливі помилки та недоліки.

Вхідними даними для веб-застосунку є текстові дані, які вводяться користувачами у відповідні поля форми, розташовані на сторінках інтерфейсу.

Ці дані, які можуть містити текстову інформацію, дати, числові значення або інші формати, передаються з клієнтської частини на сервер за допомогою протоколу HTTP, використовуючи методи POST або GET залежно від вимог до безпеки та конфіденційності. Серверна частина обробляє отримані дані,

виконуючи перевірку валідності, виконуючи необхідні обчислення або збереження їх у базу даних, після чого відправляє відповідь назад на клієнтську частину.

Для коректної та стабільної роботи веб-застосунку користувачеві необхідно мати сучасний веб-браузер, що підтримує новітні веб-стандарти, такі як HTML5, CSS3 та JavaScript ES6+. Окрім цього, мінімальною вимогою для пристрою є наявність щонайменше 6 ГБ оперативної пам’яті. Ця вимога обумовлена необхідністю забезпечення плавності роботи користувацького інтерфейсу та ефективного виконання ресурсомістких операцій, таких як рендеринг складних елементів дизайну чи робота з великими обсягами динамічних даних.

РОЗДІЛ 2. АНАЛІЗ ЗАСОБІВ РЕАЛІЗАЦІЇ ВЕБ-ЗАСТОСУНКУ

2.1 Вибір програмних засобів для реалізації проекту

Розробка сучасних веб-застосунків вимагає ретельного вибору інструментів, які забезпечують оптимальний баланс між продуктивністю, простотою використання та можливістю подальшого масштабування. Вибір технологій для цього веб-застосунку базується на характеристиках проекту, таких як тип застосунку, обсяг функціоналу, вимоги до продуктивності та естетичних аспектів дизайну.

Цей проект реалізований із застосуванням сучасного стеку технологій, який включає засоби для роботи з клієнтською та серверною частинами.

Основними критеріями вибору інструментів були:

1. Масштабованість. Обрані технології повинні легко адаптуватися до зростання проекту та впровадження нових функцій.

2. Продуктивність. Пріоритет надавався інструментам, які забезпечують високу швидкість виконання коду як на клієнтській, так і на серверній частинах.

3. Простота підтримки. Вибір інструментів був спрямований на полегшення обслуговування та розвитку проекту.

Основний стек технологій:

1. Клієнтська частина:

1) HTML.

2) CSS.

3) SCSS як інструмент для створення гнучких та модульних стилів.

4) JavaScript/TypeScript використовується для програмної логіки та забезпечення інтерактивності інтерфейсу. Були обрані через їх широку підтримку в індустрії, легкість інтеграції з іншими технологіями та гнучкість у використанні. TypeScript надає додаткові можливості, як статична типізація, що знижує ризик помилок.

5) Vue.js для компонентного підходу до розробки інтерфейсу. легкий у використанні, швидкий фреймворк, що ідеально підходить для побудови динамічних інтерфейсів.

6) Vuetify як бібліотека компонентів, що базується на Material Design.

7) PNPM для управління залежностями завдяки його ефективності та економії дискового простору.

2. Серверна частина:

1) Node.js забезпечує виконання JavaScript на сервері, що робить стек розробки єдиним для обох частин.

2) NestJS використовується для структурованої та модульної розробки серверної логіки.

3) SQLite як проста, але ефективна база даних для збереження даних.

На серверній частині Node.js і NestJS дозволяють створювати потужний і розширюваний серверний застосунок. SQLite як база даних добре підходить для локального зберігання даних через свою простоту налаштування та легку інтеграцію.

2.2 Вимоги до програмного забезпечення

Функціональні вимоги:

1. Авторизація до панелі керування за допомогою email та пароля.
2. Зміна пароля.
3. Адміністратор має можливість створювати, призупиняти та видаляти опитування.
4. Підтримка створення різних типів запитань (коментар або варіант відповіді).
5. Студент може переглядати опитування, проходити його, вказавши викладача конкретної дисципліни.
6. Виведення агрегованих результатів опитувань у вигляді графіків та таблиць у панелі керування.

7. Можливість завантажити список дисциплін та відповідних викладачів з Excel файлу для масового наповнення даними бази даних.

Нефункціональні вимоги:

1. Система повинна обробляти одночасно до 500 активних користувачів.

2. Час відповіді сервера не повинен перевищувати 10 секунд для основних операцій.

3. Архітектура має дозволяти легке масштабування при збільшенні кількості користувачів.

4. Усі паролі повинні зберігатися у хешованому (зашифрованому) вигляді.

5. Система повинна працювати в останніх версіях основних браузерів (Chrome, Firefox, Edge, Safari).

Вимоги до технологій:

1. Використання фреймворка Vue.js для побудови реактивних та інтерактивних інтерфейсів.

2. Використання Vuetify для швидкого створення адаптивного інтерфейсу, що відповідає принципам Material Design.

3. Використання NestJS для створення серверної логіки, REST API, та інтеграції з базою даних.

4. Забезпечення підтримки модульності та зручності тестування.

5. Використання реляційної бази даних (SQLite) для зберігання структурованих даних про опитування та користувачів.

6. Застосування PNPM для швидкого та ефективного управління залежностями проекту.

Вимоги до підтримки програмного забезпечення:

1. Документація для розробників, яка включає опис API, структури бази даних та ключових компонентів.

2. Інструкції для кінцевих користувачів щодо реєстрації, проходження опитувань та перегляду результатів.

2.3 Засоби для розробки клієнтської частини

2.3.1 HTML

HTML (HyperText Markup Language) є основою веб-розробки, оскільки забезпечує структуру всіх веб-сторінок. Це мова розмітки, яка використовується для визначення елементів інтерфейсу, їхньої структури та взаємодії. HTML не є мовою програмування, оскільки вона не виконує обчислень, а лише описує ієрархію і розташування контенту.

Основні особливості HTML:

1. Семантика. HTML дозволяє створювати зрозумілу структуру веб-сторінки завдяки семантичним тегам, таким як <header>, <article>, <nav>, <footer>, що підвищує доступність і SEO-оптимізацію.
2. Гнучкість. HTML працює з різними типами контенту, такими як текст, зображення, відео та мультимедіа.
3. Кросбраузерність. Код HTML може бути виконаний у будь-якому сучасному браузері без необхідності додаткових налаштувань.

Структура документа HTML:

Основний документ HTML складається з таких елементів:

1. Тег <!DOCTYPE html>. Вказує браузеру тип документа.
2. Тег <html>. Кореневий елемент документа.
3. Тег <head>. Містить метаінформацію про документ (кодировку, заголовок, підключення стилів і скриптів).
4. Тег <body>. Містить основний контент сторінки, який бачить користувач.
5. Тег <header>. Включає заголовки сторінки або її розділів, логотип і навігаційне меню.
6. Тег <main>. Використовується для позначення основного змісту сторінки, унікального для конкретної сторінки.

7. Тег <section>. Логічний розділ веб-сторінки, який може містити заголовок і пов’язаний контент.

8. Тег <article>. Самостійний елемент контенту, наприклад, стаття чи новина.

9. Тег <aside>. Додатковий контент, який не є основною частиною сторінки, наприклад, бокова панель або реклама.

10. Тег <footer>. Відображає нижній колонтитул сторінки, як-от контактну інформацію чи посилання.

11. Тег . Універсальний контейнер для групування інших елементів сторінки.

12. Тег <h1>. Головний заголовок сторінки, який використовується для позначення основної теми.

13. Тег <p>. Використовується для форматування абзаців тексту.

14. Тег <ul>. Маркований список елементів, з підпорядкованими тегами <li> для кожного пункту.

15. Тег <nav>. Визначає блок із посиланнями для навігації по сайту чи сторінці.

Нижче зображено структуру HTML тегів (див. рис. 2.1).

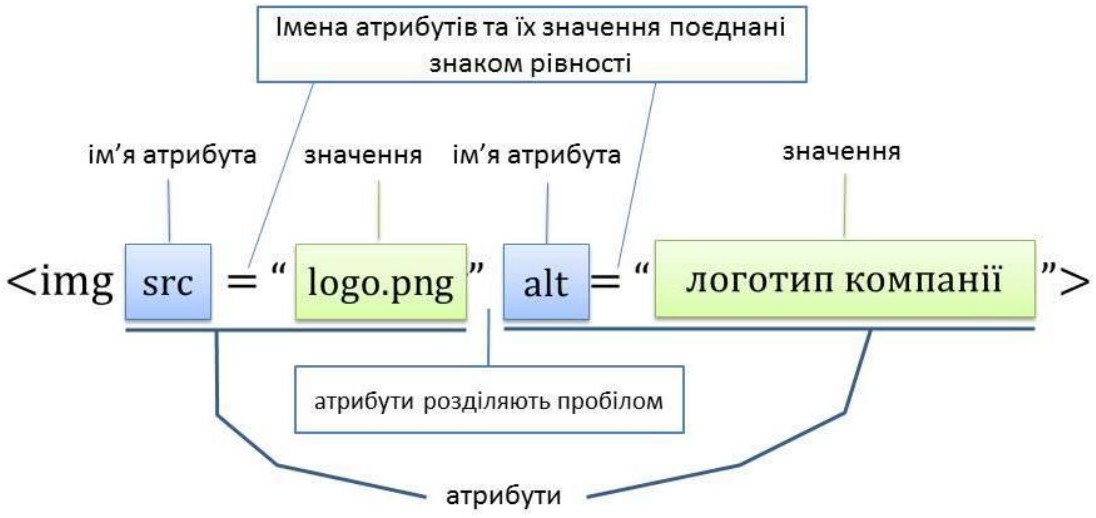

Рисунок 2.1 – Структура HTML тегів

HTML є фундаментом сучасного вебу. Без нього неможливо створити жодну веб-сторінку. Однак самостійно HTML не забезпечує стилізацію чи інтерактивність — для цього застосовуються CSS і JavaScript. У цьому проекті HTML виконує роль базового інструменту для опису структури клієнтського інтерфейсу.

Таким чином, HTML залишається ключовим елементом у створенні будь-якого веб-застосунку, забезпечуючи простоту у використанні, читабельність коду й сумісність з іншими технологіями.

2.3.2 CSS

CSS (Cascading Style Sheets) — це мова стилів, яка використовується для визначення візуального оформлення веб-сторінок, створених на HTML. Основне завдання CSS полягає у розділенні структури контенту і його стилів, що дозволяє значно спростити розробку, підтримку та модернізацію веб-проектів. Завдяки CSS можна змінювати шрифти, кольори, відступи, розташування елементів, додавати анімації та створювати адаптивний дизайн. CSS забезпечує каскадність, тобто використання кількох рівнів стилів (вбудовані, внутрішні та зовнішні стилі), які об'єднуються в єдине оформлення на основі пріоритетів.

На рисунку 2.2 зображено синтаксис CSS.

Рисунок 2.2 – Синтаксис CSS

Однією з головних переваг CSS є адаптивність, яка дозволяє веб-сторінкам коректно відображатися на різних пристроях завдяки медіа-запитам. Наприклад, можна задати окремі стилі для мобільних телефонів і великих екранів. CSS також підтримує роботу з макетами, використовуючи сучасні підходи, як-от Flexbox та Grid. Flexbox зручний для розташування елементів у рядках або стовпцях, тоді як Grid підходить для складнішого компонування у двовимірній сітці.

CSS також дозволяє створювати динамічні ефекти, такі як анімації та переходи. За допомогою властивості transition можна забезпечити плавний перехід між стилями, а @keyframes дозволяє визначати складні анімації. Стилі можна зберігати у зовнішніх файлах із розширенням .css, які підключаються до HTML-коду через тег <link>. Це дає змогу використовувати одні й ті ж стилі для багатьох сторінок і значно зменшує обсяг коду.

CSS є незамінним інструментом для створення сучасного веб-дизайну. Він дозволяє не тільки змінювати зовнішній вигляд сайту, але й робить його зручним, доступним і привабливим для користувачів.

2.3.3 SCSS

SCSS (Sassy CSS) — це розширена версія CSS, яка є частиною препроцесора SASS (Syntactically Awesome Stylesheets). Вона дозволяє розширити можливості звичайного CSS, зберігаючи при цьому сумісність із ним. SCSS дає змогу використовувати додаткові функції, які значно полегшують процес створення та підтримки стилів на великих і складних веб-проектах. Однією з основних переваг SCSS є підтримка вкладеності. Завдяки цьому можна створювати більш організовані і логічні структури стилів, зменшуючи повторення коду і спрощуючи підтримку проектів.

Ще однією важливою особливістю SCSS є змінні, які дозволяють зберігати значення, що використовуються багаторазово, наприклад, кольори, шрифти, розміри відступів та інші параметри. Це дозволяє централізовано управляти

стилями і легко змінювати значення без необхідності редагувати весь код. Крім того, SCSS підтримує міксини — спеціальні блоки стилів, які можна використовувати багаторазово в різних частинах проекту. Міксини дозволяють створювати більш гнучкий і ефективний код, зменшуючи дублювання та збільшуючи можливості кастомізації.

SCSS також має потужні функціональні можливості, наприклад, підтримку циклів і умовних операторів, що дозволяє створювати динамічні стилі, які адаптуються до різних умов. Завдяки цим можливостям можна значно зменшити кількість ручної роботи при створенні складних і змінних стилів для великих проектів. У SCSS можна використовувати імпорти для розділення стилів на кілька файлів, що дозволяє організувати код і зберігати його в чистоті, особливо у великих і складних веб-додатках.

Цей препроцесор також підтримує роботу з частковими файлами, що дозволяє розділяти код на логічні частини і знову використовувати їх в різних проектах. Важливою функцією є можливість підключати окремі файли, що дозволяє створювати стилі, які легко підтримуються і розвиваються в процесі роботи над проектом. SCSS надає більше можливостей для модульного і масштабованого підходу до створення стилів, що робить його корисним інструментом для веб-розробників, які працюють над складними і адаптивними веб-дизайнами.

SCSS значно полегшує створення складних, гнучких і масштабованих стилів для веб-сторінок. Завдяки таким можливостям, як змінні, міксини, вкладеність і динамічні функції, розробники можуть створювати ефективні, зручні для підтримки та адаптивні веб-дизайни, що спрощує роботу з великими проектами і забезпечує їх масштабованість та гнучкість.

2.3.4 JavaScript/TypeScript

JavaScript і TypeScript — це дві тісно пов’язані мови програмування, що використовуються для розробки веб-додатків. JavaScript є однією з основних мов

програмування для створення інтерактивних веб-сторінок. Завдяки своїй можливості працювати в браузері безпосередньо, JavaScript став стандартом для клієнтської логіки на веб-сайтах і веб-додатках. Його головною перевагою є гнучкість і велика кількість бібліотек, які дозволяють швидко створювати функціональні інтерфейси та маніпулювати DOM (Document Object Model).

JavaScript реалізує ООП через прототипне наслідування, що відрізняє його від мов на зразок PHP, які використовують класичне наслідування [6].

JavaScript є динамічно типізованою мовою, що означає, що типи змінних визначаються під час виконання, а не на етапі компіляції. Це дає велику свободу при розробці, але водночас може призводити до помилок, які складно відстежити. У зв'язку з цим, JavaScript використовує різні стратегії для забезпечення правильності коду, зокрема перевірку типів під час виконання, але велику частину таких помилок можна не помітити до тих пір, поки програма не запуститься.

TypeScript, у свою чергу, є надмножиною JavaScript, що додає статичну типізацію. Це означає, що в TypeScript програміст має можливість чітко вказати типи змінних, функцій та інших елементів коду на етапі написання програми. Це значно знижує ймовірність виникнення помилок, оскільки компілятор TypeScript може виявити багато типових помилок ще до того, як код буде запущений. Крім того, TypeScript дозволяє використовувати останні можливості JavaScript, а також додатково надає такі функції, як інтерфейси, абстрактні класи та інші засоби для покращення структури та організації коду.

TypeScript є особливо корисним у великих проектах, де потрібно працювати з численними модулями та складними взаємодіями між різними частинами системи. Використання TypeScript дозволяє знизити ризик виникнення помилок на етапі розробки і зробити код більш зрозумілим та підтримуваним. Ця мова стала дуже популярною серед розробників завдяки своїм перевагам, таким як покращена підтримка автодоповнення, виявлення помилок під час компіляції і краща інтеграція з інструментами для розробки.

JavaScript і TypeScript працюють разом, оскільки TypeScript компілюється у звичайний JavaScript, що дозволяє використовувати переваги статичної типізації TypeScript, при цьому отримуючи сумісність з усіма існуючими бібліотеками і фреймворками, які підтримують JavaScript. TypeScript також дозволяє поєднувати код на JavaScript з кодом на TypeScript в одному проекті, що робить перехід на використання цієї мови більш плавним і безболісним.

Типізація в TypeScript допомагає знизити складність підтримки великомасштабних проектів, дозволяючи створювати більш надійний і зручний код, в той час як JavaScript залишається невід'ємною частиною веб-розробки завдяки своїй універсальності та доступності. Обидві мови знаходять широке застосування, зокрема в розробці фронтенду, серверних додатків, а також для створення інтерактивних і динамічних веб-сторінок.

2.3.5 Vue.js

Vue.js — JavaScript-фреймворк, що використовує шаблон MVVM для створення інтерфейсів користувача на основі моделей даних, через реактивне зв'язування даних.

Vue використовує синтаксис шаблонів на основі HTML, що дозволяє декларативно зв'язувати рендеринг DOM з основними екземплярами даних в Vue. Всі Vue шаблони валідні HTML, і можуть бути розпарсені браузерами та HTML парсерами. Всередині Vue компілює шаблони в рендерингові функції віртуального DOM. В поєднанні з реактивною системою, Vue здатний розумно обчислити кількість компонентів для ре-рендингу та застосувати мінімальну кількість маніпуляцій з DOM, коли стан застосунку зміниться.

Однією з основних рис Vue є компонувальна архітектура. Кожен компонент у Vue являє собою ізольовану частину інтерфейсу, що містить власний шаблон, логіку та стилі. Такий підхід робить код чистим і зручним для підтримки, оскільки дозволяє розділяти складні інтерфейси на менші, керовані

одиниці. Це дозволяє розробникам зосередитися на створенні окремих частин інтерфейсу, що може бути повторно використано в різних частинах додатка.

Vue.js також підтримує декларативний синтаксис для роботи з даними. За допомогою механізму двостороннього зв'язування даних (two-way data binding) можна автоматично синхронізувати модель даних з відображенням, що спрощує маніпуляції з інтерфейсом. Цей підхід дозволяє значно зменшити кількість необхідного коду, оскільки оновлення даних автоматично відображається на екрані без потреби вручну оновлювати DOM.

Ще однією важливою особливістю Vue є його потужна система управління станом. З використанням Vuex, офіційного бібліотечного рішення для керування станом, можна централізовано зберігати дані додатка і керувати їх змінами в єдиному місці, що робить додаток більш організованим і масштабованим. Це особливо корисно для великих додатків, де потрібно обробляти багато взаємодій між компонентами.

Vue також підтримує інтеграцію з іншими технологіями та бібліотеками. Це дозволяє використовувати його в різних типах проектів, від простих односторінкових додатків до складних серверних рішень. Завдяки можливості плавного переходу від традиційних JavaScript додатків до Vue, можна поетапно інтегрувати фреймворк в існуючі проекти без необхідності повної переробки коду.

Перевагою Vue є також його продуктивність і швидкість. Він має невеликий розмір і оптимізований для високої продуктивності, що дозволяє йому працювати ефективно навіть на мобільних пристроях. Крім того, Vue має потужні інструменти для розробки та відлагодження, що робить процес розробки зручним і швидким.

Vue.js — це відмінний вибір для розробників, які хочуть створювати сучасні, швидкі і зручні веб-додатки. Його простота, гнучкість і потужні можливості роблять його одним з лідерів на ринку JavaScript-фреймворків.

2.3.6 Vuetify

Vuetify — це потужна бібліотека компонентів для Vue.js, яка забезпечує створення сучасних, привабливих і адаптивних інтерфейсів користувача. Вона базується на Material Design — наборі керівних принципів і стандартів для створення інтерфейсів, розроблених Google. Vuetify дозволяє легко інтегрувати дизайн матеріалу в додатки на Vue, надаючи величезну кількість готових до використання компонентів та утиліт для створення інтерфейсів. Цей фреймворк надає розробникам потужні інструменти для створення мобільних та десктопних додатків, не витрачаючи часу на розробку власних компонентів і стилів. Однією з ключових особливостей Vuetify є велика бібліотека компонентів. Вона включає різноманітні елементи інтерфейсу, такі як кнопки, діалогові вікна, форми, картки, таблиці, меню, панелі, слайдери та багато іншого. Всі ці компоненти стикаються з принципами Material Design, що дозволяє створювати додатки з приємним та інтуїтивно зрозумілим інтерфейсом. Завдяки великій кількості готових до використання компонентів, Vuetify значно прискорює процес розробки і знижує витрати часу на дизайн інтерфейсу.

Vuetify також підтримує налаштування тем і стилів, що дозволяє легко змінювати зовнішній вигляд додатка. Це можна робити як глобально, так і для окремих компонентів. Завдяки можливості налаштування кольорової палітри, шрифтів, розмірів і інших параметрів, розробники можуть адаптувати інтерфейс під потреби проекту, зберігаючи при цьому консистентність і відповідність принципам Material Design. Vuetify надає гнучкі інструменти для роботи з темами, що дозволяє підтримувати темний режим, налаштовувати колірну схему додатка та забезпечувати гармонію між елементами інтерфейсу.

Одним з важливих аспектів Vuetify є підтримка адаптивного дизайну. Завдяки вбудованій системі сіток і можливості працювати з медіа-запитами, компоненти автоматично підлаштовуються під різні розміри екранів. Це дозволяє створювати універсальні веб-додатки, які добре виглядають і функціонують на мобільних пристроях, планшетах та десктопах без необхідності

додаткової роботи для кожної платформи. Vuetify підтримує різні варіанти сітки, що дає можливість гнучко керувати розташуванням елементів на сторінці та забезпечувати адаптивність інтерфейсу.

Завдяки інтеграції з Vue.js, Vuetify дозволяє зручно працювати з компонентами та забезпечує простоту взаємодії між ними. Інтерфейс компонента Vuetify може бути налаштований за допомогою Vue.js директив і подій, що дозволяє створювати складні взаємодії з мінімумом коду. Компоненти Vuetify використовують стандартні можливості Vue для роботи з даними та станами, що дозволяє розробникам легко працювати з їхнім логічним функціоналом.

Ще однією перевагою Vuetify є підтримка міжнародних мовних стандартів, що робить його відмінним вибором для багатомовних додатків. Vuetify надає можливість локалізації компонентів, що дозволяє легко адаптувати додаток до різних мовних груп, автоматично змінюючи мову інтерфейсу залежно від вибору користувача.

2.3.7 PNPM

PNPM — це сучасний і швидкий менеджер пакетів для JavaScript, який забезпечує ефективне керування залежностями в проектах. Він створений для того, щоб вирішити проблеми, пов’язані з використанням традиційних менеджерів пакетів, таких як npm та Yarn, зокрема у питаннях швидкості, ефективності використання дискового простору та масштабованості. Завдяки своїм характеристикам PNPM здобув популярність серед розробників, шукаючих більш швидкий і зручний спосіб управління залежностями в своїх проектах.

Основною особливістю PNPM є його підхід до зберігання залежностей. Відмінність цього менеджера полягає в тому, що він використовує глобальне кешування для всіх проектів на локальному комп’ютері. Це означає, що при встановленні залежності, які вже були раніше завантажені для іншого проекту,

PNPM не завантажує їх знову, а просто посилається на них із глобального кешу. Завдяки такому підходу, значно зменшується дублювання файлів і використовуються менше дискового простору.

Іншою важливою особливістю PNPM є те, як він обробляє зв'язки між пакетами. Замість того, щоб встановлювати залежності в кожен проект у вигляді копій, PNPM використовує символьні посилання (symlinks), що дозволяє зберігати лише одну копію залежності, що використовуються в кількох проектах. Це призводить до значного зменшення обсягу файлів, що зберігаються, і спрощує процес оновлення пакетів, оскільки всі проекти в системі завжди використовують одну й ту ж версію пакету.

PNPM також вирізняється швидкістю роботи. Він має менший час на встановлення пакетів порівняно з npm або Yarn завдяки оптимізації процесу завантаження та зберігання залежностей. Кожен пакет завантажується і кешується один раз, після чого він швидко стає доступним для інших проектів, що зменшує час очікування при встановленні залежностей. Відмінною рисою PNPM є використання кешування, яке дозволяє не завантажувати пакети знову при повторних інсталяціях.

PNPM чудово підходить для роботи з монорепозиторіями — великими репозиторіями, що містять кілька проектів або модулів одночасно. Завдяки функції workspaces, PNPM дозволяє легко керувати залежностями між різними пакетами в межах одного репозиторію. Це зменшує необхідність повторного встановлення однакових залежностей для кожного модуля, забезпечуючи централізоване управління і зручну інтеграцію.

Однією з частих проблем у проектах з використанням традиційних менеджерів пакетів є так звані «примарні» залежності, коли один пакет ненавмисно отримує доступ до залежностей іншого пакету. Це часто стається через плоску структуру встановлення залежностей у npm та Yarn. PNPM ж використовує жорстку ієрархічну структуру залежностей, що дозволяє уникнути неявного використання залежностей, які не були явно зазначені у файлі

package.json. Такий підхід сприяє більш передбачуваній поведінці проекту і знижує ймовірність помилок під час роботи.

PNPM легко інтегрується з інструментами для автоматизації, такими як GitHub Actions, GitLab CI, Jenkins та іншими. Завдяки швидкості встановлення залежностей і підтримці кешування, PNPM значно скорочує час збірки проектів під час безперервної інтеграції. Інструмент надає можливість налаштовувати кешування для залежностей безпосередньо в CI/CD конвеєрі, що дозволяє ще більше оптимізувати процес розгортання.

PNPM повністю сумісний із файлами package.json і підтримує всі стандартні сценарії роботи з пакетами, такі як встановлення, видалення, оновлення і публікація пакетів. Розробники, які вже мають досвід роботи з npm або Yarn, можуть легко перейти на PNPM, оскільки команди мають подібний синтаксис. Крім того, PNPM підтримує встановлення глобальних пакетів, створення локальних бібліотек та роботу з приватними реєстрами.

2.4 Засоби для розробки серверної частини

2.4.1 Node.js

Node.js — це потужне середовище для виконання JavaScript на сервері, побудоване на движку V8, який використовується в Google Chrome. Це дозволяє розробникам використовувати JavaScript не тільки для написання клієнтського коду, але й для серверної логіки, що забезпечує більш ефективну і швидку розробку додатків. Node.js дозволяє створювати високопродуктивні серверні програми та мережеві додатки, зокрема веб-сервери, реальний час та багатозадачні додатки.

Основною перевагою Node.js є його асинхронність та подієво-орієнтоване виконання. Це означає, що код виконується без блокувань, і кожен запит обробляється незалежно від інших. Таке асинхронне виконання дозволяє

обробляти велику кількість одночасних з’єднань, що робить Node.js ідеальним для додатків, які вимагають масштабованості, таких як чат-системи, реальний час, ігри або веб-сервери з високим навантаженням.

Node.js використовує неблокуючий ввід/вивід (I/O), що дозволяє працювати з великою кількістю запитів без необхідності чекати завершення кожного з них. Це особливо корисно для веб-серверів, які обробляють багато одночасних запитів, оскільки вони не повинні бути заблоковані в очікуванні завершення операцій вводу-виводу, таких як доступ до баз даних або файлових систем.

Node.js має вбудовану підтримку роботи з потоками та обробкою подій, що дозволяє розробникам працювати з асинхронними операціями з мінімумом зусиль. Події обробляються за допомогою циклів подій, що дозволяє управляти тим, що відбувається в додатку, коли з’являються нові запити чи події.

Ще однією ключовою особливістю Node.js є його вбудовані модулі. Вони включають бібліотеки для роботи з файловою системою, створення HTTP-серверів, обробки потоків даних, мережевими запитами та іншими важливими аспектами розробки серверних додатків. Ці модулі значно спрощують розробку, оскільки вони вже інтегровані в середовище і не потребують додаткових налаштувань чи налаштування зовнішніх бібліотек.

Важливою складовою Node.js є його екосистема — NPM (Node Package Manager), один з найбільших репозиторіїв пакетів для JavaScript. NPM надає доступ до мільйонів пакетів, які можуть бути легко встановлені та використані у проектах. Це дозволяє розробникам швидко інтегрувати сторонні бібліотеки та інструменти для різних задач, від баз даних до фреймворків та інструментів для тестування.

Node.js також добре інтегрується з іншими технологіями та інструментами для розробки, зокрема з популярними фреймворками, такими як Express.js, який дозволяє швидко створювати веб-сервери та RESTful API, і Socket.io для роботи з веб-сокетами та реальним часом.

Node.js ідеально підходить для розробки API, реальних додатків, мікросервісів та будь-яких інших застосунків, де важлива асинхронність і обробка великої кількості запитів одночасно.

2.4.2 NestJS

NestJS — це фреймворк для створення серверних додатків на Node.js, який базується на TypeScript і використовує модульну архітектуру. Завдяки цьому, він дозволяє розробляти високопродуктивні, масштабовані та легко підтримувані програми. NestJS орієнтований на використання найкращих практик у розробці, зокрема об’єктно-орієнтованого програмування, функціонального підходу і реактивного програмування. Це дозволяє розробникам створювати структуровані додатки з чітким розподілом функцій і залежностей.

Однією з основних переваг NestJS є використання модульної архітектури. Програма ділиться на модулі, що відповідають за різні аспекти функціональності, такі як обробка запитів, робота з базами даних або автентифікація користувачів. Це дозволяє не тільки спрощувати розробку, а й полегшує тестування і масштабування додатків. Кожен модуль можна тестувати незалежно, і, за необхідності, додавати нові без порушення існуючої логіки.

NestJS активно використовує можливості TypeScript, зокрема декоратори, які значно зменшують кількість повторюваного коду і роблять програму більш виразною. Декоратори дозволяють описувати важливі аспекти коду, такі як маршрути, обробники запитів, залежності, а також параметри методів, що робить код більш читабельним та зрозумілим.

Іншою важливою особливістю є підтримка Dependency Injection (DI), що є потужним інструментом для впровадження залежностей у програму. DI дозволяє компоненти програми бути менш зв’язаними між собою, що полегшує їх тестування та зміну поведінки. Розробники можуть впроваджувати сервіси, бази даних, сховища даних та інші об’єкти без необхідності створювати їх вручну.

NestJS має вбудовану підтримку для популярних бібліотек і технологій, таких як Express або Fastify для створення HTTP-серверів, а також TypeORM, Sequelize і Mongoose для роботи з базами даних. Це дозволяє швидко інтегрувати необхідні інструменти в додаток, не витрачаючи часу на конфігурацію або написання зайвого коду.

Ще однією великою перевагою NestJS є підтримка різноманітних протоколів і технологій для роботи з мікросервісами. Фреймворк дозволяє створювати додатки, які можуть взаємодіяти через WebSocket, gRPC, GraphQL і REST API. Завдяки цьому, NestJS ідеально підходить для створення масштабованих, надійних і ефективних систем.

NestJS також добре інтегрується з інструментами для тестування, зокрема з бібліотекою Jest, що дозволяє легко писати юніт-тести, інтеграційні тести та тести для кінцевих точок API. Тестування є важливою складовою розробки в NestJS, що забезпечує високу якість і надійність додатків.

Завдяки простоті у використанні, масштабованості та потужним можливостям для роботи з різними технологіями, NestJS став популярним вибором серед розробників для створення складних серверних додатків. Це ідеальний фреймворк для розробки мікросервісів, RESTful API, сервісів реального часу і додатків з високими вимогами до продуктивності.

2.4.3 SQLite

SQLite — це вбудована реляційна система управління базами даних, яка зберігає дані у вигляді одного файлу на диску. Вона є легким і швидким інструментом для використання в додатках, де не потрібна складна серверна інфраструктура для обробки даних. SQLite розроблена з метою забезпечити швидкість і простоту використання при мінімальних витратах ресурсів, що робить її популярним вибором для мобільних додатків, настільних програм, а також для розробки прототипів і невеликих веб-додатків.

SQLite дозволяє використовувати SQL, який широко відомий і зрозумілий. Для графових баз необхідно використовувати спеціалізовані мови запитів, такі як Cypher, що може ускладнити розробку для невеликих проєктів [8].

Основною особливістю SQLite є те, що це "вбудована" база даних. Це означає, що вона не потребує окремого процесу або серверного програмного забезпечення для роботи. Всі операції з базою даних здійснюються безпосередньо в рамках програми, що використовує SQLite, за допомогою простих функцій або запитів SQL.

SQLite є високопродуктивною, оскільки всі операції з базою даних здійснюються в одному файлі, що зменшує накладні витрати на підключення до сервера і забезпечує швидкий доступ до даних. Вона підтримує транзакції, що дозволяє проводити кілька операцій з базою даних у межах однієї транзакції, що гарантує цілісність даних. Завдяки підтримці ACID (атомарність, консистентність, ізоляція та довговічність) транзакцій, SQLite забезпечує надійність навіть у випадку несподіваних відключень або помилок системи.

SQLite також забезпечує зручний механізм для роботи з SQL-запитами. Вона підтримує більшість SQL стандартів, включаючи селектори, вставку, оновлення, видалення даних і складні запити з об'єднанням таблиць та групуванням. Водночас вона не має підтримки деяких складних функцій великих серверних баз даних, таких як повна підтримка збережених процедур чи тригерів, однак для більшості застосувань цього достатньо.

Іншим важливим аспектом є те, що SQLite є крос-платформеним рішенням. Вона працює на всіх основних операційних системах, таких як Windows, Linux, macOS, Android і iOS. Це робить її чудовим вибором для мобільних додатків і програм, які повинні працювати на різних пристроях.

SQLite часто використовують для прототипування і в малих або середніх проектах, де відсутня потреба в розгортанні великої бази даних або сервера для зберігання даних. Вона також може бути корисною для невеликих веб-додатків, де кількість користувачів і обсяг даних обмежені, і де важливо забезпечити високу швидкість доступу.

Незважаючи на свою простоту, SQLite має потужну екосистему інструментів для управління базами даних, включаючи графічні інтерфейси, бібліотеки для роботи з програмними мовами, а також різноманітні плагіни для інтеграції в інші програми. Це дозволяє розробникам легко інтегрувати SQLite у свої додатки без необхідності впроваджувати складну серверну інфраструктуру.

2.5 Аналіз розробки дизайну веб-застосунку

Вибір правильного фреймворка для розробки дизайну веб-застосунку є важливим етапом у створенні привабливого та зручного інтерфейсу. У цьому аналізі розглядається використання Vuetify — популярного фреймворка для розробки інтерфейсів на основі Material Design, який дозволяє створювати стильні та функціональні веб-застосунки. Vuetify надає великий набір готових компонентів, що значно спрощує і прискорює процес розробки.

Однією з основних причин вибору Vuetify є його здатність автоматично генерувати адаптивний дизайн, що є важливим для забезпечення безперебійної роботи веб-застосунку на різних пристроях. Веб-застосунки, розроблені за допомогою Vuetify, відмінно виглядають на мобільних телефонах, планшетах і десктопах, завдяки використанню принципів Material Design, що гарантує зручний і зрозумілий інтерфейс для користувача.

Vuetify пропонує широкий спектр компонентів для побудови інтерфейсу, таких як кнопки, форми, таблиці, модальні вікна, спливаючі підказки, пагінація та багато іншого. Ці компоненти легко інтегруються та налаштовуються під потреби конкретного проєкту. Крім того, фреймворк дозволяє значно зменшити обсяг CSS-коду, оскільки багато стилів уже передбачені фреймворком.

Vuetify надає розробникам гнучкість у налаштуванні дизайну. Завдяки використанню тем і змінних, можна легко змінювати кольорові схеми, типографіку, розміри елементів і багато інших властивостей, що дозволяє адаптувати інтерфейс під специфічні вимоги проєкту. Це є важливим аспектом при розробці персоналізованих веб-застосунків, оскільки дозволяє зберігати

унікальність кожного проєкту, не витрачаючи багато часу на ручне стилізування кожного елементу.

Однією з ключових переваг Vuetify є його підтримка Vue.js — популярного JavaScript-фреймворка для створення інтерфейсів. Завдяки тісній інтеграції Vuetify з Vue.js, розробники можуть легко реалізовувати реактивність і динамічні зміни в інтерфейсі, що дозволяє створювати сучасні та зручні веб-застосунки.

Однак, як і у будь-якого фреймворка, у Vuetify є свої обмеження. Наприклад, для складних або дуже специфічних дизайнів може знадобитися додаткове налаштування компонентів або навіть повністю кастомізовані стилі. Тому важливо оцінити потреби проекту та зрозуміти, чи відповідає Vuetify всім вимогам щодо дизайну та функціональності.

Vuetify є одним з найкращих варіантів для швидкої розробки сучасних, адаптивних і стильних веб-застосунків. Він ідеально підходить для проектів, де потрібен стабільний, функціональний і добре виглядаючий інтерфейс, що відповідає сучасним стандартам дизайну.

2.6 Висновки до другого розділу

Під час проведеного аналізу технологій розробки веб-застосунку в другому розділі було обрано оптимальні інструменти для реалізації різних компонентів проекту.

Було розроблено вимоги до програмного забезпечення:

Для клієнтської частини веб-застосунку було обрано фреймворк Vue.js, який є популярним рішенням для розробки інтерфейсів. Його було обрано завдяки простоті у використанні, швидкості розробки та великій кількості підтримки з боку спільноти, що забезпечує постійні оновлення та вдосконалення фреймворка. Vue.js дозволяє швидко створювати ефективні і реактивні інтерфейси, що підвищує загальну продуктивність команди розробників.

Для створення користувацького інтерфейсу було вирішено використовувати Vuetify, який є компонентною бібліотекою, що підтримує принципи Material Design. Vuetify забезпечує великий вибір готових компонентів, що дозволяє швидко створювати адаптивні та стильні інтерфейси. Його використання дозволяє знизити витрати часу на створення базових елементів інтерфейсу і зосередитися на більш специфічних задачах проекту.

Для серверної частини було обрано фреймворк NestJS, побудований на основі Node.js. NestJS надає потужну архітектуру для створення масштабованих і підтримуваних серверних додатків. Вибір цього фреймворка був зумовлений його гнучкістю, модульністю та організацією коду, що дозволяє легко розширювати і підтримувати додаток. Завдяки великій спільноті розробників, NestJS постійно оновлюється і підтримується, що значно спрощує процес розробки.

Завдяки ретельно обраному набору технологій, процес розробки веб-застосунку став набагато швидшим, зручнішим та ефективнішим. Компонентно-орієнтований підхід, реалізований за допомогою Vue.js, дозволив створити модульну архітектуру, яка спрощує управління станом додатка, його тестування та повторне використання компонентів.

NestJS забезпечує структурованість серверної частини, пропонуючи інтуїтивний підхід до побудови REST API та зручну інтеграцію з базами даних.

Vuetify, заснований на принципах Material Design, надав можливість створення сучасного та адаптивного інтерфейсу без необхідності витрачати час на розробку з нуля.

Активна підтримка спільнот розробників, які працюють із такими технологіями, як Vue.js, NestJS та Vuetify, значно полегшує вирішення потенційних проблем, що можуть виникати в процесі роботи. Велика кількість документації, відеоуроків, статей та готових бібліотек дозволяє швидко знаходити відповіді на складні питання, інтегрувати нові рішення та оптимізувати існуючий код. Інструменти на кшталт PNPM сприяють швидкому

управлінню залежностями, що також позитивно впливає на загальну ефективність розробки.

Дизайн веб-застосунку був ретельно адаптований до вимог викладачів університету, які високо цінують інтуїтивність та простоту використання. Особливу увагу приділено зручності користувацького досвіду, завдяки чому застосунок забезпечує не лише функціональність, але й високий рівень задоволення користувачів під час роботи.

РОЗДІЛ 3. РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ПРОВЕДЕННЯ ОПИТУВАННЯ ЗДОБУВАЧІВ ВИЩОЇ ОСВІТИ ЩОДО ЯКОСТІ ВИКЛАДАННЯ ОСВІТНІХ КОМПОНЕНТ

3.1 Розробка загальної архітектури проекту

Архітектура веб-застосунку для збору статистики по викладачам побудована на базі сучасного стека технологій із чітким розділенням клієнтської та серверної частини. Це дозволяє забезпечити високу продуктивність, гнучкість, масштабованість та легкість у підтримці й розширенні проекту.

Проект складається з двох основних частин:

1. Серверна частина (Backend), побудована на основі NestJS.
2. Клієнтська частина (Frontend), розроблена з використанням Vue.js.

NestJS — це прогресивний фреймворк для Node.js, який дозволяє створювати масштабовані й структуровані серверні застосунки. Основою NestJS є концепція модульності, що дозволяє легко розділяти функціональність на ізольовані компоненти.

Архітектура NestJS складається з таких ключових елементів:

1. Модулі. Основний елемент архітектури, який групує логіку за функціональними напрямками (наприклад, модуль для роботи з викладачами чи статистикою).
2. Контролери (Controllers). Відповідають за обробку HTTP-запитів і передачу даних до відповідних сервісів.
3. Сервіси (Services). Містять бізнес-логіку застосунку, забезпечують взаємодію з базами даних і виконують основні операції.
4. Декоратори. Забезпечують зручне оголошення маршрутів, методів чи параметрів для контролерів.
5. ORM (TypeORM або Prisma). Використовується для роботи з базами даних (наприклад, з PostgreSQL).

На рисунку 3.1 зображено структуру модулів NestJS.

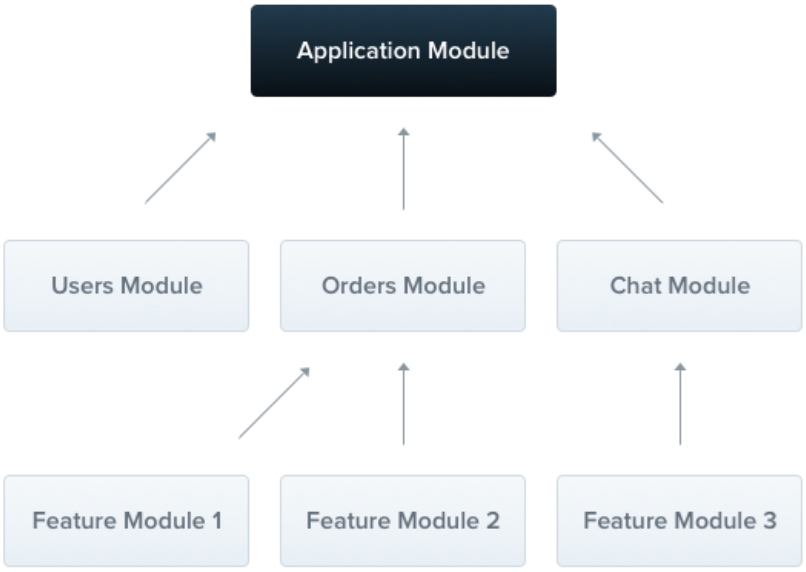

Рисунок 3.1 – Структура модулів NestJS

Центральним модулем є Application Module (або AppModule), який виступає основною точкою входу в застосунок. У ньому об’єднуються всі інші модулі, що утворюють загальну архітектуру проекту. Цей модуль є ядром застосунку та відповідає за об’єднання різних модулів. У ньому імпортуються всі спеціалізовані модулі, а також налаштовуються глобальні сервіси, середовище виконання та зовнішні залежності (наприклад, база даних, авторизація).

AppModule забезпечує інтеграцію між різними модулями й використовується для запуску застосунку. Він фактично є головним координаційним модулем, який не містить бізнес-логіки.

Клієнтська частина проекту побудована з використанням Vue.js — сучасного фреймворку для створення інтерактивних веб-інтерфейсів.

Vue.js забезпечує компонентно-орієнтовану архітектуру, де кожен елемент інтерфейсу (кнопка, форма, таблиця) реалізується у вигляді окремого компонента.

Основні елементи архітектури Vue.js:

1. Компоненти. Ізольовані частини інтерфейсу, які містять HTML, CSS і JavaScript [7]. Наприклад, компонент для відображення статистики викладачів або таблиці з даними.

2. Vuex. Централізоване сховище стану застосунку для управління даними, які використовуються у багатьох компонентах. У цьому проекті Vuex використовується для зберігання даних про викладачів та їхню статистику.

3. Маршрутизація (Vue Router). Забезпечує навігацію між сторінками, наприклад, переходи між головною сторінкою, сторінкою авторизації та сторінкою зі статистикою.

4. API-запити (Axios). Використовується для взаємодії з сервером через REST API.

Компонентна архітектура Vue.js спрощує розробку завдяки повторному використанню компонентів. Наприклад, таблиця даних може бути використана для відображення різних типів статистики, змінюючи лише вхідні параметри.

Ключові переваги використання Vue.js у проекті:

1. Легке освоєння та гнучкість.

2. Потужна екосистема, що включає інструменти для управління станом і маршрутизації.

3. Можливість створення швидких і адаптивних інтерфейсів.

Модель — це джерело даних у Vue.js. У цій частині визначається стан застосунку, з яким працює користувач. У Vue.js модель реалізована через реактивні властивості. Вона автоматично синхронізується з представленням (View) завдяки реактивності. Коли змінюється стан у моделі, Vue.js оновлює лише ті частини інтерфейсу, які залежать від цих змін.

View є інтерфейсом користувача, який побудований на основі DOM (Document Object Model). У Vue.js View описується за допомогою шаблонів (templates), які компілюються у DOM-елементи.

Vue.js спрощує створення інтерфейсу, дозволяючи описувати вигляд у декларативній формі. Шаблони Vue.js підтримують директиви (v-for, v-if тощо), що забезпечують динамічність інтерфейсу.

Vue.js використовує Virtual DOM — легковагову копію реального DOM. Замість того, щоб взаємодіяти безпосередньо з DOM, Vue.js виконує всі зміни у Virtual DOM. Після цього виконується так званий процес різниці станів (diffing), у якому визначаються зміни між поточним станом Virtual DOM і попереднім. Тільки змінені частини синхронізуються з реальним DOM, що значно підвищує продуктивність застосунків.

Vue.js дозволяє легко працювати з подіями у DOM через директиву v-on або коротку форму @. Наприклад, користувач може додати обробники подій, які автоматично синхронізуються з Vue-компонентами. Це забезпечує двосторонню прив'язку даних (data binding), що є основою інтерактивності застосунків.

Слухачі DOM використовуються для обробки подій, таких як кліки, введення тексту, прокрутка та інші дії користувача.

На рисунку 3.2 зображено архітектуру веб-застосунків з використанням Vue.js.

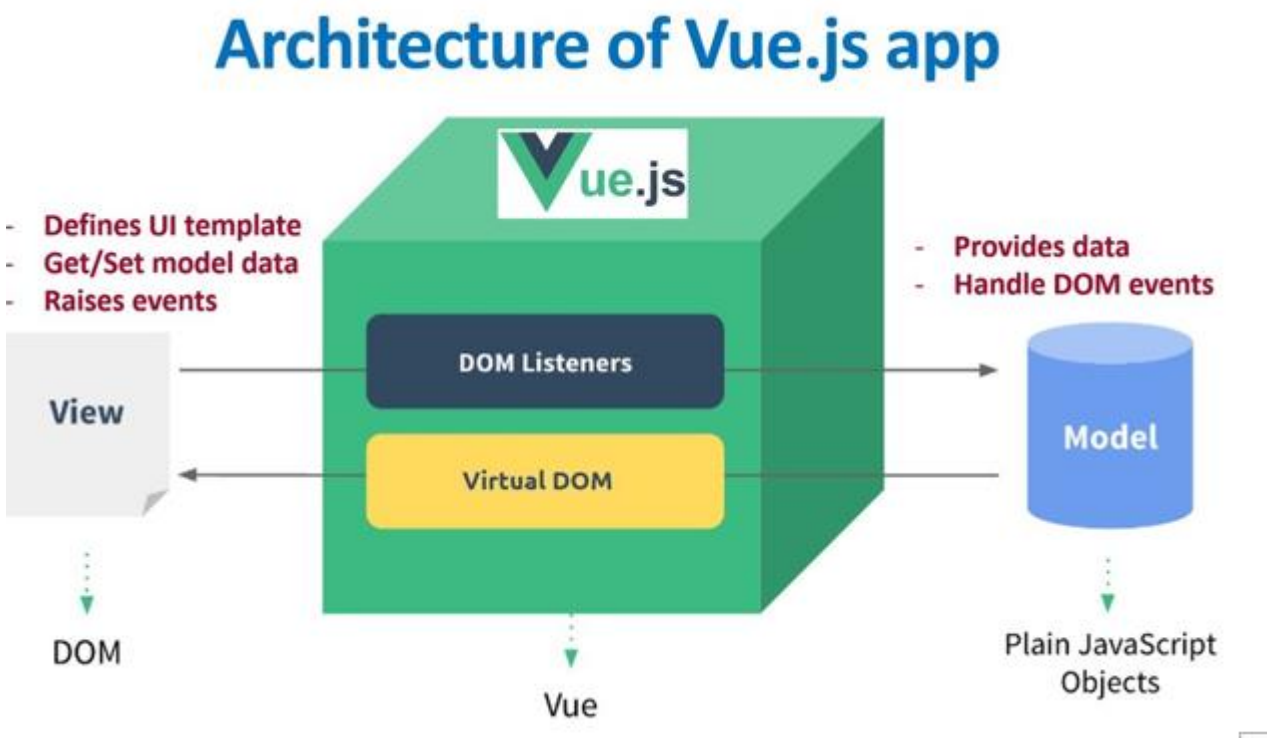

Рисунок 3.2 – Архітектура веб-застосунків з використанням Vue.js

3.2 Концептуальна модель інформаційної системи

Дана інформаційна система містить такі об’єкти та властивості:

1. Адміністратор. Адміністратор відповідає за управління системою, його обов’язки охоплюють контроль прав доступу, налаштування параметрів, а також моніторинг роботи системи. До властивостей адміністратора належать унікальний ідентифікатор (ID), повне ім’я, логін і пароль для автентифікації, а також рівень прав доступу, який визначає обсяг дозволених дій. Основні функції адміністратора включають створення, редагування та видалення облікових записів користувачів, зокрема студентів і викладачів, аналіз системних подій через логування, управління доступом до різних частин системи, а також налаштування параметрів для забезпечення її коректної роботи.

2. Студенти. Студенти використовують систему для оцінювання викладачів і дисциплін. До властивостей студентів належать ID користувача, ім’я, прізвище, номер академічної групи та перелік відвідуваних курсів. Основними функціями студентів є вибір викладачів і дисциплін для оцінювання, введення коментарів і виставлення оцінок через веб-інтерфейс. Система дозволяє оцінювати лише ті курси, які студент фактично відвідував, забезпечуючи коректність даних.

3. Оцінки. Оцінки є ключовим елементом для аналізу якості викладання. До їхніх властивостей належать ID оцінки, значення (у вигляді числового бала або текстового коментаря), зв’язок із конкретним студентом, викладачем і дисципліною. Оцінки використовуються для обчислення середніх і загальних балів, що дають змогу аналізувати ефективність роботи викладачів. Система автоматично агрегує дані оцінок, забезпечуючи їх актуальність і точність.

4. Курси. Курси представляють перелік навчальних дисциплін, що викладаються в навчальному закладі. До їхніх властивостей входять ID курсу, назва дисципліни, опис, кількість годин та викладачі, які ведуть ці курси. Курси пов’язані як зі студентами, які їх відвідують, так і з викладачами, що їх

викладають. Це дає змогу фільтрувати та структурувати дані відповідно до навчальних програм.

5. Статистика. Статистика відображає інформацію про ефективність викладання та використовується адміністраторами й викладачами. До властивостей статистики належать ID звіту, тип представлення (таблиця, графік, гістограма), а також параметри фільтрації, які дозволяють аналізувати дані за викладачем, курсом чи студентом. Основні функції статистики включають генерацію візуальних звітів, доступ до середніх і загальних балів, а також можливість експорту даних для подальшого аналізу.

Зв’язки між сутностями:

1. Один викладач може викладати декілька курсів.

2. Один студент може оцінювати кілька викладачів, але лише за ті курси, які він відвідував.

3. Курси та дисципліни пов’язані з викладачами.

Ключові процеси:

1. Збір даних. Студенти оцінюють викладачів через веб-інтерфейс.

2. Обробка даних. система розраховує середній та загальний бали для кожного викладача.

3. Візуалізація даних. Адміністраторам і викладачам надається доступ до статистики у вигляді таблиць, графіків.

4. Фільтрація та пошук. За допомогою фільтрів можна переглядати дані за певними параметрами (курс, освітня програма, дисципліна, викладач).

3.3 Математичні методи для аналізу інформаційних моделей

Веб-застосунок для збору статистики по викладачам потребував ефективних математичних методів для аналізу зібраних даних та побудови на їх основі статистичних звітів.

Основні методи, які було застосовані для обробки і аналізу даних у даній інформаційній системі:

1. Статистичний аналіз. Для оцінки ефективності викладання та розуміння загальних тенденцій серед студентів застосовуються базові статистичні методи. Серед найбільш важливих: обчислення середнього значення (середньої оцінки), медіани (оцінки, яка розподіляє дані на дві рівні частини) та стандартного відхилення (що дозволяє визначити, наскільки сильно оцінки розкидані відносно середнього значення). Це допомагає отримати загальну картину про успішність викладачів і студентів, а також виявити відхилення в оцінках, що можуть вказувати на необхідність змін у процесі викладання.

2. Кореляційний аналіз. Для аналізу взаємозв'язку між різними параметрами (наприклад, між оцінками студентів і курсами, які вони відвідують, або між оцінками та характеристиками викладачів) використовуються кореляційні методи. Кореляція дозволяє визначити, чи існує зв'язок між двома або більше змінними, та оцінити силу цього зв'язку. Наприклад, можна виявити, чи високі оцінки студентів корелюють з певними характеристиками викладача (стаж, кваліфікація) або специфічними курсами.

3. Моделі прогнозування. Для того, щоб передбачити майбутні результати оцінок або ефективність викладання на основі наявних даних, застосовуються методи прогнозування, такі як лінійна регресія або більш складні моделі машинного навчання. Лінійна регресія дозволяє побудувати модель, яка прогнозує оцінку на основі одного чи кількох факторів (наприклад, досвід викладача). Більш складні моделі машинного навчання можуть використовуватися для побудови точніших прогнозів, враховуючи більше змінних.

4. Аналіз часових рядів. Це дозволяє виявити тренди, сезонні коливання та зміну ефективності викладачів чи курсів із часом. За допомогою експоненціального згладжування можна передбачити, чи буде покращення оцінок студентів після зміни викладача або курсу.

5. Методи класифікації та кластеризації. Для аналізу великих наборів даних і виділення груп схожих викладачів або студентів застосовуються методи класифікації та кластеризації. Класифікація дозволяє автоматично групувати

викладачів або курси за певними характеристиками (наприклад, за рівнем ефективності чи за оцінками студентів). Кластеризація використовується для створення груп студентів чи викладачів, які мають схожі профілі (наприклад, за оцінками або за вибором курсів). Це допомагає виявити типові моделі поведінки та сприяти більш ефективному управлінню навчальним процесом.

6. Візуалізація даних. Після обробки та аналізу великих обсягів даних важливо представити їх у зручному для користувача форматі. Для цього використовуються різні методи візуалізації, такі як гістограми, графіки, теплові карти та діаграми. Це дозволяє адміністраторам та викладачам швидко розуміти важливі тенденції, порівнювати ефективність викладачів або дисциплін, а також легко виявляти аномалії або тенденції в даних. Візуалізація дозволяє представити складні статистичні результати у вигляді інтуїтивно зрозумілих графічних елементів, що полегшує процес прийняття рішень.

3.4 Математична модель та методи аналізу даних

Для розробки математичних моделей та методів аналізу даних, що стосуються оцінки ефективності викладання та статистики по викладачах, можна використовувати різні типи моделей і методів. Зокрема, для оцінки середнього балу викладача, прогнозування успішності студентів, а також аналізу взаємозв'язків між різними параметрами, можна застосувати відповідні математичні формули та методи: середній бал викладача по предмету, медіана оцінок викладача по предмету, стандартне відхилення оцінок, кореляція між оцінками студентів та характеристиками викладача, прогнозування оцінок студентів за допомогою лінійної регресії.

Для оцінки середнього балу викладача по предмету було використано формулу середнього арифметичного балу всіх студентів, які брали участь у заняттях з цього предмету.

Нехай: S_i — оцінка студента i і з предмету, а n — кількість студентів, які брали участь у заняттях. Тоді середній бал викладача по предмету обчислюється за формулою 3.1 [10, с. 86]:

$$S^- = \frac{1}{n}\sum_{i=1}^{n} S_i, \tag{3.1}$$

де: S^- — середній бал викладача по предмету.

Медіана є статистичним показником, який розділяє дані на дві рівні частини, тобто 50% оцінок вище, а 50% — нижче.

Нехай: S_1, S_2, …, S_n — впорядковані оцінки студентів (від найменшої до найбільшої).

Тоді медіана для непарної кількості оцінок n визначається за формулою 3.2 [10, с. 139]:

$$M = S\left(\frac{2n+1}{2}\right), \tag{3.2}$$

Для парної кількості оцінок медіана обчислюється як середнє значення двох середніх оцінок за формулою 3.3 [10, с. 139]:

$$M = \frac{S\left(\frac{n}{2}\right) + S\left(\frac{n}{2}+1\right)}{2}, \tag{3.3}$$

Стандартне відхилення дозволяє оцінити, як сильно оцінки студентів варіюються від середнього значення.

Нехай: S^- — середнє значення оцінок викладача;

S_i — оцінка кожного студента.

Тоді стандартне відхилення визначається за формулою 3.4 [10, с. 141]:

$$\sigma = \sqrt{\sum_{i=1}^{n} (S_i - \bar{S})^2}, \tag{3.4}$$

де: σ — стандартне відхилення оцінок.

Для аналізу взаємозв'язку між оцінками студентів та характеристиками викладача (наприклад, досвідом або кваліфікацією) використано коефіцієнт кореляції Пірсона. Нехай X_i — значення змінної, що характеризує викладача (наприклад, досвід), а Y_i — оцінка студента. Тоді коефіцієнт кореляції r обчислюється за формулою 3.5 [10, с. 200]:

$$r = \frac{\left(\sum_{i=1}^{n} X_i^2 - (\sum_{i=1}^{n} X_i)^2\right)\left(\sum_{i=1}^{n} Y_i^2 - (\sum_{i=1}^{n} Y_i)^2\right)}{\sum_{i=1}^{n} X_i Y_i - \left(\sum_{i=1}^{n} X_i\right)\left(\sum_{i=1}^{n} Y_i\right)}, \tag{3.5}$$

де: r — коефіцієнт кореляції Пірсона, який показує силу та напрямок зв'язку між двома змінними.

Для прогнозування майбутніх оцінок студентів на основі наявних даних застосовано лінійну регресію. Нехай: Y_i — оцінка студента, X_i — характеристика (досвід) викладача або параметр, що впливає на оцінки.

Лінійна регресія визначає залежність Y від X за формулою 3.6 [10, с. 223]:

$$Y = a + bX, \tag{3.6}$$

де: a — вільний член (константа), b — коефіцієнт регресії, що визначає нахил прямої.

Для групування схожих викладачів за певними ознаками застосовано метод k-середніх. Цей метод дозволяє поділити дані на k кластерів, мінімізуючи

відстань між точками в кожному кластері. Алгоритм кластеризації полягає в тому, що спочатку випадково вибираються k центрів кластерів, потім кожен елемент групується за найближчим центром, і центри кластерів оновлюються на основі середніх значень точок у кластері. Цей процес повторюється до досягнення стабільності.

3.5 Алгоритмічні рішення

На рисунку 3.3 зображено алгоритм збору та обробки статичних даних опитувань студентів про викладачів.

Рисунок 3.3 – Алгоритм збору та обробки статистичних даних опитувань

Алгоритмічні рішення для збору та обробки статистичних даних є важливими для оцінки ефективності викладачів. Перший етап алгоритму полягає

у зборі інформації від студентів через анкети, де вони оцінюють викладачів за різними параметрами, такими як якість викладання, доступність, комунікаційні навички тощо. Дані з анкет передаються на сервер і зберігаються в базі даних для подальшого оброблення. Студент заповнює форму оцінки викладача на веб-застосунку, і ця інформація передається на сервер, де зберігається у відповідних таблицях бази даних, прив'язуючись до конкретного викладача і курсу.

Після збору даних наступним етапом є обробка інформації, де необхідно обчислити середній бал для кожного викладача на основі отриманих відгуків. Алгоритм обчислення середнього балу полягає в підсумовуванні оцінок і поділі їх на кількість учасників оцінки. Також проводиться аналіз відхилень для оцінки стабільності та однорідності оцінок. В результаті обробки статистичних даних можна отримати відомості про середній бал викладача по кожному предмету та загальний рейтинг викладача на основі оцінок студентів.

3.6 Розробка дизайну проекту

У цьому проекті для реалізації дизайну клієнтської частини веб-застосунку було обрано бібліотеку Vuetify для компонентів користувацького інтерфейсу та препроцесор SCSS для кастомізації стилів і підвищення ефективності написання CSS-коду. Це рішення прийняте на основі їхніх переваг, гнучкості та відповідності вимогам сучасної розробки.

Vuetify — це бібліотека компонентів, яка базується на принципах Material Design, розроблених Google. Вона надає готові до використання компоненти та інструменти для створення сучасних і адаптивних інтерфейсів.

Завдяки Vuetify дизайн стає простішим, а розробники можуть зосередитися на бізнес-логіці, не витрачаючи багато часу на створення базових UI-елементів.

Бібліотеку Vuetify було з наступних причин:

1. Повний набір готових компонентів. Vuetify надає велику кількість готових UI-компонентів, таких як кнопки, таблиці, форми, сітки, каруселі,

діалоги та багато інших. Це значно прискорює процес розробки дизайну, оскільки компоненти адаптовані до різних розмірів екранів і вже оптимізовані для використання в різних браузерах.

2. Дотримання принципів Material Design. Завдяки Material Design інтерфейс виглядає сучасно й інтуїтивно зрозуміло. Це знижує поріг входу для користувачів і забезпечує єдиний стиль інтерфейсу.

3. Адаптивність та кросплатформенність. Vuetify автоматично підлаштовує компоненти під розмір екрана, що робить його ідеальним для створення адаптивних інтерфейсів, які правильно працюють як на мобільних пристроях, так і на десктопах.

4. Інтеграція з Vue.js. Оскільки Vuetify створений спеціально для Vue.js, його використання не потребує додаткових бібліотек чи складних налаштувань. Він органічно вписується в екосистему Vue, що полегшує його впровадження в проект.

5. Можливість кастомізації. Vuetify дотримується стандартів Material Design, він дозволяє змінювати стилі, кольори й поведінку компонентів за допомогою SCSS або JavaScript-тем.

Препроцесор SCSS було обрано з наступних причин:

1. Підтримка змінних. Змінні дозволяють зберігати значення кольорів, розмірів шрифтів або інших параметрів у одному місці. Це спрощує налаштування й забезпечує узгодженість у дизайні. Наприклад, зміна основного кольору проєкту здійснюється шляхом оновлення однієї змінної, а не пошуку й заміни кольору по всьому файлу стилів.

2. Вкладеність селекторів. SCSS підтримує вкладеність, що відповідає структурі HTML. Це робить стилі більш читабельними та організованими, оскільки стиль кожного елемента написаний у контексті його батьківського контейнера.

3. Міксини та функції. Міксини дозволяють створювати блоки повторюваних стилів, які можна багаторазово використовувати. Наприклад, міксин для адаптивності дозволяє легко налаштовувати стилі для різних розмірів

екранів. Функції додають можливість математичних обчислень у стилях, таких як розрахунок відносних розмірів.

4. Управління файлами. SCSS підтримує розбиття стилів на декілька файлів, що дозволяє легко організувати структуру проекту. Наприклад, можна створити окремі файли для кольорової палітри, стилів компонентів і змінних, що підвищує масштабованість.

5. Ідеальна інтеграція з Vuetify. Використання SCSS із Vuetify дозволяє легко кастомізувати стандартні компоненти, наприклад, змінювати кольори теми, розміри шрифтів або стилі кнопок, використовуючи змінні та міксини.

Адміністративна панель створена з акцентом на функціональність та ефективність роботи. Вона містить такі компоненти, як:

1. Кнопки для створення нових записів (дисципліни, освітні програми тощо).

2. Форми для введення даних.

3. Таблиці зі списками сутностей.

4. Інструменти для видалення записів.

Дизайн сторінки для початку проходження опитування зображено на рисунках 3.3–3.5.

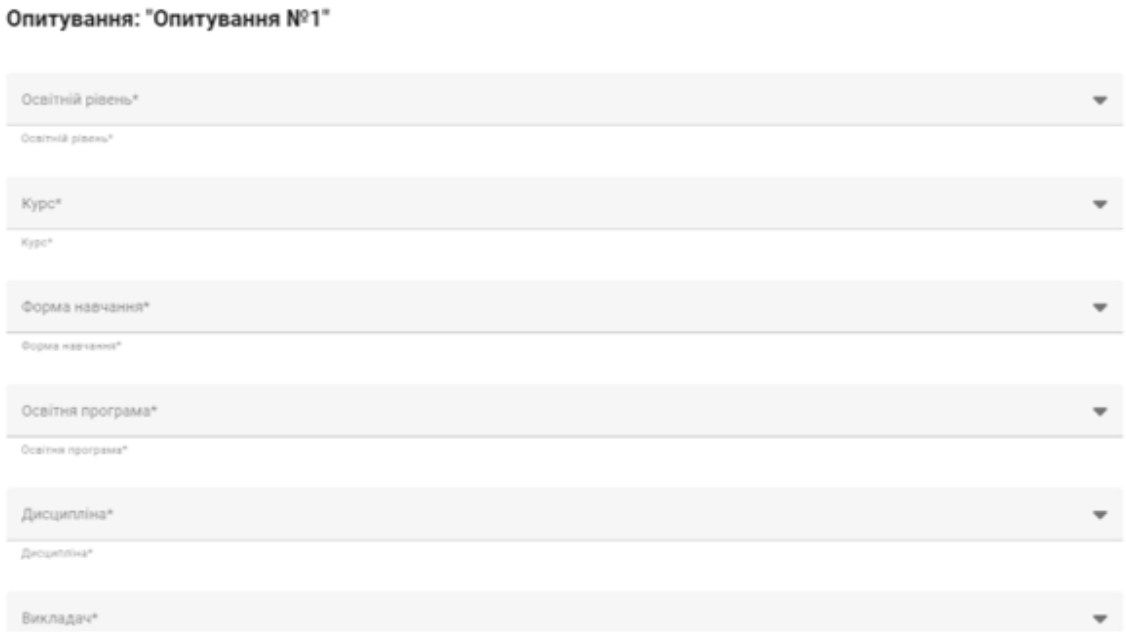

Рисунок 3.4 – Дизайн сторінки для початку проходження опитування

Рисунок 3.4 містить дизайн сторінки початку проходження опитування з наступними полями для заповнення:

1. Освітній рівень.
2. Курс.
3. Форма навчання.
4. Освітня програма.
5. Дисципліна.
6. Викладач.

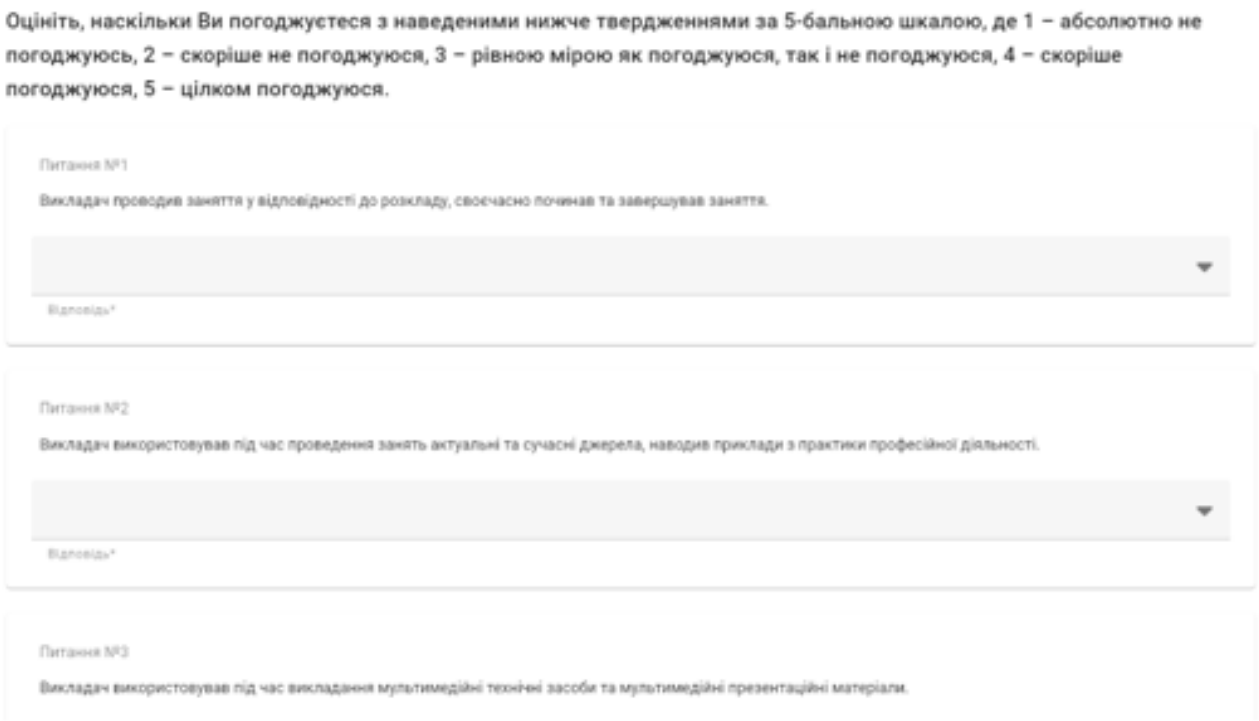

Рисунок 3.5 – Дизайн сторінки з опитуванням

Рисунок 3.5 містить дизайн сторінки з опитуванням. Поточне відкрите опитування складається з 18 питань різних типів, серед яких як закриті запитання з варіантами відповідей, так і відкриті запитання для введення коментарів.

Рисунок 3.6 – Дизайн сторінки після успішного проходження опитування

На рисунку 3.6 зображено дизайн сторінки після успішного проходження опитування. Вона містить текст: «Дякую за проходження опитування» та повідомлення з зеленим фоном: «Ви успішно пройшли опитування».

Дизайн сторінки авторизації до панелі керування зображено на рисунку 3.7. Вона містить ім’я користувача, пароль та кнопку «Увійти».

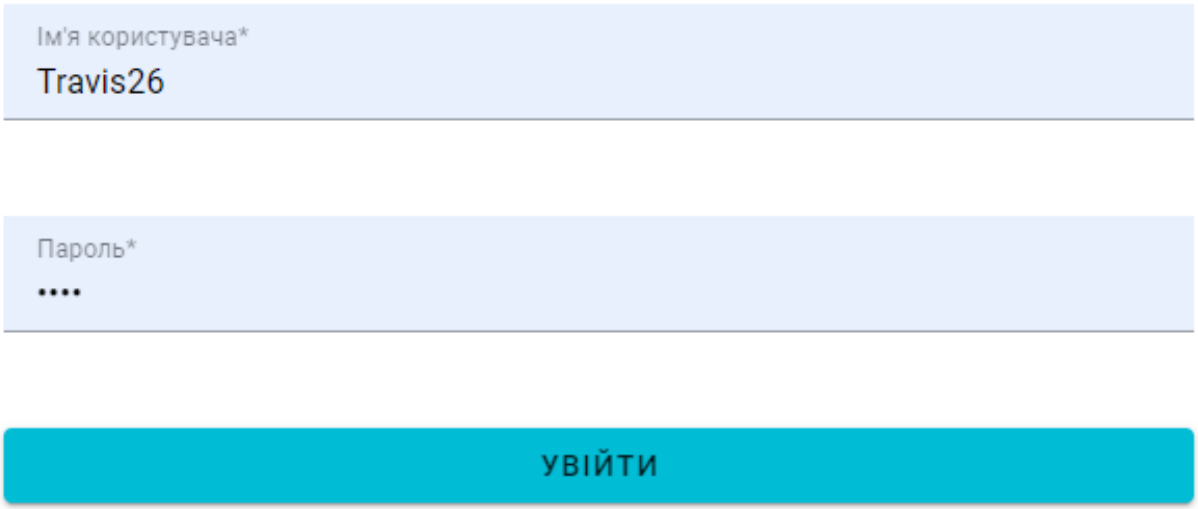

Рисунок 3.7 – Дизайн сторінки авторизації до панелі керування

Дизайн сторінки «Статистика» зображено на рисунку 3.8. Вона містить текстове поле з пошуком викладача за ПІБ, а також табличку з наступними стовпцями: ID, ПІБ, дисципліна, кількість пройдених опитувань та середній бал.

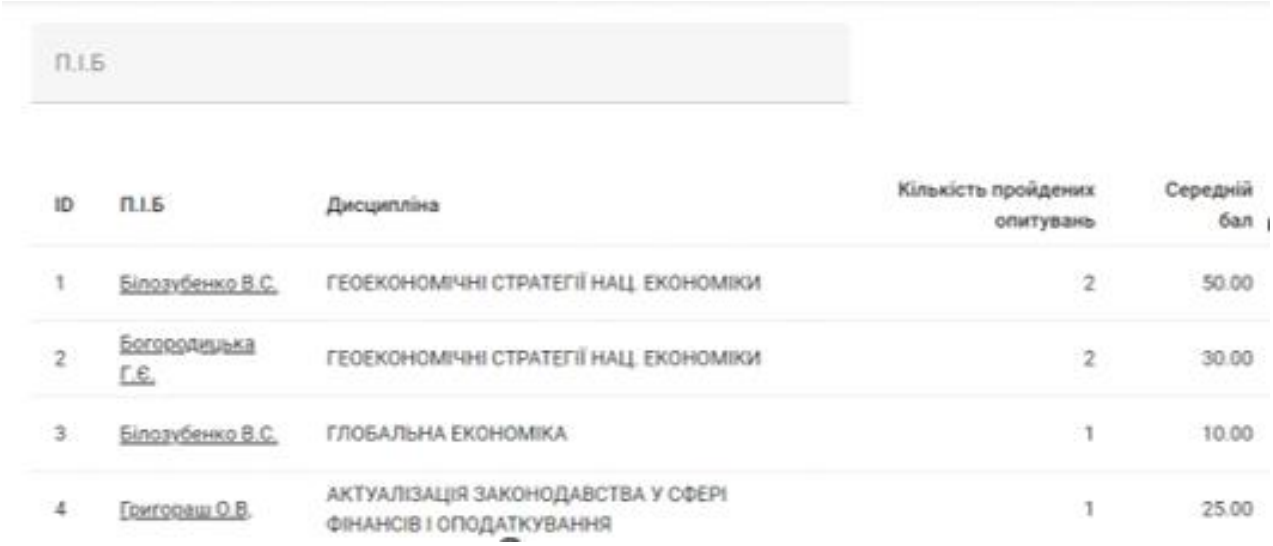

Рисунок 3.8 – Дизайн сторінки «Статистика»

На рисунку 3.9 зображено сторінку «Дисципліни». Вона складається з текстового поля «Назва», поля вибору файлу Excel для масового заповнення даними з таблички, таблиці з полями: ID, назва та дії.

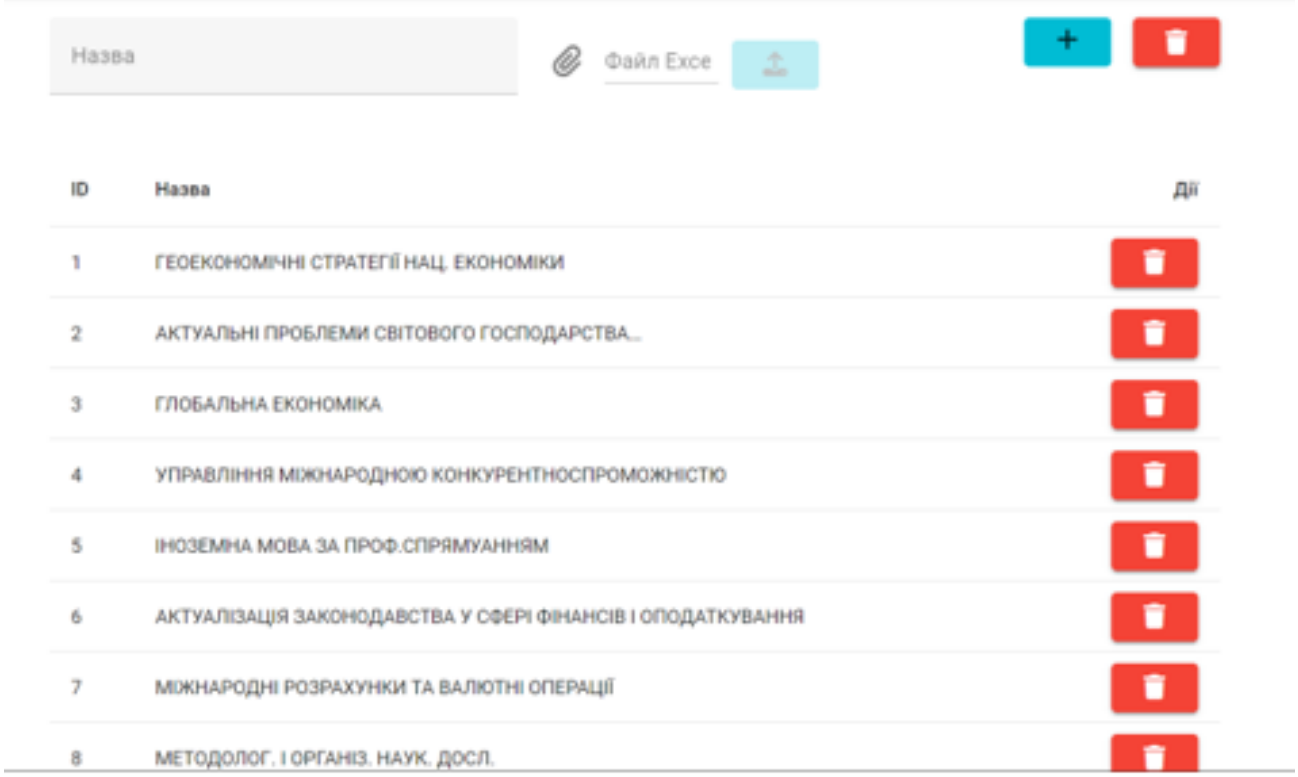

Рисунок 3.9 – Дизайн сторінки «Дисципліни»

На рисунку 3.10 зображено сторінку «Викладачі». Вона складається з текстового поля «Назва», поля вибору файлу Excel для масового заповнення даними з таблички, таблиці з полями: ID, назва та дії.

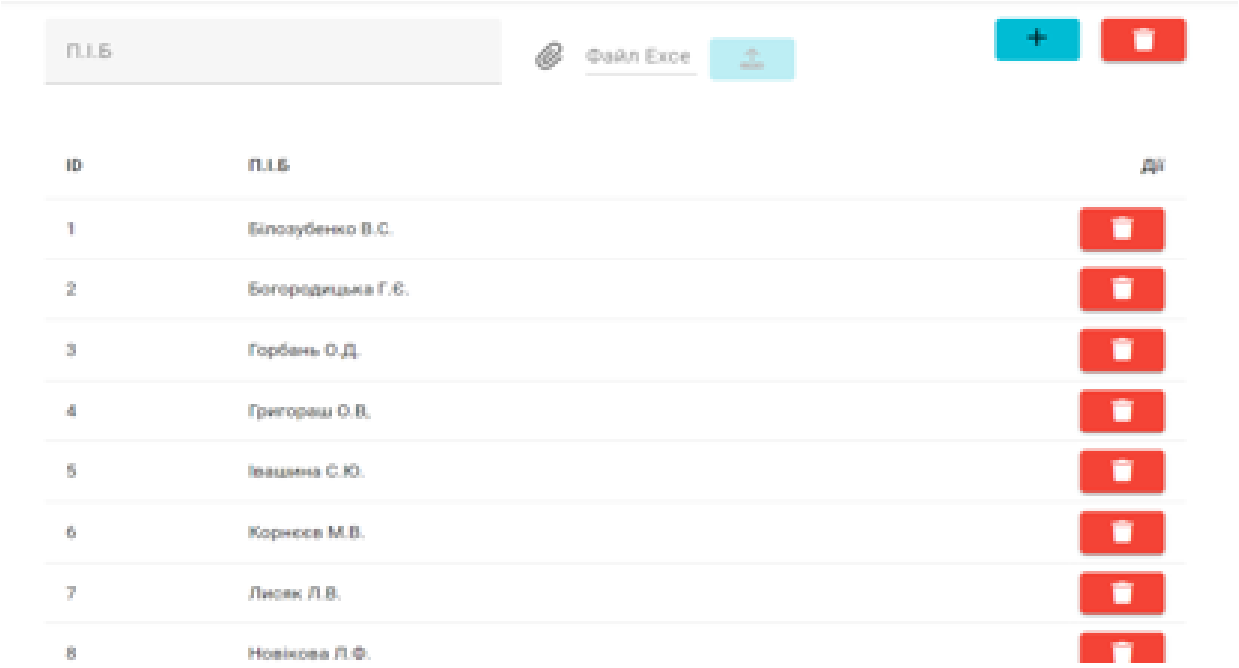

Рисунок 3.10 – Дизайн сторінки «Викладачі»

На рисунку 3.11 зображено сторінку «Освітні рівні». Вона складається з текстового поля «Назва», таблиці з полями: ID, назва та дії.

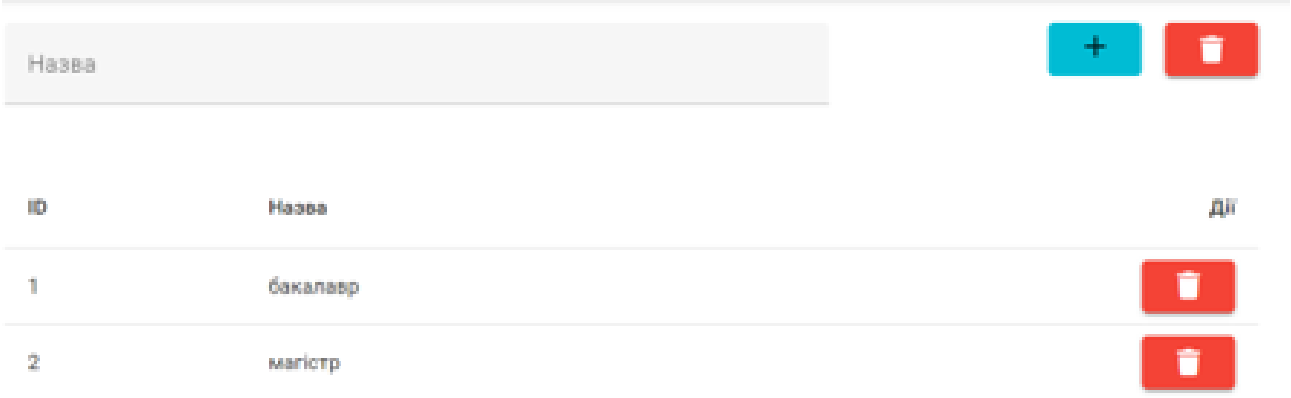

Рисунок 3.11 – Дизайн сторінки «Освітні рівні»

На рисунку 3.12 зображено сторінку «Курси». Вона складається з текстового поля «Назва», таблиці з полями: ID, назва та дії.

Рисунок 3.12 – Дизайн сторінки «Курси»

На рисунку 3.13 зображено сторінку «Форми навчання». Вона складається з текстового поля «Назва», таблиці з полями: ID, назва та дії.

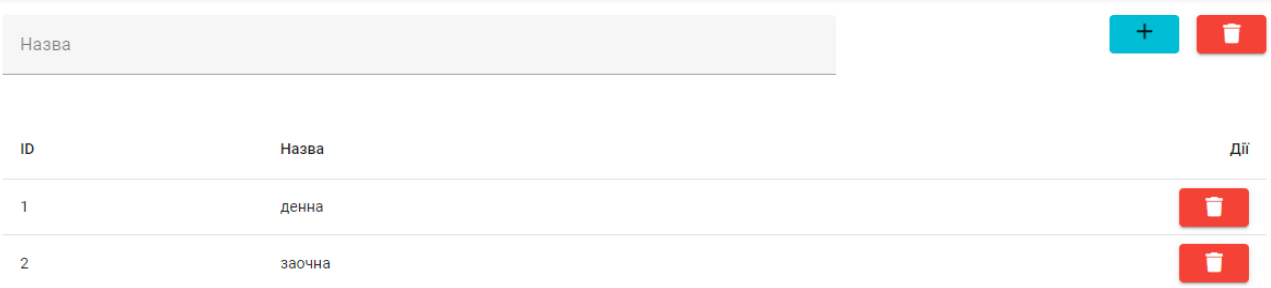

Рисунок 3.13 – Дизайн сторінки «Форми навчання»

На рисунку 3.14 зображено сторінку «Освітні програми». Вона складається з текстового поля «Назва», таблиці з полями: ID, назва та дії.

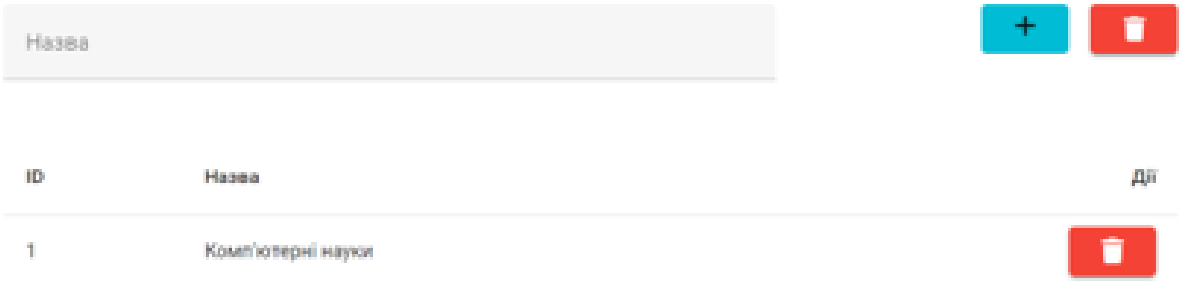

Рисунок 3.14 – Дизайн сторінки «Освітні програми»

На рисунку 3.15 зображено сторінку «Опитування» панелі керування. Вона складається з текстового поля «Назва», таблиці з полями: ID, назва та дії.

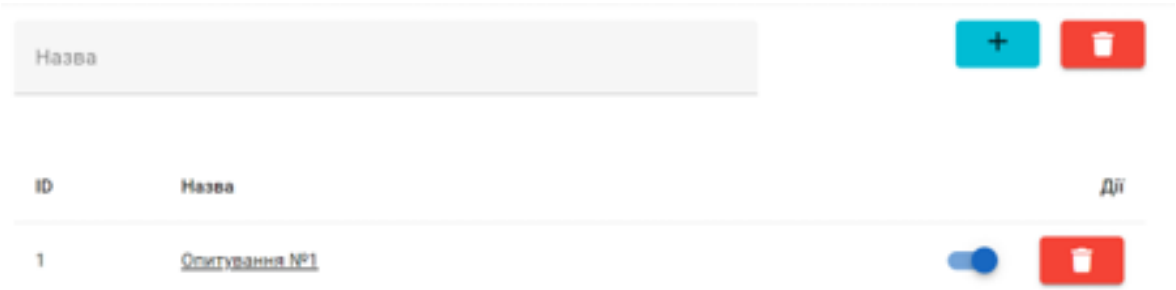

Рисунок 3.15 – Дизайн сторінки «Опитування» панелі керування

На рисунку 3.15 зображено сторінку «Змінити пароль», яка створена для забезпечення безпечного оновлення облікових даних користувача. Ця сторінка містить текстове поле для введення нового пароля та кнопку «Змінити пароль», яка активується лише після введення коректного значення текстового поля. Для підвищення рівня безпеки пароля реалізовано функцію перевірки його складності. Користувачу пропонується ввести пароль, що відповідає встановленим вимогам, таким як мінімальна довжина, використання великих та малих літер, цифр та спеціальних символів. У разі невідповідності пароля цим критеріям система виводить відповідне повідомлення з підказками для виправлення.

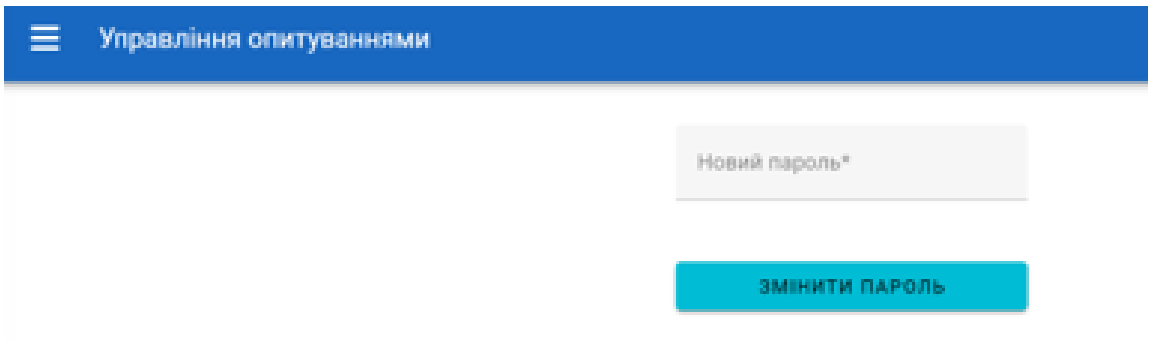

Рисунок 3.16 – Дизайн сторінки «Змінити пароль»

На рисунку 3.17 зображено сторінку «Викладач». Ця сторінка містить статистику по викладачу: загальний середній бал, гістограму (середній бал викладача за дисципліною), блок з коментарями.

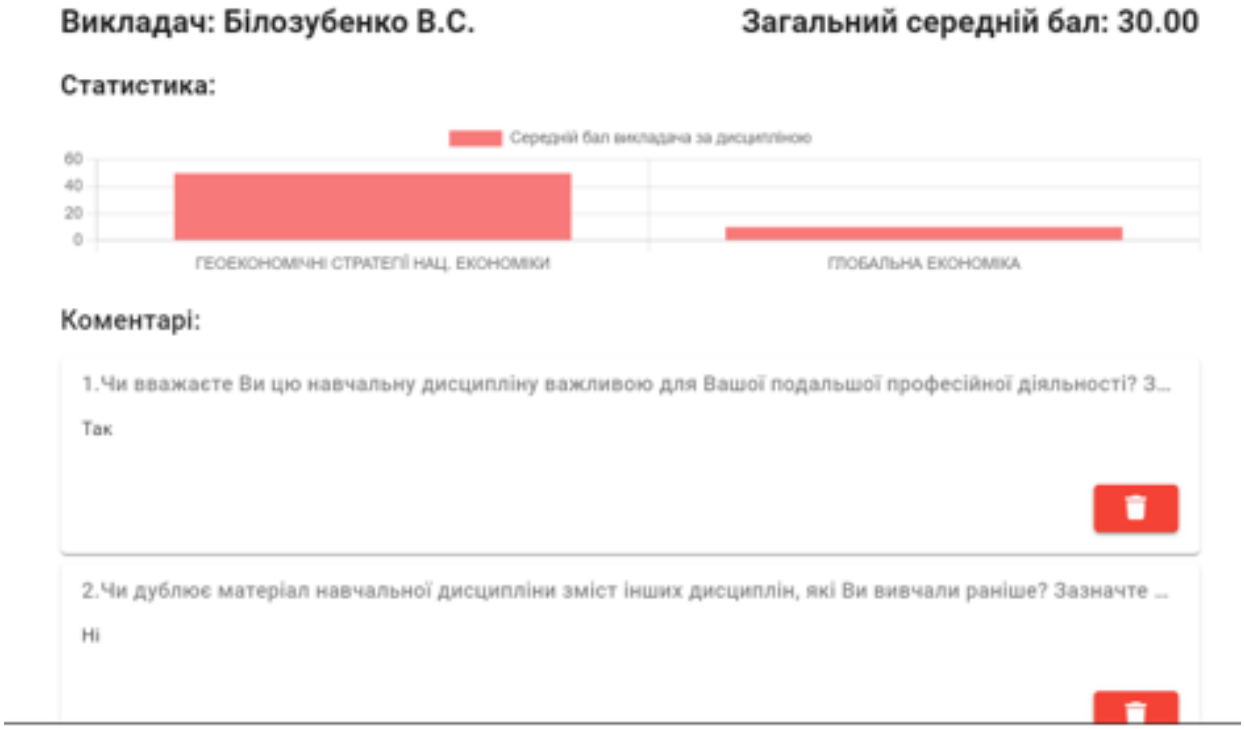

Рисунок 3.17 – Дизайн сторінки «Викладач»

3.7 Клієнтська (front-end) частина проекту

Клієнтська частина веб-застосунку створена для забезпечення взаємодії користувачів із системою. Вона включає сторінки, які розроблено відповідно до потреб студентів та адміністраторів. Веб-застосунок побудований з використанням Vue.js, що забезпечує динамічний і чуйний інтерфейс.

Перелік ключових сторінок, які формують клієнтську частину проекту:

1. Сторінка для заповнення даних перед початком опитування. Ця сторінка є стартовою для студентів перед проходженням опитування. Вона дозволяє користувачам внести необхідні дані, такі як обрана дисципліна, викладач, або освітній рівень. Інтерфейс реалізовано у формі, яка інтерактивно перевіряє правильність заповнення даних.

2. Сторінка опитування. Сторінка опитування створена з фокусом на зручність студентів. Опитування побудовано у вигляді питань із можливістю вибору відповідей (наприклад, радіокнопки чи чекбокси). Усі відповіді обробляються безпосередньо у браузері, а результати надсилаються до сервера.

3. Сторінка авторизації. Сторінка авторизації використовується для перевірки доступу до системи. Вона дозволяє адміністраторам входити в систему, вводячи логін та пароль. Використовується інтуїтивно зрозуміла форма з відповідними підказками у разі помилок.

4. Сторінка статистики. Адміністраторам доступна сторінка для перегляду результатів опитувань. Графічне представлення статистики реалізовано за допомогою таблиць. Це дозволяє швидко аналізувати результати опитувань і приймати рішення на основі даних.

5. Сторінка дисциплін. Ця сторінка дає можливість адміністраторам створювати та видаляти дисципліни. Вона містить таблицю зі списком дисциплін і кнопки для керування записами.

6. Сторінка викладачів. На цій сторінці адміністратори можуть керувати списком викладачів: додавати нових або видаляти записи. Сторінка оснащена полем пошуку для швидкого знаходження потрібного викладача.

7. Сторінка освітніх рівнів. Ця сторінка дозволяє управляти освітніми рівнями (бакалавр, магістр тощо). Інтерфейс розроблений із використанням інтуїтивно зрозумілих форм для введення назв рівнів.

8. Сторінка курсів. Адміністратори мають змогу створювати та видаляти курси за допомогою цієї сторінки.

9. Сторінка форм навчання. На цій сторінці адміністратори можуть додавати та видаляти форми навчання, такі як денна чи заочна. Всі зміни відображаються у списку доступних опцій.

10. Сторінка освітніх програм. Ця сторінка дозволяє адміністраторам створювати освітні програми, що включають курси, дисципліни та викладачів. Її інтерфейс орієнтований на зручне структурування навчального процесу.

11. Сторінка опитувань (панель керування). Панель керування опитуваннями призначена для адміністраторів. Вона дозволяє створювати нові опитування або видаляти завершені. Адміністратор може доступність обраного опитування.

12. Сторінка зміни пароля. Ця сторінка доступна тільки адміністраторам. Вона дозволяє користувачам оновити свій пароль, вказавши новий пароль.

Функціональну схему frontend зображено на рисунку 1 додатку В.

3.8 Серверна (back-end) частина проекту

Серверна частина веб-застосунку розроблена на основі NestJS, який є прогресивним фреймворком для створення серверних застосунків із використанням Node.js. Архітектура NestJS базується на модульному підході, що забезпечує масштабованість, зрозумілість і підтримку в довгостроковій перспективі.

Серверна частина веб-застосунку відіграє ключову роль у забезпеченні функціональності, безпеки та ефективної взаємодії між різними компонентами системи. Насамперед вона відповідає за обробку бізнес-логіки, яка включає перевірку даних, отриманих із клієнтської частини, та виконання операцій, таких як створення, зчитування, оновлення або видалення записів. Це дозволяє забезпечити цілісність даних і відповідність виконуваних операцій встановленим правилам.

Крім того, серверна частина реалізує роботу з базою даних, що передбачає забезпечення доступу до збережених даних через ORM (Object-Relational Mapping) або прямі SQL-запити. Завдяки цьому досягається ефективне зберігання даних та їх обробка, що дозволяє швидко отримувати необхідну інформацію, навіть при значному обсязі даних.

Також важливою функцією є забезпечення авторизації та аутентифікації. Цей компонент обмежує доступ до ресурсів системи залежно від ролі

користувача, що дозволяє захистити важливі дані й операції. Наприклад, студенти мають доступ лише до своїх результатів та опитувань, тоді як адміністратори можуть керувати сутностями, такими як викладачі, дисципліни чи форми навчання.

Ще однією важливою складовою є робота з RESTful API, яка забезпечує взаємодію між клієнтською та серверною частинами через чітко визначені HTTP-запити. Це дозволяє клієнтському інтерфейсу динамічно оновлюватися залежно від даних, які отримує сервер, забезпечуючи швидкість і зручність у використанні веб-застосунку.

Головний модуль NestJS цього проекту містить наступні модулі:

1. Answers. Модуль відповідає за отримання, створення, редагування або видалення варіантів відповідей запитань у опитуваннях.
2. Auth. Модуль відповідає за реєстрацію, авторизацію, генерацію JWT та зміну паролю користувача.
3. Comments. Модуль відповідає за отримання, створення, редагування або видалення коментарів до запитань у опитуваннях.
4. Courses. Модуль відповідає за отримання, створення, редагування або видалення курсів.
5. Education forms. Модуль відповідає за отримання, створення, редагування або видалення форм навчання.
6. Educational levels. Модуль відповідає за отримання, створення, редагування або видалення освітніх рівней.
7. Educational programs. Модуль відповідає за отримання, створення, редагування або видалення освітніх програм.
8. Excel. Модуль відповідає за отримання даних з файлів excel табличок.
9. Professors. Модуль відповідає за отримання, створення, редагування або видалення викладачів.
10. Questions. Модуль відповідає за отримання, створення, редагування або видалення запитань у опитуваннях.

11. Quizzes. Модуль відповідає за отримання, створення, редагування або видалення опитувань.

12. Students. Модуль відповідає за всі дії, що може зробити студент: пройти опитування, отримати список освітніх програм за освітнім рівнем та курсом тощо.

13. Subject. Модуль відповідає за отримання, створення, редагування або видалення списку дисциплін.

14. TypeORM. Модуль PrismaModule у NestJS відповідає за інтеграцію з базою даних через ORM Prisma. Він дозволяє налаштувати та використовувати Prisma для виконання запитів до бази даних, таких як створення, оновлення, видалення та отримання записів. Це допомагає спростити взаємодію з базою даних, надаючи зручний API для роботи з моделями, визначеними у схемі Prisma. Зокрема, PrismaModule надає доступ до Prisma Client, який використовується для виконання SQL-запитів на рівні програмного коду. У NestJS цей модуль часто імпортується у інші модулі, щоб дати доступ до функціоналу роботи з даними.

15. Users. Цей модуль відповідає за отримання, створення, редагування або видалення адміністратор панелі керування.

Блок-схеми принципу роботи авторизації та проходження опитування веб-застосунку наведено на рис. 2-3 додатку В.

Інформацію по атрибутам сутностей опитування «Коментар» та «Відповідь» логічної моделі наведено в таблицях 3.1-3.2.

Таблиця 3.1

Структура даних сутності «Коментар»

Атрибут	Тип даних	Опис
ID	Лічильник	Ідентифікаційний номер коментаря
Значення	Текстовий	Текстове значення коментаря

Запитання	Лічильник	Ідентифікаційний номер запитання
Викладач	Лічильник	Ідентифікаційний номер викладача
Створено	Дата	Дата створення
Оновлено	Дата	Дата оновлення

Таблиця 3.2

Структура даних сутності «Відповідь»

Атрибут	Тип даних	Опис
ID	Лічильник	Ідентифікаційний номер відповіді
Значення	Текстовий	Текстове значення коментаря
Створено	Дата	Дата створення
Оновлено	Дата	Дата оновлення

ER-діаграма (Entity-Relationship Diagram, діаграма «сутність-зв'язок») — це графічне представлення структури бази даних, яке відображає сутності (об'єкти) і взаємозв'язки між ними. Така діаграма використовується для планування бази даних і допомагає зрозуміти, як різні компоненти бази даних взаємодіють між собою.

Основні елементи ER-діаграми:

1. Сутність (Entity) — це об'єкт, який має значення для організації або системи. Наприклад, у базі даних для інтернет-магазину сутністю може бути "Користувач" або "Товар".

2. Атрибут (Attribute) — це характеристика сутності. Наприклад, для сутності "Користувач" атрибутами можуть бути "ім'я", "email" або "дата народження".

3. Зв'язок (Relationship) — це відношення між двома або більше сутностями. Наприклад, зв'язок "покупка" може бути між сутностями "Користувач" і "Товар".

4. Кардинальність (Cardinality) — визначає кількість об'єктів, які можуть бути пов'язані між собою. Це може бути один до одного (1:1), один до багатьох (1:N) або багато до багатьох (M:N).

На рис. 3.17 зображено ER-діаграму розробленого веб-застосунку, яка відображає основні сутності системи та взаємозв’язки між ними. Діаграма показує структуру бази даних, що включає таблиці, їхні атрибути та ключі (первинні та зовнішні), які забезпечують цілісність даних.

Основними сутностями є користувачі, опитування, питання та відповіді. Таблиця користувачів зберігає інформацію про облікові записи, включаючи унікальні ідентифікатори, логіни та паролі, а також ролі (адміністратор, викладач, студент). Таблиця опитувань містить метаінформацію про кожне опитування, зокрема його назву, опис, дату створення та статус.

Зв'язок між опитуваннями та питаннями реалізовано через зовнішній ключ, що забезпечує належне групування питань у межах кожного опитування. У свою чергу, таблиця відповідей зв’язана з питаннями, що дозволяє зберігати всі відповіді, надані користувачами.

Також на ER-діаграмі відображено атрибути для реєстрації дій, таких як час створення записів і можливість їхнього логічного видалення. Структура діаграми спрямована на оптимізацію продуктивності бази даних і забезпечення масштабованості системи.

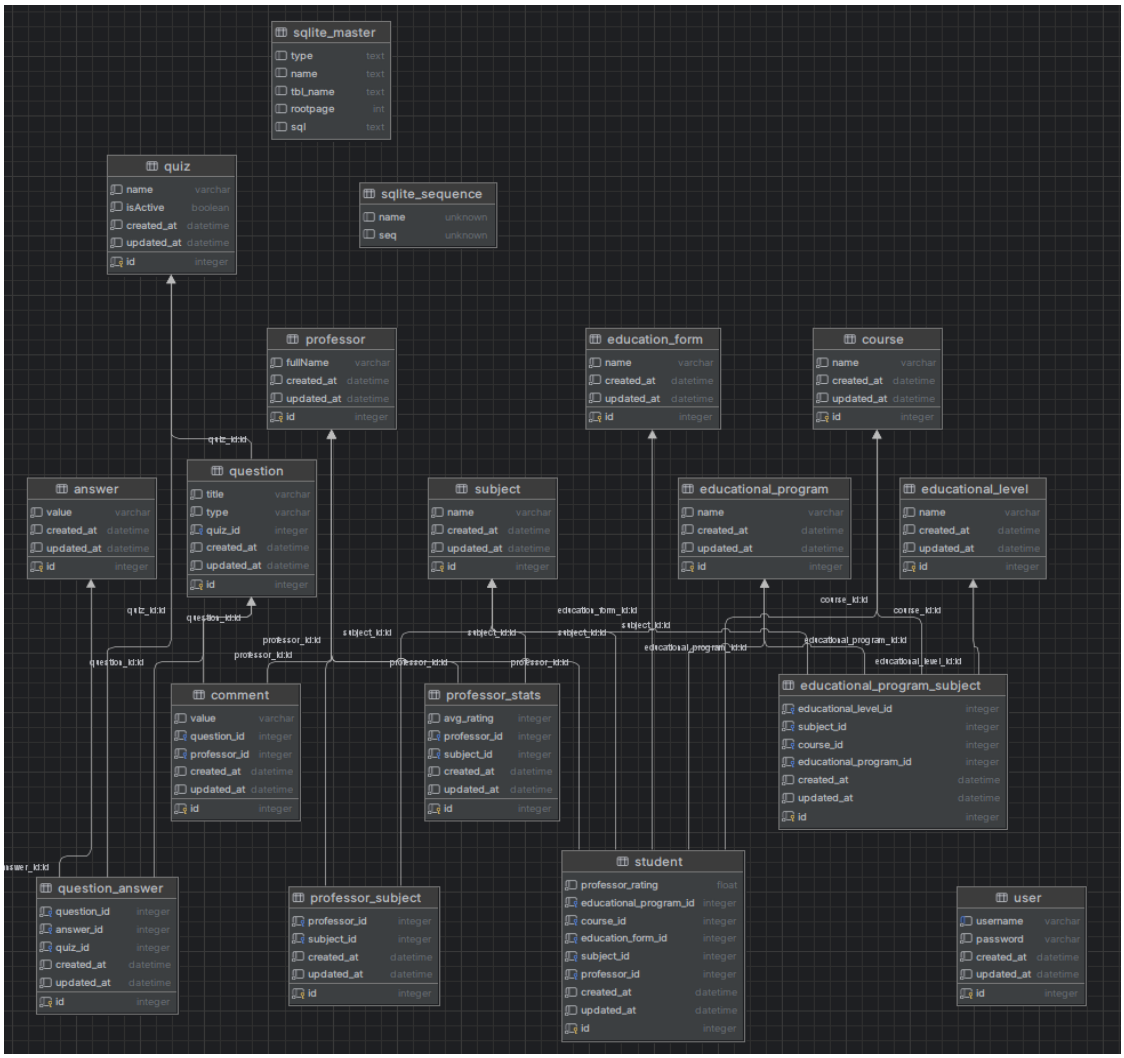

Рисунок 3.18 – ER-діаграма бази даних розробленого веб-застосунку

3.9 Тестування

Тестування є важливою частиною процесу розробки програмного забезпечення, яка забезпечує перевірку працездатності системи, її компонентів та відповідність вимогам. У розробці веб-додатків тестування сприяє виявленню помилок на ранніх етапах і забезпеченню стабільності продукту.

Для тестування швидкості завантаження сторінок було використано DevTools веб-браузеру Microsoft Edge.

Час завантаження сторінки «Опитування» склав 440 мс (див. рис. 3.19). Це є досить добрим показником для сторінки, що містить питання, варіанти відповідей, коментарі тощо.

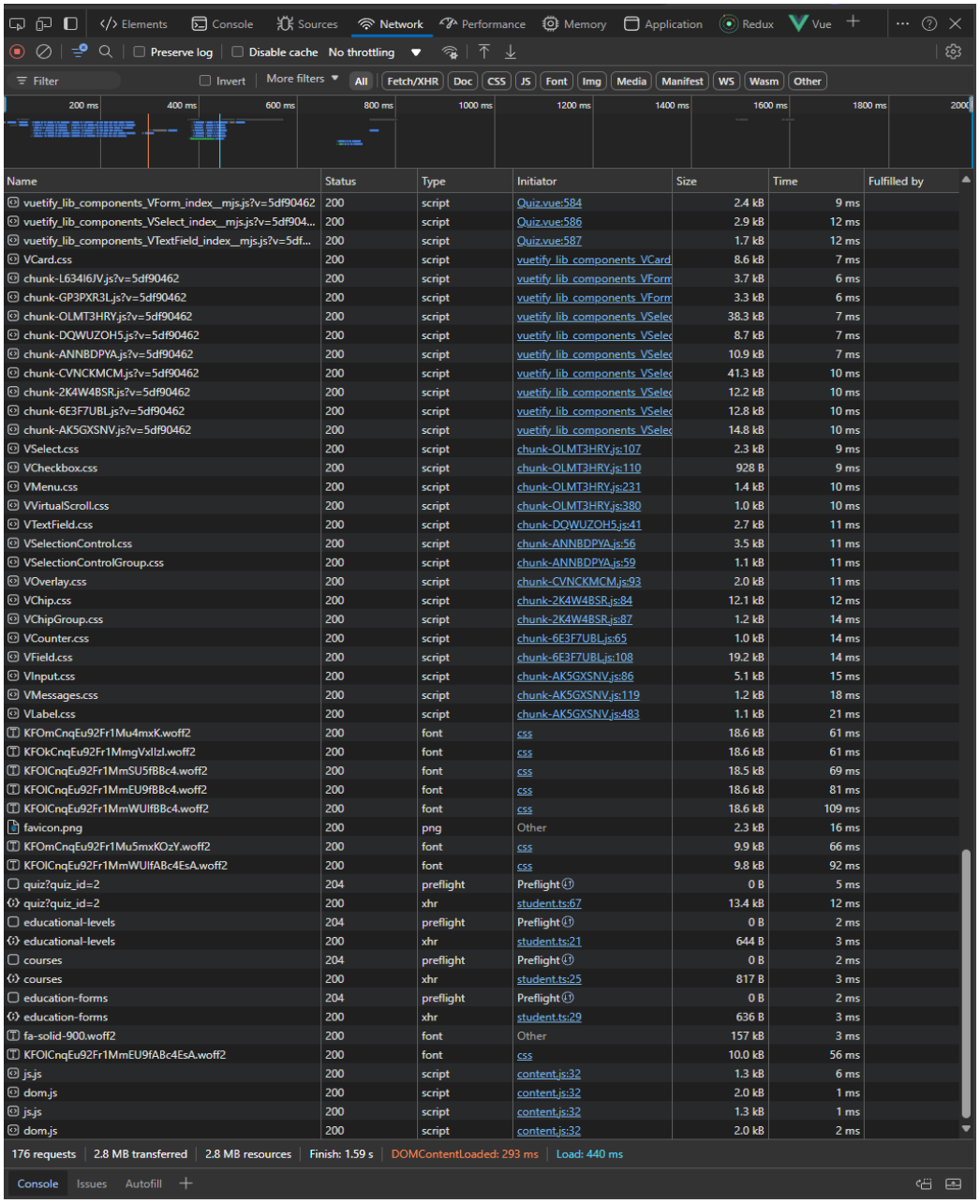

Рисунок 3.19 – Тестування сторінки «Опитування»

Час завантаження сторінки «Авторизація» склав 624 мс (див. рис. 3.20). Це є досить добрим показником для сторінки, що містить текстові поля та виконання запиту до веб-серверу.

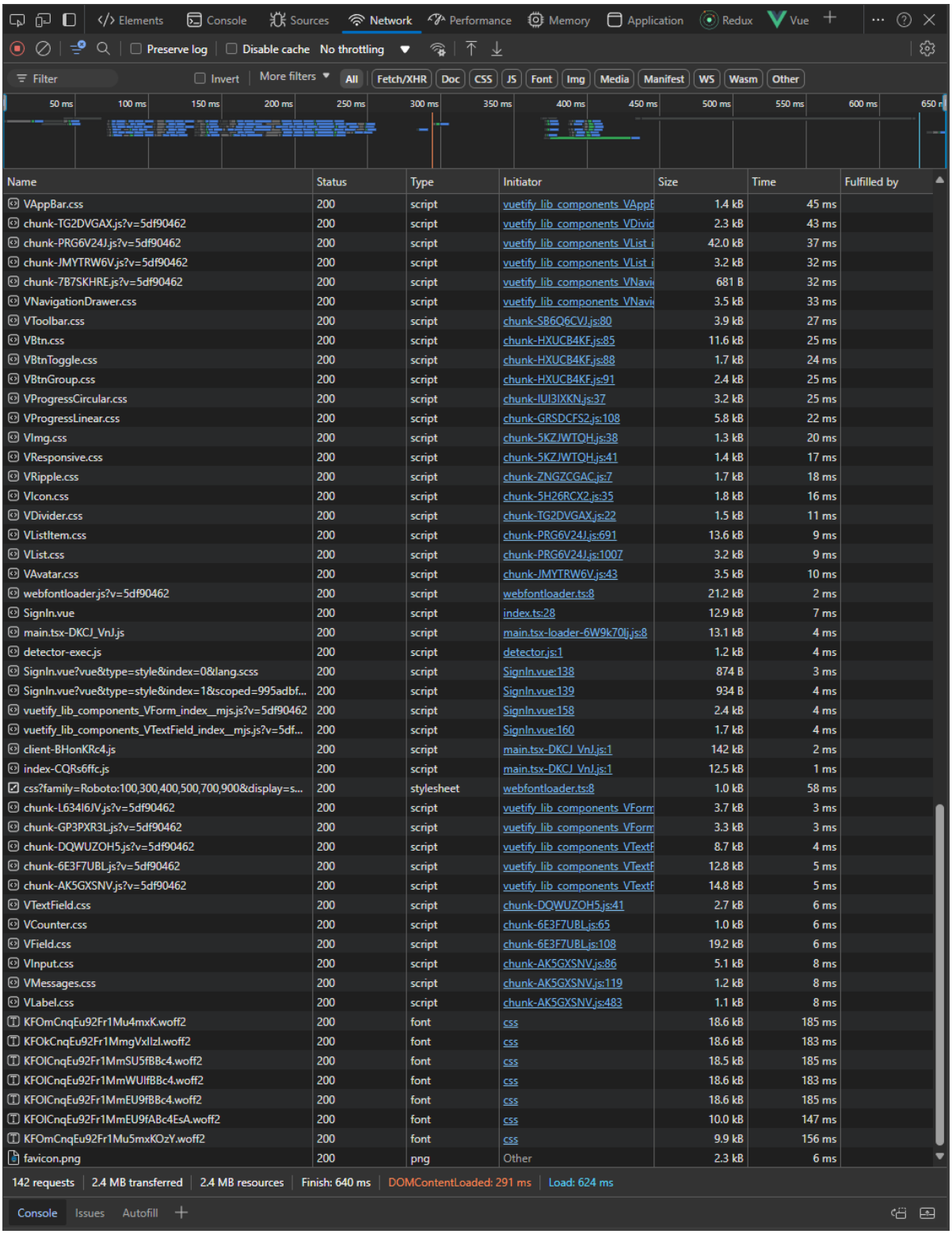

Рисунок 3.20– Тестування сторінки «Авторизація»

Час завантаження сторінки «Статистика» склав 568 мс (див. рис. 3.21). Це є досить добрим показником для сторінки, що містить табличні дані викладачів за певними дисциплінами, кількість пройдених опитувань, середній бал тощо.

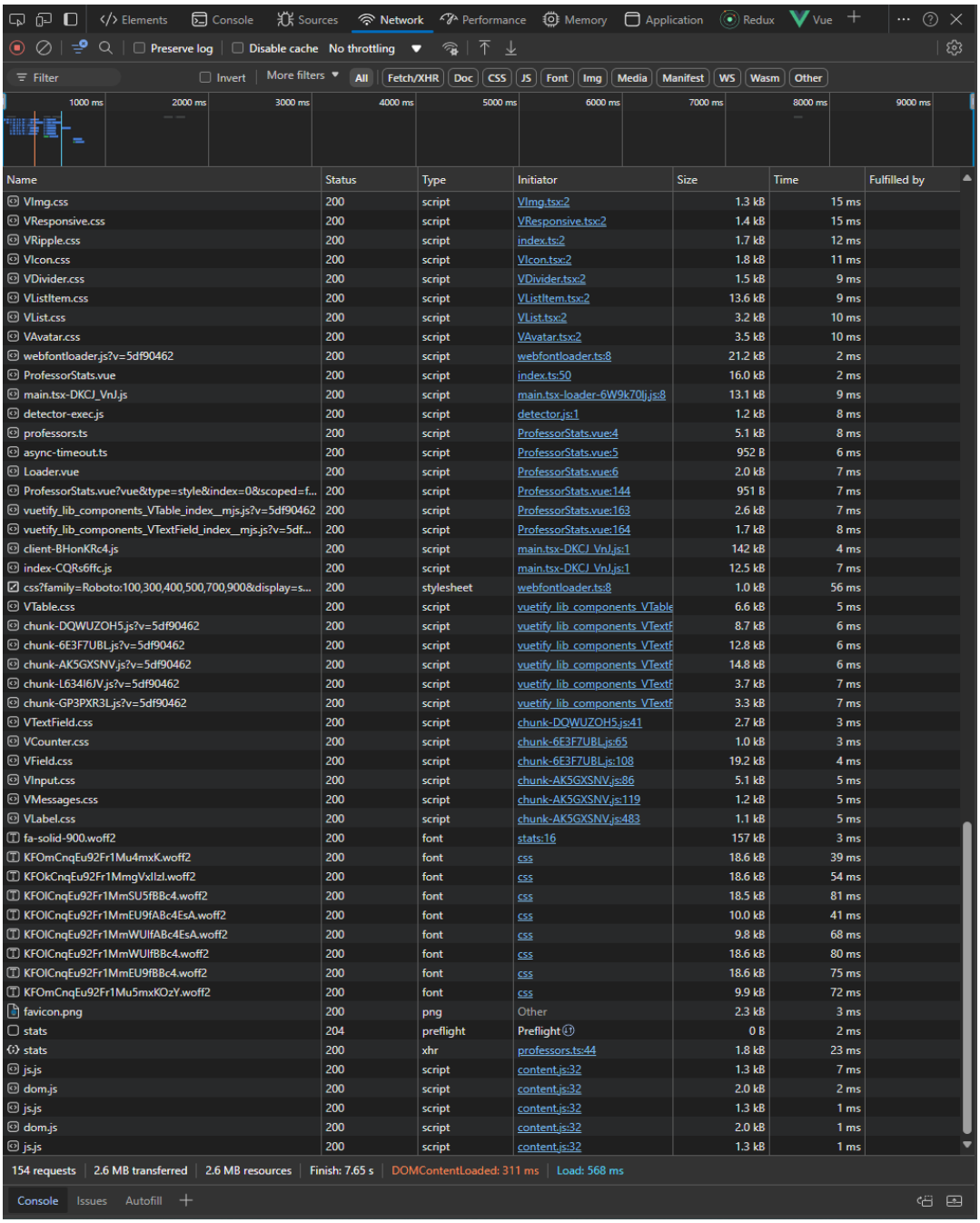

Рисунок 3.21– Тестування сторінки «Статистика»

У процесі тестування були використані інструменти оцінки продуктивності, такі як DevTools, та стандарти ISO/IEC 25010, що дозволило оцінити функціональність і продуктивність системи. Для забезпечення надійності системи проведено перевірку вразливостей із використанням OWASP ZAP, що підтвердило її захищеність від основних атак.

3.10 Висновки до третього розділу

У третьому розділі було розглянуто етапи розробки веб-застосунку для автоматизації проведення опитування здобувачів вищої освіти щодо якості викладання освітніх компонент, що включає розробку загальної архітектури, дизайну, клієнтської та серверної частин проекту, а також процес тестування.

Розробка загальної архітектури проекту забезпечила створення чіткої та ефективної структури системи, що відповідає вимогам до функціональності та масштабованості. У результаті було визначено основні компоненти системи, їх взаємодії та зв'язки, що дозволило побудувати ефективну платформу для збору статистичних даних.

Розробка дизайну проекту включала створення зручного та інтуїтивно зрозумілого інтерфейсу для користувачів, що дозволяє швидко і зручно працювати з даними. Враховуючи сучасні вимоги до інтерфейсу, було реалізовано сучасне оформлення, яке знижує навантаження на користувачів і покращує загальний досвід використання додатку.

Клієнтська частина проекту була розроблена з використанням сучасних веб-технологій, що забезпечують динамічність і зручність у взаємодії з користувачем. Вибір Vue.js дозволив реалізувати ефективну роботу інтерфейсу, який реагує на зміну даних у реальному часі.

Серверна частина проекту охоплює створення API для обробки статистичних даних, забезпечуючи ефективну роботу з великими обсягами інформації та можливість швидкої обробки запитів. Використання NestJS та SQLite дозволило забезпечити високу швидкість обробки запитів і надійність системи.

Інноваційним підходом у розробці стало використання нових алгоритмічних рішень для оптимізації процесів взаємодії між компонентами в реальному часі. Зокрема, реалізовано механізми динамічного оновлення даних за допомогою регулярних HTTP-запитів, що дає змогу зменшити необхідність повного перезавантаження сторінок. Такий підхід значно покращує

користувацький досвід і знижує навантаження на сервер, оскільки відправляються лише зміни, що відбулись, замість цілісних даних. Оновлення відбувається через періодичні запити, що виконуються асинхронно, дозволяючи користувачам отримувати актуальну інформацію без значних затримок.

Тестування проекту проводилось на всіх етапах розробки, що дозволило виявити та виправити помилки на ранніх етапах і забезпечити стабільну роботу системи. Особлива увага була приділена тестуванню продуктивності та безпеки системи, що забезпечує її захищеність та ефективність роботи в умовах високих навантажень. Для перевірки продуктивності застосунку були використані такі методи, як навантажувальне тестування, що імітує одночасну роботу великої кількості користувачів, а також стрес-тестування для оцінки поведінки системи у критичних умовах.

Тестування безпеки включало перевірку на вразливості, зокрема тестування на SQL-ін'єкції, XSS (міжсайтове скриптування) та інші поширені атаки, які можуть становити загрозу для веб-застосунків [9]. Це дозволило захистити систему від потенційних загроз і забезпечити збереження конфіденційної інформації.

У результаті виконаних робіт було створено надійний та ефективний інструмент для автоматизованого збору статистики по викладачам, який задовольняє всі вимоги до функціональності, безпеки та продуктивності.

Крім того, завдяки інтеграції сучасних технологій та високій якості тестування, продукт готовий до подальшого масштабування і вдосконалення, що дозволяє легко додавати новий функціонал у відповідь на нові потреби користувачів.

ВИСНОВКИ

В процесі розробки веб-застосунку для автоматизації проведення опитування здобувачів вищої освіти щодо якості викладання освітніх компонент було виконано комплексну роботу, що включала створення архітектури проекту, розробку дизайну, реалізацію клієнтської та серверної частин, а також тестування. Проект передбачав реалізацію ефективної платформи для збору, аналізу та представлення статистичних даних про викладачів, що полегшує їх обробку і взаємодію з адміністрацією.

Система збирає оцінки здобувачів вищої освіти і обробляє їх, надаючи інформацію. Остаточну оцінку щодо якості викладання освітніх компонент науково-педагогічними працівниками УМСФ на основі цих даних роблять уповноважені на це особи УМСФ.

Основними завданнями, які були вирішені під час розробки, стали:

1. Підбір та аналіз технічних засобів для створення веб-застосунку, які відповідають вимогам до функціональності та безпеки.
2. Розробка вимог до програмного забезпечення, які дозволяють задовольнити потреби викладачів і студентів, спрощуючи процес збору та доступу до статистичних даних.
3. Створення ефективної клієнтської частини з використанням Vue.js, що дозволило реалізувати зручний інтерфейс з динамічними елементами для взаємодії користувачів з системою.
4. Розробка серверної частини на базі фреймворку NestJS, що забезпечило високу продуктивність та надійність обробки даних.

Дизайн застосунку був створений з використанням Vuetify, що дозволило побудувати сучасний, інтуїтивно зрозумілий інтерфейс, відповідний вимогам сучасного веб-дизайну та підвищеного комфортного використання. Використання Vuetify забезпечило зручне розташування елементів управління і відображення статистики, що сприяє зручному та швидкому доступу до необхідної інформації.

Клієнтська частина була розроблена на базі Vue.js, що дозволило ефективно організувати реактивну взаємодію з користувачем. Це дозволяє забезпечити високий рівень динамічності та інтерактивності, що важливо для швидкого відображення змін у статистичних даних.

Серверна частина проекту, побудована на основі NestJS, дозволила створити гнучке API, яке ефективно взаємодіє з базою даних та забезпечує високу швидкість обробки запитів. Це дозволяє надавати користувачам актуальну інформацію в реальному часі без затримок.

У зв'язку з вимогами до якості програмного забезпечення та стандартами оцінки його функціональності, продуктивності та безпеки, одним із важливих аспектів кваліфікаційної роботи є застосування міжнародних стандартів, зокрема ISO/IEC 25010. Це дозволяє ефективно розв'язувати завдання з розробки програмного забезпечення, враховуючи вимоги до якості і відповідно до компетентностей, визначених освітньо-науковою програмою для підготовки магістрів у галузі комп'ютерних наук. Зокрема, це відповідає спеціальним компетентностям, таким як здатність оцінювати та забезпечувати якість IT-проєктів та розробляти програмне забезпечення з урахуванням наявних ресурсів та обмежень.

Крім того, розробка програмного забезпечення відповідно до стандартів є важливою складовою для досягнення результатів навчання, таких як здатність розробляти програмне забезпечення для аналізу даних та оцінювати якість інформаційних систем.

Тестування проекту проводилось за допомогою інструментів для тестування веб-сторінок, таких як DevTools (Microsoft Edge), що дозволило забезпечити стабільну та безпечну роботу застосунку, а також оптимізувати запити до бази даних.

Отже, проект був успішно реалізований і відповідає всім вимогам до функціональності, ефективності та зручності використання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Зінченко А.Ю. Проектування розподілених інформаційних систем на основі використання технології слабозв'язаних компонентів. *Системи та технології*. 2022. №1(63). С. 5-14.

2. Єпік М. О. Проектування бази даних інтелектуальної системи діагностики захворювань. *Вісник Херсонського національного технічного університету.* 2019. №2(69). С. 139-144.

3. Проценко М. М. Аналіз фреймворків як засобів розробки web-додатків. *Міжнародний науковий журнал "Інтернаука".* 2016. №6(1). С. 86-89.

4. Фірсов О. Д., Ульяновська Ю. В., Мормуль М. Ф., Пікулін Д. О. Проектування архітектури веб-застосунку для імітаційного моделювання управління транспортними потоками. *Системи та технології.* 2022. №1(63). С. 70-87.

5. Поворознюк Н. І., Бобрівник К. Є., Грибков С. В. Проектування бази даних модуля студента у системі підтримки вивчення дисциплін. *Наукові праці Національного університету харчових технологій.* 2017. №23(1). С. 7-15.

6. Лебідь О. Ю., Назарян А. А. Порівняння основних механізмів об'єктно-орієнтованого програмування в PHP та JavaScript. Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення. Випуск 55. Частина 1. Тернопіль: Міжнародна наукова інтернет-конференція, 2020, с. 67-68.

7. Ульяновська Ю. В., Назарян А. А. Розробка веб-застосунку центру інформаційних технологій Університету митної справи та фінансів. Економіко-правові та управлінсько-технологічні виміри сьогодення: молодіжний погляд: матеріали міжнародної науково-практичної конференції: у 3 т. Том 3. Дніпро: Університет митної справи та фінансів, 2023, с. 270-272.

8. Лебідь О. Ю., Назарян А. А., Рябоволенко Е. А. Графові бази

даних та їх роль у візуалізації та аналізі даних. Економіко-правові та управлінсько-технологічні виміри сьогодення: молодіжний погляд: матеріали міжнародної науково-практичної конференції: у 3 т. Том 3. Дніпро: Університет митної справи та фінансів, 2023, с. 272-274.

9. Parshyna O. A., Nazarian A. A. Methods of cryptographic data protection. Economic-legal and managerial-technological dimensions of the present: a youth perspective: materials of the international scientific and practical conference: in 3 volumes. Vol 3. Dnipro: Dnipro: University of Customs and Finance, 2023, p. 227-228.

10. О. І. Огірко, Н. В. Галайко Теорія ймовірностей та математична статистика: Навчальний посібник. Львів: Львівський державний університет внутрішніх справ, 2017. 292 с.

11. Лебідь О. Ю., Дашковський Б.О. Роль об’єктно-орієнтованого програмування в розробці великих веб-додатків. Економіко-правові та управлінсько-технологічні виміри сьогодення: молодіжний погляд: матеріали міжнародної науково-практичної конференції: у 3 т. Том 3. Дніпро: Університет митної справи та фінансів, 2023, с. 244-246.

12. Ульяновська Ю. В. Веб-фаєрволи та їх роль у захисті від крос-сайтового скриптингу. Економіко-правові та управлінсько-технологічні виміри сьогодення: молодіжний погляд: матеріали міжнародної науково-практичної конференції: у 3 т. Том 3. Дніпро: Університет митної справи та фінансів, 2023, с. 250-252.

13. Лебідь О. Ю., Кожем’яка О.Л. Деякі питання розробки веб-сервісу надання інформаційної підтримки цивільного населення під час надзвичайних ситуацій воєнного характеру. Економіко-правові та управлінсько-технологічні виміри сьогодення: молодіжний погляд: матеріали міжнародної науково-практичної конференції: у 3 т. Том 3. Дніпро: Університет митної справи та фінансів, 2023, с. 256-258.

14. Костенко В.В., Рудовіл Є.А. Основні аспекти проєктування архітектури сучасних програмних систем. Економіко-правові та управлінсько-

технологічні виміри сьогодення: молодіжний погляд: матеріали міжнародної науково-практичної конференції: у 3 т. Том 3. Дніпро: Університет митної справи та фінансів, 2023, с. 290-292.

15. Червякова Т.І. Напрями використання технологій генеративного штучного інтелекту в організації навчального процесу в закладах вищої освіти. Економіко-правові та управлінсько-технологічні виміри сьогодення: молодіжний погляд: матеріали міжнародної науково-практичної конференції: у 3 т. Том 3. Дніпро: Університет митної справи та фінансів, 2023, с. 306-308.

16. Фірсов О. Д., Маріщук А. В. Нечітка система аналізу бізнес правил предметної області під час створення бази даних. *Системи та технології.* 2022. №2(68). С. 55-56.

17. Рибальченко Л. В., Габорець О. А., Прокопович-Ткаченко Д. І. Кіберстійкість: глобальні загрози та національні стратегії кіберзахисту. *Системи та технології.* 2024. №2(68). С. 96-98.

18. Chechet A. S., Chernykh M. V., Panasiuk Ia. S., Abdullin I. I. Front-end security architecture: protection of user data and privacy. *Systems and technologies.* 2024. №2(68). p. 102-104.

19. Фірсов О. Д., Маріщук А. В. Розроблення нечіткої системи для аналізу бізнес-правил предметної області під час створення бази даних. Економіко-правові та управлінсько-технологічні виміри сьогодення: молодіжний погляд: матеріали міжнародної науково-практичної конференції: у 1 т. Том 1. Дніпро: Університет митної справи та фінансів, 2024, с. 180-182.

20. Олексійчук Ю. Ф., Ольховський Д. М., Ольховська О. В., Андрушків О. М. Проєктування, розробка та тестування web-сервісу для вибору тем дипломних робіт. *Системи та технології.* 2024. №1(67). С. 43-45.

21. Кошова О. П., Черненко О. О., Чілікіна Т. В., Комар І. І. Особливості розробки web-застосунків для системи дистанційного навчання з допомогою бібліотеки React. *Системи та технології.* 2022. №1(65). С. 20-22.

22. Олексійчук Ю. Ф., Ольховська О. В., Ольховський Д. М., Орлова Д. І. Проєктування та розробка web-сервісу для генерування та розсилки PDF-документів. *Системи та технології.* 2022. №1(65). С. 39-40.

23. Dotsenko S. I., Kamenskyi S. S. Architecture Development of Information System of an Enterprise. *Systems and technologies.* 2019. №1(57). p. 36-37.

24. Чанишев Р. І. Деякі соціально-економічні аспекти використання центрів обробки даних. *Системи та технології.* 2024. №1(67). С. 51-52.

25. Voskoboinyk V.O., Savchenko Iu.V., Karpukov L.M., Parshyna O.A., Prokopovych-Tkachenko D. I. Assessment of the state of information security using expert systems. *Systems and technologies.* 2024. №1(67). p. 72-73.

26. Zatserkovnyi R. G., Babych V. I., Plesha M. I., Khmilyarchuk L. I., Shvets O. M. A system to evaluate the robustness of an API under high loads. *Systems and technologies.* 2024. №2(66). p. 43-44.

ДОДАТОК А

КОД FRONT-END ПРОЕКТУ

Main.ts:

```
import { createApp } from 'vue';
import { createPinia } from 'pinia';
import piniaPluginPersistedstate from 'pinia-plugin-persistedstate';
import Notifications from '@kyvg/vue3-notification';
import {
  CategoryScale,
  Chart as ChartJS,
  Legend,
  LinearScale,
  Title,
  Tooltip,
  BarElement,
} from 'chart.js';

import vuetify from './plugins/vuetify';
import { loadFonts } from './plugins/webfontloader';
import router from './router';

import App from './App.vue';
```

```
loadFonts();

const pinia = createPinia();
pinia.use(piniaPluginPersistedstate);

ChartJS.register(
  Title,
  Tooltip,
  Legend,
  BarElement,
  CategoryScale,
  LinearScale
);

createApp(App)
  .use(router)
  .use(pinia)
  .use(vuetify)
  .use(Notifications)
  .mount('#app');
```

ДОДАТОК Б

КОД BACK-END ПРОЕКТУ

app.module.ts:

```
import { Module } from '@nestjs/common';

import { ConfigModule } from '@nestjs/config';

import { APP_GUARD } from '@nestjs/core';

import { TypeOrmModule } from '@nestjs/typeorm';

import { ThrottlerGuard, ThrottlerModule } from '@nestjs/throttler';


import { AuthModule } from 'modules/auth/auth.module';

import { CourseModule } from 'modules/courses/courses.module';

import { EducationalLevelModule } from 'modules/educational-levels/educational-levels.module';

import { UsersModule } from 'modules/users/users.module';

import { EducationFormsModule } from 'modules/education-forms/education-forms.module';

import { EducationalProgramsModule } from 'modules/educational-programs/educational-programs.module';

import { SubjectsModule } from 'modules/subjects/subjects.module';

import { ProfessorsModule } from 'modules/professors/professors.module';

import { QuestionsModule } from 'modules/questions/questions.module';

import { AnswersModule } from 'modules/answers/answers.module';

import { CommentsModule } from 'modules/comments/comments.module';
```

```
import { QuizzesModule } from 'modules/quizzes/quizzes.module';
import { StudentsModule } from 'modules/students/students.module';
import { TypeOrmConfigService } from 'modules/typeorm/typeorm.service';
import { ExcelModule } from 'modules/excel/excel.module';

@Module({
  imports: [
    ConfigModule.forRoot({
      isGlobal: true,
      envFilePath: `.env.${process.env.NODE_ENV}`,
    }),
    ThrottlerModule.forRoot({
      ttl: 60,
      limit: 20,
    }),
    TypeOrmModule.forRootAsync({
      useClass: TypeOrmConfigService,
    }),
    UsersModule,
    AuthModule,
    EducationalLevelModule,
    CourseModule,
```

```
    EducationFormsModule,
    EducationalProgramsModule,
    SubjectsModule,
    ProfessorsModule,
    QuestionsModule,
    AnswersModule,
    CommentsModule,
    QuizzesModule,
    StudentsModule,
    ExcelModule,
  ],
  controllers: [],
  providers: [
    {
      provide: APP_GUARD,
      useClass: ThrottlerGuard,
    },
  ],
})
export class AppModule {}
};
```

ДОДАТОК В

СХЕМИ

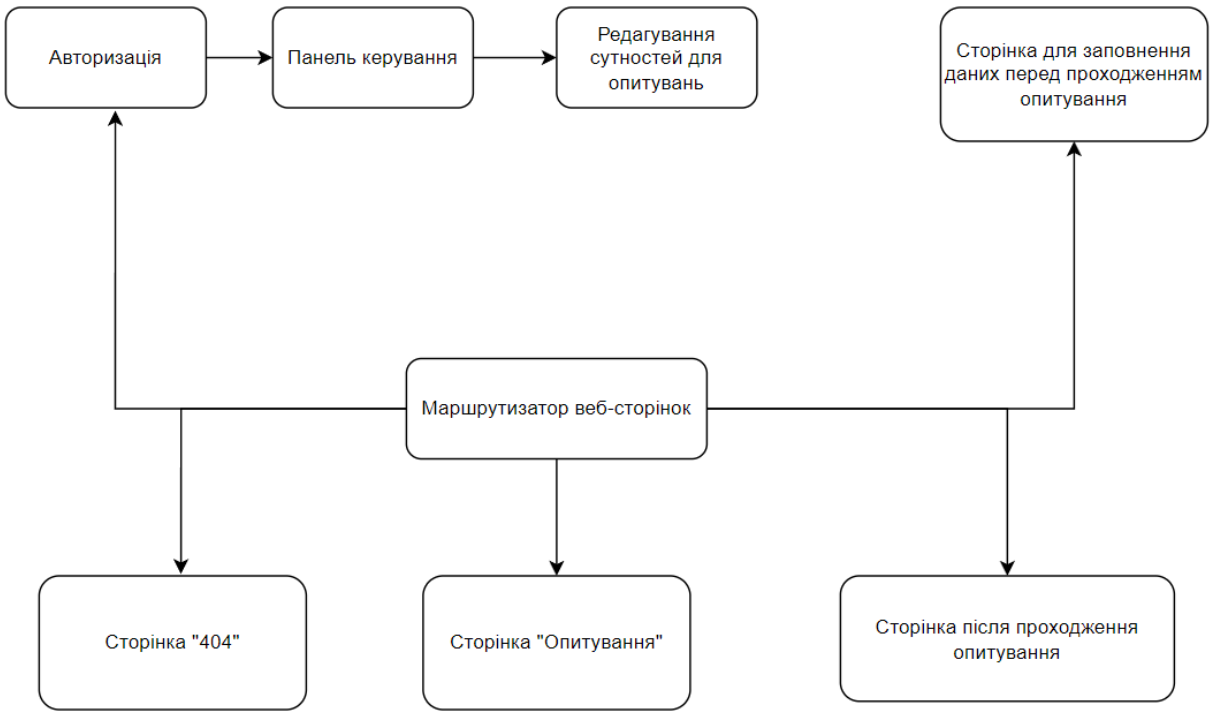

Рисунок 1 – Функціональна схема front-end проекту

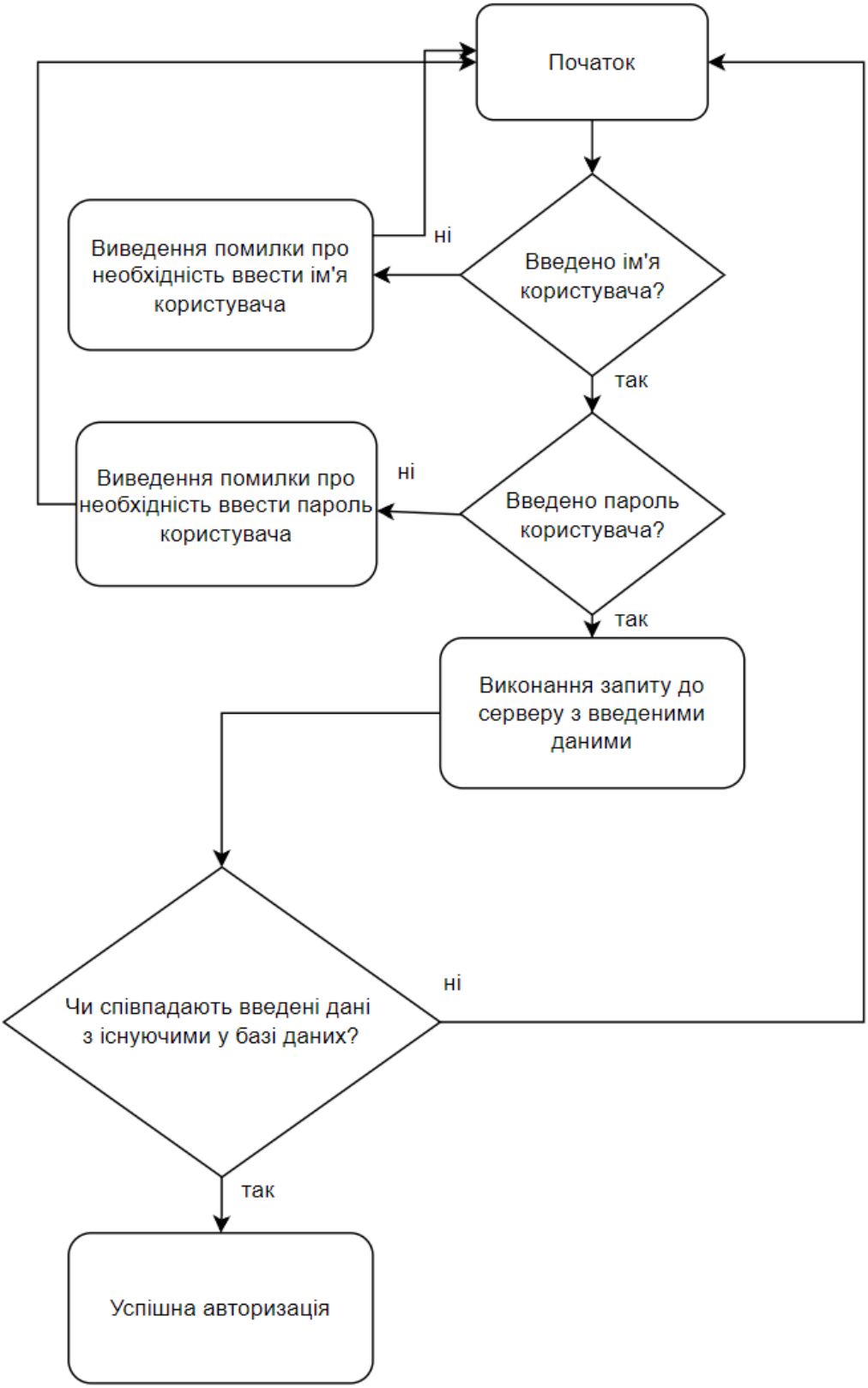

Рисунок 2 – Блок-схема принципу роботи авторизації веб-застосунку

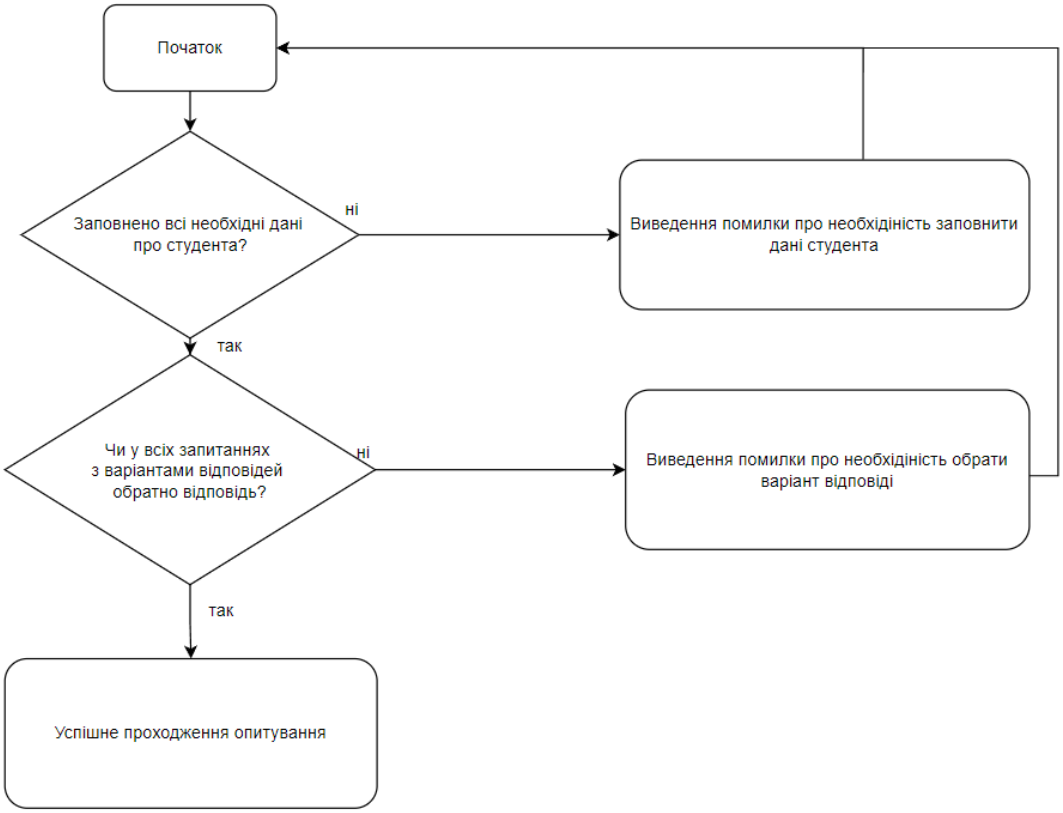

Рисунок 3 – Блок-схема роботи проходження опитування веб-застосунку