Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Дослідження засобів розробки програмного забезпечення для конвертації медіа файлів у текстовий формат.

Виконав: студент VI курсу, групи САм-61
спеціальності 124 Системний аналіз
(шифр і назва спеціальності)

_____ Бачинський А.В.
(підпис) (прізвище та ініціали)

Керівник _____ Марценюк В.П.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Дуда О.М.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 124 Системний аналіз
(шифр і назва спеціальності)

Студенту Бачинському Андрію Васильовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження засобів розробки програмного забезпечення для конвертації медіа файлів у текстовий формат.

Керівник роботи Марценюк Василь Петрович, доктор технічних наук, професор кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*29» листопада 2024 року № 4/7-1140*

2. Термін подання студентом завершеної роботи 19 грудня 2024р.

3. Вихідні дані до роботи Документаційні інтернет-джерела щодо використання веб технологій

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Аналіз середовищ для розробки програмного забезпечення.
2 Аналіз вибраних технологій для роботи з сервером та створення бази даних.
3 Проектування клієнтської частини застосунку.
4 Охорона праці та безпека в надзвичайних ситуаціях. Висновки. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Сенчишин В.С., доцент		
Безпека в надзвичайних ситуаціях	Клепчик В.М., ст. викладач		

7. Дата видачі завдання 19 грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	14.11.24-15.11.24	*Виконано*
2.	Підбір наукових джерел по темі роботи	16.11.24-20.11.24	*Виконано*
3.	Переклад та опрацювання наукових джерел	20.11.24-23.11.24	*Виконано*
	по темі кваліфікаційної роботи		
4.	Виконання дослідження щодо теми	24.11.24-10.12.24	*Виконано*
	Кваліфікаційної роботи		
5.	Оформлення першого розділу	11.12.24-12.10.24	*Виконано*
6.	Оформлення другого розділу	12.12.24-13.12.24	*Виконано*
7.	Оформлення третього розділу	13.12.24-14.12.24	*Виконано*
8.	Виконання завдання до підрозділу «Охорона праці»	08.12.24-09.11.24	*Виконано*
9.	Виконання завдання до підрозділу «Безпека в	10.12.24-12.12.24	*Виконано*
	надзвичайних ситуаціях»		
10.	Оформлення кваліфікаційної роботи	12.12.24-31.12.24	*Виконано*
11.	Нормоконтроль	14.12.24-15.12.24	*Виконано*
12.	Перевірка на плагіат	15.12.24	*Виконано*
13.	Попередній захист кваліфікаційної роботи	16.12.24	*Виконано*
14.	Захист кваліфікаційної роботи	23.12.2024	

Студент _____ Бачинський Андрій Васильович
(підпис) (прізвище та ініціали)

Керівник роботи _____ Марценюк Василь Петрович
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дослідження засобів розробки програмного забезпечення для конвертації медіа файлів у текстовий формат. // Кваліфікаційна робота освітнього рівня «Магістр» // Бачинський Андрій Васильович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група САм-61 // Тернопіль, 2024 // С. 47, рис. – 25, бібліогр. – 13.

Ключові слова: конвертація, середовище, веб-розробка, роль, php, сервер, реєстрація

Кваліфікаційна робота присвячена дослідженню засобів розробки програмного забезпечення для конвертації медіа файлів у текстовий формат.

В першому розділі кваліфікаційної роботи досліджено існуючі середовища для написання коду. Розглянуто найпопулярніші редактори коду для розробки. Було обгрунтовано вибір середовища для реалізації веб-застосунку, порівняння редакторів коду з їх властивостями у таблиці. Розглянуто основні інструменти редакторів коду для покращення роботи із написанням, та його модифікації. Висвітлено адаптивність редакторів та їх розширення у взаємодії із мовами програмування.

В другому розділі кваліфікаційної роботи обґрунтовано вибір технологій та середовище для реалізації бази даних. Було розроблено та реалізовано серверну частину. Створено допоміжну функцію для збереження та захисту інформації. Було обгрунтовано вибір технології для підтримки та взаємодії клієнтської частини застосунку із бекендом. Реалізовано функції для обміну даними. Створено запит для відправлення та отримання відповіді від сервера до фронтенду.

В третьому розділі кваліфікаційної роботи було налаштовано інструменти та сучасні бібліотеки для розробки. Створено інтерфейс клієнтської частини програмного забезпечення. Розглянуто основні функції у клієнтській частині застосунку, основні вкладки для користування застосунком. Розроблено

авторизацію та реєстрацію користувача. Реалізовано платіжну систему для оплати та продовження підписки у веб-застосунку. Висвітленно усі переваги даного застосунку та його принцип роботи.

В четвертому розділі кваліфікаційної роботи було розглянуто питання щодо охорони праці, та безпеки в надзвичайних ситуаціях. Було розглянуто питання забезпечення безпеки в різних сферах діяльності, зокрема в ІТ-сфері та цивільному захисті. Обговорено питання підготовки та перепідготовки керівного складу, органів управління, а також навчання населення діям у надзвичайних ситуаціях. Розглянуто питання щодо систематичного підходу до безпеки, щоб гарантувати не лише збереження здоров’я працівників і громадян, а й ефективне функціонування суспільства загалом.

ANNOTATION

Research of software development tools for converting media files into text format // The educational level "Master" qualification work // Bachynskiy Andriy // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, SAm-61 group // Ternopil, 2024 // P. 47, fig. – 25, references –13

Keywords: conversion, environment, web-development, role, php, server, registration

The qualification work is devoted to the research of software development tools for converting media files into text format.

In the first section of the qualification work, existing environments for writing code were investigated. The most popular code editors for development are considered. The choice of the environment for the implementation of the web application was justified, the comparison of code editors with their properties in the table. The main tools of code editors for improving work with writing and its modification are considered. The adaptability of editors and their extension in interaction with programming languages is highlighted.

In the second section of the qualification work, was justified the choice of technologies and environment for implementing the database was justified. The server-side part was developed and implemented. A helper function for storing and securing information was created. The choice of technology for supporting and interacting with the client-side application and backend was justified. Functions for data exchange were implemented. A request for sending and receiving responses between the server and the frontend was created.

In the third section of the qualification work, tools and modern libraries for development were configured. The interface of the client part of the software has been created. The main functions in the client part of the application, the main tabs for using the application are considered. User authorization and registration has been developed.

A payment system has been implemented for paying and renewing a subscription in the web application. All the advantages of this application and its principle of operation are highlighted.

The fourth section of the qualification work addressed the issues of labor protection and safety in emergency situations. The issue of ensuring security in various areas of activity was considered, in particular in the IT sector and civil protection. The issue of training and retraining of management, management bodies, as well as training the population in emergency situations was discussed. The issue of a systematic approach to security was considered in order to guarantee not only the preservation of the health of workers and citizens, but also the effective functioning of society as a whole.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

JS – JavaScript.

Vue – JavaScript фреймворк для створення інтерфейсів користувача на основі моделей даних.

JSON – JavaScript object notation.

Model – модель об'єкта.

HeidiSQL – відкритий клієнт для управління базами даних.

DOM – об'єктна модель документа.

VScode – редактор коду.

i18n – процес мовної адаптації застосунку.

Routing – маршрутизація для покращення параметрів посилань.

ЗМІСТ

ВСТУП

Актуальність теми. Сучасне суспільство стикається з постійно зростаючим обсягом мультимедійних даних, які потребують ефективної обробки та збереження. Конвертація аудіо та відео в текстовий формат має велике значення для підвищення доступності інформації, створення баз даних, аналізу тексту та автоматизації рутинних процесів. Важливість цього напряму підтверджується широким використанням у журналістиці, освіті, судовій практиці, медицині та інших сферах.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Магістр» є розробка програмного забезпечення конвертації медіа файлів у текстовий формат, та дослідження вибраних технологій для реалізації бази даних та інтерфейсу. Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

- Дослідити процеси для розробки веб-сторінок.
- Проаналізувати обрані технології та методи для розробки.
- Розробити базу даних програмного забезпечення.
- Реалізувати реєстрацію та збереження даних користувачів.
- Розмістити веб-застосунок на віддаленому сервері.

Об’єкт дослідження. Функція зчитування голосових повідомлень, звуків, для повного розпису усієї інформації у текст.

Предмет дослідження. Методи розробки програмного забезпечення для конвертації медіа файлів у текстовий формат.

Наукова новизна одержаних результатів кваліфікаційної роботи полягає у тому, що розроблений веб-застосунок дозволяє конвертувати медіа та аудіо файли у текст, завдяки розробці новітніх функцій зчитання медіа файлів це допомогає користувачам отримувати текстову інформацію із великих медіа файлів.

Практичне значення одержаних результатів. Створено веб-застосунок, який конвертує усі медіа та аудіо файли у текстовий формат і може

використуватись у різних сферах, з можливістю подальшого оновлення та модифікації.

Апробація результатів магістерської роботи. Основні результати проведених досліджень обговорювались на XII науково-технічній конференції «Інформаційні моделі, системи та технології» Тернопільського національного технічного університету імені Івана Пулюя, м. Тернопіль, 2024 р. (Додаток А)

1 АНАЛІЗ СЕРЕДОВИЩ ДЛЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Аналіз середовища VS Code

Visual Studio Code, або VS Code, є одним із найпопулярніших редакторів коду серед розробників завдяки своїй універсальності, простоті використання та багатофункціональності. Цей редактор був розроблений компанією Microsoft і доступний як безкоштовне рішення з відкритим вихідним кодом, що зробило його доступним для широкої аудиторії.

Основна перевага VS Code полягає у його розширюваності. Через вбудований маркетплейс розширень можна налаштувати редактор під будь-які потреби: додати підтримку нових мов програмування, інтегрувати інструменти для роботи з базами даних, розгортання на хмарних платформах, тестування коду чи форматування. Наприклад, розширення Prettier забезпечує автоматичне форматування коду, ESLint дозволяє підтримувати чистоту й відповідність коду стандартам, а Live Server дає змогу миттєво переглядати зміни у веб-додатках (див. рис. 1.1).

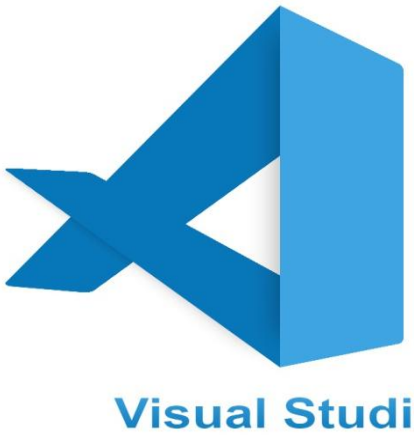


Рисунок 1.1 – Логотип Visual Studio Code

Інтеграція з Git і GitHub є ще однією сильною стороною VS Code. Завдяки зручному інтерфейсу можна легко відслідковувати зміни, виконувати коміти,

створювати гілки чи вирішувати конфлікти прямо з редактора. Це робить його ідеальним інструментом для командної розробки або роботи з проєктами, які потребують системи контролю версій.

Функції автозавершення та інтелектуальної підтримки коду роблять VS Code особливо корисним для новачків і професіоналів. За допомогою вбудованого IntelliSense редактор може пропонувати підказки під час написання коду, перевіряти синтаксис та навіть надавати інформацію про функції, методи чи змінні. Це значно прискорює процес розробки й зменшує кількість помилок.

VS Code підтримує роботу з багатьма мовами програмування без додаткових налаштувань, серед яких JavaScript, TypeScript, Python, C++, HTML і CSS. При цьому завдяки плагінам можна додати підтримку навіть для менш популярних мов чи спеціалізованих технологій. У поєднанні з терміналом, який інтегрований безпосередньо в редактор, це створює повноцінне середовище для роботи без необхідності перемикатися між кількома програмами.

Загалом Visual Studio Code вважається універсальним інструментом, який підходить для різних типів завдань: від створення невеликих веб-сайтів до роботи з великими програмними проєктами. Його популярність підтверджується активною спільнотою користувачів і регулярними оновленнями, які додають нові функції та покращують продуктивність.

1.2 Огляд існуючих середовищ для веб-розробки

Середовища для веб-розробки та редактори коду є важливими інструментами для сучасних програмістів. Вони забезпечують ефективність роботи, спрощують процес створення, тестування та налагодження веб-додатків, а також дозволяють розробникам зосередитися на творчій частині роботи, мінімізуючи технічні труднощі.

Середовище веб-розробки складається з програмних інструментів, що підтримують повний цикл розробки: від написання коду до його тестування й розгортання. Вибір правильного середовища залежить від потреб проєкту, досвіду розробника та вимог до інтеграції з іншими інструментами. Популярні середовища, такі як Visual Studio Code, JetBrains WebStorm або Sublime Text,

пропонують інтеграцію з системами контролю версій, підтримку плагінів, автоматичне завершення коду, а також вбудовані засоби для тестування та налагодження.

Редактори коду є основним інструментом, з яким розробник взаємодіє щодня. Їхня зручність та функціональність безпосередньо впливають на продуктивність. Більшість сучасних редакторів забезпечують підтримку різноманітних мов програмування, що робить їх універсальними. Наприклад, Visual Studio Code славиться своєю адаптивністю через широкий вибір розширень, які додають можливості для роботи з HTML, CSS, JavaScript, TypeScript, Python та багатьма іншими мовами. WebStorm орієнтований на більш професійний рівень і пропонує глибоку інтеграцію з інструментами розробки JavaScript. Sublime Text виділяється своєю швидкістю та простотою, що робить його зручним для швидких змін у коді.

Важливість цих інструментів важко переоцінити. Вони не лише економлять час, але й допомагають уникнути помилок завдяки автоматичному форматуванню, перевірці синтаксису та іншим функціям. Крім того, правильний вибір редактора сприяє кращій організації роботи. Наприклад, інтеграція з системами контролю версій, такими як Git, дозволяє зручно керувати кодом, зберігати історію змін і спільно працювати над проєктами. Функції автозавершення та рефакторингу значно прискорюють розробку і зменшують кількість рутинної роботи.

Правильний підхід до використання цих інструментів передбачає не лише їх вибір, але й адаптацію до конкретних потреб проєкту.

1.3 Порівняння редактору коду VScode з іншими середовищам

VScode вирізняється швидкістю, простотою використання та широкими можливостями кастомізації. Він безкоштовний, має вбудовану підтримку Git, інтелектуальний автодоповнювач IntelliSense і тисячі розширень для роботи з різними мовами програмування, фреймворками та інструментами, такими як Docker чи Jupyter Notebook. Завдяки вбудованому терміналу та функції Live

Share для командної роботи VS Code підходить як для початківців, так і для досвідчених розробників.

У порівнянні з іншими редакторами, такими як Sublime Text, Atom або Vim, VS Code вирізняється більшою функціональністю та активною підтримкою спільноти, хоча поступається Sublime у швидкості роботи з великими файлами та Vim у мінімалізмі.

Для великих Java-проєктів IntelliJ IDEA може бути кращим вибором через глибоку інтеграцію з екосистемою Java.

VS Code особливо корисний, якщо потрібна універсальність, інтеграція з сучасними технологіями та безкоштовний доступ до розширених функцій.

Таблиця 1.1 – Порівняння з іншими редакторами

Функція / Редактор	VS Code	Sublime Text	Atom	Vim
Вартість	Безкоштовно	Безкоштовно / $99	Безкоштовно	Безкоштовно
Швидкість роботи	Швидкий	Дуже швидкий	Трохи повільний	Миттєвий
Можливість кастомізації	Висока (розширення, теми)	Середня	Висока	Складна
Підтримка розширень	Велика (Marketplace)	Середня	Велика (у минулому)	Немає
Вбудована підтримка Git	Так	Ні	Ні	Ні
Робота з великими файлами	Добре	Чудово	Погано	Добре

У таблиці 1.1 продемонстровано порівняння сучасних середовищ коду, їх вартість, усі позитивні та негативні сторони використання, також порівняно швидкість роботи у кожному редакторі.

1.4 Сучасні методи створення бази даних

Сучасні методи створення баз даних поєднують класичні підходи до проєктування з новітніми технологіями, забезпечуючи ефективність,

масштабованість і зручність роботи. На етапі проєктування широко застосовуються концептуальне моделювання (наприклад, ER-діаграми), логічне моделювання з нормалізацією даних та фізичне моделювання з урахуванням конкретної СУБД.

Використовуються як реляційні бази даних, такі як PostgreSQL або MySQL, так і NoSQL-рішення для нереляційних структур, зокрема MongoDB, Redis чи Neo4j. Зростає популярність NewSQL, що об'єднують реляційну модель з можливостями горизонтального масштабування, як у CockroachDB чи Google Spanner. Хмарні сервіси (DBaaS), такі як Amazon RDS чи Azure Cosmos DB, спрощують розгортання, автоматичне масштабування та управління базами даних. Активно застосовуються ORM-фреймворки, що дозволяють працювати з базами через програмні об'єкти (наприклад, SQLAlchemy чи Hibernate), а також інструменти для автоматизації, такі як Liquibase чи Flyway, які інтегруються з CI/CD-процесами у DevOps.

Контейнери Docker і оркестрація Kubernetes дозволяють швидко розгортати бази даних як частину кластерних додатків. Для великих систем важливі розподілені бази з реплікацією та шардінгом для підвищення доступності та масштабованості.

API, такі як REST чи GraphQL, забезпечують інтеграцію баз даних із зовнішніми додатками, а для великих обсягів даних все частіше використовуються екосистеми big data на базі Hadoop чи Apache Spark. Таким чином, сучасні бази даних орієнтовані на оптимізацію під конкретні потреби, обсяг даних та архітектуру системи.

Для веб-розробки та бекенду найкращими методами створення баз даних є такі:

1. Вибір правильної архітектури бази даних: для динамічних веб-додатків зазвичай використовуються реляційні бази даних (PostgreSQL, MySQL) для структурованих даних або NoSQL-рішення (MongoDB, Redis) для швидкої обробки JSON-подібних документів чи кешування.

2. Проєктування API-залежної моделі: створення структури бази з урахуванням запитів, які виконуватимуться через API, щоб забезпечити швидкість і оптимізувати кількість запитів до бази.

3. Використання ORM: інтеграція інструментів об'єктно-реляційного відображення (наприклад, SQLAlchemy для Python, TypeORM для Node.js) для роботи з базою через програмний код, що зменшує складність запитів і помилки.

4. Міграції бази даних: застосування інструментів, таких як Sequelize (Node.js), Django Migrations (Python), або Flyway, для автоматизації змін у структурі бази під час оновлень бекенду.

5. Використання кешування: впровадження Redis чи Memcached для зберігання часто запитуваних даних і зниження навантаження на основну базу.

6. Денормалізація для високонавантажених бекендів: у випадках, коли потрібна максимальна продуктивність, часткова денормалізація (наприклад, дублювання даних) може прискорити складні запити.

7. Хмарні сервіси баз даних: для веб-додатків популярні хмарні рішення, такі як Firebase Realtime Database, Amazon DynamoDB або Supabase, які надають готові до використання бекенд-рішення.

8. Інтеграція з CI/CD: забезпечення автоматичного тестування та оновлення бази через пайплайни CI/CD для узгодженості структури бази з кодом бекенду.

9. Резервне копіювання та аварійне відновлення: впровадження автоматичних бекапів і планів відновлення бази для забезпечення стійкості додатка.

Ці методи допомагають створити стабільну, швидку та гнучку базу даних, яка відповідає потребам сучасного веб-додатку.

1.5 Висновок до першого розділу

В першому розділі кваліфікаційної роботи досліджено існуючі середовища для написання коду. Розглянуто найпопулярніші редактори коду для розробки. Було обгрунтовано вибір середовища для реалізації веб-застосунку, порівняння редакторів коду з їх властивостями у таблиці. Розглянуто основні інструменти

редакторів коду для покращення роботи із написанням, та його модифікації. Висвітлено адаптивність редакторів та їх розширення у взаємодії із мовами програмування.

2 АНАЛІЗ ВИБРАНИХ ТЕХНОЛОГІЙ ДЛЯ РОБОТИ З СЕРВЕРОМ ТА СТВОРЕННЯ БАЗИ ДАНИХ

2.1 Обґрунтування вибору мови SQL для створення бази даних

SQL (Structured Query Language) – це мова програмування, яка використовується для роботи з базами даних. Вона дозволяє зберігати, змінювати, отримувати та видаляти дані в організованих структурах (таблицях).

Переваги мови SQL полягають в наступному:

Простота використання: SQL має зрозумілу синтаксичну структуру, схожу на звичайну англійську мову.

Швидкість і ефективність: SQL оптимізована для роботи з великими обсягами даних. Вона дозволяє швидко знаходити та обробляти інформацію, навіть якщо її дуже багато.

Універсальність: SQL підтримують більшість сучасних систем управління базами даних (БД), таких як MySQL, PostgreSQL, SQLite, Microsoft SQL Server, Oracle тощо.

Навчившись одного разу, можна працювати з різними БД. Структурованість даних: Дані зберігаються в таблицях з чіткою організацією (рядки – записи, стовпці – характеристики). Це зручно для аналізу та зберігання. Потужність: SQL дозволяє виконувати складні операції:

- Фільтрувати та сортувати дані.
- Проводити підрахунки, наприклад, середнє значення або суму.
- З’єднувати дані з різних таблиць у єдине ціле.

2.2 Аналіз технології HeidiSQL для керуванням бази

HeidiSQL – це проста і зручна програма для роботи з базами даних. Вона допомогла мені підключатися до сервера з моєю базою даних SQL Server, і працювати з нею через зрозумілий графічний інтерфейс.

Завдяки HeidiSQL було створено таблиці з авторизованими користувачами, можливість редагувати дані, виконувати SQL-запити, робити

резервні копії чи переносити бази даних між серверами. Ще деякі переваги цієї технології полягають в тому, що вона безкоштовна і не потребує багато ресурсів. Також якщо надати доступ до бази іншим розробникам, вони теж в реальному часі можуть працювати із моїм проектом та вносити зміни, оновлювати запити та інше.

Великим плюсом є те, що технологія HeidiSQL, має свою історію змін запитів чи інших дій зі сторони розробника. Просто все контролювати, та відслітковувати останні дії роботи з проектом.

2.3 Взаємодія бази даних із клієнтською частиною застосунку

JavaScript взаємодіє з базою даних SQL через серверні середовища, з'єднання налаштовується за допомогою конфігурації (хост, користувач, пароль, база даних), а SQL-запити виконуються через методи. У браузері прямий доступ до SQL обмежений, тому взаємодія відбувається через API, де сервер отримує запити від фронтенду, обробляє їх і повертає дані (див. рис. 2.1).

```
const mysql = require('mysql2');
const connection = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    database: 'test'
});

connection.query('SELECT * FROM users', (err, results) => {
    if (err) throw err;
    console.log(results);
});
```

Рисунок 2.1 – Доступ до бази даних через API

У застосунку використовується як раніше зауважено база даних, створена мовою програмування SQL. Яка підтримує взаємодію та звязок за допомогою HeidiSQL.

Основною метою було розробити взаємодію сервера та клієнтською частиною, а саме запити та відповіді. Як тільки користувач виконував ту чи іншу функцію в застонуку, вона зберігалась в історії бази, так щоб при наступному відвіданню застосунку, користувач міг переглянути свої останні дії.

Також у кожного користувача який вже пройшов реєстрацію та був авторизований, дані повинні зберігатись та надійно шифруватися у базі, під особливим ключем, так щоб при спробі хаку бази даних, дані користувачів виявити було неможливо.

2.4 Реалізація власного Endpoint у застосунку

Endpoint**)** у програмуванні – це "точка входу" для взаємодії з програмою через мережу. Це адреса (URL), за якою можна отримати або відправити дані до сервера. У моєму застосунку я надсилаю запит на власний Ендпоінт, щоб сервер повернув мені список вже зареєстрованих користувачів. Також Ендпоінт сам визначає, яка саме дія виконується, отримання даних, створення, оновлення чи видалення (див. рис. 2.2).

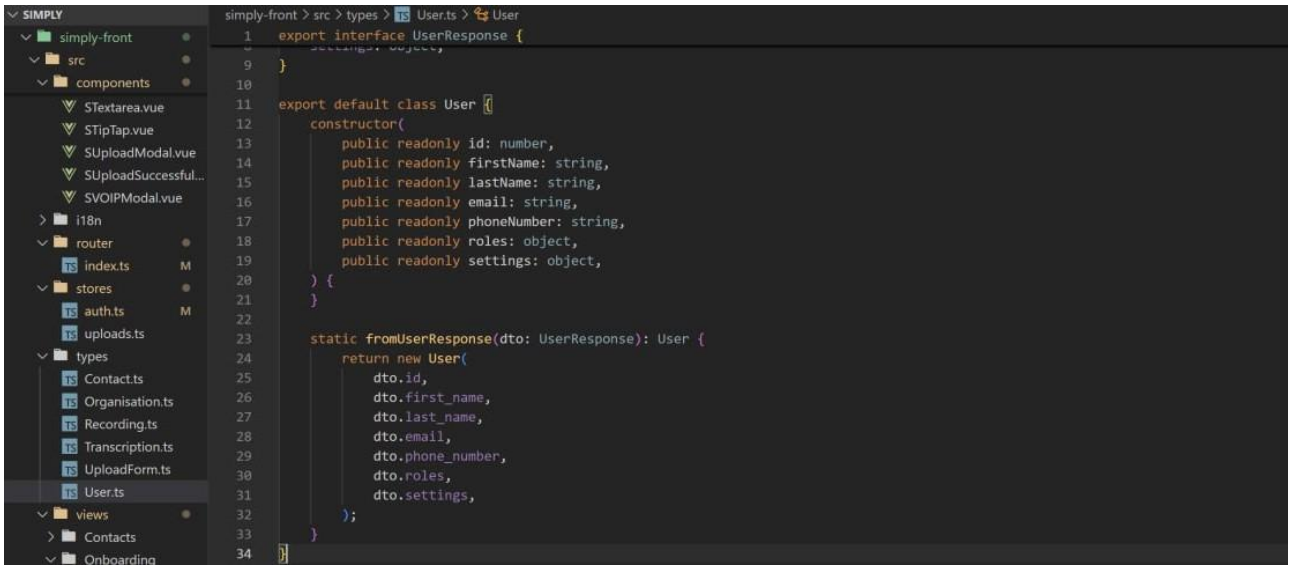


Рисунок 2.2 – Власний Endpoint

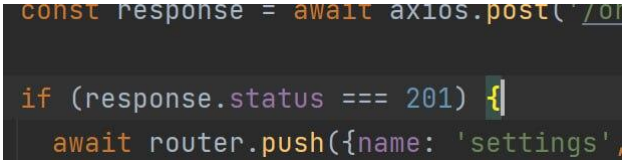
Окрім відповіді про зареєстрованих користувачів, була реалізована функція про зберігання даних на сервері про користувачів через Ендпоінт. А саме це були їхні імена, адреси, та дата реєстрації. Ці дані необхідно оновлювати

та збегірати, для того, щоб користувач зміг відновити свої дані в разі їх траті, наприклад, забув пароль або хоче змінити свою поштову адресу (див. рис. 2.3).

id	name	logo	address	postal	city	created_at	updated_at	deleted_at
1	Simply	(NULL)	(NULL)	(NULL)	(NULL)	2023-09-15 07:42:16	2023-09-15 07:42:16	(NULL)

Рисунок 2.3 – Дані користувачів у базі даних

Функція, яка виконувала запит на сервер для отримання та обробки даних, була написана на мові програмування JavaScript, за допомогою одного з найпопулярніших фреймворків Vue (див. рис. 2.4).



```
const response = await axios.post('/o
if (response.status === 201) {
  await router.push({name: 'settings'
```

Рисунок 2.4 – Функція для отримання даних

2.5 Висновок до другого розділу

В другому розділі кваліфікаційної роботи обґрунтовано вибір технологій та середовище для реалізації бази даних. Було розроблено та реалізовано серверну частину. Створено допоміжну функцію для збереження та захисту інформації. Було обгрунтовано вибір технології для підтримки та взаємодії клієнтської частини застосунку із бекендом. Реалізовано функції для обміну даними. Створено запит для відправлення та отримання відповіді від сервера до фронтенду.

3 ПРОЕКТУВАННЯ КЛІЄНТСЬКОЇ ЧАСТИНИ ЗАСТОСУНКУ

3.1 Створення застосунку на мові програмування JavaScript

JavaScript – це мова програмування, яка використовується для створення інтерактивних веб-додатків. Вона працює в браузерах і дозволяє додавати динамічну поведінку до веб-сторінок, наприклад, обробляти кліки, змінювати контент без перезавантаження, створювати анімації. JavaScript підтримує різні парадигми програмування (об'єктно-орієнтоване, функціональне) та широко використовується для серверного програмування (з Node.js). Це одна з ключових мов для сучасної веб-розробки.

Ось основні переваги JavaScript для розробки:

1. Кросплатформеність: працює в усіх сучаснихбраузерах без потреби додаткових налаштувань.

2. Інтерактивність: дозволяє створювати динамічний контент, анімації та інтерактивні елементи на сторінках.

3. Широкий екосистемний інструментарій: величезна кількість бібліотек і фреймворків, таких як React, Angular, Vue.

4. Повноцінний стек розробки: використовується як на фронтенді (браузери), так і на бекенді (Node.js).

5. Простота вивчення: синтаксис JavaScript легкий для початківців.

6. Гнучкість: підтримує різні парадигми програмування – процедурну, функціональну та об’єктно-орієнтовану.

7. Велика спільнота: безліч готових рішень, документації та підтримки для розробників.

8. Реактивність: здатність миттєво реагувати на дії користувача без перезавантаження сторінки.

9. Інтеграція з іншими технологіями: легко працює разом із HTML, CSS та сторонніми API.

10. Швидкий розвиток: сучасні стандарти та фреймворки постійно вдосконалюють її можливості.

3.2 Підключення та запуск програми за допомогою фреймворка Vue.js

Vue.js – це прогресивний JavaScript-фреймворк для побудови інтерфейсів користувача. Його основна мета – забезпечити ефективну, масштабовану та зручну розробку веб-додатків. Vue.js поєднує найкращі ідеї з інших фреймворків, таких як Angular та React, зберігаючи простоту та гнучкість. Хоча Vue.js не є номером 1 в списку фреймворків для JavaScript, він відрізняється від інших своєю постотою написання функцій та створення різних інтерактивностей.

Vue.js можна віднести до найкращих фреймворків для створення не великих проектів, при цьому він не буде займати стільки пам’яті як от React.js як продемонстровано у таблиці 3.1.

Таблиця 3.1 – Порівняння Vue.js з іншими фреймворками

Параметр	Vue.js	React	Angular	Svelte
Простота навчання	Дуже проста, інтуїтивна	Помірна, залежить від JSX	Складна через TypeScript і концепції	Дуже проста
Розмір фреймворку	~20-30 КБ	~30-40 КБ	Великий (~500 КБ)	Дуже малий (~10-20 КБ)
Реактивність	Двосторонній зв'язок	Односторонній (через state management)	Двосторонній зв'язок	Вбудована реактивність
Маршрутизація	Офіційна бібліотека Vue Router	Потребує сторонніх бібліотек	Вбудована	Потребує сторонніх бібліотек
Підтримка SSR	Через Nuxt.js	Через Next.js	Через Angular Universal	Можлива, але потребує додаткових зусиль
Розробка мобільних додатків	Vue Native, Quasar	React Native	Ionic	В процесі розробки

При створенні веб-застосунку, було підключено фреймворк за допомогою Vite (див. рис. 3.1).

```
npm create vite@latest my-vue-project --template vue
```

Рисунок 3.1 – Підключення Vue.js

Vite – це сучасний інструмент для розробки веб-додатків, який дозволяє створювати швидкі й ефективні проєкти. Відмінність Vite у швидкості та простоті налаштувань, що переконало мене вибрати його для розробки. Vite створює мінімалістичну структуру проєкту, готову до використання. Також vite використовує ES Modules у браузері, що значно пришвидшує час перезавантаження під час розробки.

Після етапу створення проекта та його підключення до фреймворка Vue.js я ініціалізував Vue прямо в HTML (див. рис. 3.2).

3.1 Розробка авторизації та реєстрації користувача

Після створення та підключення бази даних до проекту, було розроблено функції для реєстрації та авторизаціїї користувача. На початку реалізовано два компонента RegisterVue та LoginVue. Після чого створено окремий компонент з роутингом, та організація маршрутизації в компоненті Router (див рисунок 3.3).

```
<div id="app">{{ message }}</div>

<script>
  const { createApp } = Vue;
  createApp({
    data() {
      return {
        message: 'Hello, Vue.js!'
      };
    },
  }).mount('#app');
</script>
```

Рисунок 3.2 – Ініціалізація Vue.js

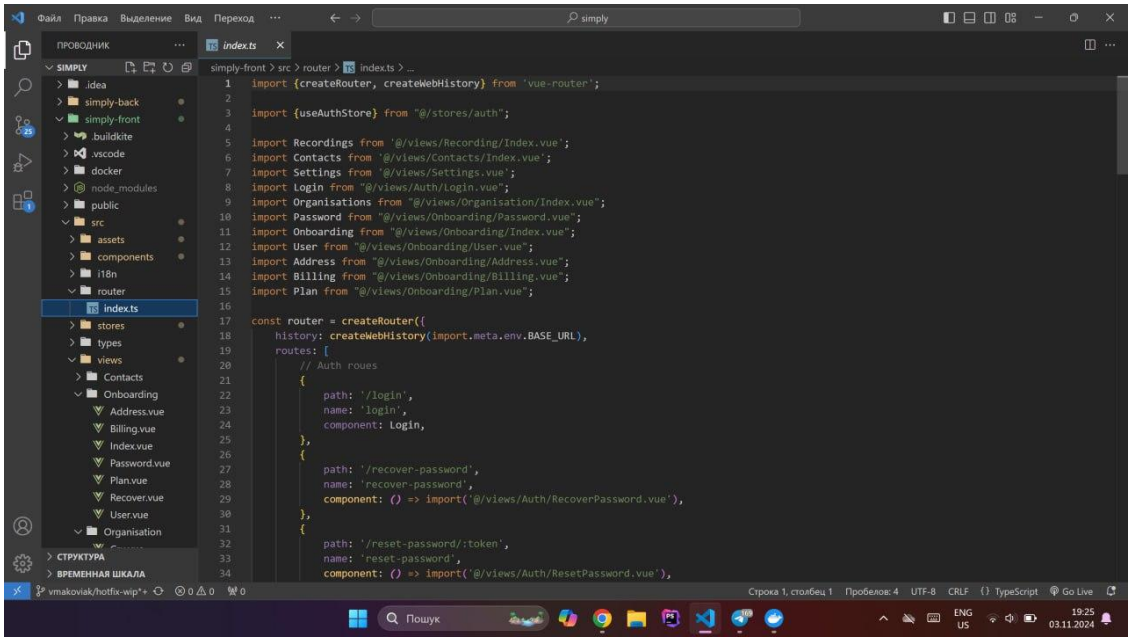


Рисунок 3.3 – Маршрутизація компонентів

Для захисту маршрутів використовувалась Meta, а для збереження стану авторизації користувача використовувався Vuex. Після чого, було інстальованно бібліотеку Axios, яка комунікувала з бекендом та відправляла запити на сервер (див рисунок 3.4).

```
import axios from 'axios';

const api = axios.create({
  baseURL: 'https://your-api.com',
  headers: {
    'Content-Type': 'application/json',
  },
});
```

Рисунок 3.4 – Запити через бібліотеку Axios

Для розробки авторизації та реєстрації, використувався звичайний HTML та фреймворк до CSS – Tailwind. Обрана технологія була саме тайлвінд, тому, що ця технологія дозволяє значно зменшити навантаженність всього коду проекту завдяки своєю короткою формою написання. Не потрібно вигадувати кастомні класи чи писати складні CSS-правила.

Також обов’язковою частиною реєстрації та авторизації була саме валідація для користувачів для забезпечення безпеки, коректності введених даних і захисту системи від помилок або зловмисних дій.

Ось як виглядала сторінка для логіну до використання фреймворка Tailwind, та після його підключення та удосконалення форми (див. рис. 3.5 та 3.6).

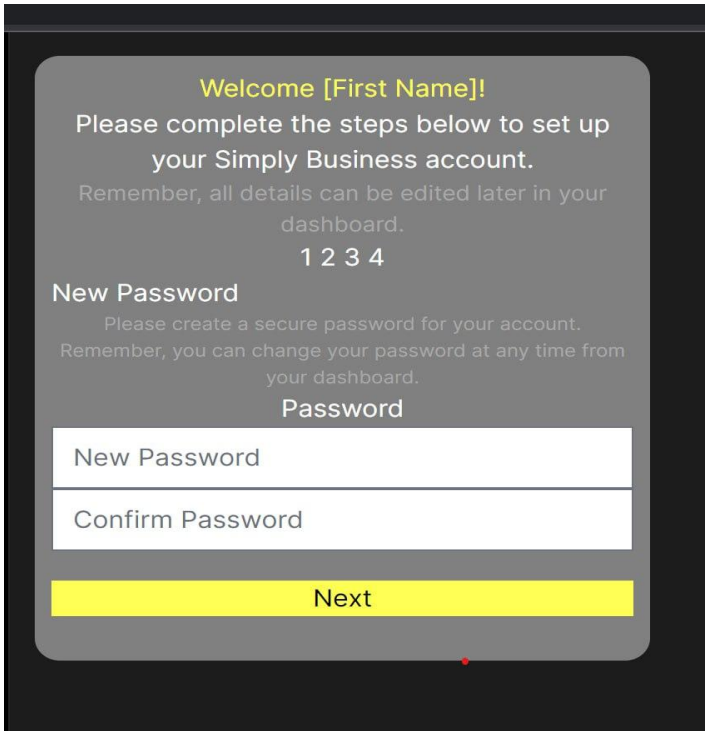


Рисунок 3.5 – Сторінка реєстрації

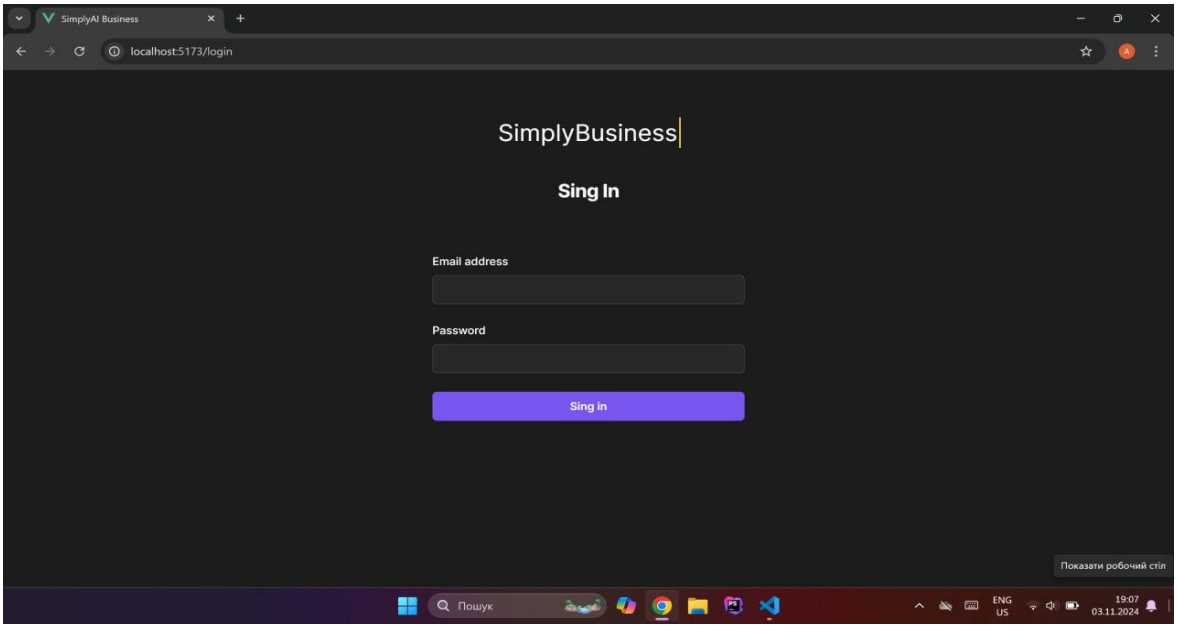


Рисунок 3.6 – Удосконаленна форма реєстрації

Також була створена функція для скидання пароля у випадку його втраті. Реалізація цієї функції на фронтенді є критичною, оскільки вона забезпечує користувачам доступ до їхніх облікових записів у разі втрати пароля та покращує загальний користувацький досвід.

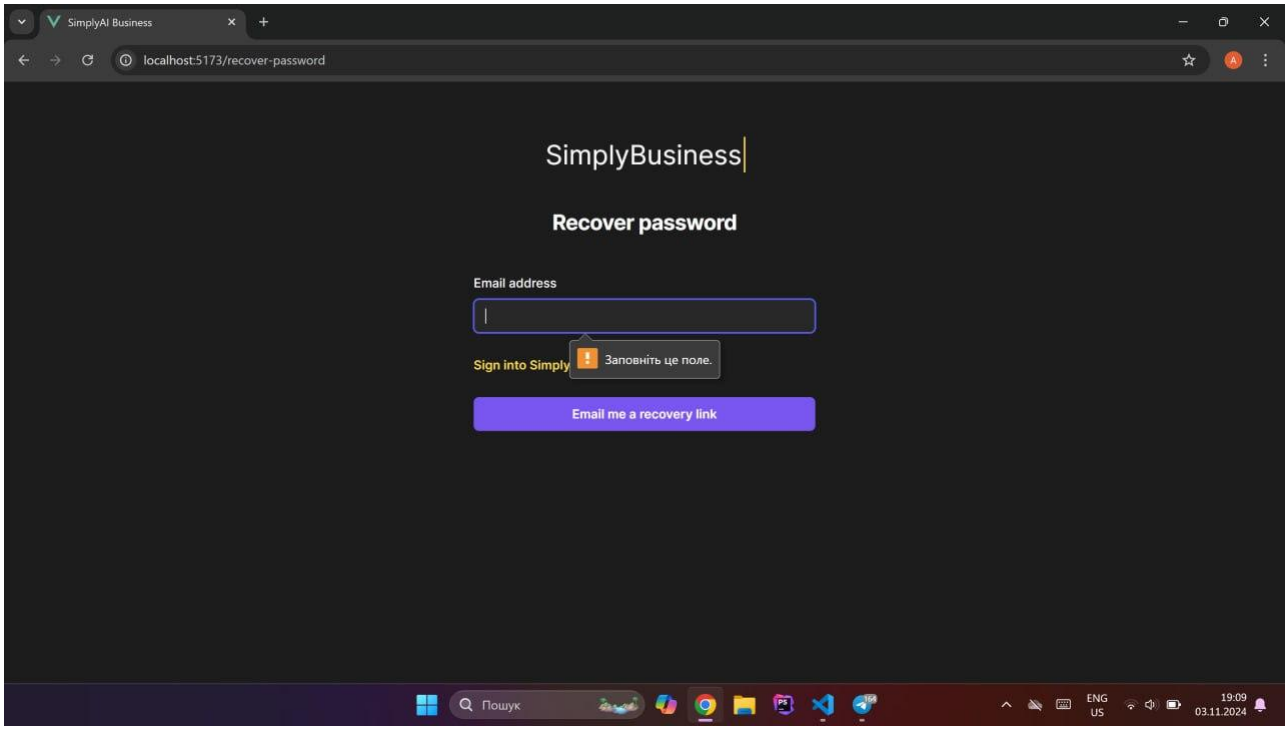


Рисунок 3.7 – Форма для відновлення пароля

Реалізація була дуже простою, на початку була написана форма для запиту скидання пароля, де вона приймає Email користувача, та відправляє запит на сервер. Та створена нова форма для встановлення нового пароля. Вона приймає новий пароль і токен, і зберігає ці дані у базі даних користувачів у зашифрованому вигляді. Ніхто крім користувача не буде знати його паролю чи любої іншої інформації (див. рис. 3.7).

3.2 Розробка інтерфейсу та основного функціоналу

Інтерфейс застосунку складався із трьох основних частин. Перша частина це бокове меню по лівій стороні, вона складалась із 3 основних частин. На верхній кнопці користувач міг переглянути важливу інформацію щодо застосунку, на другій міг переглянути контакти та службу підтримки.

В самому низу меню знаходиться збережена інформація користувача, а саме скільки часу користувач провів у застосунку, та скільки у нього залишилось вільного часу. Та остання функція у меню це – налаштування (див. рис. 3.8).

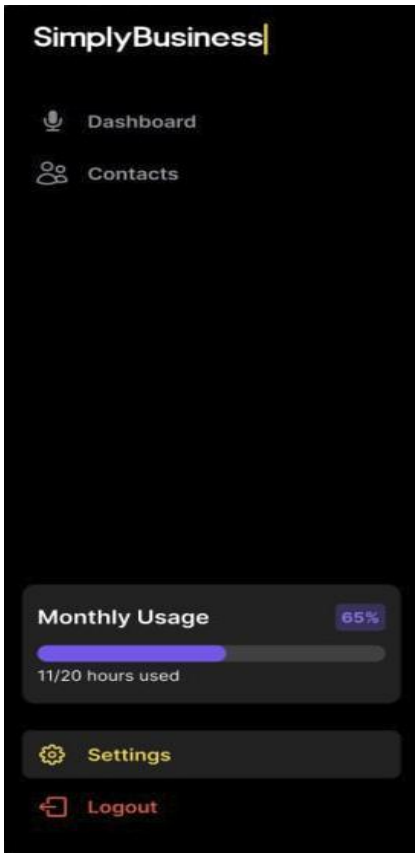


Рисунок 3.8 – Бокове меню застосунку

Другий екран застосунку складається із двох частин. Це навігація та основна панель на якій знаходиться користувач. Навігація також складалась з чотирьох частин, де користувач міг переключатись без оновлення основної сторінки, завдяки функції для фонового оновлення даних (див. рис. 3.9).

У вкладці Account користувач міг переглянути свої дані які були вже записані при реєстрації, та основну інформацію про свій профіль. Також користувач міг видалити або змінити свої дані, наприклад, змінити ім'я або поштову адресу (див. рис. 3.10).

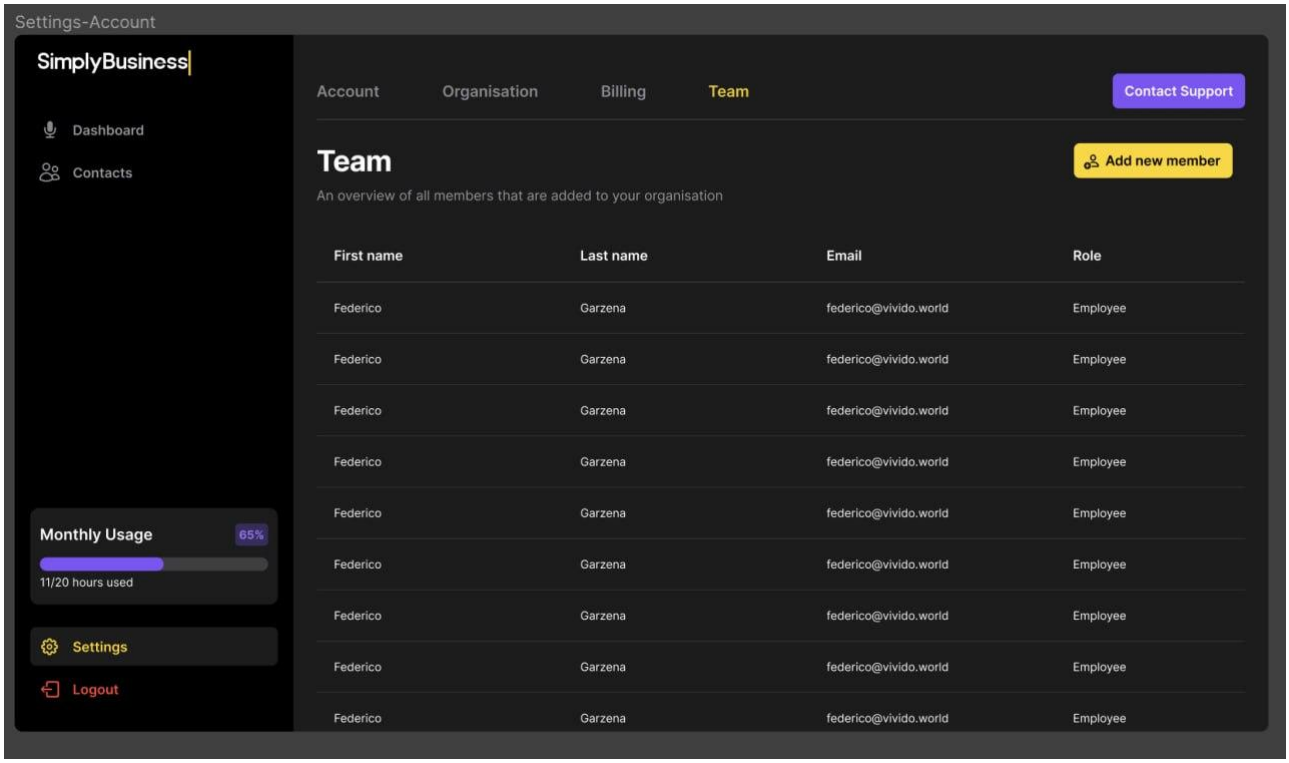


Рисунок 3.9 – Основна панель застосунку

```
const form = ref( value: {
  first_name: '',
  last_name: '',
  address: '',
  postal: '',
  city: '',
  country: '',
  email: '',
})

const rules = {
  first_name: {required},
  last_name: {required},
  address: {required},
  postal: {required},
  city: {required},
  country: {required},
  email: {required, email},
```

Рисунок 3.10 – Компонент збереження даних

Було розроблено окремий стан через Vue templates (hooks), де утримувалась вся інформації зареєстрованого користувача. Для створення функції збереження об’єкта використовувався темплейт useRef.

За допомогою бібліотеки Axios, оновлені дані змінювались та відправлялись на сервер, де в подальшому зберігались в базі даних, та відправляли нову інформацію у застосунок (див. рис. 3.11).

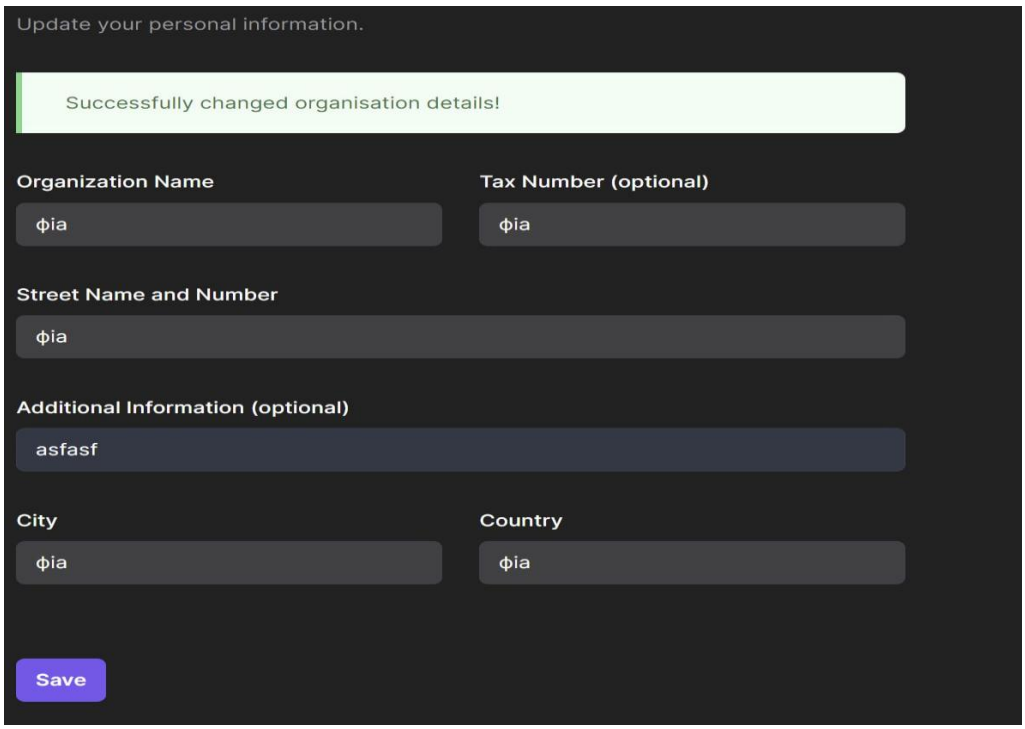


Рисунок 3.11 – Оновлення даних користувача

У вкладці Team був список членів організації створеною головою організації (див. рис. 3.12).

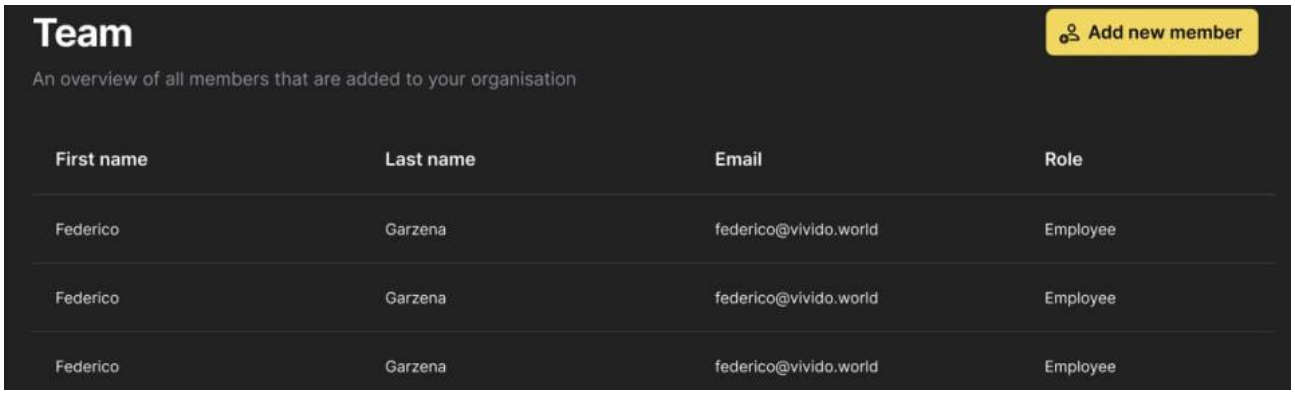


Рисунок 3.12 – Список учасників групи

Кожен із членів групи мав свою Роль, яка оновлювалась тільки лідером цієї організації. Самі спільники групи могли тільки змінювати своє імя, прізвище, та пошту.

Користувач, створивший нову організацію міг додавати до групи нових людей, та визначати їхню роль. Назви ролей лідер групи вибирав сам, та керував ними (див. рис. 3.13).

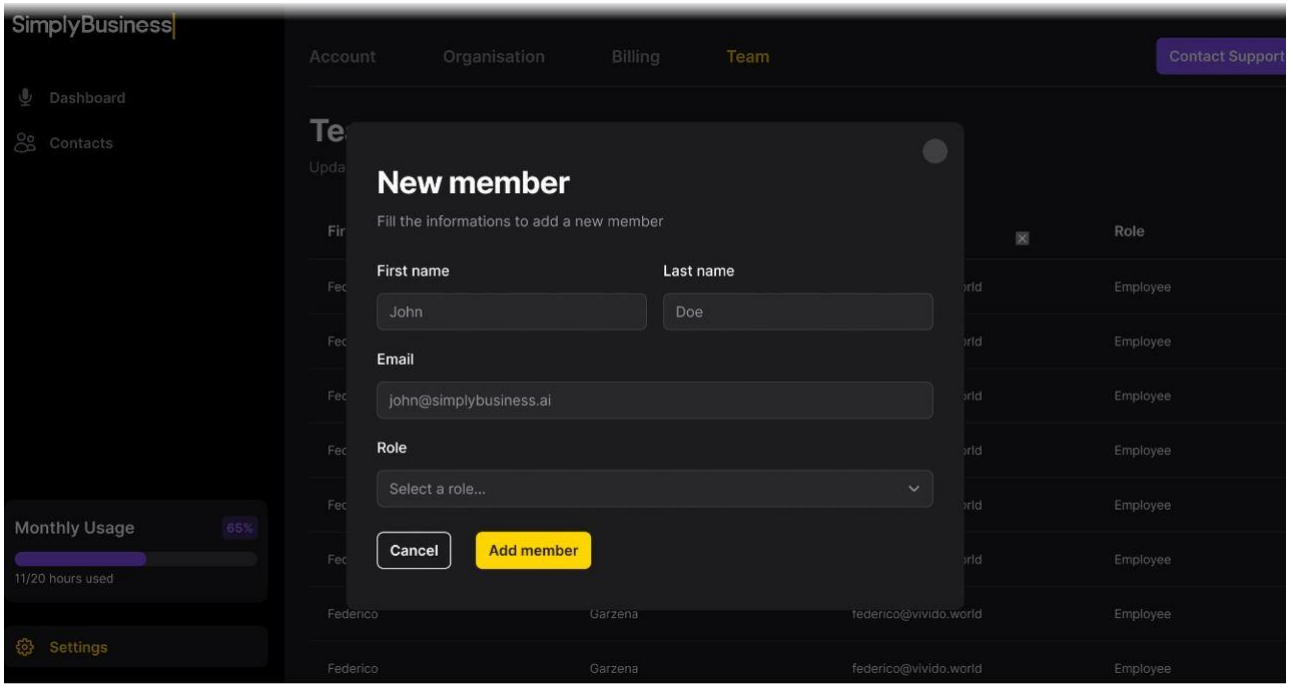


Рисунок 3.13 – Додавання учасників організації

Створення цих спільнот може бути корисним для різних груп та організацій, зокрема для медіакомпаній, які потребують транскрибування інтерв'ю чи записів, освітніх установ, де викладачі та студенти можуть перетворювати лекції та семінари на текст для зручного опрацювання. Юридичних фірм, які працюють із записами судових засідань чи інтерв'ю клієнтів.

Також це може використовуватись у дослідницьких команд, які аналізують аудіоматеріали або фокус-групи. Стартапів та компаній, які займаються створенням контенту чи обробкою даних. Організацій, що працюють у сфері соціальних досліджень чи журналістики. Він також може стати в пригоді для творчих колективів, які записують ідеї, нотатки чи дискусії, та навіть для державних установ або некомерційних організацій, які обробляють аудіо- чи відеоматеріали для звітності, архівування або аналітики.

Саме створення організації відбувалось у вкладці Organisations, де лідер групи задавав назву організації, адресу та іншу інформацію (див. рис. 3.14).

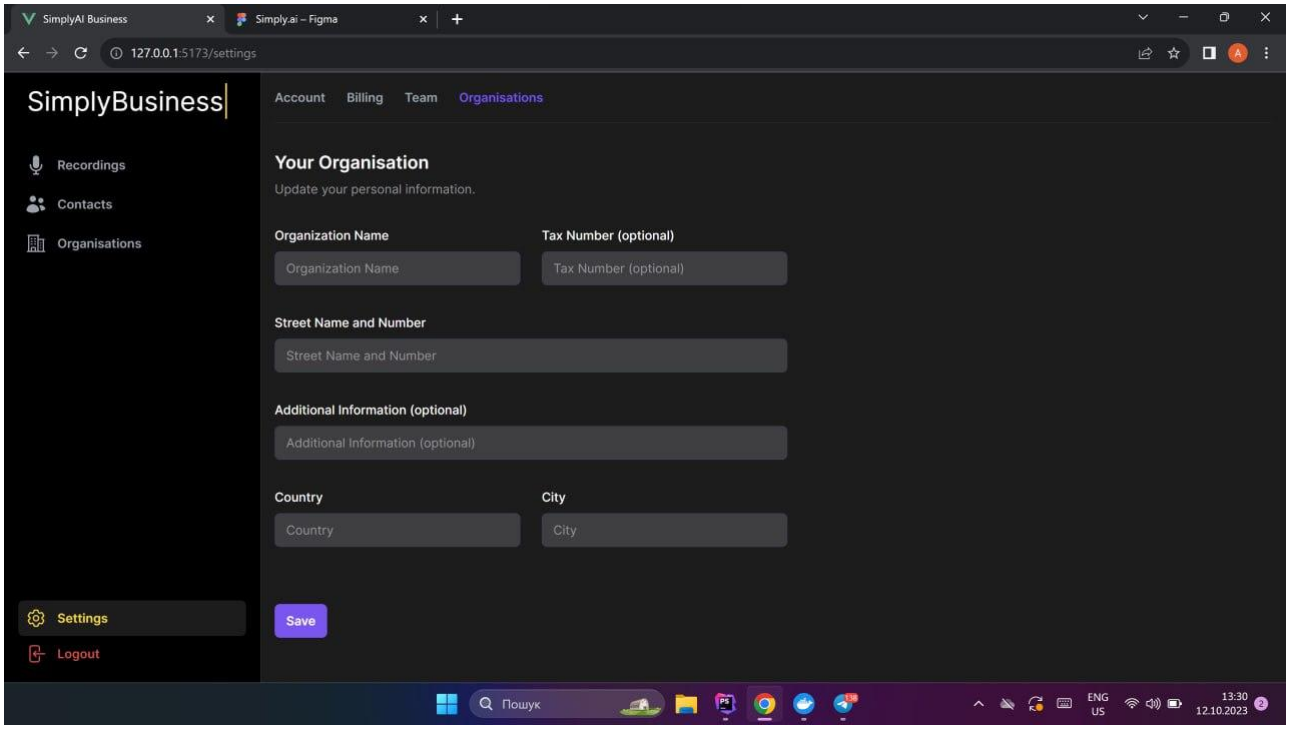


Рисунок 3.14 – Створення організації

У вкладці Billing, користувач міг переглянути свою інформацію своєї підписки, та перевіряти кількість оплачених днів (див. рис. 3.15).

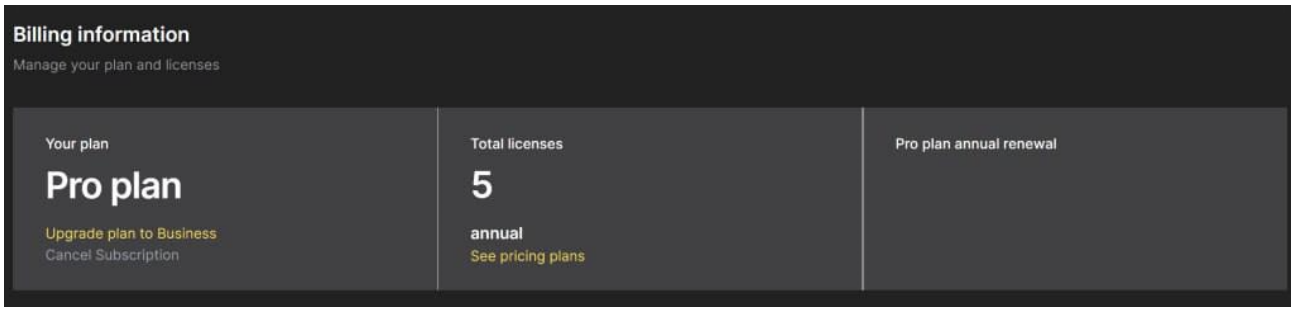


Рисунок 3.15 – Інформація щодо підписки користувача

Щоб користувач міг вільно оплачувати підписки у застосунку була реалізована платіжна система, та було створенно власний кабінет у платіжному сервісі. Платіжний сервіс був обраний PayPal.

Було створено власний обліковий запис та налаштував бізнес-профіль, після чого розроблено план підписки у якому було 2 варіанта для придбання підписки, місячна та річна (див. рис. 3.16).

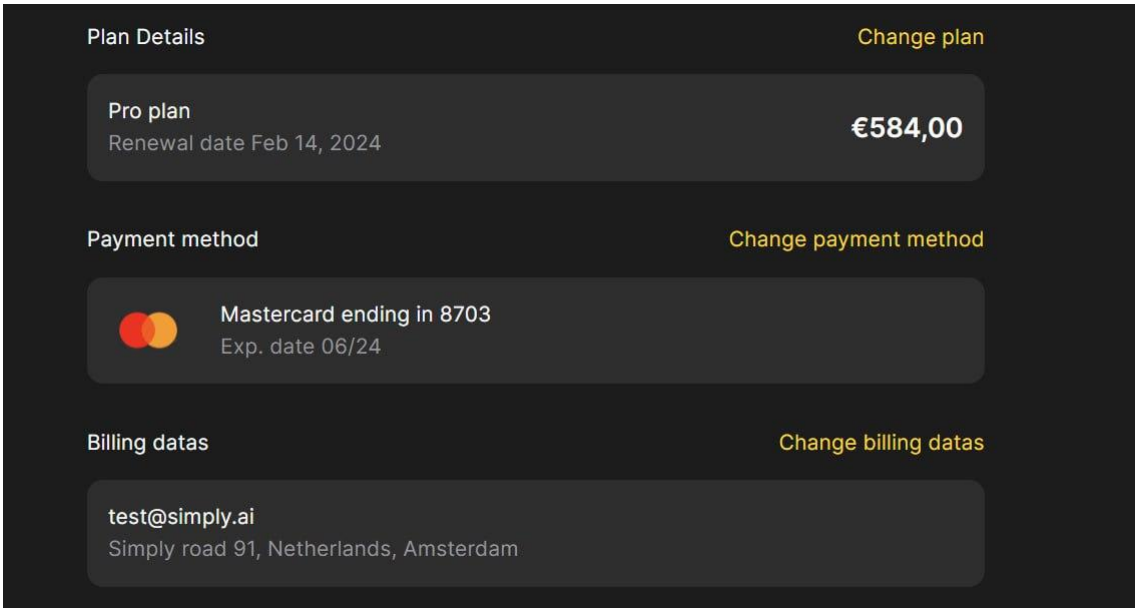


Рисунок 3.16 – План підписки

Наступним кроком розробки платіжної системи було додавання платіжної API до застосунку, встановлення SDK(Software Development Kit), для PayPal, та інсталяція його у терміналі проекту. (див. рис. 3.17).

```
npm install @paypal/react-paypal-js
```

Рисунок 3.17 – Встановлення SDK

Після успішної установки, була реалізована інтеграція API в Vue.js. На бекенді створена функця для обробки платіжних запитів, та додана кнопка для вибору оплати підписки і самої оплати.

Функція працювала наступним чином, користувач вибирав спосіб оплати, та на який термін буде зроблена оплата, після цього, натискав на кнопку «Сплатити», та запит відправлявся на бекенд, де, на сервері виконувалась функція обміну та повертала відповідь у вигляді успішної або не успішної оплати. Після успішної оплати, квитанція відправлялась користувачу на його

електрону пошту, яку він вказував при реєстрації облікового запису (див. рис. 3.18).

```
import { PayPalButtons } from '@paypal/react-paypal-js';

export default {
    components: { PayPalButtons },
    methods: {
        createSubscription(data, actions) {
            return actions.subscription.create({
                plan_id: 'P-XXXXXXXXXX', // ID плану з PayPal
            });
        },
        onApprove(data) {
            console.log('Subscription approved:', data);
        },
    },
    template: `
      <PayPalButtons
        createSubscription="createSubscription"
        onApprove="onApprove"
      />
    `,
};
```

Рисунок 3.18 – Інтеграція API в Vue.js

Після цих дій, у користувача у боковій панелі появлалась нова вкладка під назвою «Recordings». Саме тут користувач міг завантажувати свої медіа або аудіо файли для їх конвертації у текстовий формат та керувати ними (див. рис. 3.19).

Після завантаження файла у середовище, у користувача появлявся власний список всіх своїх файлів. Він міг переглядати їх, видаляти та редагувати.

Основними функціями були старт конвертації та перегляд аналізу. Властивостями файлів був їхній тип, дата завантаження, роль користувача та тривалість аналізу.

Рисунок 3.19 – Вкладка Recordings

Після успішного аналізу, відбувалась конвертація та в властивостях медіа файлу появлявся статус «Ready for review» та переглянути її можна було після натискання кнопки «View Analysis» (див. рис. 3.20).

Рисунок 3.20 – Конвертований медіа файл

Уся інформація яка знаходилась у середовищі збереження файлів, була доступна тільки користувачу, та була захищенна на сервері під спеціальною функцією для шифрування інформації.

3.3 Аналіз принципу дій конвертації медіафайлів

У сучасному світі де інформація швидко поширюється через різні медіа-формати, можливість конвертації медіа-файлів у текстовий формат стає надзвичайно корисною. Це сприяє кращій доступності інформації, дозволяючи людям з різними потребами, наприклад, з порушеннями слуху, отримувати рівний доступ до аудіо- чи відеоматеріалів.

Крім того, текстовий формат є більш зручним для зберігання, пошуку та аналізу великих обсягів інформації. Він дозволяє автоматизувати процес обробки даних за допомогою алгоритмів пошуку, аналітики та штучного інтелекту.

Це особливо важливо для наукових досліджень, навчання або журналістики, де часто потрібно швидко знайти ключову інформацію у великій кількості контенту. Ось основні сфери, де веб-застосунок максимально буде корисним:

1. Освіта. Автоматична трансформація записів лекцій у текстовий формат допомагає студентам створювати конспекти, особливо коли немає можливості записувати вручну.

2. Журналістика та медіа. Журналісти можуть використовувати ваш інструмент для автоматизації процесу транскрипції аудіо- та відеоінтерв’ю, що значно економить час. Розпізнавання тексту у відео та подкастах допомагає створювати архіви для подальшого пошуку та аналізу.

3. Судова система. Автоматичне створення протоколів із записів судових засідань зменшує навантаження на працівників канцелярій та покращує точність записів.

4. Здоров’я та медицина. Записи лікарів, зроблені у вигляді аудіо, можуть автоматично перетворюватися в текст для використання в електронних медичних картках. Інструмент дозволяє перетворювати аудіозаписи консультацій у текстовий вигляд, спрощуючи створення звітів.

5. Розробка програмного забезпечення. Інструмент може бути корисним для IT-команд для створення протоколів із записів онлайн-зустрічей, наприклад, у Zoom чи Google Meet.

Таблиця 3.2 – Користь конвертації медіа файлів.

Функція	Опис	Користь
Аудіо в текст	Конвертація аудіофайлів у текстовий формат.	Швидка транскрипція інтерв'ю, лекцій
Відео в текст	Перетворення діалогів із відео у текст.	Полегшує створення субтитрів або нотаток для презентацій
Мовна підтримка	Розпізнавання декількох мов і діалектів	Ідеально для інтернаціональних користувачів
Редагування тексту	Вбудовані інструменти для виправлення та форматування тексту	Зменшує потребу в додаткових редакторах
Експорт у різні формати	Збереження результатів у TXT, DOCX, PDF тощо	Легкість інтеграції у ваш робочий процес
Тайм-коди	Автоматична прив'язка тексту до часу у медіафайлі	Допомагає знайти потрібний момент у відео чи аудіо
Інтеграція з хмарними сховищами	Підтримка Google Drive, Dropbox, OneDrive	Доступ до файлів із будь-якого місця
Штучний інтелект для виправлень	Алгоритми вдосконалення тексту	Отримання тексту до використання
Захист даних	Шифрування файлів після обробки	Гарантія конфіденційності користувача

6. Особисте використання. Люди можуть використовувати систему для конвертації голосових нотаток у текст, полегшуючи особисту організацію. Транскрипція допомагає блогерам і подкастерам створювати текстові версії своїх матеріалів, що позитивно впливає на SEO. У таблиці 3.2 показано користь застосунку для конвертації файлів.

Розробка систем для конвертації аудіо та відео в текстовий формат є важливим напрямом в сучасній інформатиці. Створений прототип підтверджує доцільність використання новітніх алгоритмів розпізнавання мовлення для забезпечення високої точності результатів.

3.4 Висновок до третього розділу

В третьому розділі кваліфікаційної роботи було налаштовано інструменти та сучасні бібліотеки для розробки. Створено інтерфейс клієнтської частини програмного забезпечення. Розглянуто основні функції у клієнтській частині застосунку, основні вкладки для користування застосунком. Розроблено авторизацію та реєстрацію користувача. Реалізовано платіжну систему для оплати та продовження підписки у веб-застосунку. Висвітленно усі переваги даного застосунку та його принцип роботи.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Питання щодо охорони праці

ІТ-сфера, попри свою цифрову природу, має багато аспектів, які впливають на здоров'я працівників. Забезпечення охорони праці в цьому секторі включає низку заходів, спрямованих на створення безпечних, комфортних та ефективних умов роботи.

Нижче розглянуто основні аспекти охорони праці, зокрема освітлення, вентиляцію, ергономіку робочих місць, психоемоційне здоров'я та інші ключові фактори.

1. Освітлення робочого місця. Правильне освітлення є одним із найважливіших аспектів для працівників ІТ-сфери, оскільки вони тривалий час працюють за комп’ютерами. Неправильне освітлення може спричинити втому очей, головний біль і зниження продуктивності. Природне освітлення: робоче місце повинно мати доступ до природного освітлення. Для цього слід правильно розташувати робочі місця відносно вікон, уникаючи прямого сонячного світла на екрані монітора. Вікна мають бути оснащені жалюзі або шторами, щоб уникнути відблисків.

Штучне освітлення: рекомендовано використовувати LED-лампи з нейтральною температурою світла (3500–5000 K). Освітлення має бути рівномірним і не створювати різких контрастів між робочою поверхнею та монітором. Настільні лампи з регульованим рівнем освітлення дозволяють адаптувати світло під індивідуальні потреби.

2. Вентиляція і мікроклімат. Чисте повітря і комфортна температура – основа продуктивної роботи та гарного самопочуття. Вентиляція: робочі приміщення повинні бути обладнані сучасними системами вентиляції, які забезпечують регулярний обмін повітря. Рівень вуглекислого газу (CO_2) не повинен перевищувати норму (1000 ppm).

Температура і вологість: оптимальна температура для офісів: 22–25°C. Вологість повинна залишатися на рівні 40–60%. Встановлення кондиціонерів і зволожувачів допомагає підтримувати комфортний мікроклімат. Якість повітря:

уникнення накопичення пилу на техніці та меблях. Використання очищувачів повітря з HEPA-фільтрами.

3. Ергономіка робочого місця. Ергономіка є ключовим фактором у запобіганні професійним захворюванням, зокрема болю у спині, шиї та руках. Робоче крісло: має бути регульованим за висотою, з підтримкою поперекового відділу. Підлокітники повинні забезпечувати комфортну підтримку рук. Стіл: оптимальна висота – 70–80 см. На столі має бути достатньо місця для розташування монітора, клавіатури та інших пристроїв. Монітор: екран повинен розташовуватися на рівні очей, щоб уникнути надмірного нахилу голови. Відстань до монітора – 50–70 см. Периферійні пристрої: клавіатура і мишка мають бути зручними, з підставками для зап'ясть. Рекомендації для працівників: робити перерви кожні 45–60 хвилин, вставати, виконувати розминку для зняття напруги.

4. Психоемоційне здоров'я. Тривала розумова праця та високий рівень стресу можуть негативно впливати на працівників IT-сфери. Стрес-менеджмент: організація тренінгів з управління стресом. Наявність зони відпочинку в офісі. Баланс між роботою та відпочинком: гнучкий графік роботи або можливість працювати віддалено. Заохочення до використання відпусток. Психологічна підтримка: консультації з психологами або коучами.

5. Організація безпеки при роботі з технікою. Електробезпека: всі електроприлади повинні бути заземлені. Використання справних подовжувачів та розеток. Захист від перевантажень: використання стабілізаторів напруги та UPS для захисту обладнання. Протидія кіберзагрозам: регулярні тренінги з кібербезпеки для працівників. Установлення антивірусного програмного забезпечення.

6. Інші аспекти охорони праці. Організація харчування: наявність кухні або місця для прийому їжі. Доступ до чистої питної води. Профілактичні заходи: регулярний медичний огляд. Організація спортивних заходів. Евакуація і безпека: наявність чітких інструкцій з евакуації у разі надзвичайних ситуацій. Обладнання офісу вогнегасниками та аптечками. Виконання цих рекомендацій допоможе створити безпечне та комфортне робоче середовище для працівників

ІТ-сфери, що сприятиме підвищенню продуктивності та збереженню здоров'я співробітників.

4.2 Питання щодо безпеки в надзвичайних ситуаціях

Підготовка і перепідготовка керівного складу ЦЗ, її органів управління та сил, навчання населення діям у надзвичайних ситуаціях. Підготовка до надзвичайних ситуацій (НС) є ключовим елементом забезпечення безпеки населення та сталого функціонування критичних об’єктів у разі виникнення катастроф, аварій або стихійних лих. Ця діяльність включає підготовку керівного складу цивільного захисту (ЦЗ), органів управління та сил ЦЗ, а також навчання населення, щоб гарантувати їхню готовність до дій у кризових умовах.

Для ефективного реагування на НС необхідна професійна підготовка керівників, які несуть відповідальність за координацію дій та прийняття рішень. Основні напрями підготовки: оволодіння знаннями з управління силами та засобами ЦЗ у різних видах надзвичайних ситуацій; планування заходів захисту населення, територій та інфраструктури; вивчення алгоритмів прийняття оперативних рішень у кризових ситуаціях.

Перепідготовка та підвищення кваліфікації: регулярні тренінги, семінари та спеціальні курси, спрямовані на підвищення професійної компетенції керівників; використання сучасних методів навчання, таких як моделювання НС, віртуальні симуляції та польові навчання. Навчання в спеціалізованих закладах: керівники проходять підготовку у вищих навчальних закладах, які мають програми з цивільного захисту.

Додаткові програми можуть включати міжнародний досвід у сфері управління кризами.

Сили ЦЗ включають аварійно-рятувальні служби, спеціалізовані підрозділи та добровільні формування. Їхнє навчання має на меті забезпечити готовність до виконання завдань будь-якої складності. Тренування сил ЦЗ: проведення польових навчань і тренувань з ліквідації наслідків аварій; вивчення та вдосконалення навичок використання спеціалізованої техніки й обладнання; відпрацювання алгоритмів взаємодії з іншими підрозділами. Спеціалізація

персоналу: освоєння методів пошуку і порятунку постраждалих, гасіння пожеж, ліквідації хімічних і радіоактивних забруднень; навчання проведенню евакуаційних заходів.

Забезпечення безпеки населення значною мірою залежить від їхньої здатності самостійно діяти у надзвичайних ситуаціях. Підвищення обізнаності: проведення масово-роз'яснювальної роботи через засоби масової інформації, соціальні мережі та спеціальні видання; розробка та поширення буклетів, інструкцій і відеоматеріалів з використання ЗІЗ.

Практичне навчання: організація тренінгів і навчальних заходів у школах, університетах, підприємствах та громадських установах; проведення практичних занять з використанням протигазів, респіраторів, захисного одягу та інших ЗІЗ; навчання правилам користування медичними аптечками та першої допомоги. Сімейна готовність: заохочення громадян до розробки сімейних планів дій у НС; розміщення домашніх комплектів ЗІЗ та інших необхідних засобів.

Навчання людей алгоритмам дій у НС сприяє зниженню кількості постраждалих і швидшій ліквідації наслідків кризових ситуацій. Алгоритми дій: ознайомлення з сигналами тривоги та системами оповіщення; відпрацювання планів евакуації з громадських будівель та житлових зон; дії у випадках пожежі, хімічної аварії, повені, землетрусу чи техногенної катастрофи. Навчання дітей: заняття у школах, присвячені правилам поведінки у НС; інтерактивні уроки та ігри, що навчають дітей правилам безпеки.

Підготовка та перепідготовка керівного складу, органів управління та сил цивільного захисту, а також навчання населення діям у НС є важливим компонентом загальної системи забезпечення безпеки. Готовність до надзвичайних ситуацій дозволяє мінімізувати втрати та зберегти життя людей. Сучасні методи навчання, інтеграція міжнародного досвіду та використання

інноваційних підходів є ключовими чинниками підвищення ефективності системи цивільного захисту.

4.3 Висновок до четвертого розділу

В четвертому розділі кваліфікаційної роботи було розглянуто питання щодо охорони праці, та безпеки в надзвичайних ситуаціях. Було розглянуто питання забезпечення безпеки в різних сферах діяльності, зокрема в ІТ-сфері та цивільному захисті. Обговорено питання підготовки та перепідготовки керівного складу, органів управління, а також навчання населення діям у надзвичайних ситуаціях. Розглянуто питання щодо систематичного підходу до безпеки, щоб гарантувати не лише збереження здоров’я працівників і громадян, а й ефективне функціонування суспільства загалом.

ВИСНОВКИ

В першому розділі кваліфікаційної роботи освітнього рівня «Магістр»: досліджено існуючі середовища для написання коду. Обгрунтовано вибір середовища для реалізації веб-застосунку, порівняння редакторів коду з їх властивостями у таблиці. Розглянуто найпопулярніші редактори коду для розробки. Висвітлено основні інструменти редакторів коду для покращення роботи із написанням, та його модифікації. Перевірено адаптивність редакторів та їх розширення у взаємодії із мовами програмування.

В другому розділі кваліфікаційної роботи обґрунтовано вибір технологій та середовище для реалізації бази даних. Проаналізовано вибір технологій по їх властивостям та інструментам. Було розроблено та реалізовано серверну частину. Створено допоміжну функцію для збереження та захисту інформації. Було обгрунтовано вибір технології для підтримки та взаємодії клієнтської частини застосунку із бекендом. Реалізовано функції для обміну даними. Створено запит для відправлення та отримання відповіді від сервера до фронтенду.

В третьому розділі кваліфікаційної роботи було інстальовано інструменти та сучасні бібліотеки для розробки. Створено інтерфейс клієнтської частини програмного забезпечення. Розглянуто основні функції у клієнтській частині застосунку та їх робота при використанні. Висвітлено головні вкладки для користування застосунком. Розроблено авторизацію та реєстрацію користувача. Реалізовано платіжну систему для оплати та продовження підписки у веб-застосунку. Висвітленно усі переваги даного застосунку та його принцип роботи.

В четвертому розділі кваліфікаційної роботи було розглянуто питання щодо охорони праці, та безпеки в надзвичайних ситуаціях. Було розглянуто питання забезпечення безпеки в різних сферах діяльності, зокрема в ІТ-сфері та цивільному захисті.

Обговорено питання підготовки та перепідготовки керівного складу, органів управління, а також навчання населення діям у надзвичайних ситуаціях. Розглянуто питання щодо систематичного підходу до безпеки, щоб гарантувати

не лише збереження здоров’я працівників і громадян, а й ефективне функціонування суспільства загалом.

ПЕРЕЛІК ДЖЕРЕЛ

1. Bodnarchuk, I., Duda, O., Kharchenko, A., Kunanets, N., Matsiuk, O., & Pasichnyk, V. (2020). Choice Method of Analytical Platform for Smart City (No. 4374). EasyChair.

2. Kharchenko, A., Halay, I., Zagorodna, N., & Bodnarchuk, I. (2015, September). Trade-off optimal decision of the problem of software system architecture choice. In 2015 Xth International Scientific and Technical Conference" Computer Sciences and Information Technologies"(CSIT) (pp. 198-205). IEEE.

3. Kharchenko, A., Bodnarchuk, I., Raichev, I., & Morar, Y. (2014). The Method of Software Architecture Design Accounting the Quality Requirements Change.

4. Microsoft. Visual Studio Code for the Web. Visual Studio Code - Code Editing. Redefined. URL: https://code.visualstudio.com/docs/editor/vscode-web (date of access: 19.11.2024).

5. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2011). Database system concepts.

6. Створення бази даних та їх взаємодія із фронтендом: "Learning SQL" (A. M. Beaulieu) — підручник з SQL, який охоплює базові та розширені теми, "Fullstack Vue" (H. R. Burrill, L. R. Max) — книга для створення сучасних веб-додатків, що використовують Vue.js та бази даних, "Building Web Applications with Go" (J. R. Jenkins) — для розробки API та інтеграції з базами даних.

7. W3Schools.com. W3Schools Online Web Tutorials. URL: https://www.w3schools.com/sql/ (date of access: 19.11.2024).

8. Macrae, C. (2018). Vue. js: up and running: building accessible and performant web apps. " O'Reilly Media, Inc.".

9. Mooney, C. Z., Duval, R. D., & Duvall, R. (1993). Bootstrapping: A nonparametric approach to statistical inference (No. 95). sage.

10. Richardson, L., Amundsen, M., & Ruby, S. (2013). RESTful Web APIs: Services for a Changing World. " O'Reilly Media, Inc."

11. Vite - The Next Generation, Frontend Tooling. Getting Started. vitejs. URL: https://vite.dev/guide/ (date of access: 19.11.2024).

12. Window: localStorage property - Web APIs | MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage (date of access: 19.11.2024).

13. Bertino, E., & Sandhu, R. (2005). Database security-concepts, approaches, and challenges. IEEE Transactions on Dependable and secure computing, 2(1), 2-19.

ДОДАТКИ

Додаток А

Тези доповіді на конференції

Додаток Б

Програмний код головної сторінки

```
      <template>
          <table class="min-w-full divide-y divide-gray-400">
              <thead>
              <tr>
                  <th scope="col" v-for="header in headers"
                      class="px-3 py-3.5 text-left text-sm font-semibold text-
white">{{ header.title }}</th>
              </tr>
              </thead>
              <tbody class="divide-y divide-gray-600">
              <tr v-for="row in data" :key="row.id">
                  <td class="whitespace-nowrap px-3 py-2 text-sm text-gray-300"
:key="item.key" v-for="item in headers">
                      <slot
                          :name="`cell(${item.key})`"
                          :value="row[item.key]"
                          :item="item"
                          :row="row">
                          {{ row[item.key] }}
                      </slot>
                  </td>
              </tr>
              </tbody>
          </table>
      </template>

      <script setup lang="ts">
      export interface TableHeader {
          title: string,
          key: string,
      }

      const props = defineProps<{
          headers: TableHeader[],
          data: any[],
      }>();
      </script>
      <template>
          <SModal @closeModal="closeModal" :open="showModal">
              <div class="flex flex-col justify-center py-6 px-12 gap-y-4">
                  <div class="text-center">
                      <h3 class="text-base font-semibold text-white leading-6 mb-
1">{{ t('Delete recordings') }}</h3>
                      <p class="text-light text-center mx-auto" v-html="title" />
                  </div>

                  <SButton type="submit" color="danger" @click="deleteRecordings"
class="mx-auto">{{ t('Delete recordings') }}</SButton>
                  <button   class="text-sm   text-white"   @click="closeModal">{{
t('Cancel') }}</button>
              </div>
          </SModal>
      </template>
      <SModal @closeModal="closeModal" :open="showModal">
              <form @submit.prevent="submit">
                  <div class='flex text-neutral-50 text-3xl p-6 px-6 flex-col'>
                      <div class="sm:col-span-6">
                          <h3   class="font-semibold   text-white">{{   t('Change
details') }}</h3>
```

```
                        <p class="text-sm text-light my-2">{{ t('Change the
information of this user') }}</p>
                    </div>

                    <div class="grid grid-cols-1 gap-x-6 gap-y-6 sm:max-w-xl
sm:grid-cols-6 mt-5">
                        <div class="col-span-full" v-if="errors">
                            <ul class="px-4 py-2 bg-danger text-sm rounded-md">
                                <li v-bind:key="error" v-for="error in errors">
                                    {{ error[0] }}
                                </li>
                            </ul>
                        </div>

                        <div class="sm:col-span-3">
                            <SInput    name="name"    v-model="form.first_name"
:label="t('First                Name')"                :placeholder="t('John')"
:errors="v$.first_name.$errors.length" />
                        </div>

                        <div class="sm:col-span-3">
                            <SInput     name="tax"     v-model="form.last_name"
:label="t('Last                 Name')"                 :placeholder="t('Doe')"
:errors="v$.last_name.$errors.length" />
                        </div>

                        <div class="col-span-full">
                            <SSelect      name="role"      v-model="form.role"
:options="roles"       :label="t('Role')"       :errors="v$.role.$errors.length"
:allowNone="false" />
                        </div>
                    </div>

                    <div class='flex flex-col w-full'>
                        <div class='mt-2 flex'>
                            <div class="my-4">
                                <SButton   color="transparent"   class="border
border-white">{{ t('Cancel') }}</SButton>
                            </div>
                            <div class="my-4 m-5">
                                <SButton   color="secondary">{{   t('Confirm')
}}</SButton>
                            </div>
                        </div>
                    </div>
                </div>
            </form>
        </SModal>
    <div class="break-inside-avoid-column mb-4">
            <div class="relative flex items-center space-x-3 bg-dark rounded-lg
px-6 py-5">
    <!--            <div class="flex-shrink-0 p-4 bg-primary rounded-md">-->
    <!--                    <Component class="w-6 h-6" :is="dataPoint.icon ??
CalendarIcon"/>-->
    <!--            </div>-->
                <div class="min-w-0 flex-1">
                    <div>
                        <p class="text-sm text-white/40 capitalize mt-1 pb-4
inline-block mt-2.5 max-w-[60%]">{{ formattedName }}</p>

                        <button class="px-3 py-2 rounded-lg bg-white/10 inline-
block float-right ml-2.5" v-if="type === 'datapoint'" :disabled="isLoading"
@click="refreshDataPoint">
                            <font-awesome-icon    v-if="isLoading"    icon="fa-
regular fa-arrow-rotate-right" spin="spin"></font-awesome-icon>
```

```
                                <font-awesome-icon   v-if="!isLoading"   icon="fa-
regular fa-arrow-rotate-right"></font-awesome-icon>
                            </button>

                            <STipTap                                ref="summaryEditor"
@setIsSaving="setIsSaving"     :isSaving="isSaving"     :actionPanelInline="true"
:showSearch="false" @saveContent="saveContent" v-model="dataPoint.value" />
                        </div>
                    </div>
                </div>
            </div>
        <template #thead="{ sorting, sort }">
                    <TableHead v-if="multiselect" class="px-0" />

                    <TableHead   v-for="header   in   headers"   :key="header.key"
:sortable="sortable ? header.key : ''" multiple :sort="sort" @sorting="sorting">
                        {{ header.title }}
                    </TableHead>
                    <table-head v-if="editable"></table-head>
                </template>

                <template #tbody="{row}">
                    <TableBody          v-if="multiselect"          ref="tableRow"
:class="['multiselect px-0', isRowSelected(row) ? 'opacity-100' : 'opacity-0']">
                        <slot      name="cell(multiselect)"      :row="row"      v-
if="!row?.denyMultiselect">
                            <button :class="['rounded-full border-2 border-white/20
w-5  h-5  align-middle  text-white  flex  items-center  justify-center  ml-1',
isRowSelected(row)     ?     'bg-primary     selected'     :     'bg-white/10']"
@click.stop="onSelectRow(row)">
                                <font-awesome-icon       v-if="isRowSelected(row)"
icon="fa-solid fa-check" class="h-2.5"></font-awesome-icon>
                            </button>
                        </slot>
                    </TableBody>

                    <TableBody    :key="item.key"    v-for="item    in    headers"
ref="tableRow">
                        <slot
                                :name="`cell(${item.key})`"
                                :value="row[item.key]"
                                :item="item"
                                :row="row">
                            {{ row[item.key] }}
                        </slot>
                    </TableBody>
```