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 систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Соціальна мережа на базі фреймворку React та платформи Node js.

Виконав: студент IV курсу, групи СІс-43
спеціальності 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

_____ Лещишин Р.А,
(підпис) (прізвище та ініціали)

Керівник _____ Шингера Н. Я.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Луцик Н. С.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Осухівська Г. М.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2022

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'ютерної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ _____
(підпис) (прізвище та ініціали)
«__» _____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)
за спеціальністю 123 Комп'ютерна інженерія
(шифр і назва спеціальності)
Студенту Лещишину Роману Андрійовичу
(прізвище, ім'я, по батькові)
1. Тема роботи Соціальна мережа на базі фреймворку React та платформи Node js.

Керівник роботи Шингера Наталя Ярославівна, к.т.н., доцент кафедри КС
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом ректора від «_» .__ 20 року .__ № .__.__.__
2. Термін подання студентом завершеної роботи 22 червня 2022р.
3. Вихідні дані до роботи Розробка клієнт серверного застосунку на базі Node.js
4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз технічного завдання. 2 Проектна частина 3 Практична частина
4. Безпека життєдіяльності, основи охорони праці Висновки

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

1 Блок схема авторизації на сайті
2 Блок схема серверного застосунку
3 Список сторінок веб-застосунку
4 Структурна схема веб-застосунку
5 Структура Бази даних

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці	Лазарюк В. В.		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи		*Виконано*
2.	Провести огляд літературних джерел по темі випускної бакалаврської роботи		*Виконано*
3.	Провести порівняльну характеристику програмних продуктів		*Виконано*
4.	Розробити функціональну схему роботи об'єкта проектування. Розробка моделі бази даних		*Виконано*
5.	Практична реалізація об'єкта проектування		*Виконано*
6.	Описати та розробити організаційні заходи спрямовані на поліпшення стану охорони праці		*Виконано*
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»		*Виконано*
8.	Виконання завдання до підрозділу «Основи хорони праці»		*Виконано*
9.	Оформлення кваліфікаційної роботи		*Виконано*
10.	Нормоконтроль		*Виконано*
11.	Перевірка на плагіат		*Виконано*
12.	Попередній захист кваліфікаційної роботи		*Виконано*
13.	Захист кваліфікаційної роботи		

Студент _____ *Лещишин Р. А.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Шингера Н. Я.*

АНОТАЦІЯ

Соціальна мережа на базі фреймворку React та платформи Node js.// Кваліфікаційна робота освітнього рівня «Бакалавр» // Лещишин Роман Андрійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних систем та мереж, група СІс-43 // Тернопіль, 2022 //С.-69, рис.-19, додат. –4, бібліогр. – 22.

Ключові слова: фреймворк, бібліотека, модуль, компонент, соціальна мережа, сервер, клієнт, база даних.

В кваліфікаційній роботі розглянутий процес розробки веб застосунка соціальна мережа на базі фреймоврка React та платформи Node.js. Робота складається з чотирьох розділів, висновків, списку використаних джерел, додатків. У вступі описана актуальність та мета роботи та сформований план і подальшої роботи.

У першому розділі описані принципи роботи соціальних мереж, їх типи та технічні аспекти. Також був здійснений огляд предметної області.

У другому розділі була описана проектна частина, де було детально описані всі використані технологій, також було аргументований даний вибір.

У третьому розділі було описано практичну частину, де відбувалася безпосередня розробка програмного забезпечення та подальше його тестування. В цьому розділі було запущено та налаштовано три сервери та розроблено серверну і клієнтську частини.

У четвертому розділі було розглянуті питання психологічних чинників небезпеки та впливу кольору на покращення умов праці.

ANNOTATION

Social network based on the React framework and the Node js platform.// Qualifying work of the Bachelor's level of education // Leshchyshyn Roman Andriyovych // Ivan Pulyuy Ternopil National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science engineering, CIC-43 group // Ternopil, 2022 //pages.-69, fig.-19, appendix. -4, bibliography. -22.

Keywords: framework, library, module, component, social network, server, client, database.

The qualification work considers the process of developing a social network web application based on the React framework and the Node.js platform. The work consists of four sections, conclusions,

list of used sources, applications. The introduction describes the relevance and purpose of the work and formed a plan for further work.

The first section describes the principles of social networks, their types and technical aspects. The subject area was also inspected.

The second section described the project part, where all the technologies used were described in detail, this choice was also justified.

The third section described the practical part, where the software was directly developed and further tested. In this section, three servers were started and configured, and the server and client parts were developed.

In the fourth section, the issues of psychological factors of danger and the influence of color on the improvement of working conditions were considered.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

HTML – HyperText Markup Language. Мова розмітки веб сторінок.

CSS – Cascading Style Sheets. Мова стилів, для задання візуального виду документові, написаному на HTML

MVC – Model-View-Controller шаблон розробки програмного забезпечення, який використовується для створення інтерфейсів користувача

Framework – сукупність програмних рішень, що полегшує розробку програмних застосунків.

NPM – Node Package Manager менеджер пакунків для платформи Node.js

SQL – Structured Query Language – мова для реляційних баз даних

Social network – соціальна мережа.

Gulp – таск-менеджер для автоматичного виконання завдань

Модуль – окремий файл, де зосереджений певний функціонал, який в подальшому може бути імпортований в інші файли

Статичні сайти – сайт що складається зі статичних сторінок

Динамічні сайти – сайт, вміст якого може динамічно змінюватися без перезавантаження сторінк

ВСТУП

Актуальність дослідження. З ростом якості інтернету і комунікаційних технологій спілкування людей стає все простішим і переходить на нові рівні сприйняття. До створення телеграфа люди уявляли віддалене спілкування тільки через листи. Терміни доставки листів отримувачеві могли сягати до декількох тижнів або навіть місяців**.** Після створення телеграфа і повного його впровадження в масову систему звязку швидкість комунікації швидко зросла, але мала свої недоліки. Після створення інтернету і введення його в маси світ спілкування і передачі інформації різко перевернувся. На сьогодні технології настільки пішли вперед, що люди тепер не обмежуються короткими текстовими сповіщеннями, а можуть відправляти величезні листи, документи, книги, відео, аудіо. З появленням соціальних спілкування набуло тенденції спілкування як при реальному діалозі, тобто моментальної або просто швидкої відповіді на питання співрозмовника і в цьому випадку повністю неважливо на які відстані перебувають люди, головне щоб був зв'язок інтернету.

Соціальні мережі породили ціле середовище спілкування і розширили можливості людей буквально у всіх сферах. Вони дозволили людям знайомитися і спілкуватися з неймовірною кількістю людей з будь яких країн світу. Завдяки соціальним мережам появилася можливість створювати групи по інтересам незалежно від географічного положення. Тепер будь яка людина здатна ділитися своїми досягненнями, спостереженнями, думками і отримувати реакцію від інших користувачів соціальних мереж.

З появленням соціальних мереж появилася велика кількість нових професій, наприклад: блогери, таргетологи і багато інших. Люди з великими амбіціями і харизмою змогли збирати навколо себе мільйони глядачів, не будучи телеканалом або великою організацією.

Від появи перших соціальних мереж до сьогоднішнього дня дана структура сильно розгалузилася і почала охоплювати різні людські сфери, перетворюючись з соціальної мережі в мессенджери, сайти знайомств, соціальні мережі по знаходженню роботи.

Однак на даний момент актуальні і мають велику популярність соціальні мережі в класичному їх представленні, такі як VK, Facebook, Twitter.

Мета та практичне значення роботи – розробка соціальної мережі. В даному проекті розглянуто технології розробки веб-застосунків та основні методології створення соціальних мереж.

РОЗДІЛ 1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Огляд предметної області задачі проектування

Сучасне життя в плані комунікаціїй радикально відрізняється від того життя, яке було років 20 назад. Сьогодні практично кожн алюдина просинаючись зранку перевіряє власний смартфон або компютера на наявність нових повідомлень в таких соціальних мережах як WhatsApp, Snapchat чи LinkedIn. Кожна з подібних платформ містить в собі мільйони користувачів і сотні тисяч які в онлайні на даний момент часу. Соціаьлні мережі давно витіснили такі речі як мобільні SMS або електронні листи. Соціальні мережі обєднали весь світ в одне інформаційне поле.

Основним визначення соціальної мережі це – соціальна структура або організація людей утворена певними компаніями. службами або індивідуумами. Основною цілю соціальної мережі є забезпечення людей можливістю спілкування і поширення будь якої інформації.

За класифікацією соціальні мережі поділяються на широкі та нішеві: Широкі соцмережі призначені для усіх користувачів з будь якими інтересами. Зазвичай в таких соціальних мережах йде спілкування на всі можливі теми. Зазвичай саме такі соцмережі являються найбільш популярними так як підлягають під вимоги звичайно користувача, який хоче дізнатися нові новини в різних сферах діяльності, глибоко не вникаючи в сам процес. Нішеві соціальні мережі створені для окремих груп людей обєднаних однією цілю або інтересами. Таких соцмереж є сотні тисяч для будь якої сфери, від рибалки до мистецтв чи науки.

Наступними по класифікації йдуть соціальні мережі за призначенням. Вони бувають наступних видів: інформаційні, освітні, для знайомств, мультимедійні, мережі для соціальних звязків, торгові мережі.

Освітні соціальні мережі створені для студентів, найпопулярнішою з таких мереж є ResearchGate та Pinterest. Багато освітніх соціальних мереж побудовано на базі платформ в навчальних закладах для організації спілкування між студентами і викладачами. Соціальні мережі знайомств створенні для єдиної цілі –це знайомство між людьми для того щоб налагодити особисте життя чи створити сімю. Найпопулярнішими в цій сфері являються Tinder і Badoo. Мультимедійні соціальні мережі базуються на поширенні контенту у вигляді відео, зображень та різного виду блогів. Мережі для соціальних звязків використовуються для організації безпосереднього звязку для спілкування між людьми. Торгові мережі створенні для організації середовища для реклами та торгівлі різноманітних продуктів.

Також соціальні мережі класифікують за платформою. Веб соціальні мережі. До них можна підключитися тільки за допомогою персонального компютера. На початку зародження соціальних мереж більшість з них були веб мережами. На сьогоднішній день з тенденцією кросплатформенності фактично всі перейшли і на мобільні платформи. Гібридні мережі. На сьогодні це найпопулярніший клас мереж. Дані мережі являють собою гібрид веб мережі і мобільної мережі. Тобто можуть бути запущені як на настільному компютері так і на смартфоні. Мобільні мережі. Призначені тільки для роботи на смартфонах. В питанні соціальних мереж важливим є багато складових, таких як безпека, продуктивність чи персоналізація.

В плані безпеки більшість соціальних мереж має величезну кількість налаштуванью, які допомагають користувачеві редагувати доступи для перегляду чи спілкування для різноманітних категорій спілкування.

Таким чином появляється можливість створити практично закритий профіль для звичайних користувачів, або ж зробити так що вся публічна інформація про користувача буде в легкій доступності для інших.

Однак навіть при таких налаштування існує прошарок протоколів і технологій. які дозволяють користувачам захищено спілкуватися між собою, ділитися важливою інформацією.

З ростом технологій росте необхідні придумувати все складніші методи безпеки. За весь час було розроблено величезну кількість технологій, протоколів, шифрів і методик, які дозволяють людям перебуватися у відносній інформаційній безпеці в інтернеті.

У плані продуктивності соціальні мережі повинні бути максимально оптимізовані. Соціальними мережами одночасно користується величезна кількість людей, щоб залишатися на ринку необхідно щоб все працювало коректно і досить швидко для всіх користувачів. При поганій оптимізації і неможливості масштабування застосунку появиться необхідність використовувати надлишкові технічні ресурси, які в аналогічному випадку повпливають на підтримку продукту і його надійність. При хорошій структуризації усіх процесів як на стороні застосунку так і на стороні сервера приведе до менших втрат по потужності і в майбутньому дозволяти розширювати як і функціонал так і сферу впливу.

Для будь якого продукту дизайн вважається одним з ключових факторів, який впливає на популярність Саме зручний інтерфейс приваблює нових користувачів. Від перших версій як сайтів так і соціальних мереж тенденції по дизайну часто мінялися. Були періоди коли більшість сторінок в інтернеті були нагромаджені великими елементами, кнопкам добавляли різноманітних 3д ефектів і багто інших речей. З розвитком такої сфери як UI/UX дизайн і структура програмних продуктів почали набувати лаконічних, стилізованих і мінімалістичних форм, які були приємні користувачеві і не навантажували його при пошуку інформації або роботи в застосунку.

1.2 Структура соціальних мереж

Будь яка соціальна мережа базуються на наступних елементах:

- Сторінка профілю.
- Сторінка для спілкування.
- Стрічка постів

Дані компоненти вважаються базою і мінімумом того що необхідно для соціальної мережі. Зазвичай дані сторінки обростають величезною кількістю підпунктів і інших елементів для зручності користувачів. Також з розвитком соцмереж появилася величезна кількість і інших компонентів на прикладі, сторінка для ігор, музики, календарі, історія, ринка, нотаток чи закладок.

Після реєстрації в мережі першим що користувач бачить це сторінка власного профілю. На різних застосунках даний блок може виглядати по різному, але основний принцип завжди зберігається. На даній сторінці є можливість подавати інформацію про себе у вигляді фотографії, імені, дати народження і іншого. Також важливим є блок власних постів, де зберігаються всі опубліковані користувачем дописи. Також тут можуть бути компоненти для добавлення нових дописів, списки друзів, галерея фотографій та відео, бібліотеки аудіо записів. Також часто саме на сторінці профілю може бути нотатки користувача – окремий блок на сайті, де є можливість писати власні думки і записи, які не будуть публікуватися для інших.

Сторінка для спілкування може набувати будь яких видів, форм та способів представлення. Основною цілю такого компоненту це здійснення спілкування між користувачами застосунку. Вони можуть набувати різних форм, наприклад на сторінці Facebook вікно чату подається у вигляді невеликого вікна в правому нижньому куті сторінки(рис.1.1).

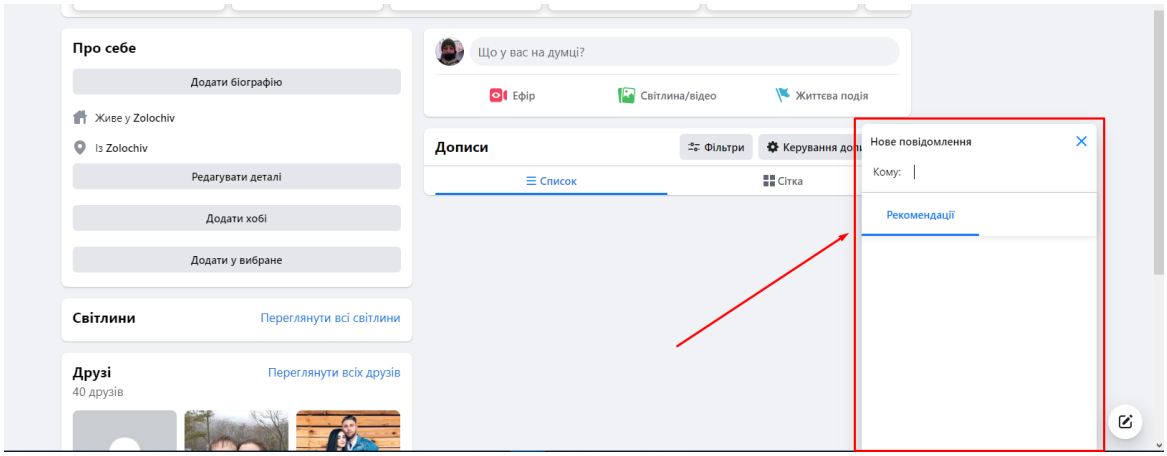

Рисунок 1.1 – Вікно чату на сторінці Facebook.

Також вікно чату може подаватися у вигляді окремої сторінки як це є в Instagram.(рис.1.2)

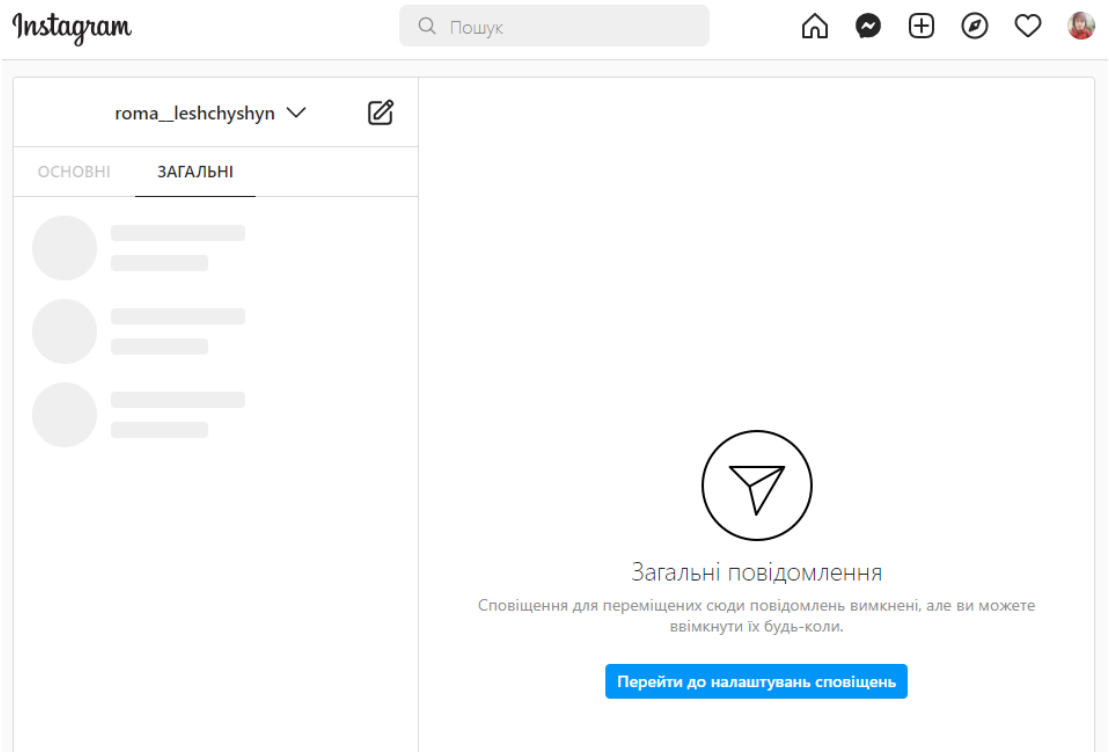

Рисунок 1.2 – Сторінка чату в Instagram.

Сторінка для постів зазвичай має вигляд стрічки, яка складається з дописів інших користувачів. У більшості соціальних мереж дана стрічка фільтрується по полярності або ж по статистиці інтересів користувача, таким чином щоб вона містила як умога більш цікаву інформацію для того хто її переглядає.

Часто прослідковується така тенденція, що стрічка постів може розгалужуватися для різних потреб. Наприклад одна стрічка призначена для постів тих користувачів на яких була здійснення підписка або які відстежуються профілем. Інша стрічка призначена для всіх інших постів інших незнайомих людей.

1.3 Технології для створення сучасних соціальних мереж

На сьогоднішній день появилася надзвичайно велика кількість технологій для розробки різноманітних веб застосунків на базі різних мов програмування.

Перші застосунки розроблялися на базі php, html, C++. Однак протягом всього часу розвивалися такі мови як Python і JavaScript. Почали появлятися різноманітні бібліотеки та інструменти. Наступним кроком в розвитку технологій стала поява фреймворків, де були зібрані всі необхідні інструменти заточені під конкретний вид діяльності. На сьогоднійшній день веб технології нараховують мілійони бібілотек та тисячі фреймворків які дозволяють розробляти веб застусунки дял будь яких цілей: форуми, веб ігри, платформи, соціальні мережі.

В процесі розвитку JavaScript ця мова перейшла з ролі декоратора сторінок і повноцінний рушій, на базі якого будуються і успішно функціонують великі проект.

Стрімкого скачку набула веб розробка в 2009 році коли відбувся реліз Node.js, платформи для створення серверних застосунків на базі JavaScript.

На перших версіях платформа підтримувалася тільки на базі Linux та Mac OS, що і в такому вигляді позволило ввести в масову експлуатацію платформи так як більшість серверів перебувають на платформі Linux.

Основною особливістю Node.js є те, що він містить у собі рушій V8, який був розроблений компанією Google для веб браузерів. Даний рушій був винесений поза середовище браузера і появилася можливість запускати програми на JavaScript поза браузером, що радикально розширило сферу впливу мови.

Для платформи Node.js появилися свої фреймворки на прикладі Express легшого і зручнішого створення сервера. Решта додаткового функціоналу підключається до платформи у вигляді модулівСписок модулів є достатньо великим, так як в них міститься весь функціонал(рис.1.3).

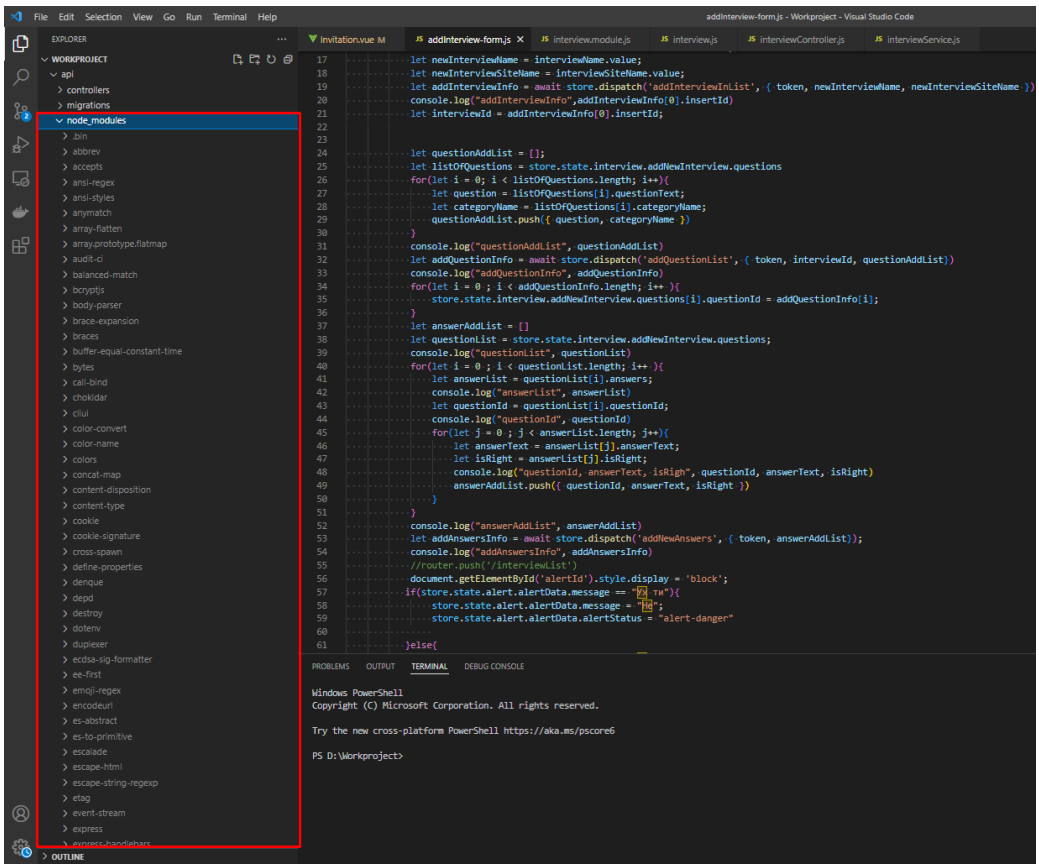

Рисунок 1.3 – Список модулів Node.js

Node.js тісно співпрацює з таким застосунком як npm. Npm використовується для того щоб завантажувати додаткові модулі в проект. Аудиторія платформи настільки велика і різноманітна, що модулі можливо найти практично під будь яку потребу.

Також Node.js використовуються не тільки для створення серверів, а ще для того щоб програмісти завантажували додаткові інструменти, які б допомагали в роботі, наприклад різноманітні препроцесори, модулі для редагування синтаксису. Також найбільш популярним модулем для веб розробників є Gulp, таск менеджер, за допомогою якого можна створювати скріпти для автоматизації певних етапів розробки.

Node.js використовується для створення серверної частини програмного забезпечення, однак в такому випадку необхідний ще один застосунок, який звязаний з беккендом. Ця частина називається фронтендом. Для фронтенд частини також необхідно створювати сервер на базі Node.js, але є одна велика різниця між двома додатками. Бекенд частина застосунку використвується для

обробки запитів та керування інформацією в базі даних. Фронтед частина забеспечує користувача інтерфейсом та набор потрібного функціоналу у вигляді кнопок, полів для вводу і багато іншого. Саме фронтенд чатсина відображається на стороні користувача і самец я частина утворює і відправляє запити на серверну частину, яка в свою чергу обробляє її і надає певний результат у вигляді відповіді на запит або ж запис оброблених даних в базу даних(рис.1.4).

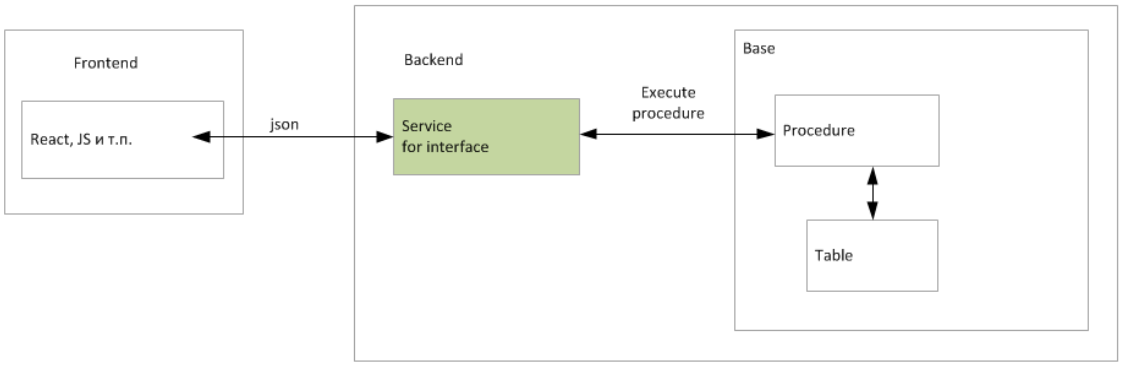

Рисунок 1.4 – Структура взаємодії частин застосунку

Фронтова частина напочатку свого розвитку могла складатися тільки з html, css, js. Однак накопичувалася велика кількість бібліотек та методологій, які в подальшому були погруповані в зручний і повномаштабний набір інструментів. Таким чином появилися препроцесори та фреймворки.

Напопулярнішими фреймворками рахуються React.js, Vue,js та Angular. Вони маю ряд радикальних відмінностей один від одного, але призначені для одного і того – це створення динамічних веб застосунків.

Динамічний сайт – це сайт, який складається з веб сторінок, які динамічно змінюються. Принцип полягає в тому, що сторінка сайту без перезавантаження здатна зміни відображення власних компонентів. Даний метод набуває все більшої популярності так як сама структура динамічних сайтів передбачає, що є один глобальний шар сторінки, до якого динамічно можуть добавлятися незалежні компоненти які можуть використовуватися безкінечну кількість разів без дублювання його в коді.

При такій методиці дуже зручно збільшувати кількість веб-сторінок на сайті, так як необхідно дописувати не цілу сторінку а окремі компоненти, яких не вистачає системі. В такому випадку система має вигляд конструктора, де додаються все нові компоненти без негативного впливу на всі інші(рис.1.5).

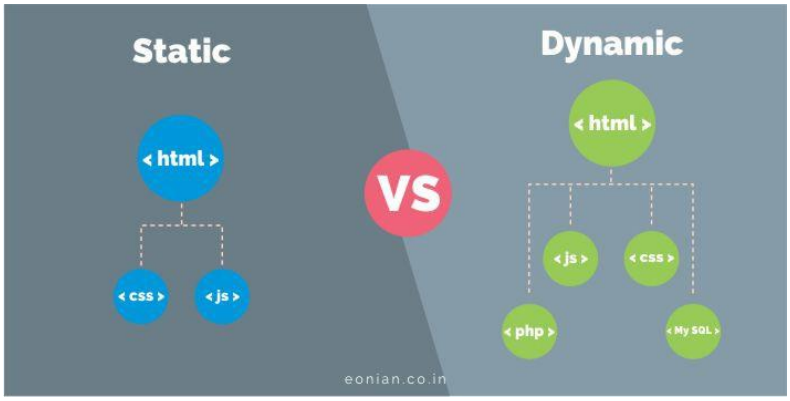

Рисунок 1.5 – Відмінність технологій для динамічного і статичного сайтів.

Однак дана система містить і ряд недоліків. Динамічні сайцти потребують більше апаратних ресурсів хостингу так як для динамічних сторінок використовуються сторонні фреймворки та ПЗ. Також подібні сайти складніше переносити на нові хостинги так як вони повинні підтримувати ті технології, які були використанні в даному продукті.

РОЗДІЛ 2 ПРОЕКТНА ЧАСТИНА

2.1 Опис та налаштування Node.js

Node.js являється платформою для розробики програмного продукту на базі рушія V8, який в свою чергу був запозичений у браузерів та підтримує мову програмування Java Script. Перевагою даноъ платформи є те що вона є середовищем з відкритим програмним кодом, що дозволяє зручно і гнучко налаштовувати будь який аспект. За допомогою Node.js появилася можливість створювати сервери на мові Java Script та різноманітні настільні застосунки.

Перевагами даної платформи є асинхронність. Під асинхронністю розуміється те, що програмний код виконується не послідовно, що дозволяє процесам, яким необхідно більше часу на обробку не займати весь ресурс і дозволити виконання програми далі.

Це актуально особливо в веб-розробці так як запити на сервер або від нього надходять не моментально і при великій кількості таких запитів без асинхронного режиму програма буде в рази повільніше виконувати свої завдання(рис.2.1).

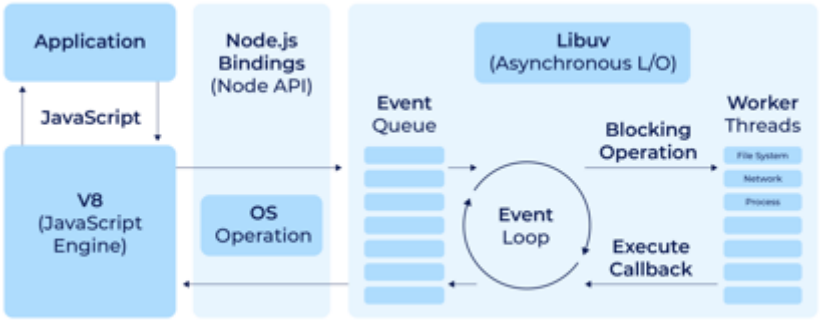

Рисунок 2.1 – Архітектура Node.js

Для того щоб все працювало корректно, необхідно в першу чергу встановити Node.js на робочу станцію, де буде відбуватися розробки і в подальшому робота самого застосунку.

Залежно від архітектури ОС потрібно завантажити і розпакувати архівне ім'я вузла v6.3.1 з ім'ям .tar.gz в / tmp, а потім, перемістити витягнуті файли в каталог / usr / local / nodejs. наприклад:

Лістинг 2.1 – Команди для налаштування node.js

```
$ Cd / tmp
$ Wget http://nodejs.org/dist/v6.3.1/node-v6.3.1-linux-x64.tar.gz
$ Tar xvfz node-v6.3.1-linux-x64.tar.gz
$ Mkdir -p / usr / local / nodejs
$ Mv node-v6.3.1-linux-x64 / * / usr / local / nodejs
```

Потрібно додати / usr / local / nodejs / bin в змінну оточення PATH. Операційні системи Вихід Linux експорт PATH = $ PATH: / usr / local / nodejs / bin макінтош експорт PATH = $ PATH: / usr / local / nodejs / bin FreeBSD експорт PATH = $ PATH: / usr / local / nodejs / bin

Потрібно використати файл MSI і дотримуватися інструкцій для установки Node.js. За замовчуванням програма установки використовує дистрибутив Node.js в C: \ Program File \ nodejs. Установник повинен встановити каталог C: \ Program File \ nodejs \ bin в змінній оточення PATH вікна.

2.2 NPM – пакетний менеджер

Пакетний менеджер виконує ряд поставлених функцій для того щоб розробнику було зручніше керувати власним проектом.

Репозиторій, де є можливість зберігати власні модулі і пакети, щоб в подальшому стандартизовано їх підтягувати для інших проектів або ж ділитися з іншими розробниками.

Утиліта для командного рядка, за допомогою відвідних команд може керувати пакетами і модулями у власному проекті, що покращує і спрощує процес розробки(рис.2.2).

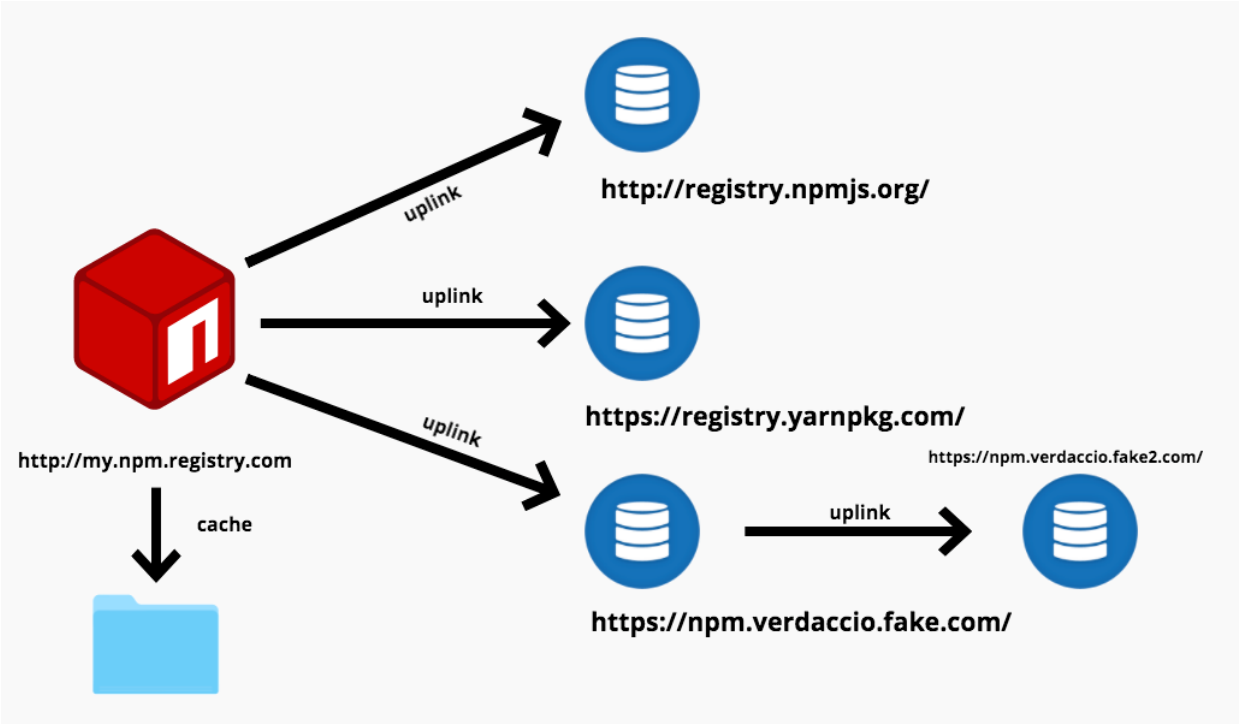

Рисунок 2.2 – Принцип роботи NPM

Пакетний менеджер зазвичай керується файлом package.json, звідти він отримує інформацію про пакети, їх версію та стан. Також у цьому файлі описана інформація про сам проект, його версія, модулі, залежності і бібліотеки.

2.3 Опис React.js

React підтримує декларативну парадигму проектування, що радикально спрощує процес розробки веб додатків.

Декларативне програмування представляє собою одну з парадигм програмування, де описується очікуваний результат а не послідовні кроки щоб отримати його.

React у свою чергу вимагає від розробника щоб він описав як повинен поводити себе і як виглядати певний компонент у кожному стані, після чого механізм фреймворка автоматично буде рендерити і вносити зміни в залежності від вказаних потреб.

Практично усі фреймворки для веб розробки слідують принципу розбиття усіх елементів на компоненти. Даний метод є дуже продуктивним і зручним, так як створений один раз компонент може використовуватися безліч раз і певними динамічними змінами у вигляді чи наповнені. Такий принцип позволяє створювати складні структури застосунків та інтерфейси.

React вважається найпопулярнішим фреймворком або бібліотекою на Java Script для розробки веб застосунків та користувацького інтерфейсу. Його було розроблено компанією Facebook. З моменту створення дана бібліотека конкурувала з таким бібліотеками як Bootstrap та фреймворками на прикладі Angular(рис.2.3).

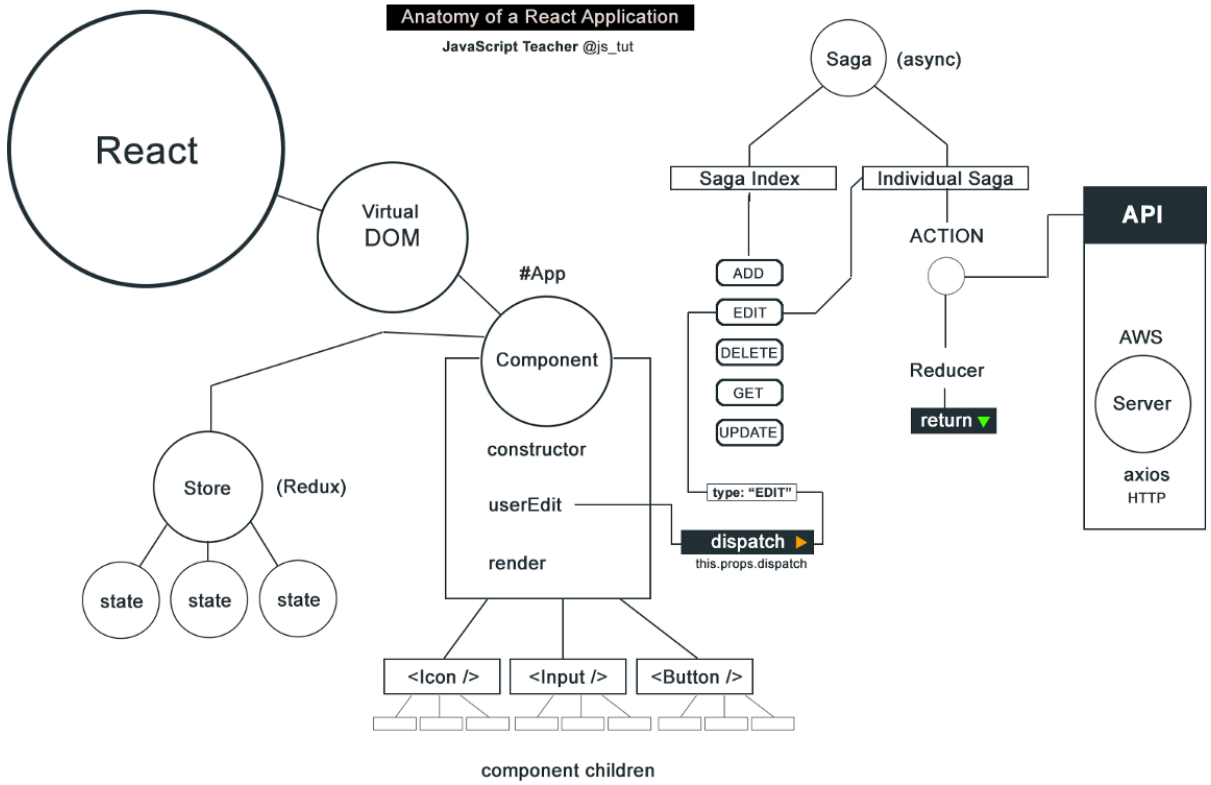

Рисунок 2.3 – Архітектура застосунку на React

Написання компонтетів відбувається у форматі JSX, її можна назвати сумішю html та javascript. За допомогою цього процес створення компенента а в подальшому і сторінки стає набагато зручнішим, легшим і зрозумілішим.
Також вадилвою особливістю фреймворка являється те, що він використовує віртуальний DOM сторінки, який зберігається в памяті, при кожних змінах Віртуальне DOM негайно змінюється. Ця система створена для того щоб DOM сторінки не впливав на продуктивність і не вимагав постійного перезавантаження сторінки.

Також великою переваою даного феріймворка є те що він адаптований під SEO, що допомагає сайтам побудованим на React попадати в топи списків при видачі в пошуковику браузерів. Це є дійсно великим плюсом так як не всі аналоги можуть забезпечити належний рівень SEO оптимізації.

2.4 Методи життєвого циклу

Важливу роль в розробці застосунку на React є методи життєвого циклу компонента. Це є однією з фундаментальних концепцій при розробці на React(рис.2.4).

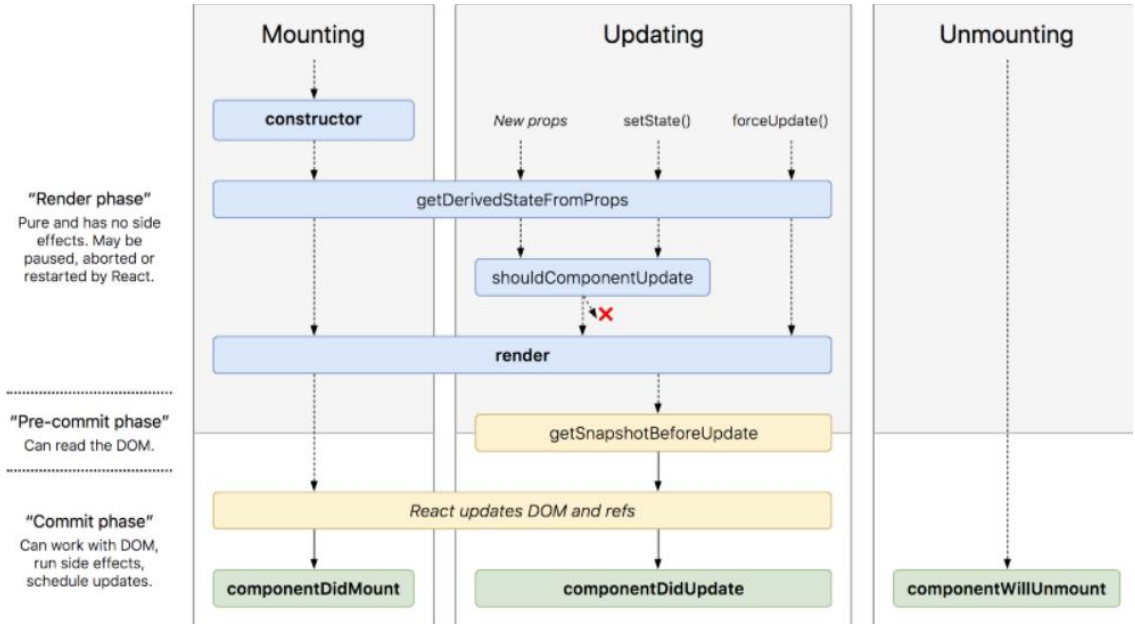

Рисунок 2.4 – Життєвий цикл компонента.

Компоненти появляються, виконують покладене на них і в подальшому закінчують власне існування. Зазвичай життєвий цикл ділиться на три етапи: Mounting, Updating, Unmounting. Важливо замітити що компонент може находитися одночасно тільки на одному етапі. Починаючи з монтування на сторінці компоненти постійно оновлюються до того часу поки вони не виконають всі завдання і будуть демонтовані зі сторінки. Такий підхід позволяє прописувати код, який буде виконуватися в різний момент часу або за певних умов чи змін.

Перший етап називається монтуванням. Процес монтування відбувається наступним чином: спочатку запускається метод constructor(). Саме в цьому методі і відбувається ініціалізація самого компонента і його стану. Після цього запускається метод getDerivedStateFromProps() і render(). Після цих кроків компонент монтується в DOM. Останнім кроком в цьому етапі є запуск componentDidMount(), де відбуваються всі асинхронні процеси.

Етап оновлення компоненту запускається кожен раз як відбулися певні зміни в компоненті. Тут запускається метод shouldComponentUpdate(), де можна керувати старими і новини даними про зміни, також тут можна керувати і самим відображенням компонента.

Останнім етапом в життєвому циклі компонента є демонтування з DOM. Даний етап запускає метод componentWillUnmount(), як здійснює розмонтовування компонента.

Існує ще багато додаткових методів, які повязані з повторним відображенням з інформацією про помилки та іншим.

2.5 Організація бази даних

Визначення у поняття база даних – це організована структура даних, які обєднані між собою за певною предметною областю ознакою або ж властивістю.

Практично уся інформація в інтернеті зберігаються в базах даних.

По цій причині постійно розвиваютсья і виникають технології для покращення збереження та керування цією інформацією. Бази даних це окрема галузь знань, в якій відбуваєтсья постійна орзробка та вдосконалення методик обробки інформації. З кожним днем даних стає більше на сотні терабайтів і виникає необхідністьоптимізовувати всі процеси, робити так щоб доступ до потрібних даних був простішим і без лишніх навантажень систем. Окремими питанням є захищеність баз даних і їх відмовостійкість. В базах берігаєтсья стільки надзвичайно важливої інформації, що при виході зі строю або будь яких некоректних операціях можуть бути втрачені дані, які поставлять під загрозу існування цілої компанії чи установи.

Існує два основних підходи до створення моделі бази даних: реляційний і не реляційний.

Найпоширенішою моделю на даний момент являється реляційна модель, її принцип полягає у тому, що інформація зберігається в таблицях, а пошук відбувається за допомогою ключів і звязків(рис.2.5).

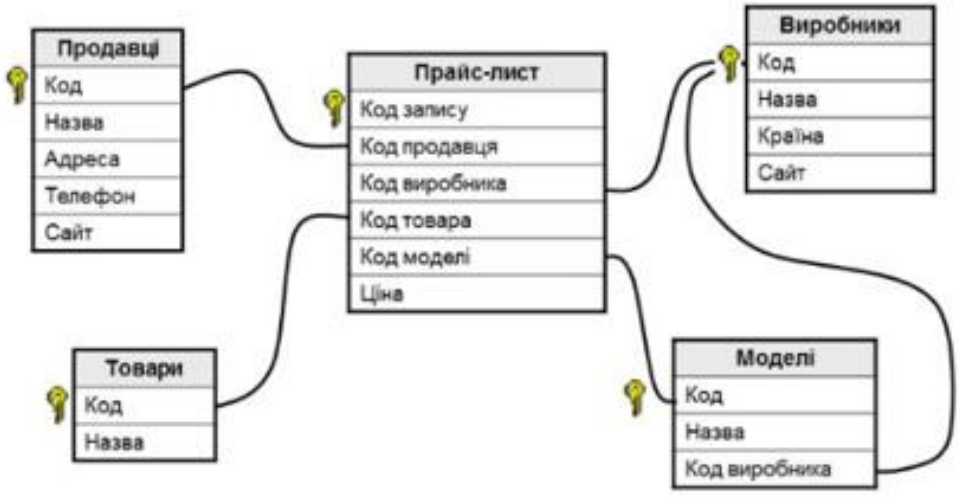

Рисунок 2.5 – Структура бази даних

В реляційній моделі представлений принцип нормалізації, який полягає у наступному:

– не повинно бути полів, які повторюються;

– в кожній таблиці повинен бути первинний ключ;

– кожному значенню первинного ключа повинна відповідати достатня інформація про тип суті або про об’єкт таблиці;

– зміна значень в полях таблиці не повинна впливати на інформацію в інших полях.

Також саме в реляційні моделі є звязки, які поділяютсья на різні види в залежності від потреб і вимог.

– Звязки, які існують між записами двох таблиць:

– один до одного – кожен запис в одній таблиці відповідає запису в іншій таблиці;

– один до багатьох – кожен запис в одній таблиці відповідає багатьом записам в іншій таблиці;

– багато до одного – кілька записів в одній таблиці відповідають одному запису в іншій таблиці;

– багато до багатьох кілька записів в одній таблиці відповідають кільком записам в іншій таблиці.

Зв’язок «один до багатьох» створюється, коли лише одне поле є первинним ключем або полем унікального індексу.

Зв’язок «один до одного» створюється, коли обидва поля є ключами або мають унікальний індекс.

Зв’язок «багатьох до багатьох» насправді є зв’язком «два до багатьох» із третьою таблицею, де первинний ключ складається з полів зовнішнього ключа двох інших таблиць.

2.6 Мова запитів SQL

SQL – декларативна мова баз даних, яка орієнтується на великий обєм обробки інформації. Дана мова розроблялася для підтримки реляцыйних баз даних.

- SQL зазвичай вирішує наступні питання:
- Модифікація структури бази даних
- Зміна параметрів
- Налаштування пріоритетів та вповноважень
- Отримання даних через запити
- Внесення змін або добавлення нових даних.

Цю мву підтримують такі бази даних: Microsoft Access, Oracle, Sybase або Informix.

Особливістю SQL є те, що вона нечутлива до регістру, винятком є прямі посилання на таблицю чи ключ.

2.7 СУБД MySQL

MySQL являється системою управляння баз даних. Являється одни з найпопулярніших СУБД так як поширюєтсья безплатно і оптимізований під веб застосунки, маючи підтримку такого типу даних як html.

MySQL являється дуже добре масштабованою, що дозволяє розробляти і керувати величезними глибокими структурами баз даних. Також MySQL являється системою з відкритим кодом, що дозволяє гнучко налаштовувати будь яких параметр для різноманітних цілей.

MySQL вважється найзахищенішою юбазою даних. По цій причині саме її використовують величезні веб додатки як Youtube, Facebook, Twitter, WordPress.Останні версіїї ззабезпечують безпеку навіть на рівні транзакцій, що позитивно впливає на різномінітні бізнес проекти та установи.

За продуктивністю MySQL також вибивається в лідери. Призначений для для дуже вимогливих систем, при цьому надаючи дуже високу швидкість обробки даних. Все це забезпечується унікально техноглогією для СУБД кеш памятю та відповідними покажчикам.

По вище вказаним критеріям можна відмітити що MySQL є економічно вигідним, з гнучким налаштування і швидкою роботою, зникнуть проблеми з простоями.

В результаті усіх факторів MySQL являться хорошим вибором для створення проекту так як підтримується такими мовами як Java Script, Python, Java. Та може бути інтегрований в багато інших мов та систем.

РОЗДІЛ 3 ПРАКТИЧНА ЧАСТИНА

3.1 Налаштування середовища розробки

Першим етап при безпосередній розробці програмного продукту це є налаштування середовища розробки, яке в подальшому буде і середовищем виконання веб застосунку.

Необхідно налаштувати сервери. Перший призначений для бекенд частини, точніше того функціоналу, який буде оброблятися на стороні сервера. Наступним сервером буде сервер для роботи фронтенд частини. Так як проект базується на різноманітних бібліотеках, необхідний сервер який буде все це підтримувати(рис.3.1).

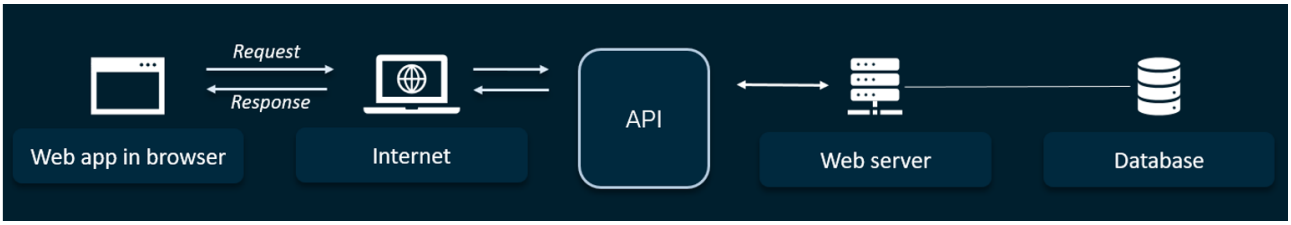

Рисунок 3.1 – Архітектура звязку веб-застосунку.

Останнім сервером буде сервер для роботи бази даних. Всі ці сервери будуть спілкуватися між собою за допомогою запитів. Фронтенд з бекендом будуть спілкуватися за допомогою API, в той час як веб сервер буде спілкуватися з базою даних відповідно того як це буде підтримувати модуль звязку.

3.1.1 Налаштування веб-сервера

Веб сервер буде налаштовуватися на базі Node.js. Найголовнішим файлом на момент налаштування середовища являється package.json. Саме в цьому файлі повністю описується іноформація про сервер, його залежності, назва, версії(рис.3.2)

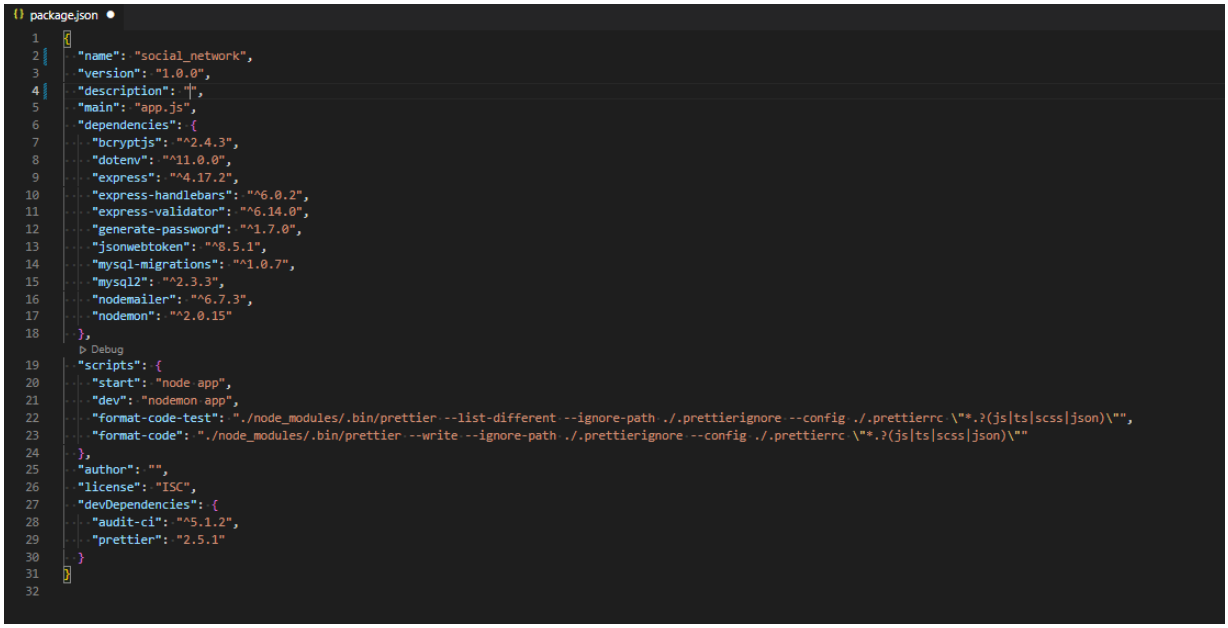

Рисунок 3.2 – Вміст package.json

В параметрі dependencies знаходиться обєкт зі списком усіх підключених модулів і їх версіій:

Лістинг 3.1– Список залежностей

```
"dependencies": {
    "bcryptjs": "^2.4.3",
    "dotenv": "^11.0.0",
    "express": "^4.17.2",
    "express-handlebars": "^6.0.2",
    "express-validator": "^6.14.0",
    "generate-password": "^1.7.0",
    "jsonwebtoken": "^8.5.1",
    "mysql-migrations": "^1.0.7",
    "mysql2": "^2.3.3",
```

```
    "nodemailer": "^6.7.3",
    "nodemon": "^2.0.15"
  }
```

Саме тут можна помітити, що підключені модулі для створення веб сервера, для налаштування змінних оточення, для генерації ключів, паролів, токенів. Також тут є модулі які забезпечують створення поштового сервера щоб надсилати листи користувачам і модуль для автоматичного перезапуску сервера при внесенні будь яких змінних в код програми.

Наступним обєктом являється параметр скриптів, де зберігаються короткі скрипти для зручності керування запуском сервера

Лістинг 3.2 – Список скриптів проекту

```
 "scripts": {
    "start": "node app",
    "dev": "nodemon app",
    "format-code-test": "./node_modules/.bin/prettier --list-
different --ignore-path ./.prettierignore --config ./.prettierrc
\"*.?(js|ts|scss|json)\"",
    "format-code": "./node_modules/.bin/prettier --write --ignore-
path ./.prettierignore --config ./.prettierrc \"*.?(js|ts|scss|json)\""
  }
```

Також важливим файлом який необхідно налашгтувати це файл змінних оточення під назвою .env. Саме тут прописані змінні, які можуть використовуватся в усьому проекті.

Лістинг 3.3 – Список змінних оточення

```
DB_HOST=94.130.131.170
DB_PORT=13306
DB_USER=sedu
DB_PASS=e0sKahyKw4JSBua3LR21pmuj9
DB_NAME=sedu
HOST=127.0.0.1
PORT=7040
```

```
TOKEN_KEY=cAtwa1kkEy
EMAIL_LOGIN=sedut1523es2315123@gmail.com
EMAIL_PASSWORD=1123516727821455
```

3.1.2 Налаштування середовища для веб-застосунку

Налаштування досить схоже з налаштуванням веб сервера так як базується на node.js. Також тут знаходиться package.json, однак містить вже інші налаштування(рис.3.3).

Рисунок 3.3 – Параметри файла package.json

Даний файл стандартизований так що містить аналогічні обєкти, але вже з іншим вмістом.

Лістинг 3.4 – Вміст поля dependencies

```
"dependencies": {
    "axios": "^0.21.1",
    "bootstrap": "4.6.0",
    "core-js": "^3.6.5",
    "jsonwebtoken": "^8.5.1",
    "module": "^1.2.5",
    "register-service-worker": "^1.7.1",
    "vee-validate": "^4.1.11",
    "vue": "^3.0.0",
    "vue-router": "^4.0.0-0",
    "vuex": "^4.0.0-0",
    "yup": "^0.32.8"
  }
```

В даному випадку вже використовуються модулі інших бібліотек так як вимоги до застосунку інші. Тут вже знаходяться валідатори, генератори токенів, модулі фреймворка, ініш бібліотеки та модулі для організації середовища змінних, який буде використовуватися всюди, де є підключений фреймворк.

3.2 Розробка серверної частини

В основу серверної частини положена MVC модель, що розшифровується як Model-View-Controller. В цьому випадку необхідно створити 3 каталоги для файлів(рис.3.4).

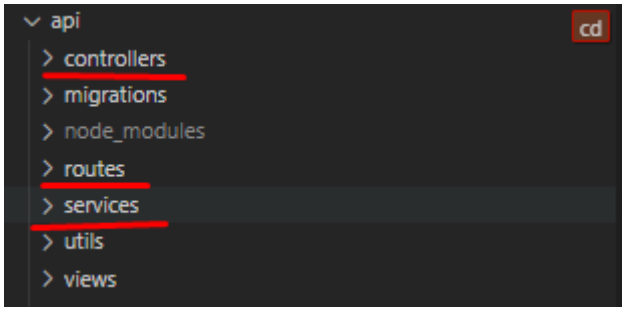

Рисунок 3.4 – Каталоги файлів.

Перший каталог routes мітить в собі файли маршрути для різних потреб. Прикладом є роутер для маршуртизації пир авторизації. Спершу йде підключення потрібних модулів в файлі.

Лістинг 3.5 – Підключення модулів та інших файлів

```
const {Router} = require('express');
const router = Router();
const {check} = require("express-validator");
var authController = require('../controllers/authController.js');
```

Далі йдуть маршрути для відпрацювання запитів.

Лістинг 3.6 – Маршрути для запитів

```
router.post("/register", [
    check("email", " Введіть корркткний Email").isEmail()
], authController.register);
router.post("/login", [
    check("email", "Введіть корркткний пароль").isEmail(),
    check("pass", "Пароль пароль невірний ").isLength({min: 4, max:
], authController.login);
router.post("/restore", [
    check("email", "").isEmail()
], authController.restore);
router.get("/recoveryPass/:recoveryToken",authController.
], authController.changePass);
```

Далі йде відповідний файл у каталозі controller. В цьому опрацьовується інформація отримана через тіло запиту і передається далі в сервіси.

В даному файлі зосереджена велика кількість методи, які опрацьовують відповідні запити. Наприклад запита на реєстрацію в системі.

Лістинг 3.7 – Контролер реєстрації

```
async register(req, res) {
        const validationErrors = validationResult(req);
        if (!validationErrors.isEmpty()) {
```

```
                return res.status(400).json({ message: "Некорректные
данные", validationErrors });
            }
            try {
                var {email} = req.body;
                var pass = req.body.pass || '';
                var obj = generalResponse.responseBody();
                var check = await authService.register(email, pass);
                if (check.success === true) {
                    obj.error = false;
                    obj.message = 'done';
                } else {
                    obj.message = "Неудалось зарегистрировать
пользователя: " + check.message;
                }
                if (process.env.SHOW_SQL) {
                    obj.sql = JSON.parse(process.env.SHOW_SQL);
                }
                res.status(generalResponse.responseStatus.success).json
(obj);
                await sendAccess(check.pass, email)
            } catch (e) {
                errorHandler(res,e);
            }}
```

Тут відбувається валідація вхідних даних на перевірку. Далі з тіла запиту отримуються дані. Після чого вони обробляються і передаються в метод, який імпортований з сервісів. В самому кінці формується відповідь від сервера і надсилається клієнтові.

Наступним йде Сервіс. Саме в цих файлах здійснюється кінцева обробка даних і запис їх в базу даних

Лістинг 3.8 – Сервіс реєстрації

```
async register(email, pass = '') {
        var conn = await this.dbConnect();
        pass = pass || passGenerator.generate({ length: 10,numbers:
```

```
            console.log('pass', pass);
            const hashPass = bcrypt.hashSync(pass, 7);
            if (!(await this.getUserByEmail(email))) {
                var sql = `INSERT INTO tbl_user (roleId, email,
password)
                    VALUES (1, '${email}', '${hashPass}')`;
                this.checkShowSql( sql);
                const [rows, fields] = await conn.execute(sql);
                this.dbClose(conn);
                return {
                    success: true,
                    pass: pass
                };
            } else {
                return{
                    success: false,
                    message: email
                }}}
```

Тут отримуються дані з контролера, також здійснюється необхідна обробка, після чого відкривається підключення до бази даних, через SQL запит записуються необхідні дані, закривається підключення і вертається результат операції назад в контролер щоб сформувати детальну відповідь від сервера.

Для того щоб всі файли корректно працювали необхідно у файлі app.js спочатку імпортувати всі роути.

Лістинг 3.9 – Імпорт муршрутів

```
const interviewRoutes = require('./routes/interview');
const interviewResultsRoutes = require(' / interviewResults');
const questionRoutes = require('./routes/question');
const questionAnswerRoutes = require('./routes/questionAnswer');
const authRoutes = require('./routes/auth');
const userRoutes = require('./routes/user');
```

Далі потрібно запустити ці роути за допомогою метода use, де в дужках прописаний шлях до запиту і сам роут.

Лістинг 3.10 – Запуск маршрутів

```
app.use('/api/interview', interviewRoutes);
app.use('/api/interviewResults', interviewResultsRoutes);
app.use('/api/question', questionRoutes);
app.use('/api/questionAnswer', questionAnswerRoutes);
app.use(authRoutes);
app.use('/api/user', userRoutes);
```

3.3 Розробка клієнтської частини

Структра клієнтської чатсини базується на двох шарах. Перший шар авторизації. Тут знаходиться заголовок та поля для вводу даних для підключення.

Також тут знаходиться кнопка для входу і посилання якщо забув пароль від системи(рис.3.5).

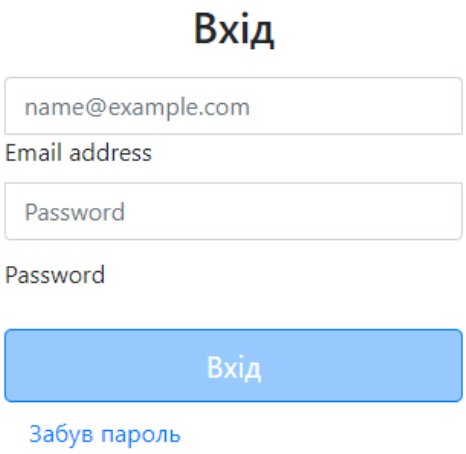

Рисунок 3.5 – Сторінка входу

Наступним шаром є сторінки самого застосунку. Шар призначені для того щоб не повторювати в кожній сторінці статичні елемент. Все що треба зробити щоб сторінка відрізнялася від одної це створення елемента для рендеру динамічних елементів в середині слоя. Такі елементи як меню, аватар, бокові панелі є статичними і не змінюються в залежності від інших компонентів(рис.3.6).

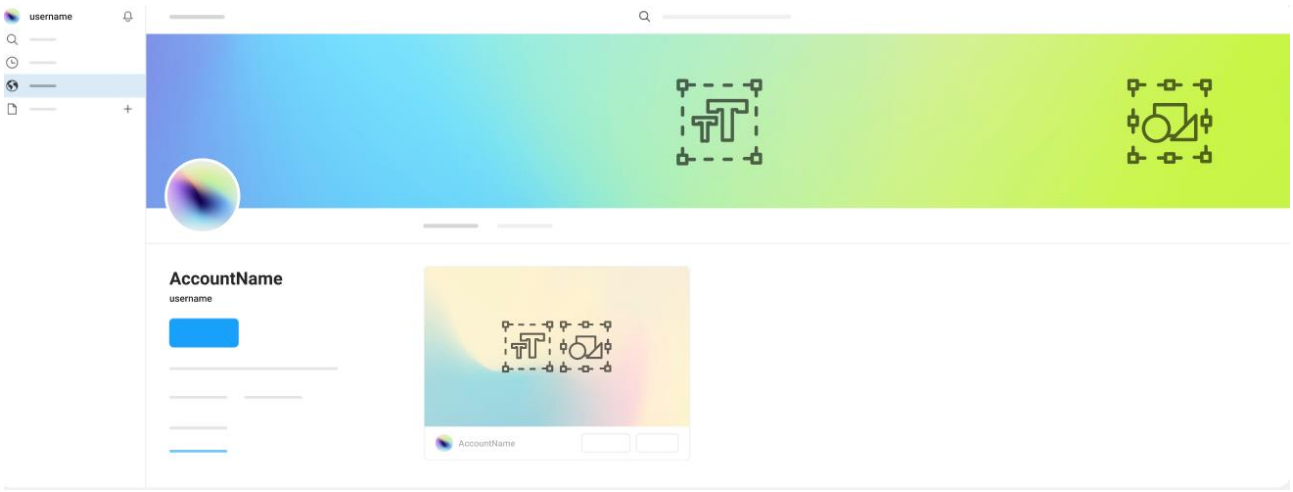

Рисунок 3.6 – Шар контенту застосунку

Структура файлів для клієнтської частини також поділяється на 4 каталоги(рис.3.7).

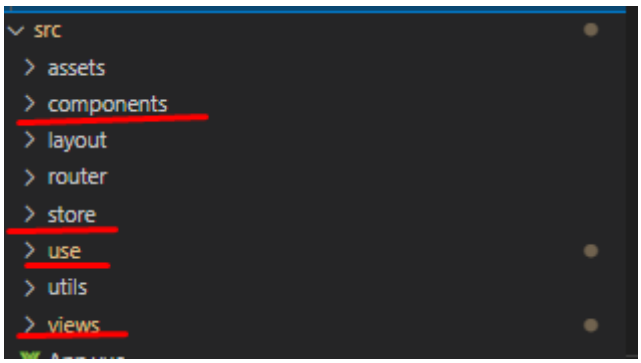

Рисунок 3.7 – Основні каталоги проекту.

В каталозі Views зберігаються шаблони конкретних видів, наприклад вікна авторизації(додаток А).

Основна структура файлів шаблонів – це блок тегів, блок скриптів та блок стилів.

Лістинг 3.11 – Структура файлів шаблонів

```
<template>
      ……
</template>
<script>
      ……
</script>
<style scoped>
      ……
</style>
```

Наступним каталогом йде католог use, де зберігаються обробники подій, та де саме відбувається запит до іншого каталогу store. Саме тут відбувається обробка інформація та формування даних для запитів, які потім передаються далі, де вже формується fetch запит до веб-сервера(Додаток Б).

Останнім важливим каталогом являється каталог модулів. Особливістю модулів є те що він оперується окремою бібліотекою. Тут відбувається структуризація даних та методів. Дані організовані так що можуть бути підтянуті з будь якого місця в проекті, де підключена потрібна бібліотека. Також тут є методи мутації за допомогою яких можна змінювати дані в обєкті state. Останнім що є в цих файлах це методи actions, де формуються фетч запити в веб сервер, керуючись даними які були отримані з форми модуля (додаток В)

3.4 Тестування системи

Тестування відбувалося за допомогою програми Postman. За допомогою даного застосунку можна тестувати арі запити та перевіряти відповіді від сервера і вхідні та вихідні дані(рис.3.8).

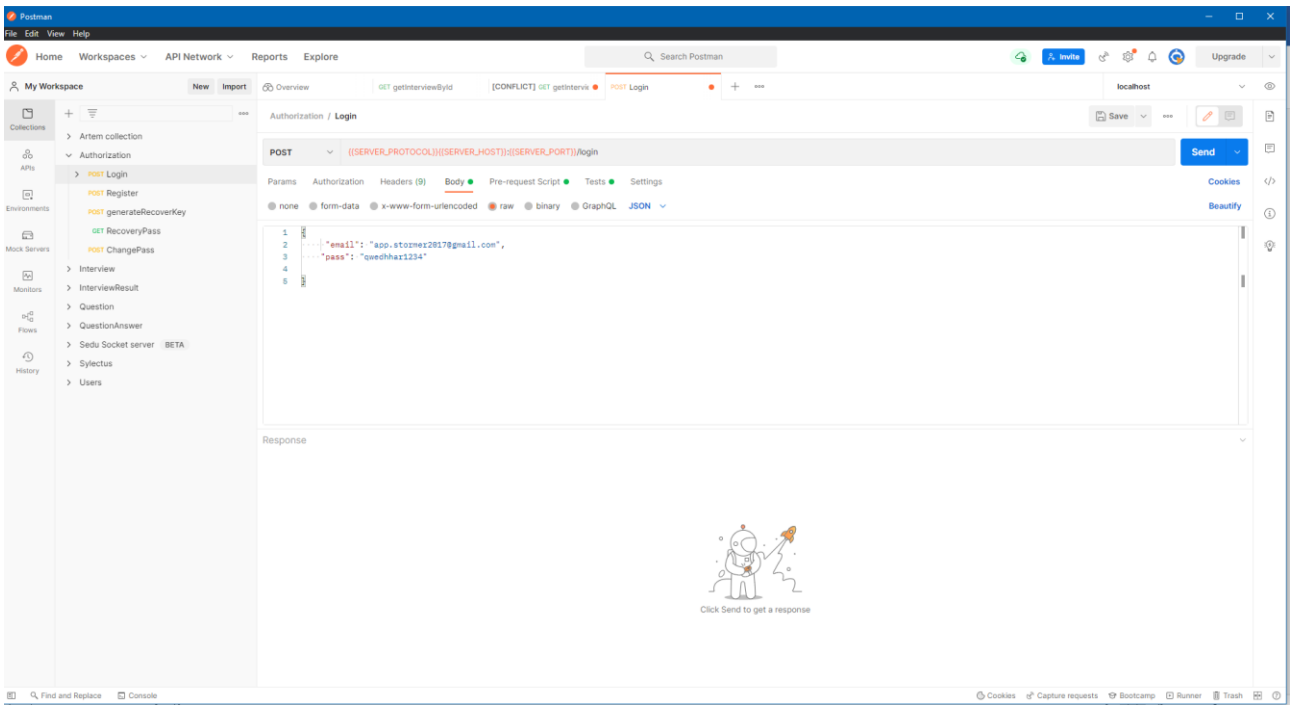

Рисунок 3.8 – Postman

Завдяки цій програмі можна вписати url адерс запиту, утворити get або post запит, після чого створити тіло запиту або ж передати інформацію в адресі. Таким чином можна перевірити корректність роботи запитів без запуску клієнтської частини.

Також тестування відбувалося за допомогою тест кейсів, де був описаний основний функціонал по яку необхідно покроково пройти перевірку. Таким чином було протестовано авторизацію, реєстрацію, надсилання даних на пошту користувача та весь функціонал який був повязаний з запитами на веб сервер і в базу даних

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Психологічні чинники небезпеки

Аналіз статистичних даних та висновки експертів у галузі безпеки життєдіяльності дають можливість стверджувати, що від 60 до 90% травм у побуті та на виробництві відбувається з вини потерпілих. Основні причини цього такі: низький рівень професійної підготовки з питань безпеки, недостатнє виховання, слабка установка людини на дотримання вимог безпеки, допуск до небезпечних робіт осіб з підвищеним ризиком травматизму, перебування людей у стані втоми чи інших психічних станах, які знижують безпеку діяльності.Виділяють комплекс чинників, що збільшують індивідуальну схильність людини до небезпеки. Це особливості темпераменту, функціональні зміни в організмі, дефекти органів відчуття, незадоволення даним видом діяльності.

Несприятливий характер діяльності (значні фізичні та розумові зусилля, незручна робоча поза, високий темп праці, нервово-емоційні перевантаження, перенапруга слухових та зорових аналізаторів, несумісність робочого місця, засобів праці, антропометричних даних людини та ін.) призводять до підвищеної фізичної та нервової втоми, яка послаблює психіку, знижує швидкість та точність орієнтації, притупляє пильність та увагу, порушує сприйняття того, що коїться. Це також спричинює травматизм.

Психологи виділяють спеціальний розділ, психологію безпеки, в якому розглядають психічні властивості та різноманітні форми психічних станів, що спостерігаються у процесі трудової діяльності. Психічні процеси становлять основу психічної діяльності. Без них неможливе формування знань та надбання життєвого досвіду.

Розрізняють пізнавальні, емоційні та вольові психічні процеси. Психічні властивості – це стійкі особливості особи: інтелектуальні, емоційні, вольові, трудові та ін.Психічні стани зумовлюють особливості психічної діяльності у конкретний період часу та можуть позитивно чи негативно впливати на всі психічні процеси. На думку багатьох психологів, ефективність діяльності (працездатність) людини залежить від рівня психічного напруження. Підвищення рівня психічного напруження істотно збільшує ефективність праці. Але існує критична межа активації, після якої результати праці знижуються аж до повної втрати працездатності. Існують два типи позамежевого психологічного напруження – гальмівний та збудливий. Гальмівний тип характеризується скутістю та сповільненістю рухів.

Людина не здатна з колишньою спритністю виконувати професій ні дії. Знижується швидкість реакцій, сповільнюється процес мислення, погіршується згадування, розпорошується увага та виникають інші негативні прояви, не властиві даній людині у спокійному стані. Збудливий тип проявляє себе гіперактивністю, багатомовністю, тремтінням рук та голосу. Оператори здійснюють численні, не продиктовані конкретною потребою, дії. Вони перевіряють стан приладів, крутять регулятори, поправляють одяг, розтирають руки. У них з'являється дратівливість, запальність, невластива їм різкість, грубість, уразливість. Позамежеві форми психічного напруження часто лежать в основі помилкових дій та неправильної поведінки у складній ситуації. що може спричинити травматизм та аварії. Серед особливих психічних станів, які мають істотне значення для безпеки життєдіяльності, психологи виділяють пароксизмальні розлади свідомості, психогенні зміни настрою та афектні стани, пов'язані з вживанням психічно активних засобів (стимуляторів, транквілізаторів, алкогольних напоїв). Параксизмальні стани – група розладів, яка характеризується-короткочасною (від кількох секунд до хвилин) втратою свідомості.

Такі стани характерні для деяких органічних захворювань головного мозку, епілепсії.

Сучасні методики дають змогу своєчасно визначити осіб із прихованою схильністю до пароксизмальних станів. Цим людям протипоказана робота на висоті, водіями автотранспорту та інша робота із підвищеною небезпекою. Психогенні зміни настрою та афектні стани виникають під впливом психічних дій. Зниження настрою та апатія можуть бути наявні від кількох хвилин до одного – двох місяців, Погіршення настрою спостерігається внаслідок конфліктних ситуацій, після загибелі близьких та в інших випадках. При цьому з'являються байдужість, млявість, загальна скутість, загальмованість, сповільнення темпу мислення. Погіршення настрою супроводжується погіршенням самоконтролю, що може стати причиною травматизму та збільшує ризик виникнення небезпечних ситуацій. Афектні стани (афект – вибух емоцій) можуть виникнути внаслідок виробничих невдач, під впливом образи. У стані афекту у людини розвивається емоційне звуження обсягу свідомості. Можуть спостерігатися різкі рухи, агресивні та руйнівні дії. Особи, схильні до афектних станів, належать до категорії з підвищеним ризиком травматизму та не повинні призначатися на посади з високою відповідальністю. Використання психічно активних засобів, включаючи алкоголь, збільшує ризик травматизму та знижує рівень безпеки діяльності. Вживання легких стимуляторів (чай, кава) допомагає у боротьбі з сонливістю і може сприяти підвищенню працездатності на короткий період.

Вживання ж активних стимуляторів на відповідальних роботах здатне викликати негативний ефект – погіршується самопочуття, зменшується швидкість реакції. Використання транквілізаторів, які діють заспокійливо та запобігають розвитку неврозів, може знижувати психічну активність, уповільнювати реакцію, викликати апатію та сонливість. Особливо потрібно підкреслити вплив на безпеку діяльності алкогольних напоїв. За різними даними, автомобільний травматизм у 40-60 % випадків пов'язаний з вживанням алкоголю. Встановлено, що 64% смертельних випадків на виробництві викликано вживанням алкоголю та помилковими діями загиблих. Для безпеки праці особливе значення має після алкогольна астенія (похмілля), яка не лише

знижує працездатність, а й призводить до загальмованості та притуплення відчуття обережності (рис.4. 1).

Рисунок 4.1 – Речовини, що негативно впливають на психіку.

Тривале вживання алкоголю спричинює алкоголізм, який супроводжується різним ступенем деградації особи. Люди, які страждають на алкоголізм, втрачають властиву їм точність та охайність у роботі. Вони дедалі частіше допускають помилки та стають нездатними для вирішення складних проблем, до швидкої та правильної орієнтації у нестандартних ситуаціях.

4.2 Вплив кольору на покращення умов праці

Колір – це візуальне явище, викликане реакцією на стимуляцію світла. Він пронизує кожен аспект людського життя, прикрашає звичайне і дарує красу та виразність повсякденним предметам. Кольори робочих місць в офісі дуже важливі для забезпечення ефективної діяльності в робочому середовищі. Кожен колір по-різному впливає на організм людини. Кожен відчуває колір по-своєму. Реакція людей на різні кольорові схеми залежить від їхньої культури, освіти, генетики та соціально-економічного рівня. Деякі кольори забезпечують спокій, деякі – комфорт, деякі – стимулюючі, а багато інших впливають по-різному. Наприклад, зелений колір часто асоціюється з природою та ростом, оскільки

більшість людей були свідками росту рослин. Синій майже універсально заспокоює, оскільки асоціюється з такими речами, як небо та вода.

Використання кольору в дизайні може вплинути на емоції та настрої людей, які переглядають ці кольорові палітри. Існують теплі, холодні і нейтральні кольори. До теплих кольорів належать відтінки червоного, оранжевого та жовтого. Загалом ці кольори енергійні та активні, мають відносно позитивні конотації. Холодні кольори включають відтінки синього, зеленого та фіолетового. Взагалі, холодні кольори є більш спокійними, ніж теплі кольори.

Нейтральні кольори часто набувають характеристик інших кольорів у палітрі і можуть використовуватися для посилення цих впливів. До основних нейтральних кольорів належать: чорний, білий, сірий, коричневий та бежевий .

Отже, слід підбирати відповідний колір, щоб забезпечити гарний настрій працівників, щоб заохотити продуктивність праці. У конфігурації чи розташуванні простору колір також відіграє важливу роль у впливі на великі чи малі площі. Наприклад, довге вузьке приміщення можна візуально розширити, якщо торцеві стіни пофарбовати в теплі, глибокі та насичені кольори, тоді як бічні стіни – в більш світлі, менш насичені кольори. Низька стеля здаватиметься менш гнітючою, якщо її колір світлий, тоді як високу стелю можна зробити нижчою за рахунок темно-синього, сірого або чорного(рис.4.2).

Рисунок 4.2 – Види кольорів за впливом на людину

Роботи, що вимагають високої концентрації, вимагають нейтральної кольорової гами. Такі роботи, як у бухгалтерів та адвокатів, вимагають холодної кольорової гами, тоді як журналісти найкраще працюватимуть у захоплюючих теплих енергійних кольорах із великим значенням контрасту.

Тим часом, вчений O'Brien припускає, що синій офіс ідеально підходить для тих, хто повинен зосередитись і сконцентруватися на числах, зелений - чудовий вибір для управління, оскільки він має балансуючий ефект, а жовтий - підходить для відділу продажу. Тому кольорову гамму, обрану для робочого місця чи офісу, слід робити з належним врахуванням, щоб забезпечити кращу якість роботи. Якщо були обрані невідповідні кольори, співробітники можуть зазнавати негативних психологічних наслідків, таких як стрес, депресія, неспроможність сконцентруватись. Як висновок, стає зрозумілим, що колір сприяє зосередженню уваги на виконуваній роботі.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи був досліджений процес розробки соціальної мережі. Були використані наступні технології: React, Node.js, SQL. Також були отримані навички п створенню серверів для різних потреб.

При створенні соціальної мережі було досліджено приницп роботи запитів в мережі. Також було дослідженй процес збірки та налаштування сервера, клієнта та бази даних.

При розробці використовувалася платформа Node.js для створення веб сервера і для роботи фреймворка з бібліотеками.

Результатом є програмний подукт – соціальна мережа, який складаєтсья з трох складових: веб-сервер, лієнт-сервер, база даних.

Веб сервер з клієнтом спілкується за допомогою арі запитів і слугує посередником між клієнтом і базою даних. База даних у свою чергу базується на реляційній моделі, де зосереджена вся необхідна інформація для роботи застосунку.

В результаті було розроблено систему, де користувачі можуть реєструватися, створювати профілі і переглядати профілі інших користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Нікольский А. Javascript на прикладах. Практика – СПб.: Наука і техніка, 2018. - 272 с. – ISBN 978-5-94387-762-9
2. Хавербеке Марейн. Яскравий JavaScript. Сучасне вебпрограмування. 3-е виб. – КиЇв, 2019. - 480 с. – ISBN 978-5-4461-1226
3. Кудрець Д. Практикум по HTML. Видавницьке рішення, 2018. — 60с. — ISBN 978-5-4496-0428-6.
4. Крокфорд Дуглас К83 Як влаштований JavaScript. — Київ: Наука і техніка, 2019. — 304 с. — ISBN 978-5-4461-1260-9
5. Кудрець Д. Основи CSS 2019. – 32 с. - ISBN 9785449621740
6. Флеваган, Девід. JavaScrirt: кишеньковй довідник, 3-е вид. : ООО ”І.Д. ВІльямс”, 2013. – 320 с. – ISBN 978-5-8459-1830-7
7. Сергіїв А. Створення сайтів на основі WordPress. — Киіів.: Видавництво «Косуля», 2015. — 128 с. – ISBN 978-5-8114-1928-9
8. Хеник Б. HTML і CSS: шлях до ідеалу. – Львів.: Друк, 2011. – 336 с. – ISBN 978-5-49807-864-9
9. Офіційний сайт React. [Електронний ресурс]. – Режим доступу: https://uk.reactjs.org(Дата звернення 23.03.2022)
10. сайт Wikipedia. [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Декларативне_програмуванн (Дата звернення 23.03.2022)
11. Інформаційни сайт. tutorialslink [Електронний ресурс]. – Режим доступу: https://tutorialslink.com/Articles/-Nodejs-Local-Environment-Setup/2314(Дата звернення 19.03.2022)
12. Інформаційни сайт. puzzleweb [Електронний ресурс]. – Режим доступу: https://puzzleweb.ru/php/00_teacher2.php(Дата звернення 25.04.2022)
13. Інформаційни сайт. futurenow [Електронний ресурс]. – Режим доступу: https://futurenow.com.ua/shho-take-sotsialni-merezhi-vydy-klasyfikatsiya-bezpeka/(Дата звернення 15.04.2022)

14. Інформаційни сайт. dspace [Електронний ресурс]. – Режим доступу: http://dspace.onua.edu.ua/bitstream/handle/11300/11778/Организация%20баз%20данных.pdf?sequence=1&isAllowed=y(Дата звернення 03.04.2022)

15. Сайт “Habr ” по програмуванню. [Електронний ресурс]. – Режим доступу: https://habr.com/ua/company/vds/blog/422893/ (дата звернення 19.03.2022)

16. Сайт зі статтями “Wikipedia”. [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/WordPress (дата звернення 13.03.2022)

17. Сайт по веб-програмуванню “Way to start”. [Електронний ресурс]. – Режим доступу: https://waytostart.ua/blog/telegram-bot (дата звернення 30.04.2022)

18. Сайт по веб-програмуванню “Westelecom”. [Електронний ресурс]. – Режим доступу: https://westele.com.ua/ua/blog/183_cto-takoe-veb-hosting.html (дата звернення 12.04.2022)

19. Сайт по веб-програмуванню “Webtec”. [Електронний ресурс]. – Режим доступу: http://www.webtec.com.ua/uk/articles/index/view/2011-05-05/web-site (дата звернення 11.03.2022)

20. Сайт по наданню хостингу “Hostiq”. [Електронний ресурс]. – Режим доступу: https://hostiq.ua/ukr/info/what-is-domain/ (дата звернення 11.03.2022)

21. Науковий сайт knutd. [Електронний ресурс]. – Режим доступу: https://er.knutd.edu.ua/bitstream/123456789/18123/1/APSD2021_V2_P130-133.pdf(дата звернення 13.06.2022)

22. Науковий сайт knutd. [Електронний ресурс]. – Режим доступу: http://elartu.tntu.edu.ua/handle/lib/3536 (дата звернення 13.06.2022)

ДОДАТКИ

Додаток А Технічне завдання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії

Кафедра комп'ютерних систем та мереж

«Затверджую»

завідувач кафедри КС

_____ Осухівська Г.М.

"____"_____ 2022 р.

СОЦІАЛЬНА МЕРЕЖА НА БАЗІ ФРЕЙМВОРКУ REACT ТА ПЛАТФОРМИ NODE JS.

ТЕХНІЧНЕ ЗАВДАННЯ

на __*6*__ листках

Вид робіт: Кваліфікаційна робота

На здобуття освітнього ступеня «Бакалавр»Спеціальність

123 «Комп'ютерна інженерія»

«УЗГОДЖЕНО»	«ВИКОНАВЕЦЬ»
Керівник курсового проєкту	Студент групи СІс-43
_____ к.т.н., Шингера Н. Я.	_____ Лещишин Р. А.
«____»_____ 2022 р.	«____»_____ 2022 р

Тернопіль 2022

1 Загальні відомості

1.1 Повна назва та її умовне позначення

Повна назва теми кваліфікаційної роботи: Соціальна мережа на базі фреймворку react та платформи node js.

Умовне позначення кваліфікаційної роботи: КС КРБ 123.226.00.00

1.2 Виконавень

Студент групи СІс-43, факультету комп'ютерно-інформаційних систем і програмної інженерії, кафедри комп'ютерних систем та мереж, Тернопільського національного технічного університету імені Івана Пулюя, Лещишин Роман Андрійович.

1.2 Підстава для виконання роботи

Підставою для виконання кваліфікаційної роботи є наказ по університету (№ 4/7-180 від 23.03.2022 р.)

1.3 Планові терміни початку та завершення роботи

Плановий термін початку виконання кваліфікаційної роботи – 23.03.2022р. Плановий термін завершення виконання кваліфікаційної роботи – 23.06.2022 р.

1.4 Порядок оформлення та пред'явлення результатів роботи

Порядок оформлення пояснювальної записки та графічного матеріалу здійснюється у відповідності до чинних норм та правил ІСО, ГОСТ, ЕСКД,ЕСПД та ДСТУ.

Пред’явлення проміжних результатів роботи з виконання кваліфікаційної роботи здійснюється у відповідності до графіку, затвердженого керівником роботи.

Попередній захист кваліфікаційної роботи відбувається при готовності роботи на 90% , наявності пояснювальної записки та графічного матеріалу.

Пред’явлення результатів кваліфікаційної роботи відбувається шляхом захисту на відповідному засіданні ЕК, ілюстрацією основних досягнень за допомогою графічного матеріалу.

2 Призначення і цілі створення системи

Призначенням програмного продукту Соціальна мережа є створення веб застосунку для подальшого створення в ньому середовища спілкування між людьми. Даний веб застосунком базується на роботі трьох серверів, які підтримують роботу клієнтської частини, серверної та бази даних.

До основних задач, які повинен виконувати веб застосунок це:

- Реєстрація в системі
- Створення профілю користувача
- Збереження його даних в базі даних
- Перегляд інших профілів.

2.2 Мета створення платформи

Метою створення платформи є розробка на застосунку на базі клієнт серверної архітектури, з використанням фреймворків та бібліотек. Веб застосунок розроблений у вигляді окремого об’єкту та забезпечує можливість спілкування між користувачами.

2.3 Характеристики об’єкту

2.3.1 Основні задачі та функції об’єкту

Основною задачею даного застосунку є можливість авторизації користувача в системі, створення та заповнення профілю користувача та перегляд інших профілів.

Щоб досягнути цих завдань необхідно необхідно вирішити певні питання, які полягають у взаємозвязку клієнтської, серверної частин та бази даних.

Доступ до даних користувачів та сервера повинен бути авторизованим та здійснюватись по захищеному протоколу зв’язку.

3 Вимоги до системи

3.1 Вимоги до платформи

При відправці запиту з клієнта на серверну частину повинна бути сформована відповідь в залежності від результату обробки запиту.

3.1.1 Вимоги до структури та функціонування системи

Структура соціальної мережі складається з наступного:

– Веб сервер
– Клієнт сервер
– База даних
– Поштовий сервер
– Основними функціональними вимогами продукту є:
– Створення профіля
– Збереження даних користувача

– Перегляд профілів інших користувачів.

3.1.2 Вимоги по діагностуванню

Діагностика програмної та апаратної частини відбувається за допомогою системного адміністратора платформи.

3.1.4 Перспективи розвитку програмного продукту

До перспектив належать:

– Адаптація під мобільні пристрої
– Реалізация захищеного звязку для спілквання
– Створення додаткового розважального функціоналу
– Оптимізація бази даних

3.1.5 Вимоги до надійності веб застосунку

В застосунку повинно бути забезпечена верифікована авторизація на сайті. Також обовязково необхідне належне забезпечення збереження особистих даних користувача.

3.1.7 Вимоги до апаратного забезпечення

Мінімальні вимоги до апаратного забезпечення сервера:

– процесор: Intel Xeon E2527 3.2GHz;
– графічний прискорювач: GeForce 960;
– оперативна пам’ять об’ємом: 64Gb DDR4 2600
– накопичувачі: 5TB Gb HDD (RAID 4).

Мінімальні вимоги до апаратного забезпечення користувача:

персональний комп’ютер користувача:

– процесор: Amd A8-6500 3.5GHz;
– графічний прискорювач: Nvidia GeForce 9600gt або Amd hd 4670;

– Оперативна пам’ять об’ємом: 3400mb

3.1.8 Вимоги до програмного забезпечення

Програмне забезпечення веб сервера: Linux Ubuntu, git, Apache, MySQL, node.js.

Програмне забезпечення клієнт сервера: Linux Ubuntu, git, Node.js, React, Bootstrap.

Використаний редактор коду Visual studio code.

4 Вимоги до документації

Документація повинна відповідати вимогам ЄСКД та ДСТУ

Комплект документації повинен складатись з:

– пояснювальної записки;

5 Стадії та етапи проектування

Таблиця 1 – Стадії та етапи виконання кваліфікаційної роботи бакалавра

№ з/п	Назва етапів роботи	Термін виконання етапів роботи
1.	Ознайомлення з завданням до кваліфікаційної роботи	
2.	Провести огляд літературних джерел по темі випускної бакалаврської роботи	
3.	Провести порівняльну характеристику програмних продуктів	
4.	Розробити функціональну схему роботи об’єкта проектування. Розробка моделі бази даних	
5.	Практична реалізація об’єкта проектування	
6.	Описати та розробити організаційні заходи спрямовані на поліпшення стану охорони праці	
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	
8.	Виконання завдання до підрозділу «Основи хорони праці»	
9.	Оформлення кваліфікаційної роботи	

10.	Нормоконтроль	
11.	Перевірка на плагіат	
12.	Попередній захист кваліфікаційної роботи	
13.	Захист кваліфікаційної роботи	

Додаток Б – лістинг шаблона авторизації

```
<template>
  <main class="form-signin" @submit.prevent="onSubmit">
    <form>
      <!--      <img class="mb-4" src="../assets/brand/bootstrap-logo.svg"
alt="" width="72" height="57">-->
      <h1 class="h3 mb-3 fw-normal text-center">Sign in</h1>
      <div :class="['form-floating',{invalid:eError}]">
        <input type="email" class="form-control" id="email" v-model="email"
@blur="eError"
               placeholder="name@example.com">
        <small v-if="eError">{{ eError }}</small>
        <label for="email">Email address</label>
      </div>
      <div :class="['form-floating',{invalid:pError}]">
        <input type="password" class="form-control" id="password" v-
model="password" @blur="pError"
               placeholder="Password">
        <label for="password">Password</label>
        <small v-if="pError">{{ pError }}</small>
      </div>
      <div class="checkbox mb-3">
      </div>
      <button class="w-100 btn btn-lg btn-primary" type="submit"
:disabled="isSubmitting || isTooManyAttempts">Sign in</button>
      <div class="text-danger" v-if="isTooManyAttempts">Too many times</div>
      <div>
        <label>
          <router-link to="/restorePass"><p class="nav-link text-
right">Forgot your password?</p></router-link>
        </label>
      </div>
      <p class="mt-5 mb-3 text-muted text-center">&copy; 2017–2021</p>
    </form>
  </main>
</template>

<script>
import {useRoute} from 'vue-router'
import {useStore} from 'vuex'
```

```
import {error} from '../utils/error'
import {useLoginForm} from '../use/login-form'
export default {
  setup() {
    const route = useRoute()
    const store = useStore()
    if (route.query.message) {
      store.dispatch('setMessage', {
        value: error(route.query.message),
        type: 'warning'
      })
    }
    return {...useLoginForm()}
  }}
</script>
<style scoped>
html,
body {
  height: 100%;
}
body {
  display: flex;
  align-items: center;
  padding-top: 40px;
  padding-bottom: 40px;
  background-color: #f5f5f5;
}
.form-signin {
  width: 100%;
  max-width: 330px;
  padding: 15px;
  margin: auto;
}
.form-signin .checkbox {
  font-weight: 400;
}
.form-signin .form-floating:focus-within {
  z-index: 2;
}
.form-signin input[type="email"] {
  margin-bottom: -1px;
```

```
  border-bottom-right-radius: 0;
  border-bottom-left-radius: 0;
}
.form-signin input[type="password"] {
  margin-bottom: 10px;
  border-top-left-radius: 0;
  border-top-right-radius: 0;
}
.form-control small {
  color: #e53935;
}
.form-control.invalid input {
  border-color: #e53935;
}
.bd-placeholder-img {
  font-size: 1.125rem;
  text-anchor: middle;
  -webkit-user-select: none;
  -moz-user-select: none;
  user-select: none;
}
@media (min-width: 768px) {
  .bd-placeholder-img-lg {
    font-size: 3.5rem;
  }
}
</style>
```

Додаток В – лістинг форми авторизції

```
import { useStore } from 'vuex'
import { useField, useForm } from 'vee-validate'
import * as yup from 'yup'
import {computed, watch} from 'vue'
import {useRouter} from 'vue-router'
export function addInterview(){
    const router = useRouter()
    const store = useStore();
    let questions = [];
    let alertData = {};
    console.log("Success")
    const { handleSubmit, isSubmitting, submitCount } = useForm();
    const onSubmit = handleSubmit(async values => {
        try {
            const token = store.getters["auth/getToken"]
            let addNewInterview = store.state.interview.addNewInterview;
            let newInterviewName = interviewName.value;
            let newInterviewSiteName = interviewSiteName.value;
            let addInterviewInfo = await
store.dispatch('addInterviewInList', { token, newInterviewName,
newInterviewSiteName })
            console.log("addInterviewInfo",addInterviewInfo[0].insertId)
            let interviewId = addInterviewInfo[0].insertId;
            let questionAddList = [];
            let listOfQuestions =
store.state.interview.addNewInterview.questions
            for(let i = 0; i < listOfQuestions.length; i++){
                let question = listOfQuestions[i].questionText;
                let categoryName = listOfQuestions[i].categoryName;
                questionAddList.push({ question, categoryName })
            }
            console.log("questionAddList", questionAddList)
            let addQuestionInfo = await store.dispatch('addQuestionList',
{ token, interviewId, questionAddList})
            console.log("addQuestionInfo", addQuestionInfo)
            for(let i = 0 ; i < addQuestionInfo.length; i++ ){
```

```
                store.state.interview.addNewInterview.questions[i].questi
onId = addQuestionInfo[i];
            }
            let answerAddList = []
            let questionList =
store.state.interview.addNewInterview.questions;
            console.log("questionList", questionList)
            for(let i = 0 ; i < questionList.length; i++ ){
                let answerList = questionList[i].answers;
                console.log("answerList", answerList)
                let questionId = questionList[i].questionId;
                console.log("questionId", questionId)
                for(let j = 0 ; j < answerList.length; j++){
                    let answerText = answerList[j].answerText;
                    let isRight = answerList[j].isRight;
                    console.log("questionId, answerText, isRigh",
questionId, answerText, isRight)
                    answerAddList.push({ questionId, answerText, isRight
})
                }
            }
            console.log("answerAddList", answerAddList)
            let addAnswersInfo = await store.dispatch('addNewAnswers', {
token, answerAddList});
            console.log("addAnswersInfo", addAnswersInfo)
            //router.push('/interviewList')
            document.getElementById('alertId').style.display = 'block';
           if(store.state.alert.alertData.message == "Ух ти"){
                store.state.alert.alertData.message = "Не";
                store.state.alert.alertData.alertStatus = "alert-
danger"
           }else{
               store.state.alert.alertData.message = "Ух ти";
               store.state.alert.alertData.alertStatus = "alert-
success"
           }
           setTimeout(function(){
                document.getElementById('alertId').style.display =
'none';
```

```
            }, 1000);
        } catch (error) {
            console.log("error", error)
        }
    })
    return {
        onSubmit,
    }
}
```

Додаток Г – лістинг файла модуля авторизації

```
import axios from 'axios'
const TOKEN_KEY = 'jwt-token'
export default {
    namespaced: true,
    state: {
        token: localStorage.getItem(TOKEN_KEY),
        userId: 0,
        roleId:0,
        authorizedUser: {},
        alertNotActive: true,
        alertSuccess: true,
        alertDanger: false,
        SendrestoreMessege: '',
        email: ''
    },
    mutations: {
        setToken(state, token) {
            state.token = token
            localStorage.setItem(TOKEN_KEY, token)
        },
        logout(state) {
            state.token = null
            localStorage.removeItem(TOKEN_KEY)
        },
        setUserInfo(state, userInfo) {
            state.authorizedUser = userInfo;
            state.userId = userInfo.userId;
            state.roleId = userInfo.roleId;
        },
        deleteToken(state) {
            state.authorizedUser = {};
            state.userId = 0;
            state.token = '';
            delete localStorage[TOKEN_KEY];
        },
        setAlert(state, alertSet) {
```

```
            state.alertNotActive = alertSet.alertNotActive;
            state.alertDanger = alertSet.alertDanger;
            state.SendrestoreMessege = alertSet.SendrestoreMessege;
        }
    },
    actions: {
        async login({ commit, dispatch }, payload) {
            const response = await fetch('http://127.0.0.1:7040/login', {
                method: 'POST',
                headers: {
                    'content-type': 'application/json'
                },
                mode: 'cors',
                body: JSON.stringify({
                    email: payload.email,
                    pass: payload.password
                })
            })
            const res = await response.json();
            commit('setToken', res.data.token);
            await dispatch("getUserInfo");
        },
        async getUserInfo({ commit, state, dispatch }) {
            const response = await
fetch('http://127.0.0.1:7040/api/user/getUserInfo', {
                headers: {
                    'Authorization': 'Bearer ' + state.token
                }
            });
            const res = await response;
            const resBody = await response.json();
            if (res.status != 200 || (res.status == 200 && (!resBody.data
|| !resBody.data.email))) {
                commit('deleteToken');
                return false;
            } else {
                commit('setUserInfo', resBody.data);
                return true;
            }
```

```
        },
        async restorePass({ commit, dispatch }, payload) {
                const response = await
fetch('http://127.0.0.1:7040/restore', {
                    method: 'POST',
                    headers: {
                        'content-type': 'application/json'
                    },
                    mode: 'cors',
                    body: JSON.stringify({
                        email: payload.email
                    }),
                })
            return  await response.json();
            }
    },
    getters: {
        getToken(state) {
            return state.token
        },
        isAuthenticated(state, getters) {
            return !!getters.getToken
        }
```