НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Наталія АУШЕВА
«___»_____2022 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерний моніторинг та геометричне моделювання процесів і систем»

спеціальності 122 «Комп'ютерні науки»

на тему: «Веб-система проведення інтерв'ю в частині backend»

Виконав:
студент IV курсу, групи ТР-81
Мордас Іван Сергійович _____

Керівник:
Доцент, к.т.н
Ходаковський Олексій Володимирович _____

Рецензент:
_____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ — 2022 року

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

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Спеціальність: 122 Комп'ютерні науки

Освітня програма: Комп'ютерний моніторинг та геометричне моделювання процесів і систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Наталія АУШЕВА
(підпис)
«___»_____2022р.

ЗАВДАННЯ

на дипломну роботу студенту

Мордасу Івану Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Веб-система проведення інтерв'ю в частині backend

керівник роботи Ходаковський Олексій Володимирович, кандидат технічних наук, доцент
(прізвище, ім'я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від «08» червня 2022р. №965-с

2. Строк подання студентом роботи 10.06.2022р.

3. Вихідні дані до роботи мова програмування Java Script, середовище розробки Visual Studio Code.

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) розробити веб-сервіс для проведення дистанційного інтерв'ю, проаналізувати вже існуючі програмні рішення, реалізувати алгоритм для обміну повідомлення між користувачами; реалізувати можливість підключення у відео форматі.

5. Перелік ілюстративного матеріалу

Функції JavaScript, CSS, HTML, бібліотека Redux, принципи роботи WebSocket і HTTP, створення гілки в Git, приклад JSON, клієнтські сервіси, клієнтське сховище даних, зміни даних у сховищі в залежності від дії, реєстрація користувача, перевірка обов'язкових полів , головна сторінка кімнатами

6. Дата видачі завдання «10» вересня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	25.05.2022	
2.	Вивчення та аналіз задачі	02.05.2022-08.05.2022	
3.	Розробка архітектури та загальної структури системи	09.05.2022-15.05.2022	
4.	Розробка структур окремих підсистем	16.05.2022-22.05.2022	
5.	Програмна реалізація системи	23.05.2022-25.05.2022	
6.	Оформлення пояснювальної записки	26.05.2022-28.05.2022	
7.	Захист програмного продукту	25.05.2022	
8.	Передзахист	07.06.2022	
9.	Захист	20.06.2022	

Студент _____ Мордас І.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Ходаковський О.В
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Метою даної роботи є розробка веб-системи для швидкого, ефективного та дистанційного інтерв'ю з використанням відео зв'язку між потенційним кандидатом на певну посаду і роботодавцем.

Було проаналізовано головні недоліки і переваги вже існуючих програмних продуктів і на основі отриманих даних створено багатофункціональну архітектуру, яка надає користувачам широкий спектр можливостей.

Даний веб-застосунок складається з частини frontend, де реалізований візуальний графічний інтерфейс для інтерактивної взаємодії з користувачем та backend частини, яка відповідає за логіку програми, обробку даних та обмін інформацією.

Записка складається з вступу, п'яти розділів, висновку, додатку та списку використаних джерел. Загальний обсяг роботи становить 61 сторінки.

Ключові слова: веб-система, архітектура, інтерв'ю, графічний інтерфейс, візуалізація даних.

ABSTRACT

The aim of this work is to develop a web-based system for fast, efficient and remote interviews using video communication between a potential candidate for a position and an employer.

The main disadvantages and advantages of existing software products were analyzed and based on the obtained data, a multifunctional architecture was created, which provides users with a wide range of opportunities.

This web application consists of the Front-end part, which implements a visual graphical interface for interactive user interaction and the Back-end part, which is responsible for program logic, data processing and information exchange.

The record consists of an introduction, five chapters, sources, appendix and a list of used. The total volume of the work is 61 pages.

Keywords: web system, architecture, interview, graphical interface, data visualization.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

Back-end — це частина програмного застосунку або коду, який відповідає за логіку програми і звичайний користувач не має доступу.

Front-end — гнучкий користувацький інтерфейс, що надає змогу користувачеві взаємодіяти з програмно-апаратною частиною сервісу.

JavaScript — це мова програмування або мова сценаріїв, яка дозволяє програмісту швидко і ефективно реалізовувати складні функції у веб-просторі.

Three.js — це міжплатформена бібліотека, що має великий спектр функціоналу для створення і відображення комп'ютерної графіки та взаємодії з 3D простором.

Visual Studio Code — це розроблений Microsoft фірмою середовище програмного коду з великою кількістю інструментів. Головними перевагами є його кросплатформеність, підтримка великої кількості мов програмування, вбудовані функції для швидкого і ефективного керування коду.

JSON — це відкритий файловий стандарт для обміну інформації, був розроблений на основі мови програмування JavaScript.

URL (уніфікований вказівник ресурсу) — стандартизована система адрес певного ресурсів.

PeerJs — це бібліотека мови програмування JavaScript, яка спрощує однорангові дані, відео- та аудіо- дзвінки.

NodeJs — це між платформне середовище з відкрити кодом. Надає можливості програмісту використовувати JavaScript, щоб створити серверну частину програму з визначеним набором допоміжних інструментів.

WebSocket — це протокол створений з метою обміну повідомлення між браузером і сервером в режимі реального часу. Через один TCP-сокет, він забезпечує двох направлений напівдуплексний канал зв'язку. Широкий спектр застосувань даної технології у багатокористувацьких іграх, чатах.

HTTP — це протокол передачі даних у мережі, широкого застосовується у комп’ютерних мережах. Схема за якою відбувається обмін повідомленнями це «запит-відповідь».

CORS — це механізм, який дозволяє на основі HTTP-заголовка вказувати будь-які джерела, з якого повинен браузер завантажити дані.

Meeting room — це кімната для переговорів, де відбувається конференція чи зустріч між двома або більше учасниками.

АПЕПС — скорочена назва кафедри Теплоенергетичного факультету КПІ, яке розшифровується, як автоматизація проектування енергетичних процесів і систем.

Бібліотека — це набір інструментів, об’єктів, підпрограм, в якій реалізовані шаблонні, базові алгоритми, які програміст може використовувати при розробці програмного продукту з метою полегшення розподілу розробки додатків.

Користувацький інтерфейс — це простір, який розроблений з метою взаємодії користувача з системою. Зазвичай він робиться максимально простим і в той же час зручним для користувача.

Рекрутер — це людина, яка займається пошуком і оцінкою нових потенційних кандидатів на певну посаду в компанії.

Веб-застосунок — це інтернет-додаток, який складається з серверною і клієнтською частиною і взаємодіє між собою за допомогою HTTP запитів. Ключовою особливістю є те, що відкривається не залежно від браузера чи операційної системи.

ВСТУП

В сучасному світі шаленими темпами розвиваються інформаційні технології, з кожним роком попит на спеціалістів зростає і настає момент коли потенційний кандидат на посаду повинен пройти інтерв'ю в бажану компанію або фірму. Проаналізувавши дані за останні роки, ми бачимо, що все більше і більше співбесід між кандидатом на посаду і представниками компанії відбуваються в онлайн, тобто в дистанційному форматі. Така форма проведення має ряд переваг таких як економія часу, швидкий обмін даними, місце проведення інтерв'ю. Проте ключовою проблемою для суспільства постало питання, як і через який застосунок чи веб-додаток проводити такі співбесіди?

Ідея розробити веб-систему для проведення інтерв'ю виникла через недосконаліть і складнощі вже наявних програмних продуктів. Зазвичай процес вступу ділиться на дві частини, це теоретична і практична. Більшість актуальних систем таких Zoom, Skype, WebEx, CodeSandbox надають можливості підключитися користувачам і провести у форматі питання-відповідь перший етап співбесіди. Найбільші труднощі виникають в процесі практичної частини, адже для цього потрібний додатковий функціонал і можливості даних програм.

Актуальність даної теми спричинена критичною ситуацією в світі, коли розвиток технологій йде шаленими темпами, а існуючі системи, не можуть задовільнити всіх потреб, світ стає мобільним і найбільше цінується свобода в пересуванні, в місці вибору інтерв'ю чи роботи. Наша система і створена з метою вирішення цих проблем, щоб надати доступ і всі можливості для швидкого і ефективного проведення співбесід по всьому світі, де для користувача і роботодавця буде надано широкий спектр можливостей.

Варто звернути увагу, що дану систему можна буде використовувати в навчальному процесі. Адже на етапі проведення практичних занять викладач може підключитися зі студентом і в режимі реального часу побачити, хід думок і етапи вирішення конкретної задачі.

ЗАВДАННЯ

Завданням даної роботи є створення веб системи, яка забезпечить доступ для проведення інтерв'ю в дистанційному режимі. Користувачам буде наданий широкий спектри можливостей для проведення, як теоретичної так і практичної частини. Тобто передбачається створення віртуального редактору, за допомогою якого, можна буде виконувати поставленні задачі і в режимі реального часу буде відображатися результат, також можна буде створювати нові кімнати і додавати туди нових учасників, фільтрувати різні запити.

Система буде розроблена у вигляді веб-додатку, який буде підтримуватися у різних браузерах і буде сумісним з різними операційними системами.

Користувачі матимуть різні права доступу, адже таким чином будуть певні обмеження до критичних налаштувань від звичайного користувача, з метою забезпечення цілісності і безперебійної роботи сайту. В сучасному світі коли інформація становить велику цінність і її захист становить одну з пріоритетних цілей ми передбачили використання приватних кімнат. Доступ до яких буде закритий і вхід здійснюватиметься за допомогою паролю.

Ключовим моментом будуть конференції з використанням відео зв'язку, таким чином при необхідності, можна буде увімкнути камери і провести опитування співрозмовника.

Перед розробкою буде проведено аналіз вже існуючих систем, з метою виявлення всіх наявних недоліків і їхніх усунення в нашому веб-додатку, в той же час за основу буде взято ту частину функціоналу, яка набула широкої популярності.

Актуальність даної теми спричинена геополітичною ситуацією в світі, коли більшість людей вимушено переходять на дистанційні форми роботи, навчання. Тому і було прийнято рішення про розробку універсальної і ефективної системи для проведення інтерв'ю, яка буде покривати практичні, і теоретичні частини.

Підсумувавши вище описані вимог, основним функціоналом, який повинен бути реалізований:

— реєстрація користувача в системі і його авторизація і автентифікація;

— створення онлайн редактора, який буде підтримувати різні мови програмування і консоллю, де відображатиметься результат і покрокові шаги;

— перегляд базових сторінок з коротким описом даного програмного продукту і його розробників;

— користувач повинен мати змогу фільтрувати кімнати для інтерв'ю;

—створення додаткового функціоналу для редагування, додавання кімнат для проведення інтерв'ю, тобто забезпечення всім необхідним функціоналом.

— Забезпечити безперервний зв'язок і обмін даними між користувачами за допомогою веб-сокетів;

— вибір мови програмування, якою буде здійснюватися інтерв'ю коду;

— створення приватних кімнат, доступ до яких можна буде отримати за допомогою унікального ідентифікатора.

В ході виконання даної дипломної роботи необхідно створити універсальний програмний продукт, з широким спектром можливостей для ефективного і зручного проведення інтерв'ю в дистанційному режимі.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

Останні два роки наший світ дуже сильно змінився особливо через пандемію Covid-19. Розвинута економіка була досить сильно відкинута на роки. Люди - були шоковані та вимушені залишалися в домівках та не мали змоги продуктивно працювати, як раніше в офісах разом з колективом. Роботодавці, в свою чергу, понесли дуже сильних збитків для свого бізнесу та були вимушені швидко щось змінювати та придумувати для наступної його підтримки, намагатися відкривати робочі місця для нових співробітників.

IT сфера вимушена була кардинально змінитися, щоб мати змогу працювати в умовах пандемії, сьогодні IT це один з найбільших секторів економіки в розвинутих країнах, який приносить колосальні внески в розвиток держав та світу в цілому. Завдяки спеціалістам, які ще вчора працювали в дружньому колективі та сучасному офісі, дуже швидко та легко адаптувалися до умов пандемії та карантину та почали виконувати роботу у себе в домівках з більшою ефективністю, що грало тільки на руку роботодавцям.

Очевидним є факт, що для розвитку компанії потрібно відкривати нові позиції та надавати робочі місця для нових спеціалістів. Звичайно кожен роботодавець хоче знайти кращих з кращих для ефективної роботи. Під час пандемію з цим з'явилися деякі проблеми.

Ще до пандемії, як правило, кандидатів просили проходити співбесіду в офісі. Хороший рекрутер міг з легкістю визначити кращі навички і вміння потенційного кандидата та зрозуміти чи досить цього для того щоб отримати це робоче місце.

В онлайн режимі коли навіть хороший рекрутер проводить співбесіду у нього досить легко може з'явитися думка що це чудовий та технічно обізнаний кандидат з великим набором технологій, а насправді він не зможе показати результату.

Коли пандемія тільки починалася, проведення співбесід було досить неприємним - було дуже неорганізовано та не зручно, навіть після двох років від початку, є компаніє де співбесіди проводять дуже незручно. Як правило, перший етап співбесіди це розмова HR менеджером і зазвичай їх ці зустрічі проходять у Skype, Zoom, Google Meet. Другий етап або технічне інтерв'ю, в свою чергу проходить в онлайн редакторах таких як CodeSandbox або CodePen та інші, щоб рекрутер мав змогу побачити, як пише кандидат код та думає в цілому. Досить часто співбесіди розтягуються на декілька днів і це також приносить деякі незручності і для потенційного кандидата і для рекрутера, який в свою чергу переключається з пріоритетних завдань. Важливо, щоб інтерв’ю проходило швидко і ефективно.

Слід згадати що дуже часто можуть попадатись недоброчесні кандидати, які можуть відкривати важливу інформацію на другому моніторі, та просто шукати її в інтернеті під час розмови, посилаючись на поганий зв'язок і в такий спосіб успішно пройти інтерв’ю.

Саме об'єднання ключових функцій, таких як онлайн редактор та спілкування в режимі онлайн, в одній зручній та зрозумілій веб-системі для комфортного проведення співбесіди і було головною ціллю під час розробки системи.

Аналог даної веб-системи, яка б могла поєднати ключові функцій або хоча б частково схожу, на жаль, знайти не вдалося. За допомогою розробленої веб-системи можна з легкістю визначити якість навичок та вмінь потенційного кандидата. Також не буде жодних сумнівів в доброчесності кандидата так як співбесіда буде онлайн з відео та аудіо зв'язком. Система поєднала саме ці функції для того щоб кандидат та рекрутер могли швидко та зручно провести співбесіду, не звертаючи увагу на технічні казуси.

1.1 Можливості використання системи Skype

Skype - це в своєму роді унікальна та безкоштовна система, одна з перших надала можливість користувачам з різних куточків світу підтримувати зв'язок онлайн використовуючи досить зручний програмний продукт, а зараз навіть просто використовуючи веб-систему. Популярність системи Skype дуже стрімко зростає протягом останніх років через пандемію у світі. Людям просто стає необхідно зв'язуватися онлайн та спілкуватися з родичами, так як це досить зручно та безкоштовно. Система Skype провідна, а тому і досить популярна у своєму сегменті.

Як правило, Skype люди використовують у різних цілях, найчастіше для того, щоб підтримувати зв'язок з друзями та родичами, так як система досить комфортна та інтуїтивно зрозуміла, не встановлює лімітів при спілкуванні. На рисунку 1.1 представлено приклад використання системи Skype.

Рисунок 1.1 — Приклад використання Skype

Не дивним є факт, що IT сектор останнім часом набагато частіше почав використовувати Skype в своїх цілях. Дуже часто різні співбесіди з кандидатами, зустрічі з клієнтами компанії, технічні зустрічі, ранкові мітинги або ретроспективи проходять у Skype. Це й не дивно, інтерфейсу системи Skype вистачає, щоб обговорити технічні моменти чи провести з колегами зустріч, система на цьому і спеціалізується. У випадку якщо потрібно провести технічну співбесіду та перевірити вміння кандидата писати код, то тут виникають проблеми.

Система не має змоги інтегрувати таку можливість, тому що вона не спеціалізується на проведення технічної співбесіди з написанням коду.

1.2 Можливості та використання CodeSandbox

CodeSandbox — це досить відома у IT веб-система, яка надає змогу для програмістів писати код та запускати код. Завдяки зручному та комфортному, кожного дня створюються нові програми та пишуть тисячі нових файлів коду.

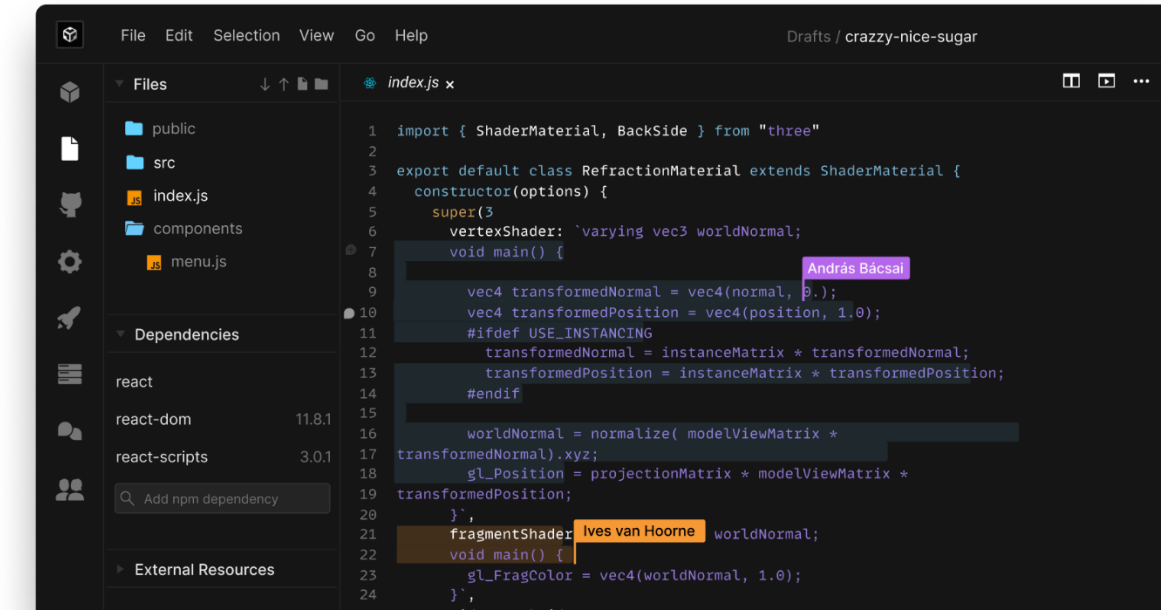

Рисунок 1.2 — Приклад використання CodeSandbox

CodeSandbox хоч і має платні підписки, але дуже велика кількість функцій доступна безкоштовно. Система підтримує різні мови програмування, інтегрується з різними сервісами наприклад GitHub, можна використовувати парне програмування, що дуже зручно при співбесіді, та підтримує ще багато зручних корисних функцій.

Рисунок 1.3 — Логотип CodeSandbox

Спочатку можна навіть подумати CodeSandbox спеціалізується на проведенні співбесід. Як правило, технічні спеціалісти досить часто кажуть запустити код в редакторі цієї системи, оптимізувати його або виявити помилки чи недоліки.

Але у системі є деякі недосконалості, що приводять до незручності проведення співбесіди. Рекрутер і кандидат повинні спілкуватися в одній системі, а працювати з кодом зовсім в іншій. На жаль, CodeSandbox не підтримує чат, відео або аудіо зв'язок між користувачами, хоча це могло б бути досить ефективно і зручно. Тому користувачі вимушені спілкуватися через додаткову систему, так як без неї провести співбесіду буде просто неможливо.

Система не має змоги інтегрувати цю можливість, тому що вона також не спеціалізується на проведенні технічних співбесід з онлайн спілкуванням.

1.3 Аналіз та опис предметної області

Головною ціллю даної системи є спрощення, зручність та комфорт проходження першого і другого етапу співбесіді для обох сторін.

Як правило така веб-система створюються на основі двох підсистем що виконують свої різні функції для належної роботи системи в цілому. Перша підсистема це клієнт - простими словами те що бачить користувач, комфортний і зрозумілий інтерфейс який підтримує зв'язок з сервером використовуючи протоколи зв'язку. Сервер в свою чергу це підсистема що отримує та обробляє запити від клієнта, ідентифікує користувачів, виконує код, що написав користувач і перенаправляє дані між клієнтами онлайн.

1.4 Функціональні особливості системи та їх аналіз

Розроблена веб-система потрібно щоб мала підтримку двох типів користувачів, що можуть працювати з нею: потенційний кандидат(користувач) та рекрутер або технічний спеціаліст(адміністратор):

Кандидат повинен мати змогу отримати доступ до ключових функції веб-системи:

— Сторінка списку кімнат, переглядати і шукати кімнату для співбесіди;

— Налаштування фільтру для кімнат;

— Сторінка опису та інформації веб-системи;

— Головна сторінка кімнати, а саме:

- Редактор коду у режимі онлайн чи офлайн;
- Термінал або консоль;
- Список користувачі, що онлайн;
- Онлайн чат;
- Відео, аудіо чат;

Головне це те щоб кандидат мав змогу легко авторизуватися а спочатку і зареєструватися у веб-системі, подивитись інформацію про вже створені кімнати для співбесіди, заходити і виходити у кімнати, писати та запускати код, використовувати онлайн чат, відео та аудіо чат.

Рекрутер або технічний спеціаліст повинен мати доступ до розширеного функціоналу веб-системи:

— Сторінка реєстрації та авторизації, а також автентифікації у системі;
— Сторінка списку кімнат, переглядати і шукати кімнату для співбесіди;
— Налаштування фільтру для кімнат;
— Створення та видалення кімнати;
— Налаштування параметрів кімнати:
  - Вибір мови програмування кімнати;
  - Пароль доступу до кімнати;
  - Максимальна допустима кількість користувачів та інше;
— Сторінка опису та інформації веб-системи;
— Головна сторінка кімнати, а саме:
  - Редактор коду у режимі онлайн чи офлайн;
  - Термінал або консоль;
  - Список користувачі що онлайн;
  - Онлайн чат;
  - Відео, аудіо чат;

Рекрутер повинен досить легко, а головне швидко спочатку зареєструватися, а потім зручно авторизуватися у системі. Рекрутер в основному повинен мати доступ до такого ж функціоналу як і у кандидата, але крім цього повинен мати доступ і до розширених функцій, а саме: видалити, створити і змінити налаштування кімнати для того щоб комфортніше провести співбесіду.

Веб-система адаптована для використання тільки за допомогою персональних комп'ютерів, під мобільні пристрої система також адаптована, але вона не передбачає використання за допомогою мобільних приладів.

Висновки до розділу

Системи які можуть частково бути аналогом розробленої веб-системи розглянуто у цьому розділі. Але на жаль ці системи не спеціалізується під проведення технічного інтерв'ю.

Крім цього розглянуто деякі деталі і особливості веб-системи які потрібно реалізувати для того щоб вона могла належно виконувати свої головні функції.

Системи, що розглянуто у цьому розділі, поодинці досить зручні та ефективні та мають багато переваг, такі як:

— Сучасність та комфорт використання, сильна підтримка провідних спеціалістів;
— Безкоштовне використання системи, завдяки чому можна отримати доступ до більшості ключових функцій системи;
— При онлайн використанні міцний зв'язок;

Але у цих системах присутні і суттєві недоліки:

— Системи не спеціалізуються під проведення технічних співбесід;
— Не підтримують інтеграцій або розширень що могли б надати відсутні функцій;

2 ОПИС ЗАСОБІВ РОЗРОБКИ

Для виконання поставлених задач необхідно потужний набір інструментів, який відповідає сьогоденню і підтримує необхідний функціонал. Веб-застосунок, повинен працювати швидко і без затримок, адже звичайного користувача цікавить, зручність і надійність. Проаналізувавши різні мови програмування, було вирішено обрати одну з найвідоміших і широко використаних мов програмування для веб-сервісів, а саме JavaScript. Головною перевагою є наявність великої кількості підтримуваних бібліотек.

Середовище програмування було використано Visual Studio Code.

Система для проведення інтерв'ю умовно складається з двох частин, а саме програмно-апаратної частини, де реалізована вся логіка програми і клієнтської сторони з користувацьким інтерфейсом.

2.1 Середовище програмування Visual Studio Code

Visual Studio Code — це розроблений компанією Microsoft у 2015 році редактор коду, для створення програмного забезпечення та інших інструментів. Даний редактор підтримує ряд особливостей, таких як:

- налагодження коду, користувач має змогу запустити своє програмне забезпечення в режимі пошуку помилок і шаг за шагом пройтися по кожній строчці коду. Такий функціонал дає змогу виявити помилки і швидко їх виправити.
- переробка коду, дозволяє програмісту, при написанні програмного забезпечення, дотримуватися певних загальноприйнятих правил написання коду, здійснювати реструктуризацію коду;

— підсвічування синтаксису, для зменшення напруги на очі і зручності в читанні коду;

— вбудовану систему контролю версій Git, що дає швидко і ефективно її використовувати без додаткових складних налаштувань;

Таким чином ми бачимо ряд переваг, які і зробили дане середовище популярним у використанні по всьому світу.

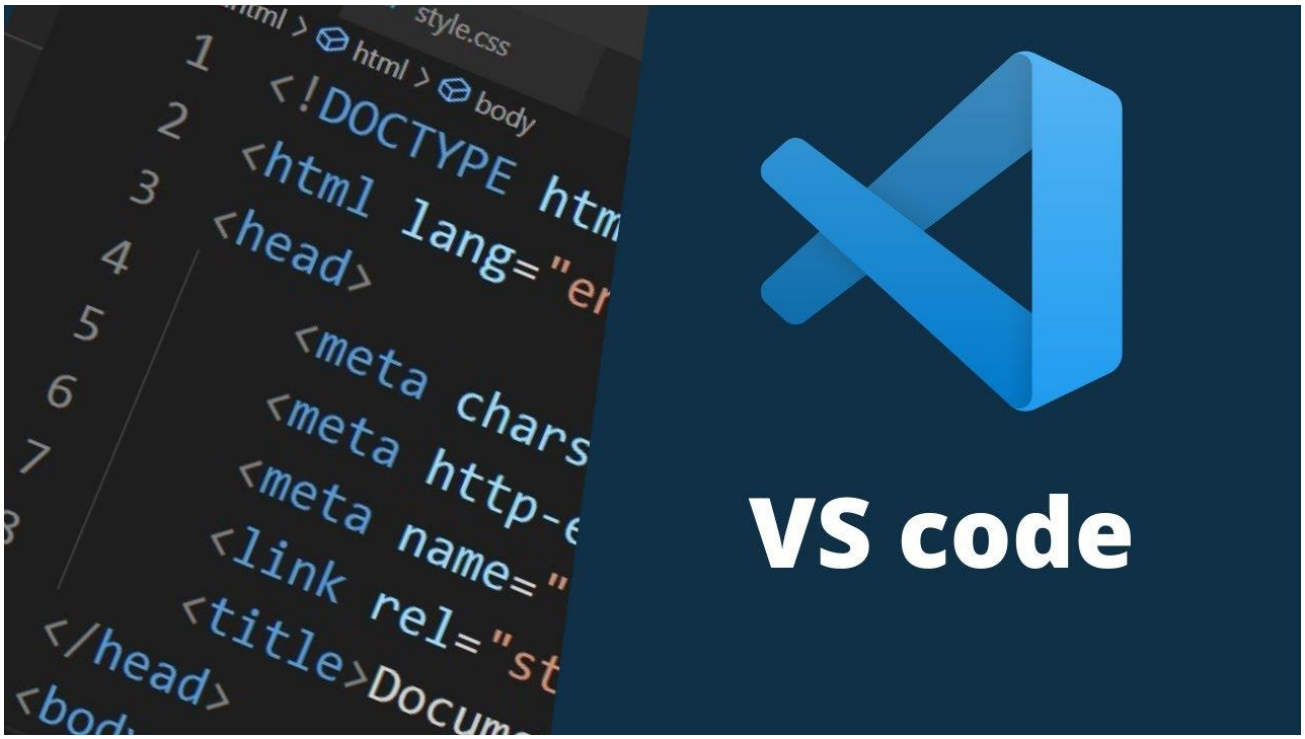

Рисунок 2.1 — Логотип Visual Studio Code

Головною особливістю є те що фірма розробник постійно оновлює даний програмний продукт, тож користувач має можливість використовувати найсучасніші інструменти в світі розробки. Також забезпечується підтримка таких мов програмування як Java, JavaScript, Go, Node.js, Python, C++, Fortran.

Даний продукт дає змогу розробити програми з графічним інтерфейсом, веб-сайти, веб-застосунки

2.2 Мови програмування JavaScript, HTML, CSS

Для створення веб-системи для проведення інтерв'ю в дистанційному форматі було використано, такі засоби як HTML, CSS та мова програмування JavaScript.

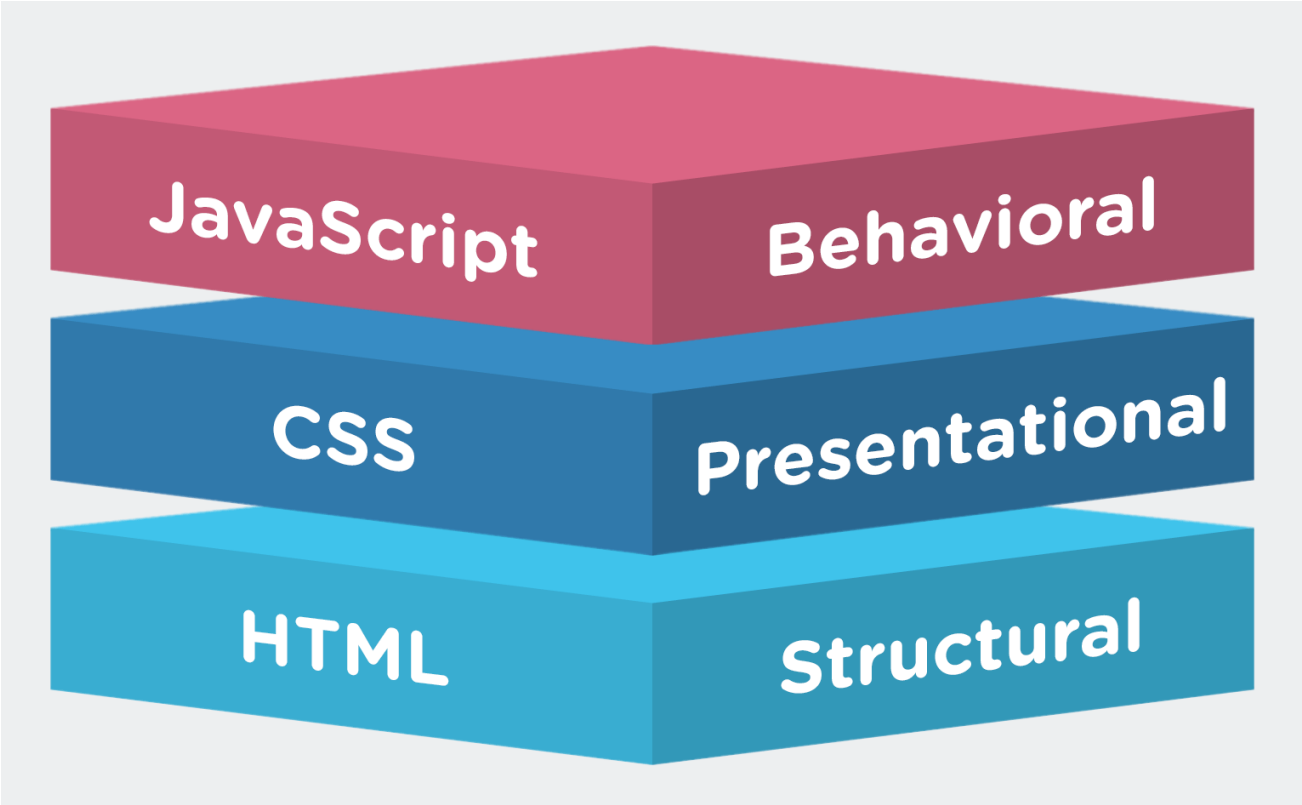

Рисунок 2.2 — Функції JavaScript, CSS та HTML

HTML — це мова тегів, за допомогою яких створюється базова розмітка, структура веб-сторінок, веб-додатків. Сама назва це абревіатура, яка розшифровується як Hyper Text Markup Language, тобто з англійської означає мова розмітки гіпертексту.

Також програміст використовуючи HTML, набирає теги в окремий текстовий файл з розширенням html. А вже при запуску даного файлу через браузер відбувається інтерпретація тексту і відображення тієї чи іншої структури на веб-сторінці.

CSS — каскадна мова стилів, яка дозволяє створити зовнішній вигляд веб-сторінки, прописавши шрифти, колір та розмір тексту, розмістивши у певному порядку структурні елементи. Іншими словами це не від'ємна частина веб сайту, яка забезпечує гарний і зручний вигляд веб-сайту. Більше того, за допомогою CSS можна створити адаптивність сайту, адже важливо, щоб застосунок був пристосований, до різних типів приладів.

JavaScript — це мова програмування, яка забезпечує інтерактивність веб-сторінки. Порівняно з HTML і CSS це найбільш складна частина, адже саме вона забезпечує логіку і взаємодію всіх компонентів на стороні клієнта і стороні сервера.

Таким чином поєднавши всі ці мови програмування ми поетапно розробили веб-сайт для дистанційного проведення інтерв'ю. Тобто спочатку ми за допомогою HTML створили базову розмітку, потім надали певного стилю відповідно до вимог з використанням CSS і реалізували логіку.

2.3 Framework React для створення інтерфейсів користувача

Особливістю мови програмування JavaScript є те, що в ній міститься великий набір бібліотек. Кожна така бібліотека, робить написання коду більш зручним і швидким, адже в них містяться вже заздалегідь заготовленні шаблони, які не потрібно реалізовувати заново, а потрібно просто брати їх і використовувати.

React — це бібліотека JavaScript, яка розроблена для створення інтерфейсів користувача. Також вона є дуже гнучкою, ефективною і в той же час декларативною. Також особливостями є одностороння передача даних, віртуальна об'єктна модель документа.

Більшість програмістів використовують дану бібліотеку, щоб скомпонувати складні інтерфейси з більш простих, тобто поєднати їх.

2.4 Бібліотека Redux

Redux — це бібліотека JavaScript з відкритим програмним кодом, яка використовується для управління станом програми. Часто дану бібліотеку застосовують разом з React. Розробниками є Ден Абрамов та Ендрю Кларк, які створили Redux у 2015 році.

Рисунок 2.3 —Бібліотека Redux

Даний продукт є своєрідним контейнером станів для додатків в JavaScript. За допомогою нього більшість розробників оптимізують свій код. Також Redux доповнює бібліотеку React, адже він зберігає стан всього додатку в дереві об’єктів в одному сховищі. В свою чергу дані передаються від батьківського до дочірнього компонента. Це дозволяє швидко і ефективно зробити налагодження або перевірку програми.

Іншою основною концепцією є те, що стан призначений тільки для читання, в свою чергу це гарантує незмінність.

Доволі часто застосовують Redux коли потрібно реалізувати стан кешування сторінки, так наприклад, користувач будучі на одній сторінці вирішив перейти на іншу, а потім повернувся назад і в цей момент важливо, щоб він повернувся в той же стан сторінки, який був і до цього.

2.5 Використання і основні принципи роботи веб-сокетів

WebSocket — це спеціальний протокол обмінну даними між сервером і браузером за допомогою постійного зв’язку. Дана технологія є унікальною, адже дозволяє працювати з двох направленим потоком даних.

Один з найпопулярніших методів взаємодії між сервером і клієнтом це HTTP запроси. Тобто за допомогою спеціальних запитів (GET, POST, PUT, DELETE) відбувається взаємодія з сервером, для отримання даних. Така комунікація має ряд переваг проте є недосконалою і в конкретних випадках є незручною з великою кількістю недоліків.

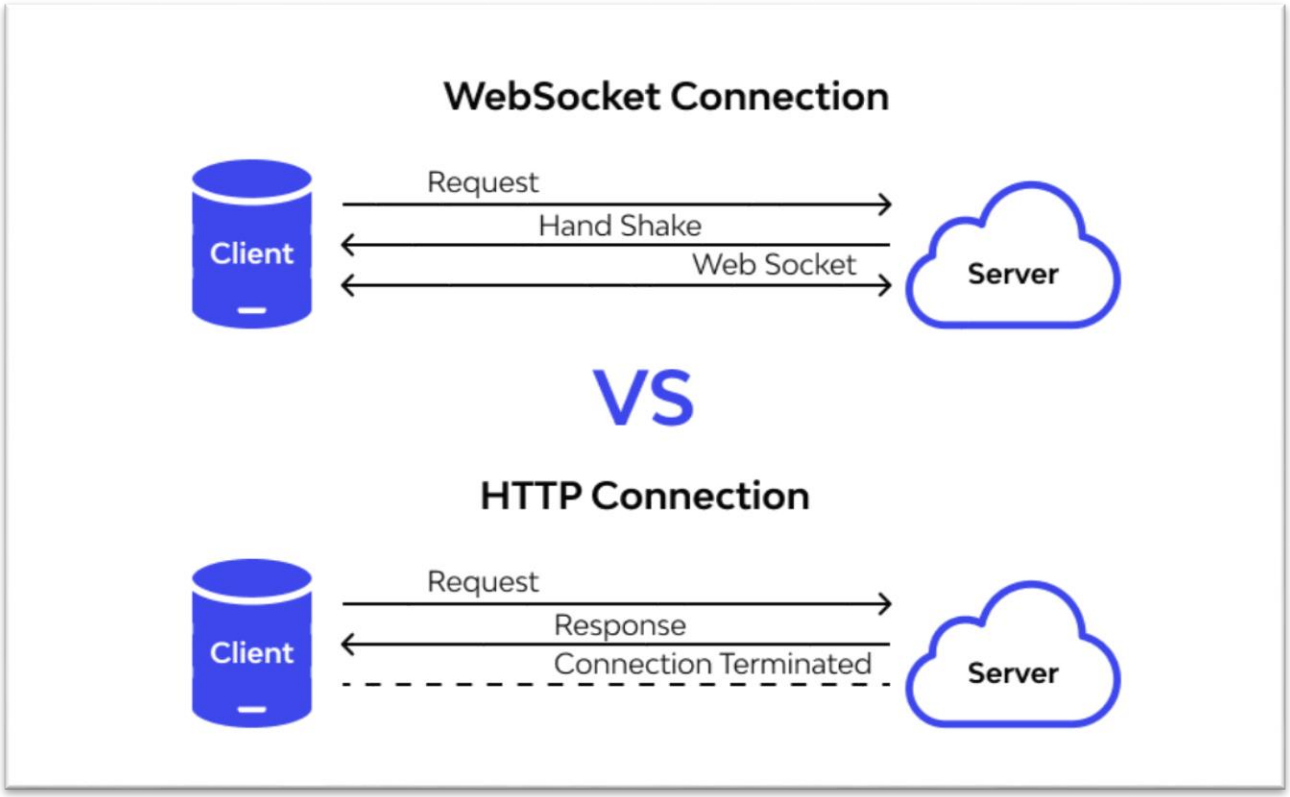

Рисунок 2.4 —Порівняння принципів роботи WebSocket і HTTP

На відміну від HTTP запитів, веб-сокети працюють по зовсім іншій схемі, адже не варто надсилати кожного разу робити запит для отримання відповіді, в

той же час потрібно просто прослуховувати сервер. Таким чином спочатку встановлюється з’єднання , а далі відбувається відслідковування повідомлень.

Дана технологія є популярною, коли потрібно розробити застосунок, який в режимі реального часу буде отримувати певні дані і взаємодіяти з ними, також широко застосовується для створення чатів для спілкування і в ігровій сфері.

2.6 Система контролю версій Git

В ході написання програмного продукту я використовував розподілену систему контролю версіями. Дана технологія є унікальною, адже надає спеціальні можливості розробнику змінювати та зберігати різні версії проекту.

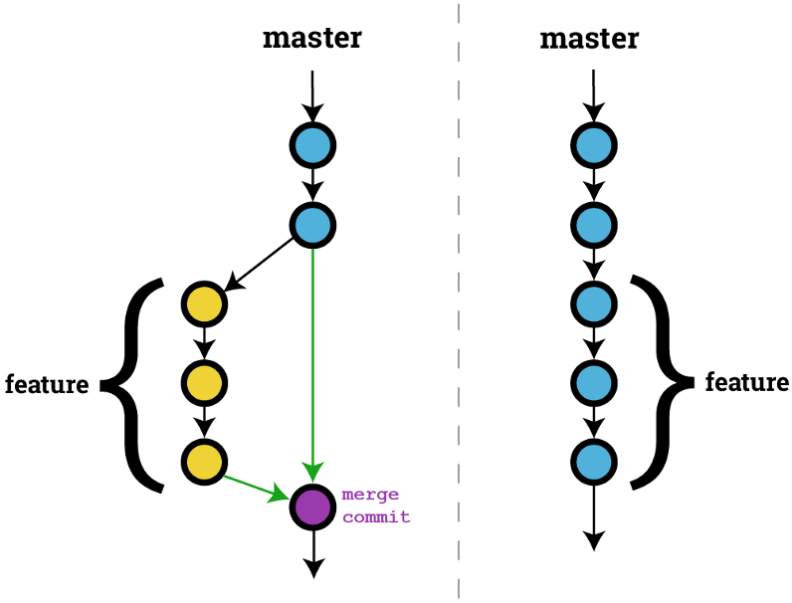

Рисунок 2.5 — Створення гілки в Git

Git дає змогу створити стани системи і зберегти його у вигляді комітів. Також в мене було створена головна гілка, в якій зберігався робочий код і від нього я створював додаткові, де експериментував з певними доопрацюванням.

2.7 Середовище виконання Node.js

Node.js — це програмна платформа, для створення і виконання коду на мові програмування JavaScript. Дана середа виконання побудована на основі двигуна JavaScript Chrome V8 і дозволяє транслювати написаний розробником код в машинний, тобто в низькорівневий код, який може комп'ютер може запускати без додаткової обробки.

Node.js забезпечує весь необхідний функціонал і можливості для створення серверної частини для веб-застосунків. Також варто зазначити, що дане середовище виконання має відкритий доступ до початкового коду і в разі потреби можна його переглянути на GitHub.

Для використання Node.js можна використовувати будь який текстовий редактор, що дає розробнику право вибору. Адже при необхідності Notepad++ чи звичайний блокнот може бути використаний. В моєму випадку я використовував середу програмування Visual Studio Code, який надає зручний функціонал для створення веб-застосунків.

Іншою перевагою на користь Node.js є те що дана платформа використовує неблокуючу, керуючу подіями, модель введення-виводу. В основу даної моделі покладено можливість створювати запроси паралельно і сервер буде їх оброблювати. Такий підхід є дуже зручним і практичним, фактично використання багато поточності є непотрібним.

Варто звернути увагу і на Node модулі, іншими словами це блоки або частини коду, які можна використовувати багаторазово. За замовчуванням, великий вибір модулів установлюється заздалегідь і його можна відразу використовувати. Проте в разі потреби можна написати свої блоки і використовувати у різних частинах проекту.

Бібліотека npm є надзвичайно великою і в ній зібранні певні шаблони і рішення з багатьох проектів і саме це надає певної універсальності, адже не потрібно придумувати нові алгоритми, якщо вони були реалізовані раніше.

2.8 Формат обміну даними JSON

Для обміну даними між серверною частиною і клієнтською, був використаний формат обміну даними JSON, який розшифровується як JavaScript Object Notation. Даний формат є унікальним і надзвичайно простий у використанні і в той же час ефективним.

Офіційно розробником вважається Дуглас Крокфорд, який створив і популяризував його на початку 2000 років.

Популярність даному формати забезпечила його структура, яка є інтуїтивно зрозумілою для розробника і звичайної людини. Правила синтаксису:

— Дані завжди зберігаються у в парах ключ-значення;

— Об’єкти завжди відокремлюються комою;

— Масиви відокремлюються квадратними дужками;

Розглянемо на рисунку 2.6 JSON і його структури.

Рисунок 2.6 — Приклад JSON

Отже, даний формат обміну даними ідеально підійшов для взаємодії серверної частини і користувацький. Тобто спочатку формується запит від користувача на виконання певної дії, наприклад реєстрації, всі дані збираються в

JSON за принципом ключ-значення і відправляються на сервер де відбувається перевірка отриманих даних і відповідна певно обробка.

Висновки до розділу

В даному розділі були розглянуті всі ключові системи, веб-сервіси, інструменти, які використовувалися для створення веб-застосунку.

Поєднавши всі найновітніші технології, які доступні на даний момент було розроблено програмний продукт, який є універсальним і сумісним з різними операційними системами і браузерами. Також кожна бібліотека чи технологія використана мною робила процес розробки більш ефективним і зручним.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Веб-система була розроблена для того щоб користувачі мали змогу дуже якісно швидко та зручно проводити різного роду співбесіди чи інтерв'ю. Для реалізації таких можливостей, у системі було розроблено велика кількість різноманітних та корисних функцій, а саме підтримку сучасних та популярних мов програмування, а також онлайн чатів, як відео так і аудіо, комфортний клієнтський інтерфейс, та різні рівні допуску до веб-системи в цілому.

Під час розробки веб-системи було використано велику кількість новітніх технологій, що у свою чергу досить часто використовуються у різного роду веб проектах та системах, нерідко навіть у комерційних з різними доробками. Також ми намагалися притримуватися різних стандартів написання коду та принципів програмування, які займають сьогодні ключові місця у розробці клієнт серверних застосунків. Такі принципи як YAGNI, DRY, KISS, SOLID нерідко використовувалися у проекті у різних випадках, де можна було це зробити не порушуючи основної структури проекту.

3.1 Структура проекту

Веб-система була побудована на основі клієнт-серверного застосунку, де компоненти взаємодіють між собою за допомогою REST запитів. Нижче наведена клієнтська частина проекту.

Переглянувши спроектовану структуру проекту, можна з легкістю визначити за що відповідає кожна папка чи компонент у проекті. Перш за все це дві головні папки, де зберігається серверна частина - папка Server, та клієнтська - папка src. Крім цього у проекті є папка node_modules де зберігаються основні фреймворки, модулі чи бібліотеки, які були використані під час розробки.

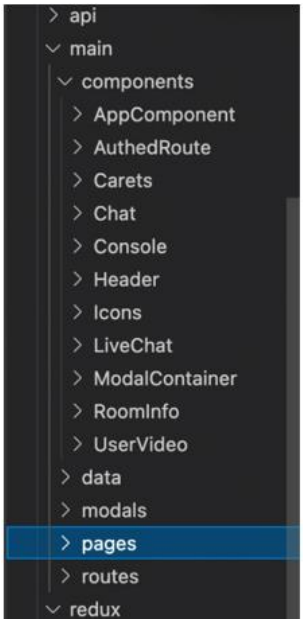

Рисунок 3.1 — Приклад клієнтської частини проекту

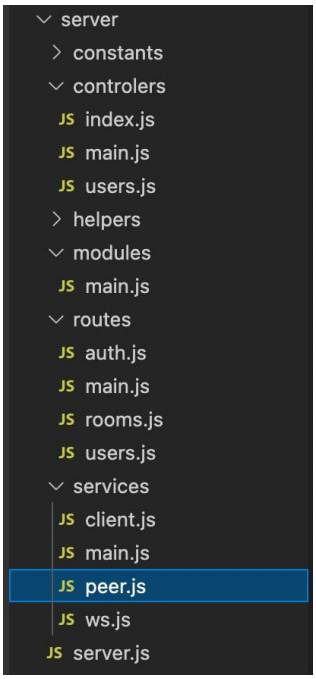

Рисунок 3.2 — Приклад серверної частини проекту

В папці “client” описана клієнтська частина, а саме компоненти, сервіси для взаємодії з сервером, різні сторінки застосунку, модальні вікна, функції та менеджер станів.

В свою чергу, папка “server” містить інший досить важливий функціонал клієнт-серверного застосунку. У цій папці описані, контролери та модулі, різного роду сервіси що обробляють дані та маршрутизатори, щоб якісно обробляти запити що надходять від клієнта, та багато іншого не менш важливого серверного функціоналу.

Можна побачити що, структура була розроблена досить продумано та якісно, тому проект досить легко підтримується і не потребує великої кількості людей, легко масштабується та може бути дуже швидко інтегрований, як підсистема в інші веб застосунки. Це надає застосунку певної універсальності, адже його можна буде вдосконалювати.

3.2 Клієнтська частина проекту

Для розробки клієнтської частини веб-системи було використано досить новий фреймворк React. Цей фреймворк зовсім нещодавно отримав сильне оновлення та перехід до функціонального стилю програмування, підтримку хуків та ще досить багато функцій для розробки веб-застосунків.

Фреймворк досить сучасний, підтримує велику кількість бібліотек, різні стилі та принципи програмування. Також за допомогою цього фреймворку, а саме функції зміни життєвого циклу компонентів та мемоізацій, що надає цей фреймворк, можна проводити різні маніпуляції з DOM, дуже точно змінювати та оновлювати його, таким чином що користувач навіть не помітить цього. У нашому випадку це був пріоритет при виборі засобів розробки, адже коли відбувається трансляція відео чи аудіо між користувачами не повинно бути затримок, або втрачених кадрів. За допомогою React, нам вдалося досить сильно оптимізувати розроблену веб-систему таким чином що у користувачів буде можливість ефективно взаємодіяти з нею навіть на досить застарілому обладнанню.

Для того, щоб ефективно взаємодіяти з сервером, а саме вірно відправляти та отримувати запити, на клієнті було використано різні бібліотеки, технології та підходи програмування. Для такої роботи, вкрай потрібно було запрограмувати різного роду сервіси. Такі сервіси були описані у пункті 3.4.

Як правило, при роботі з веб-системами користувачі отримують, формують, оновлюють, або набувають різного роду дані, які звичайно ж потрібно десь зберігати в одному місці, та у випадку необхідності отримати у різних компонентах системи. Спеціально для такого випадку і був використаний менеджер станів Redux. Це досить відома бібліотека для зберігання та обробки даних, що може використовуватися з різними фреймворками. У нашому випадку ми використовуємо React, тому потрібно було використовувати свого роду адаптер React-Redux, що використовується для ефективного об'єднанню цих бібліотек, так як ми використовували функціональний підхід то, за допомогою хуків useDispatch, useSelector, та інших . Сам Redux не підтримує асинхроність, тому саме для цього використовувалися Redux-Saga та різні ефекти цієї бібліотеки.

Для приємного вигляду веб-системи, використовувалися велика кількість бібліотек та різні технології програмування. В основному використовувався препроцесор SCSS та технологія BEM, але також потрібно згадати про Styled-components, Material UI, Ant Design, що нерідко допомагали швидко написати стилі або додати вже компоненти зі стилями. Використовуючи такі підходи проект має чудово написані стилі які можна додавати не переживаючи, що вони будуть десь перезаписані.

Для ефективного переміщення між сторінками веб-системи використовувалася бібліотека з також функціональним стилем програмування - React-Router. Яка нещодавно була оновлена, та добавлено декілька нових хуків що були нами і використанні.

3.3 Серверна частина проекту

Серверна частина розробленого додатку була написана за допомогою Node.js. Платформи, яка основана на двигуні JavaScript V8 та дозволяє будувати саме високопродуктивні серверні додатки, а також з використання відомого та провіреного часом програмного каркасу Express.js. Завдяки такому вибору технологій клієнтська та серверна частини розробленої системи використовують одну мову програмування - JavaScript. Що дуже сильно полегшує наступну підтримку та розробку проекту та не потребує великої кількості спеціалістів різного типу.

Перш за все,Express.js був використаний для створення якісної маршрутизації розробленої системи, було добавлено велика кількість ендпоінтів, починаючи від автентифікації клієнта та закінчуючи надсиланням унікальних даних. Також були добавленні ендпоінти для створення, редагування та видалення кімнати, автентифікації, як адміністратор, надсилання даних кімнати та інше. Базові запроси використовують звичайний протокол передачі даних а саме HTTP та різні його методи такі як GET, POST, PUT, DELETE.

Але крім цього у застосунку також потрібно було використовувати інші протоколи наприклад для взаємодії Websockets. Спеціально для цього на сервері було розроблено сервіс для обробки запитів від клієнта, починаючи від встановлення зв'язку між сокетом, закінчуючи розривами зв'язку та.

Для самої взаємодії між веб-сокетами було використано бібліотеку ‘ws’. Також на серверній стороні було розроблено алгоритм кімнат, а саме приєднання та вихід з кімнати та повідомлення різними сповіщення клієнтів кімнати.

Для встановлення аудіо та відео зв’язку між клієнтами було використано допоміжну бібліотеку peer.js, яка досить схожа на протокол передачі даних веб-сокетами. Саме ця бібліотека дозволяє створювати peer-to-peer зв'язок між користувачами та допомагає передавати відео та аудіо дані між ними не

втрачаючи їх. А з використанням вже розробленого алгоритму кімнат, це стає досить ефективним та чудовим вирішенням стійкого онлайн зв’язку між клієнтам.

3.4 Розробка клієнтських сервісів для взаємодії з сервером

На клієнті, для того щоб підтримувати принципи програмування необхідно було зробити декілька сервісів, які будуть ефективно та якісно взаємодіяти з сервером не заважаючи основній роботі клієнта та не засмічуючи код викликами різних запитів до сервера. Крім цього це також потрібно було, щоб розмежувати бізнес та технічні частини проекту, для кращої та ефективнішої підтримки проекту у майбутньому, для обробки помилок, що можуть прийти з сервера, та для викидання запобіжників помилок у випадку коли їх неможливо опрацювати, та просто щоб не тривожити користувачів неочікуваною помилкою. На рисунку 3.3 показані клієнтські сервіси.

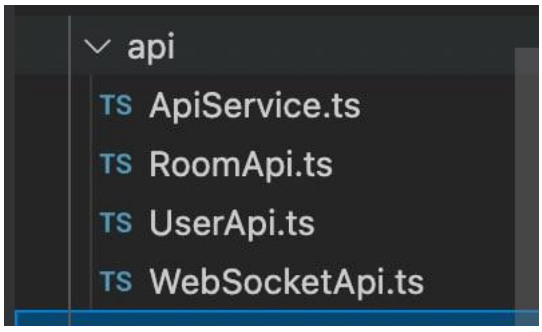

Рисунок 3.3 — Клієнтські сервіси

У системі були розроблені наступні сервіси UserService, WebsocketService, RoomsService, ApiService. Ім'я кожного було спеціально підібрано для детального опису роботи цього сервісу.

ApiService - ключовий сервіс для роботи з сервером, тут були описані базові моменти такі як обробка помилок та типізація методів запитів до сервера.

UserApi - це сервіс де описано базові запити авторизації користувача та клієнта, різні запити про надання інформації, вихід та створення акаунтів.

WebSocketsApi - в свою чергу був реалізований іншим методом та використовується для побудови неблокуючого канала за допомогою eventchanel із бібліотеки Saga. Це надалі допоможе легко взаємодіяти з веб-сокети сервера та між різними користувачами що знаходяться у кімнаті. Завдяки цьому сервісу був побудований онлайн чат, відео та аудіо чати. Цей сервіс був розроблений так, що за допомогою Saga він буде прослуховує сервер, на дії користувача та користувачів що під'єднані до кімнати. У випадку якщо станеться несподіваний розрив зв'язку, сервіс намагається відновити його декілька.

3.5 Розробка загального менеджера станів на основі Redux

В клієнтському застосунку для ефективного зберігання даних про користувача його налаштування, та іншу досить важливу інформацію, яку він може отримати при роботі з застосунком, було використано менеджер станів Redux. Redux - це свого роду досить простий але ефективний підхід у програмуванні що використовує ключові поняття: store, action, reducer.

Store - серце Redux, або головне сховище інформації, як правило сюди направляють дані, тут змінюють їх або видаляють, та звідси їх і отримують. Action - це як правило об'єкт у якого є два поля, одне необхідне - type, інше опциональне - payload, завдяки цьому обєкту, як правило, ми говоримо що ми хочемо змінити і яким чином. Reducer - це чиста функція, тобто функція що завжди при одних і тих самих повертає один і той самий результат, та не робить побічних ефектів. Завдяки цій функції, і змінюється наше сховище в залежності від типу дії.

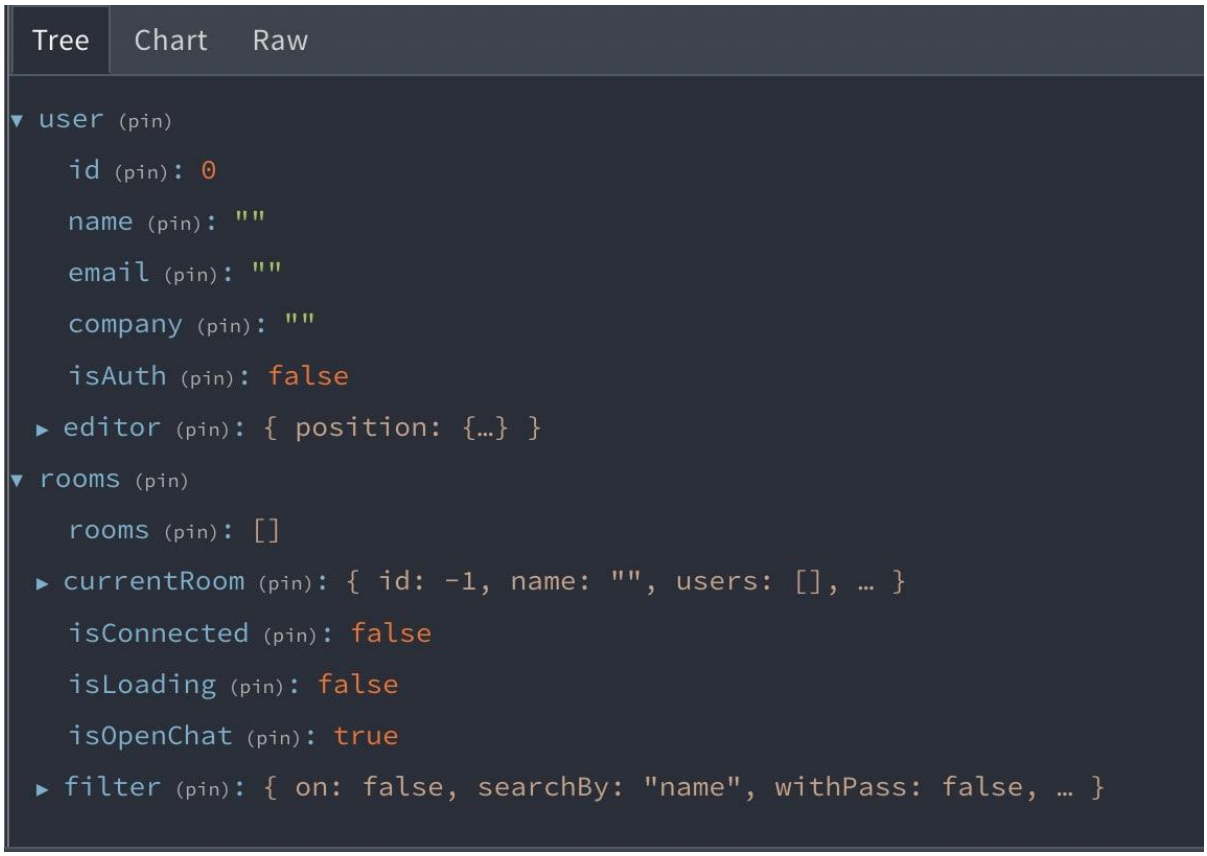

Рисунок 3.4 — Клієнтське сховище даних

На рисунку 3.5 наглядно показано як змінюється сховище в залежності від дії.

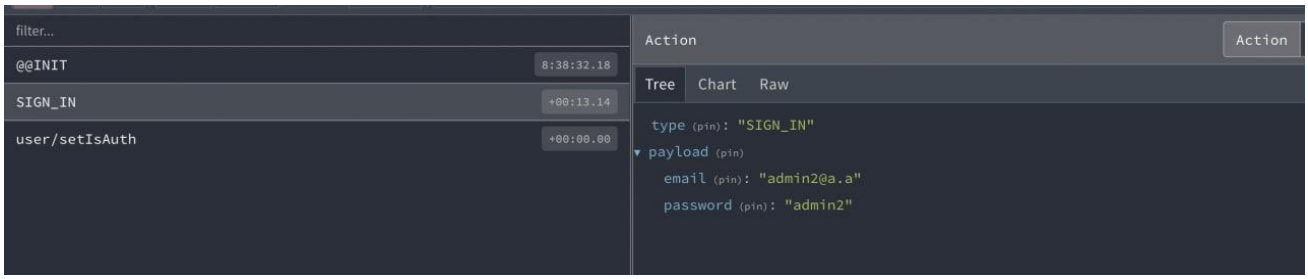

Рисунок 3.5 — Зміна даних у сховищі в залежності від дії

Сама технологія Redux не підтримує роботу з асинхронностью. Тому для асинхронної зміну даних у нашому сторі було використано ще один фреймворк Saga. Saga в свою чергу має велику кількість допоміжних ефектів для асинхронної

взаємодії з нашим сховище. Saga була вибрана не випадково, тому що саме вона підтримує гнучку та якісну взаємодію з Websockets та Redux.

Висновок до розділу

У розділі було представлено технології, методи за допомогою яких була побудований клієнт-серверний застосунок. Було описані фреймворки і паттерни за допомогою яких будувалася клієнтська та серверна частини. Були описанні сервіси та менеджери для обробки даних на клієнтській стороні та різного виду ендпоінти для відправки та обробки на серверній. Як вже описувалося вище в системі також використовувалися деякі з сучасних принципів програмування.

4 РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Веб-система для проведення інтерв'ю в дистанційному форматі, має простий і зрозумілий інтерфейс, адже потрібно, щоб користувач зміг з легкістю використовувати застосунок. Відповідно до прав доступу, буде доступний різний функціонал, адже адміністратор повинен мати можливості для створення, редагування та видалення кімнат.

4.1 Реєстрація користувача та вхід в систему

Для використання застосунку необхідно увійти в систему, якщо користувач має логін і пароль то потрібно їх просто ввести, проте передбачається, що новий користувач перш за все повинен зареєструватися.

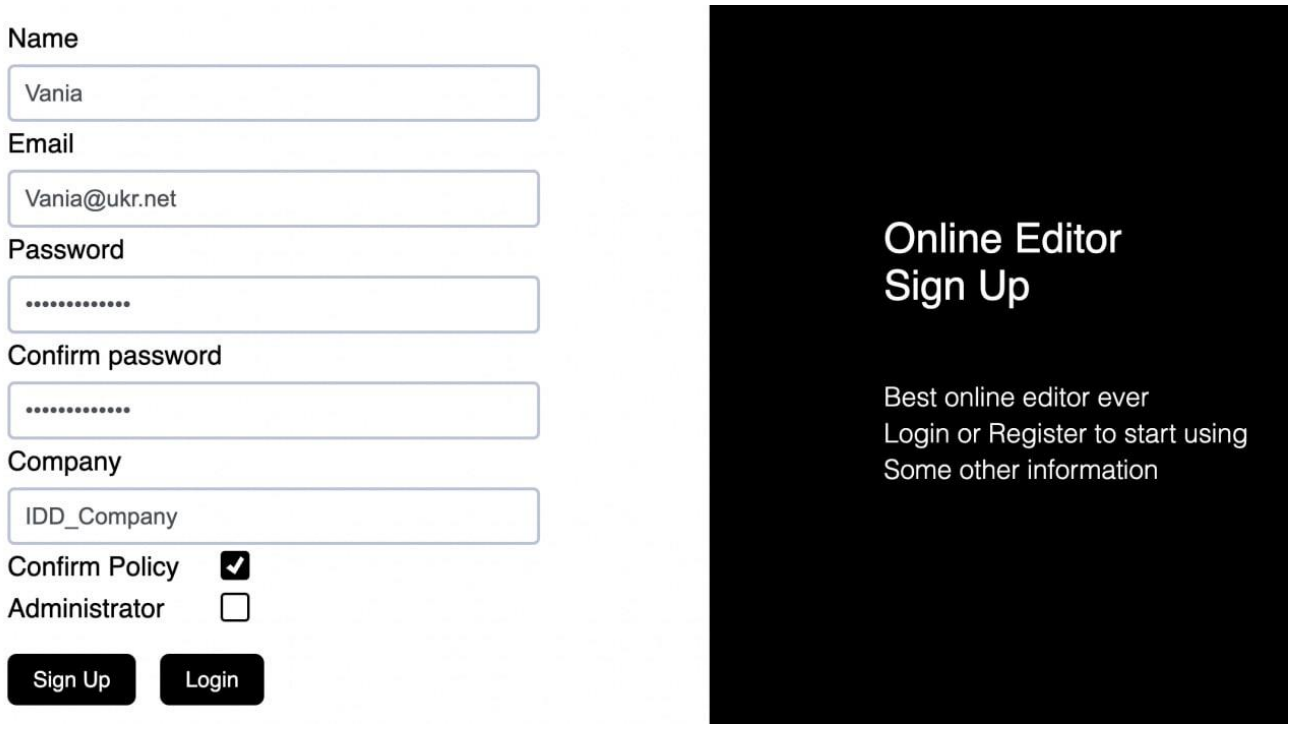

Рисунок 4.1 — Реєстрація користувача

Після ведення даних для реєстрація відбувається перевірка полів, адже важливо, щоб в систему була записана достовірна інформація про користувача. У випадку, якщо дані будуть не коректними то застосунок сповістить про помилку, як показано на рисунку 4.2.

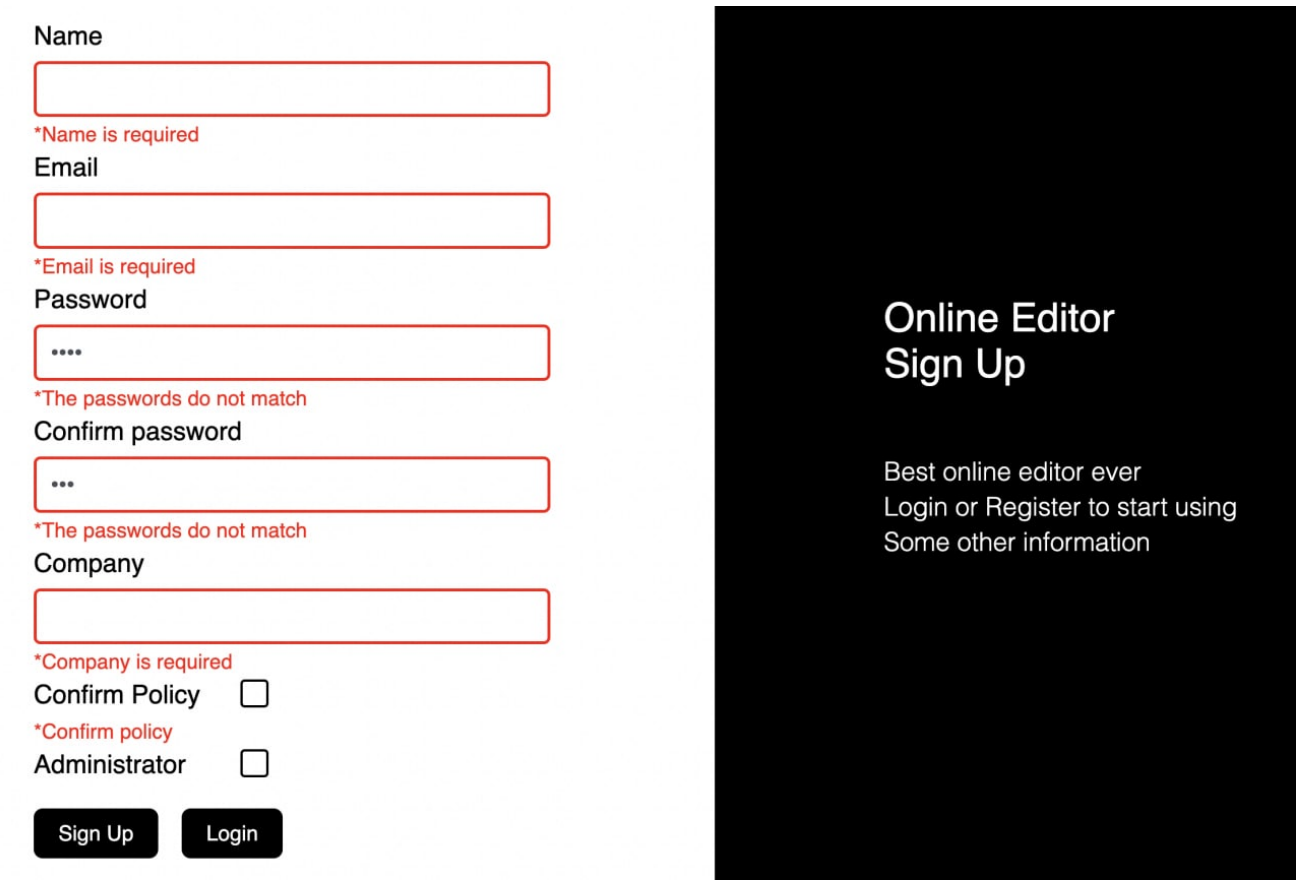

Рисунок 4.2 —Перевірка обов’язкових полів

Кожне поле під час реєстрації є обов’язковим, у випадку якщо воно не заповнене то висвітиться помилка, яка про це сповістить.

4.2 Інтерфейс початкової сторінки

Після реєстрації перед користувачем відкривається головна сторінка з кімнатами.

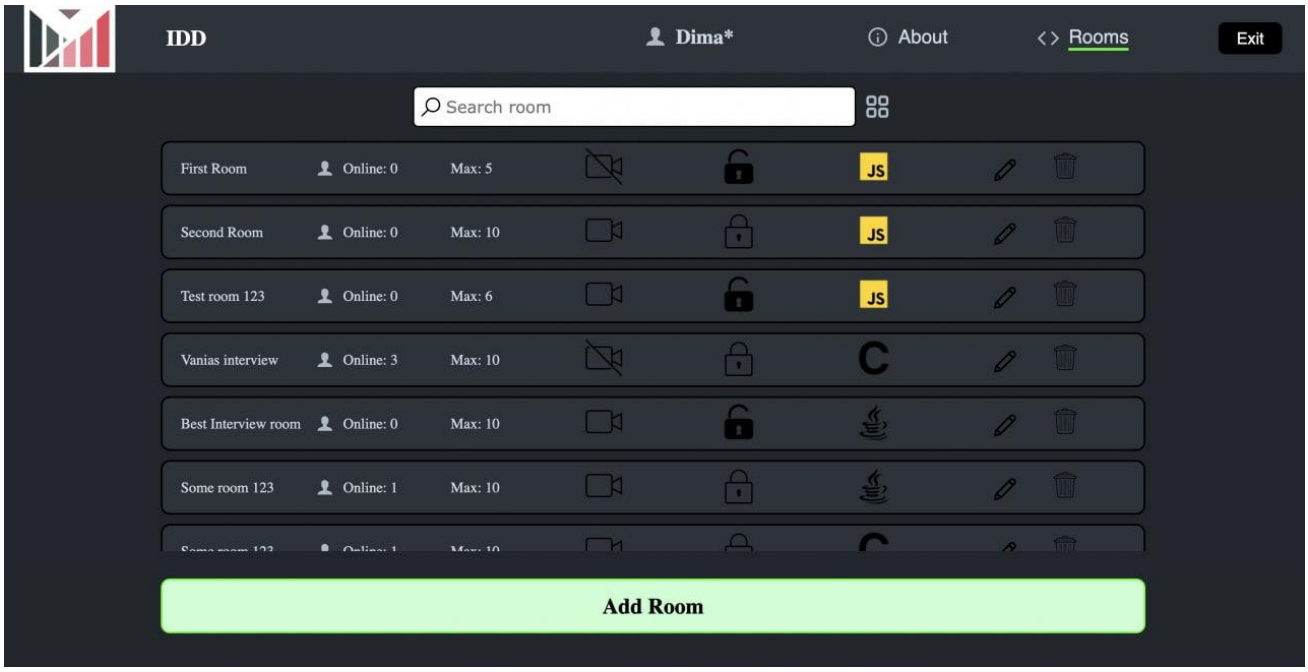

Рисунок 4.3 — Головна сторінка з кімнатами

Кожна створена кімната має свої характеристики, умовно їх можна розділити на дві групи, а саме з відкритим та закритим доступом. В першому випадку доступ до даного простору може отримати будь-який користувач і в системі відображається відкритий замочок як зображено на рисунку 4.3. Інший тип кімнат, це з обмеженим доступом, до потрібно мати унікальний пароль для входу в кімнату. Такий підхід дозоляє створити конфіденційні бесіди.

У випадку, якщо зайшов адміністратор сайту, то буде активна кнопка створення кімнати, де вводяться основні параметри. Наприклад не всі кімнати будуть підтримувати відео- зв'язок, відповідний знак буде відображатися, щоб користувач інтуїтивно міг зрозуміти, чи задовольняє його вимогам даний простір чи потрібно перейти до іншого.

Ще одним важливим компонентом є відображення інформації про максимальну кількість учасників, назви та число активних на даний момент користувачів.

Система підтримує пошук кімнат по назві або по унікальному ключу, дана можливість передбачає швидку взаємодію і навігацію по сторінці, адже не потрібно витрачати час на пошуки.

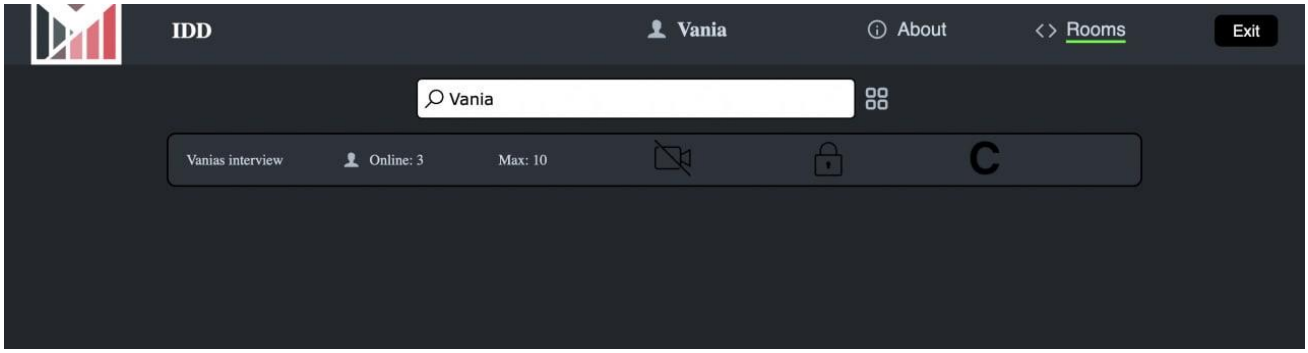

Рисунок 4.4 — Пошук за назвою

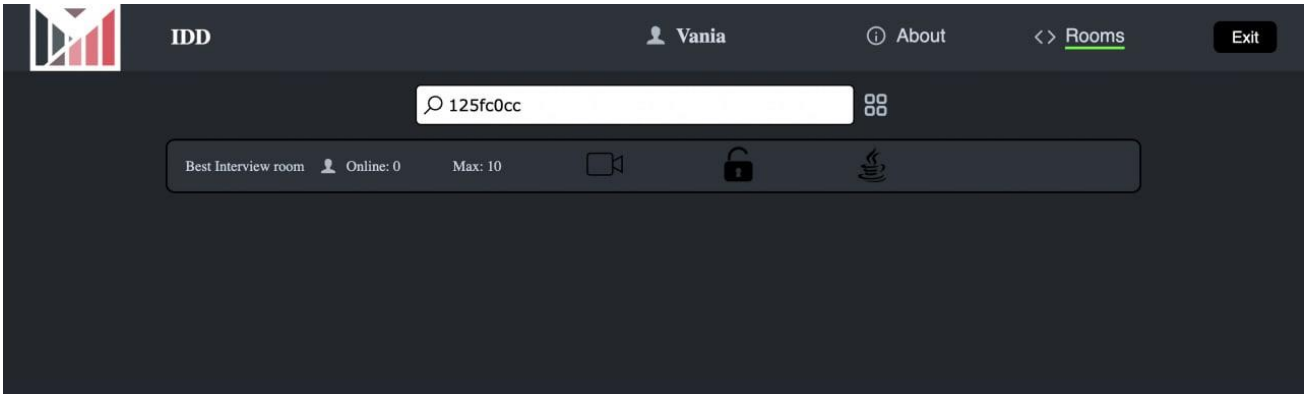

Рисунок 4.5 — Пошук за ідентифікатором

Також можна знаходити кімнати за унікальним ідентифікатором, для цього потрібно ввести значення в поле для пошуку. На рисунку 4.5 можна побачити результат роботи і одну знайдену кімнату, яка відповідає запиту.

Пошук за назвою або ідентифікатором зручний , проте в той же час не є універсальним і не покриває всіх потреб, тому було розроблено спеціальний фільтр пошуку. Передбачається, що коли кімнат буде величезна кількість то за допомогою даного функціоналу можна буде швидко і легко здійснювати навігацію і пошук

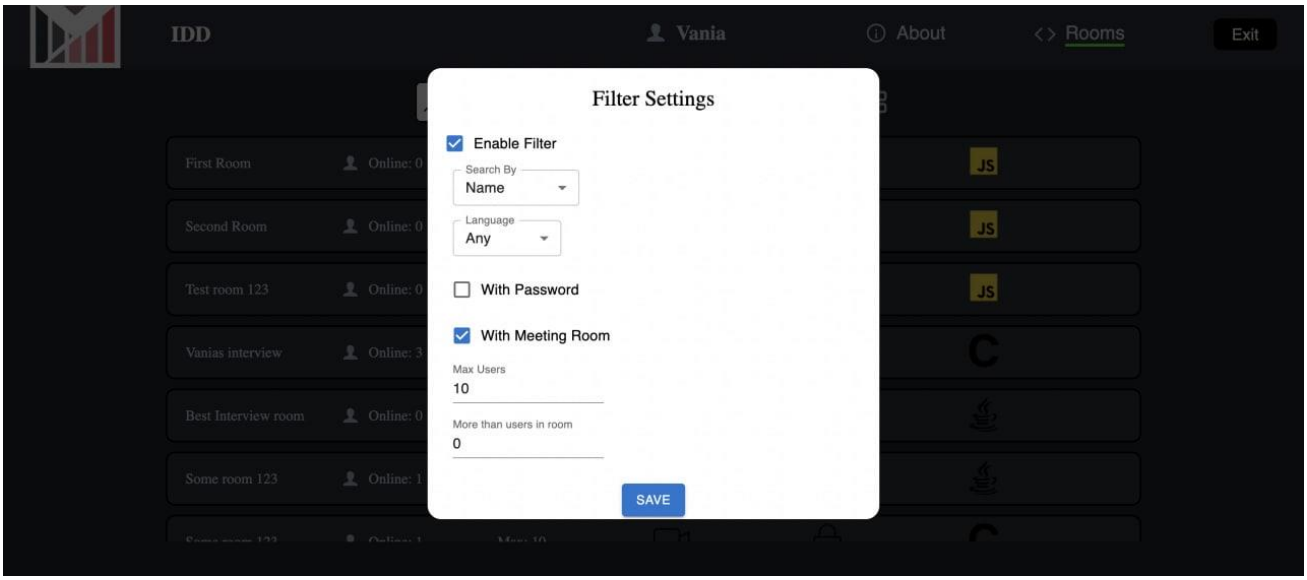

Рисунок 4.6 — Використання фільтрів

Фільтри дають змогу ввести параметри для пошуку кімнати з конкретними вимогами, а саме:

— мовою програмування:

— наявність відео зв’язку;

— з відкритим чи закритим доступом;

— максимальною кількістю користувачів;

Система аналізує введенні дані і в разі знаходження кімнати, яка задовольняють всім введеним параметрам, відображає її користувачеві.

4.3 Інтерфейс кімнати для інтерв’ю

Користувач при виборі конкретної кімнати автоматично переходить, головну сторінку для проведення інтерв’ю. В конкретному випадку можна спостерігати, що користувач Vania, зареєструвався і проводить інтерв’ю з Дмитром. З лівого краю відображається спеціальний редактор, де в режимі реального часу можна писати програмний код і результат відображається в консолі.

На рисунку 4.7 можна бачити макет і головні компоненти.

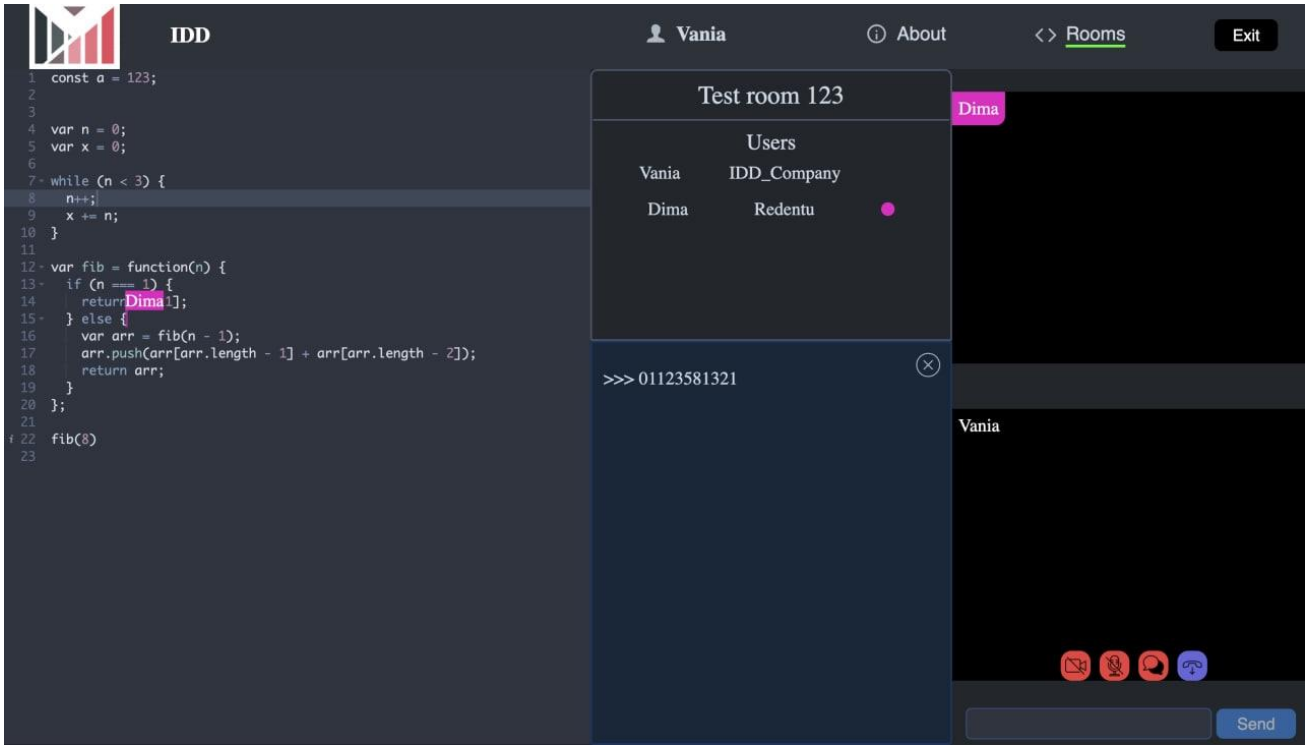

Рисунок 4.7 —Сторінка для проведення інтерв'ю

У верхній частині можна спостерігати інформацію про підключених осіб, їх імена і назву компаній. Також відповідним кольором підсвічується будь –яка діяльність того чи іншого користувача.

Дані компоненти забезпечують можливість проведення практичної частини, проте варто звернути увагу, що можна підключитися і до відео-конференції. На рисунку 4.7 ми бачимо, що з правого боку увімкнені камери для взаємодії один з одним, такий функціонал був реалізований за основними принципами ZOOM і відповідно можна увімкнути камеру, звук, чат або завершити розмову, для цього відображаються спеціальні знаки, які є інтуїтивно зрозумілими і зручними для користувачів. З метою безпеки і захищеності даних ми на рисунку 4.7 підключилися до веб конференції з відключеними камерами, щоб продемонструвати роботи конференції та захистити реальні обличчя розробників.

Веб-застосунок для проведення інтерв'ю працює за допомогою веб сокетів, тож така технологія надає можливість обмінюватися інформацією в режимі реального часу. Також продемонстровано весь функціонал, який був доступний для одного користувача. Розглянемо, що в цей же час буде бачити інша особа яка підключення до цієї кімнати рисунок 4.8.

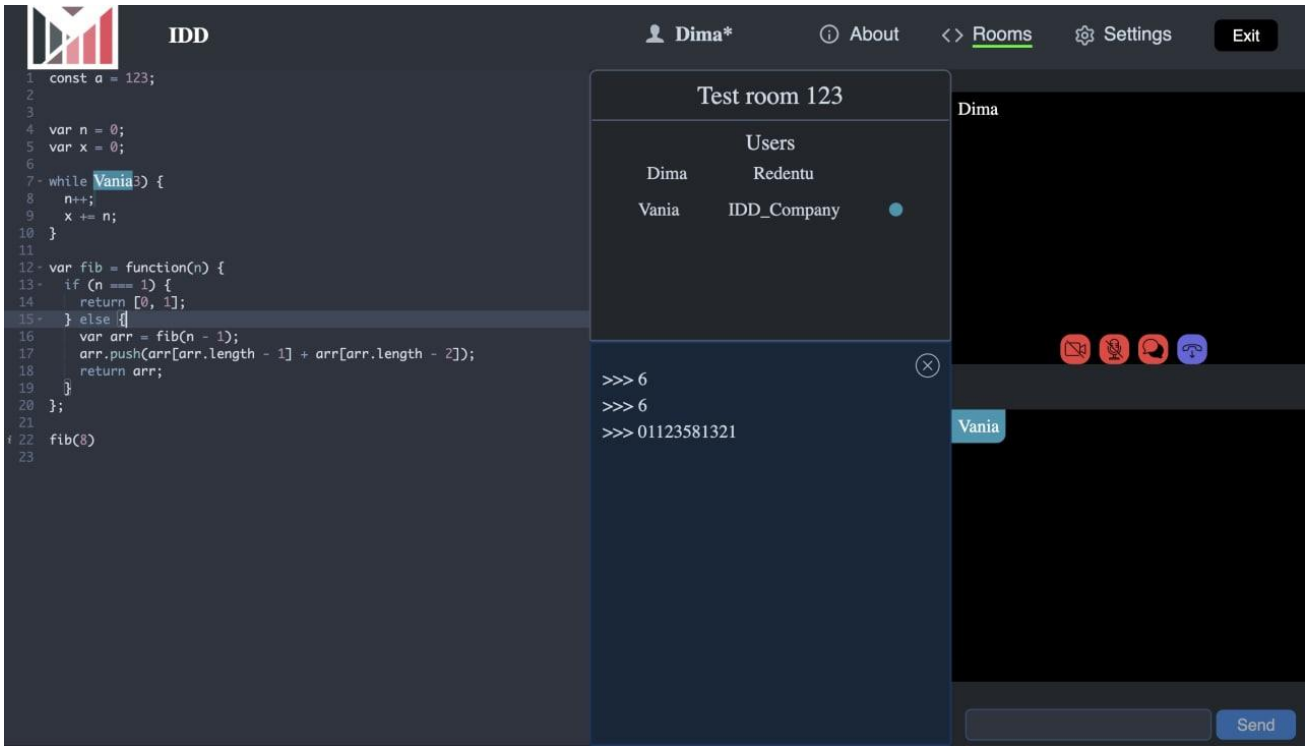

Рисунок 4.8 —Інший користувач системи підключений до кімнати

На рисунку 4.8 ми бачимо іншого користувача системи, з іменем Dima, біля нього стоїть спеціальний знак "*", це означає, що він є адміністратором і має більш широкий спектр можливостей таких як створення, редагування та видалення кімнат.

Розглянемо можливості, які дають можливість відразу змінити адміністратор.

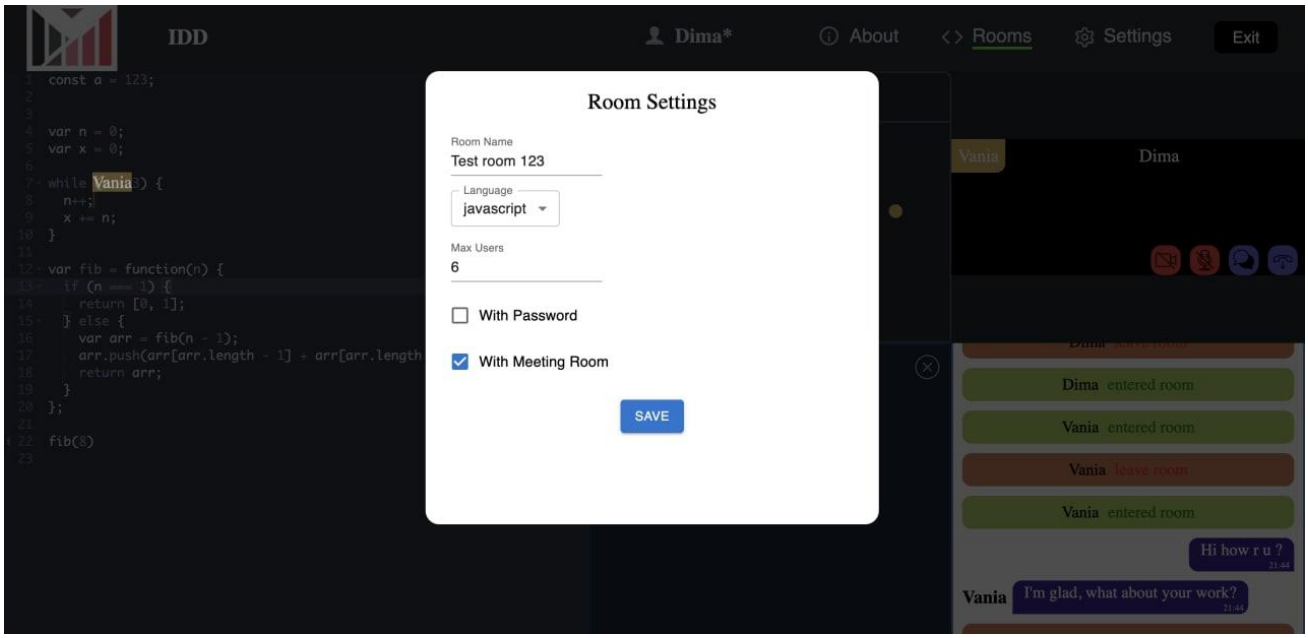

Рисунок 4.9 — Налаштування кімнати

Такий спектр можливостей дозволяє створювати такі кімнати з конкретними вимогами і потребам и для зручного і швидкого їх використання.

Рисунок 4.10 — Чат для спілкування

В сучасному світі надзвичайно важливо мати можливість обмінюватися повідомленнями, адже важко уявити систему для проведення інтерв’ю, якщо ти не можеш відправити повідомлення. На рисунку 4.10 можна побачити інтерфейс

для спілкування. Перш за все приходять певні повідомлення про приєднання користувачів, чи залишення даної конференції.

У випадку, якщо не потрібно використовувати його можна звернути і він не буде займати багато місця.

Висновки до розділу

У даному розділі описаний зовнішній інтерфейс та основний функціонал розробленої системи. Була розроблена покрокова інструкція з використання даного програмного продукту. Я вважаю, що всі поставленні задачі були реалізовані і в результаті ми маємо зручний застосунок, який вже готовий до використання.

Дизайн є сучасним і в той же час простим, адже в простоті краса і сила. Також було проведено тестування даного програмного продукту незалежними користувачами.

5 МАСШТАБУВАННЯ ТА ІНТЕГРУВАННЯ ПРОЕКТУ

В сучасному світі одним з найважливіших моментів при розробці програмного продукту є можливість його розширення і вдосконалення в довго строковій перспективі. Бізнес аналітики перед початком роботи складають план, де детально розписують терміни, до яких зазвичай повинна бути робоча версії програми. В той же час заздалегідь плануються різні версії даного програмного продукті, кожна з яких буде мати певний додатковий функціонал. Тож надзвичайно важливо з самого початку писати код, який буде мати можливість масштабуватися, доповнюватися новим функціоналом та редагуватися в залежності від потреб.

Мій застосунок має базовий функціонал, який забезпечує зручну роботу користувача. Проте ряд додаткових можливостей планую розробити. Так на даний момент додаток підтримує не всі мови програмування, а лише найпопулярніші. Дане обмеження може бути досить критичним для проведення інтерв'ю.

Реалізована можливість для підключення за допомогою відео- конференції, це допомагає за допомогою камери бачити один одного, проте максимальна кількість учасників є обмеженою, адже це додаткове навантаження на додаток. Передбачається в наступних версіях мого застосунку вирішити дане питання і доопрацювати програмний продукт.

На даний момент система реалізована, як веб-застосунок, тобто за допомогою будь-якого браузера можна відкрити і почати використовувати його. Проте в майбутньому передбачається створення десктопного застосунку, тобто окремого додатку, який можна буде скачати через Play Market і використовувати його як на персональному комп'ютері, ноутбуці так і на телефоні.

ВИСНОВКИ

В ході виконання даної дипломної роботи було створено програмний продукт для проведення інтерв’ю в дистанційному режимі. Під час розробки було проведено ряд заходів для проведення аналітичної роботи з виявлення недоліків у вже наявних системах і виправлення їх в розробленій системі.

Веб-застосунок є унікальним і зручним у використанні, надає широкий спектр можливостей для зручної взаємодії, а саме:

— реєстрація користувачів;

— вибір мови програмування;

— швидкий доступ на відеоконференції;

— редагування і створення кімнат для проведення інтерв’ю;

— віртуальний редактор з консоль для виведення результатів;

— захищені кімнати для обмеження доступу;

Дана система є ефективною і легко доступною, адже вона підтримується на всіх можливих браузерах. Розробка здійснювалася за допомогою найновіших технологій доступних на даний момент і з дотриманням базових принципів програмування , то ж веб-застосунок є масштабованим і легко підтримуваним, а найголовніше перспективним і актуальним в наш час.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. React - JavaScript бібліотека [Електронний ресурс] — Режим доступу до ресурсу: https://uk.reactjs.org/.
2. Redux Toolkit [Електронний ресурс] — Режим доступу до ресурсу: https://redux-toolkit.js.org/.
3. Redux-Saga - An intuitive Redux side effect manager [Електронний ресурс] — Режим доступу до ресурсу: https://redux-saga.js.org/.
4. TypeScript: JavaScript With Syntax For Types. [Електронний ресурс] — Режим доступу до ресурсу: https://www.typescriptlang.org/.
5. WebSocket - Web APIs [Електронний ресурс]. — Режим доступу до ресурсу: https://developer.mozilla.org/uk/docs/Web/API/WebSocket/ .
6. WebSocket - Сучасний збірник JavaScript [Електронний ресурс] — Режим доступу до ресурсу: https://learn.javascript.ru/websockets/.
7. PeerJS - Simple peer-to-peer with WebRTC [Електронний ресурс]. — Режим доступу до ресурсу: https://peerjs.com/.
8. Axios Docs [Електронний ресурс]. — Режим доступу до ресурсу: https://axios-http.com/docs/intro/ .
9. Styled Components [Електронний ресурс]. — Режим доступу до ресурсу: https://www.styled-components.com/.
10. Sass: Syntactically Awesome Style Sheets [Електронний ресурс]. — Режим доступу до ресурсу: https://sass-lang.com/.
11. Material UI - React style library [Електронний ресурс]. — Режим доступу до ресурсу: https://mui.com/.
12. Node Package Manager [Електронний ресурс]. — Режим доступу до ресурсу: https://docs.npmjs.com/ .
13. Node.js [Електронний ресурс]. — Режим доступу до ресурсу: https://nodejs.org/uk/docs/ .

14. Express.js [Електронний ресурс]. — Режим доступу до ресурсу: https://expressjs.com/en/ .

15. React Router [Електронний ресурс]. — Режим доступу до ресурсу: https://reactrouter.com/.

16. Styled Components [Електронний ресурс]. — Режим доступу до ресурсу: https://styled-components.com/docs.

17. React Hook Form - Simple React forms validation [Електронний ресурс] — Режим доступу до ресурсу: https://react-hook-form.com/ .

ДОДАТОК А

Веб-система проведення інтерв'ю

Текст програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР81337_22Б 12-1

Аркушів 9

Київ — 2022

Залежності проекту:


```
{
  "name": "diploma",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@emotion/react": "^11.9.0",
    "@emotion/styled": "^11.8.1",
    "@mui/material": "^5.6.0",
    "@reduxjs/toolkit": "^1.7.2",
    "@testing-library/jest-dom": "^5.16.2",
    "@testing-library/react": "^12.1.3",
    "@testing-library/user-event": "^13.5.0",
    "@types/draft-js": "^0.11.9",
    "@types/jest": "^27.4.0",
    "@types/node-sass": "^4.11.2",
    "@types/react": "^17.0.39",
    "@types/react-dom": "^17.0.11",
    "@types/simple-peer": "^9.11.4",
    "@types/styled-components": "^5.1.24",
    "ace-builds": "^1.4.14",
    "axios": "^0.26.0",
    "classnames": "^2.3.1",
    "cors": "^2.8.5",
    "draft-js": "^0.11.7",
    "node-sass": "^7.0.1",
    "peer": "^0.6.1",
    "react": "^17.0.2",
    "react-ace": "^9.5.0",
    "react-dom": "^17.0.2",
    "react-hook-form": "^7.27.1",
    "react-redux": "^7.2.6",
    "react-router": "^6.2.1",
    "react-router-dom": "^6.2.1",
    "react-scripts": "5.0.0",
    "redux": "^4.1.2",
    "redux-saga": "^1.1.3",
    "sass": "^1.49.8",
    "simple-peer": "^9.11.1",
    "styled-components": "^5.3.3",
    "typescript": "^4.5.5",
  },
}
```

Створення сховища Redux:

```
import { configureStore } from '@reduxjs/toolkit';
import createSagaMiddleware from 'redux-saga';
import { rootSaga } from './sagas/root.saga';
import { rooms } from './slices/rooms.slice';
import { user } from './slices/user.slice';

const sagaMiddleware = createSagaMiddleware();

const middleware = [sagaMiddleware];

export const store = configureStore({
  reducer: { user, rooms },
  middleware,
});

sagaMiddleware.run(rootSaga);
```

Створення ключових саг:

```
import { Task } from 'redux-saga';
import { call, cancel, delay, fork, put, select, take, takeEvery } from 'redux-saga/effects';
import { RoomsApi } from '../../api/RoomApi';
import { Room } from '../../utils/types/rooms.types';
import { roomsAction } from '../actions/sagaActions';
import { getRoomId } from '../selectors/rooms.selectors';
import { getUserId } from '../selectors/user.selectors';
import { RoomsActions } from '../slices/rooms.slice';
import { send } from './ws.saga';

function* fetchingRooms() {
  try {
    while (true) {
      const rooms: Room[] = yield call(RoomsApi.getRooms);
      yield put(RoomsActions.setRooms(rooms));
      yield delay(5000);
    }
  } catch (error) {}
}

function* getRooms() {
  const fetching: Task = yield fork(fetchingRooms);

  yield take(roomsAction.STOP_FETCH_ROOMS);
```

```
  yield cancel(fetching);
}

function* leaveRoom() {
  const roomId: number = yield select(getRoomId);
  const userId: number = yield select(getUserId);

  try {
    yield call(send, { type: 'LEAVE', roomId, userId });
  } catch (error) {
    console.log(error);
  }
}

function* joinLive({ payload }: any) {
  const roomId: number = yield select(getRoomId);
  const userId: number = yield select(getUserId);

  try {
    yield call(send, { type: 'JOIN_LIVE', roomId, userId, userPeerId: payload });
  } catch (error) {
    console.log(error);
  }
}

export function* roomsSaga() {
  yield takeEvery(roomsAction.START_FETCH_ROOMS, getRooms);
  yield takeEvery(roomsAction.LEAVE, leaveRoom);
  yield takeEvery(roomsAction.JOIN_LIVE, joinLive);
}


import { PayloadAction } from '@reduxjs/toolkit';
import { call, put, select, takeEvery } from 'redux-saga/effects';
import { editorAction } from '../actions/sagaActions';
import { getRoomId } from '../selectors/rooms.selectors';
import { getUser, getUserId } from '../selectors/user.selectors';
import { UserActions } from '../slices/user.slice';
import { send } from './ws.saga';

function* changeCursor({ payload: position }: PayloadAction<{ x: number; y: number }>) {
  const roomId: number = yield select(getRoomId);
  const user: number = yield select(getUser);
  yield put(UserActions.refreshCursor({ position }));
```

```
  yield call(send, { type: 'CURSOR', roomId, user, position });
}

function* changeCode({ payload: code }: PayloadAction<string>) {
  const roomId: number = yield select(getRoomId);
  yield call(send, { type: 'CODE', roomId, code });
}

function* sendMessageInChat({ payload: message }: PayloadAction<string>) {
  const roomId: number = yield select(getRoomId);
  const userId: number = yield select(getUserId);
  yield call(send, { type: 'CHAT', roomId, userId, message });
}

export function* editorSaga() {
  yield takeEvery(editorAction.CODE, changeCode);
  yield takeEvery(editorAction.CURSOR, changeCursor);
  yield takeEvery(editorAction.SEND_CHAT, sendMessageInChat);
}
```

Створення сторінки з кімнатами Redux:

```
import React from 'react';
import { useDispatch } from 'react-redux';
import { useNavigate } from 'react-router';
import {
  joinRoomAction,
  signOutAction,
  startFetchingRooms,
  stopFetchingRooms,
} from '../../../redux/actions/sagaActions';
import { getRooms } from '../../../redux/selectors/rooms.selectors';
import { getUserName } from '../../../redux/selectors/user.selectors';
import { getIcon } from '../../../utils/helpers/getIcon';
import { useAppSelector } from '../../../utils/hooks/redux';
import { useRooms } from '../../../utils/hooks/useRooms';
import { VideoIcon } from '../../components/Icons/Editor.icons';
import { UserIcon } from '../../components/Icons/Header.icons';
import {
  FilterIcon,
  PasswordOffIcon,
  PasswordOnIcon,
  SearchIcon,
} from '../../components/Icons/Rooms.icons';
```

```
import { FilterModal } from '../../modals/FilterModal/FilterModal';

import './Rooms.styles.scss';

export const Rooms: React.FC = () => {
  const dispatch = useDispatch();
  const navigate = useNavigate();
  const [openModal, setOpenModal] = React.useState<boolean>(false);
  const [search, setSearch] = React.useState<string>('');
  const rooms = useAppSelector(getRooms);
  const userName = useAppSelector(getUserName);

  React.useEffect(() => {
    const disconnect = () => {
      window.confirm('Are u sure leave?') && dispatch(signOutAction());
    };

    dispatch(startFetchingRooms());
    window.addEventListener('popstate', disconnect);

    return () => {
      dispatch(stopFetchingRooms());
      window.removeEventListener('popstate', disconnect);
    };
  }, []);

  const onRoomEnterned = (roomId: number) => {
    const room = rooms.find((room) => room.id === roomId);
    if (room?.password.length) {
      if (window.prompt() === room.password) {
        dispatch(joinRoomAction(roomId));
        navigate(/rooms/${roomId});
      } else {
        alert('Incorrect room password');
      }
    } else {
      dispatch(joinRoomAction(roomId));
      navigate(/rooms/${roomId});
    }
  };

  return (
    <div className="rooms">
      <div className="rooms-searcher">
```

```
  <SearchIcon />
  <input
   type="text"
   placeholder="Search room"
   value={search}
   onChange={(e: any) => setSearch(e.target.value)}
  />
  <FilterIcon onClick={() => setOpenModal((prev) => !prev)} />
</div>
<ul className="rooms-list">
  {rooms.map((room, index) => (
    <li key={index}>
      <span onClick={() => onRoomEnterned(room.id)}>
        {room.name}
      </span>
      <span onClick={() => onRoomEnterned(room.id)} className="users-online">
        <UserIcon />
        Online: {room.users.length}
      </span>
      <span onClick={() => onRoomEnterned(room.id)}>Max: {room.maxUsers}</span>
      <span>
        {room.isLiveChat ? <VideoIcon video={false} /> : <VideoIcon video={true} />}
      </span>
      <span>{room.password ? <PasswordOnIcon /> : <PasswordOffIcon />}</span>
      <span className="lang">{getIcon(room.language)}</span>
      )}
    </li>
  ))}
</ul>
{isAdmin && (
  <div className="create" onClick={() => {}}>
    <h2>Add Room</h2>
  </div>
      <div>
        <svg
          onClick={() => onRoomEnterned(room.id)}
          className="edit"
          width="32"
          height="32"
          viewBox="0 0 32 32"
          style={{ fill: '#000000' }}/>
        <svg
          onClick={() => onRoomEnterned(room.id)}
          className="delete"
```

```
            width="40"
            height="40"
            viewBox="0 0 100 100"
            style={{ fill: '#000000' }} />
        </div>
    )}
{openModal && <FilterModal setOpen={setOpenModal} />}
  </div>
 );
};
```

Хук для підтягування кімнат та їх фільтрації:

```
import React from 'react';
import { getFilterSettings, getRooms } from '../../redux/selectors/rooms.selectors';
import { Room } from '../types/rooms.types';
import { useAppSelector } from './redux';
import { useDebounce } from './useDebounce';

const useFilter = () => {
 const rooms = useAppSelector(getRooms);
 const filter = useAppSelector(getFilterSettings);
 const [filteredRooms, setFilteredRooms] = React.useState<Room[] | []>([]);

 React.useEffect(() => {
  setFilteredRooms(
   rooms.filter((room) => {
    if (
     (room.password.length && filter.withPass)
     (!room.password.length && !filter.withPass)
    ) {
     if (filter.maxUsers === 0  room.maxUsers === filter.maxUsers) {
      if (room.users.length >= filter.withUsersMore) {
       if (room.isLiveChat === filter.withLiveChat) {
        if (filter.lang === 'any' || room.language === filter.lang) {
         return room;
        }
       }
      }
     }
    }
   }),
  );
```

```
  }, [filter]);

  return filter.on ? filteredRooms : rooms;
};

export const useRooms = (search: string) => {
  const rooms = useFilter();
  const filter = useAppSelector(getFilterSettings);
  const debounce = useDebounce(search, 100);

  const [searchedRooms, setSearchedRooms] = React.useState<Room[] | []>([]);

  React.useEffect(() => {
    setSearchedRooms(rooms);
  }, [rooms]);

  React.useEffect(() => {
    setSearchedRooms(
      filter.searchBy === 'name'
        ? rooms.filter((room) => room.name.toLowerCase().includes(search.toLowerCase()))
        : rooms.filter((room) => room.id === +search.toLowerCase()),
    );
  }, [debounce]);

  return searchedRooms;
};
```