МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри _____Давиденко Є.О.

підпис

«___» _____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

КОРПОРАТИВНА СОЦІАЛЬНА МЕРЕЖА НАВЧАЛЬНО-ВИХОВНОГО ЗАКЛАДУ

Спеціальність «Інженерія програмного забезпечення»

121 – КРМ.1 – 608м.1810821

Здобувач _____Парахненко Р.А.

підпис

«__» _____ 2024 р.

Керівник д-р техн.наук, професор _____Швед А.В.

підпис

«__» _____ 2024 р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Зав. кафедри _____ Давиденко Є.О.

«_____» _____ 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи магістра

Видано студенту групи 608м факультету комп'ютерних наук

_____Парахненко Ростислав Андрійович_____
(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи «Корпоративна соціальна мережа навчально-виховного закладу»
Затверджена наказом по ЧНУ від «10» __листопад____ 2023 р. № ___234___

2. Строк представлення кваліфікаційної роботи «____» _____ 2024 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Вхідні дані для роботи включають в себе вимоги функціональності до програмного забезпечення пов'язаних з користуванням корпоративною соціальною мережею навчально-виховного закладу. Результатом є розроблена та працююча соціальна мережа, яка допомагає покращити комунікацію, співпрацю та обмін інформацією між вчителями та батьками.

4. Перелік питань, що підлягають розробці:
   - аналіз предметної сфери розробки програмного забезпечення корпоративної соціальної мережі навчально-виховного закладу
   - аналіз факторів та вимог для функціонування застосунку
   - проєктування та розробка програмного забезпечення соціальної мережі
   - тесту

Керівник роботи _____д-р техн. наук, професор Швед А.В._____
(посада, прізвище, ім'я, по батькові)

_____
(підпис)

Завдання прийнято до виконання

_____Парахненко Ростислав Андрійович_____
(прізвище, ім'я, по батькові студента)

_____
(підпис)

Дата видачі завдання «____» _____ 2024 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Корпоративна соціальна мережа навчально-виховного закладу _____

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРМ	10.11.2023р.	11.11.2023р.	виконано
2.	Огляд літератури за темою роботи	11.11.2023р.	18.11.2023р.	виконано
3.	Складання календарного плану КРМ	18.11.2023р.	24.11.2023р.	виконано
4.	Аналіз предметної області	24.11.2023р.	28.11.2023р.	виконано
5.	Розробка проєктних рішень	28.11.2023р.	03.12.2023р.	виконано
6.	Моделювання та конструювання ПЗ	03.12.2023р.	15.12.2023р.	виконано
7.	Розробка, тестування проєкту	15.12.2023р.	7.01.2024р.	виконано
8.	Відгук керівника КРМ	7.01.2024р.	8.01.2024р.	виконано
9.	Оформлення КРМ та презентації	8.01.2024р.	10.02.2024р	виконано
10.	Попередній захист	10.02.2024р	16.02.2024р.	виконано
11.	Рецензування	16.02.2024р.	17.02.2024р.	виконано
12.	Завершення оформлення КРМ та презентації	17.02.2024р.	18.02.2024р.	виконано
13.	Захист кваліфікаційної роботи	27.02.2024р.	27.02.2024р.	виконано

Розробив студент ___Парахненко Ростислав Андрійович_____ _____
(*прізвище, ім'я, по батькові)* *(підпис)*
«__» _____2024 р.

Керівник роботи __д-р техн. наук, професор Швед А.В_____ _____
(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«__» _____ 2024

АНОТАЦІЯ

до кваліфікаційної роботи магістра
«Корпоративна соціальна мережа навчально-виховного закладу»
Студент 608м гр.: Парахненко Ростислав Андрійович
Керівник: Швед А.В.

Корпоративна соціальна мережа навчально-виховного закладу використовується для покращення комунікації та співпраці між учасниками навчального процесу. Ця платформа надає зручний засіб для обміну інформацією, спільного вирішення завдань, організації навчальних заходів.

Об’єктом дослідження є процеси пов'язані із організацією системи комунікаціної взаємодії між учасниками навчально-виховного середовища. Предметом кваліфікаційної роботи є підходи та інформаційні технології розробки програмного забезпечення корпоративної соціальної мережі. Метою кваліфікаційної роботи є підвищення ефективності комунікації, співпраці та обміну інформацією між учасниками навчально-виховного процесу за рахунок розробки корпоративної соціальної мережі.

У розділі 1 досліджено особливості розробки, впровадження та оцінки ефективності корпоративних соціальних мереж у навчальних закладах, включаючи аналіз використання, принципи створення, імплементацію, аналіз аналогів програмного забезпечення та формулювання вимог до програмного забезпечення. У розділі 2 описано планування розробки програмного забезпечення та основні функціональні можливості системи, а також визначає структуру бази даних для зберігання інформації. У розділі 3 було проаналізовано процес вибору технологій для реалізації клієнтської та серверної частин програми, а також бази даних та додаткових бібліотек. У розділі 4 було представлено реалізацію програмного забезпечення корпоративної соціальної мережі для навчально-виховних закладів, включаючи планування розробки, використання інструментів та бібліотек.

КРМ викладено на 68 сторінок, містить 4 розділи, 30 рисунків, 26 джерел переліку посилань.

Ключові слова: соціальна мережа, навчальний заклад, React, Firebase.

ABSTRACT

for the master’s qualification work
"Corporate Social Network for Educational Institutions"
Student 608m group: Parakhnenko Rostyslav Andriiovych
Supervisor: Shved A.V.

The corporate social network of an educational institution is utilized to enhance communication and collaboration among participants in the educational process. This platform provides a convenient means for exchanging information, solving tasks collectively, and organizing educational events.

The object of the research is the processes related to organizing the system of communication interaction among participants of the educational environment. The subject of the qualification work involves approaches and information technologies for developing software for a corporate social network. The aim of the qualification work is to enhance communication, collaboration, and information exchange among participants of the educational process through the development of a corporate social network.

In Chapter 1, the focus is on examining the development, implementation, and evaluation intricacies of corporate social networks within educational institutions. This includes analyzing usage patterns, establishing creation principles, implementation strategies, exploring software analogs, and detailing software requirements. Chapter 2 delves into the planning of software development, outlining the system's core functionalities and defining the database structure for information storage. Chapter 3 scrutinizes the technology selection process for both client-side and server-side implementations, alongside database choices and supplementary libraries. Finally, Chapter 4 presents the practical implementation of the corporate social network software for educational institutions, covering development planning, tool usage, and library integration.

The corporate social network research paper spans 68 pages, comprising 4 chapters, 30 figures, and referencing 26 sources.

Keywords: social network, educational institution, React, Firebas

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ДСТУ – Державний стандарт України ЕК– екзаменаційна комісія

ЗВО – заклад вищої освіти

ІПЗ – інженерія програмного забезпечення ККРМ – комплексна кваліфікаційна робота магістра КРМ – кваліфікаційна робота магістра

МОНУ – Міністерство освіти і науки України ОПП – освітньо-професійна програма

ПЗ – програмне забезпечення ПІБ – прізвище, ім’я, по батькові

ЧНУ ім. Петра Могили – Чорноморський національний університет імені Петра Могили

ARIS – Architecture of Integrated Information Systems CASE – Computer-Aided Software Engineering

DFD – Data Flow Diagram

ECTS – European Credit Transfer and Accumulation System IDEF – I-CAM DEFinition or Integrated DEFinition Methods SRS – Software Requirements Specification

WoS – Web of Science

БД – база даних

ВСТУП

Корпоративна соціальна мережа навчально-виховного закладу – це інноваційний інструмент, спрямований на підвищення ефективності комунікації та співпраці всередині освітньої громади. Розробка та впровадження такої мережі передбачає створення інтегрованої платформи, що сприяє обміну інформацією між вчителями, учнями і батьками, враховуючи сучасні стандарти освіти.

Об’єктом дослідження є процеси пов'язані із організацією системи комунікаціної взаємодії між учасниками навчально-виховного середовища.

Предметом кваліфікаційної роботи є підходи та інформаційні технології розробки програмного забезпечення корпоративної соціальної мережі.

Метою кваліфікаційної роботи є підвищення ефективності комунікації, співпраці та обміну інформацією між учасниками навчально-виховного процесу за рахунок розробки корпоративної соціальної мережі

Для досягнення поставленої мети необхідно виконати наступні завдання:

1) Аналіз предметної сфери розробки корпоративної соціальної мережі навчально-виховного закладу.
2) Аналіз існуючих аналогі та їх переваги.
3) Вибір бази даних.
4) Специфікація вимог.
5) Реалізація застосунку.

Актуальність теми полягає в тому, що сучасне освітнє середовище потребує ефективних засобів комунікації та співпраці між всіма учасниками навчального процесу. Розвиток інформаційних технологій вимагає створення інноваційних інструментів, які забезпечать зручну та швидку взаємодію між вчителями, учнями та батьками. Корпоративна соціальна мережа для навчальних закладів може стати важливим інструментом у підвищенні якості освіти, забезпечуючи зручний доступ до інформації, спільні ресурси та можливість взаємодії в реальному часі. Така мережа сприятиме активному взаємодії між учасниками навчального процесу, що позитивно вплине на його результативність та ефективність.

1 ОСОБЛИВОСТІ РОЗРОБКИ ТА ІМПЛЕМЕНТАЦІЇ В ПРОГРАМНИХ ЗАСОБІВ ВНУТРІШНЬОЇ КОМУНІКАЦІЇ В ОСВІТНІЙ СОЦІАЛЬНІЙ СИСТЕМІ

1.1 Опис предметної сфери розробки корпоративних соціальних мереж у навчально-виховних закладах

Цільовими орієнтирами даної кваліфікаційної роботи є проведення глибокого аналізу існуючих програмних засобів внутрішньої соціальної мережі навчально-виховного закладу. Основні завдання включають:

– Аналіз функціональності:

Ретельне вивчення та аналіз функціональних можливостей існуючих програмних засобів, зокрема їхньої спроможності сприяти ефективній внутрішній комунікації та співпраці в освітньому середовищі.

– Оцінка безпеки та конфіденційності:

Проведення аналізу систем безпеки існуючих рішень для забезпечення високого рівня захисту особистої інформації користувачів та конфіденційних даних.

– Вивчення архітектури та технічних характеристик:

Розгляд та порівняння архітектурних рішень та технічних характеристик існуючих соціальних мереж, визначення їхньої ефективності та масштабованості.

– Аналіз користувацького досвіду (UX):

Оцінка користувацького інтерфейсу та вивчення досвіду використання програмних засобів для забезпечення приємного та зручного користувацького досвіду.

– Визначення переваг та недоліків існуючих рішень:

Ідентифікація сильних сторін та недоліків різних соціальних мереж, що використовуються в освітньому середовищі, для здобуття уроків та оптимізації власного розроблення.

– Проведення порівняльного аналізу:

Систематичне порівняння різних програмних засобів на основі отриманих

результатів аналізу з метою визначення оптимальних рішень для розробки внутрішньої соціальної мережі навчально-виховного закладу.

У сфері професійної діяльності важливою ролью відіграють корпоративні соціальні мережі. Це спеціально адаптовані інструменти, які дозволяють вирішувати професійні завдання та оптимізувати внутрішні процеси в організації. Вони виникли в нещодавньому минулому, паралельно зі зростанням популярності звичайних соціальних мереж. Важливо розрізняти корпоративну соціальну мережу від інтранету.

Інтранет – це внутрішня мережа організації, яка обмежена віртуальним простором конкретної компанії і захищена від непов'язаного доступу до корпоративних вузлів. Відмінною особливістю є те, що інтранет складається з внутрішньокорпоративних веб-сайтів, або навіть декількох таких сайтів. Його побудову можна поділити на дві основні технології: системи електронного корпоративного управління та корпоративні портали.

Отже, порівнюючи інтранет та корпоративну соціальну мережу, важливо враховувати, що інтранет — це більш широке поняття, яке включає в себе кілька різних компонентів. Зазвичай термін "інтранет" асоціюється із корпоративним порталом. Однак на практиці інтранети часто не користуються популярністю серед користувачів, і, в більшості випадків, вони вступають в конкуренцію з месенджерами та звичайними соціальними мережами.

У порівнянні корпоративного порталу та корпоративної соціальної мережі визначаються різноманітні відмінності та особливості, які визначають їхню призначеність та функціональність.

Спрямування:

- *Корпоративний портал:* Орієнтований на надання внутрішньокорпоративної інформації, ресурсів та інструментів для оптимізації бізнес-процесів і комунікації в організації.
- *Корпоративна соціальна мережа:* Зорієнтована на створення інтерактивного простору для спілкування, обміну ідеями та співпраці між співробітниками, сприяючи вирішенню завдань і покращенню

комунікації.

Комунікація та співпраця:

- *Корпоративний портал:* Забезпечує базові інструменти комунікації, але часто акцентується на односторонньому розповсюдженні інформації.
- *Корпоративна соціальна мережа:* Зорієнтована на двосторонній обмін інформацією, сприяючи взаємодії та вирішенню завдань через співпрацю.

Особистий профіль:

- *Корпоративний портал:* Зазвичай має обмежений особистий профіль, орієнтований на професійні дані та робочі завдання.
- *Корпоративна соціальна мережа:* Надає більший акцент на особистий профіль та взаємодію між співробітниками на особистому рівні.

Спільноти та групи:

- *Корпоративний портал:* Може мати можливості для формування спільнот, але зазвичай це є менш вираженим аспектом.
- *Корпоративна соціальна мережа:* Активно сприяє формуванню спільнот та груп інтересів для ефективної роботи над проєктами та обміну знаннями.

Гнучкість та адаптивність:

- *Корпоративний портал:* Зорієнтований на стабільну роботу та надання визначеного набору інструментів.
- *Корпоративна соціальна мережа:* Забезпечує більшу гнучкість та адаптивність для сприяння швидкій взаємодії та змінам в командній роботі.

Такі платформи створюють можливості для проведення онлайн конференцій та семінарів, які у деяких випадках стають необхідною альтернативою особистим зустрічам, що не завжди є практичним. Крім того, важливо враховувати простоту включення нових членів у колектив. Корпоративні соціальні мережі допомагають вирішувати питання соціалізації внутрішньо в колективі, де новачки швидше

знаходять спільноту, нові знайомства та наставників, які готові відповісти на їхні запитання. Це призводить до полегшення комунікації всередині колективу та покращує взаємини між його учасниками.

1.2 Аналіз використання соціальних мереж у навчальних закладах

Започаткування розвитку сучасних соціальних мереж датується 1995 роком, коли Ренді Конрад створив сайт Classmates.com з метою пошуку однокласників. Цей веб-ресурс відрізнявся від сучасних соціальних мереж, оскільки користувачі не мали можливості створювати профілі, вести онлайн-щоденники та додавати друзів. Однак він дозволяв визначати місцезнаходження користувачів. До цього часу Classmates.com активно функціонує, маючи більше 50 мільйонів зареєстрованих користувачів, несмотря на конкуренцію. Першою сучасною соціальною мережею був сайт Six degrees, який з'явився в 1997 році і дозволяв користувачам додавати одне одного в друзі. Ще через два роки, у 1999 році, був запущений LiveJournal. У 2002 році виник Friendster, де користувачі могли шукати потенційних друзів; на сьогодні цей ресурс найбільш популярний в Азії.

Сучасні соціальні мережі відіграють ключову роль у покращенні взаємодії та комунікації в навчальних закладах. Вони створюють відкритий простір для обміну інформацією та ідеями між студентами, викладачами та адміністрацією. Така взаємодія сприяє активній обговоренні навчальних питань та розвитку спільноти.

Застосування соціальних мереж в навчальних процесах також сприяє покращенню навчання. Вони надають можливість впровадження інтерактивних та зручних для вивчення форматів, сприяючи залученню студентів та підтримці їхнього активного навчання. Відкритий обмін знаннями та ресурсами стає джерелом нових ідей і підходів.

Однак, разом із вагомими перевагами, важливо враховувати питання безпеки та конфіденційності при використанні соціальних мереж у навчальних закладах. Заходи забезпечення безпеки та приватності стають ключовими аспектами при реалізації цих платформ для освітніх цілей.

1.3 Принципи створення корпоративних мереж у навчальних закладах

Створення корпоративних мереж у навчальних закладах ґрунтується на кількох принципах, які визначають необхідність цього заходу:

Інтеграція інформаційних ресурсів: Створення корпоративних мереж дозволяє об'єднати інформаційні ресурси навчального закладу в єдину систему. Це сприяє ефективнішому управлінню та обміну даними між відділами та структурними підрозділами.

Покращення комунікації: Корпоративні мережі сприяють поліпшенню комунікації всередині навчального закладу. Вони забезпечують зручний обмін інформацією між вчителями, учнями, адміністрацією та іншими учасниками освітнього процесу.

Забезпечення безпеки даних: Корпоративні мережі дозволяють ефективно впоратися з питаннями безпеки даних. Застосування сучасних технологій дозволяє захищати конфіденційні дані, забезпечуючи доступ лише авторизованим користувачам.

Сприяння викладацькому процесу: Корпоративні мережі дозволяють ефективно впроваджувати технології в навчальний процес. Це може включати в себе використання електронних платформ для навчання, віддалені лекції, електронні тести та інші інноваційні методи.

Зручний доступ до ресурсів: Корпоративні мережі дозволяють забезпечити зручний доступ до навчальних ресурсів для всіх учасників освітнього процесу. Це може включати в себе віддалений доступ до бібліотечних ресурсів, електронних підручників та інших освітніх матеріалів.

Підвищення ефективності управління: Створення корпоративних мереж допомагає впоратися з завданнями управління навчальним закладом. Це включає в себе автоматизацію бухгалтерських та адміністративних процесів, що полегшує роботу адміністрації.

Ці мережі сприяють впровадженню сучасних методик та педагогічних

підходів, забезпечуючи вчителів та учнів доступом до інноваційних освітніх ресурсів. Такий підхід дозволяє підтримувати активне використання інформаційних технологій у навчальному процесі та розвивати цифрову грамотність серед учнів.

Важливим аспектом створення корпоративних мереж у навчальних закладах є забезпечення ефективної адміністрації та управління. Це включає в себе автоматизацію обліку студентів, ефективний моніторинг навчальних досягнень, а також оптимізацію бухгалтерських та фінансових процесів. Завдяки корпоративним мережам адміністрація може оперативно реагувати на потреби навчального закладу та вчасно впроваджувати важливі рішення для покращення умов навчання та розвитку закладу в цілому.

1.4 Імплементація та впровадження корпоративних мереж в освітній процес

Імплементація корпоративних мереж у навчальних закладах – це складний та відповідальний процес, що передбачає кілька етапів впровадження. Першим кроком є стратегічне планування, де адміністрація визначає мету та завдання впровадження, а також враховує потреби освітнього закладу. Основною метою є підвищення якості навчання та сприяння розвитку студентів у цифровому суспільстві.

Після стратегічного планування слід вибрати відповідні технологічні рішення для імплементації. Оцінюються різні платформи та системи, здійснюється вибір, враховуючи технічні характеристики та безпеку даних. Наступним етапом є впровадження та конфігурація системи, що передбачає розробку та реалізацію плану впровадження, а також налаштування мережевої інфраструктури.

Важливим аспектом успішної імплементації є навчання персоналу. Адміністратори, вчителі та інший персонал повинні володіти необхідними навичками для використання корпоративних мереж у повсякденній діяльності. Навчання має за мету забезпечити ефективне використання можливостей нових технологій у навчальному процесі. Після успішної імплементації, важливо

забезпечити постійну оцінку та підтримку системи. Аудит ефективності дозволяє виявляти можливі покращення, а технічна підтримка забезпечує безперебійне функціонування корпоративних мереж у динамічному середовищі освітнього закладу. Паралельно із технічною підтримкою важливо створити культуру використання корпоративних мереж серед всіх учасників освітнього процесу. Це включає в себе регулярні тренінги, відкриті лекції та інші формати подій, що сприяють зростанню цифрової грамотності та розумінню можливостей корпоративних мереж у досягненні освітніх цілей.

Додатковим аспектом є забезпечення безпеки та конфіденційності даних. Введення корпоративних мереж вимагає високого рівня захисту від несанкціонованого доступу, а також розробки етичних правил використання цифрових ресурсів. Це забезпечує довіру всіх учасників освітнього процесу та зберігає репутацію навчального закладу.

У кінці процесу впровадження, важливо провести аналіз результатів та взяти до уваги відгуки користувачів. Це дозволяє коригувати стратегію розвитку корпоративних мереж та забезпечує їхню відповідність змінюючимся потребам освітнього середовища.

Крім того, створення ефективної системи зворотного зв'язку і взаємодії з користувачами є ключовим етапом у процесі імплементації корпоративних мереж.

Регулярні опитування, фокус-групи та інші інструменти збору думок і вражень використовуються для визначення ступеня задоволення користувачів, а також для виявлення можливих аспектів вдосконалення системи.

Важливим етапом є також інтеграція корпоративних мереж із загальносистемними та методико-методологічними освітніми стандартами. Це забезпечить синергію між інноваційними технологіями та традиційними методами навчання, сприяючи розвитку комплексної освітньої програми.

Загальний успіх впровадження корпоративних мереж у навчальному закладі залежить від комплексного підходу, врахування індивідуальних потреб користувачів та здатності системи пристосовуватися до швидких змін у сучасному освітньому середовищі.

Особливу увагу слід приділити питанням доступності та інклюзивності. Впровадження корпоративних мереж має бути спрямоване на те, щоб забезпечити однаковий доступ та можливості для всіх учасників освітнього процесу, незалежно від їхніх індивідуальних потреб та обмежень.

Збалансований підхід до імплементації технологій в освітньому середовищі дозволяє створити інноваційну та ефективну систему, яка сприяє розвитку всіх учасників освітнього процесу та відповідає сучасним вимогам освітньої галузі.

1.5 Оцінка ефективності корпоративних мереж у навчальних закладах

Оцінка ефективності корпоративних мереж у навчальних закладах є ключовим етапом, що визначає, наскільки ці технології відповідають потребам освітнього процесу. Основними аспектами оцінки є якість навчання, зручність використання для педагогічного персоналу та залучення студентів.По-перше, ефективність корпоративних мереж визначається їхнім впливом на якість навчання. Важливо визначити, наскільки технології сприяють активізації навчального процесу, підвищенню залучення студентів та покращенню засвоєння навчального матеріалу.

Оцінка ефективності корпоративних мереж також включає в себе розділ, присвячений зручності їх використання вчителями та іншим педагогічним персоналом. Важливо враховувати, наскільки легко вчителям адаптуватися до нових технологій, як вони використовують їх у педагогічній практиці, та наскільки це впливає на їхню роботу та взаємодію зі студентами.

По-друге, у рамках оцінки ефективності корпоративних мереж, особлива увага приділяється їхньому впливу на мотивацію та залучення студентів. Важливо визначити, чи стимулюють ці технології активну участь студентів, чи надають можливості для самостійного навчання та розвитку, а також чи сприяють вони формуванню інтерактивного навчального середовища.

Проводячи оцінку ефективності корпоративних мереж, важливо детально розглядати отримані результати та здійснювати аналіз їхнього впливу на

досягнення поставлених цілей. У подальшому розділі будуть визначені та обгрунтовані можливі шляхи вдосконалення, які допоможуть оптимізувати використання корпоративних мереж у навчальному процесі та досягти максимального позитивного впливу.

Наступним критерієм є вплив корпоративних мереж на студентів. Якщо технології стимулюють активну участь, сприяють взаємодії між студентами та надають можливості для творчого виявлення знань, це може свідчити про їхню ефективність.

Якщо технології стимулюють активну участь, сприяють взаємодії між студентами та надають можливості для творчого виявлення знань, це може свідчити про їхню ефективність. Для більш точної оцінки впливу на студентів важливо враховувати їхню активність у віртуальних спільнотах, рівень залучення до онлайн-ресурсів та здатність використовувати технології для самостійного навчання. Якщо корпоративні мережі сприяють розвитку критичного мислення, комунікаційних навичок та збільшенню мотивації до навчання, це може вказувати на їх високу ефективність у контексті підвищення академічного успіху та розвитку студентів.

Ще одним важливим аспектом є рівень зручності та доступності корпоративних мереж для всіх учасників освітнього процесу. Це включає в себе якість інтерфейсу, можливості адаптації для різних пристроїв, швидкість доступу до необхідної інформації та наявність технічної підтримки. Оцінка цих параметрів дозволяє зрозуміти, наскільки зручно та ефективно користувачі можуть взаємодіяти з системою та використовувати її для досягнення освітніх цілей.

Наступним важливим етапом є оцінка моніторингу та підтримки корпоративних мереж. Чітко визначені процедури моніторингу дозволяють вчасно виявляти можливі проблеми та удосконалювати роботу системи. Технічна підтримка гарантує вирішення виникаючих труднощів та надає користувачам необхідну допомогу. Це сприяє стабільності та ефективності корпоративних мереж у навчальному процесі.

На завершальному етапі розділу буде проведений детальний аналіз отриманих результатів оцінки ефективності корпоративних мереж. Проведення

порівняння фактичних досягнень з визначеними завданнями та метою впровадження технологій є ключовим етапом визначення успішності системи. Перевірка відповідності досягнень зазначеним цілям дозволяє визначити, наскільки корпоративні мережі відповідають вимогам освітнього закладу та чи вони ефективно сприяють досягненню покладених завдань.

На основі отриманих даних будуть визначені можливі шляхи вдосконалення корпоративних мереж у навчальному закладі. Це може включати в себе модернізацію технічної інфраструктури, вдосконалення методик використання технологій в навчанні, а також розширення функціоналу системи для більшого забезпечення потреб учасників освітнього процесу. Шляхи подальшого розвитку будуть визначатися з урахуванням виявлених недоліків та рекомендацій для оптимального використання корпоративних мереж у сучасній освіті.

1.6 Аналіз аналогів програмного забезпечення

Аналіз аналогів програмного забезпечення демонструє різноманітність підходів до створення корпоративних соціальних мереж для навчально-виховних закладів. Одним з найпопулярніших аналогів є платформа Microsoft Teams for Education, яка надає широкі можливості для навчання на відстані, співпраці та комунікації в навчальному середовищі. Її функціональність включає в себе інструменти для відеоконференцій, обміну файлами, організації класів та роздачі завдань. Ще одним аналогом є платформа Google Classroom, яка також надає інструменти для співпраці та навчання на відстані, використовуючи різноманітні можливості Google Apps. Крім того, існують інші аналоги, які можуть включати в себе Learning Management Systems (LMS), такі як Moodle, Schoology та Canvas, які також пропонують широкий спектр функціональності для навчання та співпраці в освітньому середовищі.

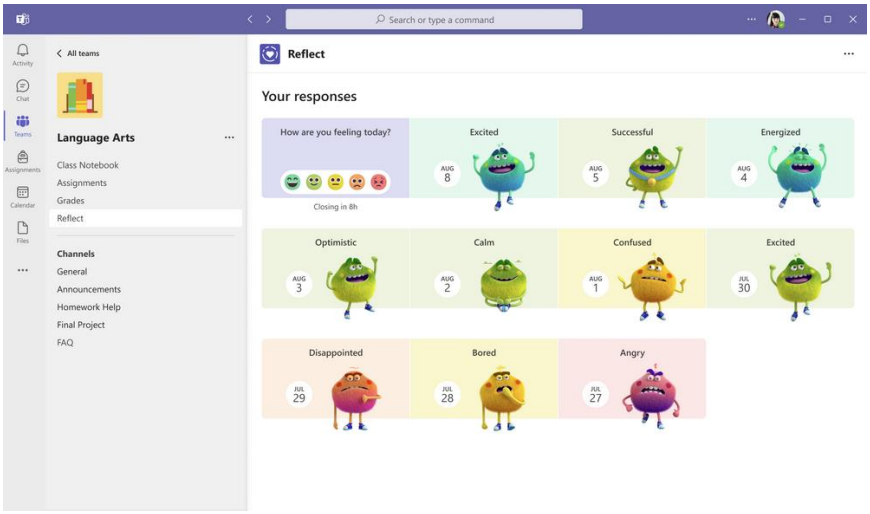

Рисунок 1.1 – Платформа Microsoft Teams for Education

Платформа Microsoft Teams for Education є одним з провідних інструментів для навчання та співпраці в освітніх закладах. Ключові аспекти цього аналогу:

1. Функціональність: Microsoft Teams for Education надає широкий спектр інструментів для навчання на відстані, включаючи можливості відеоконференцій, обміну файлами, спільної роботи над документами, завданнями та тестами, а також інтеграцію з іншими сервісами Microsoft, такими як Office 365.
2. Комунікація: Платформа дозволяє вчителям, учням і батькам спілкуватися один з одним через чати, відеоконференції та обговорення, що сприяє ефективній комунікації в навчальному середовищі.
3. Організація: Teams дозволяє організувати класи, роздавати завдання та матеріали, вести журнали та оцінки, що полегшує керування навчальним процесом для вчителів.
4. Безпека і конфіденційність: Платформа має вбудовані інструменти для захисту конфіденційності та безпеки даних учасників навчального процесу, зокрема, що стосується дотримання вимог GDPR.
5. Інтеграція: Microsoft Teams інтегрується з іншими сервісами та продуктами Microsoft, що робить його зручним інструментом для користувачів, які вже використовують ці продукти.

Ще один аналог такий як Moodle забезпечує управління навчанням та навчанням на відстані, яке широко використовується в освітній галузі. Ця платформа надає різноманітні інструменти для організації курсів, спілкування між учасниками, завдань, тестування та звітності. За допомогою Moodle викладачі можуть створювати інтерактивні курси з різними типами матеріалів, такими як тексти, відео, аудіо, завдання для самостійного виконання, форуми для обговорення, онлайн-тести та багато іншого. Студентам же надається зручний доступ до навчального матеріалу, можливість взаємодії з викладачами та співробітниками, а також можливість виконання завдань та отримання зворотного зв'язку. Moodle відомий своєю гнучкістю та можливістю адаптації до різних потреб освітніх закладів, що робить його одним з популярних інструментів для навчання на відстані.

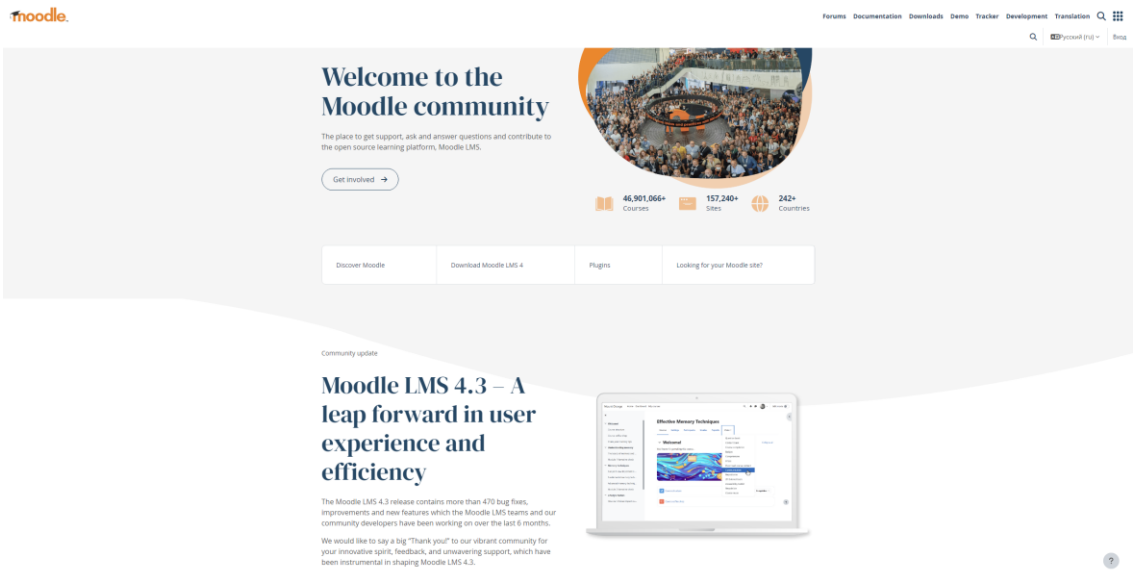

Рисунок 1.2 – Платформа Moodle

Переваги платформи Moodle включають:

1. Безкоштовність та відкритість: Moodle є безкоштовною та відкритою для використання, що дозволяє навчальним закладам економити кошти на програмному забезпеченні та розробці.
2. Гнучкість і налаштовуваність: Платформа надає велику гнучкість у налаштуванні навчального процесу та інтерфейсу для відповідності конкретним потребам навчального закладу.

3. Різноманітність функціональності: Moodle має багатий набір функцій, таких як створення курсів, завдань, форумів для обговорення, онлайн тестування тощо, що дозволяє викладачам ефективно організовувати навчальний процес.
4. Спільнота користувачів та розвиток: Існує активна спільнота користувачів та розробників, що постійно вносять вдосконалення та розвивають платформу, забезпечуючи її актуальність та підтримку.
5. Мультиплатформеність: Moodle може бути використаний на різних платформах та пристроях, що дозволяє користувачам мати доступ до навчального матеріалу з будь-якого пристрою з підтримкою веб-браузера.

Безпека: Платформа має механізми безпеки, що дозволяють захищати конфіденційні дані користувачів та забезпечувати безпеку навчального середовища.

1.7 Специфікація вимог до програмного забезпечення корпоративної соціальної мережі навчально-виховного закладу

Виконуючи розробку корпоративної соціальної мережі навчально-виховного закладу, необхідно дотримуватися таких пунктів:

– ретельний аналіз цільової аудиторії допоможе визначити основні потреби та очікування користувачів, а також спрямує розробку функціоналу та інтерфейсу.грамотно виконувати проєктування відповідно до прийнятої нормативно-технічної документації;

– чітке формулювання мети соціальної мережі та визначення конкретних завдань допоможе визначити напрямок роботи та очікувані результати.працювати з технічною літературою та іншими джерелами інформації, робити критичний аналіз публікацій з теми кваліфікаційної роботи;

– визначення основних функцій соціальної мережі, таких як створення профілю, спільнот, чат, альбомів тощо, а також унікальних особливостей, які відрізнять її від інших платформ.

– розробка зручного та естетичного інтерфейсу, який сприяє легкості використання та позитивному враженню від користування.

– аналіз конкурентів та урахування їхніх переваг та недоліків для розробки унікального та конкурентоспроможного продукту.

Функціональні вимоги данної корпоративної соціальної мережі визначаються з урахуванням потреб навчального закладу та його учасників. Однією з ключових функціональних вимог є можливість створення особистих облікових записів для всіх користувачів мережі, що включають вчителів та батьків. Даний обліковий запис має надавати зручний інтерфейс для управління особистою інформацією, включаючи персональні дані, розклади та актуальні новини.

Важливим аспектом є можливість взаємодії та обміну інформацією між вчителями та батьками в режимі реального часу. Функціональність чату та системи повідомлень дозволить учасникам мережі оперативно обговорювати питання, робити оголошення та швидко отримувати відповіді на питання.

Функціональні вимоги корпоративної соціальної мережі також включають можливість створення та перегляду галерей, де користувачі можуть додавати та переглядати фотографії, відео та інші мультимедійні матеріали, пов'язані з освітнім процесом. Крім того, важливо, щоб галереї були доступні для певних груп користувачів або всієї освітньої спільноти в цілому.

Щодо функціональності коментування постів вчителя, вона повинна включати можливість залишати коментарі та відгуки під кожним публікацією. Коментарі можуть бути відкритими для усіх учасників групи або обмеженими для конкретних користувачів. Коментування сприятиме взаємодії, обговоренню ідей та наданню конструктивного фідбеку вчителям.

Функціональні вимоги корпоративної соціальної мережі включають в себе систему повідомлень, яка повідомлятиме учасників спільноти про події та оновлення, які стосуються їхніх дітей. Зокрема, батьки можуть отримувати сповіщення про нові публікації, коментарі, або інші важливі події, пов'язані з активністю їхніх дітей чи вчителів.

Окрім того, важливо розробити функціонал для просмотру інформації профілів між родителями. Це може включати в себе можливість перегляду публічних профілів інших користувачів.

Загалом, функціональні вимоги корпоративної соціальної мережі повинні враховувати потреби всіх її учасників, забезпечуючи зручність взаємодії, ефективний обмін інформацією та сприяючи покращенню якості навчання та співпраці всередині навчально-виховного закладу.

Висновки до розділу 1

Було проведено глибокий аналіз ключових аспектів корпоративних соціальних мереж в контексті навчально-виховних закладів. Визначено цілі та завдання корпоративної соціальної мережі, виокремлено відмінності в порівнянні з порталами та розглянуто особливості їх використання в освітній сфері.

Аналіз використання соціальних мереж у навчальних закладах дозволив виявити переваги та обмеження, які можуть впливати на ефективність їх впровадження. Принципи створення корпоративних мереж у навчальних закладах визначили основні принципи та вимоги до розробки таких платформ з урахуванням освітніх потреб.

Імплементація та впровадження корпоративних мереж в освітній процес висвітлили ключові етапи та стратегії, необхідні для успішної реалізації таких інновацій у навчальному середовищі. Оцінка ефективності корпоративних мереж у навчальних закладах визначила критерії та методики оцінки результативності, враховуючи потреби учасників освітнього процесу.

Загальний аналіз цього розділу підкреслює важливість розробки та впровадження корпоративних соціальних мереж для підвищення ефективності комунікації та співпраці всередині навчально-виховних закладів.

2. ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОРПОРАТИВНОЇ СОЦІАЛЬНОЇ МЕРЕЖІ ДЛЯ НАВЧАЛЬНО-ВИХОВНИХ ЗАКЛАДІВ

2.1 Планування розробки програмного забезпечення

Для розробки корпоративної соціальної мережі для навчально-виховних закладів необхідно врахувати наступні функціональні вимоги:

1. Авторизація:
   - 1.1 Можливість авторизації за допомогою логіну та паролю або через обліковий запис Google.
   - 1.2 Функція "Sign out" для виходу з облікового запису.
2. Створення постів:
   - 2.1 Можливість створювати та публікувати нові пости на стіні користувача.
3. Перегляд інформації:
   - 3.1 Можливість переглядати профілі інших користувачів, включаючи їх особисті дані та створені пости.
4. Коментування постів:
   - 4.1 Функція коментування публікацій інших користувачів.
5. Вподобання постів:
   - 5.1 Можливість виражати зацікавленість у публікаціях інших користувачів, ставлячи "лайки" або "вподобання".
6. Додавання у друзі:
   - 6.1 Функція додавання інших користувачів у список друзів з можливістю обміну повідомленнями та перегляду обміненої інформації.
7. Чат із друзями:
   - 7.1 Можливість обмінюватися текстовими повідомленнями з іншими користувачами, які додані у список друзів.

Ці функціональні вимоги стануть основою для подальшої розробки та імплементації програмного забезпечення корпоративної соціальної мережі для навчально-виховних закладів.

Під час процесу розробки програмного забезпечення для корпоративної соціальної мережі необхідно також врахувати інші аспекти, що впливають на функціональність та ефективність системи. Один з них - це врахування безпеки даних та конфіденційності користувачів. Забезпечення захисту від несанкціонованого доступу до особистої інформації та відповідність законодавству з питань захисту даних - це критично важливий аспект для будь-якої соціальної мережі, особливо в контексті освітніх закладів, де обробка особистих даних має специфічні вимоги та обмеження.

Крім того, важливо врахувати масштабованість системи та її здатність працювати ефективно при зростанні кількості користувачів та обсягу оброблюваних даних. Правильне планування архітектури системи, використання ефективних алгоритмів та технологій масштабування допоможе забезпечити стабільну та продуктивну роботу соціальної мережі навіть при великому навантаженні.

До інших аспектів, які потрібно врахувати під час розробки, входить зручність користування та інтуїтивний інтерфейс. Для того, щоб користувачі змогли максимально комфортно використовувати соціальну мережу, важливо забезпечити зрозумілість та доступність всіх функцій, легкість навігації та швидкий доступ до потрібної інформації.

2.2. Діаграма варіантів використання

Діаграма прецедентів відображає ключові можливості користувачів у системі корпоративної соціальної мережі. Починаючи з етапу авторизації, користувачі мають доступ до головної сторінки, де вони можуть переглядати та створювати пости. Крім того, користувачі можуть взаємодіяти з іншими учасниками через функціонал чату та додавати їх у друзі. Через ці можливості вони можуть зручно обмінюватися інформацією та спілкуватися один з одним.

Крім базових можливостей, таких як створення постів та взаємодія з іншими користувачами, система також забезпечує розвинуті функціональність для зручного використання. Наприклад, можливість завантажувати зображення до постів дозволяє користувачам поділитися важливими моментами свого дня або цікавими фотографіями. Коментування постів та вираження вподобайок не лише сприяють взаємодії, а й створюють атмосферу співпраці та підтримки серед учасників мережі.

Створення та видалення постів, а також взаємодія з коментарями та вподобайками, надає користувачам широкий спектр можливостей для вираження власної думки та взаємодії з контентом. Додавання у друзі та перегляд інформації про них дозволяє розширити соціальне коло користувача та отримати більше інформації про їхню активність в системі. Загалом, ця діаграма відображає різноманітні можливості, які користувачі мають у корпоративній соціальній мережі, сприяючи активному взаємодії та обміну інформацією в навчально-виховному середовищі.

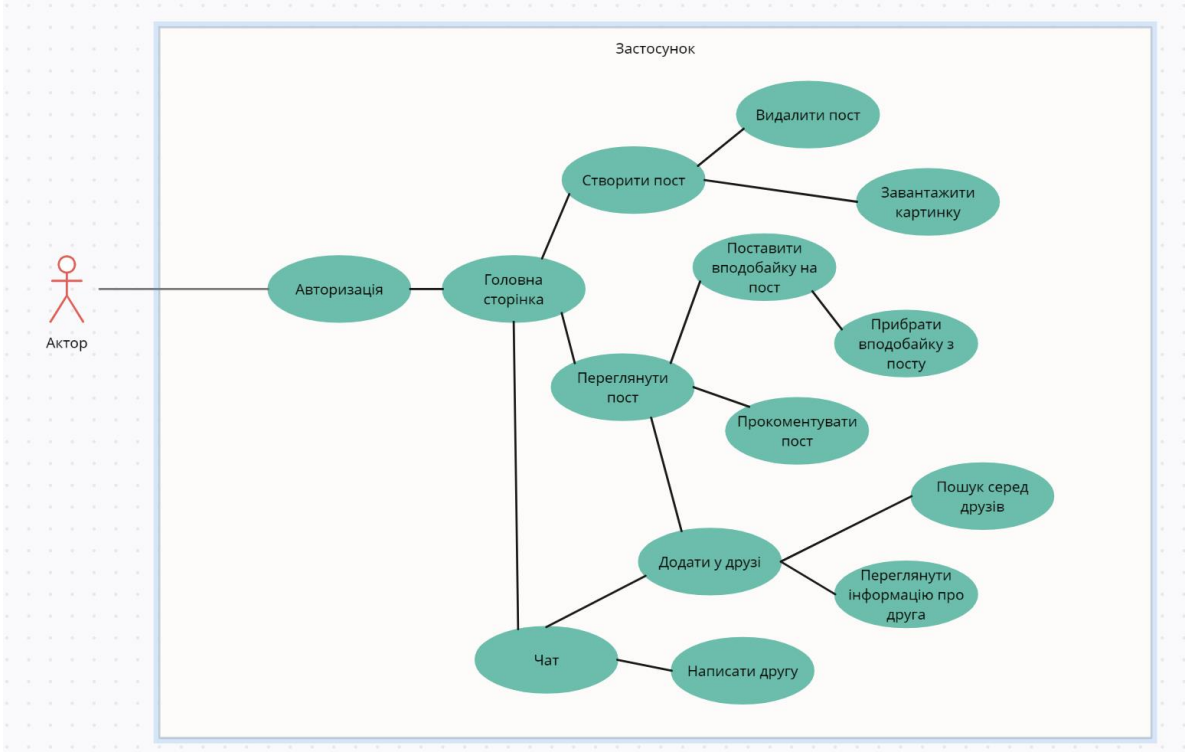

Рисунок 2.5 – Діаграма прецедентів

Ця діаграма прецедентів також відображає важливі аспекти взаємодії користувачів з платформою, такі як можливість завантаження зображень для постів

та їхній подальший перегляд. Важливо відзначити, що система також надає можливість взаємодії з контентом через функціонал вподобайок та коментарів. Це сприяє активному взаємодії користувачів зі змістом, що вони переглядають, та дозволяє виражати свої думки та враження.

2.3. Структура бази даних

Структура бази даних для корпоративної соціальної мережі є критично важливою для забезпечення ефективної роботи системи. Кожна таблиця в базі даних відображає різні аспекти функціональності соціальної мережі і забезпечує надійне зберігання та організацію важливих даних. Наприклад, таблиця "Користувачі" містить інформацію про усіх користувачів платформи, включаючи їхні особисті дані та дані для входу до системи. Таблиця "Пости" забезпечує зберігання текстового вмісту постів, що дозволяє користувачам ділитися своїми думками та ідеями з іншими учасниками мережі.

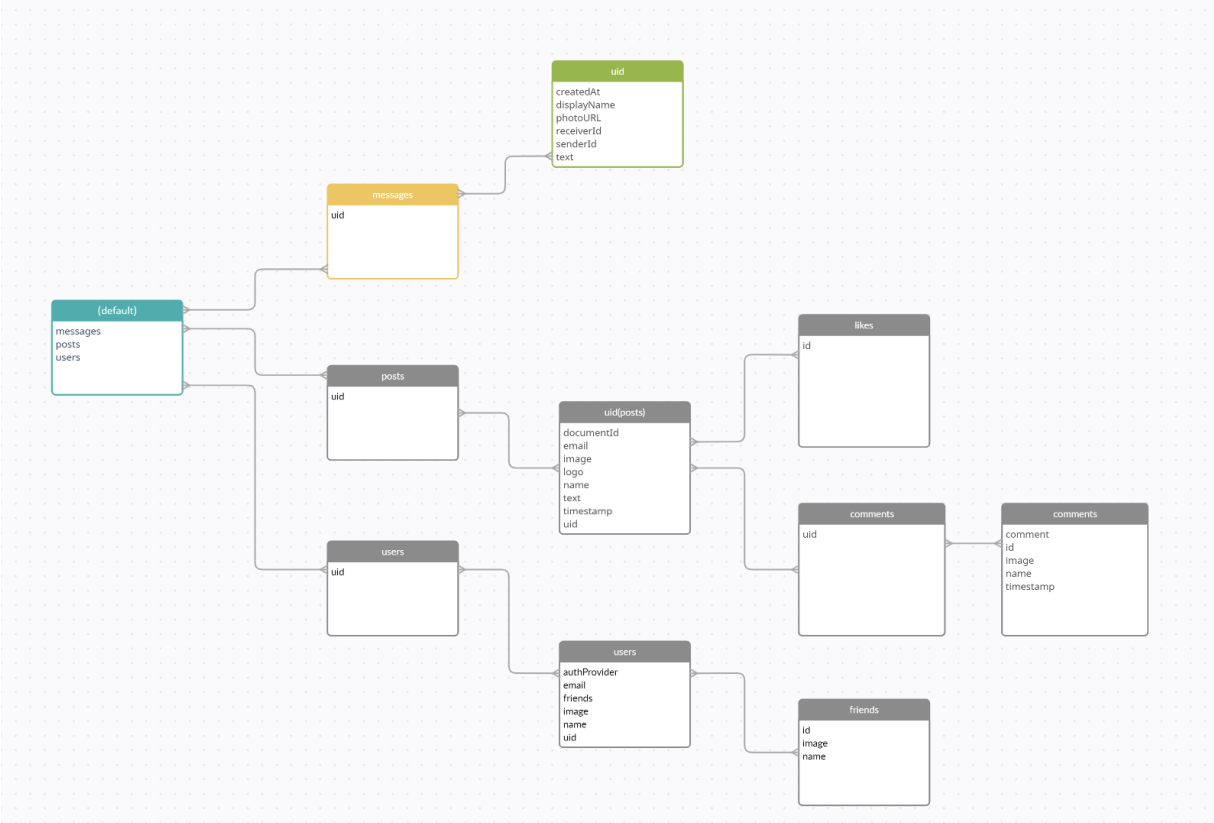

Рисунок 2.6 – Структура бази даних

Висновки до розділу 2

У розділі 2 було проведено моделювання та проєктування програмного забезпечення для корпоративної соціальної мережі в навчально-виховних закладах. Було визначено акторів системи та можливості. Крім того, було створено діаграму варіантів використання, що ілюструє основні функції системи та взаємодію з її користувачами. Найважливішим аспектом роботи є проектування структури бази даних, яка дозволить ефективно зберігати та керувати інформацією про користувачів, пости, коментарі та інші дані, необхідні для функціонування мережі.

У контексті проєктування програмного забезпечення також було узагальнено вимоги до функціональності системи та визначено основні процеси, які вона повинна підтримувати. Планування розробки програмного забезпечення включало в себе розгляд різноманітних аспектів визначення вимог і аналізу потреб користувачів. Результатом цього етапу є чіткий огляд завдань та процесів, що покладаються на програмний продукт, що створюється, а також стратегія їх виконання, що сприяє успішній розробці та впровадженню корпоративної соціальної мережі для навчально-виховних закладів.

3 ВИБІР ТЕХНОЛОГІЙ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОРПОРАТИВНОЇ СОЦІАЛЬНОЇ МЕРЕЖІ

3.1 Вибір технологій реалізації клієнтської частини

Аналіз технологій, спрямованих на розробку сучасних клієнт-серверних застосунків, зосереджується на дослідженні клієнт-серверної архітектури конкретного застосунку. В даному розділі буде проведений детальний аналіз структури та функціоналу застосунку, що базується на клієнт-серверній архітектурі. Розглядатимуться ключові аспекти, такі як розподілення завдань між клієнтом та сервером, взаємодія між ними, а також особливості обробки даних та забезпечення безпеки в рамках цієї архітектури.

При виборі інструментів для розробки корпоративної соціальної мережі для навчально-виховних закладів варто ретельно аналізувати потреби проєкту та специфіку його функціональності. Використання JavaScript та бібліотеки React для фронтенду - це логічний вибір, оскільки ці технології є одними з найбільш поширених та ефективних у сфері веб-розробки на сьогоднішній день. React надає зручні засоби для розробки інтерактивних інтерфейсів, а JavaScript - широкі можливості для реалізації різноманітних функцій та взаємодії з користувачем.

Додатково до вибору JavaScript та React для фронтенду, варто врахувати їхню популярність у спільноті розробників та наявність великої кількості ресурсів, документації та сторонніх бібліотек, що значно спрощує процес розробки. Будучи відкритими та поширеними, JavaScript та React дозволяють легко залучати нових розробників до проєкту, а також швидко реагувати на зміни та вдосконалення.

Технології JavaScript та React також відомі своєю ефективністю та швидкістю роботи. Розробка за допомогою цих інструментів дозволяє створювати швидкі та реактивні веб-додатки, які забезпечують приємний користувацький досвід та високу продуктивність. Крім того, за допомогою різних оптимізацій та інструментів, таких як віртуальний DOM, можна досягти максимальної швидкодії

та ефективності роботи додатку, навіть при великому обсязі даних та складних інтерфейсах.

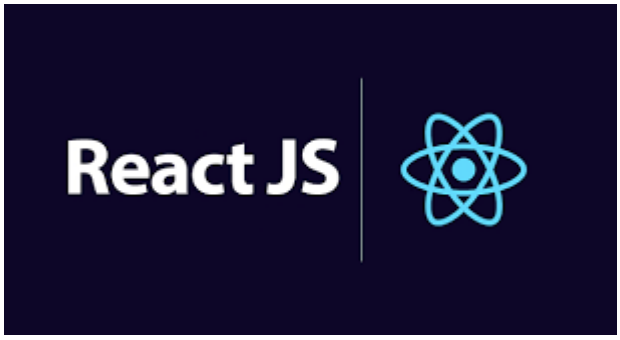

Рисунок 3.1 – Інструменти розрозробки front-end-частини

Основним фокусом цього аналізу буде виявлення та визначення ролі кожного компонента (клієнта та сервера) у забезпеченні функціональності застосунку. Важливим аспектом буде розгляд того, які завдання виконує клієнт, а які – сервер, і як вони взаємодіють для досягнення спільних цілей. Проведений аналіз також охопить розгляд розподілених систем, механізмів комунікації та обміну даними між клієнтом і сервером.

Детальна характеристика принципів взаємодії, використання протоколів та алгоритмів передачі даних буде важливою частиною аналізу. Також будуть розглянуті аспекти безпеки, так як вони мають критичне значення у сучасних застосунках, особливо тих, які працюють з конфіденційною інформацією.

Далі буде проведено порівняння різних типів клієнт-серверних архітектур та їхню придатність для конкретного застосунку. Аналіз ефективності роботи, витрат ресурсів та швидкості обробки запитань визначить оптимальні параметри для вибору конкретного підходу в розробці застосунку.

Такий глибокий аналіз дозволить отримати повний обсяг інформації про клієнт-серверну архітектуру застосунку, що є ключовим етапом перед його подальшим розвитком та впровадженням.

Архітектура клієнт-сервер визначає загальні принципи взаємодії між двома пристроями, при цьому деталі взаємодії конкретизуються мережевими протоколами. Згідно з цією концепцією, мережеві пристрої поділяються на клієнтські, які завжди ініціюють запити, та серверні, які відповідають на ці запити,

надаючи потрібні дані. У цьому взаємодії завжди починає клієнт, і правила, за якими ця взаємодія відбувається, визначаються мережевими протоколами.

Існують два основних типи архітектури взаємодії клієнт-сервер:

1. Двохрівнева архітектура клієнт-серверної взаємодії.
2. Багаторівнева архітектура клієнт-серверної взаємодії, часто відома як трьохрівнева архітектура.

Перевагою клієнт-серверної моделі взаємодії є можливість розподілу програмного коду між двома основними частинами: клієнтським застосунком та сервером. Такий підхід сприяє зниженню системних вимог до обчислювальних машин клієнтів, оскільки більшість операцій відбувається на стороні сервера. З іншого боку, недоліками є високі витрати на якісне мережеве обладнання та необхідність у кваліфікованому адміністраторі.

Існує значна кількість застосунків, розроблених для різних операційних систем, які взаємодіють між собою за допомогою мережевого з’єднання. Наприклад, популярні месенджери, такі як WhatsApp та Viber, встановлюють зв'язок між собою за допомогою мережевого з’єднання, і в основі цього взаємодії лежать сокети.

Сокети, що є інтерфейсом для зв'язку різних пристроїв в одній мережі, існують у двох основних типах: клієнтські та серверні. Основна різниця полягає в тому, що сервер прослуховує вхідні з’єднання і обробляє надходящі запити, тоді як клієнт підключається до сервера. При запуску сервер починає прослуховувати певний порт для отримання вхідних з’єднань, і клієнт, при підключенні, повинен знати IP-адресу сервера та відповідний порт.

Одним із ключових практичних використань сокетів є їх використання для забезпечення засобів комунікації. Крім того, WebSocket представляє собою протокол напівдуплексного зв'язку над TCP-з'єднанням, спрямований на обмін повідомленнями між клієнтом і веб-сервером в реальному часі. Зараз ведеться стандартизація API Web Sockets в W3C, а чернетка стандарту цього протоколу вже отримала затвердження в IETF.

WebSocket виявляється незамінним для розробників, які творять програми з

інтенсивним обміном даними, що вимагають високої швидкості обміну і стабільності каналу. Ілюстрацією можуть служити додатки, що використовують сокет для реалізації своєї бізнес-логіки. Наприклад, системи котирування валют, акцій та біржової статистики забезпечують моніторинг в реальному часі, оновлюючи дані на клієнтських пристроях через сокет-з'єднання. В текстових чатах всі учасники бесіди майже одночасно отримують повідомлення один від одного за допомогою того ж самого з'єднання, незалежно від кількості користувачів.

У стилізації інтерфейсу було обрано Tailwind CSS який має свої переваги. Ця бібліотека надає можливість швидко створювати привабливий та сучасний дизайн, використовуючи готові компоненти та стилізаційні класи. Використання Tailwind спрощує роботу зі стилями та дозволяє швидко реалізувати дизайн, що відповідає сучасним трендам та потребам користувачів.

Tailwind CSS - це потужний інструмент для розробки веб-інтерфейсів, який базується на концепції "utility-first" стилів. Основна ідея Tailwind CSS полягає в тому, щоб використовувати набір малих, незалежних класів для стилізації елементів, замість написання власних CSS правил. Це дозволяє розробникам швидко та ефективно створювати складні інтерфейси, не втрачаючи час на написання і керування CSS кодом.

Однією з переваг Tailwind CSS є його велика кількість готових класів, які включають в себе стилі для різноманітних властивостей, таких як розмір тексту, колір, відступи, вирівнювання та багато інших. Це дозволяє розробникам швидко додавати необхідний функціонал до веб-сайтів без необхідності в пошуку або написанні власних CSS правил.

Крім того, Tailwind CSS надає можливість легко кастомізувати стилі за допомогою конфігураційних файлів, що дозволяє створювати унікальні дизайни, відповідні конкретним потребам проєкту. Велика активна спільнота розробників також забезпечує постійний розвиток та підтримку фреймворку, що робить його надійним інструментом для будь-якого проєкту.

Рисунок 3.2 – Бібліотека для стилізації користувацького інтерфейсу

Далі буде проведено порівняння різних типів клієнт-серверних архітектур та їхню придатність для конкретного застосунку. Аналіз ефективності роботи, витрат ресурсів та швидкості обробки запитань визначить оптимальні параметри для вибору конкретного підходу в розробці застосунку.

Такий глибокий аналіз дозволить отримати повний обсяг інформації про клієнт-серверну архітектуру застосунку, що є ключовим етапом перед його подальшим розвитком та впровадженням.

3.2 Вибір технології реалізації серверної частини

Враховуючи швидкий темп технологічного розвитку, буде проведений огляд новітніх методів та підходів, які використовуються для поліпшення функціональності та продуктивності таких застосунків. Аналіз тенденцій є ключовим етапом для забезпечення конкурентоспроможності та актуальності розроблюваного продукту.

Однією з ключових переваг Firebase є його висока швидкодія та простота використання. Ця платформа забезпечує швидке розгортання серверної частини додатку без необхідності встановлення та конфігурування власного сервера. Firebase також надає широкий спектр сервісів, таких як аутентифікація, база даних реального часу, зберігання файлів, хостинг веб-сторінок та інші, що дозволяє легко і швидко імплементувати різноманітні функціональність та взаємодію з клієнтами.

Вибір Firebase також обумовлений його можливістю масштабування. Платформа автоматично масштабується з ростом навантаження, що дозволяє забезпечити стабільну та надійну роботу додатку навіть при великій кількості користувачів та трафіку. Крім того, Firebase надає безкоштовний тарифний план для початкового рівня, що дозволяє економити кошти на початкових етапах розробки та тестування додатку.

Однією з істотних переваг Firebase є його інтеграція з іншими сервісами Google, такими як Google Analytics, Google Cloud Platform та інші, що відкриває широкі можливості для аналізу даних, моніторингу роботи додатку та розвитку проєкту в цілому. Такий підхід дозволяє отримати повний контроль над розробкою та ефективно використовувати ресурси для досягнення поставлених цілей.

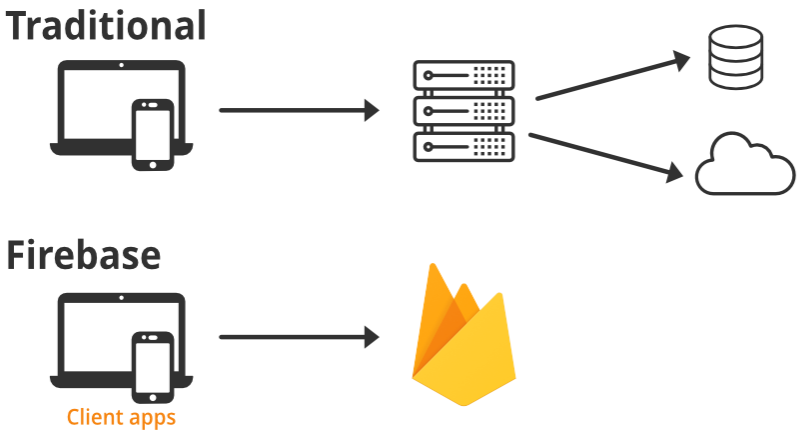

Рисунок 3.3 – Переваги Firebase

Однією з ключових тенденцій у розробці клієнт-серверних додатків є зростання важливості використання хмарних технологій. Хмарні рішення надають можливість ефективного зберігання та обробки даних, а також забезпечують гнучкість та масштабованість застосунків. Розробники все частіше використовують хмарні платформи для розгортання та керування клієнт-серверними додатками, що дозволяє прискорити процес розробки і підтримки.

Ще однією важливою тенденцією є акцент на безпеці та конфіденційності. У зв'язку із зростанням кількості кіберзагроз та порушень безпеки, розробники

зосереджуються на вдосконаленні заходів безпеки в клієнт-серверних додатках. Використання шифрування, механізмів аутентифікації та контролю доступу стає необхідністю для забезпечення захищеності даних та запобігання неправомірному доступу.

Окрім того, відзначається зростання інтересу до розробки додатків для розумних пристроїв та Інтернету речей (IoT). Забезпечення взаємодії між клієнтами та IoT-пристроями стає актуальною задачею, що вимагає розширених можливостей та оптимізованих протоколів.

3.3 Вибір бази даних

Бази даних є основним елементом, що забезпечує зберігання та організацію великого обсягу інформації про студентів, викладачів, навчальні плани, результати екзаменів та інші аспекти навчання. Розглядається роль баз даних у забезпеченні доступу до актуальної та структурованої інформації для всіх учасників навчального процесу.

Однією з важливих функцій баз даних є забезпечення безпеки інформації. Вони використовують різноманітні механізми шифрування та автентифікації для захисту конфіденційності та цілісності даних. Це особливо важливо в контексті обробки особистих даних студентів та зберігання конфіденційної інформації.

Більшість сучасних баз даних володіють розширеними можливостями аналізу даних, що дозволяє проводити детальний моніторинг навчального процесу. Застосування аналітичних інструментів може сприяти виявленню тенденцій, вдосконаленню навчальних програм, а також адаптації до потреб студентів та ринку праці.

Управління мережевим обладнанням та забезпеченням хостингу є складною задачею, і нині набуває популярності концепція Backend as-a-Service (BAAS). Це рішення стало важливим для хмарних обчислень серед розробників та підприємств, які обмежені фінансовими або технічними можливостями для створення та утримання власної інфраструктури. У світі технологій, Google Firebase визнано однією з провідних платформ BAAS на ринку.

Firebase – це платформа для розробки мобільних додатків з величезним функціоналом. Починалася вона як стартап, а сьогодні її використовують при розробці кращих кроссплатформних додатків. Головна перевага платформи в тому, що вона дозволяє розробнику не відволікатися на створення backend, тобто прихованої від користувача програмної частини проєкту, наприклад, серверного коду. І це спрощує і прискорює створення мобільних додатків, дає можливість повністю зосередитися саме на UX / UI, тобто, на призначеному для користувача інтерфейсі і досвіді. Саме зв'язка Firebase з фреймворком Flutter дозволяє програмістам компанії AVADA MEDIA створювати швидкі програми для Android і iOS, які вирішують найрізноманітніші завдання.

Використання рішень BAAS усуває необхідність в самостійному управлінні базами даних для резервного копіювання та придбання відповідного мережевого обладнання. Замість цього вони надають можливість підключення сервісу до створеного додатку за допомогою спеціальних API для кожної конкретної послуги. Firebase, наприклад, пропонує 7 таких API, що охоплюють широкий спектр бекенд-технологій для різноманітних додатків на різних платформах, таких як Android, IOS, Web та Unity.

Firebase Realtime Database виступила піонером серед продуктів Firebase і залишається найбільш визнаним та стабільним сервісом на цій платформі. Ця база даних в реальному часі представляє собою хмарне сховище NoSQL, яке можна легко інтегрувати у програми для забезпечення миттєвого доступу до даних на різних платформах. Однією з переваг цього сервісу є можливість автономної роботи бази даних, кешуючи дані на пристрої та автоматично синхронізуючи їх при підключенні до Інтернету.

Дані в Firebase Realtime Database зберігаються у форматі JSON, що дозволяє користувачам зручно отримувати доступ до них. З точки зору безпеки, цей сервіс забезпечує контроль доступу до даних на основі налаштованих дозволів. Цю функціональність можна реалізувати за допомогою аутентифікації Firebase та встановлення прав доступу на основі ідентифікаторів користувачів або правил безпеки.

Рисунок 3.4 – Приклад представлення даних у базі даних

Cloud Firestore представляє собою ще одну хмарну базу даних NoSQL в режимі реального часу. На відміну від Firebase Realtime Database, Cloud Firestore спроєктований з урахуванням корпоративного використання, що призводить до забезпечення масштабованості, складних моделей даних і розширених можливостей запитів. Консоль Firebase може служити інструментом для перегляду даних у обох базах даних. Ще однією схожістю є наявність SDK для роботи з кодом на стороні сервера для обох баз даних, які підтримують Python, Node.js, Golang, Ruby, PHP, Java, .NET та C#.

Аутентифікація Firebase - це функція аутентифікації Google, спеціально розроблена для програм, які використовують Firebase. Ця функція дозволяє використовувати вбудований інтерфейс або створювати власний для аутентифікації

користувачів. Користувачі можуть увійти в систему за допомогою користувацьких облікових даних, електронної пошти або облікових записів у соціальних мережах.

Послуги хостингу. У випадку розробки прогресивного веб-додатка або мобільної цільової сторінки, невід'ємною частиною буде хостинг. Firebase пропонує статичний веб-хостинг для додатків, створених з використанням HTML, CSS та JavaScript. З точки зору безпеки, він використовує стандартні протоколи HTTPS і SSL для доставки файлів та інших типів даних.

Хмарні функції. Це інтеграція існуючого продукту Google в Firebase. Хмарні функції дозволяють запускати бек-код з хмари на основі подій. Такий підхід називається архітектурою без сервера і передбачає побудову додатків як набору окремих функцій, ізольованих у хмарі та з'єднаних між собою за допомогою API. Google визнається одним із провідних постачальників безсерверних рішень.

Test Lab - це послуга, яка надає віртуальні та фізичні пристрої для тестування додатка в реальному середовищі. Ця послуга може бути інтегрована в існуюче тестове середовище, таке як Android Studio, або в інструменти тестування браузера. Test Lab генерує звіти про збої та надає скріншоти як результати тестів. Також, якщо відсутні тести, використовується бот, який автоматично сканує програму, виявляє помилки та генерує звіти. Однак слід зауважити, що Test Lab не інтегрується з інструментами для тестування iOS.

Моніторинг продуктивності впроваджує автоматизацію процесу відслідковування ефективності програми в реальному часі. Цей інструмент надає докладну інформацію про продуктивність на основі кореневих причин, аналізуючи не лише саму продуктивність програми, але й якість підключення до сервера та час відповіді для різних типів мереж.

Google Analytics визначається як провідний інструмент у цьому напрямку, оскільки він є відомим інтегрованим засобом на платформі Firebase. Google Analytics забезпечує корисні показники стосовно утримання користувачів, активності користувачів та інших аспектів їхньої поведінки. Щодо Firebase, відзначається безкоштовністю та необмеженими можливостями звітності.

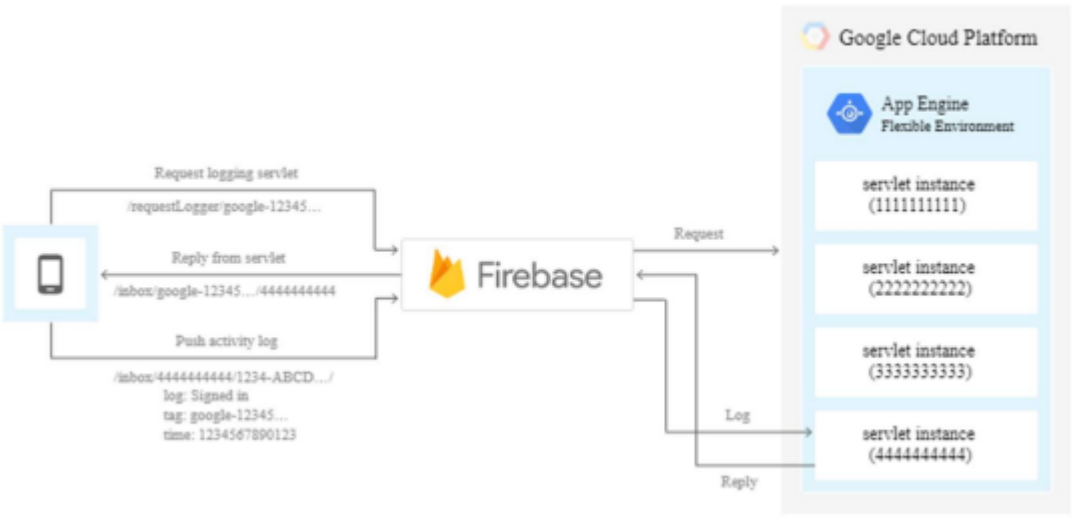

Рисунок 3.5 – Приклад використання бази даних

Вільний старт: По-перше, використання Firebase не вимагає оплати за більшість своїх послуг та є абсолютно безкоштовним. Це дозволяє вам оцінити, чи відповідає він вашим потребам, розібратися у всіх його особливостях. Коли вам потрібно більше простору бази даних або певні додаткові сервіси, завжди можна обрати відповідний план. Сторінка цін включає калькулятор, який можна налаштовувати за різними параметрами, що є стандартною практикою для хмарних сервісів.

Коротка документація: Висока якість технічної документації, API та посилання на SDK роблять продукт більш доступним та легким у використанні для користувача. При вивченні сторінки продуктів Firebase можна помітити, що вона містить всю необхідну інформацію щодо інтеграцій, доступних платформ, інструкцій та списків підтримуваних технологій. Крім того, є можливість ознайомитися з каналом YouTube Firebase, який виглядає досить активним щодо випуску нових відео та оголошень.

Доступний інтерфейс користувача та простота інтеграції: У більшості випадків використання Firebase не вимагає глибоких знань мов програмування і пропонує інтеграцію через свій простий у використанні користувацький інтерфейс.

Крім зазначених переваг Firebase, існує ще декілька сервісів, які можуть

запропонувати схожі можливості:

AWS Aurora: Основною перевагою сервісу AWS Aurora у сфері Backend as a Service (BaaS) порівняно з Firebase є використання реляційної бази даних, що гарантує більшу структуру даних та ефективні запити. Цей сервіс спеціально розроблений для корпоративних програм і забезпечує підтримку міграції даних між різними платформами. Таким чином, цей аспект може мати перевагу в разі необхідності високої технічної зрілості та підтримки міграцій.

Parse Server: Хоча ера платформи Parse завершилася в 2017 році, все ще доступна її версія з відкритим кодом. Серед переваг можна відзначити багато функцій перед встановленням, загальну конфігурацію та можливість вибору середовища для розміщення. З іншого боку, його інтерфейс командного рядка є досить нестабільним, а підтримується спільнотою розробників, а не великою корпорацією.

Back4App: Це реалізація Parse Server, яка в основному виправила його недоліки. Back4App пропонує аналогічні функції, що виконує Firebase, за винятком більш гнучкого моделювання даних та налаштувань запитів до бази даних.

3.4. Додаткові бібліотеки для реалізації функціоналу та користувацького інтерфейсу

Додаткові бібліотеки грають важливу роль у розширенні функціоналу та поліпшенні користувацького інтерфейсу веб-додатків. При виборі таких бібліотек слід керуватися конкретними потребами та вимогами проєкту. Ось кілька типових бібліотек, які можна використовувати для покращення функціоналу та інтерфейсу корпоративної соціальної мережі для навчально-виховних закладів:

1. formik та yup: Formik - це бібліотека для управління формами в React, а yup - для валідації даних форм, що допомагає забезпечити правильну та безпечну обробку введених користувачем даних.
2. firebase та firebase-admin: Firebase інструмент, який дозволяє легко зберігати та керувати даними в реальному часі, автентифікувати користувачів та реалізовувати інші функції бекенду без необхідності у власному сервері.

3. react-router-dom: Ця бібліотека дозволяє вам легко керувати маршрутами вашого веб-додатку, що робить навігацію між сторінками зручною та ефективною.
4. react-slick та slick-carousel: Ці бібліотеки надають інструменти для створення каруселей та слайдерів, що може бути корисним для відображення контенту та зображень у вашому додатку.
5. @emotion/react та @emotion/styled: Ці бібліотеки дозволяють використовувати emoji CSS для стилізації компонентів React.
6. @material-tailwind/react та @mui/material: Ці бібліотеки містять готові компоненти для інтерфейсу користувача, що допоможуть вам швидко розробити стильний та функціональний дизайн вашого додатку.
7. web-vitals: Ця бібліотека допомагає вимірювати та аналізувати різні метрики про продуктивність вашого веб-додатку, такі як час завантаження сторінки, час рендерингу та інші. Вона дозволяє вам вчасно виявляти та виправляти проблеми продуктивності для забезпечення оптимального користувацького досвіду.
8. Бібліотека @mui/material (Material-UI) є однією з найпопулярніших інструментів для розробки користувацького інтерфейсу в React-додатках. Вона базується на дизайн-мові Material Design від Google і надає готові компоненти, які допомагають створювати стильні та сучасні інтерфейси. Material-UI пропонує широкий спектр компонентів, включаючи кнопки, текстові поля, таблиці, картки, модальні вікна та багато іншого. Ця бібліотека дозволяє розробникам ефективно будувати UI, зосереджуючись на функціональності додатку.

Ці додаткові бібліотеки значно полегшують розробку та підтримку додатків, роблять їх більш надійними та зручними для користувачів. Вони доповнюють функціональність основних інструментів та роблять процес розробки більш продуктивним.

Рисунок 3.6 – Конфігураційний файл залежності проєкту

Висновок до розділу 3

У розділі 3 було проведено ретельний аналіз різних технологій з метою вибору найбільш підходящих для реалізації клієнтської та серверної частин програмного забезпечення корпоративної соціальної мережі. Враховуючи специфіку завдань та потреб користувачів, були визначені оптимальні інструменти та технології, які забезпечать найвищу продуктивність, безпеку та зручність

використання системи. Крім того, було ретельно вивчено різні варіанти баз даних та додаткових бібліотек для забезпечення різноманітності функціоналу та користувацького інтерфейсу. Результатом цього етапу є чітко визначені технологічні стеки та інструменти, які будуть використані для розробки програмного забезпечення, що забезпечить ефективне та надійне функціонування корпоративної соціальної мережі для навчально-виховних закладів.

На основі результатів вибору технологій та аналізу потреб користувачів було визначено оптимальні методи реалізації програмного забезпечення. Вибір конкретних технологій і бібліотек для клієнтської та серверної частин програми, а також бази даних, здійснювався з урахуванням їхньої сумісності, швидкодії, масштабованості та можливості інтеграції з іншими системами. В результаті цього етапу розробки вдалося знайти оптимальний баланс між технічними можливостями та вимогами функціональності, що дозволить створити сучасну та ефективну соціальну мережу для навчально-виховних закладів, сприяючи покращенню комунікації та співпраці в освітньому середовищі.

4 РОЗРОБКА КОРПОРАТИВНОЇ СОЦІАЛЬНОЇ МЕРЕЖІ ДЛЯ НАВЧАЛЬНИХ ЗАКЛАДІВ

4.1 Створення проєкту та підключення до Firebase

Перший крок у створенні проєкту - це створення нового застосунку React. Для цього ми використовуємо команду npx create-react-app, яка автоматично налаштує заготовку даного проєкту React з усіма необхідними налаштуваннями та залежностями. Після запуску цієї команди ми вказуємо ім'я проєкту, наприклад, social-media-app, і зачекаємо, поки процес завершиться.

Після створення проєкту переходимо до підключення до Firebase. Для цього знадобиться конфігураційний об'єкт Firebase, який можна отримати з консолі Firebase. Тут ми вказуємо ключі та параметри даного проєкту Firebase, які будуть використовуватися для з'єднання зі збереженими даними в хмарі. Після цього ми встановлюємо Firebase SDK у проєкт, додаючи відповідні пакети та налаштовуючи його для взаємодії з нашим React застосунком.

Після успішного підключення до Firebase можна почати розробку застосунку, використовуючи всі можливості, які надає ця платформа. Firebase пропонує широкий набір інструментів, таких як база даних Firestore для збереження даних, автентифікація для управління користувачами, зберігання файлів у хмарі, сервіси аналізу даних та інші.

Для створення бази даних в Firebase використано сервіс Firestore, який забезпечує нам можливість зберігати, керувати та синхронізувати дані між різними клієнтами. Перш ніж розпочати роботу з Firestore, ми повинні створити базу даних в консолі Firebase та налаштувати правила доступу до даних відповідно до наших потреб та безпеки.

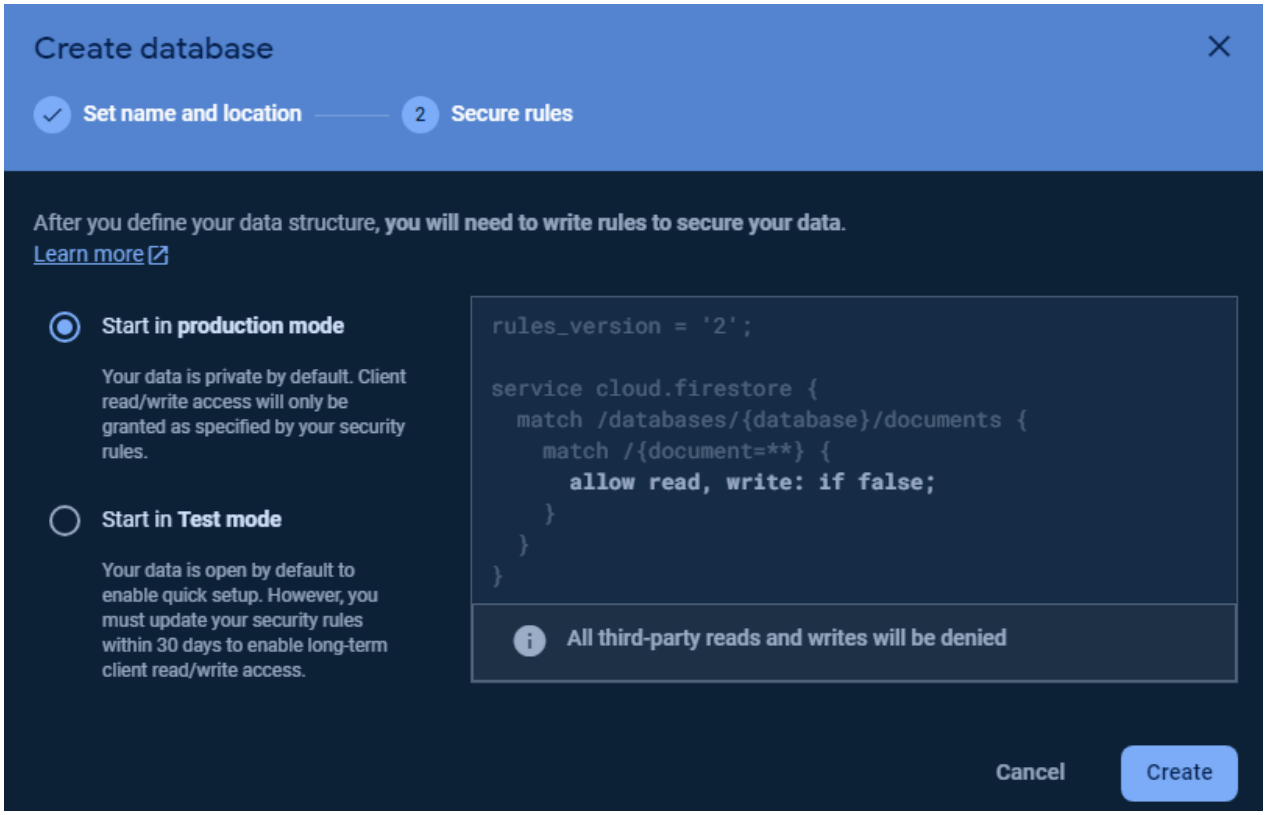

Рисунок 4.1 – Створення бази даних в сервісі Firestore

Після створення бази даних ми можемо почати визначати колекції та документи, які будуть зберігати нашу інформацію. У базі даних Firebase створено три основні колекції для зберігання різноманітної інформації, що відповідає потребам застосунку. Ось короткий огляд кожної з них:

1. Колекція "messages": Ця колекція призначена для зберігання повідомлень, які відправляють користувачі один одному. Кожне повідомлення може містити різну інформацію, таку як текстовий контент, фотографії, відео тощо. Для кожного повідомлення можуть бути визначені такі поля, як ID автора, дата та час відправлення, вміст повідомлення тощо.
2. Колекція "users": У цій колекції ми зберігаємо інформацію про користувачів нашого застосунку. Кожен документ у цій колекції представляє окремого користувача та містить такі дані, як ім'я, електронна пошта, профільне зображення, дата реєстрації тощо. Ці дані дозволяють нам ідентифікувати користувачів та забезпечити їм персоналізований досвід використання застосунку.

3. Колекція "posts": У цій колекції зберігається інформація про публікації, які роблять користувачі. Кожен документ у цій колекції представляє окремий пост і містить такі дані, як автор посту, текстовий контент, зображення, кількість лайків, коментарі тощо. Ці дані дозволяють нам відображати публікації користувачів у стрічці новин та забезпечити можливість взаємодії з ними.

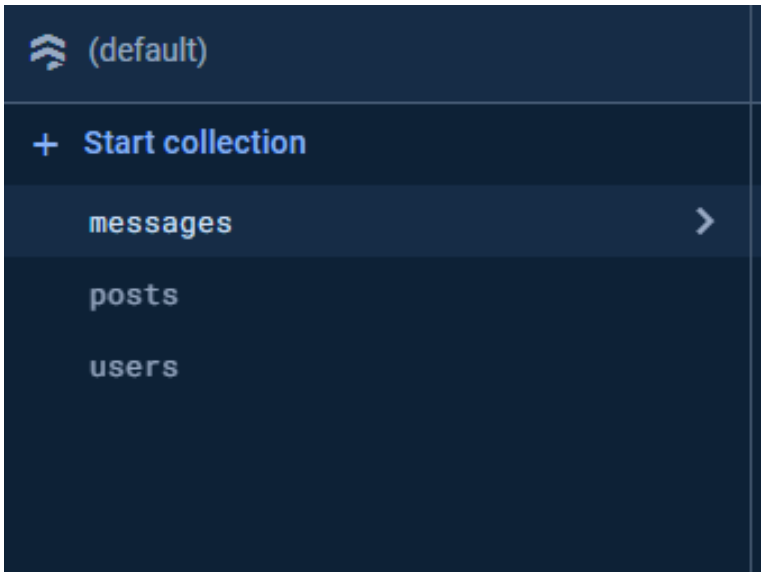

Рисунок 4.2 – База даних Firestore Database

4.2 Реалізація авторизації користувачів

У проєкті реалізація авторизації користувачів є критичною складовою, оскільки вона забезпечує безпеку та захист особистої інформації користувачів. Для цього використано Firebase Authentication, яка надає надійний механізм аутентифікації з використанням різних провайдерів, таких як електронна пошта та пароль, Google, Facebook, Twitter тощо.

Код наведено у додатку в представляє компонент React для сторінки авторизації користувача. Його основні складові:

1. Імпорт компонентів з бібліотеки @material-tailwind/react: Card, CardHeader, CardBody, CardFooter, Typography, Input, Button. Ці компоненти використовуються для оформлення візуальної частини сторінки.

2. Імпорт бібліотек та хуків з React та інших джерел: useFormik з бібліотеки Formik, Yup для валідації форм, AuthContext з контексту додатка, auth та onAuthStateChanged з Firebase для автентифікації користувачів, useNavigate для навігації між сторінками, ClipLoader для відображення загрузки.
3. Визначення компоненту Login, який буде експортуватися за замовчуванням.
4. Внутрішній стан loading, який відповідає за відображення статусу завантаження.
5. Використання ефекту useEffect для перевірки авторизації користувача при завантаженні сторінки. Це робиться шляхом слідкування за змінами статусу авторизації за допомогою onAuthStateChanged.
6. Визначення початкових значень та схеми валідації за допомогою Formik та Yup.
7. Функція handleSubmit, яка викликається при надсиланні форми. Вона перевіряє валідність введених даних, а потім викликає функцію loginWithEmailAndPassword для входу користувача за допомогою електронної пошти та пароля.
8. Виклик useFormik з визначеними раніше значеннями початкових даних та схемою валідації.

Цей компонент відповідає за відображення форми входу користувача та взаємодію з Firebase для авторизації. Його стан і логіка допомагають забезпечити коректну роботу процесу авторизації та зручний інтерфейс для користувачів.

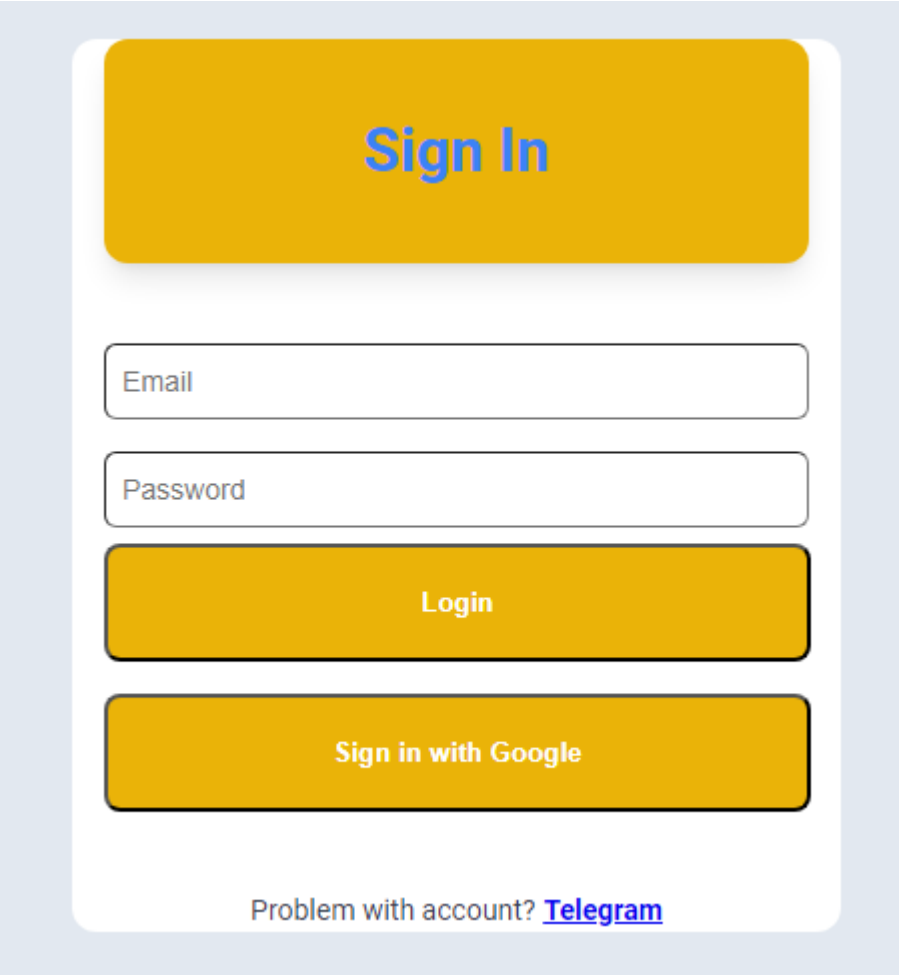

Рисунок 4.3 – Інтерфейс авторизації

Для авторизації обов'язково потрібно ввести коректну електронну пошту та пароль. В цьому компоненті форми входу користувача, валідація проводиться з використанням бібліотеки Yup, яка перевіряє, чи введені дані відповідають встановленим вимогам. У випадку невідповідності вимогам, користувач буде проінформований про помилку, і форма не буде надіслана до сервера для авторизації. Таким чином, забезпечується безпека та коректність процесу авторизації.

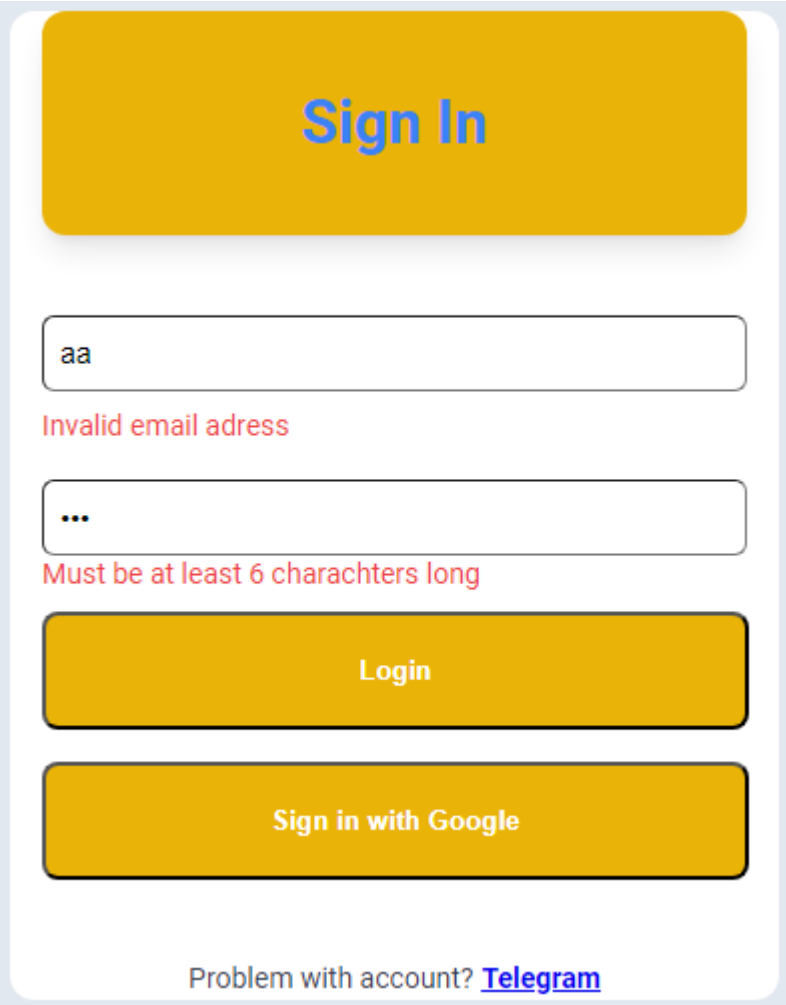

Рисунок 4.4 – Інтерфейс авторизації за коректними вимогами

Кожен користувач має свій власний запис в базі даних, де зберігаються його дані. При реєстрації нового користувача, його дані, такі як ім'я, електронна пошта та інші, які він надав, записуються в базу даних. Це дозволяє системі ідентифікувати користувача при подальших входах та взаємодії з системою. Кожен користувач має унікальний ідентифікатор, який використовується для доступу до його даних у базі даних. Така організація бази даних дозволяє ефективно керувати користувачами та забезпечити безпеку їхніх персональних даних.

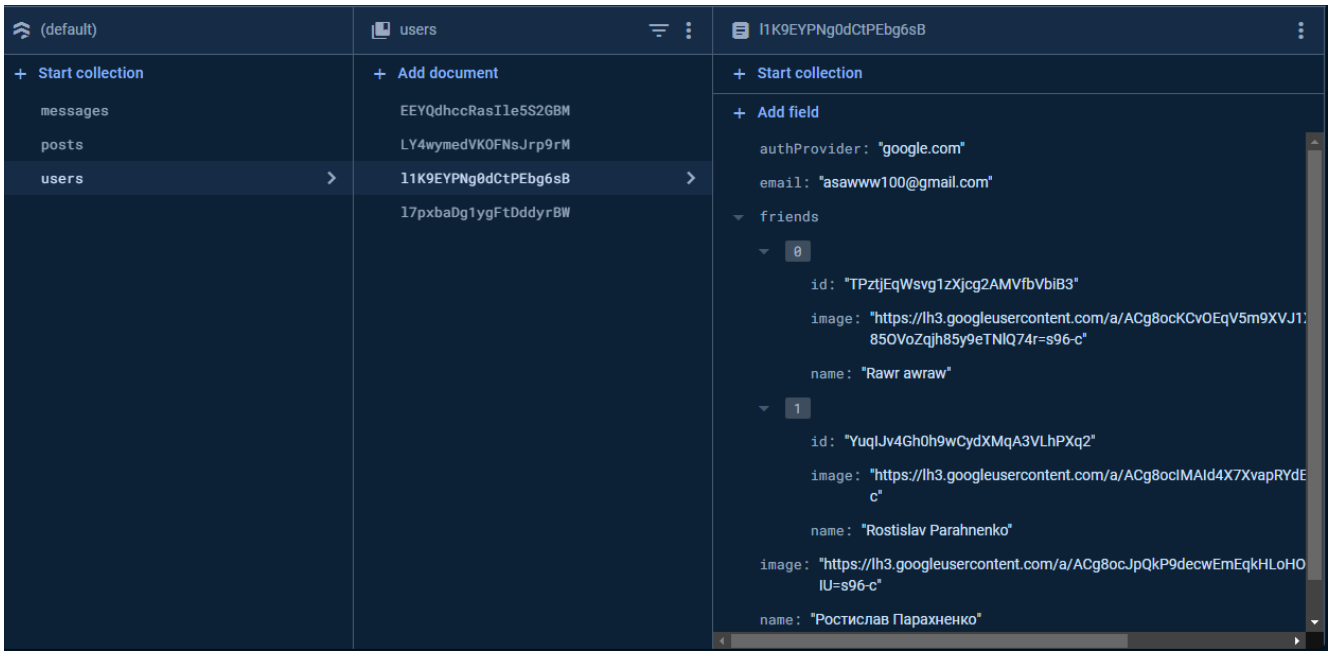

Рисунок 4.5 – Дані користувача у базі даних

Firebase надає зручний інтерфейс для керування користувачами та їх автентифікацією через вкладку Authentication. Тут можна переглядати та керувати списком зареєстрованих користувачів, перевіряти їх статус автентифікації та виконувати інші дії, пов'язані з управлінням користувачами. Це дозволяє зручно відслідковувати та керувати користувачами безпосередньо з інтерфейсу Firebase, що спрощує процес розробки та управління.

Рисунок 4.6 - Інтерфейс для керування користувачами

4.3 Реалізація головної сторінки

Після успішної авторизації користувача потрібно перенаправити його на головну сторінку додатку. На цій сторінці можна відобразити основний контент застосунку, який може включати такі елементи, як стрічка новин, можливість створення та перегляду постів, а також функціонал для спілкування та взаємодії з іншими учасниками мережі.

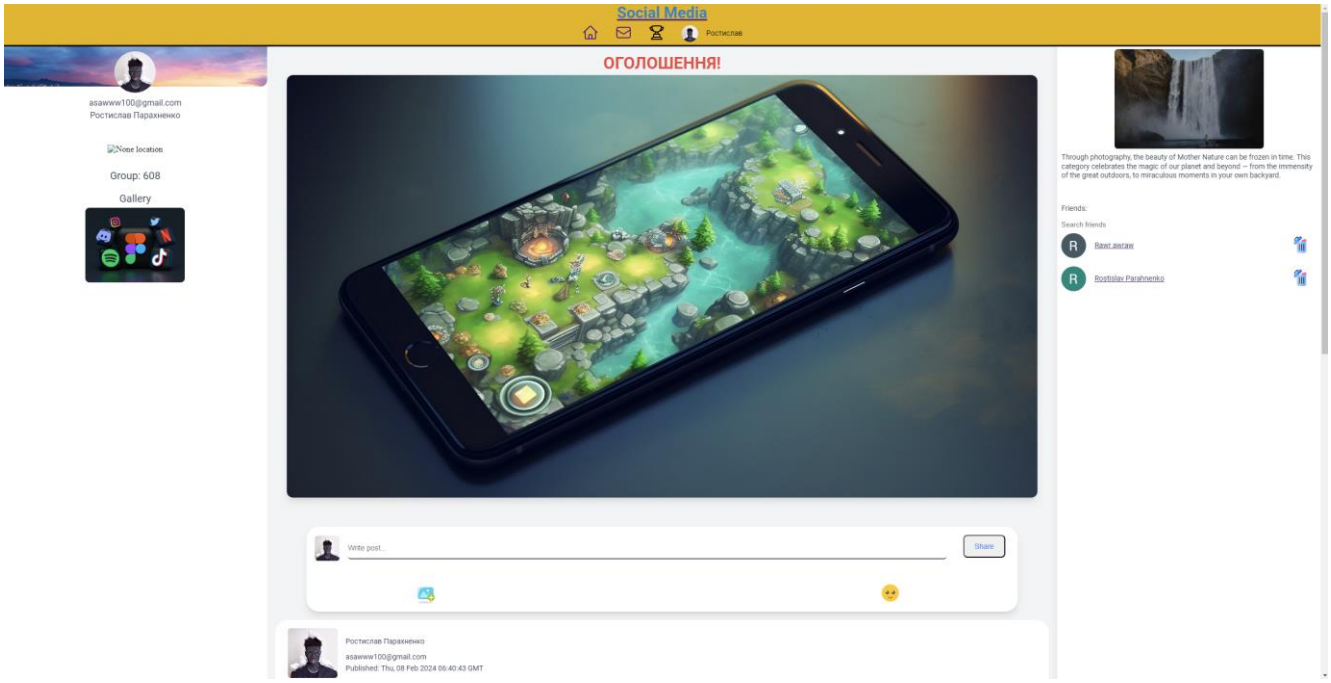

Рисунок 4.7 – Головна сторінка застосунку

На головній сторінці може бути розділ з оголошеннями, який доступний для перегляду всім користувачам, але редагування цих оголошень може бути доступним лише адміністраторам застосунку. Це забезпечить контрольований доступ до змін у важливому контенті і дозволить забезпечити актуальність та достовірність інформації, що публікується.

Ліва частина застосунку відведена для відображення інформації про профіль користувача. Це може включати особисті дані, такі як ім'я, електронна адреса, зображення профілю тощо. Крім того, ця область містить зображення що змінюються, таким чином можна відобразити нагороди, що були присвоєні під час проведення конкурсів в навчальних закладах. В додатку в наведено функціональність.

Функціональний компонент React, який використовує хук стану useState для зберігання масиву data, що ініціалізується порожнім масивом за замовчуванням. Також ми використовуємо useRef для створення змінної count, яка зберігатиме значення лічильника, але не буде спричиняти повторне викликання рендера компонента.

У функціональному компоненті ми також маємо доступ до значень user і userData через контекст AuthContext.

Функція handleRandom приймає масив arr та встановлює значення стану data на випадковий елемент з цього масиву.

У useEffect ми маємо початкове значення масиву imagelist, який містить об'єкти з ідентифікаторами та шляхами до зображень. Потім ми викликаємо handleRandom один раз для встановлення початкового значення зображення.

Далі ми використовуємо setInterval, щоб автоматично змінювати зображення кожні 2 секунди протягом 5 ітерацій. Після кожної ітерації змінюється значення count.current, яке використовується для перевірки кінця циклу. У випадку досягнення 5 ітерацій ми зупиняємо інтервал за допомогою clearInterval.

Функція, яка передається у useEffect, повертає функцію очищення, яка видаляє інтервал перед відмонтуванням компонента, щоб уникнути витоку пам'яті.

Отже, цей код використовується для автоматичної зміни зображень кожні 2 секунди протягом обмеженої кількості ітерацій.

{user?.email || userData?.email} використовується для відображення електронної пошти користувача. Він перевіряє, чи існує об'єкт користувача user або userData, і відображає електронну пошту, якщо вона доступна. Якщо об'єкт user містить інформацію про користувача, він використовує електронну пошту, яка зазвичай доступна через user.email. Якщо немає інформації в user, він шукає в userData, де можуть зберігатися додаткові дані користувача, наприклад, якщо вони були завантажені з бази даних.

У випадку, якщо електронна пошта користувача не знайдена ні в user, ні в userData, буде відображено null. Додавши **?.**, ми уникнемо помилки, якщо user або

userData будуть нульовими або невизначеними. Це спрощує роботу з об'єктами, що можуть мати властивості, які можуть бути null або undefined.

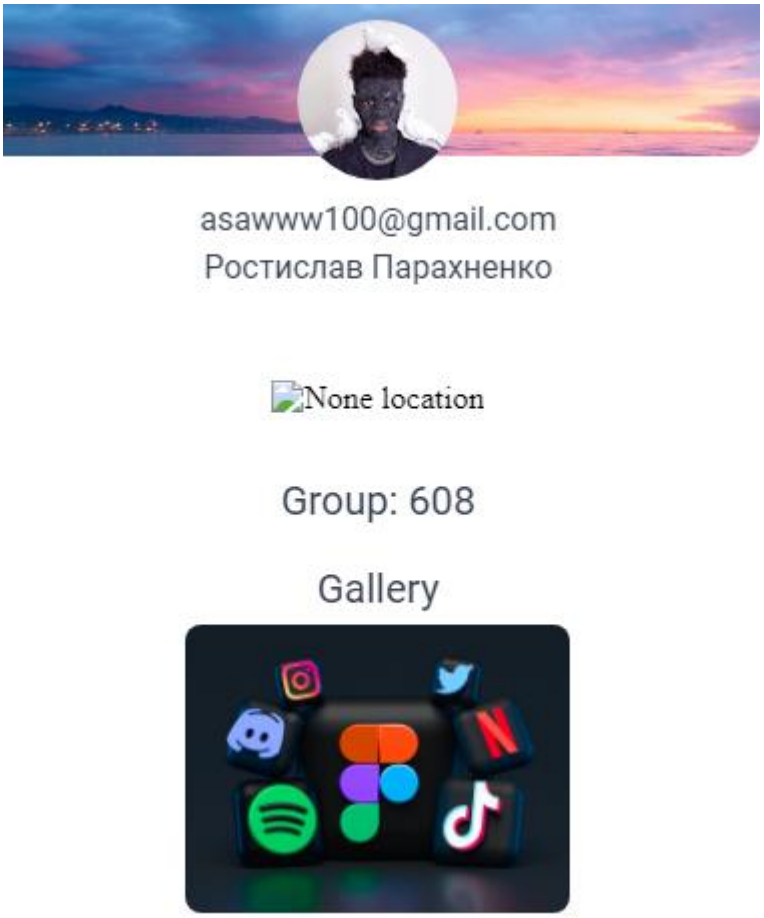

Рисунок 4.8 – Ліва частина застосунку

Для створення постів у нашому застосунку можна використовувати функціонал додавання нового запису до відповідної колекції бази даних, наприклад, колекції "posts" в Firebase Firestore. При цьому, кожен пост може мати унікальний ідентифікатор, дату створення, автора (ідентифікатор користувача або ім'я користувача) та зміст поста (текст, фото, відео тощо).

Для керування станом використано PostReducer для зберігання та маніпулювання даними, що стосуються постів у нашому застосунку.

У PostReducer визначені дії (actions), такі як додавання нового поста, видалення поста, редагування поста, додавання коментарю, та вподобання. Кожна дія може мати відповідний тип та дані, необхідні для виконання операції з постом.

Константи postActions визначають різні типи дій, такі як надсилання нового поста (SUBMIT_POST), додавання лайку до поста (ADD_LIKE), додавання коментаря до поста (ADD_COMMENT) та обробка помилки (HANDLE_ERROR). Вони використовуються для ідентифікації типу дії у PostsReducer.

postsState - це об'єкт, який містить початковий стан для постів у застосунку. У цьому об'єкті визначені властивості, такі як error, posts, likes та comments, які відображають стан помилок, списку постів, списку лайків та списку коментарів відповідно.

PostsReducer - це функція, яка приймає поточний стан і дію, та повертає новий стан на основі виконаної дії. У цій функції використовується оператор switch, щоб визначити, яку дію потрібно виконати в залежності від типу дії. Наприклад, при отриманні дії SUBMIT_POST, стан постів оновлюється згідно з надісланими постами. В інших випадках, якщо дія не визначена або відбувається обробка помилки, повертається поточний стан без змін.

Для постів було розширино можливості, дозволяючи користувачам додавати картинки та емодзі. Це дозволяє їм зробити свої повідомлення більш виразними та емоційними. Для цього ми використали спеціальні компоненти та бібліотеки, які надають зручний інтерфейс для вибору та вставки зображень та емодзі. Користувачі можуть легко вибирати картинки або емодзі та додавати їх до своїх постів, щоб краще виразити свої думки та почуття. Це робить наші пости більш цікавими та привабливими для аудиторії, сприяючи активнішій взаємодії та обміну враженнями.

Для зберігання зображень у базі даних було реалізовано використання сервісу зберігання Firebase - Firebase Storage. Цей сервіс надає можливість зберігати та керувати медіа-файлами, такими як зображення, аудіо та відео. Після завантаження зображення користувачем, воно зберігається у Firebase Storage, а посилання на нього зберігається у відповідному об'єкті посту в базі даних Firestore. Це дозволяє забезпечити ефективне та безпечне зберігання медіа-контенту, забезпечуючи швидкий доступ до нього для відображення в інтерфейсі додатку.

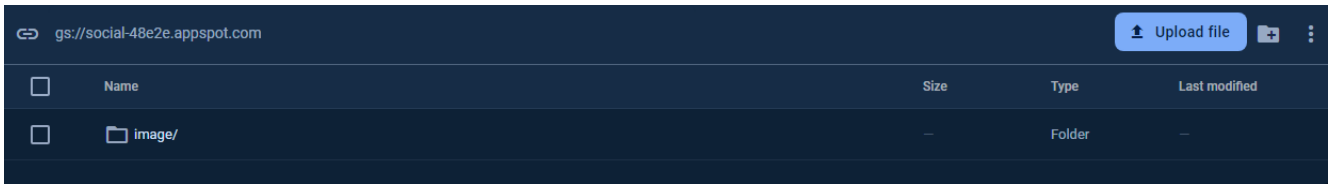

Рисунок 4.9 – Storage у базі даних для зберігання картинок

В додатку в наведено, ініціалізування об'єкту зберігання Firebase, використовуючи функцію getStorage(). Далі вказується метадані для завантажуваних файлів. В конкретному випадку, встановлюється, що тип вмісту може бути файлами форматів jpeg, jpg, png, gif, та svg. Це обмеження допомагає забезпечити, що в Firebase Storage будуть завантажуватися лише файли відповідних форматів.

В додатку в наведено стан file, який зберігатиме завантажений файл. Функція handleUpload встановлює цей файл, обираючи перший файл зі списку файлів, який вибирає користувач через інтерфейс вибору файлу у компоненті.

Функція submitImage відповідає за завантаження обраного файлу на сервер Firebase Storage та збереження його в базі даних. Перевіряється тип файлу, щоб переконатися, що він відповідає одному з підтримуваних типів. Після цього створюється посилання на файл у сховищі Firebase, і сам файл завантажується за допомогою функції uploadBytesResumable. Прогрес завантаження відображається за допомогою обробника подій 'state_changed'. Після успішного завантаження зображення, його посилання отримується за допомогою функції getDownloadURL і зберігається для подальшого використання. У разі помилки під час завантаження файлу відображається сповіщення про помилку, а також відповідний стан помилки оновлюється за допомогою диспетчера дій.

Стан progressBar використовується для відстеження прогресу завантаження файлу на сервер Firebase Storage. Він оновлюється під час завантаження, щоб відображати користувачеві поточний прогрес завантаження файлу.

Ці стани використовуються для управління введеним текстом користувачем та вибраним емодзі. emojiPicker визначає, чи показувати вікно вибору емодзі, а postText зберігає текст посту, який користувач вводить. Функція handleEmojiSelect

викликається при виборі емодзі користувачем та додає його до поточного тексту посту.

Після встановлення всіх необхідних елементів, таких як текстове поле для введення посту та кнопка для відправлення, користувач може ввести свій пост та вибрати потрібні емодзі. Після цього він натискає кнопку "Відправити", щоб опублікувати свій пост.

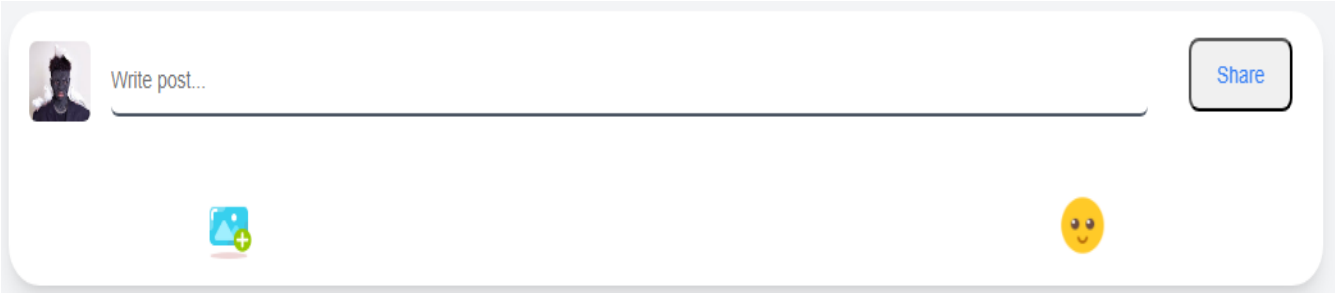

Рисунок 4.10 – Інтерфейс відправлення постів

Після відправлення поста він зберігається в базі даних разом із вибраною картинкою, текстом, інформацією про вподобання, коментарями та іншими деталями. Кожен пост також містить дані про автора, час публікації та іншу важливу інформацію. Користувач може переглянути цю інформацію, прокоментувати або вподобати пост, якщо ця можливість доступна. Також автор посту має можливість видалити його, якщо він вважає це необхідним.

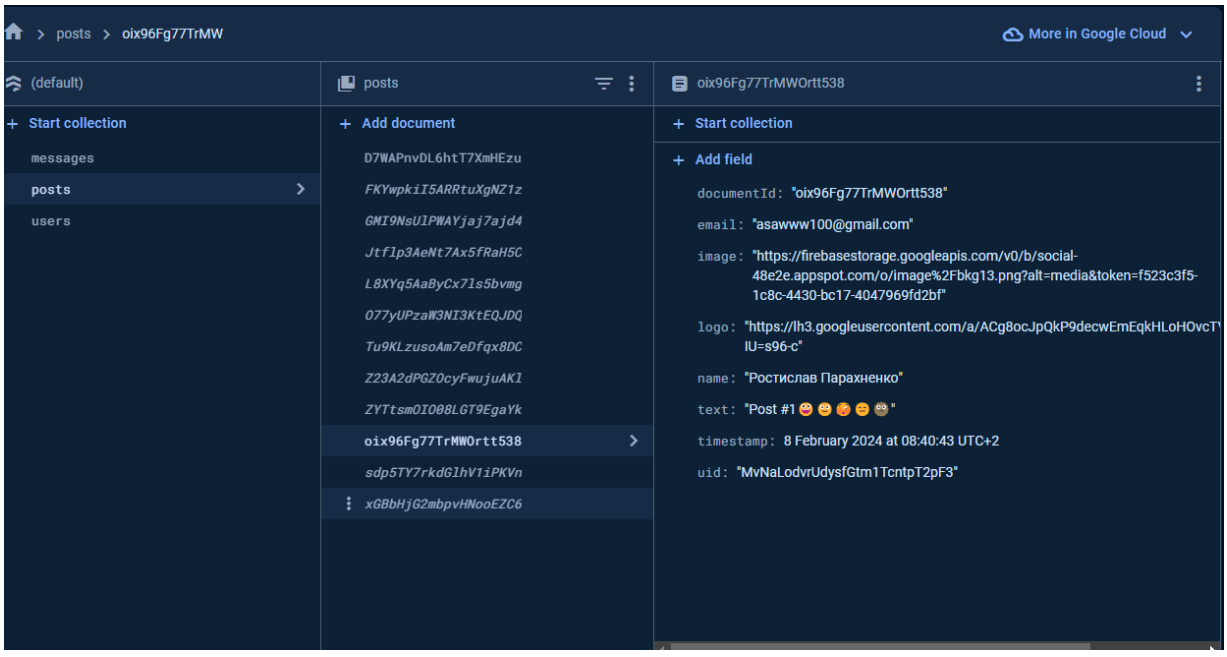

Рисунок 4.11 – Збережені дані відправленого поста

Функція handleSubmitPost,в додатку наведено, виконується при надсиланні форми для створення нового посту. Вона перехоплює подію надсилання форми (параметр e) та перевіряє, чи введений користувачем текст не є порожнім. Якщо поле введення тексту не порожнє, то виконується спроба додати новий пост до бази даних.

У тілі try виконується метод setDoc, який додає новий документ до колекції postRef в базі даних. Об'єкт, який передається у функцію setDoc, містить різноманітну інформацію про створений пост, таку як ідентифікатор користувача (UID), зображення, текст, дату та час створення посту тощо. Після успішного створення посту поле введення тексту очищається, готуючись до надходження нового вмісту.

У разі помилки під час додавання посту до бази даних, функція спрацьовує блок catch, де обробляється виняток. В цьому випадку встановлюється тип помилки HANDLE_ERROR, який може використовуватися для відображення повідомлення про помилку або іншої відповідної обробки помилки. Інформація про помилку також виводиться до консолі для подальшого аналізу та відлагодження програми.

Якщо користувач залишив поле введення тексту порожнім, то також спрацьовує блок else, де також встановлюється тип помилки HANDLE_ERROR.

Функція handleLike, наведено в додатку в, відповідає за обробку дії натискання на кнопку "Лайк" на пості. При натисканні на цю кнопку вона спершу перевіряє, чи вже є лайк від поточного користувача на даному пості.

Для цього вона створює запит до колекції likesCollection, в якому шукає документи, де поле id співпадає з UID поточного користувача. Після отримання результатів запиту перевіряється наявність документу в результаті.

Якщо документ існує (тобто користувач вже поставив лайк), то він видаляється з колекції likesCollection за допомогою функції deleteDoc.

Якщо ж документ не знайдено (тобто користувач ще не ставив лайк на цей пост), то створюється новий документ в колекції likesRef з полем id, яке відповідає UID поточного користувача.

У випадку виникнення помилки під час виконання операцій з базою даних, виводиться повідомлення про помилку та додаткова інформація про неї виводиться до консолі для подальшого аналізу.

Функція deletePost, наведено в додатку в, відповідає за обробку дії видалення поста. При натисканні на відповідну кнопку вона спочатку перевіряє, чи поточний користувач є автором поста, який він намагається видалити.

Якщо умова виконується (тобто користувач є автором поста), то виконується видалення документу за допомогою функції deleteDoc, яка приймає посилання на документ, який потрібно видалити.

Якщо ж умова не виконується (тобто користувач намагається видалити пост, який не належить йому), то виводиться повідомлення "You can't delete other users' posts".

У випадку виникнення помилки під час виконання операцій з базою даних, виводиться повідомлення про помилку в консоль для подальшого аналізу.

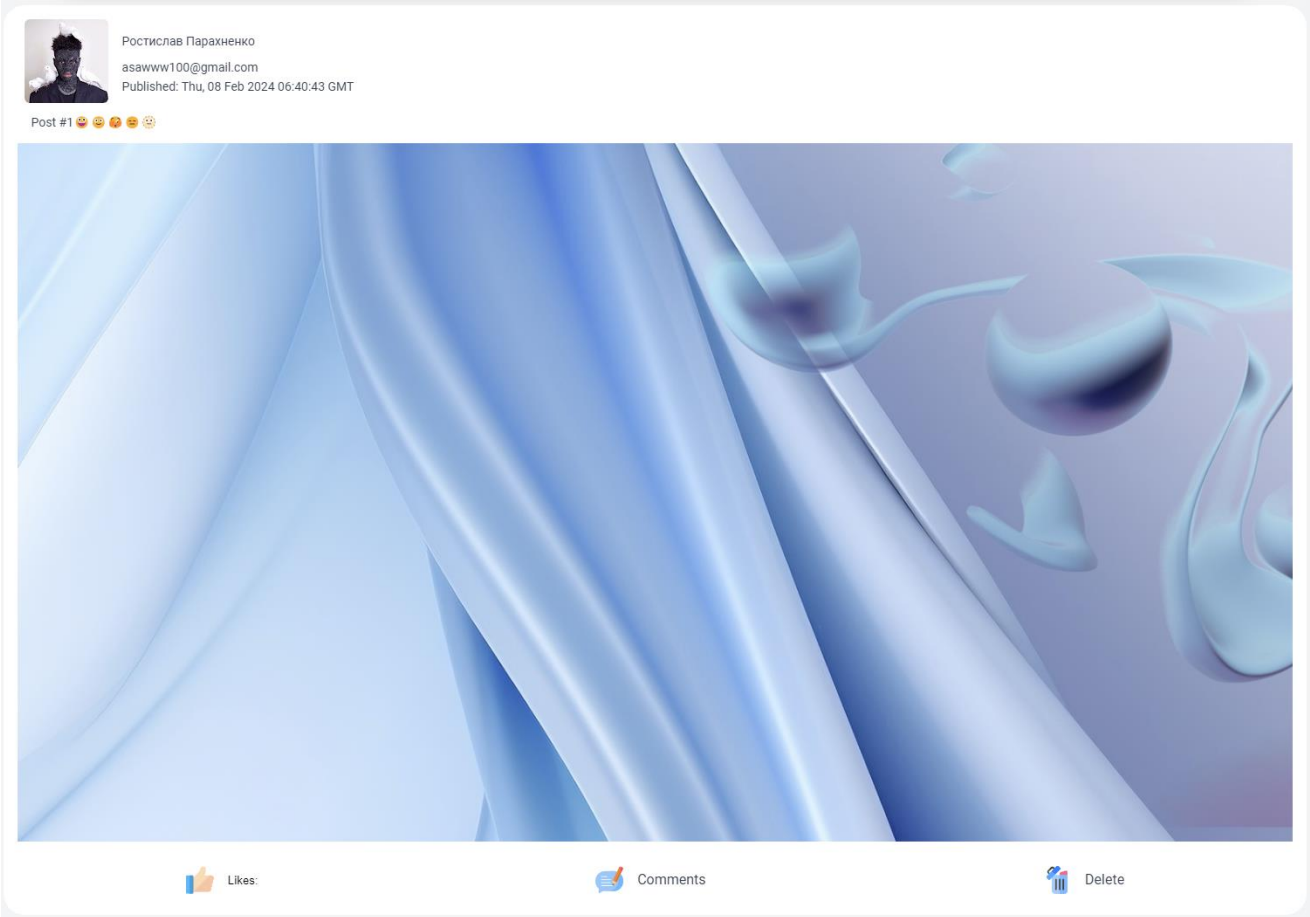

Рисунок 4.12 – Інтерфейс сформованого посту

Так, після додавання коментарію у базі даних формується новий документ, що представляє сам коментар. Цей документ містить різноманітну інформацію про коментар, таку як ідентифікатор користувача, ім'я, текст коментаря, час створення тощо. Кожен коментар пов'язаний з конкретним постом за допомогою унікального ідентифікатора поста.

Додавання коментарію зазвичай виконується шляхом створення нового документу у відповідній колекції коментарів для певного поста у базі даних. Після цього дані про коментарі оновлюються у відповідному пості, щоб мати можливість зручно відображати їх користувачам у відповідному контексті.

Така організація бази даних дозволяє зберігати коментарі разом з відповідними постами та ефективно керувати ними, що робить процес спілкування користувачів у мережі більш зручним та організованим.

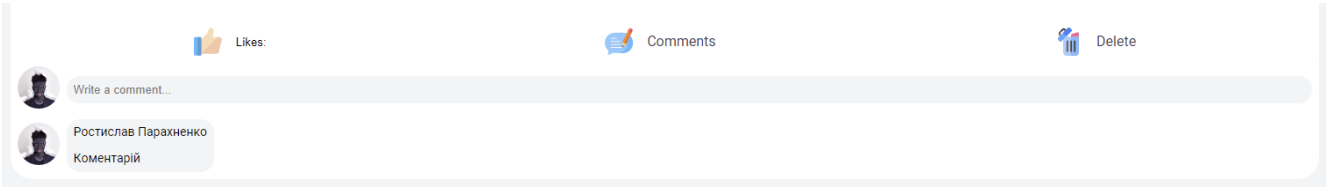

Рисунок 4.13 – додавання коментарію до посту

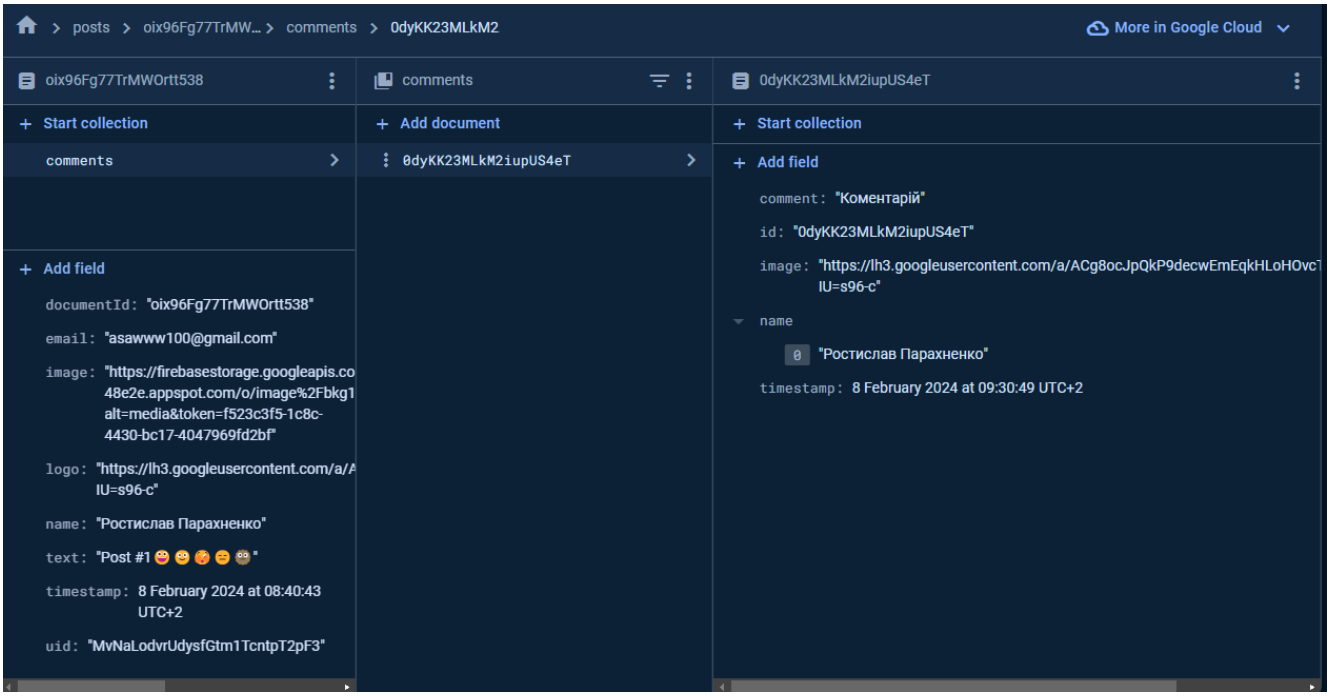

Рисунок 4.14 – Створений запис із коментарем до відповідного посту

Так, при постановці вподобання під постом, ідентифікатор користувача, який поставив вподобання, також зберігається в базі даних разом з відповідним постом. Це дозволяє системі точно відслідковувати, які користувачі вподобали певний пост.

Коли користувач натискає на кнопку вподобання, відповідний ідентифікатор користувача додається до списку вподобань для відповідного поста у базі даних. Після цього інформація про вподобання оновлюється, і відповідне відображення на сторінці оновлюється, щоб відобразити, хто із користувачів поставив вподобання.

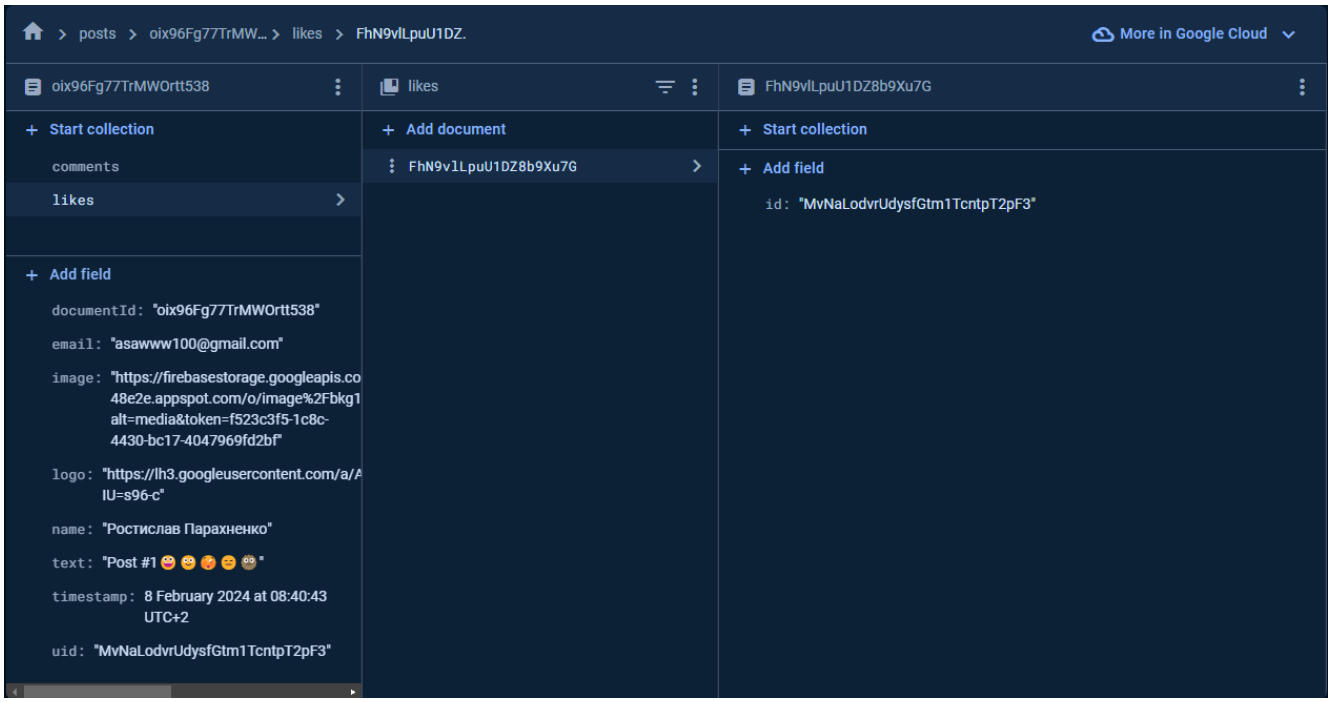

Рисунок 4.15 – Запис вподобання до бази даних

Можливість додавання користувачів у список друзів є важливою функціональністю для соціальної мережі. Це відкриває можливість користувачам спілкуватися та обмінюватися повідомленнями з іншими користувачами, які вони обирають як друзів

Основний алгоритм функції, наведено в додатку в, такий:

Спочатку виконується запит до бази даних за документом користувача, який хоче додати іншого користувача у список друзів. Запит виконується за допомогою функції query, де шукається документ з колекції "users" за умовою, що поле "uid" дорівнює ідентифікатору поточного користувача.

Після отримання даних документу, виконується оновлення цього документу, додаючи до поля "friends" нового друга. Оновлення документу відбувається за допомогою функції updateDoc, де до поля "friends" додається новий об'єкт з інформацією про користувача, якого додають у друзі.

Якщо операція успішно виконується, користувач успішно додається до списку друзів поточного користувача. У випадку помилки, відображається повідомлення про помилку в консолі.

Цей фрагмент коду відповідає за функціонал додавання друзів і відображення їх списку у правій частині інтерфейсу.

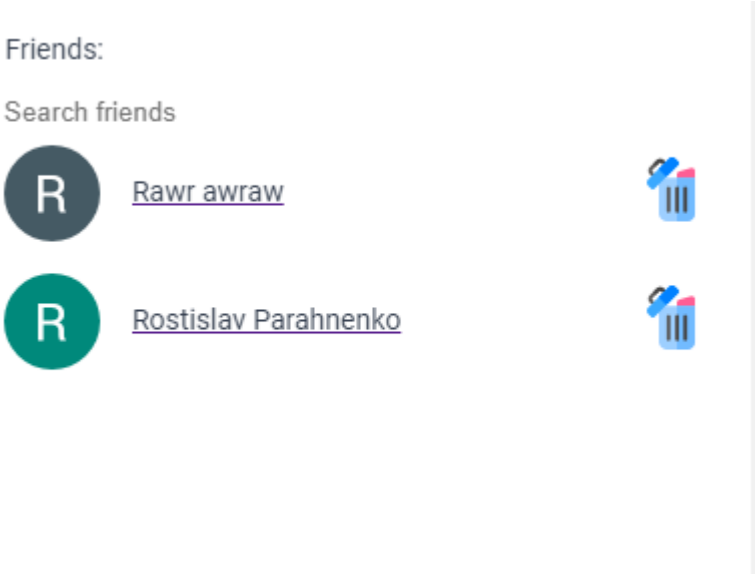

Рисунок 4.16 – Відображення друзів у правій частині застосунку

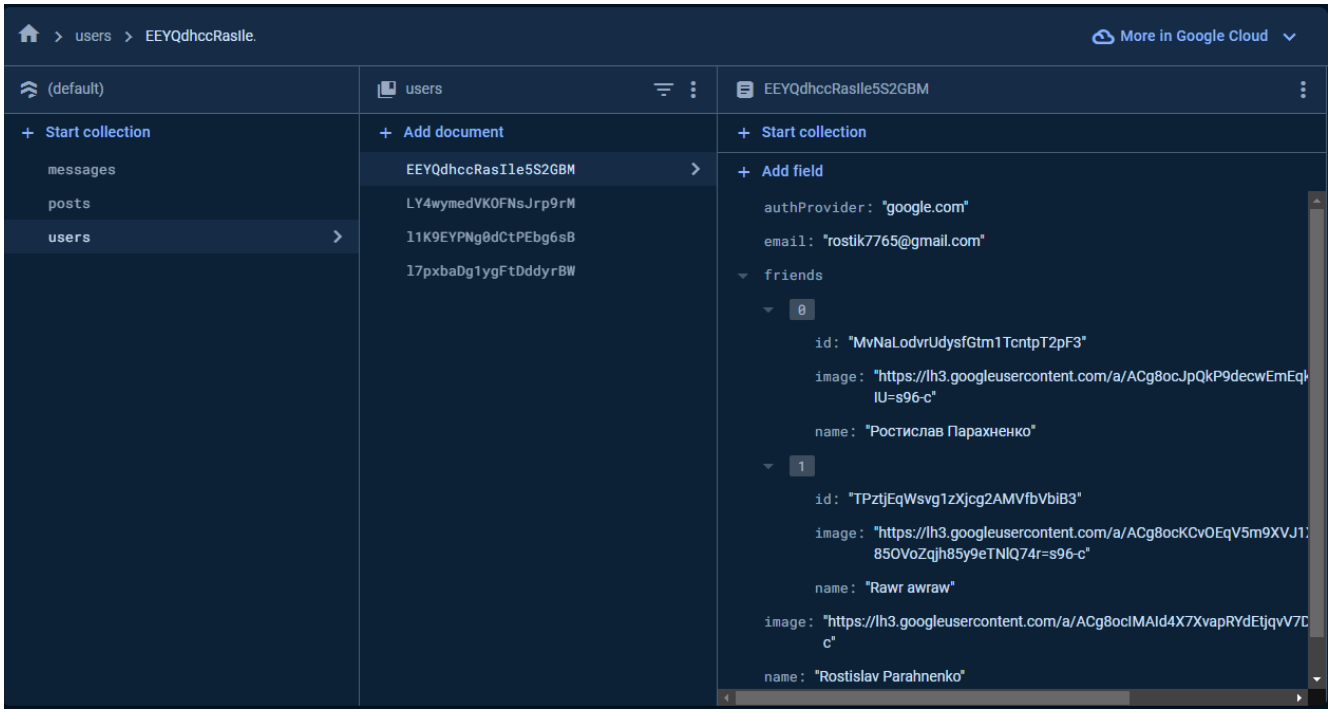

Рисунок 4.17 – Запис доданих друзів до бази даних

При натисканні на доданого друга відбувається перехід на його профіль, де можна переглянути додаткову інформацію про нього.

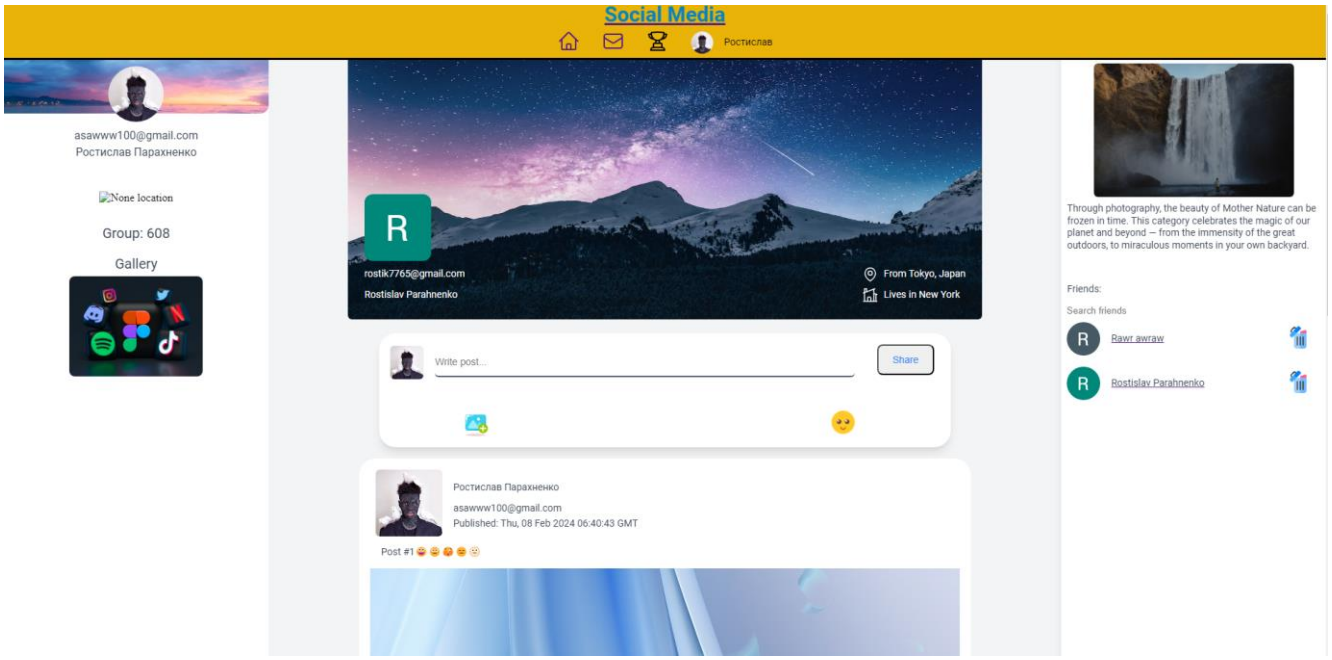

Рисунок 4.18 – Профіль друга із інформацією про нього

При натисканні на іконку конверта в навігаційному меню відбувається перехід у чат, де користувач може спілкуватися із своїми друзями.

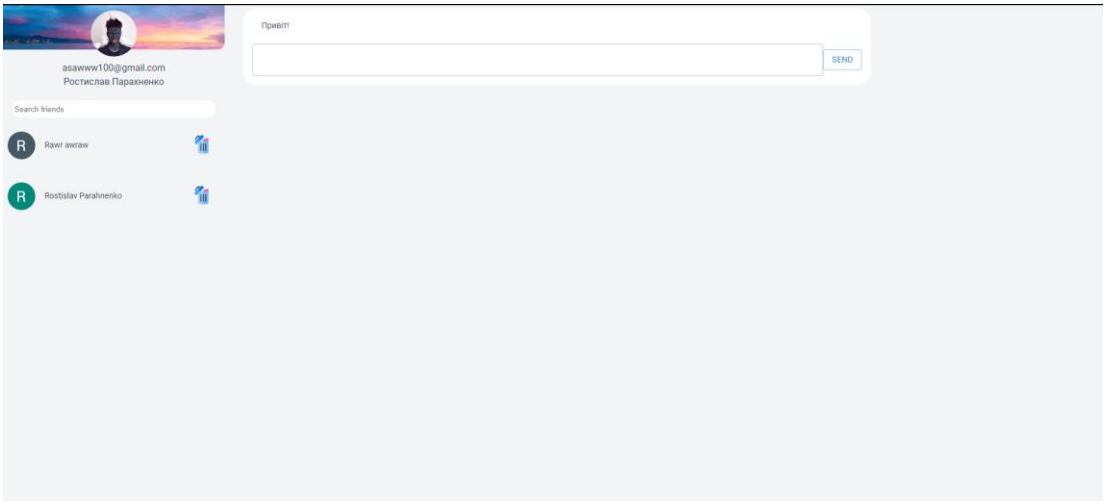

Рисунок 4.19 – Інтерфейс чату із друзями

Після натискання на обраного друга і початку чату інтерфейс переходить у режим чату з цим користувачем. Це дає можливість обмінюватися повідомленнями, файлами та іншими даними з цим конкретним користувачем. Ваш застосунок дозволяє користувачам легко взаємодіяти між собою, роблячи спілкування ефективним та зручним.

Функція startChatWithUser,в додатку в наведено, призначена для початку чату з обраним користувачем. Ось як вона працює:

Передаються параметри selectedUserId (ідентифікатор обраного користувача) і currentUserUid (ідентифікатор поточного користувача). Функція встановлює обраного користувача, що потрібно почати чат. Здійснюється запит до колекції messages у базі даних Firestore для отримання всіх повідомлень між поточним користувачем і обраним користувачем. Результати запиту у вигляді об'єктів документів обробляються, і дані кожного повідомлення додаються до масиву messages.

Після завершення обробки даних масив messages передається в стан компоненту для відображення у чаті.

Ця функція дозволяє завантажувати історію чату з обраним користувачем та відображати її у вікні чату.

Ця функція, наведено в додатку в, відправки повідомлень гарантує, що нове повідомлення буде надіслане до відповідного чату між вами і обраним користувачем. Коли користувач натискає кнопку або надсилає повідомлення у текстове поле, дані повідомлення (такі як ID відправника і отримувача, вміст повідомлення, час відправки тощо) додаються до бази даних за допомогою функції addDoc. Після успішного надсилання текстове поле очищається, готуючи інтерфейс для нових повідомлень. Також після надсилання повідомлення викликається функція fetchMessages, яка оновлює список повідомлень, щоб відобразити нове повідомлення.

Ця функція допомагає забезпечити зручну та безперервну комунікацію між користувачами вашого застосунку.

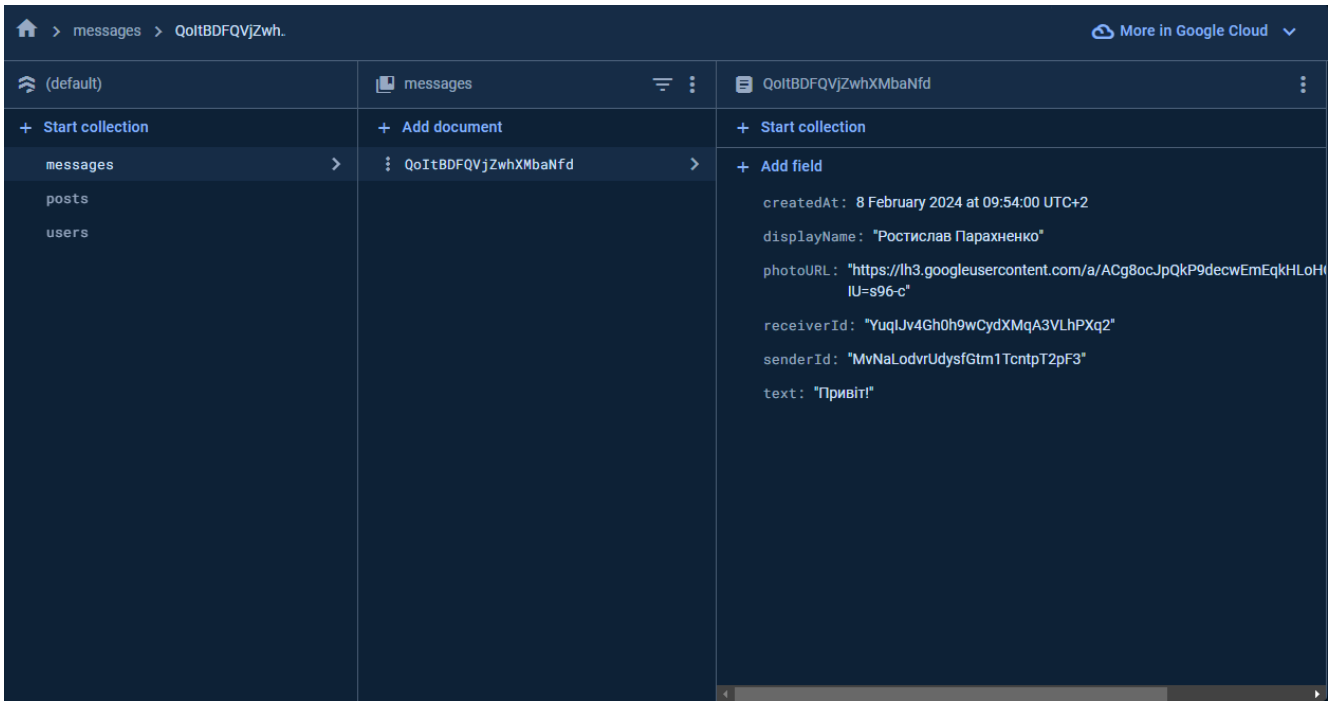

Рисунок 4.20 – Запис кожного повідомлення до бази даних

Для оптимізації чату, базу даних було проіндексовано. Індексація дозволяє швидко знаходити та отримувати доступ до даних у базі даних за допомогою запитів. Вона покращує швидкість виконання запитів, особливо тих, що використовують фільтрацію, сортування або об'єднання даних.

Під час індексації визначені певні поля, за якими часто відбуваються запити, та створюються спеціальні структури даних, що допомагають зберігати та швидко

знаходити ці дані. Це дозволяє зменшити час відповіді на запити та поліпшити загальну продуктивність застосунку.

Рисунок 4.21 – Індексація полів

4.4 Аналіз та тестування застосунку

Була проведена оцінка та аналіз ефективності та результативності розробленого застосунку на основі даних, отриманих під час його використання. Цей аналіз має на меті виявлення сильних та слабких сторін програми, виявлення можливих проблем або недоліків у функціонуванні та визначення можливостей для подальшого вдосконалення.

Досягнення успішного впровадження застосунку було результатом докладних планувань, ретельної розробки та ефективного тестування. Використання сучасних технологій та найкращих практик у розробці дозволило створити стабільний продукт.

Під час тестування було охоплено широкий спектр функціональності нашого застосунку. Зокрема, було протестовано такі ключові можливості:

1. Створення постів: Впевнилися, що користувачі можуть легко створювати нові пости з використанням зручного інтерфейсу.
2. Відправка повідомлень: Перевірили, що користувачі можуть ефективно взаємодіяти між собою за допомогою системи обміну повідомленнями.
3. Аутентифікація: Переконалися, що процес аутентифікації працює надійно і безпечно, забезпечуючи доступ користувачів до їх облікових записів.
4. Додавання до друзів: Перевірили функціонал додавання користувачів у список друзів та взаємодії з цим списком.

5. Додавання коментарів: Впевнилися, що користувачі можуть легко додавати коментарі до постів та взаємодіяти з іншими користувачами.
6. Видалення постів: Перевірили, що користувачі можуть безпечно видаляти свої пости, забезпечуючи правильну роботу цієї опції.

Під час тестування також була перевірена можливість користувачів завантажувати зображення до своїх постів та профілю. Впевнилися, що ця функція працює ефективно і безперебійно, дозволяючи користувачам легко додавати свої зображення та забезпечуючи їх відображення вірно та швидко.

Також було перевірено, що інформація про користувача, така як ім'я, електронна пошта та інші дані, відображається коректно на їх профілі. Впевнилися, що ці дані відображаються точно та оновлюються відповідно до змін, які користувач може внести у свій профіль. Це забезпечує коректну інформацію про користувачів у застосунку та сприяє покращенню їх взаємодії з іншими користувачами.

Після успішного завершення тестування та впровадження програмного забезпечення ми отримали позитивний зворотний зв'язок від користувачів, які відзначили зручний та інтуїтивно зрозумілий інтерфейс застосунку. Вони висловили задоволення від роботи з програмою та відзначили її корисність у спілкуванні та обміні інформацією.

Загалом, результати використання програмного забезпечення є дуже задовільними, а користувачі висловлюють позитивні враження від його функціональності та можливостей. Важливою частиною нашої стратегії є постійне вдосконалення та реагування на потреби користувачів для забезпечення їхнього задоволення та задоволення їхніх потреб.

Висновки до розділу 4

У цьому розділі ми детально розглянули процес створення, розробки та впровадження корпоративної соціальної мережі для навчальних закладів. Починаючи з планування розробки програмного забезпечення і вибору необхідних інструментів, ми перейшли до реалізації різноманітних функціональностей та інтерфейсів, включаючи аутентифікацію, створення постів, спілкування з друзями та інше.

Під час розробки було використано різні інструменти та технології, такі як React, Firebase, Material-UI, які допомогли нам створити потужний та ефективний застосунок. Ми успішно реалізували функціональності, такі як авторизація, створення та відправка постів, можливість додавання друзів, обмін повідомленнями та багато іншого.

Під час тестування було виявлено та виправлено різні помилки, а також проведено тестування на різних пристроях для забезпечення коректної роботи застосунку на різних платформах та в різних умовах використання. В результаті розроблений застосунок виявився функціональним, стабільним та ефективним і готовим до використання в навчальних закладах.

У цілому, реалізація цієї корпоративної соціальної мережі була успішною, і вона може стати цінним інструментом для спілкування та співпраці в навчальному середовищі.

ВИСНОВКИ

У ході аналізу і вивчення різних аспектів корпоративної соціальної мережі для навчально-виховних закладів було виявлено, що цей інноваційний інструмент може ефективно підвищити рівень комунікації та співпраці всередині освітньої громади. На підставі аналізу існуючих програмних засобів, цілей та завдань корпоративної соціальної мережі, а також технологій для створення клієнт-серверних застосунків, були визначені оптимальні стратегії реалізації та впровадження такого інструменту у сучасних навчальних установах.

Для досягнення поставленох мети виконано наступні завдання:

1) Проаналізовано предметну сферу розробки корпоративної соціальної мережі навчально-виховного закладу.

2) Проаналізовано існуючі аналоги, визначено їх переваги та недоліки.

3) Проведено аналіз вимог та специфікацію вимог до програмного забезпечення застосунку.

4) Спроєктовано та розроблено базу даних застосунку.

5) Спроєктовано, програмно реалізовано та протестовано застосунок корпоративної соціальної мережі навчально-виховного закладу.Результати дослідження підтвердили, що розроблена корпоративна соціальна мережа для навчально-виховних закладів є ефективним інструментом для покращення комунікації та співпраці між учасниками навчального процесу. Мета кваліфікаційної роботи була досягнута, оскільки було розроблено та успішно впроваджено функціональну та безпечну платформу, що сприяє побудові спільноти в освітньому середовищі та поліпшенню загальної ефективності навчального процесу.

Загалом, отримані результати надають чітке уявлення про можливості та перспективи розробки корпоративної соціальної мережі для навчально-виховних закладів. Розгорнута функціональність та сучасні технології дозволять створити інноваційний інструмент, спрямований на підвищення ефективності освітнього процесу та покращення взаємодії всіх учасників освітньої громади.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Advanced Configuration. URL: https://facebook.github.io/create-react-app/docs/advanced-configuration (дата звернення 30.12.2023)
2. Analyzing the Bundle Size. URL: https://facebook.github.io/create-react-app/docs/analyzing-the-bundle-size (дата звернення 30.12.2023)
3. Code Splitting. URL: https://create-react-app.dev/docs/code-splitting/ (дата звернення 30.12.2023)
4. CSS-Tricks. URL: https://css-tricks.com (дата звернення 10.02.2024)
5. Deployment. URL: https://facebook.github.io/create-react-app/docs/deployment (дата звернення 30.12.2023)
6. Dev.to. URL: https://dev.to (дата звернення 30.12.2023)
7. Firebase. URL: https://firebase.google.com/docs (дата звернення 30.12.2023)
8. FreeCodeCamp. URL: https://www.freecodecamp.org (дата звернення 10.02.2024)
9. GitHub. URL: https://github.com (дата звернення 30.12.2023)
10. JavaScript MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення 30.12.2023)
11. Medium. URL: https://medium.com (дата звернення 30.12.2023)
12. Minify. URL: https://facebook.github.io/create-react-app/docs/troubleshooting#npm-run-build-fails-to-minify (дата звернення 30.12.2023)
13. Mozilla Developer Network (MDN). URL: https://developer.mozilla.org (дата звернення 10.02.2024)
14. npm Documentation. URL: https://docs.npmjs.com (дата звернення 10.02.2024)
15. React. URL: https://react.dev/ (дата звернення 30.12.2023)
16. Redux. URL: https://redux.js.org/usage/ (дата звернення 30.12.2023)
17. Redux ToolKit. URL: https://redux-toolkit.js.org/usage/usage-guide (дата звернення 30.12.2023)

18. Smashing Magazine. URL: https://www.smashingmagazine.com (дата звернення 10.02.2024)
19. Stack Overflow. URL: https://stackoverflow.com (дата звернення 30.12.2023)
20. The Net Ninja. URL: https://netninja.dev (дата звернення 10.02.2024)
21. W3Schools. URL: https://www.w3schools.com (дата звернення 30.12.2023)
22. Google Developers. URL: https://developers.google.com (дата звернення 10.02.2024)
23. Codecademy. URL: https://www.codecademy.com (дата звернення 10.02.2024)
24. Mozilla Developer Network (MDN). URL: https://developer.mozilla.org (дата звернення 10.02.2024)
25. FreeCodeCamp. URL: https://www.freecodecamp.org (дата звернення 10.02.2024)
26. Codecademy. URL: https://www.codecademy.com (дата звернення 10.02.2024)

ДОДАТОК

```
return (
   <div className='w-full' style={{ margin: 0, padding: 0 }}>
    <div className='fixed top-0  w-full bg-white z-40' >
     {/* Navbar component */}
     <Navbar />
    </div>
    <div className='flex bg-gray-100 flex-col  h-screen'>
     <div className='flex  flex-col h-screen w-[20%]  mt-12'>
      <div className='flex  flex-col items-center relative'>
        <img src={nature} alt='nature' className='h-28 w-full rounded-r-xl
object-cover' />
        <div className='absolute -bottom-4'>
          <Tooltip content='Profile' placement='top'>
            <Avatar src={userData?.image ? userData?.image : avatar}
className='z-30 w-20 h-20 object-cover rounded-full' size='md' />
          </Tooltip>
        </div>
      </div>
      <div className='flex flex-col items-center pt-6'>
        <p className='font-roboto font-medium text-md mb-2 text-gray-700
no-underline tracking-normal leading-none'>
          {user?.email || userData?.email}
        </p>
        <p className='font-roboto font-medium text-md mb-2 text-gray-700
no-underline tracking-normal leading-none'>
          {user?.name || userData?.name}
        </p>
```

```
        </div>
        <input
        className="border-0 outline-none mt-4 h-[30px] mx-4 rounded-xl"
        name="input"
        value={input}
        type="text"
        placeholder=" Search friends"
        onChange={(e) => setInput(e.target.value)}
       ></input>
        {friendList?.length > 0 ? (
        searchFriends(friendList)?.map((friend) => {
         return (
          <div
          className="flex items-center justify-between hover:bg-gray-100 mt-2
border border-gray-300 rounded p-2 duration-300 ease-in-out"
          key={friend.id}
          onClick={()=> startChatWithUser(friend.id, user.uid)}
         >
          <div className="flex items-center my-2 cursor-pointer">
           <div className="flex items-center">
            <Avatar
             className='rounded-full size-12'
             src={friend?.image || avatar}
             alt="avatar"
            ></Avatar>
            <p className="ml-4 font-roboto font-medium text-sm text-gray-700
no-underline tracking-normal leading-none">
             {friend.name}
            </p>
           </div>
```

```
            </div>
              <div className="mr-4">
                <img
                  onClick={() =>
                    removeFriend(friend.id, friend.name, friend.image)
                  }
                  className="cursor-pointer"
                  src={remove}
                  alt="deleteFriend"
                ></img>
              </div>
            </div>
          );
        })
      ) : (
        <p className="mt-10 font-roboto font-medium text-sm text-gray-700 no-
underline tracking-normal leading-none">
          Add friends to check their profile
        </p>
      )}
    </div>
      {/* Main chat area in the center */}
        <div className='flex-auto  w-[60%] absolute  left-[20%] top-14 bg-gray-
100 rounded-xl p-4 mt-4 '>
          <div className='w-80% mx-auto'>
          <div className='mb-4'>
            <div className='flex flex-col py-4 mx-4 px-4 bg-white rounded-3xl'>
              {/* <div className='flex items-center pb-4 ml-2'>
                <Avatar size='sm' variant='circular' alt='avatar' src={
userData?.photoURL } />
```

```
                <div className='flex flex-col'>
                  <p className='ml-4 py-2 font-roboto font-medium text-sm text-gray-700 no-underline tracking-normal leading-none'>name</p>
                  <p className='ml-4 font-roboto font-medium text-sm text-gray-700 no-underline tracking-normal leading-none'>Published: timestamp</p>
                </div>
              </div> */}
               <div>
                {data.map((message) => (
                 <div key={message.id}>
                  <p  className='ml-4 py-2 font-roboto font-medium text-sm text-gray-700 no-underline tracking-normal leading-none'>{message.text}</p>
                 </div>
                ))}
               </div>
               <div>
               </div>
              <div className=''>
              </div>
              <div className='flex justify-around items-center pt-4'>
               <TextField
                value={value}
                fullWidth
                rowsMax={2}
                variant={'outlined'}
                onChange={e => setValue(e.target.value)}
                />
              <Button onClick={sendMessage} variant={'outlined'}>
               Send
              </Button>
```

```
                </div>
              </div>
              </div>
            </div>
        </div>
      </div>
    )
}
```

```
return (
  <div className='flex flex-col items-center'>
    <div className='flex flex-col py-4 w-[90%] bg-white rounded-3xl shadow-lg'>
      <div className='flex items-center border-b-2 border-gray-300 pb-4 pl-4 w-full'>
        <Avatar src={userData?.image || avatar} size='sm' className='w-12 h-12' variant='circular' alt='avatar' ></Avatar>
        <form className='w-full' onSubmit={handleSubmitPost}>
          <div className='flex justify-between items-center'>
            <div className='w-full ml-4'>
              <input
              onChange={(e) => setPostText(e.target.value)}
              value={postText}
              ref={text}
              placeholder='Write post...'
              className=' block py-2.5 px-0  text-sm text-gray-900 bg-transparent border-0 border-b-2 border-gray-300 appearance-none  dark:border-gray-600 dark:focus:border-blue-500 focus:outline-none focus:ring-0 focus:border-blue-600 peer w-full bg-white rounded-md' type='text' name='text'  />
            </div>
            <div className='mx-4'>
              {image && <img alt='previewImage' src={image} className='h-24 rounded-xl'></img>}
            </div>
            <div className='mr-4'>
              <Button onClick={handleSubmitPost} className=' bg-gradient-to-r from-blue-500 via-blue-600 to-blue-700 hover:bg-gradient-to-br focus:ring-4 focus:outline-none focus:ring-blue-500 text-blue-500 font-medium rounded-lg text-sm px-5 py-2.5 text-center me-2 mb-2' variant='text' type='submit' >
```

```
                Share
               </Button>
              </div>
             </div>
            </form>
           </div>
           <span className='bg-blue-700 py-1 rounded-md'
style={{width:`${progressBar}%`}}>
            {}
           </span>
           <div className='flex justify-around items-center pt-4'>
            <div className='flex items-center'>
              <label htmlFor='addImage' className='cursor-pointer flex items-
center'>
               <img src={addImage} alt='addimg' className='h-10 mr-4' />
               <input onChange={handleUpload} id='addImage' type='file'
style={{display:'none'}} />
              </label>
              {file &&
              <Button onClick={submitImage} variant='text' className='bg-
gradient-to-r from-blue-500 via-blue-600 to-blue-700 hover:bg-gradient-to-br
focus:ring-4 focus:outline-none focus:ring-blue-500 text-blue-500 font-medium
rounded-lg text-sm px-2 py-1 text-center me-2 mb-2'  type='submit'>
               Upload
              </Button> }
            </div>
            <div className='flex items-center'>
             {/* <img src={live} alt='live' className='h-10 mr-4' /> */}
             <p className='font-roboto font-medium text-md text-gray-700 no-
underline tracking-normal leading-none'></p>
```

```
            </div>
            <div className='flex items-center' >
              <img src={smile}  alt='smile' className='h-10 mr-4 cursor-pointer z-
20'onClick={()=> setEmojiPicker(!emojiPicker)}   />
                {emojiPicker && (
                 <div  className='absolute z-10 pl-10' >
                    <EmojiPicker onEmojiClick={handleEmojiSelect}
lazyLoadEmojis={true} />
                 </div>
                 )}
              </div>
             </div>
            </div>
            <div className='flex flex-col py-4 w-full'>
             {state.error
             ? <div className='flex justify-center items-center'><Alert
color='red'>Something went wrong, try again...</Alert></div>
             : <div>{state?.posts?.length > 0 && state?.posts?.map((post, index) => {
              return <PostCard
              logo={post?.logo}
              id={post?.documentId}
              key={index}
              uid={post?.uid}
              name={post?.name}
              image={post?.image}
              email={post?.email}
              text={post?.text}
              timestamp={new Date(
               post?.timestamp?.toDate()
              )?.toUTCString()}
```

```
        />
      })}</div>}
    </div>
    <div ref={scrollRef} >
      {/* ref for later */}
    </div>
  </div>
 )
}
```

Налаштування: firebase

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import {getAuth, onAuthStateChanged} from 'firebase/auth'
import {getFirestore} from 'firebase/firestore'
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyDI8az_GvO7-R4ECVo-k0kcJB-pmpyAx6M",
  authDomain: "social-48e2e.firebaseapp.com",
  projectId: "social-48e2e",
  storageBucket: "social-48e2e.appspot.com",
  messagingSenderId: "522921723708",
  appId: "1:522921723708:web:b4f515bc3d9e38d8630039",
  measurementId: "G-YP5GKGBQ2G"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);

const auth = getAuth(app)
const db = getFirestore(app)

export {auth, db, onAuthStateChanged}
```

Авторизації користувача:

```
import { useFormik } from "formik";
import { useContext, useEffect, useState } from "react";
import * as Yup from 'yup'
import { AuthContext } from "../AppContext/AppContext";
import {auth, onAuthStateChanged} from '../firebase/firebase'
import {useNavigate} from 'react-router-dom'
const {signInWithGoogle, loginWithEmailAndPassword} =
useContext(AuthContext)
const navigate = useNavigate()

const [loading, setLoading] = useState(false)

useEffect(() => {
  setLoading(true);
  onAuthStateChanged(auth,(user)=> {
    if(user) {
      navigate('/')
      setLoading(false);
    }else{
      setLoading(false);
    }
  })
},[navigate])

let initialValues = {
    email: "",
    password: ""
}
```

```
const validationSchema = Yup.object({
    email: Yup.string().email("Invalid email adress").required("Required"),
    password: Yup.string().required("Required").min('6', "Must be at least 6 charachters long").matches(/^[a-zA-Z]+$/, 'Password can only contain letters'),
})

const handleSubmit = (e) => {
    e.preventDefault()
    const {email, password} = formik.values
    if(formik.isValid === true){
        loginWithEmailAndPassword({email, password})
        setLoading(true)


    }else{
        setLoading(false)
        alert('check your input fields')
    }
    console.log('formik', formik)
  }


const formik = useFormik({initialValues, validationSchema, handleSubmit})
```

Головна сторінка застосунку:

```
let [data, setData] = useState([])

const count = useRef(0)
const {user, userData} = useContext(AuthContext)

const handleRandom = (arr) => {
```

```
    setData(arr[Math.floor(Math.random() * arr?.length)])
}

useEffect(() => {

    const imagelist = [
        {
            id: '1',
            image: laptop
        },
        {
            id: '2',
            image: media
        },
        {
            id: '3',
            image: apps
        },
        {
            id: '4',
            image: tik
        },
    ]
    handleRandom(imagelist)
    let imgAdd = 0
    let startImg = setInterval(() =>{
        imgAdd++
        handleRandom(imagelist)
        count.current = imgAdd
        if(imgAdd === 5){
```

```
            clearInterval(startImg)
        }
    }, 2000)
    return ()=> {
        clearInterval(startImg)
    }
}
    , [])
```

Ліва частина застосунку:

```
export const postActions = {
    SUBMIT_POST: 'SUBMIT_POST',
    HANDLE_ERROR: 'HANDLE_ERROR',
    ADD_LIKE: 'ADD_LIKE',
    ADD_COMMENT: 'ADD_COMMENT'
}
export const postsState = {
    error: false,
    posts: [],
    likes: [],
    comments: []
}
export const PostsReducer = (state, action) => {
    switch(action.type){
        case postActions.SUBMIT_POST:
            return {
                ...state,
                error:false,
                posts: action.posts
            }
         case postActions.ADD_LIKE:
```

```
            return {
                ...state,
                error: false,
                 likes: action.likes
            }
          case postActions.ADD_COMMENT:
            return {
                ...state,
                error: false,
                 comments: action.comments
            }
          case postActions.HANDLE_ERROR:
            return {
                ...state,
                error: true,
                posts: []
            }
          default:
             return state
      }
    }
```

Збереження картинок у базі даних:

```
  const storage = getStorage()
  const metadata = {
    contentType: ['image/jpeg', 'image/jpg', 'image/png', 'image/gif',
'image/svg+xml']
  }
```

Зберігання файлу:

```
const [file, setFile] = useState(null)
```

```
  const handleUpload = (e) => {
    setFile(e.target.files[0])
  }

  const submitImage = async () => {
    const fileType = metadata.contentType.includes(file['type']);
    console.log('file', file);
    if (!file) return;

    if (fileType) {
      try {
        const storageRef = ref(storage, `image/${file.name}`);
        const uploadTask = uploadBytesResumable(storageRef, file,
metadata.contentType);
        await uploadTask.on(
          'state_changed',
          (snapshot) => {
            const progress = Math.round((snapshot.bytesTransferred /
snapshot.totalBytes) * 100);
            setProgressBar(progress);
          },
          (error) => {
            alert(error);
          },
          async () => {
            await getDownloadURL(uploadTask.snapshot.ref).then((downloadURL)
=> {
              setImage(downloadURL);
            });
          }
```

```
      );
    } catch (err) {
      dispatch({
        type: HANDLE_ERROR,
      });
      console.log(err.message);
    }
  }
};
```

Прогрес завантаженого файлу:

```
const [postText, setPostText] = useState('');
 const [emojiPicker, setEmojiPicker] = useState('');
 const handleEmojiSelect = (event) => {
   setPostText((prevText) => prevText + event.emoji);
   setSelectedEmoji(event.native);


 };
```

Створення поста:

```
const handleSubmitPost = async(e) => {
   e.preventDefault()
   if(text.current.value !== ''){
   try{
      await setDoc(postRef, {
        documentId: document,
        uid: user?.uid || userData?.uid,
        logo: user?.photoURL,
        name: user?.displayName || userData?.name,
        email: user?.email || userData?.email,
        text: text.current.value,
        image: image,
```

```
        timestamp: serverTimestamp()
      })
      text.current.value = ''

    }catch(err){
     dispatch({
      type: HANDLE_ERROR,
     })
     console.log(err.message)
    }
  }
   else{
    dispatch({
     type: HANDLE_ERROR
    })
   }
  }
```

Функція handleLike:

```
const handleLike = async(e) => {
      e.preventDefault();
      const q = query(likesCollection, where("id", "==", user?.uid));
      const querySnapshot = await getDocs(q);

      const likesDocId = await querySnapshot?.docs[0]?.id;
      try {
       if (likesDocId !== undefined) {
         const deleteId = doc(db, "posts", id, "likes", likesDocId);
         await deleteDoc(deleteId);
       } else {
         await setDoc(likesRef, {
```

```
            id: user?.uid,
          });
        }
      } catch (err) {
        alert(err.message);
        console.log(err.message);
      }
    };
```

Функція deletePost:

```
  const deletePost = async(e) => {
      e.preventDefault()
      try{
        if(user?.uid === uid){
          await deleteDoc(singlePostDocument)
        }else{
          alert('You cant delete other users posts')
        }
      }catch(err){
        console.log(err.message)
      }
    }
```

Функція addUser:

```
const addUser = async() => {
      try{
          const q = query(collection(db, 'users'), where('uid', '==', user?.uid))
          const doc = await getDocs(q)
          const data = doc.docs[0].ref
          await updateDoc(data, {
              friends: arrayUnion({
                  id: uid,
```

```
                image: logo,
                name: name,
            })
        })
    }catch(err){
        console.log(err.message)
    }
}
```

Реалізація чату:

```
const startChatWithUser = async (selectedUserId, currentUserUid) => {
    try {
        setSelectedUser(selectedUserId);

        const q = query(
            collection(db, 'messages'),
            where('senderId', '==',  selectedUserId), // Все сообщения, отправленные вами или выбранным пользователем
            where('receiverId', '==', currentUserUid), // Все сообщения, отправленные вам или выбранному пользователю
            orderBy('createdAt')
        );

        const querySnapshot = await getDocs(q);
        const messages = [];
        querySnapshot.forEach((doc) => {
            messages.push({
                id: doc.id,
                ...doc.data()
            });
        });
```

```
      setData(messages);
    } catch (error) {
      console.error('Error loading messages:', error);
    }
};
```

Функція відправки повідомлень:

```
const sendMessage = async (e) => {
    e.preventDefault();
    try {
      await addDoc(messagesRef, {
        senderId: user.uid,
        receiverId: selectedUser,
        displayName: user.displayName,
        photoURL: user.photoURL,
        text: value,
        createdAt: serverTimestamp(),
      });
      setValue('');
      // Обновляем список сообщений после успешной отправки
      fetchMessages();
    } catch (err) {
      console.log(err.message);
    }
  };
```