МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Кафедра електроніки і комп'ютерної техніки

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи бакалавра на тему: «*Створення програмного додатку зв'язку кінцевих пристроїв доступу*.
»

Завідуючий кафедрою Опанасюк А.С.
Керівник дипломного проекту Доброжан О.А
Виконав студент групи ТК-71 Беркаш С.І.

Суми 2021

ЗМІСТ

ВСТУП

Мобільні пристрої та програмні продукти стали частиною нашого буденнішнього життя, вони – повноцінне продовження нас. У наш час все більше стає актуальним спілкування у соціальних чатах.

Тому створення власного чату, закритого типу, через який можуть спілкуватися тільки ті, хто його завантажив, є доречним для спрощення своєї роботи, бо переваги видно неозброєним оком: це і моментальна обробка інформації, і великий спектр функціоналу, і гнучкість персоналізованих налаштувань. Для всього цього використовується програмне забезпечення, яке і виконує частину функцій людини, замінює паперові сховища даних і прискорює роботу.

Однією з найбільш важливих проблем є обмін інформацією між людьми.

Об'єкт - процес інформаційної підтримки користувачів чату.

Предмет - інформаційна підтримка користувачів чату.

Мета - було вирішено створити програмний додаток, що дозволяє надсилати інформацію навчального процесу в чаті його користувачам.

Завдання дипломної роботи полягають:

1. Збір та обробка інформації про свій проект;
2. Вибір середовища розробки проекту;
3. Реалізація проекту в вибраному середовищі розробки.

Вимоги до проекту з точки зору користувача:

1. Зручний і простий інтерфейс;
2. Прийнятні системні вимоги;
3. Ясна керівництво до даного проекту;
4. Прийнятні терміни реалізації проекту.

Даний проект буде реалізований за допомогою Android Studio

Наукова новизна. Розроблено модель сучасної комунікації, яка відрізняється від вже існуючої необхідністю впровадження месенджерів, що дозволяє підвищити ефективність навчання на кафедрі.

Проаналізовано застосування месенджерів як новітнього каналу навчальної комунікації на світових та українських підприємствах, що дозволило виділити переваги використання саме месенджерів як для студентів, так і для викладачів.

Визначено напрями використання месенджерів в університетах, що дозволяє підвищити ефективність зовнішньої комунікації у навчальному процесі (підтримка та сервіс; розсилання повідомлень; просування та маркетинг; реклама в соціальних мережах) та підвищити ефективність внутрішньої (навчальної) комунікації кафедри (компенсація недоліків електронної пошти).

Надано порівняльний аналіз можливостей навчальних месенджерів, що дозволяє зробити обґрунтований вибір потрібного варіанту месенджеру.

РОЗДІЛ 1

1.1. Огляд літератури та постановка задачі проектування

1.1.1 Сутність та особливості сучасних комунікацій за допомогою веб-додатків

Ринок технологій для мобільних пристроїв є основним з найбільш стрімко висхідних в сьогоденні. Інформаційні технології осягають в усі галузі людської діяльності, особливо важливим використанням таких технологій займає в навчальних напрямках. Для збільшення можливостей та вирощування якості освіти з’являється велика кількість програмних додатків для допомоги студентам та викладачам організувати розповсюдження актуальної інформації. Для прискорення та актуалізації роботи додатків в них використовують новітні технології підтримки учбового процесу на основі використання мобільних технологій з чітким алгоритмом послідовно виконаних дій, які дозволяють контролювати процес навчання та відслідковувати динаміку показників підготовки студентів та оптимізувати обробку результатів діагностичних досліджень якості навчання за допомогою інформаційної системи, однією із таких являється використання месенджерів.

Миттєві інформаційні повідомлення або обмін миттєвими повідомленнями - це телекомунікаційна послуга, яка використовується для обміну текстовими повідомленнями між комп’ютерами або пристроями інших користувачів через комп’ютерну мережу (як правило, Інтернет). Зазвичай з самого початку це невеликі текстові повідомлення. Але з розвитком система також додала інші функції, такі як передача файлів, зображень, звукових сигналів та повідомлень, відео та виконання спільних дій, таких як малювання або гра в ігри. Для використання цього типу

зв'язку потрібна клієнтська програма. Клієнтів миттєвих повідомлень часто називають Інтернет-пейджерами або месенджерами. Різниця між миттєвими повідомленнями та електронною поштою, як тут, полягає в тому, що обмін повідомленнями відбувається в режимі реального часу. Під час надсилання електронного листа воно зберігатиметься у поштовій скриньці на сервері. Щоб отримати повідомлення, одержувач повинен перевірити свою поштову скриньку та забрати його. В Інтернет-пейджері завжди підтримується зв’язок між користувачами, і надіслані повідомлення негайно доставляються користувачам. Передача повідомлень може бути між двома або кількома співрозмовниками (конференція, чат). Система обміну миттєвими повідомленнями працює за певним протоколом. Протокол може бути серверним або безсерверним. Найпоширенішим є серверний протокол, в якому месенджер не працює поодинці, а підключений до центрального комп’ютера мережі обміну повідомленнями, який називається сервером. Ось чому месенджер називається клієнтом (клієнтською програмою).

На сьогоднішній день інформація розглядається як один з важливих ресурсів розвитку суспільства, та особливо навчання в університеті. Експерти розташовують інформацію на одному рівні з матеріальними, енергетичними та людськими ресурсами. Ефективність процесу навчання залежить від вміння менеджерів працювати з людьми, а також, від того, як вони працюють з інформацією. Основною вимогою для прийняття ефективного та об’єктивного рішення є наявність точної інформації. Єдиним способом отримання такої інформації вважається комунікація. [1, с. 79]

В сучасних умовах навчання особливу важливість набуває інформаційне забезпечення студентів та викладачів як передумова його ефективного функціонування. Ефективним засобом є застосування

комунікацій, як системи підтримки інформаційної взаємодії кафедри з його внутрішнім та зовнішнім середовищем.

Комунікації як діяльність суб'єкта – це комплекс зусиль зі створення, розвитку та підтримки репутації в очах цільових груп громадськості та суспільства в цілому задля досягнення рівня довіри до суб'єкта та реалізації цілей у його основній діяльності.

Основними завданнями комунікацій у навчальному процесі є:

1. вивчення зовнішнього впливу на діяльність суб'єкта;

2. налагодження відносин між суб'єктом і цільовими аудиторіями;

3. створення сприятливого середовища для ведення навчання;

4. допомога суб'єкту економічної діяльності в отриманні точної та своєчасної інформації;

5. пошук, вивчення та виявлення нових можливостей для навчання;

6. налагодження комунікацій всередині кафедри. [2, с. 34]

Неактуальні комунікаційні способи – одна з головних сфер виникнення проблем. Для збільшення актуальності передаваної інформації у навчальних закладах необхідно створювати та користуватися актуальніми месенджерами.

Комунікація відтворюється за допомогою передавання різної інформації, емоцій, файлів та думок від одного користувача до іншого в реальному часі.

При передаванні інформації важливо врахувати декілька факторів: наскільки актуальна інформація, як швидко передається. Комунікація включає в себе і те, що передається, і те, як це «що» передається.

Майже неможливо переоцінити важливість комунікацій в управлінні підприємством. Якщо люди не можуть обмінюватися інформацією, вони не зуміють працювати разом, формулювати цілі й досягати їх. Проте комунікації – це складний процес, що складається з взаємозалежних

кроків. Кожен з цих кроків дуже потрібний для того, щоб зробити думки зрозумілими іншій особі.

1.1.2 Роль веб-додатків для комунікації та варіанти аналогів

Способом комунікації називають особливе направлення або технологія, що створено для надсилання повідомлень користувачам. У сьогоднішньому менеджменті крім звичних каналів спілкування – новини по телевізору, журнали, зустрічі віч-на-віч, смартфони, білборди, плітки, виставки і т. д. – користуються все більшим попитом порівняно нові – факсимільний зв'язок, мобільний зв'язок, телеконференції, селекторний зв'язок, конференц-зв'язок, телемости, електронна пошта, Інтернет.

Рутинні повідомлення містять в собі різні дані або просто закріплюють, наприклад, в письмовій формі те, про що у менеджерів вже були доведені до розуміння і домовленість. Повідомлення такого типу можуть передаватися і через канали з меншою комунікацією. До письмових комунікацій доводиться вдаватися і тоді, коли одержувачі повідомлень значно віддалені один від одного або коли інформація носить офіційний характер. [3] .

На сьогоднішній день існує багато аналогів месенджерів для спілкування. У кожному є свої недоліки та переваги. Для того, щоб створити власний, потрібно розглянути існуючі.

Телеграм

Телеграм - це кросплатформений додаток для миттєвої передачі повідомлень. Випущена у 2013 році, написана на на мові високого рівня C++.

В Телеграм є не тільки передача повідомлень, а й дзвінки, у тому числі й шифровані, боти та канали.

Особливості додатку у тому, что при завантаженні его на новий пристрій, файли завантажуються автоматично, які зберігаються на сторонньому сервері.

Телеграм підходить для корпоративних цілей до 50000 осіб в одному чаті. Зв’язатися з користувачем можна не тільки по номеру телефона, а й через коротке ім’я, яке вводиться в налаштуваннях.

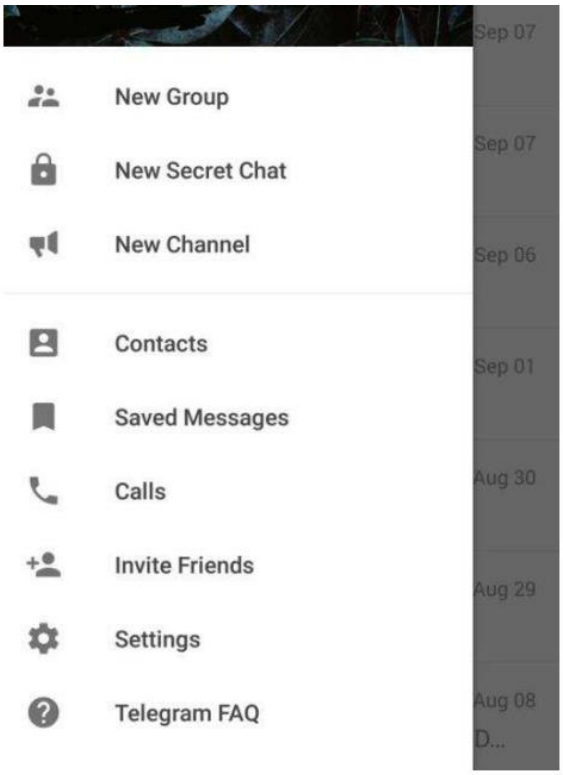

Рисунок 1.1 - Інтерфейс користувача на мобільній платформі Android

Viber

Viber також являється кроссплатформенною програмою, яка дозволяє робити дзвінки та обмін повідомленнями через Wi-Fi та мобільну

передачу даних. Реалізація була в 2010 році, написана на таких мовах як: Java, C, Python, C++, Objective-C.

Viber можно вважати корпоративним месенджером, через те, що магазини та торгові мережі можуть використовувати ваші дані в базі для надсилання рекламних пропозицій та акцій.

Також в месенджері доступні чати та боти.

Великим недоліком є те, що у додатку багато реклами, яка часто висвічується.

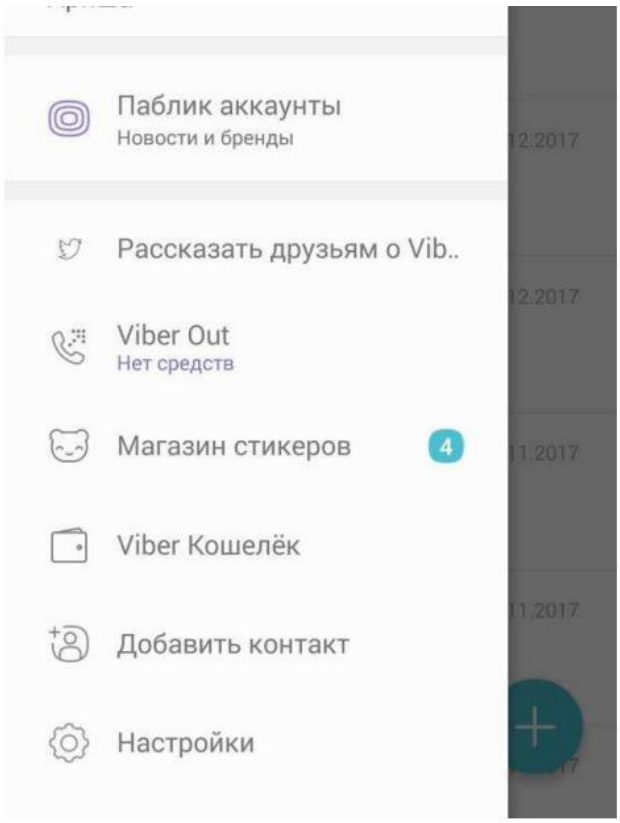

Рисунок 1.2 - Інтерфейс користувача на мобільній платформі Android

1.1.3. Постановка задачі проектування

Метою даного проекту є створення веб-додатку месенджера закритого типу для передачі повідомлень для кафедри Енергетики на

платформі Android з використанням середовища розробки Android Studio, за допомогою якого студенти та викладачі можуть обмінюватися інформацією, актуальніми новинами матеріалами для навчання.

Середовище Android Studio призначено як для невеликих команд розробників мобільних додатків (навіть в кількості однієї людини), або ж великих міжнародних організацій з GIT або іншими подібними системами управління версіями. Досвідчені розробники зможуть вибрати інструменти, які більше підходять для масштабних проектів. Рішення для Android розробляються в Android Studio з використанням Java або C ++. У використанні даного проекту було обрано мову Java.

Java - мова програмування загального призначення. Відноситься до об'єктно-орієнтованим мовам програмування, до мовам з сильною типізацією. В Java реалізований механізм управління пам'яттю, який називається складальником сміття або garbage collector. Розробник створює об'єкти, а JRE за допомогою збирача сміття очищає пам'ять, коли об'єкти перестають використовуватися.

Також, при створенні програмного додатку було використано платформу Firebase, як середовище збереження бази даних.

Firebase - головною перевагою платформи в тому, що вона надає розробнику не відволікатися на створення бекенд, тобто прихованої від користувача програмної частини проекту, 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.

1.2 Розроблення, обґрунтування алгоритму функціонування та структурної схеми програмного додатку

Месенджери - невід'ємна частина людей, які звикли швидко передавати інформацію в будь-який час.

У месенджері користувач може створювати власний список контактів. Контакти можуть бути згруповані у групи за назвою. Більшість протоколів дозволяють зберігати список контактів на сервері, що створює певні зручності:

- користувач може переглядати свої контакти на іншому комп'ютері;
- користувач може задавати власні правила для контактів. Наприклад, список заблокованих (заборонених) контактів від яких він не бажає отримувати повідомлення («чорний список»). Або список контактів для яких його статус завжди видимий;
- користувач може зберігати власні примітки для контактів

Популярність ідеї використання месенджерів як нової платформи для комунікації в університетах стрімко зростає. Світовий ринок самих месенджерів вже сформований і постійно створюються нові корисні продукти, що дозволяють підтримувати комунікацію не тільки з родичами та друзями, а й з різними організаціями та учбовими закладами, що і являє для нас головною метою.

Тому вважаємо за необхідне удосконалити існуючу модель сучасної комунікації за допомогою впровадження месенджерів, що дозволяє підвищити ефективність управління підприємством (див. рис. 3).

Рисунок 2 – Удосконалена модель сучасної комунікації у месенджері

Поширення месенджерів пов’язано з тими перевагами які вони надають: ненав’язливість, асинхронність, конфіденційність, дешевизна, можливість докласти фото або відео.

Збільшення доступності смартфонів, розповсюдження і здешевлення швидкого мобільного інтернету також сприяє зростанню популярності месенджерів [5].

Використання месенджерів надає ряд переваг, серед яких:

1. Як правило, використовується мобільний номер телефону. Змінити номер телефону значно складніше, ніж електронну адресу.

2. Миттєві push-повідомлення. Можна бути впевненими в тому, що повідомлення дійде до користувача.

3. Персональність. Месенджери – досить приватна територія, на якій можна вирішувати проблеми користувача в більш довільній формі, на відміну від електронної пошти.

4. Різноманітність контенту. На відміну від традиційних засобів, месенджери дозволяють ділитися не тільки текстом, а й картинками, відео та ін.

5. Цілодобова підтримка.

6. Простота налаштування.

РОЗДІЛ 2

2.1 Розроблення та програмування програмного додатку

Проаналізувавши предметну область, було визначено мету проектування та необхідні завдання та обрано спосіб реалізації відповідно до вимог до формування функціональної частини мобільного додатку. Наступним кроком є розробка програмного додатка, що містить процес створення та впровадження програмних продуктів.

Першим етапом проектування мобільного додатка є створення контекстної діаграми А-0 в нотації IDEF, оскільки вона містить стислий та структурований опис процесу використання програми. При розробці програми була розроблена контекстна діаграма, як показано на рис 1. 2. Це верхня діаграма. Рисунок А-0 містить функціональний блок, вхідні та вихідні дані, елементи керування та інструменти. Діаграма побудована в середовищі BPWin.[6]

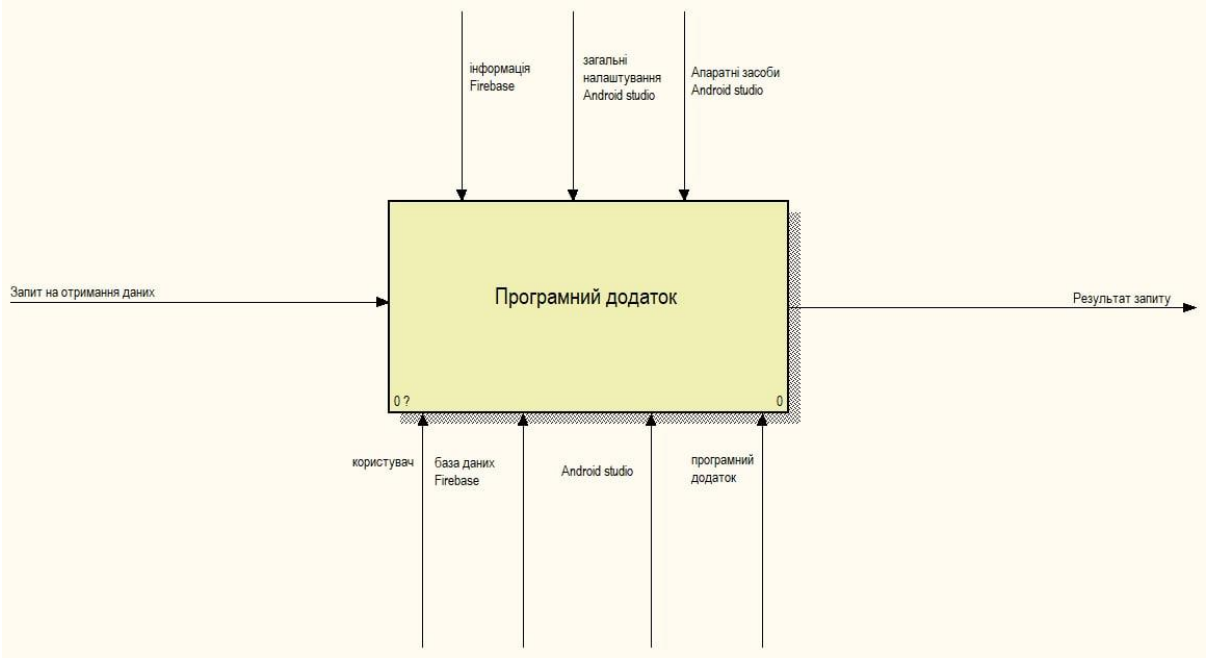

Рисунок 2.1 – Контекстна діаграма

Після опису процесу використання програмним додатком необхідно провести її розбиття на невеликі фрагменти, з метою детального опису, а саме функціональна декомпозиція. Діаграма А-0 була розбита на три функціональних блоки:

- аутентифікація користувача
- база даних користувачів
- створення чату

При досліджені предметної області описано такі сформовані дані:

- вхідні дані: запит на отримання даних
- вихідні дані: результат запиту
- контроль: інформація Firebase, загальні налаштування Android Studio, користувач.
- інструменти: апаратні засоби Android Studio, база даних Firebase, програмний додаток.

Діаграма першого рівня декомпозиції зображено на рис.2.2.

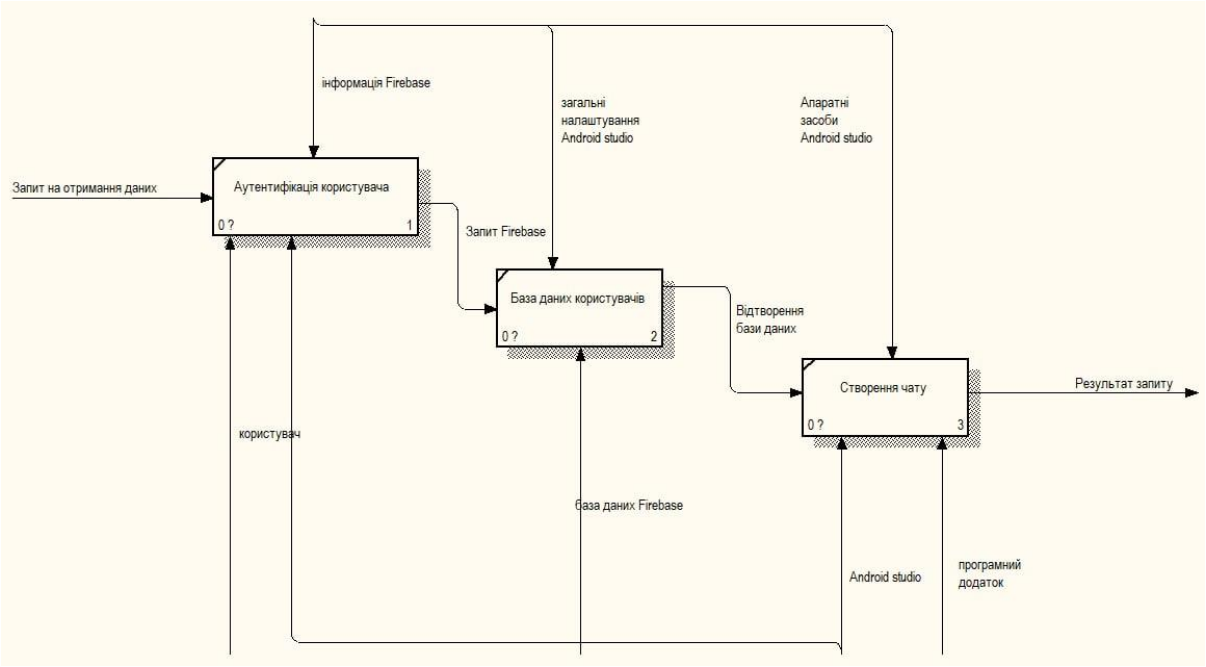

Рисунок 2.2 – Діаграма декомпозиції

Наступним етапом було проведено декомпозицію рівня “Аутентифікація користувача” на такі функціональні блоки: “Процес авторизації”, “Занесення до бази даних”, “Відображення користувачів”, “Створення чату користувачів”.

Діаграма декомпозиції другого рівня зображена на рис.2.3.

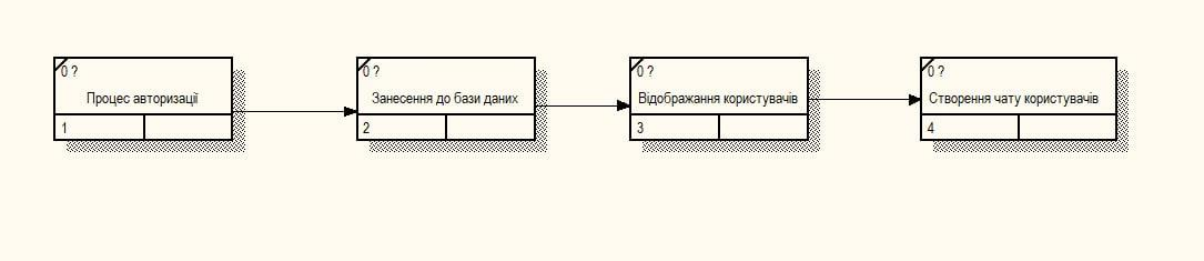

Рисунок 2.3 – Декомпозиція другого рівня

2.2. Вибір електронної бази даних

Для вирішення усіх поставлених завдань необхідно створити пристрій, який виконуватиме усі функції, розглянуті при розробці діаграм.

Основою є програмний додаток, завданням якого є реалізація технології підтримки учбового процесу на основі використання мобільних технологій з чітким алгоритмом послідовно виконуваних дій. Додаток створено на платформі Android Studio, яка являє собою інтегроване середовище створення додатків для андроїда. Android Studio перевершує конкурента за багатьма параметрами, до яких можна віднести: гнучкість середовища розробки; більший набір функцій; процес розробки, який підлаштовується під розробника.

Для зберігання даних було обрано платформу Firebase, яка використовується з Android Studio. Це і сервер, і база даних, і хостинг, і аутентифікація в одній платформі. Так, Firebase Realtime Database надає

розробникам API, який синхронізує дані додатки між клієнтами і зберігає їх в хмарному сховищі.[7]

Для управління базою даних достатньо відобразити її у браузері та переглядати дані у поточний час(рис. 2.4.)

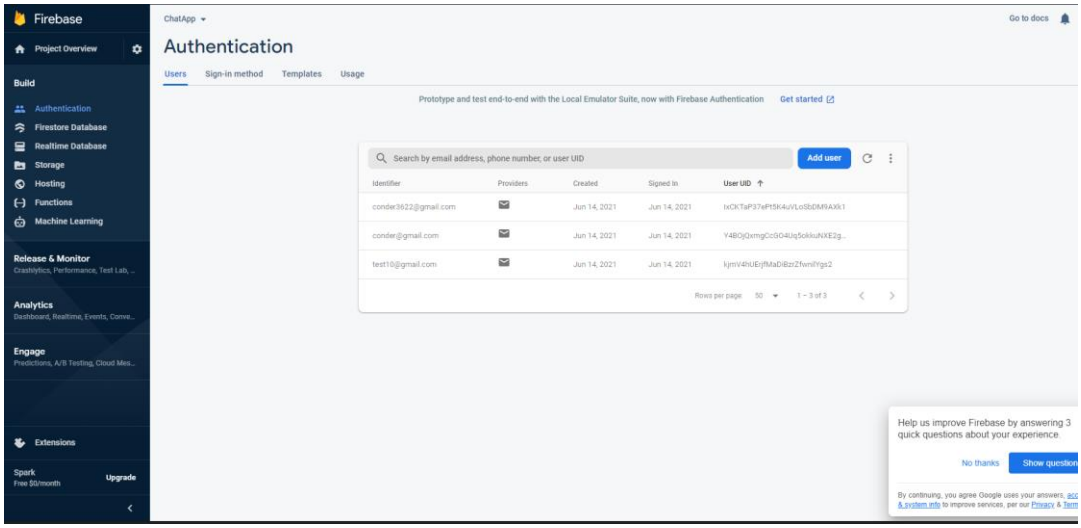

Рисунок 2.4 - База даних зареєстрованих користувачів

Для відображення повідомлень можна переглянути їх в базі даних, доречно нагадати, що повідомлення, користувачів та іншу інформацію може редагувати та видаляти адміністратор. [8] Перегляд повідомлень зображено на Рисунку 2.5.

Рисунок 2.5 - Перегляд повідомлень в базі даних

2.3 Проектування варіантів використання програмного додатку

Для створення діаграми варіантів використання додатку було визначено двох “акторів”: “Користувач” та “Адміністратор”.[9]

Основними функціями для “Користувач” визначено:

- аутентифікація;
- перегляд інформації через app;
- користування чатом;
- перегляд зареєстрованих користувачів.

“Адміністратор” має більш розширений функціонал:

- управління користувачами (видалення, додавання);
- Видалення та додавання інформації до Firebase;
- Управління повідомленнями (редагування, додавання, видалення).

Кожен “актор” має різні доступні функції в програмному додатку, таким чином “Користувач” не може самостійно видаляти та редагувати дані з бази. В той самий час “Адміністратор” має весь функціонал для управління програмним додатком.

На основі отриманих даних було розглянуто всі доступні до утворення варіанти використання додатку, було розроблено Use Case діаграма, представлена на рис.2.6.

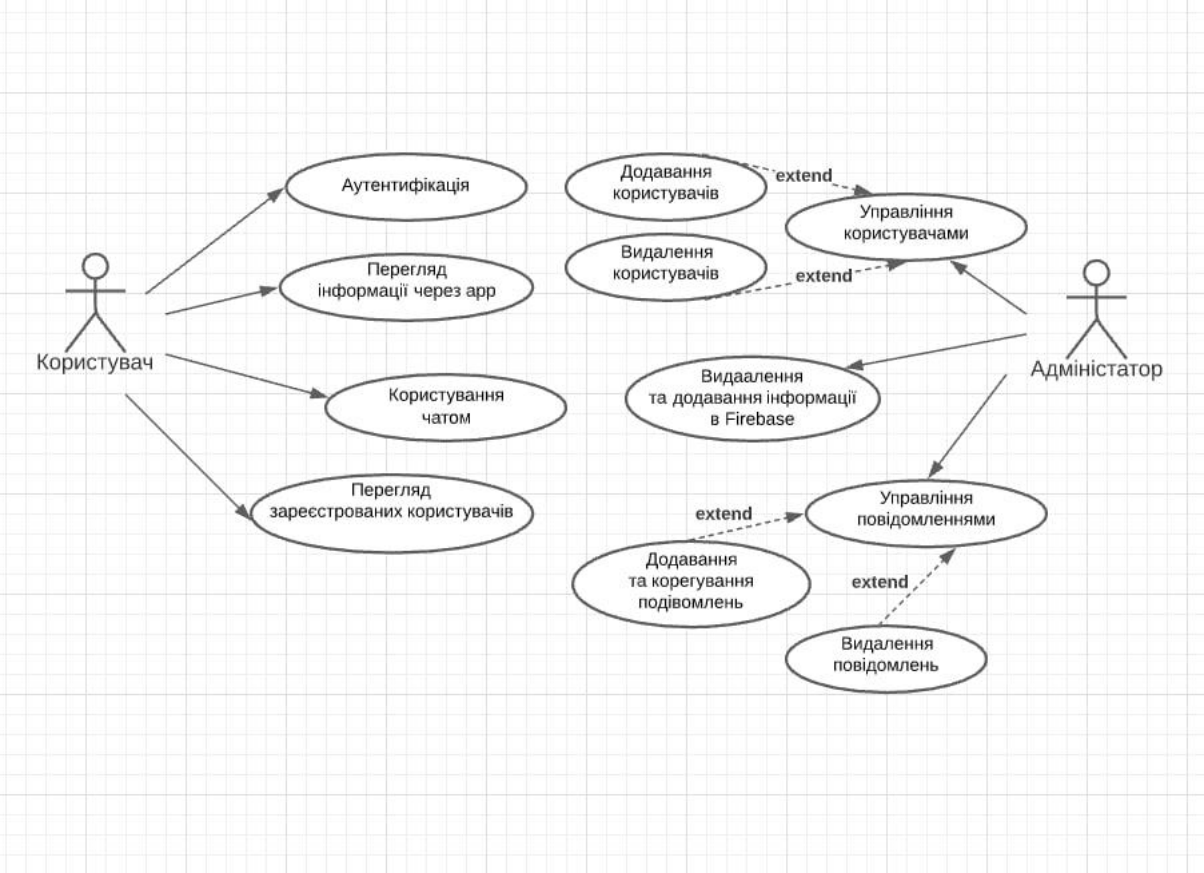

Рисунок 2.6 - Діаграма варіантів використання програмного додатку

На цьому етапі створюється контекстна діаграма з базовою структурою системи. Потім розроблено розгорнуту схему, щоб описати програму більш докладно. Діаграма варіантів використання була розроблена, щоб проілюструвати взаємодію програми з різними користувачами. Водночас були визначені всі можливі «актори» та варіанти використання програмного додатку.[10]

РОЗДІЛ 3

3.1 Архітектура програмного додатку

На рис. 3.1 наведено рисунок архітектури мобільного додатку. [11]

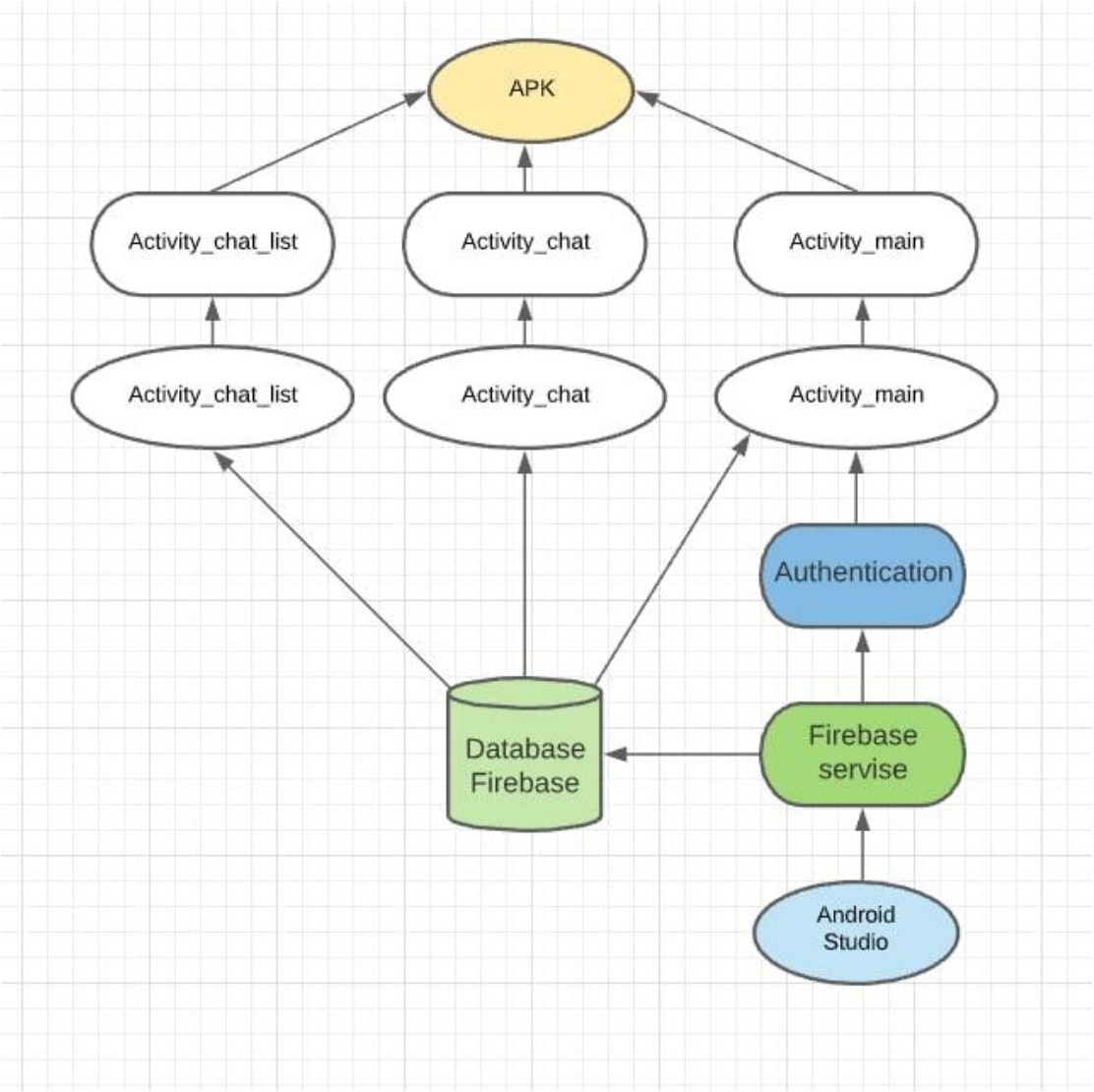

Рисунок 3.1 — Архітектура мобільного додатку

Користувач потрапляє у програмний додаток з головного екрану, який створено у середовищі Android Studio. Після чого реєструється, введені дані направляються до бази даних Firebase, де постійно зберігаються. Після реєстрації користувачу надається список тих, хто вже

зареєстрований у програмному додатку, та можливість створили чат, в якому надсилаються повідомлення для всіх учасників.

3.2 Пояснення програмної реалізації

У першому етапі розробки додатку є створення проекту на платформі Android Studio. Мова використання для написання програмного додатку було обрано Java. [12]

Демонстрація робочого середовища Android Studio (рис.3.2)

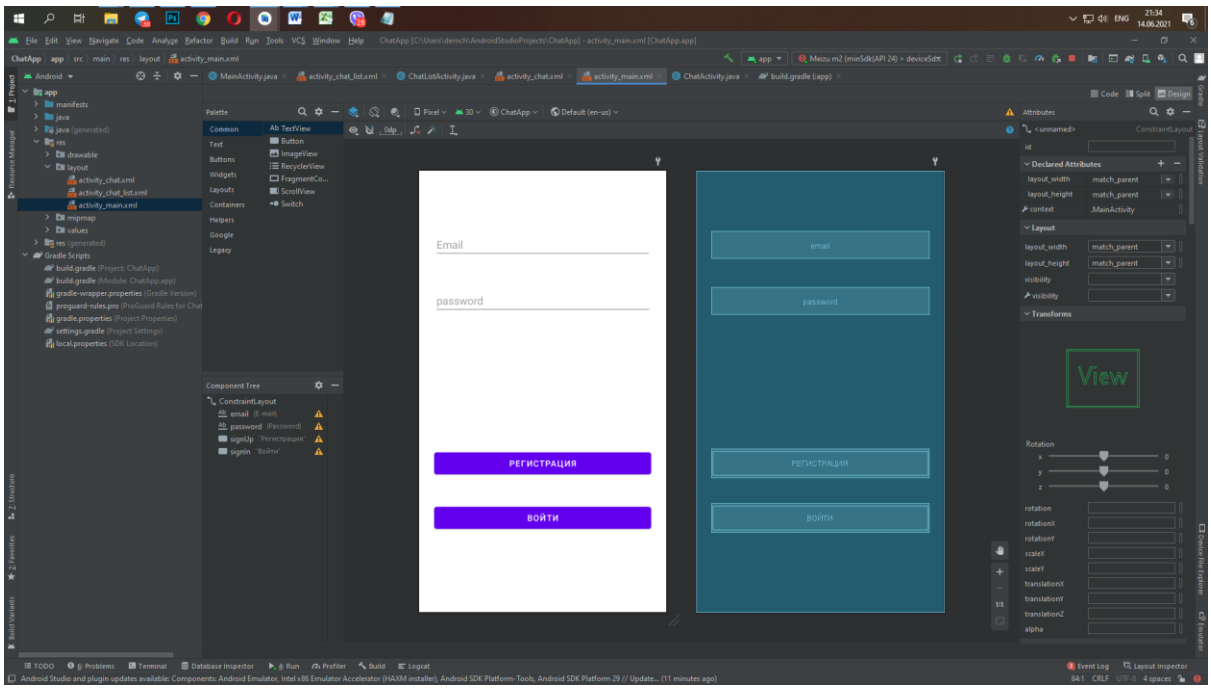

Рисунок 3.2 - Робоче середовище Android Studio

Дизайн додатку створено безпосередньо у середовищі Android Studio. Кінцевий дизайн зображено на рис. 3.3

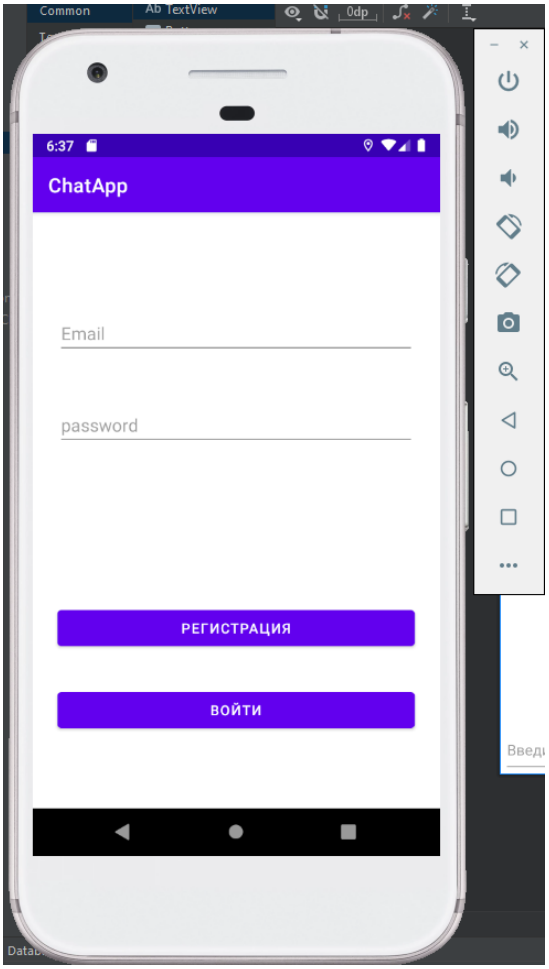

Рисунок 3.3 - Дизайн додатку в Android Studio

Також відображаються класи додатку та layout (макет, що розподіляє розташування елементів у додатку). Зображені вони на Рисунку 3.4.

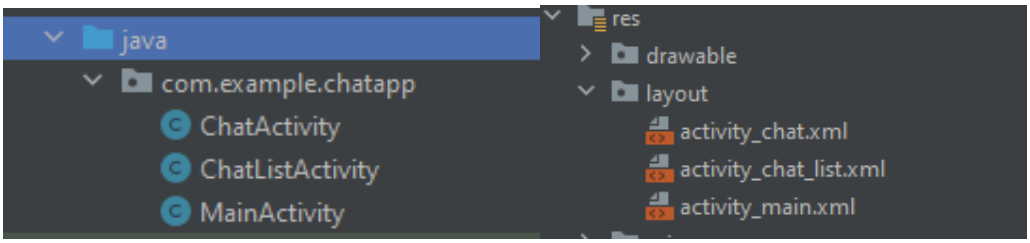

Рисунок 3.4 - класи додатку та layout додатку

Основні класи, які були використані, зазначено у табл. 3.1.

Таблиця 3.1 — Основні класи середовища [13]

Клас	Визначення функціоналу
ChatActivity	Діяльність чату дозволяє учасникам курсу проводити синхронну дискусію в режимі реального часу
ChatListActivity	Відображення поточних повідомлень у реальному часі.
MainActivity	Клас MainActivity представляє звичайний клас Java, на початку якого ідею визначення пакетів даних класу

3.3. Використання програмного додатку

Процес використання мобільного додатку налічує швидку та оперативну взаємодію між студентами кафедри Електроніки та їх викладачами. Користувач має можливість користуватися чатом, в якому як отримати потрібну інформацію, так и запитати самостійно. При цьому даний чат розрахований для користування обмеженою кількістю людей, саме це гарантує безпеку інформації, яка туди потрапляє.

Адміністратором додатку можуть виступати викладачі кафедри, та редагувати всю інформацію, що додається до чату, це гарантує прозорість у користуванні.

Процес тестування передбачає випробування програмного продукту, та головною метою має демонстрацію того, що даний мобільний додаток виконано відповідно технічному завданню та відповідає поставленим вимогам.

В процесі тестування помилок виявлено не було, що підтверджує правильну працездатність продукту, що дозволяє впровадити його в тестування.

На рисунках 3.5 — 3.12 представлені вікна роботи додатку.

При першому заходжені на головну сторінку додатку, користувач бачить поля для введення паролю та логіну.

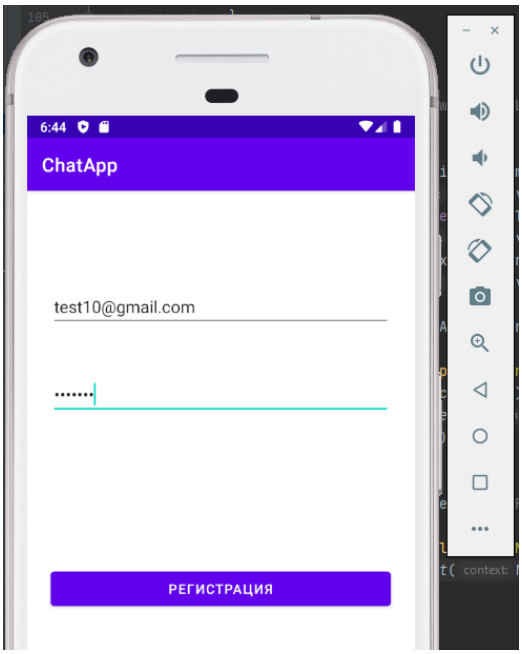

Рисунок 3.5 - Перевірка на реєстрацію користувача

Після успішної реєстрації користувач потрапляє до вікна, де відображені всі користувачі цього чату. [14]

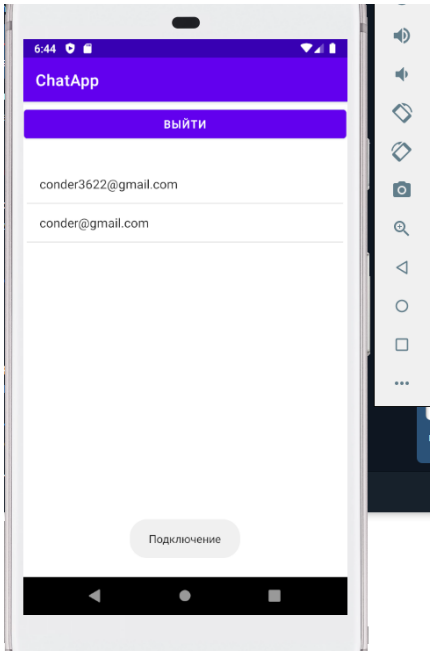

Рисунок 3.6 - Вікно після реєстрації, де відображаються всі зареєстровані користувачі

Якщо користувач неправильно вводить у відведених полях, йому відображається повідомлення з помилкою.(рис 3.7)

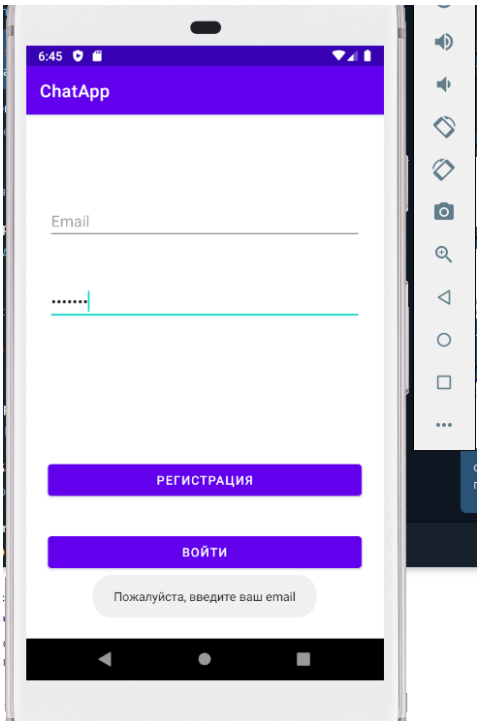

Рисунок 3.7 - Перевірка на запослення полів

Якщо такого користувача не існує, також відображається повідомлення.(рис.3.8)

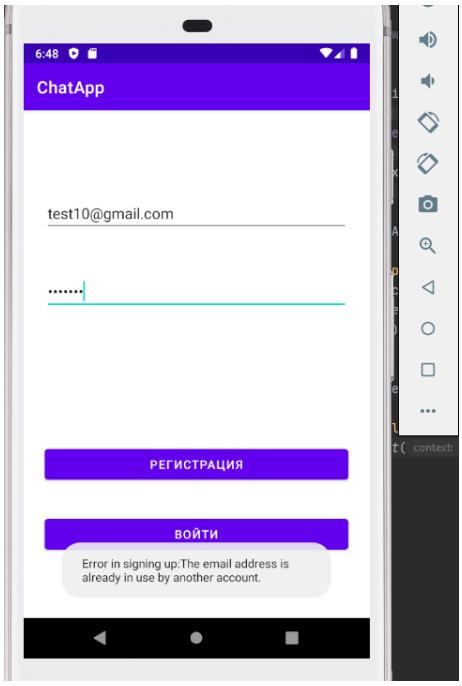

Рисунок 3.8 - Повідомлення, якщо користувач вже зареєстрований

Коли користувач зайшов до свого акаунту без помилок, може надіслати повідомлення до чату. Виглядає це наступним чином (рис.3.9)

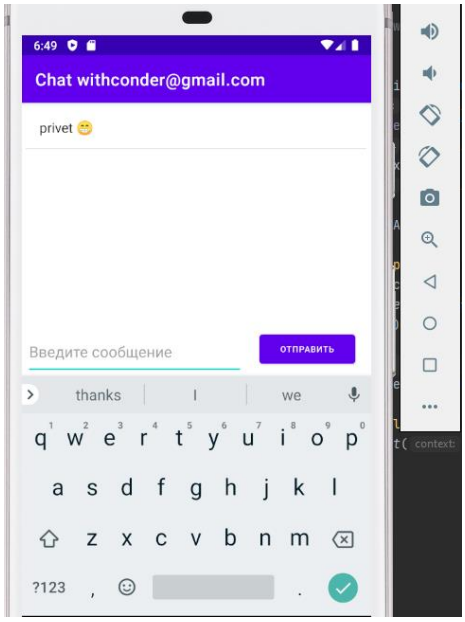

Рисунок 3.9 - Відправлення повідомлення до чату

ВИСНОВКИ

Сьогоднішній розвиток інформаційно-комунікаційних технологій розкриває неймовірні можливості для використання мобільних технологій в системі освіти для навчальних закладів. Мобільне навчання, що базується на інтенсивному застосуванні сучасних мобільних засобів та технологій, надає можливість зробити процес гнучким, доступним для багатьох і персоналізованим.

Мобільне навчання стає новим освітнім ступенем, на основі якого створюється нове навчальне середовище, де студенти та учні можуть отримати доступ до навчальних матеріалів у будь який час, що стимулює того, хто навчається, до самоосвіти та навчання протягом життя.

В процесі написання дипломної роботи була реалізована технологія підтримки учбового процесу на основі використання мобільних технологій, що дозволяються підтримувати та керувати процес навчання, а також відслідковувати показники підготовленості студентів.

Отримана інформаційна система у вигляді програмного мобільного чату обробляє результати діагностичних дослідів якості навчання, що реалізує більш якісний контакт між студентами та викладачами кафедри Електроніки.

Для розробки програмного додатку, а саме чату для спілкування, було задіяно платформу Android Studio, з мовою програмування Java.

На початку розробки даної роботи був проведений детальний аналіз предметної області проекту. Під час проектування для інтерактивного додатку, були розроблені: діаграма IDEF0, діаграма IDEF3, архітектура додатку, діаграма варіантів використання.

Актуальність даної роботи важко переоцінити, так як аналогів на кафедрі немає, а потреба у даному програмному додатку існує.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дафт Р.Л. Менеджмент / Річард Дафт Р.Л., 2008. – 864 с.

2. Лайхіф Д. М. комунікації: стратегії та навички / Д.М. Лайхіф, Д.М. Пенроуз., 2001. – 218 с.

3. Цандер Е. Практика управління. Обнинск: Титул, 1992.

4. Що таке Firebase [Електронний ресурс] – Режим доступу до ресурсу: https://avada-media.ua/services/firebase/

5. Чат-революция: почему боты убьют мобильные приложения [Електронний ресурс] – Режим доступу до ресурсу: http://rb.ru/longread/bots-arethe-new-apps/

6. Діаграма декомпозицій: [Електронний ресурс] – Режим доступу до ресурсу: http://www.itstan.ru/funk-strukt-analiz/diagrammy-dekompozicii.html

7. Створення графічного інтерфейсу: [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/java/android/1.3.php

8. Створення Firebase: [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/

9. Діаграма варіантів використання : [Електронний ресурс] – Режим доступу до ресурсу: http://khpi-iip.mipk.kharkiv.edu/library/case/leon/gl4/gl4.html

10. Основи UML: [Електронний ресурс] – Режим доступу до ресурсу: https://pro-prof.com/archives/2594

11. Архітектура програмного додатку, створення: [Електронний ресурс] – Режим доступу до ресурсу: http://dspace.wunu.edu.ua/jspui/bitstream/316497/24194/1/%D0%BE%D0%BF%D0%BE%D1%80%D0%BD%D0%B8%D0%B9%20%D0%BA%D0%BE%D0%BD%D1%81%D0%BF%D0%B5%D0%BA%D1%82%20%D0%BB%D0%B5%D0%BA%D1%86%D1%96%D0%B9.pdf

12. Робота в Android Studio: [Електронний ресурс] – Режим доступу до ресурсу: https://itproger.com/course/java-android

13. Основні класи Android Studio: [Електронний ресурс] – Режим доступу до ресурсу: http://developer.alexanderklimov.ru/android/java/class.php

14. Перевірка користувача на наявність доступу: [Електронний ресурс] – Режим доступу до ресурсу: https://coderoad.ru/63634326/%D0%BF%D1%80%D0%BE%D0%B2%D0%B5%D1%80%D0%BA%D0%B0-%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%D0%BD%D0%BE%D1%81%D1%82%D0%B8-email-%D0%B8-%D0%B8%D0%BC%D0%B5%D0%BD%D0%B8-%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F-Firebase-%D0%B2-Android-Studio

15. Створення месенджерів для навчального закладу: [Електронний ресурс] – Режим доступу до ресурсу: https://vrc.rv.ua/case_study/vet-marketing/

16. Пошук цільової аудиторії: [Електронний ресурс] – Режим доступу до ресурсу: https://creativesmm.com.ua/jak-znajjtu-svoju-cilovu-auditoriju/

17. Start Android - учебник по Android [Електронний ресурс]: – Режим доступу: http://startandroid.ru/ru/

ДОДАТОК А.
ТЕХНІЧНЕ ЗАВДАННЯ

1. **Призначення й мета створення програмного додатку**

1.1 Призначення програмного додатку

Створений додаток для кафедри електроніки орієнтований на використання студентами та викладачами.[15]

1.2 Мета створення програмного додатку

Формалізація мети роботи полягає в полегшенні пошуку інформації навчального процесу для студентів, знаходячись на факультеті, а також, інформацію про актуальні зміни у розкладі.

Для досягнення мети було виконано наступні етапи:

- збір необхідних матеріалів ;
- розробка ПП;
- проведення аналізу введених даних;
- тестування роботи продукту та аналіз результатів.

1.3 Цільова аудиторія

У цільовій аудиторії [16] системи можна виділити наступні групи:

1. Студенти.
2. Викладачі.
3. Співробітники.
4. Аспіранти.
5. Інші зацікавлені відвідувачі.

2. Вимоги до програмного додатку

2.1 Вимоги до структури й функціонування програмного додатку

Розроблюваний додаток повинен реалізовуватись у вигляді мобільного додатку, доступного на мобільній платформі Play Market [17]. Додаток повинен складатися із взаємозалежних розділів та мати необхідний набір функціональних можливостей.

2.2 Вимоги до персоналу

Для користуванням даним додатком потрібні навички володіння мобільним пристроєм. Користування додатком не потребує професійних вмінь. З боку користувача це вміння надсилати повідомлення, реєструватися.

Адміністратор додатку має мати навички в роботі з базою даних Firebase.

2.3 Вимоги до збереження інформації

Вся основна інформація буде зберігатися в базі даних Firebase.

2.4 Вимоги до розмежування доступу

Інформація додатку не є вільнодоступною. Доступ до бази даних Firebase має лише головний адміністратор.

Користувачів сайту можна розділити на 2 групи відповідно до прав доступу:

1. Користувачі
2. Адміністратор

Користувачі мають доступ тільки до загальнодоступної частини додатку.

Адміністратор може виконувати всі ті ж дії, що й Редактор, і крім того:

1. додавати користувачів із правами Редактора;
2. додавати й видаляти інформацію з баз даних.

Доступ до адміністративної частини повинен здійснюватися з використанням унікального логіна й пароля.

2.5 Основні вимоги. Структура додатку

Додаток повинен складатися з наступних розділів:

1. Головна.
   1.1 Розділ реєстрації
   1.3 Розділ авторизації
2. Розділ зареєстрованих користувачів
   2.1 Панель Користувачі
3. Розділ Створеного чату
   3.1 Панель Чат користувачів

2.6 Навігація

Інтерфейс користувача додатку має забезпечувати наочне, інтуїтивно зрозуміле представлення структури інформації, розміщеної на ньому, швидкий і логічний перехід до розділів і сторінок. Навігаційні елементи повинні забезпечувати однозначне розуміння користувачем їх змісту: посилання на сторінки повинні бути мати заголовок, умовні позначки відповідати загальноприйнятим. Графічні елементи навігації повинні бути мати альтернативний підпис.

Система повинна забезпечувати навігацію по всіх доступних користувачеві ресурсам і відображати відповідну інформацію. Для

навігації повинна використовуватися система контент-меню. Меню повинне являти собою текстовий блок (список гіперпосилань) у лівій колонці або у верхній частині сторінки (залежно від затвердженого дизайну).

2.7 Наповнення додатку (контент)

Сторінки всіх розділів сайту повинні формуватися програмним шляхом на підставі інформації з бази даних на сервері.
Android Studio — наповнення та редагування всіх програмних та візуальних даних виконується тільки головним адміністратором.
BD Firebase — виконується користувачами, які мають права доступу (головний адміністратор).

2.8 Вимоги до функціональних можливостей

Система керування контентом (адміністративна частина додатку) повинна надавати можливість додавання, редагування й видалення вмісту статичних і динамічних сторінок.

2.10 Вимоги до інформаційного забезпечення

Реалізація сайту відбувається з використанням:

- Android Studio
- Firebase

2.15 Вимоги до апаратного забезпечення

Апаратне забезпечення серверної частини повинне задовольняти наступним вимогам:

- Мобільний пристрій з операційною системою Android версії не менше 6.0
- Не менш 500 МБ вільного місця на диску.

3. Склад і зміст робіт зі створення додатку

Докладний опис етапів роботи зі створення додатку наведено в табл. 1.

Таблиця 1 – Етапи створення сайту

№	Склад і зміст робіт	Строк розробки (у робочих днях)
1	Розробка шаблону Web: Проектування розмітки та наповнення додатку	2-3 дні
2	Авторизація: Розроблення та реалізація блоку Авторизації	1 день
3	Адмін панель та особистий кабінет:	3 дня
4	Розробка шаблону Android:	3 дня
5	Підключення Firebase	1 день
6	Виконання програмної частини додатку	5 днів
7	Наповнення додатку	1 день
8	Налаштування зв`язку з Firebase	2 день
9	Завершення роботи: перевірка та тестування реалізованого функціоналу	3 дні
10	Загальна тривалість робіт і строк закінчення проекту	22 дні

4. Вимоги до складу й змісту робіт із введення сайту в експлуатацію

Для створення умов функціонування, при яких гарантується відповідність створюваного додатку вимогам сьогодення ТЗ і можливість його ефективної роботи, в організації Замовника повинен бути проведений певний комплекс заходів.

ДОДАТОК Б. Файли реалізації

```
package com.example.chatapp;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.util.Patterns;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

public class MainActivity extends AppCompatActivity {


    FirebaseAuth mAuth;
    EditText password, email;
    Button signUp, signIn;
    DatabaseReference databaseReference;
```

```
private boolean isEmailValid (CharSequence email){
if (email == null){
   return false;
} else{
   return Patterns.EMAIL_ADDRESS.matcher(email).matches();
}
}

public void sendUser(){
   Intent intent = new Intent(MainActivity.this, ChatListActivity.class);
   startActivity(intent);
}

@Override
public void onStart() {
   super.onStart();
   FirebaseUser user = mAuth.getCurrentUser();
   if (user != null){
      sendUser();
   }
}

@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);


   mAuth = FirebaseAuth.getInstance();
   databaseReference = FirebaseDatabase.getInstance().getReference();
   password = findViewById(R.id.password);
   email = findViewById(R.id.email);
   signUp = findViewById(R.id.signUp);
   signIn = findViewById(R.id.signIn);

   signUp.setOnClickListener(new View.OnClickListener() {
```

```
        @Override
        public void onClick(View v) {
           if (email.getText().toString().isEmpty()){
              Toast.makeText(MainActivity.this, "Пожалуйста, введите ваш email", Toast.LENGTH_SHORT).show();
           } else if (!isEmailValid(email.getText().toString())){
              Toast.makeText(MainActivity.this, "Напишите действующий email!", Toast.LENGTH_SHORT).show();
           } else if (password.getText().toString().isEmpty()){
              Toast.makeText(MainActivity.this, "Пожалуйста, введите свой пароль", Toast.LENGTH_SHORT).show();
           } else {

mAuth.createUserWithEmailAndPassword(email.getText().toString(), password.getText().toString()).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                   if (task.isSuccessful()){

databaseReference.child("users").child(mAuth.getCurrentUser().getUid()).child("email").setValue(email.getText().toString()).addOnCompleteListener(new OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull Task<Void> task) {
                           if (task.isSuccessful()){
                              Toast.makeText(MainActivity.this, "Подключение", Toast.LENGTH_SHORT).show();
                                 sendUser();
                           }
                        }
                     }).addOnFailureListener(new OnFailureListener() {
                        @Override
                        public void onFailure(@NonNull Exception e) {
                           Toast.makeText(MainActivity.this, "Ошибка при сохранении данных:"+e.getMessage(), Toast.LENGTH_SHORT).show();
                        }
```

```
                    });{

                    }
                }

            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                Toast.makeText(MainActivity.this, "Error in signing up:" +
e.getMessage(), Toast.LENGTH_SHORT).show();
            }
        });
    }
  });

  signIn.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View v){
      if (email.getText().toString().isEmpty()){
        Toast.makeText(MainActivity.this, "Пожалуйста, введите ваш
email", Toast.LENGTH_SHORT).show();
      } else if (!isEmailValid(email.getText().toString())){
        Toast.makeText(MainActivity.this, "Напишите действующий
email!", Toast.LENGTH_SHORT).show();
      } else if (password.getText().toString().isEmpty()){
        Toast.makeText(MainActivity.this, "Пожалуйста, введите свой
пароль", Toast.LENGTH_SHORT).show();
      } else {
        mAuth.signInWithEmailAndPassword(email.getText().toString(),
password.getText().toString()).addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
          @Override
          public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()){
```

```
                        Toast.makeText(MainActivity.this, "Подключение!", Toast.LENGTH_SHORT).show();
                        sendUser();
                    }
                }
            }).addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    Toast.makeText(MainActivity.this, "Ошибка при сохранении данных: "+ e.getMessage(), Toast.LENGTH_SHORT).show();
                }
            });
        }
    }
});

    }
}
```

```
package com.example.chatapp;

import androidx.appcompat.app.AppCompatActivity;
import androidx.annotation.NonNull;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.Toast;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
```

```
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;
import java.util.Arrays;

public class ChatListActivity extends AppCompatActivity {

    ListView userListView;
    ArrayAdapter arrayAdapter;
    ArrayList<String> users = new ArrayList<>();
    FirebaseAuth mAuth;
    DatabaseReference databaseReference;
    Button signOut;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_chat_list);

        mAuth = FirebaseAuth.getInstance();
        databaseReference = FirebaseDatabase.getInstance().getReference();
        userListView = findViewById(R.id.userListView1);
        signOut = findViewById(R.id.signOut);

        databaseReference.child("users").addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                if (snapshot.exists()){
                    for (DataSnapshot dataSnapshot : snapshot.getChildren()){
                        String email = dataSnapshot.child("email").getValue().toString();
                        if (!email.equals(mAuth.getCurrentUser().getEmail())){
                            users.add(email);
                        }
                    }
```

```
                arrayAdapter = new ArrayAdapter(ChatListActivity.this, android.R.layout.simple_list_item_1, users);
                userListView.setAdapter(arrayAdapter);
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {
            Toast.makeText(ChatListActivity.this, "Не удалось загрузить пользователей", Toast.LENGTH_SHORT).show();
        }
    });

    signOut.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            mAuth.signOut();
            Intent intent = new Intent(ChatListActivity.this, MainActivity.class);
            startActivity(intent);
        }
    });

    userListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            Intent intent = new Intent(ChatListActivity.this, ChatActivity.class);
            intent.putExtra("email", users.get(position));
            startActivity(intent);
        }
    });
  }
}

package com.example.chatapp;
```

```
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

public class ChatActivity extends AppCompatActivity {
    EditText message;
    Button send;
    ListView chatListView;
    ArrayAdapter arrayAdapter;
    ArrayList<String> messages =new ArrayList<>();
    FirebaseAuth mAuth;
    DatabaseReference databaseReference;


    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_chat);

        mAuth =FirebaseAuth.getInstance();
        databaseReference = FirebaseDatabase.getInstance().getReference();

        message = findViewById(R.id.message);
        send = findViewById(R.id.send);
        chatListView = findViewById(R.id.chatListView);

        Intent intent = getIntent();

        String otherEmail =intent.getStringExtra("email");
        String email = mAuth.getCurrentUser().getEmail();

        setTitle("Chat with " + otherEmail);

        send.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (message.getText().toString().isEmpty()){
                    Toast.makeText(ChatActivity.this,     "Напишите     сообщение!",
Toast.LENGTH_SHORT).show();
                } else {

                    Map<String, Object> messageData = new HashMap<>();
                    messageData.put("sender", email);
                    messageData.put("receiver", otherEmail);
                    messageData.put("message", message.getText().toString());


databaseReference.child("chats").addListenerForSingleValueEvent(new
ValueEventListener() {
                        @Override
                        public void onDataChange(@NonNull DataSnapshot snapshot) {
                            int count;
```

```
                    if (snapshot.exists()){
                        count = (int) (snapshot.getChildrenCount() +1);
                    } else {
                        count = 1;
                    }

databaseReference.child("chats").child(String.valueOf(count)).setValue(messag
eData).addOnCompleteListener(new OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull Task<Void> task) {
                            if (task.isSuccessful()){
                                message.setText("");

                            }
                        }
                    }).addOnFailureListener(new OnFailureListener() {
                        @Override
                        public void onFailure(@NonNull Exception e) {
                            Toast.makeText(ChatActivity.this, "Error in sending
message:" + e.getMessage(), Toast.LENGTH_SHORT).show();
                        }
                    });

                }

                @Override
                public void onCancelled(@NonNull DatabaseError error) {

                }
            });
        }
    }
});

    databaseReference.child("chats").addValueEventListener(new
ValueEventListener() {
        @Override
```

```
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            if (snapshot.exists()){
                for (DataSnapshot dataSnapshot : snapshot.getChildren()){
                    if
(dataSnapshot.child("sender").getValue().toString().equals(email)                ||
dataSnapshot.child("receiver").getValue().toString().equals(email)){
                        String                              message                =
dataSnapshot.child("message").getValue().toString();
                        if
(!dataSnapshot.child("sender").getValue().toString().equals(email)){
                            message = "> " + message;
                        }
                        messages.add(message);
                    }
                }
                arrayAdapter        =        new        ArrayAdapter(ChatActivity.this,
android.R.layout.simple_list_item_1, messages);
                chatListView.setAdapter(arrayAdapter);
            }

        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }
    });

  }
}
```