БАКАЛАВРСЬКА

РОБОТА

КР.КІ-15.00.00.000 ПЗ

Група КІ-21-1

Пена Йосип

Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти і газу
Інститут інформаційних технологій
Кафедра комп'ютерних систем і мереж

Пена Йосип-Альберто

УДК 004.4

БАКАЛАВРСЬКА РОБОТА

Розробка мобільного додатку для організації онлайн голосування засобами Java

Комп'ютерна інженерія
(назва освітньої програми)

123 - Комп'ютерна інженерія
(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Здобувач освітнього ступеня *Пена Й.А.*
(підпис, ініціали та прізвище здобувача)

Науковий керівник *Гарасимів Тарас Григорович, асистент*
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту

Завідувач кафедри КСМ

д.т.н.,проф. *С.І. Мельничук*
(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2025 рік

Івано-Франківський національний технічний університет нафти і газу

(повне найменування вищого навчального закладу)

Факультет *інформаційних технологій*

Кафедра *комп'ютерних систем і мереж*

Освітній ступінь *бакалавр*

Спеціальність *123 – Комп'ютерна інженерія*

ЗАТВЕРДЖУЮ

Завідувач кафедри КСМ

(С.І. Мельничук)

«_____» _____ 2025 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Пені Йосипу-Альберто

1. Тема проекту (роботи) *Розробка мобільного додатку для організації онлайн голосування засобами Java*

керівник проекту (роботи) *Гарасимів Т.Г., асистент*

затверджені наказом вищого навчального закладу від «*05*» 05 2025 року *№275/7*

2. Строк подання студентом роботи *12 червня 2025р.*

3. Вихідні дані до роботи *Матеріали і результати отримані під час проходження переддипломної практики, методичні вказівки, технічна література.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) *1. Аналіз предметної області. 2. Розробка структури додатку та бази даних. 3. Реалізація графічного дизайну та огляд додатку.*

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

_____.

6. Консультанти розділів роботи

7. Дата видачі завдання *29 січня 2025р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	*Постановка задачі та обґрунтування необхідності розробки мобільного додатку для організації голосування*	*Лютий, 2025*	
2	*Збір інформації про онлайн сервіси голосування*	*Березень, 2025*	
3	*Розробка структури мобільного додатку*	*Квітень, 2025*	
4	*Розробка мобільного додатку*	*Травень, 2025*	
5	*Охорона праці*	*Травень, 2025*	
6	*Оформлення пояснювальної записки*	*Червень, 2025*	

Студент _____ *Пена Й.А.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Гарасимів Т.Г.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська робота присвячена розробці мобільного додатку, який призначений для організації онлайн голосування.

Основною метою роботи є створення зручної та простої у використанні програми, що дозволяє користувачам легко створювати голосування, а також брати в них участь.

Структура бакалаврської роботи включає кілька основних частин: вступ, загальну характеристику предметної області, проєктування та реалізацію бази даних, а також розробку графічного інтерфейсу користувача.

У вступі обґрунтовано актуальність теми дослідження, визначено мету та сформульовано основні задачі, які необхідно вирішити для досягнення поставлених цілей.

Перший розділ роботи присвячений аналізу предметної області, у якому детально розглядаються основні поняття, а також проводиться огляд і порівняння існуючих аналогічних сервісів для онлайн голосування.

У другому розділі наведено опис структури бази даних, що використовується в додатку, а також загальний огляд архітектури мобільного додатку, що дозволяє зрозуміти його будову та принципи функціонування.

Третій розділ присвячено демонстрації сценаріїв роботи додатку, а також опису деяких ключових особливостей реалізації програмного коду, що підкреслює важливі технічні рішення, застосовані під час розробки.

Ключові слова: голосування, тести, база даних, додаток, Android, Java, Kotlin.

ANNOTATION

The bachelor's thesis is dedicated to the development of a mobile application for organizing online voting.

The main goal of the work is to create a user-friendly and easy-to-use program that allows users to easily create votes and participate in them.

The structure of the bachelor's thesis includes several key parts: introduction, general characteristics of the subject area, design and implementation of the database, and the development of the graphical user interface.

The introduction substantiates the relevance of the research topic, defines the goal, and formulates the main tasks that need to be solved to achieve the set objectives.

The first chapter is devoted to the analysis of the subject area, where the main concepts are examined in detail, and an overview and comparison of existing similar online voting services are provided.

The second chapter presents the description of the database structure used in the application, as well as a general overview of the mobile application's architecture, allowing an understanding of its design and principles of operation.

The third chapter focuses on demonstrating the application's use scenarios and describing some key features of the program code implementation, highlighting important technical decisions made during development.

Keywords: voting, tests, database, application, Android, Java, Kotlin.

ЗМІСТ

ВСТУП

У сучасну епоху стрімкого розвитку цифрових технологій мобільні додатки стали невід'ємною складовою повсякденного життя мільйонів людей по всьому світу. Вони охоплюють широке коло сфер — від комунікацій та розваг до фінансів, освіти і навіть політики.

Одним із перспективних напрямів їхнього застосування є організація та проведення онлайн-голосувань. Завдяки мобільним технологіям з’явилася можливість значно спростити та модернізувати процес збору думок громадськості, ухвалення колективних.

Актуальність обраної теми обумовлена зростаючою потребою у сучасних цифрових інструментах для швидкого, прозорого та надійного проведення голосувань. У світі, де віддалена участь у громадському житті, бізнес-процесах і навчанні стає нормою, можливість організувати електронне голосування за допомогою мобільного додатку відкриває нові горизонти для демократії, управління та соціальної взаємодії. Такий підхід дозволяє не лише знизити витрати на організацію процесу, а й забезпечити високий рівень доступності, прозорості та довіри з боку користувачів.

Для реалізації подібних проєктів найдоцільніше використовувати сучасні технології розробки мобільного програмного забезпечення, зокрема мови програмування Java та Kotlin, які є основними інструментами створення Android-додатків. Вони забезпечують розробників широкими можливостями щодо проектування інтерфейсів, реалізації логіки додатків, взаємодії з базами даних та серверною частиною, а також впровадження алгоритмів захисту персональних даних. Kotlin, як більш сучасна мова, відома своєю лаконічністю та безпечністю, тоді як Java — своєю стабільністю та широкою підтримкою серед спільноти розробників.

Об’єктом дослідження у цій роботі є процес створення мобільного додатку для онлайн-голосування, що включає всі етапи життєвого циклу програмного продукту: від аналізу потреб користувачів та постановки задач до

створення графічного інтерфейсу, програмування функціональних можливостей та впровадження механізмів захисту інформації.

Предметом дослідження є конкретні засоби та методи, що застосовуються під час розробки такого додатку.

Метою роботи є створення повнофункціонального мобільного додатку для онлайн-голосування, який би відповідав сучасним вимогам до зручності користування та функціональності. Додаток має бути інтуїтивно зрозумілим для користувача, надавати можливість проводити голосування у зручній формі, переглядати результати в реальному часі.

У ході дослідження буде застосовано низку методів, зокрема:

- аналіз наукової, технічної та методичної літератури з питань мобільного програмування, електронного голосування;
- проектування інтерфейсу користувача з урахуванням принципів юзабіліті та сучасних дизайнерських трендів;
- розробка програмної логіки з використанням інструментів Android SDK, Kotlin та Java;
- тестування функціоналу на різних пристроях для виявлення та усунення можливих помилок.

Результатом роботи стане готовий до впровадження мобільний додаток для онлайн-голосування, який може бути адаптований під різні сценарії.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд предметної області

Онлайн-голосування поступово перетворюється на важливий інструмент у процесах ухвалення рішень, завдяки стрімкому розвитку цифрових технологій та зростаючій потребі суспільства в швидких, доступних і прозорих механізмах комунікації.

Сучасні онлайн-голосування активно застосовуються в різних сферах:

Державне управління: У деяких країнах вже запроваджено електронне голосування як альтернативу звичайним виборчим дільницям. Прикладами можуть бути Естонія, де громадяни можуть голосувати онлайн на парламентських виборах, використовуючи національну ID-карту. Електронне голосування дозволяє спростити організацію виборів, зменшити черги, уникнути підробок і полегшити участь громадян, які перебувають за кордоном.

Корпоративне середовище: У компаніях та акціонерних товариствах часто виникає потреба у прийнятті колективних рішень шляхом голосування — наприклад, під час зборів акціонерів або прийняття важливих внутрішніх рішень. Онлайн-голосування забезпечує оперативний зворотний зв'язок, можливість документального підтвердження рішень і мінімізує витрати часу, пов’язані з фізичними зустрічами.

Освітні установи: У школах, коледжах та університетах онлайн-голосування застосовується для обрання представників студентського самоврядування, голосування за реалізацію ініціатив, оцінювання якості викладання або вибору позакласних активностей. Такий формат зручний для учасників, підвищує рівень зацікавленості студентства та спрощує адміністрування процесу.

Громадські організації та ініціативи: Неприбуткові організації, місцеві громади та ініціативні групи часто використовують онлайн-голосування для

ухвалення рішень щодо розподілу бюджету участі, вибору пріоритетних проєктів або оцінки ефективності вже реалізованих заходів. Це забезпечує прозорість процесу, відкритість і залучення більшої кількості учасників.

З огляду на настільки широке використання, виникає потреба у створенні та розвитку додатку для онлайн-голосування, який має поєднувати в собі простоту у використанні, зручність інтерфейсу, а також доступність на різних мобільних пристроях.

Особливо важливим є баланс між функціональністю та простотою — додаток не повинен бути перевантаженим складними налаштуваннями або заплутаною навігацією. Це дозволить залучити користувачів різного віку та рівня цифрової грамотності, забезпечуючи максимальну ефективність та охоплення цільової аудиторії.

Таким чином, зростання попиту на онлайн-голосування безпосередньо формує запит на розробку ефективних програмних рішень, здатних задовольнити потреби сучасного суспільства у динамічному, цифровому середовищі.

1.2 Огляд аналогів

У рамках розробки мобільного додатку для онлайн-голосування доцільним є попередній аналіз найбільш поширених онлайн-сервісів, які вже використовуються для організації опитувань, тестування, збору думок і прийняття колективних рішень. Такий аналіз дозволяє визначити ключові переваги й недоліки існуючих рішень, а також виявити найбільш вдалі підходи до реалізації функціоналу та інтерфейсу.

Проаналізуємо найбільш поширені онлайн сервіси, які використовуються для організації й проведення голосування.

Розглянемо спершу рішення від Google, а саме Forms.

Google Forms (рис.1.1) – це безкоштовний онлайн-сервіс від компанії Google, який дозволяє створювати форми, анкети, опитування, тести та реєстраційні форми [11]. Кожен користувач Google сервісів має доступ до форм.

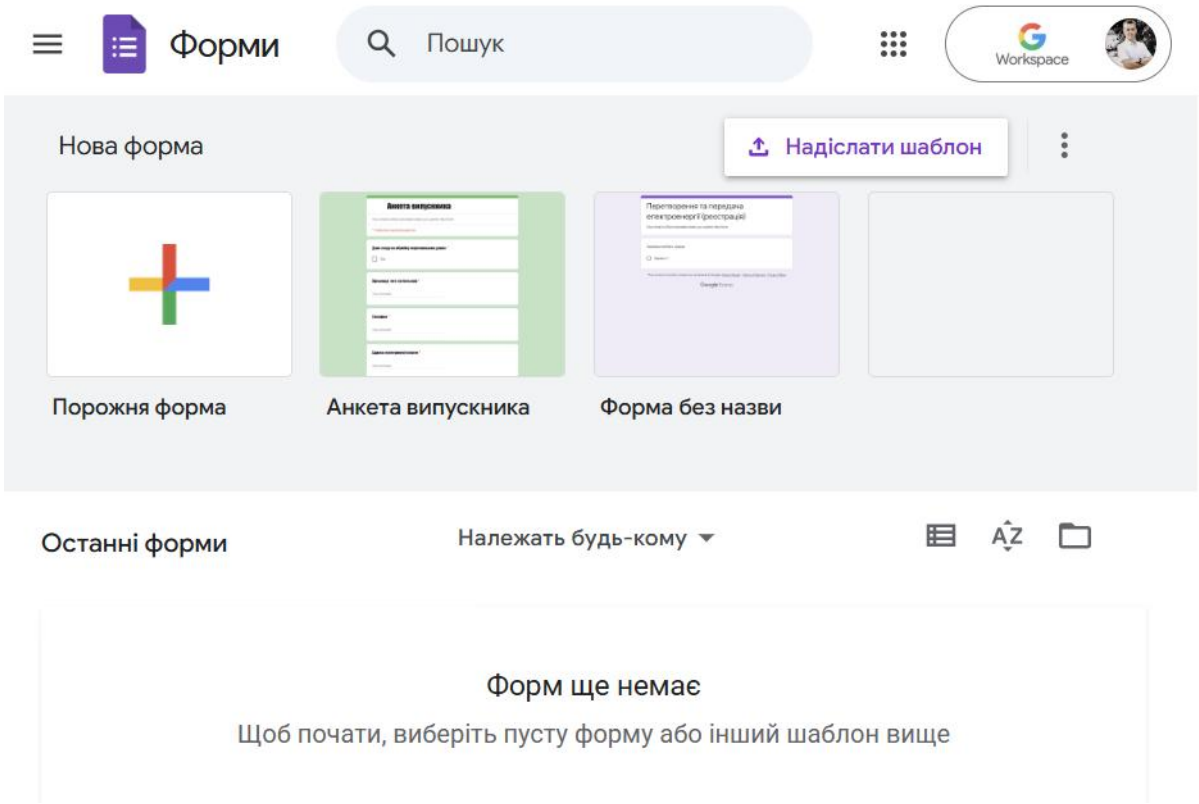

Рисунок 1.1 – Онлайн сервіс Google Forms

Форми можна надсилати за посиланням. Тобто доступ до них вільний, а користувачам невідомо, що їм потрібно пройти голосування чи тестування.

На рисунку 1.2 зображена сторінка створення форми. Сама сторінка виглядає не переповненою, але і не максимально спрощена для користувачів.

Серед переваг даного сервісу можна відмітити можливість створення тестів і зручний збір відповідей.

Ще одним потужним сервісом для створення опитувань є Typeform – платформа, що вирізняється сучасним дизайном та інтуїтивним інтерфейсом (рис. 1.3) [12]. Основна ідея Typeform – зробити процес проходження опитування приємним і схожим на діалог із користувачем.

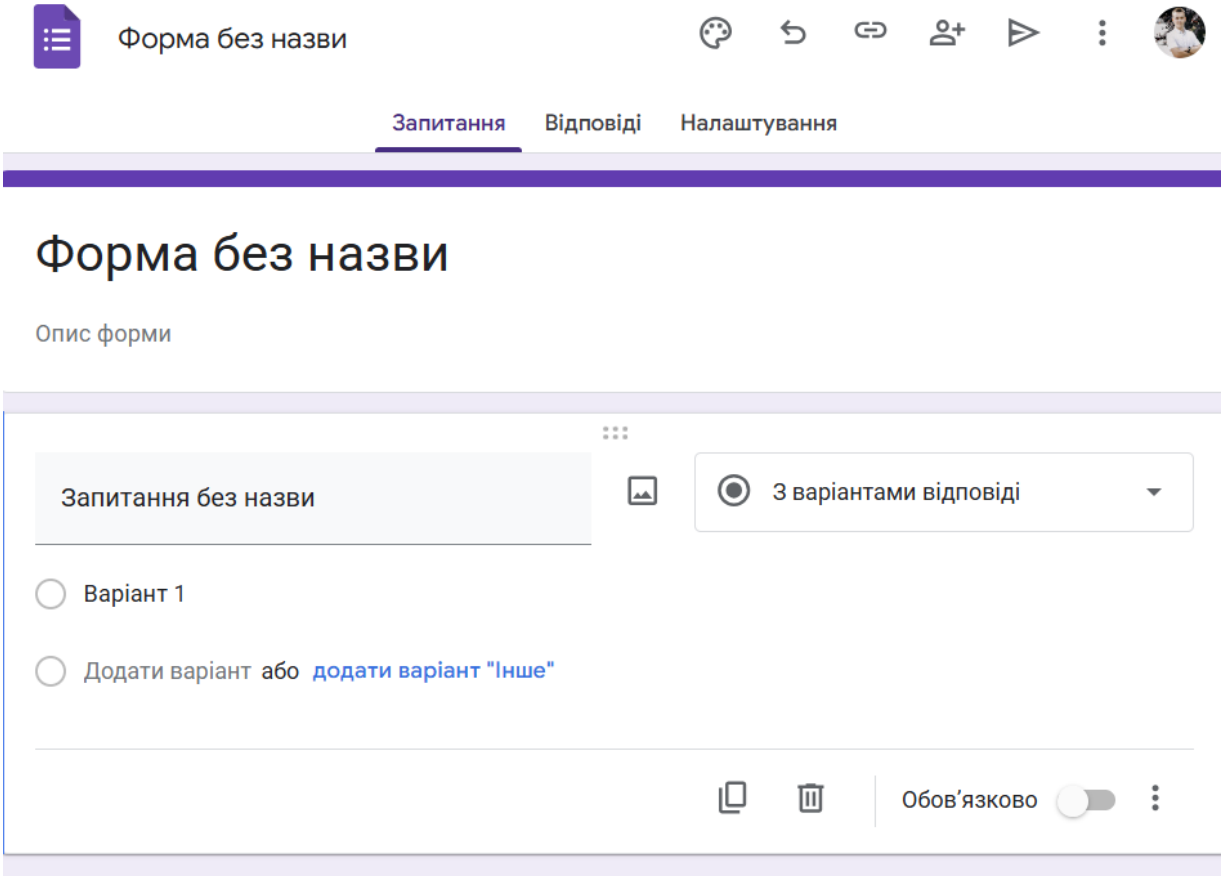

Рисунок 1.2 – Створення форми в Google Forms

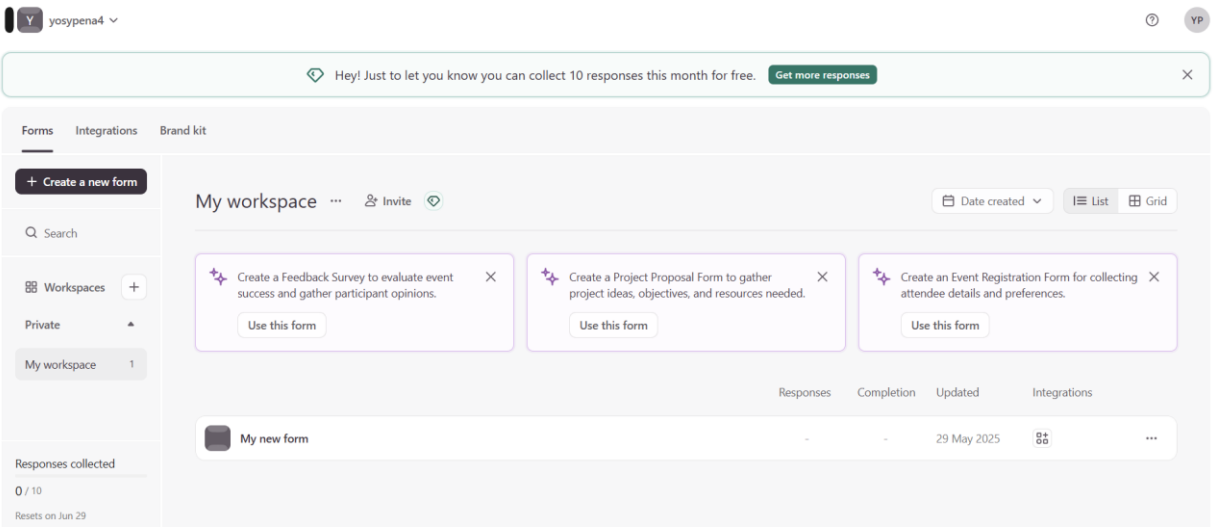

Рисунок 1.3 – Онлайн сервіс TypeForm

Сервіс підтримує інтерактивний формат опитування, де кожне питання подається окремо, що знижує когнітивне навантаження на користувача. При створенні форми доступні численні інструменти налаштування (рис. 1.4).

Є можливість створювати власний дизайн форм, це зроблено в дуже гнучкому та не зовсім легкому виді.

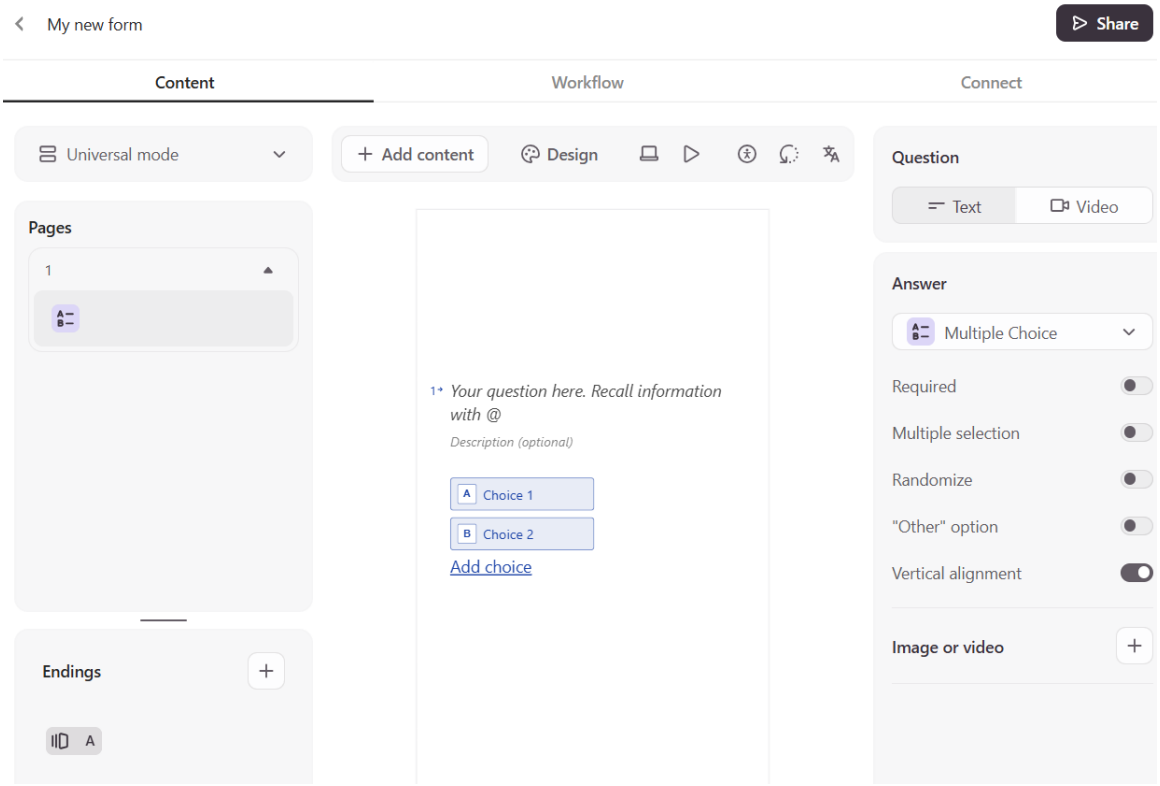

Рисунок 1.4 – Створення форми у TypeForm

SurveyMonkey є одним із найстаріших і найпопулярніших онлайн-сервісів для створення опитувань, широко використовуваним у бізнесі, маркетингу та дослідженнях (рис. 1.5) [13].

Серед преваг даного сервісу можна відмітити послідовність розділів для створення дизайну, з'єднання, збирання відповідей та аналіз результатів, проте такий підхід не є швидким у виконанні самого голосування.

Також розглянемо можливість голосування у таких додатках як Дія та Telegram.

У державному додатку Дія голосування створюють адміністратори самого сервісу, стосуються, в більшості, державних рішень.

У месенджері Telegram є можливість у групових чатах створювати голосування.

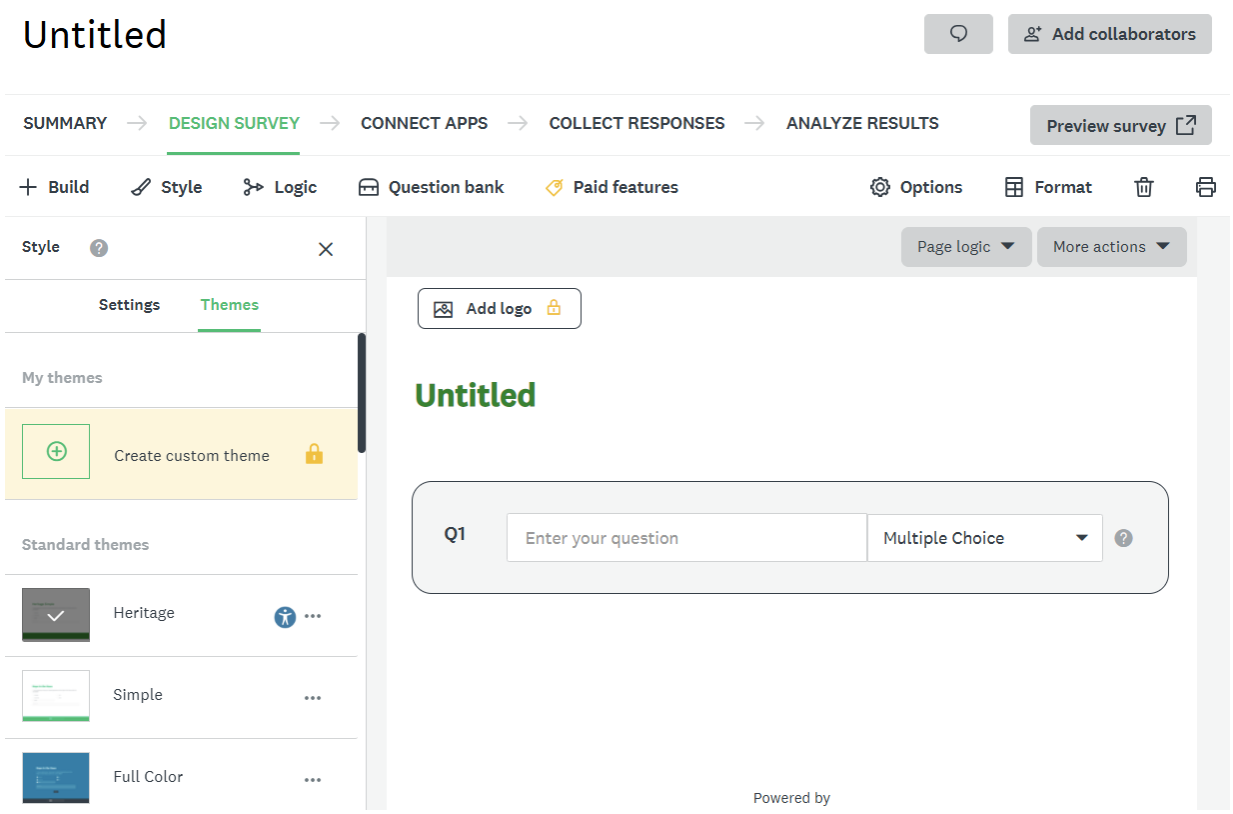

Рисунок 1.5 – Інтерфейс створення опитування у SurveyMonkey

Функціонал таких голосувань обмежений, оскільки ці додатки з вбудованим голосуванням передбачені не тільки для цієї функції. Проте перевагою даних вбудованих голосувань є простий та зрозумілий інтерфейс.

На рисунку 1.6 зображено функціонал та реалізація дизайну голосування в державному додатку Дія.

Дизайн інтерфейсу додатку Дія саме для учасників голосування є досить доступний для простого користувача, що є великою перевагою і надалі буде взятий за основу під час розробки інтерфейсу власного додатку.

На рисунку 1.7 зображено функціонал голосування в месенджері Telegram.

Форма створення голосування в додатку Telegram виконано у дуже зручний та легкий спосіб. Для простого користувача не буде перешкод для створення повноцінного голосування. У подальшій розробці інтерфейсу власного додатку буде взято за основу саме такий дизайн форми створення голосування.

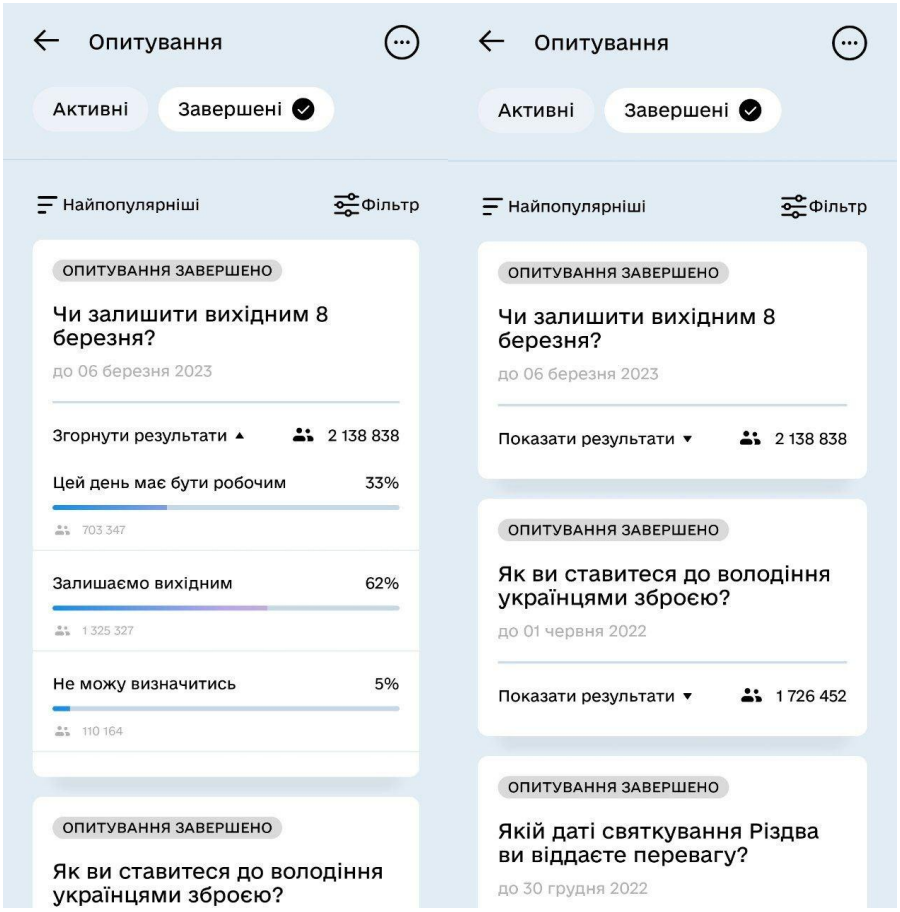

Рисунок 1.6 – Інтерфейс функціоналу голосування в державному додатку Дія

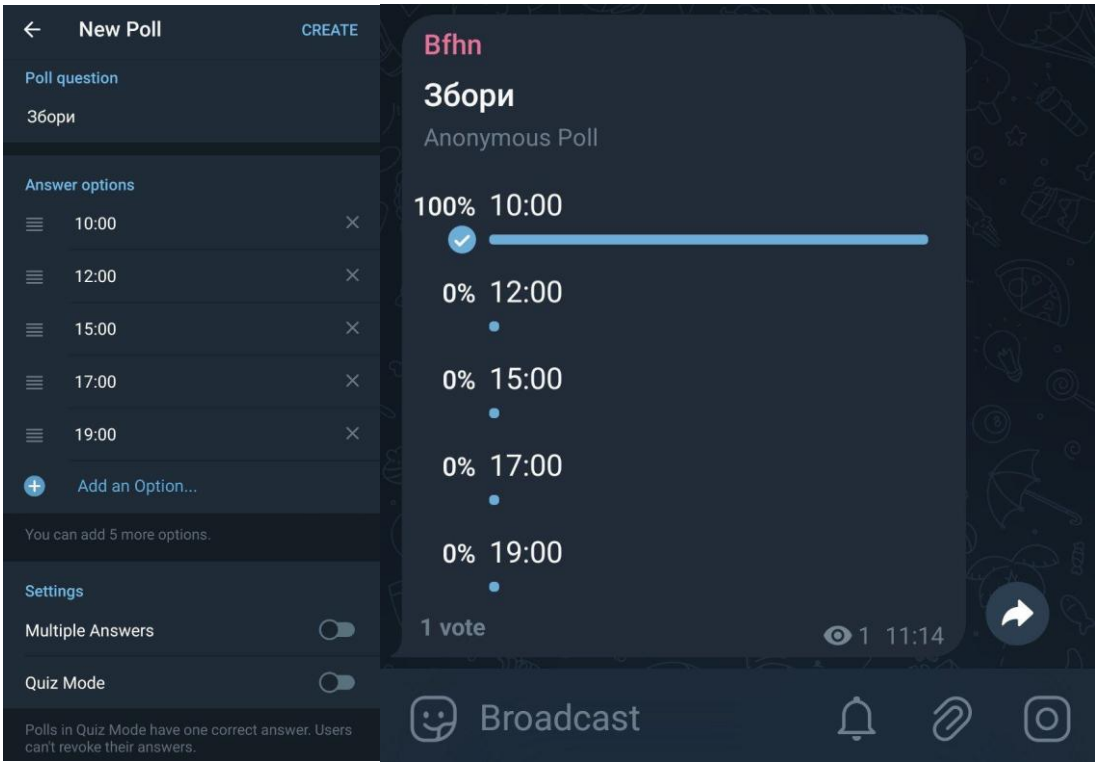

Рисунок 1.7 – Інтерфейс функціоналу голосування в месенджері Telegram

Проведений аналіз дозволяє зробити висновок, що жоден з наявних сервісів не поєднує в собі повноцінний функціонал спеціалізованих платформ для голосування з одночасним збереженням простоти та мобільності. Google Forms, Typeform і SurveyMonkey надають розширені можливості, проте не завжди зручні на мобільних пристроях. У свою чергу, Дія та Telegram мають інтуїтивно зрозумілий інтерфейс, але обмежені у функціональності.

Таким чином, актуальною є розробка нового мобільного додатку, що об'єднає найкращі риси вищезгаданих платформ: простоту створення голосувань та зрозумілий інтерфейс. Такий додаток стане ефективним інструментом для організації онлайн-голосувань у найрізноманітніших сферах життя.

1.3 Обгрунтування вибору програмного забезпечення

На першому етапі розробки основною мовою програмування була обрана Java – одна з найпопулярніших об’єктно-орієнтованих мов у світі. Вона має десятиліття історії, активну спільноту та величезну кількість напрацьованих бібліотек і фреймворків. Серед ключових переваг Java можна виокремити наступні:

- Платформонезалежність – код, написаний на Java, компілюється у спеціальний байт-код, який виконується на віртуальній машині JVM. Завдяки цьому одна й та сама програма може працювати на різних операційних системах без потреби у зміні коду.

- Масштабованість і стабільність – Java широко використовується у великих проєктах, де важлива надійність і чітка структура коду.

- Підтримка Android – Java була основною мовою розробки для Android з моменту створення платформи, тому має глибоку інтеграцію з Android SDK, а також підтримується всіма інструментами розробника, такими як Android Studio.

У контексті даної дипломної роботи, де йдеться про розробку мобільного додатку для Android, Java стала логічним вибором через перевірену часом

інфраструктуру та гнучкість у реалізації функціональних компонентів застосунку.

У ході подальшої розробки виникла потреба в сучаснішому, більш лаконічному та гнучкому рішенні. Тому до процесу розробки було залучено Kotlin – сучасну статично типізовану мову програмування, яка з 2017 року визнана офіційною мовою розробки для Android.

Основні переваги Kotlin, які обумовили його використання:

- Сумісність з Java – Kotlin також працює на JVM, що дозволяє безперешкодно використовувати вже написаний на Java код [8]. Це особливо важливо у великих проєктах або при поступовому переході з Java на Kotlin.

- Скорочення обсягу коду – завдяки лаконічному синтаксису, кількість рядків коду у Kotlin значно менша, ніж у Java [8]. Це зменшує складність коду, підвищує його читабельність та пришвидшує процес розробки.

- Null-безпека – однією з головних переваг Kotlin є вбудована система захисту від помилок, пов’язаних з null-значеннями (NullPointerException), яка є однією з найбільш поширених проблем у Java [8].

- Функціональне програмування – Kotlin підтримує сучасні підходи до розробки, такі як використання лямбда-виразів, функцій вищого порядку, операторів map, filter, reduce тощо [8]. Це дозволяє створювати більш модульний та адаптивний код.

- Офіційна підтримка Android SDK – Android Studio та супутні інструменти розробки вже містять повну підтримку Kotlin, зокрема автодоповнення, дебагінг, генерацію коду та шаблони [8].

Таким чином, для створення ефективного, безпечного та зручного мобільного додатку доцільно було використати комбінацію двох мов — Java як стабільну та перевірену платформу з багатим досвідом у мобільній розробці, та Kotlin як сучасне рішення, що забезпечує високу швидкість розробки, гнучкість і безпечність коду.

1.4 Постановка задачі

У результаті проведеного аналізу деяких онлайн сервісів було визначено їх основні переваги та недоліки, у результаті чого було сформовано основні вимоги для удосконалення додатку для голосування.

Мета роботи полягає у створенні простого та зручного онлайн додатку для голосування на операційній системі Android. Для цього потрібно вирішити такі задачі:

- описати логічну модель бази даних, її таблиць та зв’язків;
- відтворити структуру бази даних;
- зробити макет графічного дизайну додатку;
- відтворити та забезпечити коректну роботу інтерфейсу додатку.

У підсумку, результатом виконання даної роботи має стати мобільний додаток, який дозволить організовувати онлайн-голосування у зручній формі, доступній для широкого кола користувачів. Його реалізація має поєднувати зручність, мінімалізм інтерфейсу та функціональну ефективність на основі сучасних технологій мобільної розробки.

2 РОЗРОБКА СТРУКТУРИ ДОДАТКУ ТА БАЗИ ДАНИХ

2.1 Структура додатку

Архітектура є фундаментом усього програмного забезпечення, адже від її якості залежить не лише технічна надійність, але й масштабованість, адаптивність та зручність подальшої підтримки і розвитку додатку.

У цьому контексті розробники зазвичай звертаються до шаблонів архітектури. Найбільш популярними серед них є MVC (Model-View-Controller), MVP (Model-View-Presenter) та MVVM (Model-View-ViewModel). Кожен з них має свої особливості й переваги. У випадку з Android-додатками особливо популярним є шаблон MVVM, оскільки він найбільше відповідає специфіці Android-розробки та сумісний із сучасними бібліотеками, такими як Jetpack Compose, LiveData, ViewModel тощо.

MVVM дозволяє досягти чіткого розмежування між виглядом інтерфейсу (View), логікою взаємодії (ViewModel) та обробкою даних (Model). Завдяки цьому компоненти додатку можна розробляти, тестувати та змінювати незалежно один від одного. Наприклад, зміна способу зберігання даних не вимагає змін у відображенні інтерфейсу.

Крім того, добре продумана архітектура мобільного додатку дозволяє легко масштабувати систему. Якщо згодом потрібно буде додати нові функції, нові типи користувачів або змінити логіку взаємодії з сервером — усе це можна реалізувати із мінімальними зусиллями, не порушуючи цілісності коду. Таким чином, архітектура є не просто технічним планом, а стратегічною основою всього життєвого циклу програмного продукту.

На практиці архітектура мобільного додатку виявляється також у структурі файлів, іменуванні класів, розміщенні компонентів у директоріях, розмежуванні UI і логіки тощо. Окрім основного поділу на View, ViewModel і Model, застосовуються також допоміжні рівні, такі як Repository (для інкапсуляції

доступу до даних), Service (для взаємодії з API), Util (для винесення загальнокорисних функцій) тощо.

Саме ця багаторівнева побудова – від загального до конкретного – забезпечує прозорість, контрольованість і передбачуваність поведінки додатку. Вона дозволяє кожному члену команди швидко зорієнтуватися у проєкті, а новим учасникам – легко включатися в процес розробки.

Таким чином, грамотне проєктування архітектури мобільного додатку – це не просто обов'язкова частина процесу, а необхідна умова для створення якісного, стабільного та конкурентоспроможного програмного продукту. У цьому світлі шаблони MVVM, а також принципи SOLID, KISS, DRY, Clean Architecture тощо набувають не абстрактного, а практичного значення, і повинні бути невід'ємною частиною будь-якого серйозного проєкту.

При вході в додаток користувач потрапляє на головну сторінку програми, з якої може перейти до сторінки авторизації або до сторінки реєстрації. При авторизації вказується електронна пошта та пароль. При реєстрації потрібно спочатку вибрати роль (Адміністратор або Учасник голосування), після чого ввести електронну пошту та пароль для реєстрації.

На рисунку 2.1 зображена UML use case діаграма вище згаданих дій.

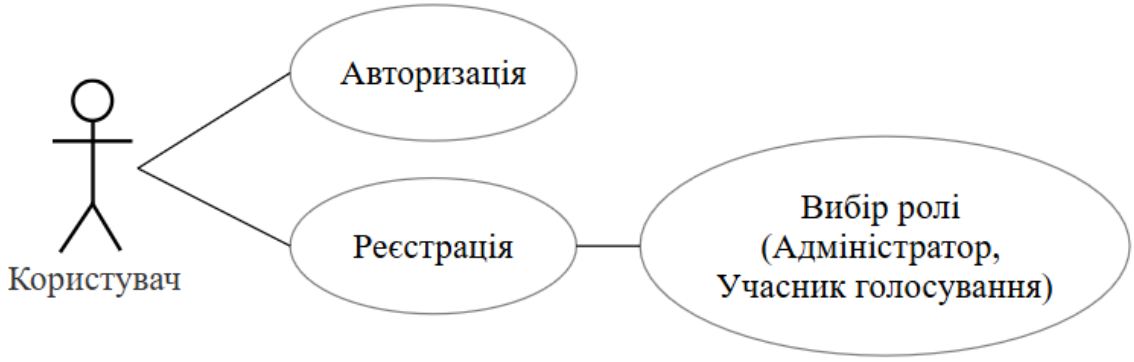

Рисунок 2.1 – UML use case діаграма входу користувача в додаток

Після авторизації в залежності від ролі користувача з'являється або сторінка Адміністратора або Учасника голосування.

Зі сторінки Учасника голосування відображаються всі активні голосування призначені цьому користувачу. Користувач може проголосувати.

На рисунку 2.2 зображена UML use case діаграма сторінки Учасника голосування.

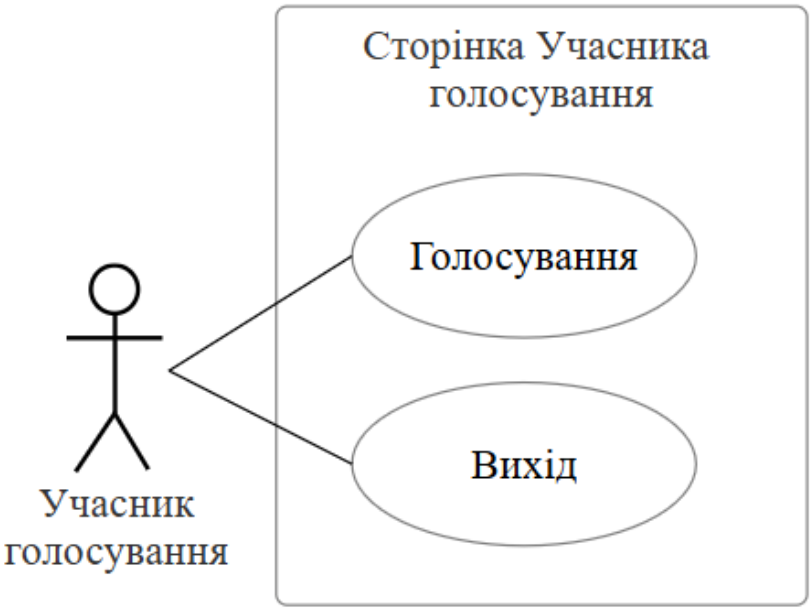

Рисунок 2.2 – UML use case діаграма сторінки Учасника голосування

Зі сторінки Адміністратора відображаються створені ним активні та неактивні голосування. Користувач може створити голосування, змінити неактивне голосування, активувати нове голосування, закрити активне голосування, видалити голосування та створити нове голосування.

При створені та редагуванні голосування відкривається вікно, де Адміністратор повинен ввести або змінити назву голосування, учасників голосування, самі запитання та відповіді.

На рисунку 2.3 зображена UML use case діаграма сторінки Адміністратора.

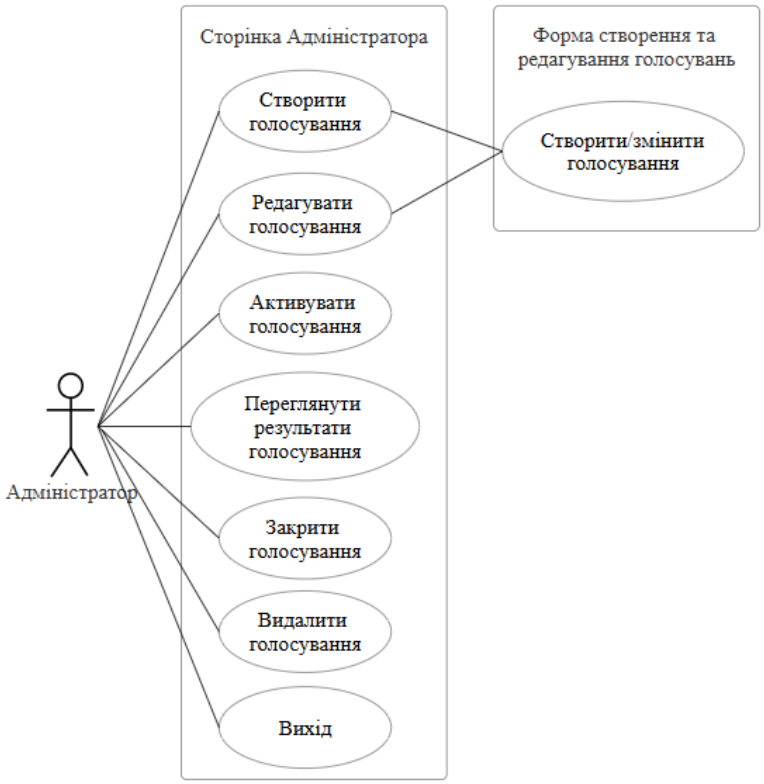

Рисунок 2.3 – UML use case діаграма сторінки Адміністратора

Рисунок 2.4 – UML діаграма активностей

Кожне голосування має свій шлях. На рисунку 2.4 зображена UML діаграма активностей. Ця діаграма активностей описує типовий сценарій взаємодії Учасника голосування з інтерфейсом електронної системи голосування. Основна мета цього процесу – надати користувачу змогу переглянути питання голосування, заповнити форму відповіді та успішно надіслати свій голос. Діаграма побудована у вигляді послідовного потоку дій, який супроводжується умовною логікою перевірки правильності заповнення форми.

Файли у проекті поділені на Activities, Composables, Models, Services, Utils.

У папці Activities знаходяться дії з додатком:

- MainActivity для керування навігацією, входом в додаток і основними функціями додатку;
- LoginActivity для керування авторизацією;
- RegisterActivity для керування реєстрацією нових користувачів;
- AdminDashboardActivity для керування інтерфейсом Адміністратора;
- VoterDashboardActivity для керування інтерфейсом Учасника голосування;
- VoteDetailActivity для керування безпосередньо самими голосуваннями, їх питаннями та відповідями.

У папці Composables знаходяться інтерфейси:

- WelcomeScreen – інтерфейс головного екрану;
- LoginScreen – інтерфейс екрану авторизації;
- RegisterScreen – інтерфейс екрану реєстрації;
- AdminDashboard – інтерфейс Адміністратора;
- VoterDashboard – інтерфейс Учасника голосування.
- VoteDetail – інтерфейс голосувань.

У папці Models знаходяться моделі:

- User – модель користувача;
- Vote – модель голосування;
- Question – модель запитання;

- Answer – модель відповіді.

У папці Services знаходяться сервіси:

- FirebaseService для підключення та роботи бази даних;
- AuthService для роботи з аутентифікацією;
- VoteService для роботи з голосуваннями.

У папці utils знаходяться службові файли:

- Constants – константи;
- Extensions – розширення;
- Theme – UI теми, кольори, компоненти стилю тощо.

На рисунку 2.5 зображена UML діаграма класів об’єктів моделей.

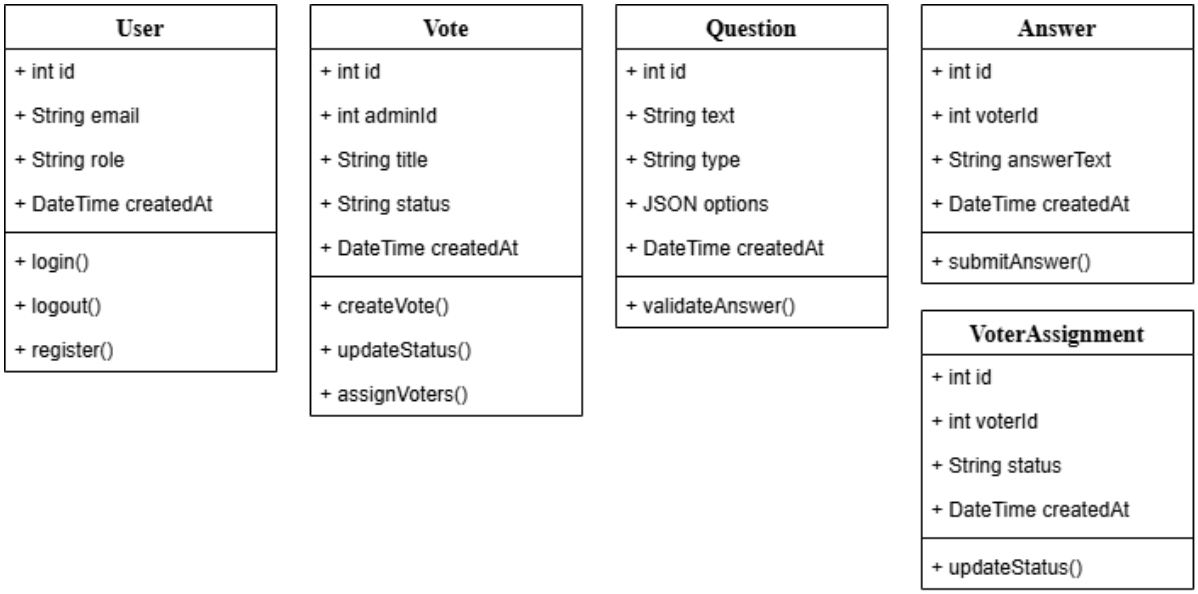

Рисунок 2.5 – UML діаграма класів об’єктів моделей

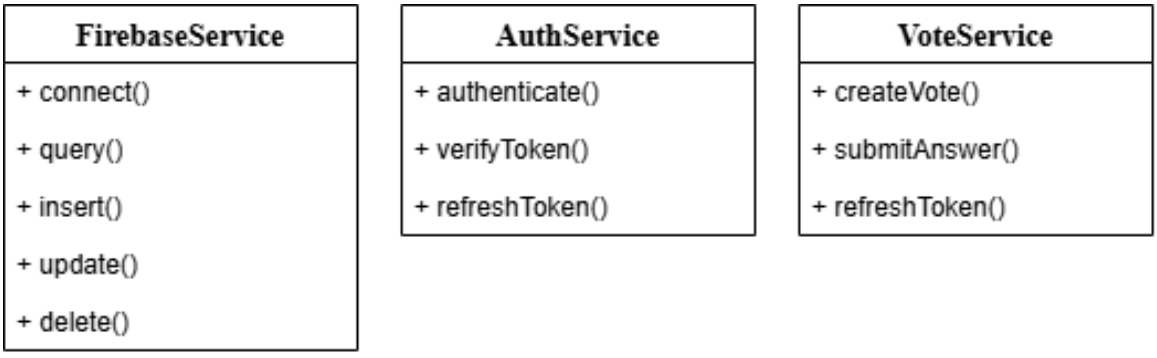

Рисунок 2.6 – UML діаграма класів сервісів

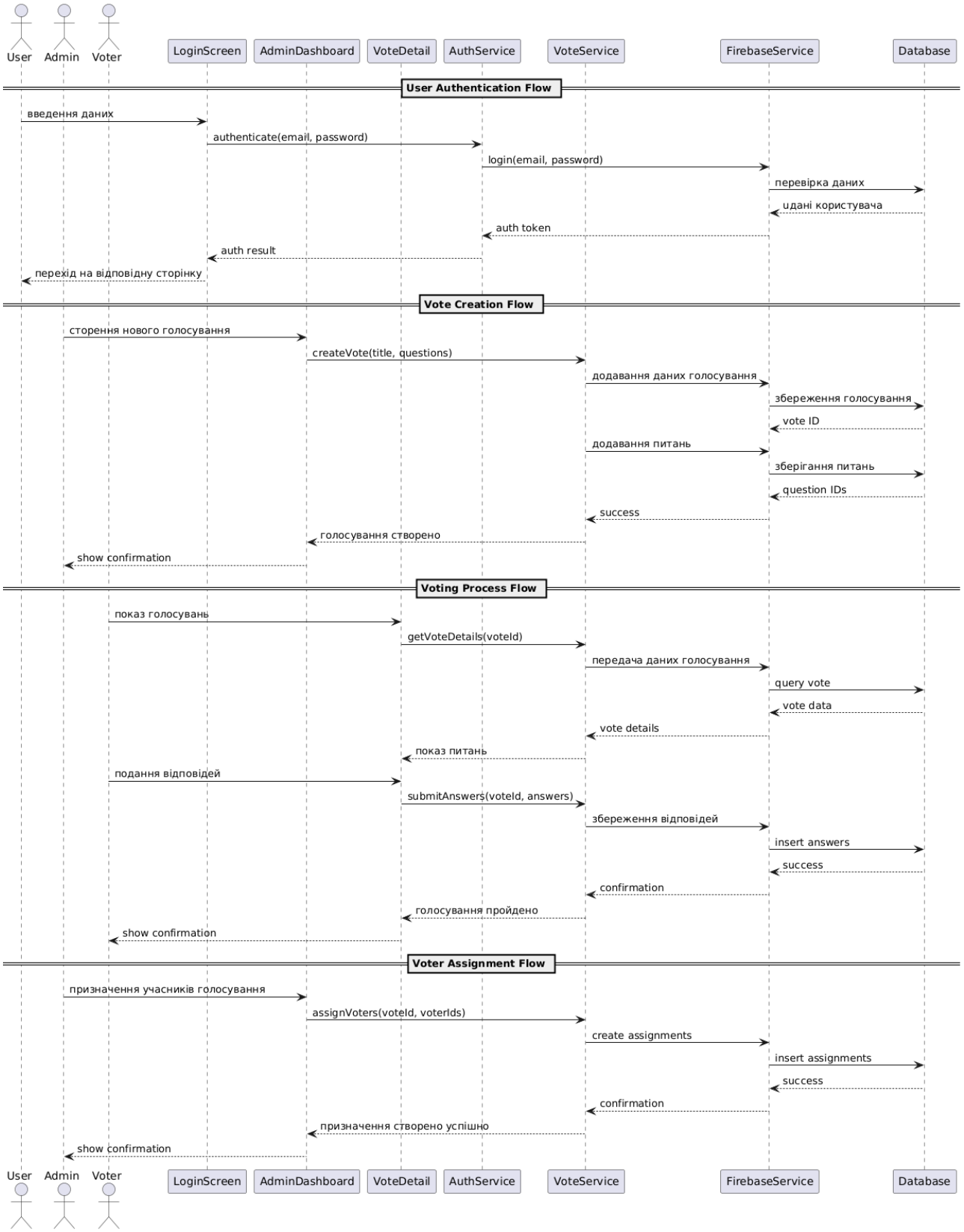

Рисунок 2.7 – UML діаграма послідовностей

На рисунку 2.6 зображена UML діаграма класів сервісів.

На рисунку 2.7 зображена UML діаграма послідовностей.

Представлена діаграма послідовностей дозволяє чітко прослідкувати всю логіку взаємодії між користувачами, інтерфейсом, сервісними модулями та базою даних. У кожному сценарії видно, як дані передаються між компонентами, і як організована архітектура клієнт-сервер із використанням Firebase як бекенд сервісу. Такий підхід забезпечує модульність, зручність масштабування та зрозумілість кожного процесу.

Першим і найважливішим етапом взаємодії користувача із системою є процес аутентифікації. Саме з нього починається робота будь-якого користувача, незалежно від його ролі – чи то звичайний виборець, чи адміністратор. Аутентифікація слугує вхідною точкою, яка забезпечує контроль доступу до функціоналу системи.

Процес стартує з того моменту, коли користувач відкриває інтерфейс екрана входу (LoginScreen). Через зручну та інтуїтивно зрозумілу форму користувач вводить свої облікові дані, зазвичай це електронна пошта та пароль. Ці дані передаються у систему через інтерфейс, який виконує роль проміжної ланки між користувачем і логікою додатку.

Далі введена інформація надсилається до спеціального модуля – AuthService. Цей модуль відіграє роль логічного контролера, який відповідає за перевірку автентичності користувача. AuthService обробляє вхідні дані та формує запит до FirebaseService – спеціального проміжного шару, який реалізує зв’язок із базою даних.

FirebaseService надсилає запит до хмарної бази даних (Database). На цьому етапі система перевіряє, чи існує вказаний користувач у системі, і чи збігається пароль з тим, що збережений у базі. У випадку позитивного результату – база даних повертає необхідну інформацію про користувача. Цей процес зазвичай виконується у декілька мілісекунд, але є критично важливим для безпеки всієї платформи.

Після цього Firebase генерує токен авторизації, який підтверджує успішну ідентифікацію користувача. Токен передається назад до AuthService, який, у свою чергу, надсилає остаточний результат назад до інтерфейсу. У разі успішної аутентифікації користувач автоматично перенаправляється на головну панель керування (Dashboard), де й починається основна робота в системі.

Функціонал створення голосування є доступним лише адміністратору. Цей процес розпочинається в інтерфейсі AdminDashboard, де адміністратор заповнює спеціальну форму створення нового голосування. У цій формі вказуються основні параметри, такі як назва голосування, дата проведення, опис тощо.

Після заповнення форми ці дані надсилаються до VoteService – логічного модуля, що відповідає за керування голосуваннями. VoteService обробляє отриману інформацію та ініціює звернення до FirebaseService, який, у свою чергу, формує запит до бази даних для збереження нового запису.

Після збереження інформації база повертає у відповідь унікальний ідентифікатор голосування (vote ID). Цей ідентифікатор буде використовуватись у подальшій роботі для призначення питань, відповідей та користувачів.

Далі VoteService додає конкретні питання до голосування. Для цього він повторно звертається до FirebaseService, який знову виконує операції над базою даних. Після успішного збереження кожного питання база повертає відповідні ідентифікатори питань, і VoteService отримує підтвердження завершення операції.

У результаті інтерфейс адміністратора отримує відповідь про успішне створення голосування, а на екрані з’являється повідомлення про те, що голосування створено успішно та готове до подальших дій.

Після створення голосування настає час його проведення. Участь у голосуванні починається тоді, коли виборець відкриває інтерфейс голосування (VoteDetail). Цей інтерфейс автоматично надсилає запит до VoteService із метою отримати повну інформацію про доступне голосування.

VoteService, як і в попередніх етапах, звертається до FirebaseService, а той – до бази даних, щоб отримати актуальні дані: список питань, дату завершення,

опис тощо. Після отримання даних, відповідь передається назад через усі рівні до VoteDetail, де виборець бачить повний зміст голосування.

Після ознайомлення та заповнення відповідей виборець натискає кнопку «Надіслати», і система передає дані назад до VoteService. Той, у свою чергу, відправляє інформацію до FirebaseService для збереження у базі. Після успішного збереження користувач отримує підтвердження про участь у голосуванні, що завершує цей етап.

Окремою важливою функцією адміністратора є призначення виборців до певного голосування. У відповідному інтерфейсі адміністратор обирає голосування, до якого потрібно надати доступ, і вказує список ідентифікаторів виборців, яким це голосування буде доступне.

Після заповнення форми інтерфейс надсилає ці дані до VoteService. Цей сервіс обробляє запит та формує спеціальні записи призначень, які потім передаються через FirebaseService у базу даних для збереження.

Після збереження інформації база повертає підтвердження, яке передається назад на інтерфейс адміністратора.

Таким чином, без необхідності ручного втручання в базу даних, адміністратор отримує змогу легко та швидко керувати доступом до голосувань, що робить процес більш гнучким, безпечним і масштабованим.

2.2 Структура бази даних

У рамках реалізації даного програмного продукту було прийнято рішення використовувати Firebase як основну платформу для створення та обслуговування бази даних. Це обґрунтоване вибором інструменту, що ідеально підходить для мобільних застосунків, завдяки простоті інтеграції, широкому функціоналу та високій надійності.

Firebase надає розробнику готовий бекенд, який не потребує окремого налаштування серверної інфраструктури, що особливо важливо для невеликих команд, мобільних додатків або MVP-версій. Завдяки цьому можна суттєво

скоротити час розробки та зосередитися безпосередньо на реалізації логіки додатку.

Cloud Firestore – основа для зберігання даних

Однією з ключових технологій у складі Firebase є Cloud Firestore – хмарна документоорієнтована NoSQL-база даних нового покоління. Вона підтримує зберігання даних у вигляді колекцій і документів, що дозволяє більш гнучко та масштабовано підходити до організації інформації порівняно з традиційними SQL-рішеннями.

Firestore дозволяє зручно зберігати, оновлювати та отримувати дані з можливістю синхронізації в реальному часі, що є критично важливим у застосунках, пов’язаних із взаємодією багатьох користувачів — наприклад, при голосуванні, коли потрібно негайно відображати оновлення або результати.

На відміну від класичних реляційних баз даних, Firestore не вимагає жорстко визначеної структури таблиць та складних SQL-запитів. Проте, навіть у NoSQL-середовищі важливо дотримуватися логічної структури, яка відповідає потребам системи, забезпечує зв’язки між сутностями та дозволяє ефективно масштабувати рішення в майбутньому.

Зв’язки між об’єктами реалізуються через вкладені документи або посилання між колекціями. Обробка таких зв’язків виконується на рівні логіки програми – через сервіси, які реалізують потрібні запити та збирання даних.

База даних буде складатися з 5 таблиць.

Для початку в нас буде таблиця користувачів із їхніми ідентифікаторами, логінами (електронними адресами) і ролями (Адміністратор або Учасник голосування).

Таблиця 2.1 – Таблиця users

Ключове поле	NOT NULL	Назва	Тип даних	Вміст
+	+	id	int	Ідентифікатор користувача
	+	email	string	Електронна адреса користувача
	+	role	string	Роль користувача (Адміністратор, Учасник голосування)

Далі потрібно створити таблицю голосувань з їхніми ідентифікаторами, ідентифікаторами Адміністратора, який створив голосування, назвою та статусом голосування і електронними адресами учасників голосування.

У самих голосуваннях будуть питання. Для них також потрібна своя таблиця.

Там зберігатиметься ідентифікатор питання, ідентифікатор голосування якому це питання належить, тип питання (відкрите або з відповідями), назва питання, тобто саме питанням і при типі запитання з відповідями – опції відповідей.

Таблиця 2.2 – Таблиця votes

Ключове поле	NOT NULL	Назва	Тип даних	Вміст
+	+	id	int	Ідентифікатор голосування
	+	adminId	int	Ідентифікатор Адміністратор
	+	title	string	Назва голосування
	+	status	string	Статус голосування (чернетка, активне, закрите)
	+	voterEmails	array	Електронні адреси учасників голосування
	+	createdAt	timestamp	Позначка часу створення голосування
	+	updatedAt	timestamp	Позначка часу редагування голосування

Таблиця 2.3 – Таблиця questions

Ключове поле	NOT NULL	Назва	Тип даних	Вміст
+	+	id	int	Ідентифікатор питання
	+	voteId	voteId	Ідентифікатор голосування
	+	type	string	Тип питання
	+	text	string	Назва питання
		options	array	Опції відповідей
	+	createdAt	timestamp	Позначка часу створення питання

Також повинна бути таблиця відповідей з ідентифікаторами самих відповідей, голосування та учасників голосування яким належать конкретні відповіді та сам текст відповіді.

Таблиця 2.4 – Таблиця answers

Ключове поле	NOT NULL	Назва	Тип даних	Вміст
+	+	id	int	Ідентифікатор відповіді
	+	voteId	int	Ідентифікатор голосування
	+	voterId	int	Ідентифікатор Учасника голосування
	+	answerText	string	Текст відповіді
	+	createdAt	timestamp	Позначка часу створення відповіді

Також потрібна службова таблиця для приналежності Учасника голосування до самого голосування. Там мають бути поля ідентифікаторів приналежності, голосування та учасника голосування та статус виконання голосування учасником.

Таблиця 2.5 – Таблиця voterAssignments

Ключове поле	NOT NULL	Назва	Тип даних	Вміст
+	+	id	int	Ідентифікатор належності голосування
	+	voteId	int	Ідентифікатор голосування
	+	voterId	int	Ідентифікатор Учасника голосування
	+	status	string	Статус голосування (очікується, проголосовано)
	+	createdAt	timestamp	Позначка часу

Структура бази даних зображена на рисунку 2.8.

У Firebase існує окремий інструмент для керування аутентифікацією користувачів, який доступний через меню Authentication у консолі Firebase (рис. 2.8).

Цей інструмент є важливою частиною платформи, оскільки забезпечує безпечний доступ користувачів до додатку та дозволяє адмініструвати облікові записи без написання додаткового серверного коду.

Там можна переглянути всіх користувачів, їхні ідентифікатори, дату реєстрації та дату останнього входу. Звідси зразу є можливість створити або видалити користувача. Також є можливість переглянути статистику входів у додаток.

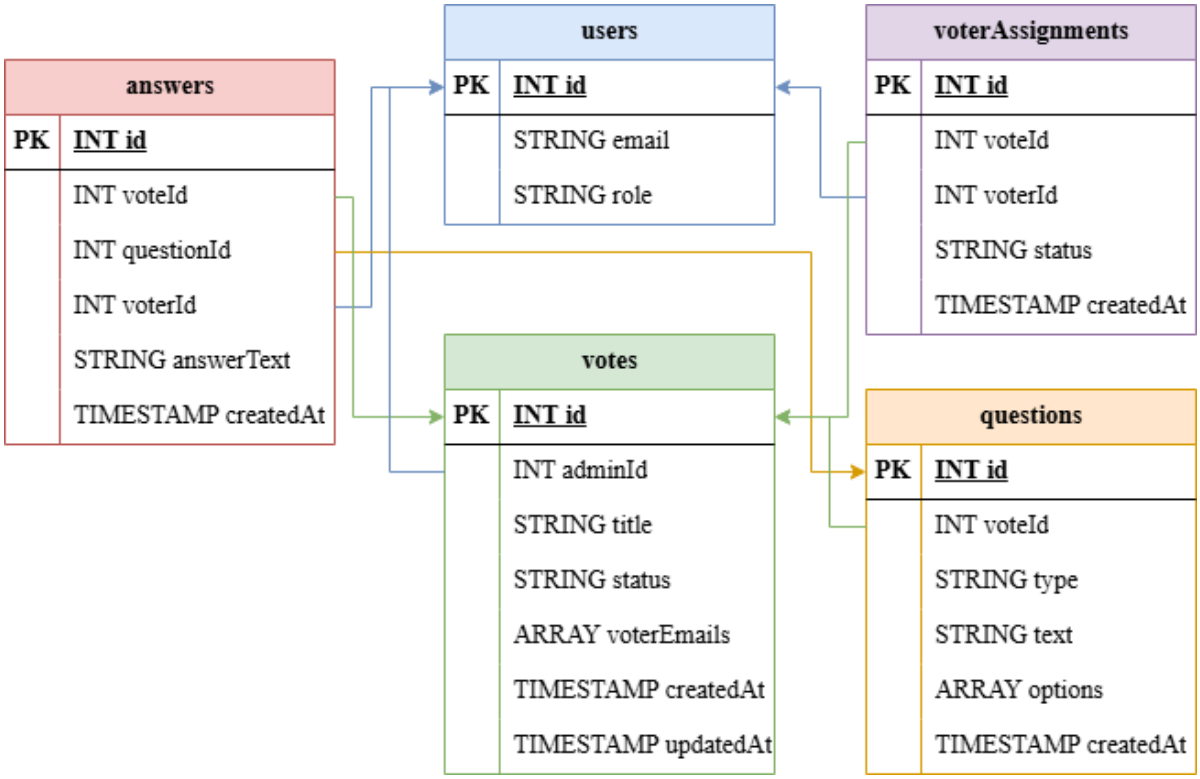

Рисунок 2.8 – Структура бази даних

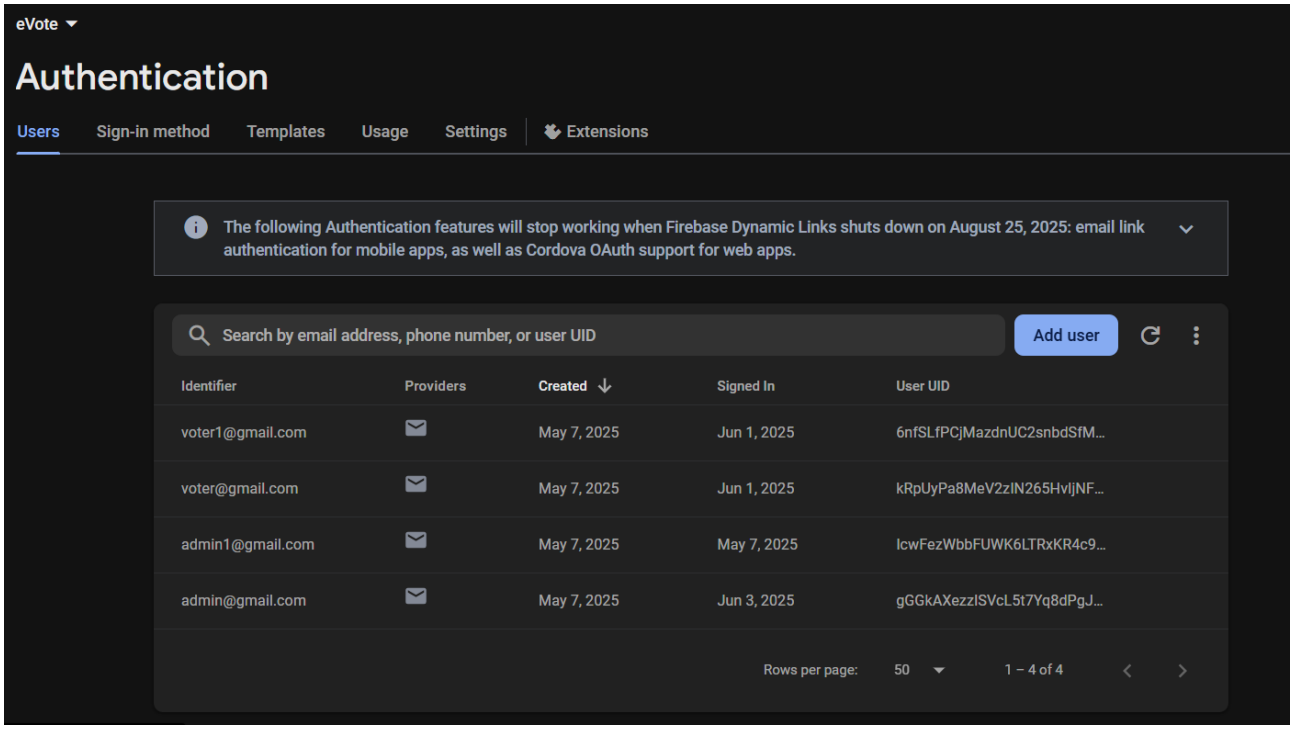

Рисунок 2.8 – Меню Authentication у консолі Firebase

Використання цього модуля в межах проєкту дозволяє значно спростити процес реєстрації, входу та управління користувачами, зберігаючи при цьому високий рівень безпеки та зручності як для користувача, так і для розробника.

Висновок до розділу

Основна увага приділена створенню логічної моделі бази даних, яка включає п'ять основних таблиць: `users`, `votes`, `questions`, `answers` та `voterAssignments`. Ці таблиці забезпечують зберігання всієї необхідної інформації про користувачів, голосування, питання, відповіді та зв’язки між ними. Використання хмарної бази даних Firebase дозволило забезпечити швидкий доступ до даних, їх надійне зберігання та можливість синхронізації в реальному часі.

Архітектура додатку була побудована з урахуванням сучасних підходів до розробки мобільних застосунків. Для реалізації було обрано шаблон MVVM (Model-View-ViewModel), який забезпечує чітке розділення логіки додатку, інтерфейсу користувача та обробки даних. Це дозволило створити модульну систему, яка легко піддається тестуванню, масштабуванню та подальшому розвитку. Використання мов програмування Java та Kotlin сприяло створенню ефективного та безпечного коду, а також забезпечило сумісність з різними версіями Android.

У розділі було наведено UML-діаграми, які візуалізують структуру додатку, взаємодію між компонентами та логіку роботи системи. Ці діаграми допомогли краще зрозуміти архітектуру проекту та взаємозв’язки між його елементами.

Реалізована структура бази даних та архітектура додатку забезпечують стабільну роботу системи та можливість її подальшого вдосконалення.

3 РЕАЛІЗАЦІЯ ГРАФІЧНОГО ІНТЕРФЕЙСУ ТА ОГЛЯД ДОДАТКУ

3.1 Огляд проекту

Додаток eVote призначений для онлайн голосувань, головна мета якого є простота та зрозумілість для користувача.

У додатку використовуються нові технології та дизайн. Зокрема додаток виконано у контексті Material Design, тобто згідно рекомендацій розробки графічного інтерфейсу від Google [9].

Однією з особливостей дизайну є підтримка динамічної теми – додаток автоматично адаптується до світлого або темного режиму, встановленого на пристрої користувача.

Крім того, всі екрани забезпечені плавними переходами та анімаційною реакцією на натискання кнопок, що підвищує зручність та приємність користування додатком.

Додаток написаний за допомогою мов програмування Java і Kotlin, які є найпопулярнішими та найсучаснішими для написання додатків на Android.

Однією з ключових складових будь-якого мобільного додатку є інтерфейс користувача. Саме інтерфейс визначає, наскільки легко та приємно користувачу взаємодіяти з додатком. Хороший інтерфейс повинен бути не лише естетично привабливим, а й інтуїтивно зрозумілим, адаптивним та функціонально логічним. У випадку додатку для онлайн-голосування особливо важливою є простота та зручність використання, адже він має бути доступним для широкого кола користувачів різного віку та рівня технічної підготовки.

3.2 Головний екран

На екрані входу користувача розміщено кілька ключових елементів, що створюють перше враження про додаток і сприяють зручній навігації. Зокрема, тут знаходиться логотип додатку, який відображає бренд і допомагає

користувачам ідентифікувати програму. Поруч із логотипом розміщено привітальне повідомлення, яке налаштовує користувача на позитивний досвід взаємодії та задає дружній тон спілкування.

Для подальшої роботи з додатком користувачу пропонуються дві основні кнопки – «Вхід» та «Реєстрація». Ця простота вибору дозволяє швидко зорієнтуватися, чи є у користувача вже обліковий запис, чи необхідно створити новий. Такий підхід мінімізує час на пошук потрібної дії та робить процес початку роботи максимально інтуїтивним.

Інтерфейс додатку повністю адаптований під поточні системні налаштування операційної системи Android, що є важливою особливістю з точки зору користувацького досвіду (UX). Якщо на пристрої активовано темну тему оформлення, додаток автоматично переключається у відповідну темну кольорову схему. У разі використання світлої теми система відповідно налаштовує зовнішній вигляд додатку, що дозволяє зберігати єдиний стиль інтерфейсу та підвищує комфорт при використанні.

Перемикання між світлою і темною темами здійснюється за допомогою спеціальної функції EVoteTheme, яка перевіряє поточні налаштування системи і автоматично застосовує відповідну кольорову палітру в додатку.

Кольори, які використовуються у програмі, записані в окремий файл Color.kt:

```
val PrimaryBlue = Color(0xFF2196F3)
val SecondaryBlue = Color(0xFF1976D2)
val AccentGreen = Color(0xFF4CAF50)
val ErrorRed = Color(0xFFE53935)
val BackgroundLight = Color(0xFFF5F5F5)
val BackgroundDark = Color(0xFF121212)
val TextPrimary = Color(0xFF212121)
val TextSecondary = Color(0xFF757575)
```

На рисунку 3.1 зображено вигляд головної сторінки у темній та світлій темах.

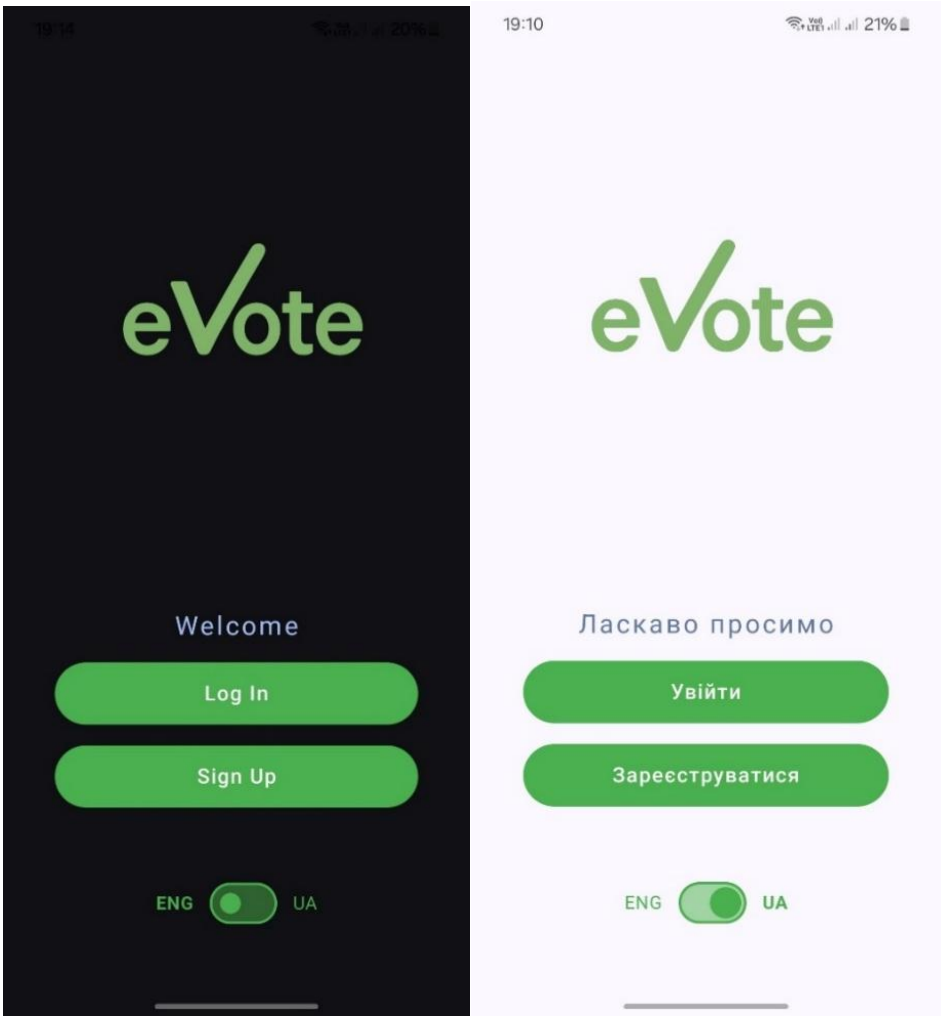

Рисунок 3.1 – Головна сторінка в темній та світлій темах

3.3 Реєстрація

Для реєстрації спершу потрібно обрати роль користувача: Адміністратор або Учасник голосування. На рисунку 3.2 зображено сторінку обирання ролі.

Далі, залежно від обраної ролі користувача відбувається процедура реєстрації нового користувача у системі. Для цього передбачено спеціальний інтерфейс, що містить три обов’язкових поля для введення інформації: поле для логіну, поле для паролю, а також поле для підтвердження паролю, яке забезпечує правильність введення та зменшує ризик помилок при реєстрації.

Цей підхід спрямований на підвищення безпеки облікового запису та мінімізацію ймовірності введення некоректних даних. Валідація паролів включає перевірку на збіг основного паролю і підтвердження, що допомагає уникнути випадкових помилок.

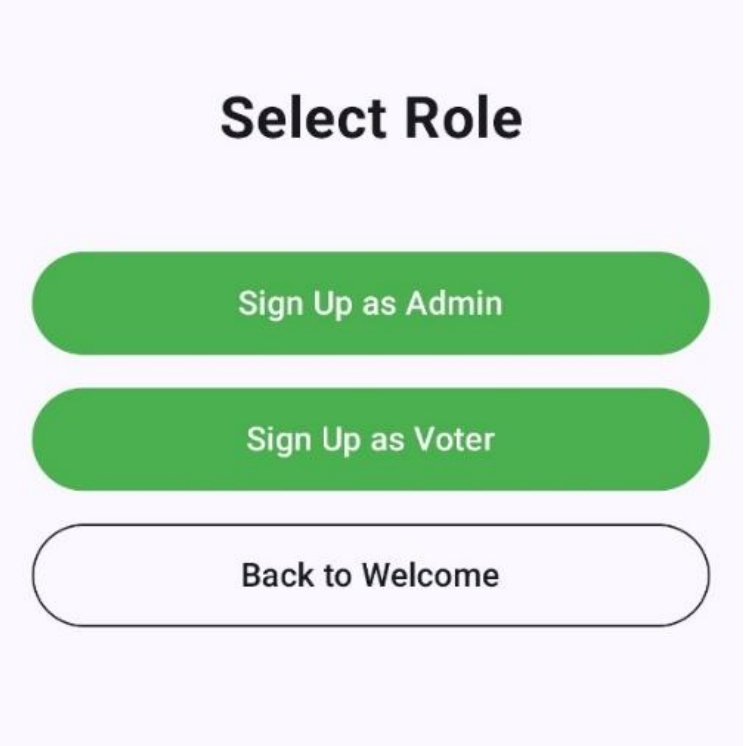

Рисунок 3.2 – Сторінка обирання ролі

Технічно процес реєстрації реалізовано у спеціальному класі AuthRepository.kt, де міститься метод signUp. Цей метод відповідає за створення нового користувача у базі даних. Він приймає введені користувачем дані, виконує необхідну валідацію і взаємодіє з базою даних для збереження облікової інформації:

```
    suspend fun signUp(email: String, password: String, isAdmin:
Boolean): Result<FirebaseUser> {
            return withContext(Dispatchers.IO) {
                try {
                    val authResult =
auth.createUserWithEmailAndPassword(email, password).await()
                    val user = authResult.user ?: throw
Exception("User creation failed")
    firestore.collection("users").document(user.uid)
                        .set(mapOf(
                            "email" to email,
                            "role" to if (isAdmin) "admin" else
"voter"
```

```
                )).await()
            Result.success(user)
        } catch (e: Exception) {
            Result.failure(e)
        }
    }
```

На рисунку 3.3 зображено сторінку реєстрації нового користувача.

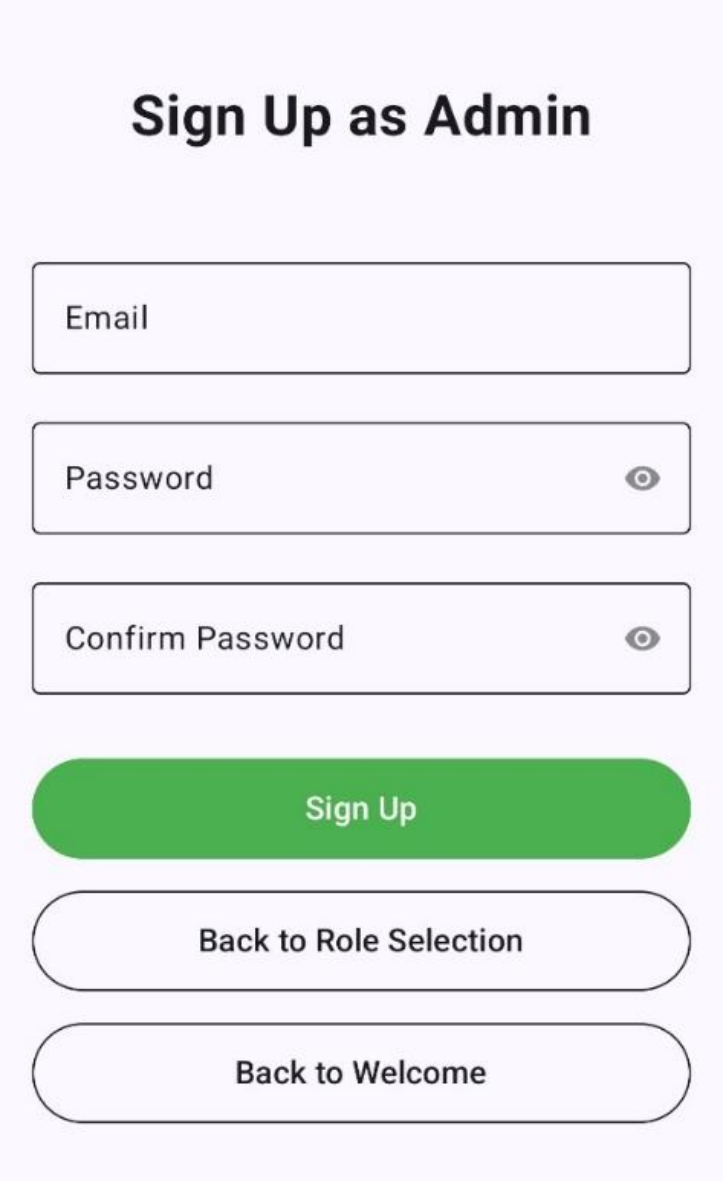

Рисунок 3.3 – Форма реєстрації

3.4 Авторизація

На сторінці авторизації, що зображена на рисунку 3.4, є два поля: логін (електронна адреса) та пароль.

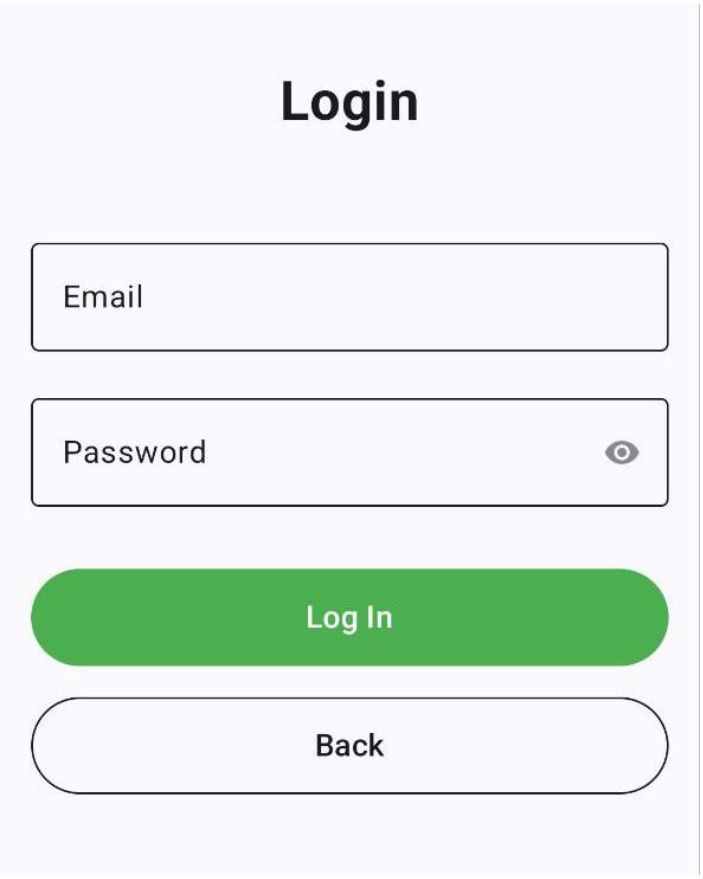

Рисунок 3.4 – Форма авторизації

Після того, як користувач вводить свої облікові дані – логін та пароль – він активує процес входу, натискаючи відповідну кнопку на екрані авторизації.

Система проводить порівняння введених користувачем значень з тими, що збережені у базі даних. В разі, якщо логін і пароль збігаються із зареєстрованими в системі даними, відбувається успішна авторизація. Це означає, що користувач підтверджує свою особу, і система дозволяє йому доступ до внутрішніх ресурсів додатку.

Після успішної авторизації користувач автоматично перенаправляється на сторінку свого профілю або на основний інтерфейс додатку. Важливо, що вибір

цієї сторінки залежить від ролі користувача, яка була визначена під час його реєстрації. Так, наприклад, адміністратор отримує доступ до спеціальних функцій керування системою, тоді як звичайний учасник голосування потрапляє на сторінку, де може брати участь у голосуваннях.

У разі, якщо логін або пароль були введені некоректно, тобто не збігаються з даними, збереженими у базі – система миттєво відобразить користувачу повідомлення про помилку. Це повідомлення інформує про те, що введені дані є неправильними, і пропонує повторити спробу входу. Такий механізм дозволяє підвищити безпеку системи та запобігає несанкціонованому доступу.

Загалом, цей процес реалізує зручний, інтуїтивно зрозумілий та безпечний механізм аутентифікації, який гарантує, що лише авторизовані користувачі отримують доступ до відповідних розділів додатку відповідно до своїх прав і ролей.

3.5 Інтерфейс Учасника голосування

Інтерфейс користувача, призначений для Учасника голосування, створений з особливим акцентом на зручність, простоту та інтуїтивну зрозумілість. Головною метою при його розробці було забезпечити максимально комфортну взаємодію, щоб користувач міг швидко орієнтуватися в застосунку, легко знаходити доступні голосування та без зайвих зусиль брати в них участь.

Завдяки сучасним принципам дизайну, інтерфейс динамічно відображає актуальний стан голосувань, що доступні користувачу. У випадку, якщо на момент звернення немає жодного активного голосування, користувач бачить порожній список із чітким та зрозумілим повідомленням про відсутність доступних голосувань. Це допомагає уникнути плутанини і дає зрозуміти, що нових голосувань наразі немає.

Якщо ж голосування доступні, інтерфейс відображає повний список таких голосувань, надаючи користувачу можливість переглянути деталі кожного з них.

Завдяки цьому забезпечується швидкий доступ до необхідної інформації та спрощується процес вибору голосування для участі.

Технічна реалізація цього функціоналу базується на функції loadVotes, яка відповідає за завантаження голосувань із бази даних. Ця функція містить логіку перевірки користувача, його ролі, а також відбору голосувань, які призначені саме для нього.

Крім того, передбачена можливість використання параметра forceRefresh, що забезпечує примусове оновлення списку голосувань, гарантуючи, що користувач бачить найактуальнішу інформацію.

Для наочності, на рисунку 3.5 представлено два варіанти відображення інтерфейсу Учасника голосування: перший — коли список голосувань порожній, а другий – із заповненим переліком активних голосувань. Такий підхід дозволяє краще орієнтуватися у поточному стані програми та підказує користувачу, які дії йому варто виконати далі.

Загалом, описаний інтерфейс сприяє підвищенню зручності використання додатку та підвищує залученість користувачів у процес голосування.

На рисунку 3.6 зображено розгорнуте та виконане опитування.

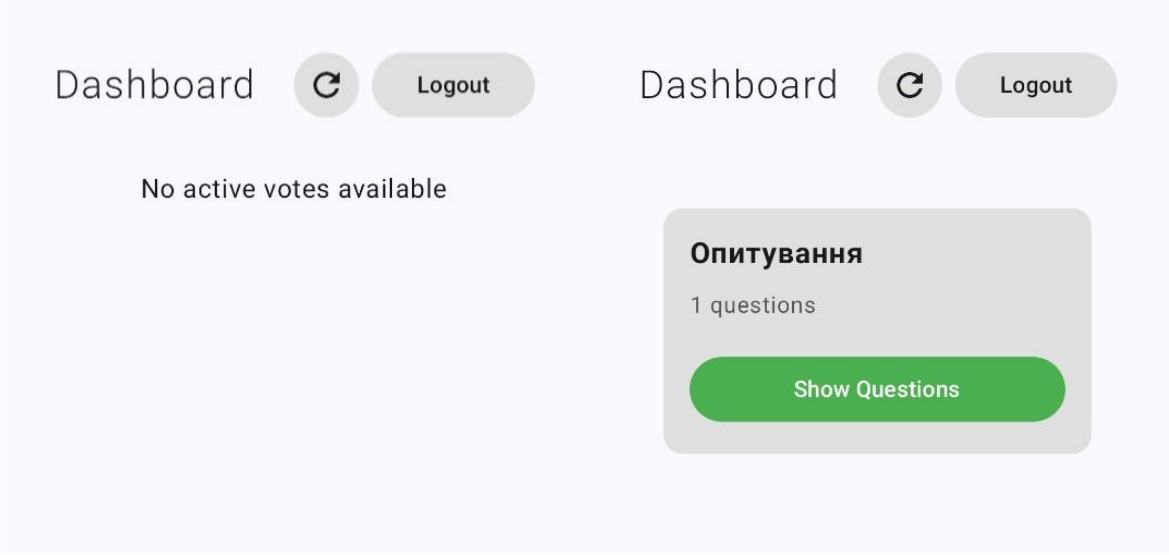

Рисунок 3.5 – Інтерфейс Учасника голосування

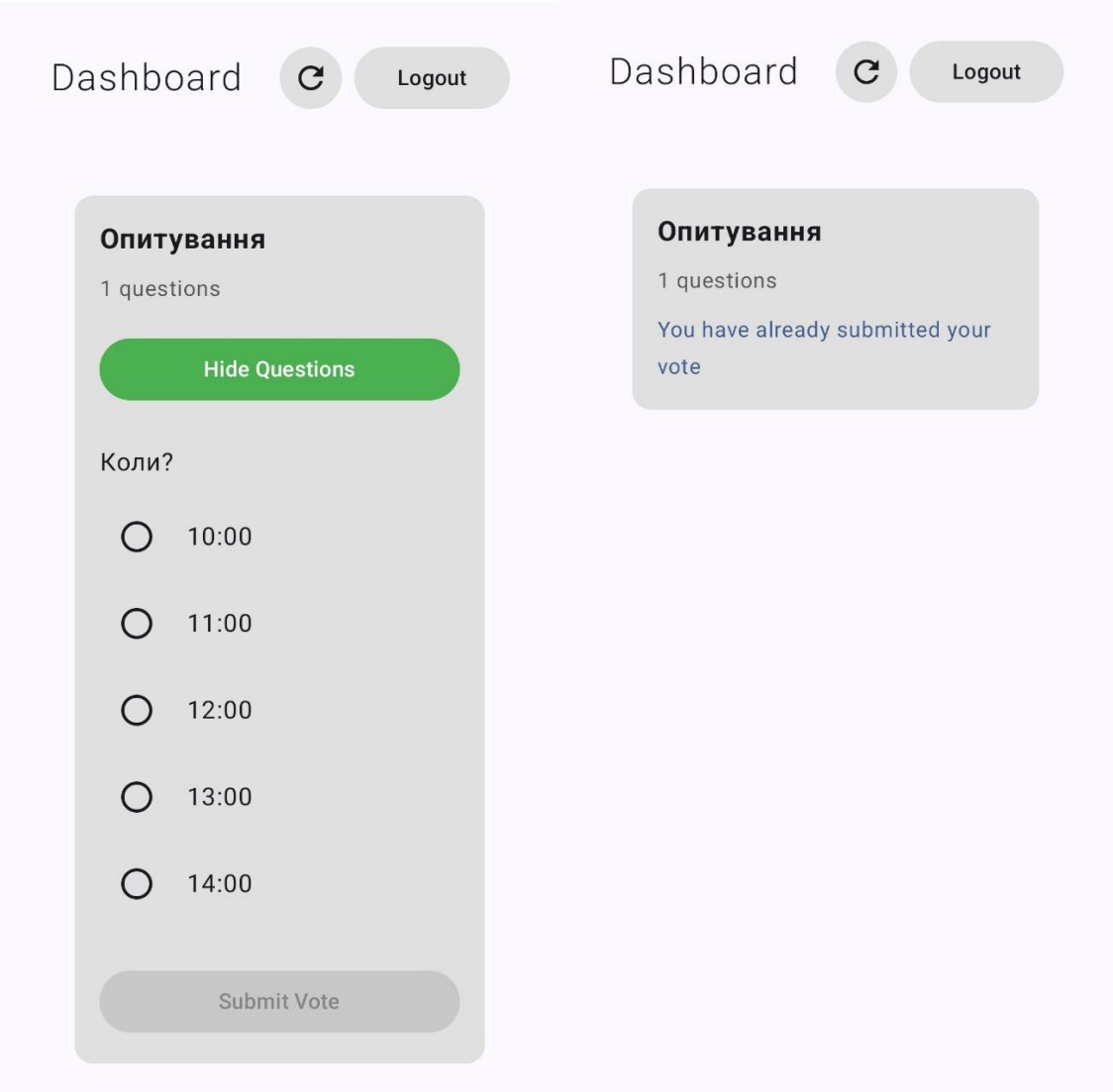

Рисунок 3.6 – Інтерфейс Учасника голосування

3.6 Інтерфейс Адміністратора

Інтерфейс Адміністратора, як і інтерфейс Учасника голосування, спроектований з урахуванням ключових принципів простоти, інтуїтивної зрозумілості та зручності використання. Це забезпечує швидкий та ефективний доступ до основних функцій управління голосуваннями, що є критично важливим для адміністратора системи.

Адміністратор має можливість легко створювати нові опитування, користуючись зручно розміщеною кнопкою в нижній частині інтерфейсу. Цей

процес спрощений до мінімуму – достатньо заповнити необхідні поля, після чого опитування зберігається як чернетка.

Перед тим, як опитування стане доступним для голосування учасниками, його необхідно активувати. При активації в базу даних записується відповідний статус опитування — це може бути “чернетка” (draft), “активне” (active) або “закрите” (closed). Саме цей статус визначає, чи можуть користувачі брати участь у голосуванні. Для перевірки активності опитування застосовується запит до бази даних:

```
db.collection("votes").whereEqualTo("status", "active")
```

Це гарантує, що до голосування можуть долучатися лише активні опитування.

Важливою функцією є те, що після активації голосування адміністратор втрачає можливість редагувати його. Це спеціальний захід, який захищає дані від випадкових або навмисних змін під час проведення голосування, забезпечуючи цілісність та достовірність результатів.

Після завершення голосування адміністратор може закрити опитування. У статусі “закрите” опитування більше не приймає нових голосів, проте вся інформація залишається збереженою в базі даних. Це дозволяє проводити подальший аналіз результатів та надавати доступ до підсумків навіть після завершення голосування.

На рисунку 3.7 представлено приклад інтерфейсу Адміністратора, де видно створене голосування та основні дії з ним.

Для запобігання випадковому видаленню голосування реалізовано механізм підтвердження дії. При спробі видалення відкривається діалогове вікно із запитом підтвердження, як показано на рисунку 3.8. Це допомагає уникнути втрати важливих даних через необережність.

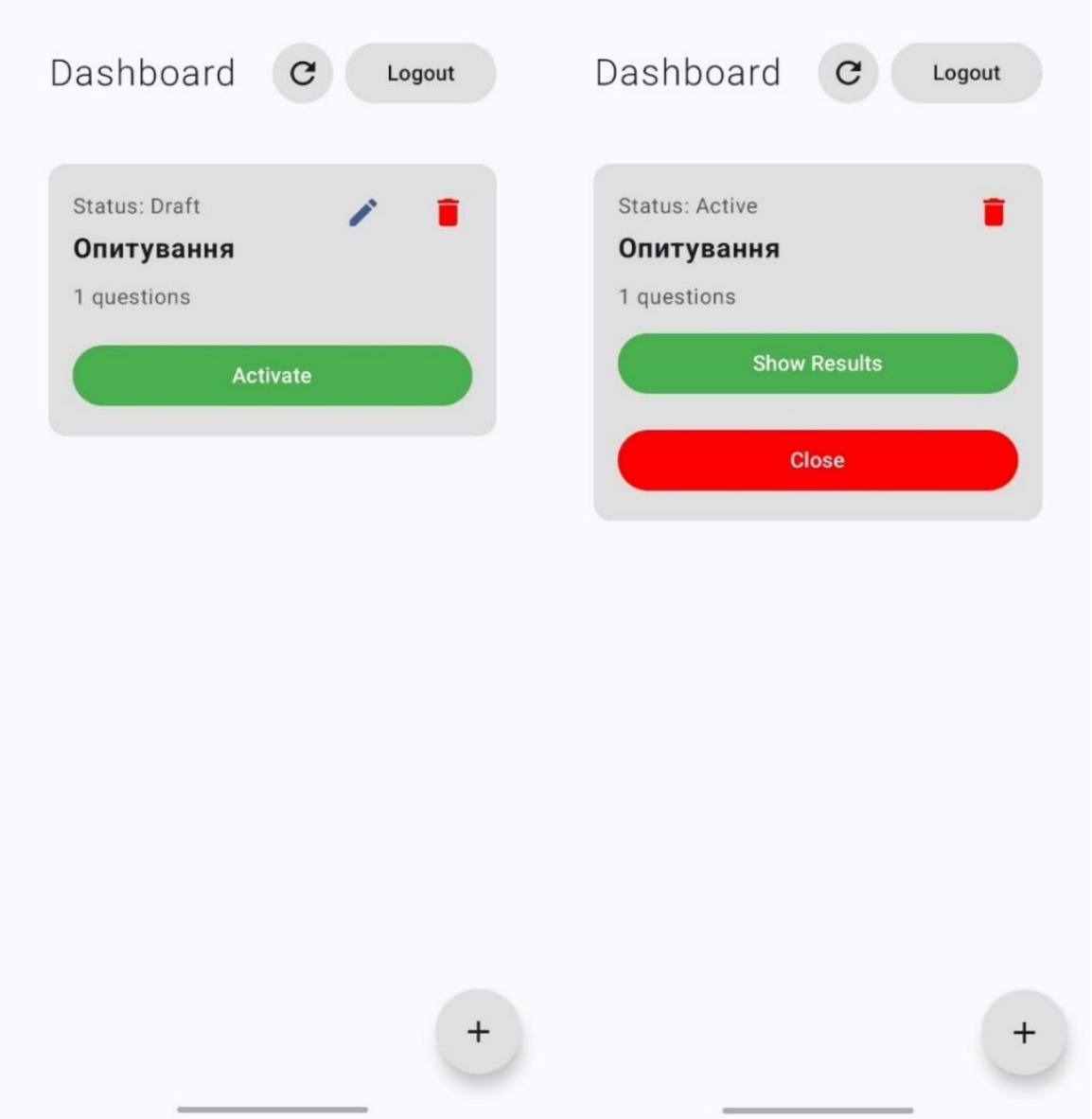

Рисунок 3.7 – Інтерфейс Адміністратора

Крім того, інтерфейс містить окрему кнопку для швидкого перегляду результатів голосування. Така організація дозволяє адміністратору миттєво отримувати доступ до аналітики і підсумків голосування, без необхідності довго шукати цю функцію у меню чи інших розділах.

Загалом, інтерфейс Адміністратора є простим, логічним і водночас функціональним, що сприяє ефективному управлінню голосуваннями на всіх етапах їхнього життєвого циклу.

Для видалення голосування передбачено механізм підтвердження дії, щоб запобігти випадковому видаленню – при спробі видалити голосування з’являється відповідне підтверджувальне вікно, як показано на рисунку 3.8.

Перегляд результатів голосування здійснюється через окрему кнопку, що спрощує адміністратору доступ до аналітики та підсумків без потреби шукати цю функцію у меню.

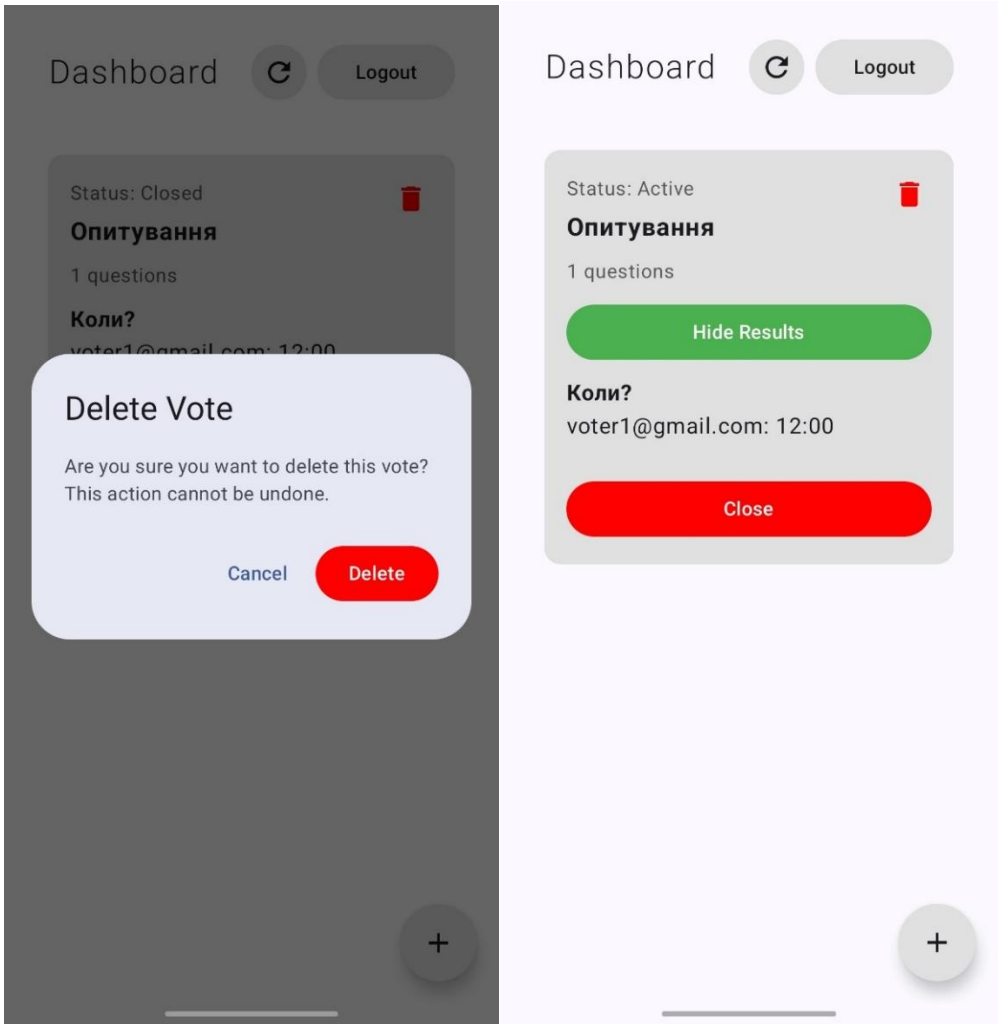

Рисунок 3.8 – Інтерфейс Адміністратора

3.7 Створення та редагування голосувань

При додаванні та редагуванні голосування випливає вікно, зображене на рисунку 3.9.

Тут є поля для введення назви голосування, питань, відповідей (якщо потрібно) та учасників голосування.

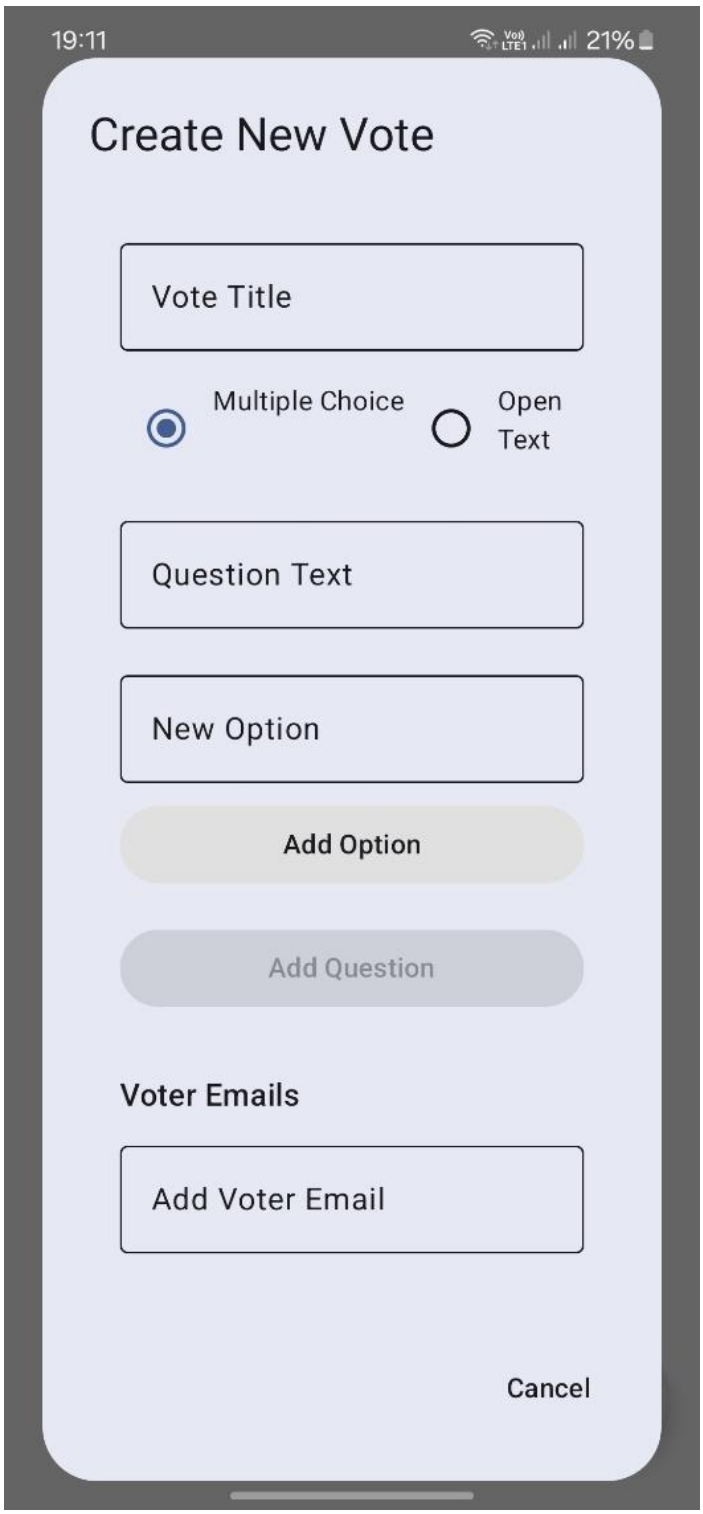

Рисунок 3.9 – Форма створення та редагування голосувань

3.8 Тестування додатку

Тестування є важливим етапом розробки будь якого програмного забезпечення. Воно дозволяє перевірити правильність роботи додатку, виявити можливі помилки або некоректну поведінку компонентів, а також забезпечити стабільність і зручність у користуванні.

У рамках дипломного проєкту проводилось функціональне тестування, а також модульне (unit) тестування окремих логічних частин додатку. Додаток eVote розробляється за допомогою Android SDK, Kotlin та Java, тому для тестування використовувалися стандартні інструменти Android, такі як JUnit, Espresso, а також Firebase Emulator Suite для тестування з підключенням до бази даних.

Для модульного тестування було створено окремий пакет test/ у проекті. Тестування проводилось із використанням бібліотеки JUnit4.

Приклад тестування класу валідації паролю:

```
class Validator {
    fun isPasswordValid(password: String): Boolean {
        return password.length >= 6 && password.any {
it.isDigit() } && password.any { it.isLetter() }
    }
}
class ValidatorTest {
    private val validator = Validator()
    @Test
    fun testValidPassword() {
     assertTrue(validator.isPasswordValid("Test123"))
    }
    @Test
    fun testShortPassword() {
        assertFalse(validator.isPasswordValid("T1"))
    }
    @Test
    fun testNoDigits() {
```

```
assertFalse(validator.isPasswordValid("Password"))
        }
        @Test
        fun testNoLetters() {
         assertFalse(validator.isPasswordValid("123456"))
        }
    }
```

Тестування моделі голосування:

```
    data class Vote(
        val id: String = "",
        val title: String = "",
        val status: String = "draft"
    )
    class VoteTest {
        @Test
        fun testVoteCreation() {
            val vote = Vote(id = "1", title = "Test Vote", status
= "active")
            assertEquals("Test Vote", vote.title)
            assertEquals("active", vote.status)
        }
    }
```

Для тестування з Firebase використовувався Firebase Emulator Suite. Було перевірено: створення користувача через Firebase Auth, авторизація користувача, запис та зчитування голосування у Firestore, перевірка, що користувач бачить лише свої голосування.

Приклад тесту на наявність кнопок входу:

```
    @RunWith(AndroidJUnit4::class)
    class LoginScreenTest {
        @get:Rule
        val rule =
ActivityScenarioRule(LoginActivity::class.java)
```

```
        @Test
        fun testLoginButtonDisplayed() {
            onView(withId(R.id.loginButton))
                .check(matches(isDisplayed()))
        }
        @Test
        fun testNavigateToRegister() {
    onView(withId(R.id.registerLink)).perform(click())
    onView(withId(R.id.registerButton)).check(matches(isDisplayed
()))
        }
    }
```

Проводилось ручне тестування вигляду інтерфейсу у світлій та темній темах. Усі елементи інтерфейсу коректно адаптуються, шрифти контрастні, кнопки мають анімації натискання.

Загалом, проведене тестування забезпечило високу якість додатку, допомогло виявити і усунути помилки на ранніх стадіях розробки, а також підтвердило відповідність додатку заявленим вимогам.

Висновок до розділу

В процесі розробки мобільного додатку eVote особлива увага була приділена саме візуальній частині інтерфейсу користувача, адже саме зручність і простота використання є ключовими факторами, що визначають успіх будь-якого програмного продукту. Головною метою було створити максимально комфортний, інтуїтивно зрозумілий і ненавантажений дизай, який дозволяє користувачам швидко орієнтуватися у функціоналі додатку, не відволікаючись на зайві складності або непотрібні елементи. Такий підхід сприяє підвищенню користувацького досвіду і збільшує мотивацію до активної взаємодії з системою.

Однією з важливих складових інтерфейсу є процес створення та редагування голосувань. Він організований таким чином, щоб користувачі могли

легко і швидко додавати питання, варіанти відповідей і вибирати учасників, не витрачаючи багато часу на розуміння налаштувань. Інтерфейс зроблено компактним, логічним і зрозумілим, що дає змогу уникнути плутанини і помилок при роботі з голосуваннями. Всі елементи управління розміщені у зручних зонах, що забезпечує швидкий доступ до основних функцій.

Процедура проведення самого голосування також розроблена з урахуванням потреб кінцевого користувача. Вона виконана у легкому та простому форматі, що значно підвищує комфорт для учасників, дозволяючи їм зосередитися на суті питання, а не на технічних деталях інтерфейсу. Це особливо важливо, адже спрощення процесу стимулює більшу активність і залученість користувачів до голосувань, що у свою чергу підвищує якість прийнятих рішень.

Важливим аспектом є те, що весь розроблений функціонал і візуальний комфорт реалізовані саме у форматі мобільного додатку, адаптованого для екрану смартфона. На відміну від багатьох аналогічних сервісів, які орієнтовані переважно на великі екрани десктопів і ноутбуків, цей додаток забезпечує повну функціональність і при цьому зберігає зручність використання на компактних дисплеях мобільних пристроїв. Така адаптивність дає можливість користувачам брати участь у голосуваннях будь-де та будь-коли, що є важливою перевагою у сучасному мобільному світі.

Ще одним вагомим плюсом проєкту є застосування мови програмування Java разом із її віртуальною машиною JVM. Це дозволяє досягти кросплатформенності, тобто одна й та сама версія коду може без змін запускатися на різних операційних системах, таких як Android, Windows або Linux. Така універсальність коду спрощує процес підтримки, оновлення та масштабування додатку, а також значно знижує витрати на розробку.

Щодо якості програмного забезпечення, всі функціональні модулі додатку пройшли ретельне тестування на різних рівнях. Зокрема, було проведено як модульне тестування окремих класів і методів, так і інтеграційне тестування взаємодії між компонентами додатку. Тестування здійснювалося як в емуляторі, так і на реальних мобільних пристроях, що підтвердило стабільність і

коректність роботи додатку у різних умовах. Результати тестів свідчать про високу надійність і відповідність системи вимогам, що були поставлені на початку розробки.

Крім того, особливу увагу було приділено адаптації інтерфейсу під різні теми оформлення, зокрема світлу та темну теми, що відповідає сучасним тенденціям у дизайні мобільних додатків. Такий підхід забезпечує комфортне використання додатку в різних освітлювальних умовах і підтримує загальний позитивний досвід користувача.

Таким чином, розроблений додаток eVote є зручним, інтуїтивно зрозумілим і технічно надійним інструментом для організації та проведення голосувань на мобільних пристроях. Його функціонал і дизайн сприяють ефективній взаємодії користувачів з системою, а використання сучасних технологій і ретельне тестування забезпечують високу якість і стабільність роботи.

ВИСНОВКИ

Під час виконання дипломного проєкту було розроблено мобільний додаток, призначений для організації онлайн голосувань. Ця система забезпечує можливість проведення різноманітних типів опитувань і голосувань, що робить її універсальним інструментом для колективного прийняття рішень у різних сферах.

Перед безпосереднім створенням програмного забезпечення було проведено детальний аналіз існуючих аналогічних сервісів та платформ. В результаті цього аналізу були виділені їхні сильні сторони, а також визначені недоліки та аспекти, які потребують удосконалення. Ці висновки стали основою для формування вимог до майбутнього додатку та напрямків його розробки.

Було описано та відтворено логічну структуру бази даних, яка забезпечує збереження та коректну обробку інформації про користувачів, голосування, питання та відповіді.

Розроблений додаток відзначається зручним та інтуїтивно зрозумілим інтерфейсом, що не перевантажує користувача зайвими елементами, а також відповідає сучасним стандартам user-friendly дизайну.

Програма створена для операційної системи Android, але завдяки використанню мови програмування Java та її віртуальної машини JVM має потенціал кросплатформності, що дозволяє запускати додаток на різних пристроях і операційних системах без необхідності значних змін у коді.

Таким чином, виконана робота дозволила досягти поставленої мети дипломного проєкту – створення простого, зручного та ефективного онлайн додатку для голосування, який може успішно використовуватися на мобільних пристроях з операційною системою Android.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Horstmann C. S. Core Java Volume I–Fundamentals : підруч. / Horstmann C. S., Cornell G. – Prentice Hall, 2018. – 850 с.

2. Sierra K. Head First Java : підруч. / Sierra K., Bates B. – O'Reilly Media, 2019. – 720 с.

3. Eckel B. Thinking in Java. – Prentice Hall, 2006. – 1150 с.

4. Isakova S. Kotlin in Action : підруч. / Isakova S., Jemerov D. – Manning Publications, 2017. – 350 с.

7. Smith J. Online Voting Systems: Security and Usability – Journal of E-Government Studies. – 2020.

8. Kotlin Documentation. URL: https://kotlinlang.org/docs/home.html (дата звернення: 01.06.2025).

9. Material Design Guidelines. URL: https://material.io/design (дата звернення: 01.06.2025).

10. Firebase Documentation. URL: https://firebase.google.com/docs (дата звернення: 01.06.2025).

11. Google Forms. URL: https://docs.google.com/forms/ (дата звернення: 01.06.2025).

12. Typeform. URL: https://www.typeform.com/ (дата звернення: 01.06.2025).

13. SurveyMonkey. URL: https://www.surveymonkey.com/ (дата звернення: 01.06.2025).

14. Shah M. Learning Firebase for Android. – Packt Publishing, 2018. – 330 с.

15. Fatih K. Programming Kotlin. – Packt Publishing, 2020. – 400 с.

16. Henderick B. Kotlin Programming: The Big Nerd Ranch Guide. – Big Nerd Ranch Guides, 2019. – 300 с.

17. Burt H. Kotlin for Android Developers. – Leanpub, 2019. – 285 с.

18. Murphy M.L. The Busy Coder’s Guide to Android Development. – CommonsWare, 2021. – 720 с.

19. Parycek P., Edelmann N., Krimmer R. E-Voting: A Comparative Analysis. – Springer, 2016. – 246 с.

20. ISO/IEC 25010:2011. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE). – Geneva: ISO, 2011.

21. Meier R. Professional Android. – Wrox Press, 2021. – 816 с.

22. Bloch J. Effective Java. – 3rd ed. – Addison-Wesley, 2018. – 416 с.

БІБЛІОГРАФІЧНА ДОВІДКА

Тема бакалаврської роботи: ***Розробка мобільного додатку для організації онлайн голосування засобами Java***

Обсяг пояснювальної записки 51 аркушів:

5 таблиць;

25 рисунків;

0 додаток.

Дата завершення роботи: *04 червня 2025р*.

Підпис студента- _____*Пена Й.А.*