МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ
Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії
Кафедра комп'ютерних наук та прикладної математики

"До захисту допущений"
Зав. Кафедри комп'ютерних наук та прикладної математики _____
«___» _____ 20__ р.

КВАЛІФІКАЦІЙНА РОБОТА

Розробка кросплатформного додатку для голосового керування мобільним пристроєм

Виконав: Ковальчук Андрій Олександрович _____
(прізвище, ім'я, по батькові) (підпис)
Група КН-41

Керівник: професор, д.т.н. Турбал Ю.В. _____
(науковий ступінь, вчене звання, посада, прізвище,ініціали) (підпис)

Рівне 2024

ЗМІСТ

РЕФЕРАТ

Кваліфікаційна робота: 48 с., 46 рисунків, 1 таблиця, 13 джерел

Метою кваліфікаційної роботи є розробка кросплатформного додатку для голосового керування мобільним пристроєм з використанням Flutter, Android Studio та Xcode, який дозволяє голосове управління пристроєм, інтерактивні чати з ChatGPT та інтеграцію Google API.

Об'єкт дослідження - процес створення кросплатформного додатку госового керування для мобільних пристроїв.

Предмет дослідження - методи та засоби реалізації голосового керування мобільними пристроями з інтеграцією популярних API.

Методи дослідження - технології Flutter, Android Studio, Xcode, інтеграція з Google API та ChatGPT API.

Основні досягнення роботи включають розробку кросплатформного додатку для голосового керування мобільним пристроєм, інтеграцію API ChatGPT для інтерактивних чатів та використання Google API для розпізнавання мовлення.

Практична цінність полягає у створенні додатку, який дозволяє користувачам взаємодіяти з мобільним пристроєм через голосові команди, що підвищує зручність та дозволяє людям з вадами зору або моторики легко користуватися мобільними пристроями.

Ключові слова: КРОСПЛАТФОРМНІСТЬ, FLUTTER, ANDROID, IOS, РОЗПІЗНАВАННЯ МОВИ, GOOGLE API, CHATGPT API, МОБІЛЬНИЙ ДОДАТОК, ТЕСТУВАННЯ, ІНТЕГРАЦІЯ, ГРАФІЧНИЙ ІНТЕРФЕЙС, ГОЛОСОВІ КОМАНДИ, TTS

ВСТУП

У сучасному світі мобільні додатки стають невід'ємною частиною нашого повсякденного життя. Щодня ми проводимо значну кількість часу зі смартфонами, активно використовуючи мобільні додатки для різноманітних цілей. Вони забезпечують доступ до банківських послуг, спілкуватися, проводити час у соціальних мережах, дозволяють здійснювати покупки, навчатись та працювати. Все більше компаній вирішують перенести свої послуги на мобільні платформи, прагнучи зробити їх доступними для ширшого кола користувачів.

На сьогоднішній день на ринку мобільних пристроїв домінують дві основні операційні системи: Android та iOS. Кожен постачальник програмного забезпечення зацікавлений у тому, щоб його додаток працював на обох цих системах. Для цього існує кілька підходів.

Традиційний підхід до розробки додатків передбачає створення окремих версій для кожної операційної системи з використанням нативних інструментів та середовищ розробки. Оскільки операційні системи мають свої специфічні особливості, такий метод потребує значних ресурсів, як часових, так і фінансових, а також глибокого розуміння особливостей кожної платформи.

Альтернативним варіантом є використання кросплатформних технологій для створення додатків. Цей підхід набирає популярності, оскільки дозволяє розробникам одночасно створювати додатки для різних операційних систем. Це суттєво знижує витрати та час розробки, адже не потрібно розробляти окремі версії додатку для кожної платформи.

Кросплатформеність вказує, що додаток, розроблений цим підходом, може функціонувати на різних операційних системах. Це значно скорочує час і витрати на розробку, оскільки немає потреби створювати окремі версії додатку для кожної платформи. Крім того, такі додатки простіше оновлювати і підтримувати. Проте, кросплатформені додатки мають і свої мінуси: вони зазвичай поступаються в продуктивності нативним додаткам та можуть мати

обмежений доступ до деяких специфічних функцій кожної операційної системи.

Одним із сучасних рішень для створення кросплатформних додатків є Flutter. Цей фреймворк, розроблений компанією Google у 2017 році, дозволяє розробляти додатки для iOS, Android, вебу, а також десктопні додатки для Windows, macOS і Linux. Flutter використовує мову програмування Dart і базується на графічному рушії Skia, який забезпечує високоякісну візуалізацію інтерфейсів. Особливістю Flutter є власний набір віджетів, які відображаються безпосередньо на екрані, минаючи нативні компоненти операційних систем. Це дозволяє додаткам мати однаковий вигляд і поведінку на різних платформах.

Однією з головних переваг Flutter є його можливість використовувати платформозалежні функції через Platform Channels, що дозволяє Dart-коду взаємодіяти з нативними API Android та iOS. Крім того, Flutter підтримує роботу з нативним кодом Android та iOS, що дає розробникам гнучкість у створенні складних додатків. Завдяки багатому набору вбудованих віджетів та інструментів, Flutter дозволяє швидко і ефективно створювати сучасні та привабливі мобільні додатки. Цей фреймворк забезпечує високу продуктивність і дозволяє тонко налаштовувати інтерфейс під специфічні вимоги кожної платформи.

Одним із сучасних рішень для розпізнавання мови є технологія Speech-to-Text, яка дозволяє перетворювати усне мовлення на текстові дані. Ця технологія використовується в багатьох додатках, включаючи асистентів, транскрипцію розмов, голосове управління та багато інших.

Однією з ключових особливостей Speech-to-Text є полегшення введення тексту для користувачів з фізичними обмеженнями або тих, хто не володіє швидким набором тексту. Дозволяє користувачам виконувати інші завдання паралельно та покращує доступність додатків для користувачів з обмеженими можливостями

Метою роботи є розробка кроссплатформного додатку для дистанційного керування пристроєм за допомогою голосових команд з використанням Flutter. Порядок роботи включає огляд літератури, розробку дизайну додатку, реалізацію основних функціональних можливостей та написання підсумкової документації.

РОЗДІЛ 1

НАТИВНІ ТЕХНОЛОГІЇ

1.1. Огляд існуючих рішень

Платформа Flutter не єдине існуюче рішення на ринку, що дозволяє створювати додатки для різних операційних систем. Сьогодні існує декілька інших конкурентних платформ, кожна з яких має свої переваги та недоліки. Деякі з них будуть описані нижче.

Одним із рішень, яке варто розглянути, є Kotlin Multiplatform, розроблене компанією JetBrains. Це інноваційна технологія, що дозволяє розробникам писати спільний код для різних платформ, включаючи Android, iOS, веб та JVM. Платформа підтримує повний доступ до SDK кожної платформи та забезпечує високу продуктивність додатків. Це досить нова технологія, тому складність налаштування може стати викликом для новачків. Проте ця технологія має великий потенціал для створення ефективних і функціональних кросплатформних рішень і в майбутньому здатна замінити Flutter.

Ще одним рішенням є Xamarin, відкрита платформа від Microsoft для створення додатків для iOS і Android за допомогою .NET. Xamarin одним із перших рішень для розробки кросплатформних додатків та до виходу Flutter був одним з лідерів. Xamarin дозволяє створювати UI, або на кілька платформ одразу, або окремо для кожної з них. Додатковим плюсом є можливість писати код на C#, яка є розвиненою мовою програмування та легкою до вивчення. Крім того, Microsoft нещодавно випустила оновлення для Xamarin під назвою .NET MAUI (Multi-platform App UI), яке покращує розробку в Xamarin. Однак варто зазначити, що Flutter, у порівнянні з Xamarin, забезпечує кращу продуктивність і результати. Найбільшим недоліком Xamarin є нижча продуктивність, що робить його найкращим варіантом для простих додатків, які не мають багато функціоналу.

Ще однією досить популярною платформою, яку варто розглянути, є React Native. Найбільшою відмінністю від вищеописаних рішень є використання веб-технологій та JavaScript. Створена компанією Facebook, ця платформа призначена для розробки кросплатформних додатків для Android та iOS. Однією з головних переваг React Native є доступ до нативних API, що дозволяє створювати додатки з нативною продуктивністю та виглядом. Проте, серед недоліків можна відзначити, що для дуже складних і ресурсомістких додатків може знадобитися написання додаткового нативного коду для забезпечення оптимальної продуктивності.

Останнім обговорюваним рішенням рішенням є NativeScript. Ця бібліотека, як і React Native використовує веб-технології та дає можливість розробляти кросплатформні додатки для Android та iOS, використовуючи JavaScript або TypeScript. Основною перевагою NativeScript є те, що він забезпечує доступ до нативних API та використовує нативні елементи графічного інтерфейсу. Завдяки цьому створені мобільні додатки виглядають і працюють так само, як і нативні додатки, створені з використанням середовищ розробки для конкретних операційних систем. NativeScript надає повний доступ до всіх функцій пристрою, включаючи сенсори, камеру, GPS та інші.

Оглядаючи описані платформи, можна зробити висновок, що використання цих платформ забезпечує значну економію часу та ресурсів завдяки повторному використанню коду. Кожна платформа розроблена різними компаніями, кожна з яких має свій підхід до створення мобільних додатків. Вибір найкращої платформи залежить від конкретних вимог проекту, адже кожна з них має свої унікальні переваги та недоліки. Таким чином, вирішити, яка платформа є найкращою, можна лише враховуючи специфіку та цілі конкретного додатка.

Технологія розпізнавання мови (Speech-to-Text) стає все більш популярною завдяки своїй зручності та ефективності, але це не єдине доступне на рішення. Сьогодні є кілька подібних технологій, які мають як переваги, так і недоліки. Деякі з них будуть описані нижче.

Перше рішення, яке буде обговорено, - це Microsoft Azure Speech Services. Цей сервіс від Microsoft також забезпечує високу якість розпізнавання мови та підтримку багатьох мов. Він легко інтегрується з іншими продуктами Microsoft. До недоліків використання можна віднести те, що сервіс може бути дорогим для великих проектів та для роботи сервісу необхідне стабільне підключення до інтернету.

IBM Watson Speech to Text пропонує високоякісне розпізнавання мови з можливістю налаштування моделей під специфічні потреби користувачів. Перевагами є високий рівень безпеки та конфіденційності даних, підтримка багатьох мов та можливість аналізу настроїв в аудіо. Недоліками є висока вартість використання для масштабних проектів, складність налаштування та необхідність стабільного інтернет-з'єднання.

Amazon Transcribe від AWS надає можливість автоматичного перетворення мови в текст і підтримує великий спектр мов. Перевагами є підтримка розпізнавання мови в реальному часі, легка інтеграція з іншими сервісами AWS, автоматичне вставлення розділових знаків та формування абзаців, а також здатність розпізнавати мову в умовах значного фонового шуму. Недоліками є висока вартість для великих обсягів даних, потреба в стабільному інтернет-з'єднанні та обмежена підтримка деяких менш розповсюджених мов або діалектів.

Дивлячись на описані вище сервіси розпізнавання мови, можна зробити висновок, що кожне з них має свої особливості та підходить для різних завдань. Вибір відповідного сервісу залежить від конкретних вимог проекту,

таких як необхідна точність розпізнавання, підтримувані мови, вартість використання та інтеграція з іншими системами.

1.2. Android

1.2.1. Архітектура

Android — це мобільна операційна система від Google, яка є найпоширенішою у світі. Вона використовується на переважній більшості мобільних пристроїв. Однією з основних переваг Android є її відкритість у порівнянні з iOS, яка може працювати лише на пристроях від Apple. Архітектура Android базується на ядрі Linux і структурована у декілька рівнів, як показано на рисунку 1.

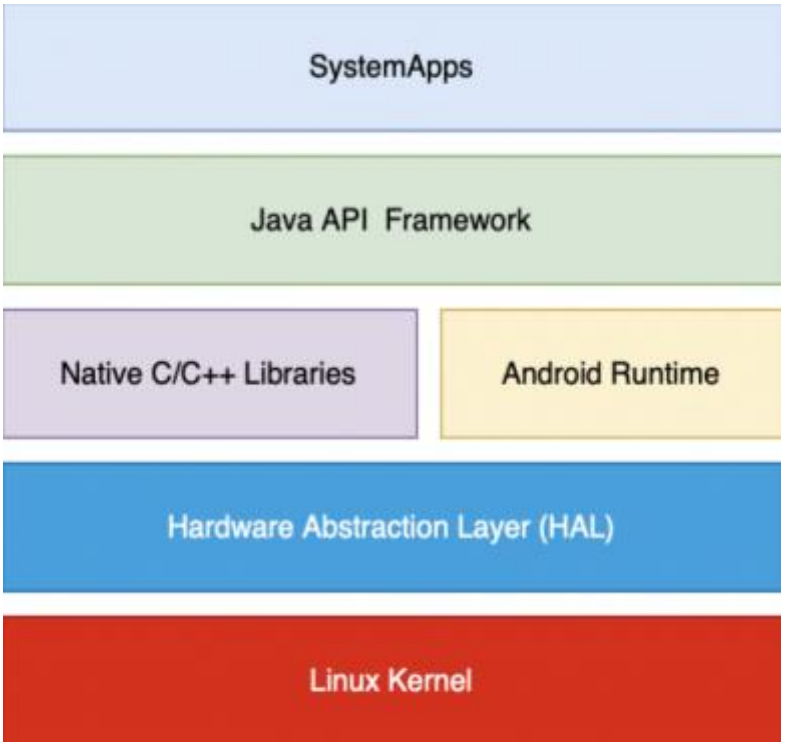

Рис 1. Архітектура Android

Для кращого розуміння архітектури Android, коротко буде описано кожен її рівень.

Ми використовуємо верхні рівні архітектури для створення додатків, що надають відповідні можливості. У випадку Android це рівень «Java API

Framework», який забезпечує доступ до API, написаного на Java. Цей рівень включає наступні компоненти:

• **Activity Manager** – відповідає за керування життєвим циклом додатків, зокрема за запуск, зупинку, паузу та відновлення діяльності додатків.

• **Менеджер ресурсів** - відповідає за надання доступу до різних ресурсів додатка, таких як файли, зображення, звукові файли

• **Система перегляду** - відповідає за створення та управління графічним інтерфейсом додатка

• **Content Providers** - відповідають за надання додаткам доступу до даних, збережених в інших додатках або до спільних наборів даних

• **Notification Manager** - відповідає за створення та відображення повідомлень, надісланих додатками

На самому нижньому рівні знаходиться ядро Linux, яке використовується Android для забезпечення доступу до основних функцій, таких як багатопотокова робота, керування живленням і пам'яттю. Цей рівень також підтримує різні типи драйверів, включаючи аудіо, Wi-Fi та USB.

Вище ядра Linux знаходиться рівень апаратно-програмної абстракції, який містить модулі для взаємодії з апаратним забезпеченням пристрою, такими як камера та Bluetooth, забезпечуючи доступ до цих компонентів з вищих рівнів архітектури.

Наступним рівнем є рівень бібліотек C/C++. Ці бібліотеки доступні через «Java API Framework» і надають функціонал для різних задач. Наприклад, бібліотека SQLite дозволяє працювати з базами даних, а OpenGL забезпечує підтримку 2D і 3D графіки. Цей рівень також включає середовище виконання операційної системи Android. До версії Android 5.0 використовувалася віртуальна машина «Dalvik», яка здійснювала компіляцію додатків методом

«Just-in-Time» (JIT). З версії Android 5.0 замість неї використовується «Android Runtime» (ART), яка застосовує компіляцію «Ahead-of-Time» (AOT). З версії Android 7.0 в ART також підтримується JIT-компіляція.

Найвищий рівень архітектури включає стандартні додатки Android, такі як камера, календар і SMS, які забезпечують базовий функціонал для користувачів.

1.2.2. Середовище Android Studio

Android Studio — це інтегроване середовище розробки, призначене для створення додатків для пристроїв з операційною системою Android. Розроблене та випущене Google у 2014 році, Android Studio замінило Eclipse, яке раніше використовувалося для розробки додатків для цієї платформи. Це середовище є кросплатформним, що означає, що воно підтримує розробку програмного забезпечення на Windows, Linux і macOS. Однією з головних переваг Android Studio є можливість створення додатків для всіх типів пристроїв Android, включаючи мобільні телефони, планшети та Android TV.

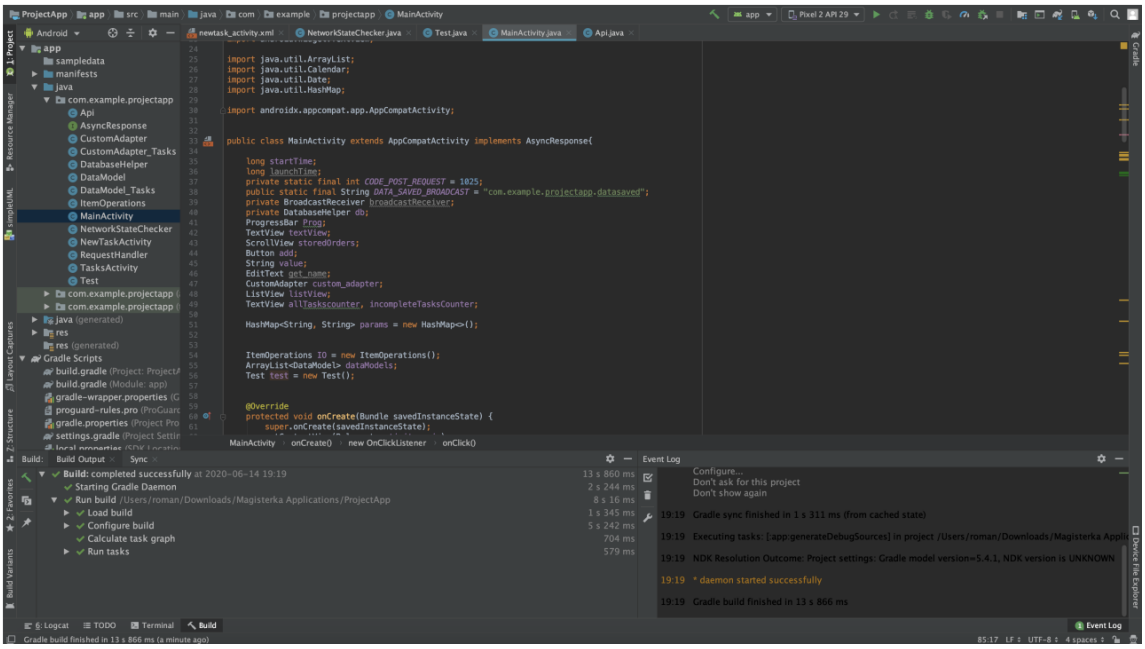

Рис 2. Середовище Android Studio

При розробці додатків реалізація логіки додатка та створення графічного інтерфейсу розділяються. Для створення графічного інтерфейсу Android

Studio надає потужний інструмент, який містить усі необхідні елементи, такі як кнопки, текстові поля, списки тощо. Інтерфейс можна створювати як за допомогою коду, так і за допомогою графічного редактора.

Однією з переваг використання Android Studio є можливість постійного попереднього перегляду того, як виглядає додаток. Режим попереднього перегляду дозволяє побачити, як додаток виглядатиме на різних пристроях, з різними розмірами екрана та в різних версіях операційної системи Android. Це забезпечує розробникам зручний спосіб перевірки та налаштування інтерфейсу під час процесу розробки.

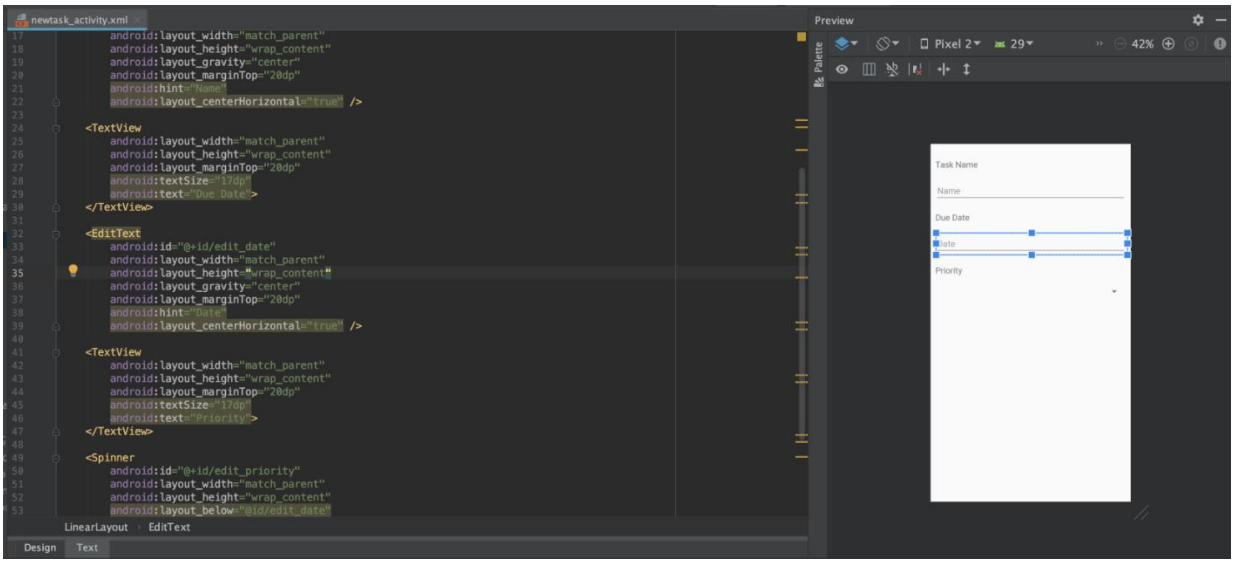

Рис 3. Android Studio. Розробка графічного інтерфейсу

Коли справа доходить до створення логіки додатку, більшість додатків пишуться на Java, об'єктноорієнтованій мові, яка нині належить Oracle. Як альтернативу можна використовувати мову програмування Kotlin, яка, хоча і набирає популярність, все ще не настільки широко використовувана, як Java.

Для тестування реалізованих додатків в Android Studio використовується вбудований інструмент Profiler, який дозволяє розробнику аналізувати використання оперативної пам’яті, навантаження на центральний процесор та інші параметри додатка. Для налагодження додатку використовується інструмент AVD Manager, який дозволяє створювати емулятори мобільних

пристроїв з обраною версією операційної системи. Крім того, є можливість перевіряти роботу додатку безпосередньо на фізичному пристрої.

1.3. IOS

1.3.1. Архітектура

IOS — це операційна система, створена для мобільних пристроїв компанії Apple. Її архітектура базується на операційній системі macOS і має багато спільних рис. Подібно до Android, архітектура iOS розділена на різні рівні, як показано на рис. 4.

Рис 4. Архітектура IOS

При розробці додатків для цієї операційної системи рекомендовано використовувати високі рівні архітектури. Зокрема, в архітектурі iOS найвищим рівнем є платформа “Cocoa Touch”, яка надає доступ до всіх необхідних інструментів для створення додатків та доступу до нижчих рівнів архітектури. Одним з ключових інструментів “Cocoa Touch” є платформа UIKit, що дозволяє розробляти більшість додатків для iOS. Використовуючи UIKit, програміст працює з класом UIViewController, який у рамках шаблону

«Model-View-Controller» (MVC) відповідає за зв'язок між моделлю та інтерфейсом користувача. В інтерфейсі розміщуються графічні елементи, які через контролер взаємодіють з моделлю. Кожен інтерфейс додатку повинен контролюватися окремим екземпляром UIViewController.

Додатково, UIKit пропонує широкі можливості для анімації та взаємодії з користувачем. Інші важливі компоненти "Cocoa Touch" включають "Foundation framework", що забезпечує основні функціональні можливості, такі як управління даними, робота з датами, рядками та колекціями. Взаємодія з апаратним забезпеченням і датчиками здійснюється за допомогою "Core Motion" та "Core Location", що дозволяють створювати більш інтерактивні та контекстно-залежні додатки.

Наступним важливим інструментом є Storyboard, який використовується для створення графічного інтерфейсу додатку. Storyboard забезпечує візуальне представлення інтерфейсу користувача та дозволяє розробникам зручно працювати з розташуванням і взаємодією між елементами. Використовуючи Storyboard, можна налаштовувати переходи між екранами, а також задавати взаємодії між різними елементами інтерфейсу. Storyboard буде розглянуто більш детально при описі Xcode.

Наступним рівнем архітектури iOS є рівень «Медіа», що включає графіку, відео, аудіо та анімації. Цей рівень забезпечує роботу з мультимедійним контентом через бібліотеки, такі як "AVFoundation" для аудіо та відео, "Core Animation" для анімацій, та "Core Graphics" для двовимірної графіки. Завдяки цим бібліотекам можна створювати інтерактивні та мультимедійні додатки з багатим користувацьким досвідом.

Наступним рівнем архітектури є «Core Services», який включає стандартні служби для iOS, такі як місцезнаходження, iCloud, контакти тощо. Як і з рівнем «Медіа», ми можемо працювати з «Core Services» через «Cocoa

Touch». Цей рівень надає доступ до важливих функцій, що дозволяє інтегрувати сервіси та забезпечувати їхню роботу в додатках.

Останній, найнижчий рівень — це рівень «Core OS», який забезпечує такі можливості операційної системи iOS, як аутентифікація, безпека та доступ до апаратного забезпечення пристрою. Цей рівень включає основні функції операційної системи, що забезпечують стабільність і безпеку роботи додатків.

1.3.2. Середовище Xcode

Xcode — це інтегроване середовище розробки, створене та випущене Apple, яке використовується для створення додатків для macOS та iOS. Одним з його недоліків є те, що створювати мобільні або десктопні додатки можна лише на пристроях з встановленою операційною системою macOS. На відміну від цього, Android Studio дозволяє створювати додатки на будь-якому пристрої з операційними системами Windows, Linux або macOS.

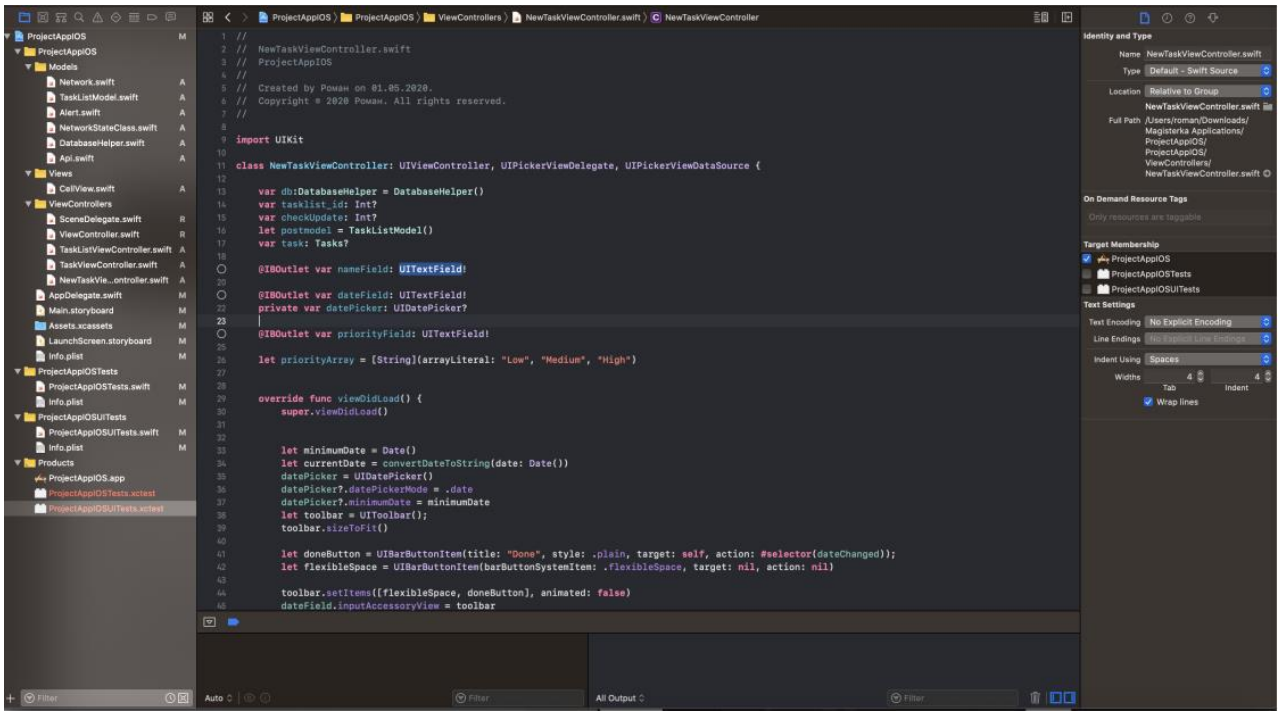

Рис 5. Середовище Xcode

Однак варто відзначити, що Apple створила чудове середовище, яке надає розробнику всі необхідні інструменти для роботи. Для створення

графічного інтерфейсу Xcode пропонує інструмент під назвою Storyboard. З його допомогою можна створювати графічний інтерфейс додатку без написання коду, додаючи необхідні елементи (кнопки, написи, текстові поля тощо). Усі елементи розміщуються у відповідному файлі Storyboard, що полегшує їхнє керування. Завдяки режиму попереднього перегляду, програміст може постійно перевіряти, як додаток виглядатиме на різних пристроях.

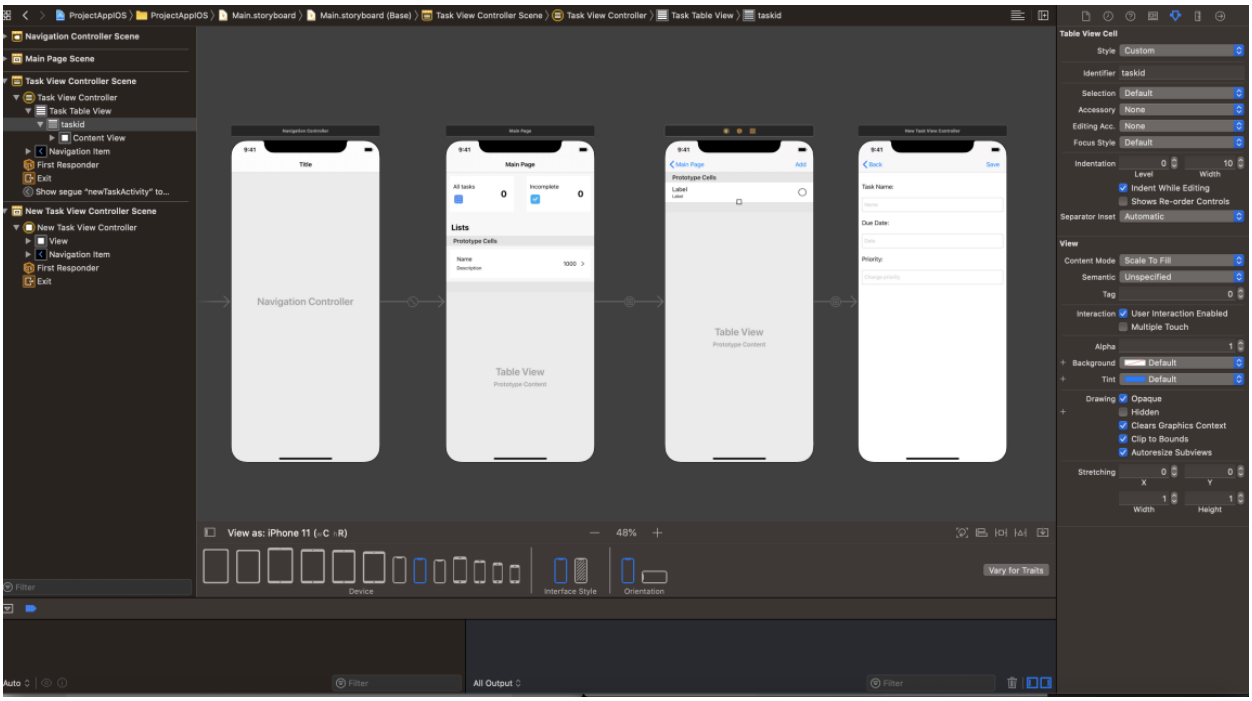

Рис 6. Xcode, Storyboard

Вся логіка додатку базується на мовах програмування Swift або Objective-C. Swift, створений Apple як альтернатива Objective-C, є найкращим вибором для створення нативних мобільних додатків для iOS, оскільки він швидший і компілюється безпосередньо в машинний код. Крім того, Swift має більш сучасний синтаксис і забезпечує вищу продуктивність, що робить його популярним серед розробників.

Xcode також надає потужний інструмент для тестування розроблених додатків, відомий як інструменти розробника. Використання цього інструменту дозволяє аналізувати додатки для виявлення їхніх слабких сторін. Якщо у вас немає фізичного пристрою з iOS, ви можете використовувати

емулятор, наданий Xcode, для запуску і перевірки роботи додатка. Проте емулятор підходить лише для перевірки функціональності. Для тестування продуктивності слід використовувати реальний пристрій, оскільки емулятори використовують ресурси комп'ютера, що може спотворити результати тестування. Крім цього, Xcode надає можливості для інтеграції з різними системами контролю версій, такими як Git, що спрощує спільну роботу над проєктом і керування версіями коду.

1.4. Архітектура Flutter

Flutter — це кросплатформенний UI-інструментарій, розроблений для створення високопродуктивних додатків для різних операційних систем, таких як iOS, Android, веб та десктоп. Архітектура Flutter базується на декількох основних принципах та складається з багатошарової системи, що забезпечує гнучкість та масштабованість додатків. Архітектура flutter розділена на різні рівні, як показано на рис.7.

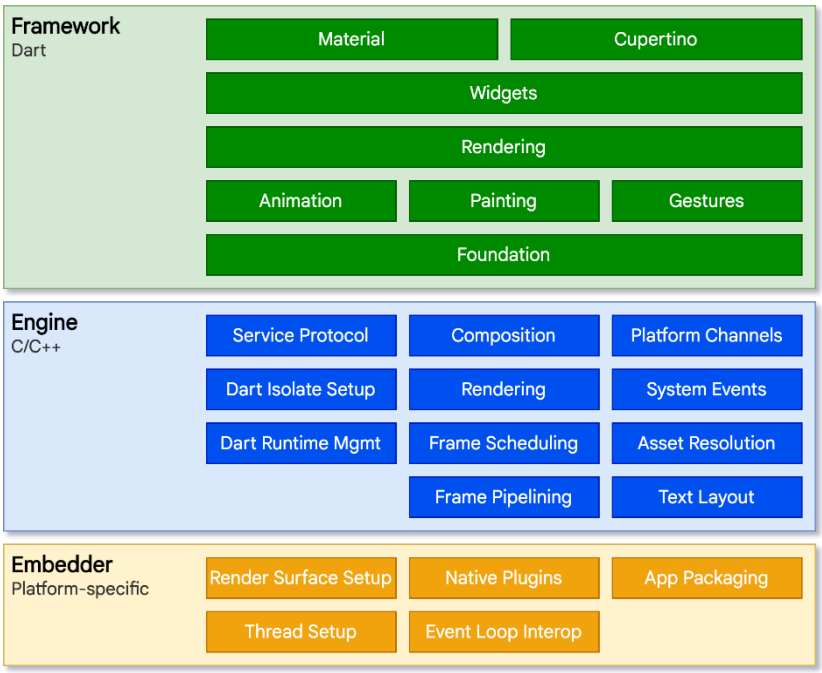

Рис.7. Архітектура Flutter

На найвищому рівні архітектури розташовується Flutter Framework, що складається з набору бібліотек, написаних на мові програмування Dart. Цей

фреймворк включає компоненти для створення користувацького інтерфейсу, управління станом, анімацій та інших функцій. Побудований на принципах реактивного програмування, він забезпечує автоматичне оновлення інтерфейсу користувача у відповідь на зміни стану додатка.

Ядро Flutter — це Flutter Engine, який здебільшого написаний на C++. Це основний компонент системи, відповідальний за рендеринг сцен, обробку анімацій та управління графічним контентом. Engine використовує бібліотеки Skia та Impeller для рендерингу графіки, а також надає низькорівневі API, які підтримують роботу всіх додатків, створених на Flutter.

На найнижчому рівні архітектури Flutter знаходиться Embedder, який забезпечує взаємодію з операційною системою. Embedder відповідає за доступ до системних ресурсів, таких як рендеринг, обробка вводу, доступність та інші сервіси ОС. Він написаний мовами, що відповідають конкретним платформам: Java і C++ для Android, Objective-C/Objective-C++ для iOS і macOS.

У Flutter все представлено у вигляді віджетів. Віджети є основними будівельними блоками інтерфейсу користувача, що визначають зовнішній вигляд та поведінку UI-компонентів.

Flutter використовує реактивний підхід до створення інтерфейсу користувача, де зміни в стані автоматично викликають оновлення відповідних компонентів. Це реалізовано за допомогою stateful та stateless віджетів, що дозволяє ефективно керувати станом додатку. Крім того були розроблені додаткові патерни до керуванням станом додатку, серед яких найбільш популярним є BLoC, який я обрав у своїй роботі.

BLoC (Business Logic Component) — це архітектурний патерн, який використовується у Flutter для керування станом додатка та розділення бізнес-логіки від інтерфейсу користувача (UI). Основна концепція BLoC полягає в

створенні чистого коду, що легко підтримувати, тестувати та масштабувати. Ось принципи роботи BLoC:

• Події (Events) відображають дії, що можуть статися у додатку, наприклад, натискання кнопки. Вони передаються до BLoC через Sink.

• Стан (State) визначає поточний стан додатка. Він змінюється відповідно до подій та передається на інтерфейс користувача через Stream.

• BLoC клас виконує ключову роль у обробці подій та керуванні станом. Він отримує події через Sink, обробляє їх та оновлює стан, який потім передається на інтерфейс користувача через Stream.

РОЗДІЛ 2

РОЗРОБКА І ВПРОВАДЖЕННЯ ДОДАТКУ

2.1. Архітектура додатку та схеми дизайну

Архітектура мого додатку складається з окремих елементів, що забезпечують його функціональність і взаємодію з користувачем, як показано на рис.8. Нижче кожен з них буде описано більш детально.

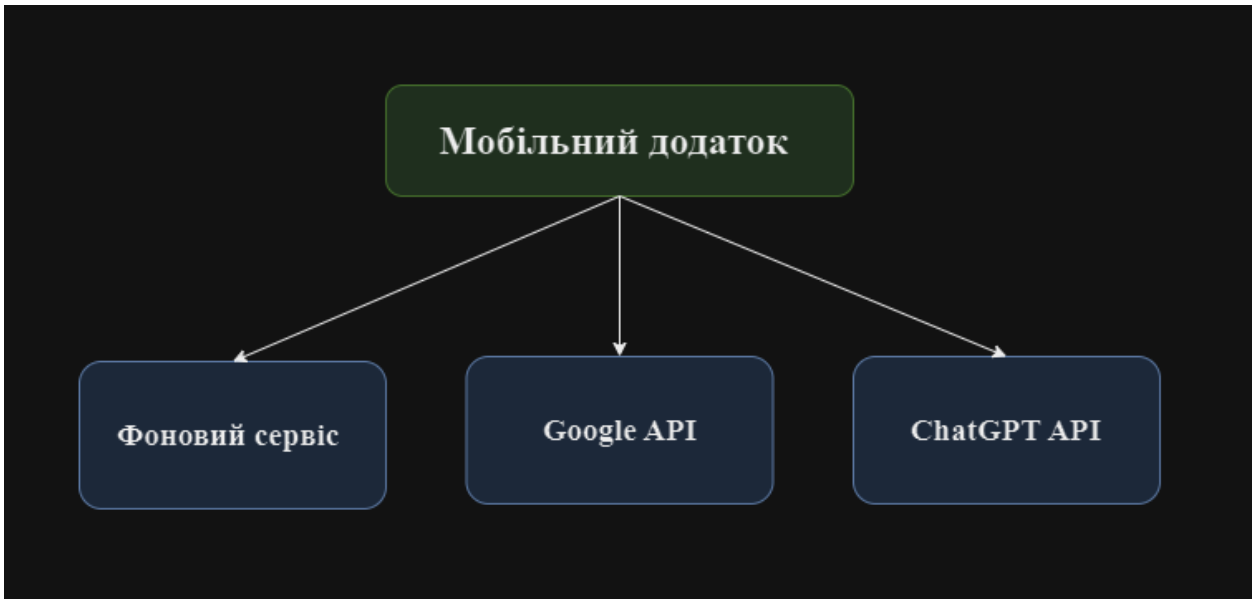

Рис.8. Архітектура додатку

Перший елемент, з яким пов'язаний мобільний додаток, — це сам додаток, розроблений на Flutter. Він забезпечує користувачам інтерфейс для взаємодії з функціональністю додатку, такою як управління пристроєм через голосові команди та використання можливостей ChatGPT для інтерактивних чатів. Головний файл додатку (main.dart) ініціалізує додаток та завантажує головне меню. Для керування додатку використовується патерн BLoC (Business Logic Component), який дозволив ефективно відокремити бізнес-логіку від інтерфейсу користувача і максимально полегшив тестування та масштабування коду. Дякуючи цьому вдалося мінімізувати ризик некоректної роботи додатку

Іншим важливим елементом є фоновий сервіс, який забезпечує виконання задач у фоновому режимі, навіть коли додаток неактивний. Фоновий сервіс використовує пакет flutter_background_service для реалізації цієї функціональності. Він відповідає за постійне прослуховування голосових команд та виконання відповідних дій, таких як запуск додатків або встановлення таймерів.

Також одним з ключових елементів є Google API, який забезпечує інтерактивну взаємодію з користувачами. Додаток відправляє запити до Google API та отримує відповіді, які виконують дії відповідно до запиту. Це може бути як і отримання потрібної музики або відео, так і побудова потрібного для користувача маршруту. Для виконання запитів використовується Dio - бібліотека для мови програмування Dart, яка надає можливості для здійснення HTTP-запитів у Flutter-додатках.

Ще одним ключовим елементом є API ChatGPT, який забезпечує інтерактивну взаємодію з користувачами. Додаток відправляє запити до API ChatGPT та отримує відповіді, які потім озвучуються користувачам за допомогою Text-to-Speech (TTS). Цей функціонал реалізовано у файлі chat_gpt.dart.

Основні екрани додатку:

• **MainMenu** - головне меню додатку, де користувачі можуть обирати різні функціональні можливості, такі як управління пристроєм або взаємодія з ChatGPT. Клас MainMenu забезпечує навігацію між різними екранами додатку.

• **ControlApp** - клас, що відповідає за управління пристроєм за допомогою голосових команд. Користувачі можуть запускати або зупиняти фоновий сервіс, який слухає голосові команди. Він також забезпечує відображення статусу мікрофону та останнього розпізнаного повідомлення.

Він забезпечує прослуховування голосових команд та виконання відповідних дій навіть коли додаток знаходиться у фоновому режимі. Сервіс налаштовується та запускається під час ініціалізації додатку.

• **ChatGPT** - клас для інтеграції з ChatGPT API. Він забезпечує взаємодію з користувачем через текстовий або голосовий інтерфейс, відправляє повідомлення до API та обробляє відповіді. Клас також використовує TTS для озвучення відповідей користувачу.

2.2. Графічний інтерфейс і структура меню

Оскільки додаток розроблений за допомогою Flutter, для створення графічного інтерфейсу використовувалися різні компоненти Flutter. Усі кнопки, мітки, списки або текстові поля відповідним чином перетворюються на власні елементи операційної системи, на якій працює додаток. Тому, створюючи графічний інтерфейс, важливо враховувати особливості як для iOS, так і для Android. Ось детальний опис інтерфейсу вашого додатку.

MainMenu є головним меню додатку, де користувачі можуть обирати різні функціональні можливості, такі як управління пристроєм або взаємодія з ChatGPT. В коді 1 показано приклад створення інтерфейсу.

Інтерфейс створено за допомогою Flutter і включає такі елементи:

• AppBar - панель додатка з назвою "Main Menu".

• Фонове зображення - декоративне зображення, що додає естетики додатку.

• Кнопки навігації - круглі кнопки з іконками для переходу до різних функцій додатку.

Графічний інтерфейс додатку наведено на рис.9.

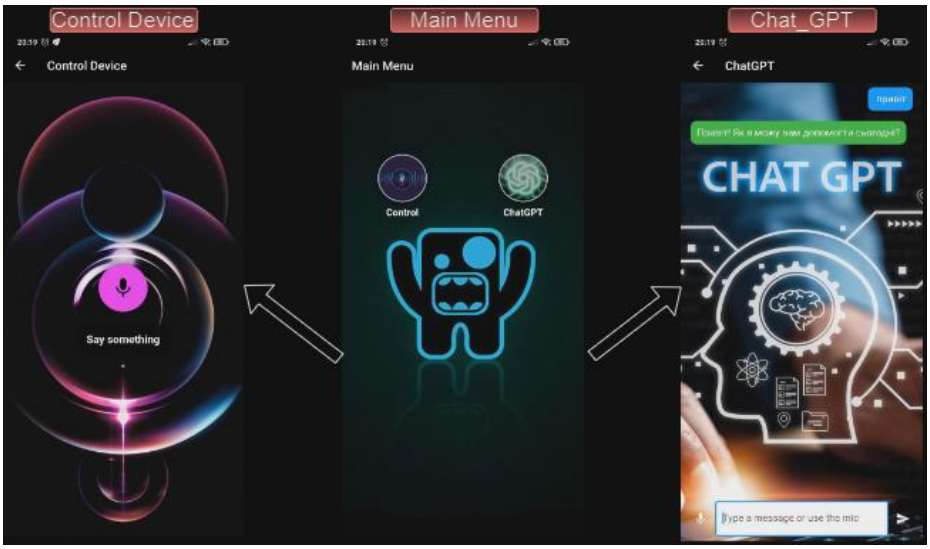

Рис.9. Створення графічного інтерфейсу

2.3. Впровадження Технологій

Однією з функціональних можливостей розроблених додатків було забезпечення взаємодії з користувачем за допомогою сучасних технологій. Для досягнення цієї мети було інтегровано три основні технології: розпізнавання голосу (Speech-to-Text), використання Google API та інтеграція API ChatGPT. Ці технології дозволяють додатку надавати інтерактивні можливості, покращуючи користувацький досвід та розширюючи функціональність додатку.

2.3.1. Реалізація технології Speech-to-Text

Технологія розпізнавання мови дозволяє користувачам взаємодіяти з додатками за допомогою голосових команд, що значно покращує користувацький досвід та доступність. Впровадження функціоналу Speech-to-Text у Flutter-додатках може бути виконано за допомогою плагіна speech_to_text. Плагін speech_to_text дозволяє розпізнавати голос та перетворювати його у текст в режимі реального часу. Він забезпечує інтеграцію з системами розпізнавання мови на платформах Android та iOS. Використання цього плагіна дозволяє створювати додатки, які можуть виконувати голосові команди, шукати інформацію тощо.

Перед тим як почати використання необхідно ініціалізувати плагін та налаштувати його в коді додатку.

В даному додатку використовується технологія розпізнавання мови для забезпечення голосового управління та взаємодії з користувачем. Ця функціональність дозволяє користувачам взаємодіяти з додатком за допомогою голосових команд, що значно покращує користувацький досвід. Виведення відбувається в режимі реального часу,для розуміння коректності поданої користувачем команди.Приклад наведено на рис. 10.

Рис. 10. Приклад голосової команди та її виведення

2.3.2. Впровадження Google API

Google API надає доступ до широкого спектра сервісів Google, таких як YouTube, Google Maps, Google Drive тощо. Впровадження Google API в додатки дозволяє розширити функціональність, інтегруючи можливості цих сервісів. В даному додатку використовується Google API для пошуку відео та музики на платформах YouTube та YoutubeMusic.

Використання Google API у Flutter-додатку передбачає виконання HTTP-запитів до сервісів Google з використанням API-ключа для автентифікації. Після отримання результатів запитів, вони можуть бути оброблені та відображені у додатку. Для роботи з Google API необхідно отримати API-ключ від Google Cloud Platform. Цей ключ використовується для автентифікації запитів до Google API. У рис.11 наведено основний код для інтеграції Google API.

```
Future<Either<Failure, String?>> search({
  required String query,
  required SpeechState state,
}) async {
  try {
    final response = await dio.get(
      "youtube/v3/search",
      queryParameters: {
        "part": "id",
        "q": query,
        "regionCode": "US",
        "type": "video",
        "videoCategoryId": "10",
        "key": ApiKeys.youtubeApiKey,
      },
    );
    final data = response.data;
    return Right(YoutubeIds.fromJson(data).ids.firstOrNull);
  } catch (e, _) {
    return Left(
      Failure(errorMessage: "Request failed ${e.toString()}"),
    );  // Left
  }
```

Рис.11. Інтеграція Google API

Для інтеграції пошуку відео та музики на платформах YouTube використовується метод searchYoutube з класу GoogleApi, який виконує HTTP-запит до YouTube Data API та повертає список ID відео/музики.

```
void _searchVideo(String query) async {
  final googleApi = GoogleApi();
  try {
    List<String> videoIds = await googleApi.searchYoutube(query);
    if (videoIds.isNotEmpty) {
      _launchYoutubeVideo(videoIds.first);
    }
  } catch (e) {
    print('Error: $e');
  }
}
void _launchYoutubeVideo(String videoId) async {
  final url = 'https://www.youtube.com/watch?v=$videoId';
  if (await canLaunch(url)) {
    await launch(url);
  } else {
    throw 'Could not launch $url';
  }
}
```

Рис.12.Використання Google API

Інтеграція Google API у Flutter-додатки відкриває широкі можливості для взаємодії з різними сервісами Google. Використання API для пошуку відео на YouTube дозволяє створити більш інтерактивний та зручний додаток для користувачів. Це також розширює функціональність додатку, надаючи доступ до багатьох інших сервісів Google.

2.3.3. Впровадження API ChatGPT

ChatGPT API від OpenAI надає можливість інтеграції потужного генеративного моделювання мови у ваші додатки. Це дозволяє створювати інтерактивні чат-боти, які можуть відповідати на запити користувачів, генерувати текст на основі вхідних даних та забезпечувати інші функції обробки природної мови.

ChatGPT API дозволяє надсилати запити до моделі ChatGPT та отримувати відповіді, які можна використовувати для різних завдань, таких як обробка запитів користувачів, генерація тексту, підтримка діалогів тощо. Для використання API необхідно мати API-ключ, який можна отримати на платформі OpenAI.

```dart
import 'dart:convert';
import 'dart:io';
import 'package:flutter/material.dart';
  final String apiKey = 'YOUR_API_KEY';

  Future<String> sendMessage(String message) async {
    final uri = Uri.parse('https://api.openai.com/v1/chat/completions');
    final requestBody = jsonEncode({
      'model': 'gpt-4',
      'messages': [
        {'role': 'user', 'content': message},
      ],
    });

    final httpClient = HttpClient();
    try {
      final request = await httpClient.postUrl(uri);
      request.headers.set('Content-Type', 'application/json');
      request.headers.set('Authorization', 'Bearer $apiKey');
      request.add(utf8.encode(requestBody));

      final response = await request.close();
      final responseBody = await response.transform(utf8.decoder).join();

      if (response.statusCode == 200) {
        final decodedResponse = jsonDecode(responseBody);
        final chatCompletion = ChatCompletion.fromJson(decodedResponse);
        return chatCompletion.choices.first.message.content;
      } else {
        throw Exception(
            'Failed to load response: ${response.statusCode} - ${response.reasonPhrase}');
      }
    } catch (e) {
      throw Exception('Failed to connect to ChatGPT API: $e');
    } finally {
      httpClient.close();
    }
  }
}
```

Рис.13. Інтеграція ChatGPT API, який дозволяє надсилати запити до API та обробляти відповіді.

```
Future<void> _sendMessage(String message) async {
  final chatGpt = ChatGPT();
  try {
    String response = await chatGpt.sendMessage(message);
    setState(() {
      _messages.add({'role': 'assistant', 'content': response});
    });
  } catch (e) {
    setState(() {
      _messages.add({'role': 'bot', 'content': 'Error: $e'});
    });
  }
}
```

Рис.14. Метод sendMessage з класу ChatGPT, який надсилає запит до API та повертає відповідь.

Інтеграція ChatGPT API у Flutter-додатки дозволяє створювати інноваційні та інтерактивні рішення, що використовують можливості генеративного моделювання мови. Використання API для обробки запитів користувачів забезпечує високу якість та точність відповідей, що покращує загальний користувацький досвід.

РОЗДІЛ 3

ПЕРЕВІРКА ВІДПОВІДНОСТІ ФУНКЦІОНАЛЬНИМ ВИМОГАМ

Найважливішим аспектом роботи є тестування створеного додатку. Для цього були проведені різні види тестів функціональності додатку. Тестування проводилося на двох мобільних пристроях з операційною системою IOS і Android відповідно.

Використані пристрої:

• iPhone 14 із встановленою iOS версії 17.5

• Xiaomi Redmi Note 7 із встановленим Android версії 10.

Таблиця 1 містить інформацію про використані мобільні пристрої.

Таблиця 1. Інформація про пристрій

	iPhone 14	Xiaomi Redmi Note 7
Процесор	Apple A15 Bionic	Qualcomm Snapdragon 660
Частота процесора	3,2 GHz	2,20 GHz
Кількість ядер процесора	6	8
Вбудована пам'ять	128 GB	32 GB
ОЗП	6 GB	3 GB

3.1. Тестування функції «Робота додатку у фоновому режимі»

Тест полягає в можливості використання додатку у фоновому режимі. Тому, щоб перевірити роботу додатку, я використовував функціонал свого додатку в інших вікнах.

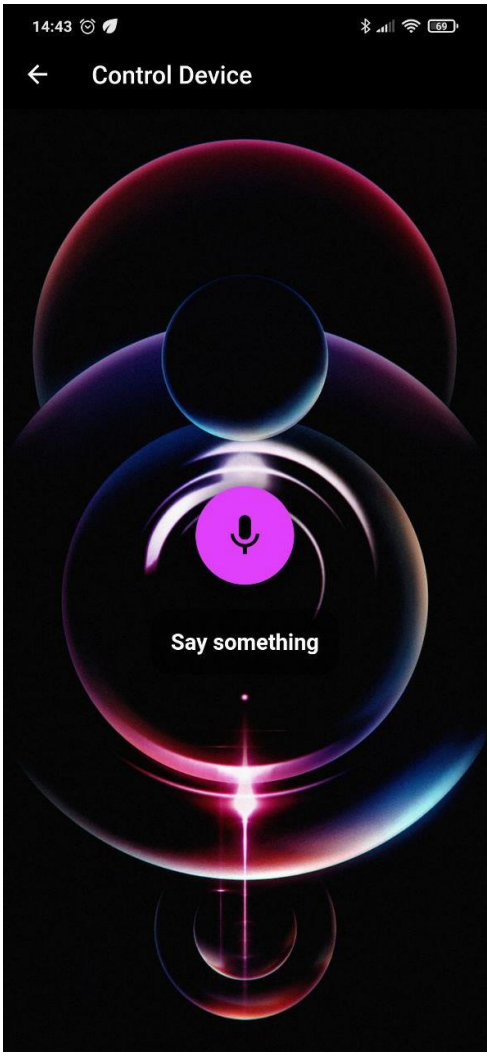

Рис.15. Вікно додатку “Control Device”

Рис.16. Задання команди

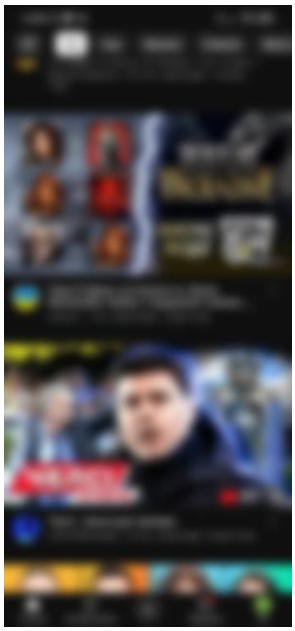

Рис.17. Відкриття YouTube

Рис.18. Задання команди

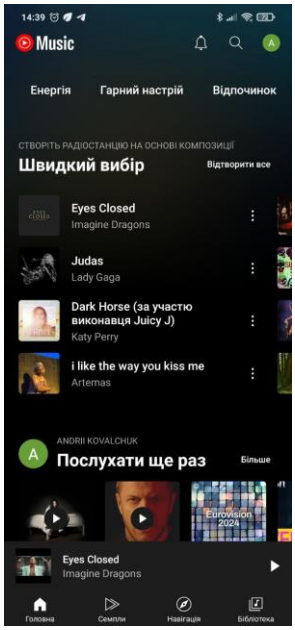

Рис.19. Відкриття YouTubeMusic

Після відкриття вікна “Control Device” я сказав голосову команду “open youtube” та після цього відкрився додаток YouTube. На рис. 16-17 зображено перехід до Youtube за допомогою певної голосової команди.

На рис. 18 зображено команду, яку я оголосив у фоновому режимі перебуваючи у додатку Youtube. На рис. 19 зображено результат цієї команди, а саме перехід у додаток YouTubeMusic.

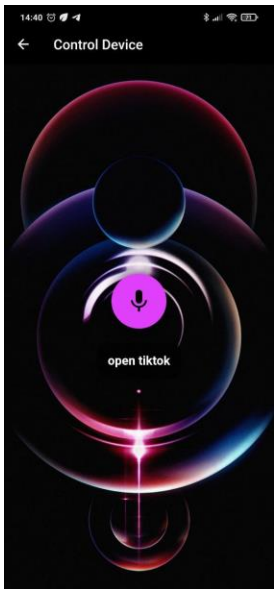

Рис.20. Задання команди

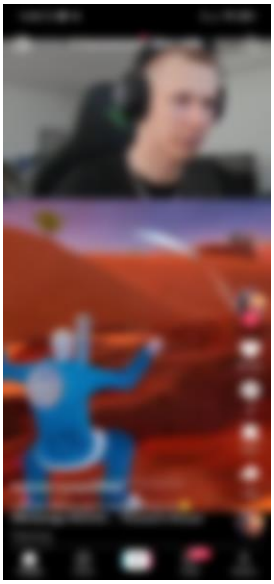

Рис.21. Відкриття Tiktok

Рис.22. Задання команди

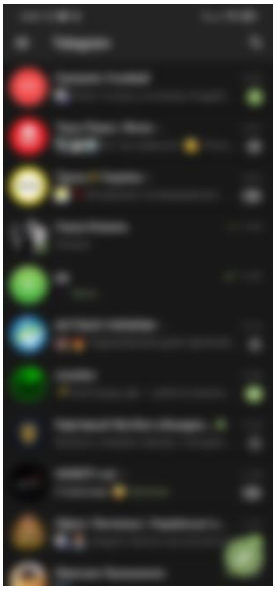

Рис.23. Відкриття Telegram

На рис. 20 зображено команду, яку я оголосив у фоновому режимі перебуваючи у додатку YoutubeMusic. На рис. 21 зображено результат цієї команди, а саме перехід у додаток TikTok.

На рис. 22 зображено команду, яку я оголосив у фоновому режимі перебуваючи у додатку Tiktok. На рис. 23 зображено результат цієї команди, а саме перехід у додаток Telegram.

Рис.24. Задання команди

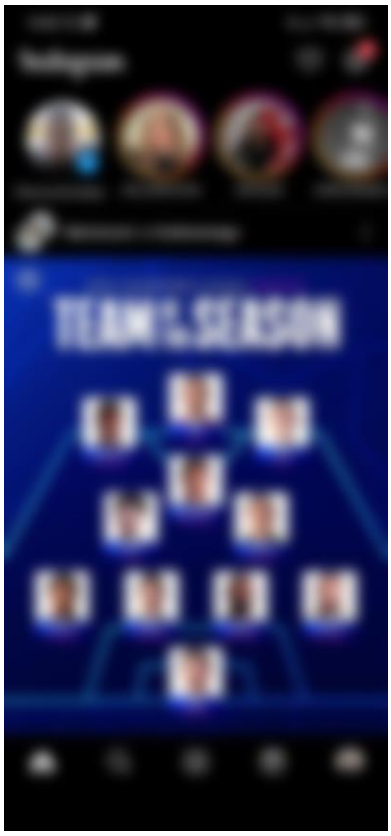

Рис.25. Відкриття Instagram

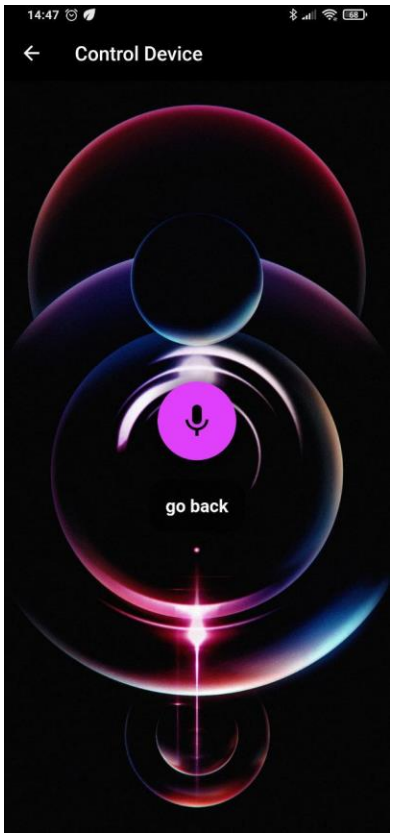

Рис. 26.Повернення в додаток

На рис. 24 зображено команду, яку я оголосив у фоновому режимі перебуваючи у додатку Telegram. На рис. 25 зображено результат цієї команди, а саме перехід у додаток Instagram.

Також я створив фонову команду “go back”, яка буде повертати користувача назад до додатку. На рис. 26 показано результат її виконання.

Таким чином я перевірив роботу свого додатку у фоновому режимі за допомогою команд переходу до інших мобільних додатків. Я можу вільно перемикатися між встановленими додатками та виконувати команди перебуваючи в них

3.2. Тестування функції «Play video»

Тест полягає в можливості відкриття потрібного користувачу відео за допомогою платформи Youtube. Тому, щоб це перевірити, я спробував відкрити декілька посилань.

Рис.27. Задання команди

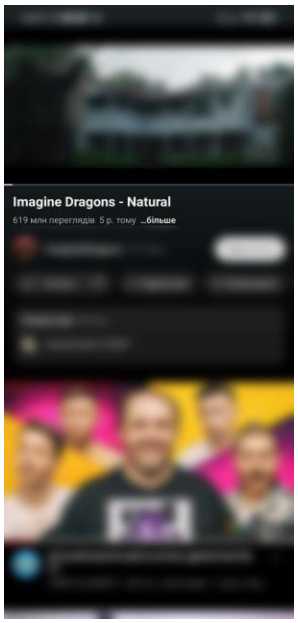

Рис.28. Відкриття потрібного відео на Youtube

Рис.29. Задання команди

Рис.30. Відкриття потрібного відео на Youtube

На рис. 27 зображено команду, яку я оголосив для відкриття потрібного мені відео. На рис. 28 зображено результат цієї команди, а саме перехід у додаток Youtube та відкриття відео “Imagine Dragons – Natural”.

На рис. 29 зображено команду, яку я оголосив для відкриття потрібного мені відео. На рис. 30 зображено результат цієї команди, а саме перехід у додаток Youtube та відкриття відео “Teresa&Maria”.

Таким чином я перевірив відкриття потрібного мені відео на двох різних мовах. Також перевірив можливість зупинки відео за допомогою відстеження рівня гучності звуку, який записується, щоб вирішити, чи зупиняти відтворення. Амплітуда звуку постійно відстежується через таймер, який періодично перевіряє поточний рівень амплітуди, коли ведеться запис. Якщо амплітуда перевищує певний поріг, мікрофон вмикається та очікує голосову команду.

3.3. Тестування функції «Play music»

Тест полягає в можливості відкриття потрібної користувачу музики за допомогою платформи YoutubeMusic. Тому, щоб це перевірити, я спробував відкрити декілька посилань.

Рис.31. Задання команди

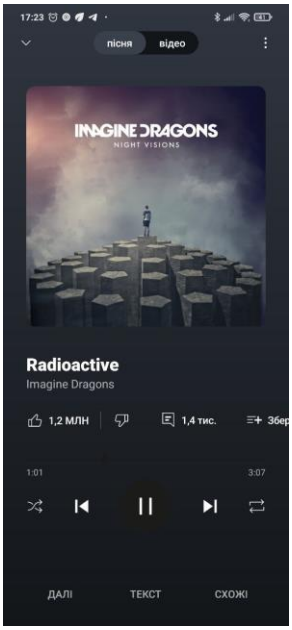

Рис.32. Відкриття потрібної музики на YoutubeMusic

Рис.33. Задання команди

Рис.34. Відкриття потрібної музики на YoutubeMusic

На рис. 31 зображено команду, яку я оголосив для відкриття потрібної мені пісні. На рис. 32 зображено результат цієї команди, а саме перехід у додаток YoutubeMusic та відкриття пісні “Imagine Dragons – Radioactive”.

На рис. 33 зображено команду, яку я оголосив для відкриття потрібної мені пісні. На рис. 34 зображено результат цієї команди, а саме перехід у додаток YoutubeMusic та відкриття пісні “Океан Ельзи - Обійми”.

Таким чином я перевірив відкриття потрібної мені пісні на двох різних мовах. Також перевірив можливість зупинки музики за допомогою відстеження рівня гучності звуку, який записується, щоб вирішити, чи зупиняти відтворення. Амплітуда звуку постійно відстежується через таймер, який періодично перевіряє поточний рівень амплітуди, коли ведеться запис. Якщо амплітуда перевищує певний поріг, мікрофон вмикається та очікує голосову команду.

3.4. Тестування функції «Побудова шляху»

Тест полягає в можливості побудови шляху з місцязнаходження користувача до потрібного місця призначення за допомогою додатку Google Maps. Тому, щоб перевірити роботу цієї функції, я спробував зробити декілька запитів.

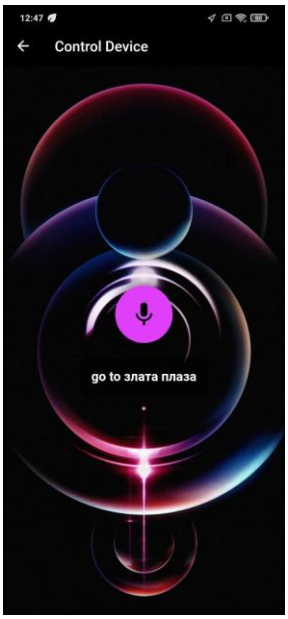

Рис.35. Задання команди

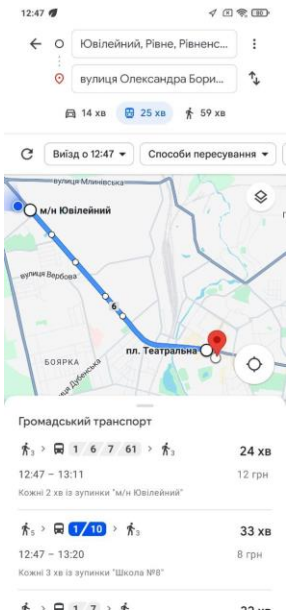

Рис.36. Побудова маршруту

Рис.37. Задання команди

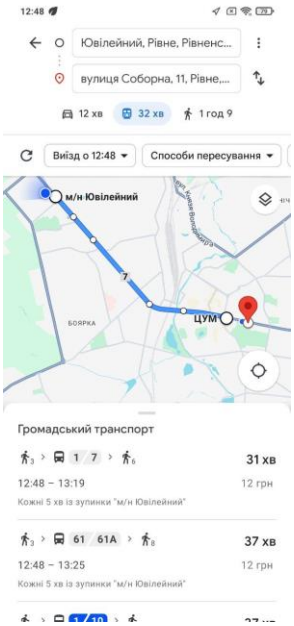

Рис.38. Побудова маршруту

На рис. 35 зображено команду, яку я оголосив для побудови маршруту до потрібного місця призначення. На рис. 36 зображено результат цієї команди, а саме перехід у додаток Google Maps та побудови маршруту до торговельно-розважального центру "Злата Плаза" на вулиці Олександра Борисенка, 1.

На рис. 37 зображено команду, яку я оголосив для побудови маршруту до потрібного місця призначення. На рис. 38 зображено результат цієї команди, а саме перехід у додаток Google Maps та побудови маршруту до першого корпусу НУВГП на вулиці Соборна, 11.

Таким чином я перевірив побудову потрібних мені маршрутів та правильність введення даних.

3.5. Тестування функцій «Будильник та Таймер»

Тест полягає в можливості поставити Будильник та Таймер на потрібний користувачу час. Тому, для перевірки я спробував виконати ці голосові команди.

Рис.39
Виставлення таймера

Рис.40
Виставлення будильника

На рис. 39 зображено команду, яку я оголосив для встановлення таймеру на 10 секунд. Виконання команди має голосовий супровід, при встановленні говориться що “Поставлено таймер на 10 секунд”, а після закінчення часу: “ Таймер виконано”.

На рис. 40 зображено команду, яку я оголосив для встановлення будильнику на 20:00. Виконання команди має голосовий супровід, при встановленні говориться що “Будильник встановлено на 20:00”, а в 20:00: “Будильник спрацював”.

Таким чином я перевірив роботу даних функцій та правильність введення даних.

3.6. Тестування функції «Зміна гучності»

Тест полягає в можливості змінити гучність пристрою за допомогою голосових команд. Тому, для перевірки я спробував виконати потрібні для цього команди.

Рис.41. Зменшення гучності

Рис.42. Збільшення гучності

Рис.43. Вимкнення гучності

Рис.44. Максимальна гучность

На рис. 41 зображено команду, яку я оголосив для зменшення гучності пристрою на 10%. На рис. 42 зображено результат команди, яка збільшила гучність пристрою на 10%.

На рис. 43 зображено команду, яку я оголосив для встановлення гучності пристрою на мінімум. На рис. 44 зображено результат команди, яка встановила максимальну гучність на пристрої.

Таким чином я перевірив роботу даних функцій та правильність введення даних.

3.7. Тестування вікна «Chat_GPT»

Тест полягає в можливості подання голосового запиту до Chat_GPT та отриманні голосової відповіді. Тому, для перевірки я спробував надіслати запити на різних мовах.

Рис.45. Запит до ChatGPT

Рис.46. Запит до ChatGPT

Після відкриття вікна “ChatGPT” я активував мікрофон та за допомогою голосової команди надіслав запит та після цього отримав відповідь на це відповідь від чату GPT. На рис. 41-42 приклади запитів, надісланих на різних мовах та отриманих на них відповідей. Ці відповіді були відтворені за допомогою технології TTS(Text-to-Speech).

ВИСНОВКИ

У даній кваліфікаційній роботі проведено розробку кросплатформного додатку для голосового керування мобільним пристроєм з використанням технології Flutter. Проект включає інтеграцію сучасних технологій, таких як Google API та ChatGPT API, що забезпечує розширену функціональність та інтерактивність додатку.

Робота представляє новий підхід до створення кросплатформних додатків, що дозволяє значно скоротити час розробки та зменшити витрати. Використання Flutter та інших передових технологій демонструє можливості сучасних інструментів для розробки мобільних додатків.

Розроблений додаток дозволяє користувачам зручно керувати мобільними пристроями за допомогою голосових команд. Інтеграція з популярними сервісами Google API та ChatGPT API забезпечує широкі можливості для розширення функціональності додатку, покращуючи користувацький досвід.

Використання кросплатформних технологій дозволяє знизити витрати на розробку додатків, що є важливим фактором для малих та середніх підприємств. Соціальна цінність полягає в покращенні доступності мобільних додатків для широкого кола користувачів, включаючи людей з обмеженими можливостями.

У роботі використано метод кросплатформної розробки з акцентом на інтеграцію голосових команд та розширені функціональні можливості. Детально проаналізовано існуючі рішення та обрано найбільш ефективні технології для реалізації поставлених завдань.

Розробка додатку дозволяє значно скоротити час та ресурси, необхідні для створення аналогічних нативних додатків для різних платформ.

Впроваджені технології забезпечують високу продуктивність та гнучкість у розробці додатків.

Розроблений додаток успішно виконує поставлені завдання, забезпечуючи голосове управління та інтерактивну взаємодію з користувачами. В подальшому розвитку додатку планую розширити функціонал та покращити інтеграцію з іншими популярними сервісами.

Використання кросплатформних технологій та інтеграція з розширеними API буде зростати в майбутньому, що відкриває нові можливості для створення інноваційних мобільних додатків. Розроблені рішення можуть бути адаптовані для різних галузей, що сприятиме подальшому розвитку мобільних технологій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Maurice Sharp, Erica Sadun, Rod Strougo, A Hands-on Guide to the Fundamentals of iOS Programming, Addison-Wesley

2. Learning Android Application Programming: A Hands-On Guide to Building Android Applications

3. Beginning Flutter: A Hands On Guide to App Develo pment 1st Edition

4. Learn Google Flutter Fast: 65 Example Apps

5. Android Developer Documentation, url: https://developer.android.com/docs/

6. Apple Developer Documentation, url: https://developer.apple.com/documentation/

7. Xcode Documentation, url: https://developer.apple.com/documentation/xcode/

8. Android Platform Guide, url: https://developer.android.com/guide/platform

9. OpenAI Platform Overview, url: https://platform.openai.com/docs/overview

10. YouTube Data API v3 Documentation, url: https://developers.google.com/youtube/v3/docs

11. Google Docs API Reference, url: https://developers.google.com/docs/api/reference/rest

12. Google Maps Documentation, url: https://developers.google.com/maps/documentation

13. Dart & Flutter Package Repository, url: https://pub.dev/