CHAPTER 1

INTRODUCTION

Welcome to Ai Chat App, an advanced platform designed to revolutionize the way users communicate through intelligent chat solutions. Ai Chat App harnesses the power of the Gemini API to offer an engaging and responsive chat experience, developed using Kotlin and Android Studio.

Ai Chat App is a cutting-edge platform designed to revolutionize user interactions through Ai driven chat solutions. The core objective of this App is to offer a seamless and intuitive interface, allowing users to engage with an Ai chatbot for a variety of purposes, including seeking assistance, retrieving information, and having casual conversations. Utilizing the capabilities of the Gemini API, this App delivers precise and contextually relevant responses, enhancing the overall user experience.

Ai Chat App provides developers and tech enthusiasts with a detailed and modular framework. The app's architecture ensures smooth integration and customization, making it an ideal tool for a variety of purposes.

1.1 PROJECT DESCRIPTION

AI Chat App is an innovative platform designed to enhance user communication through intelligent, Ai driven chat solutions. The primary goal of Ai Chat App is to provide a user-friendly interface where users can engage with an Ai chatbot for various purposes, such as assistance, information retrieval, and leisure conversations. By leveraging the Gemini API, Ai Chat App ensures accurate and context-aware responses, making interactions more meaningful and effective.

Key Features:

- Seamless Interaction: Users can easily communicate with the AI chatbot, which understands and responds to a wide range of queries.
- Responses: The AI provides precise and relevant answers, thanks to the advanced natural language processing capabilities of the Gemini API.
- User Friendly Interface: The app's design ensures a smooth and engaging user experience, making it accessible to a broad audience.

1.2 AIM OF THE PROJECT

The aim of the Ai Chat App project is to use advanced technology to improve how people interact with computers. The app makes it easy for users to talk with a smart computer program called a chatbot. By using the Gemini API, the app can give accurate and helpful responses to users, making conversations more natural and useful. The main goal is to make sure users enjoy talking with the chatbot and find it helpful for getting information quickly. The app is designed to grow and improve with new technology so it can keep helping more people in the future. Overall, this project wants to show how smart apps like this can change how we use technology every day, making it easier and more enjoyable for everyone.

1.3 PRODUCT SCOPE

The product scope of Ai Chat App includes a versatile digital platform aimed at facilitating seamless communication through AI-driven chat solutions. This project aims to develop a sophisticated platform centered around AI-powered chat functionalities. The app will enable users to engage with an intelligent chatbot for various interactions, such as obtaining information, receiving assistance, and enjoying casual conversations. Through integration with the Gemini API, the app ensures precise and contextually relevant responses, enhancing the overall user experience.

The platform is designed to be versatile and adaptable, supporting seamless integration into different digital environments. It offers developers a modular framework to customize and extend chatbot capabilities for diverse applications, including customer service, personal assistants, and interactive companions.

By emphasizing usability and functionality, the app aims to set new standards in user interaction with AI technology. It seeks to foster efficient and intuitive communication while promoting innovation in AI-driven solutions across industries and user demographics.

CHAPTER 2

SYSTEM REQUIREMENT AND SPECIFICATION

System requirement specification is a detailed statement of the effects that a system is required to achieve. A good specification gives a complete statement of what the system is to do, without any commitment as to how the system is to do it. It consists only externally observable behaviour and omits any design or implementation bias.

2.1 HARDWARE REQUIREMENTS

DEVICE : Android device supporting Android version 7.0 Nougat and above

PROCESSOR : Processors above 28 nm Architecture

RAM : Minimum 1 GB

INTERNET CONNECTION : Required for app functionality

2.2 SOFTWARE REQUIREMENTS

OPERATING SYSTEM : Android Operating System

DEVELOPMENT (IDE) : Android Studio

APPLICATION PROGRAMMING INTERFACE: Google Gemini API

2.2.1 ANDROID STUDIO:

Android Studio is the official integrated development environment (IDE) for Google's Android operating system. It is specifically designed for Android app development and offers a robust set of tools and features tailored to this purpose. Android Studio provides a user-friendly interface and supports the Kotlin programming language, among others, making it suitable for developing modern Android applications.

Key Features of Android Studio include:

- Kotlin Support: Android Studio fully supports Kotlin, a modern programming language that offers concise syntax and powerful features.

- Layout Editor: An intuitive drag-and-drop interface for designing app layouts, supporting both XML and visual editing.
- Performance Profiler: Tools for analysing app performance, identifying bottlenecks, and optimizing code.
- Code Templates: Built-in templates for common Android components, speeding up development.
- Gradle Integration: Built on the Gradle build system, Android Studio simplifies dependency management and build configuration.
- Emulator: A fast and feature-rich emulator to test apps on various Android device configurations.

Android Studio's integration with Google's Android SDK (Software Development Kit) and its continuous updates ensure developers have access to the latest tools and features for Android app development. It remains a preferred choice for developers aiming to create high-quality, efficient Android applications.

This IDE was used extensively in the development of the AI Chat App, leveraging its features to ensure compatibility, performance, and usability on Android devices.

2.2.2 KOTLIN:

Kotlin is a modern programming language developed by JetBrains, known for its versatility and interoperability with existing Java codebases. It is designed to be concise, expressive, and fully interoperable with Java, making it a preferred choice for Android app development.

Key Features of Kotlin:

- Conciseness: Kotlin reduces boilerplate code compared to Java, allowing developers to write cleaner and more readable code.
- Null Safety: Kotlin includes null safety features that help prevent null pointer exceptions, a common issue in many programming languages.
- Interoperability: Kotlin seamlessly integrates with Java, allowing developers to use existing Java libraries and frameworks within Kotlin projects.
- Tooling Support: Kotlin is supported by major IDEs such as Android Studio, providing robust tooling for development, debugging, and testing.

Kotlin has emerged as the preferred language for Android app development, prized for its modern features and seamless integration with Android Studio and the Android SDK. It enhances performance, readability, and maintainability, allowing developers to create robust and efficient Android applications more easily. By utilizing Kotlin in the development of our AI Chat App, we gain access to a contemporary and efficient programming language that not only boosts developer productivity but also facilitates effortless integration with Android platform features and libraries. This makes Kotlin an ideal choice for building sophisticated and user-friendly mobile applications tailored to meet modern development standards.

2.2.3 GEMINI API:

The Gemini API is a powerful tool for integrating advanced natural language processing (NLP) capabilities into applications. It provides developers with a comprehensive set of functionalities to enhance user interactions through intelligent responses and context-aware communication.

The key aspects of the Gemini API:

- Natural Language Processing: The Gemini API leverages NLP to analyze and understand user input, enabling the application to respond intelligently and contextually.
- Context-Aware Responses: It provides capabilities to maintain context across conversations, improving the relevance and accuracy of responses.
- Integration Flexibility: Developers can integrate Gemini API seamlessly into various platforms and programming languages, including Kotlin for Android development.
- Standardized Commands: The API supports standard commands and operations for managing conversational flows, such as handling user queries and generating appropriate responses.

By integrating the Gemini API into the AI Chat App, developers can enhance its functionality with advanced language understanding capabilities. This integration allows the app to deliver more personalized and effective user experiences, making interactions smoother and more natural.

This API plays a crucial role in enabling the app to handle complex queries, provide accurate information, and adapt to user preferences dynamically. It represents a significant advancement in enhancing the capabilities of AI-driven chat applications like yours.

CHAPTER 3

SYSTEM DESIGN

Systems design involves planning and defining the structure, components, interfaces, and data of a system to meet specific requirements. It applies principles from systems theory to ensure that the resulting system is well-organized, efficient, and capable of fulfilling its intended functions effectively.

3.1 BLOCK DIAGRAM

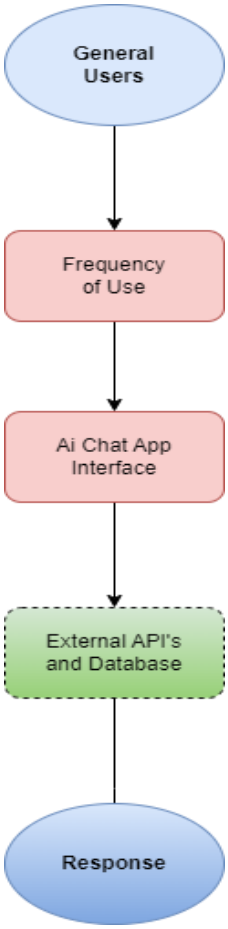

Fig: 3.1 Block diagram

The AI chatbot application's block diagram illustrates a structured interaction process designed to facilitate seamless communication and data retrieval. It begins with General Users initiating interactions with the system, which records and analyzes the Frequency of Use to understand user engagement patterns. This data is crucial as it feeds into the AI Chatbot App Interface, which serves as the primary interaction point for users.

The interface is equipped to handle various queries and requests, which it forwards to External APIs and Databases. These external resources are integral to the process, providing necessary information and enhancing the chatbot's ability to deliver accurate and relevant responses.

Once the required data is retrieved from these external sources, it is processed by the AI Chatbot App Interface, ensuring that the information is tailored to meet the user's needs. The refined and contextually appropriate Response is then delivered back to the General Users, completing the interaction loop. This comprehensive approach ensures that the chatbot not only responds efficiently but also continuously improves based on user interaction data. By leveraging external data sources, the system maintains high accuracy and relevance in its responses, thereby enhancing user satisfaction and engagement.

3.2 USE CASE DIAGRAM

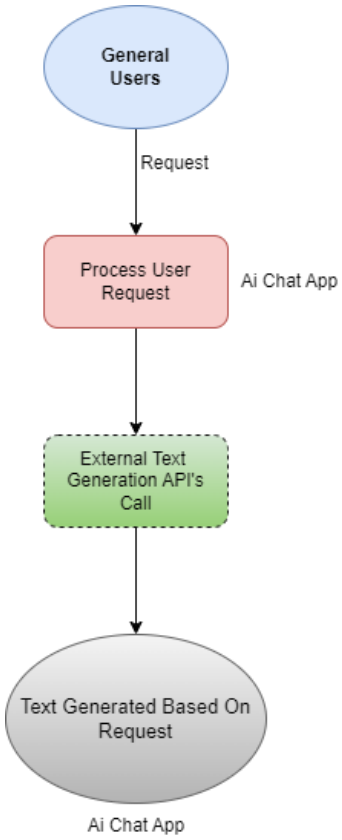

Fig: 3.2 Use Case Diagram

The use case diagram for the AI Chatbot App demonstrates how users interact with the system to generate text. The primary actors involved are general users and the AI Chatbot App itself. General users initiate the process by making a request to generate text. The AI Chatbot App processes this request and interacts with external text generation APIs to fulfil the user's request. Finally, the generated text is delivered back to the user. This streamlined interaction ensures that users can easily obtain text through a simple and efficient process facilitated by the AI Chatbot App.

After receiving an API response, you'll pass this information to Gemini as a function response, and then Gemini will either answer the original prompt by summarizing the results in the model response, or Gemini will output a subsequent function call to get more information for your query. Depending on the available functions and the original prompt, Gemini might return one or more function calls in sequence until it has gathered enough information to generate a summary of the results.

The seamless communication between the AI Chatbot App and external text generation APIs, along with robust text processing capabilities, guarantees high-quality text generation tailored to user requests. This process highlights the app's ability to efficiently handle user inputs, manage external API interactions, and deliver desired outputs, providing a user-friendly experience.

CHAPTER 4

IMPLEMENTATION

- Here we discuss about the codes that are used in the implementation.
- Android Studio served as the primary Integrated Development Environment (IDE) for developing the AI Chat App.

4.1 SETTING UP THE DEVELOPMENT ENVIRONMENT:

1. **Install Android Studio:**

    - Download and install Android Studio, set up the Android SDK, configure an Android Virtual Device (AVD), and start developing Android applications using Kotlin or Java.

2. **Set Up the Project Directory:**

    - Create a directory for Ai Chat App within the Android Studio Projects folder (e.g., D:\AndroidStudioProjects\Ai_ChatBot_App).

3. **Install and Set Up a Virtual Device:**

    - Install Android Emulator: Launch Android Studio and navigate to the AVD Manager. Click on "Create Virtual Device" to begin the setup process.
    - Select Device Configuration: Choose a device type, such as Pixel or Nexus, and select a system image with the desired Android version. Click "Next" to proceed with the installation.

4. **Sync Gradle:**
    - Click on the "Sync Project with Gradle Files" icon to initiate the synchronization process.
    - This action ensures that Android Studio updates and resolves dependencies specified in your project's Gradle files, enabling smooth integration of libraries and plugins essential for your AI Chat App development.

4.2 MAINACTIVITY.KT

```
package np.com.bimalkafle.aichatbotbot

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        val chatViewModel = ViewModelProvider(this)[ChatViewModel::class.java]
        setContent {{
            val drawerState = rememberDrawerState(DrawerValue.Closed)
            val scope = rememberCoroutineScope()
            ModalNavigationDrawer(
                drawerState = drawerState,
                drawerContent = {
                    DrawerContent(
                        onDismiss = {
                            scope.launch {
                                drawerState.close()
                            }
                        },
                    )
                },
            content = {
                    Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->ChatPage(
                    modifier = Modifier.padding(innerPadding),
```

```
                    viewModel = chatViewModel,
                     onMenuClick = {
                          scope.launch {
                              drawerState.open()  }
                       }
                    ) }
                }
            )  }
        } }
```

4.3 API-KEY INTEGRATION

```
package np.com.bimalkafle.aichatbotbot

object Constants {
    val apiKey = "YOUR_API_KEY"
}
```

4.4 CHATMODEL

```
package np.com.bimalkafle.aichatbotbot

class ChatViewModel : ViewModel() {
    val messageList by lazy {
        mutableStateListOf<MessageModel>()
    }
    val generativeModel : GenerativeModel = GenerativeModel(
        modelName = "gemini-pro",
        apiKey = Constants.apiKey
    )
```

```
fun sendMessage(question : String){
        viewModelScope.launch {
            try{
                        val chat = generativeModel.startChat(
                            history = messageList.map {
                                content(it.role){ text(it.message) }
                            }.toList()
                        )
                messageList.add(MessageModel(question,"user"))
                messageList.add(MessageModel("Typing....","model"))
                val response = chat.sendMessage(question)
                messageList.removeLast()
                messageList.add(MessageModel(response.text.toString(),"model"))
            }
catch (e : Exception){
                messageList.removeLast()
                messageList.add(MessageModel("Error : "+e.message.toString(),"model"))
            }
        }
    }
}
```

CHAPTER 5

USER MANUAL AND SNAPSHOTS

5.1 ANDROID STUDIO PAGE

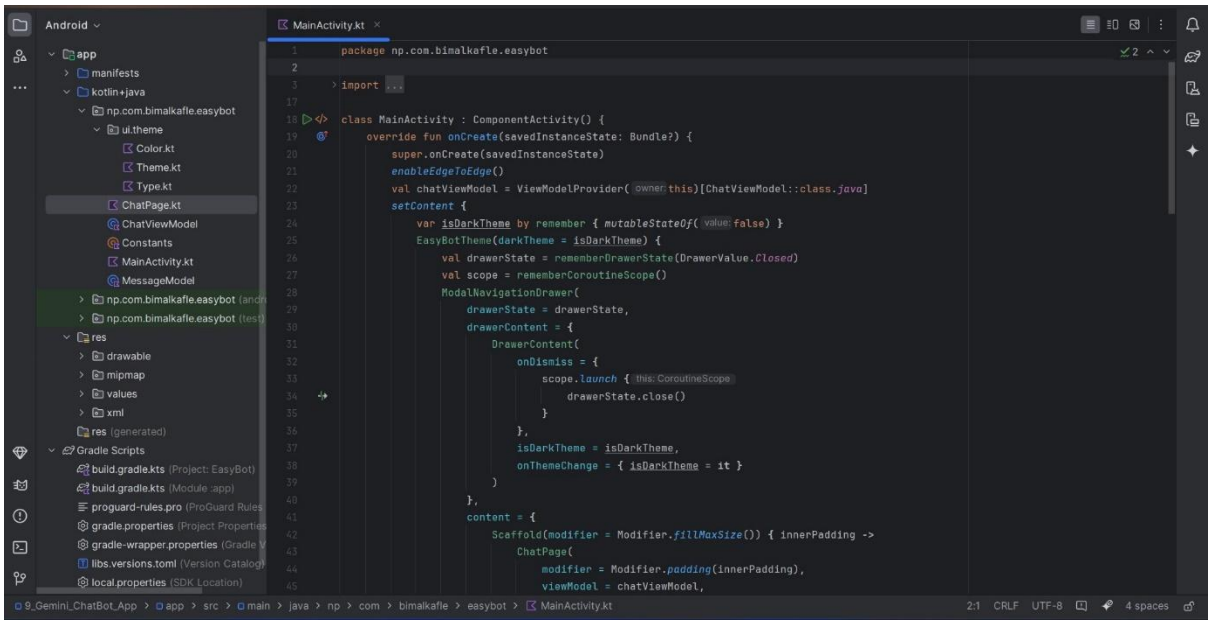

Fig: 5.1 Android Studio Page

5.2 ANDROID STUDIO EMULATOR

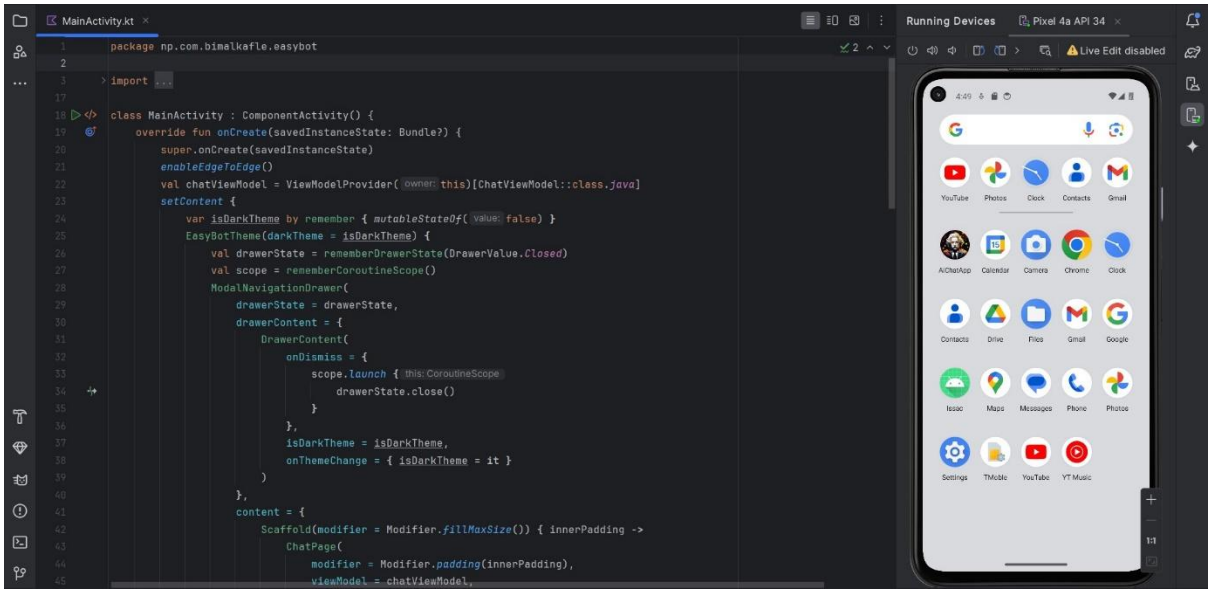

Fig: 5.2 Emulator Interface

5.3 APP HOME PAGE

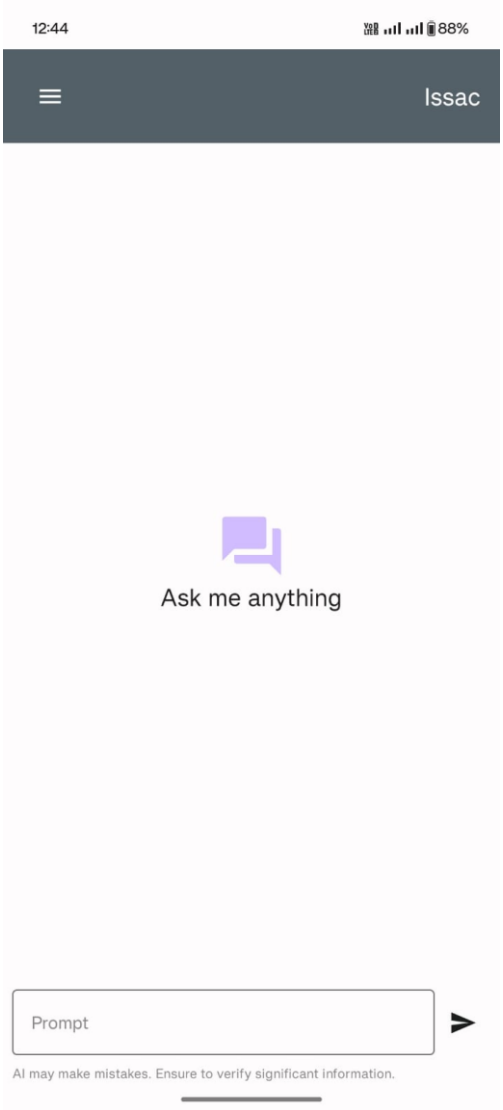

Fig: 5.3 App Home Page

5.4 APP MENU PAGE

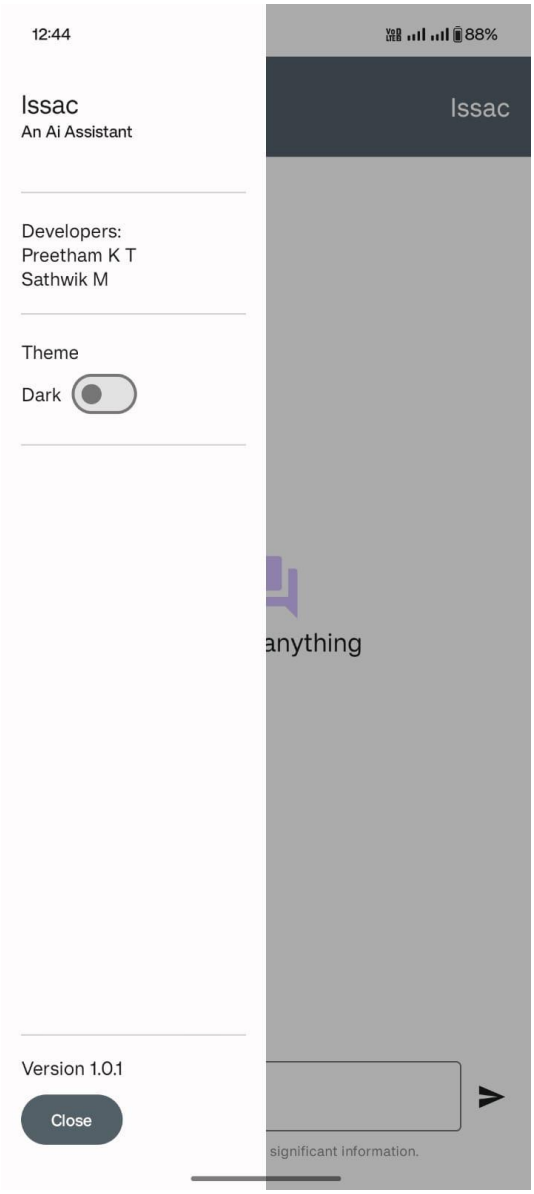

Fig: 5.4 App Menu Page

5.5 PROMPT INTERFACE

Fig: 5.5 Prompt Interface

5.6 TEXT GENERATION INTERFACE

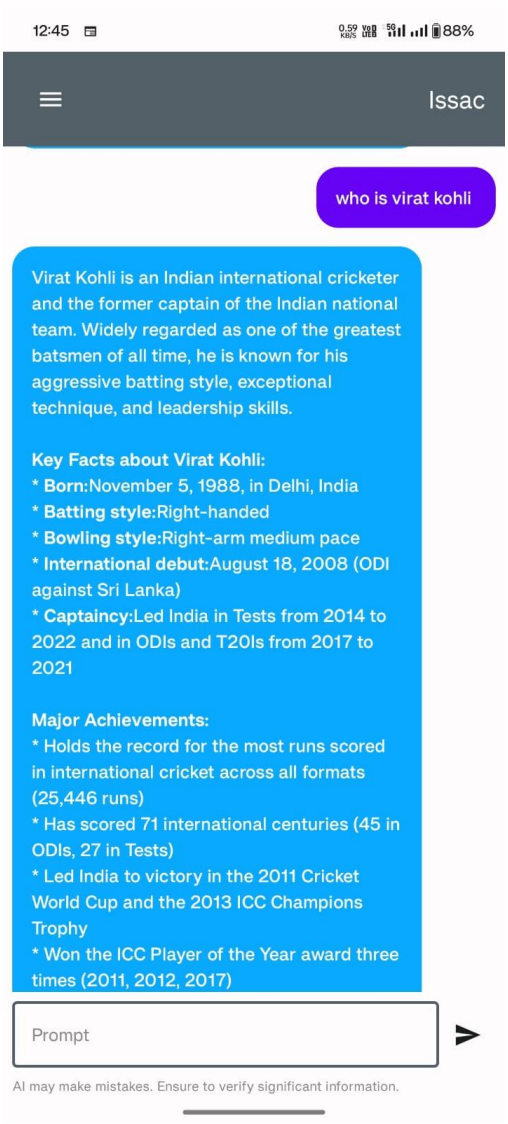

Fig: 5.6 Text Generation Interface

CHAPTER 6

RESULTS AND DISCUSSION

We built the "Issac" AI Chat App with a vision to simplify and enhance text generation for users. By implementing this application, we aim to provide users with an efficient and accessible tool for generating high-quality text based on their prompts, utilizing the robust Gemini API. Refer to Fig. 6.1 for a detailed view of the application's architecture and user interaction.

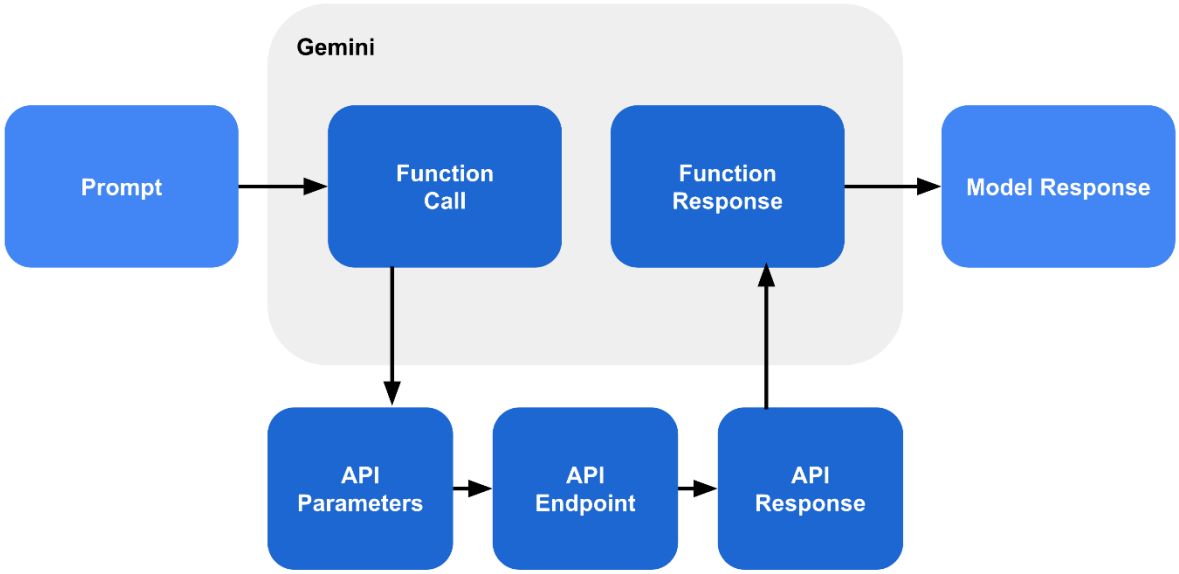

Fig: 6.1 API Architecture

After the implementation phase, we conducted extensive testing using a comprehensive testing algorithm that covers all user cases. This ensures that users do not miss any features and can use the app seamlessly without any confusion or inconvenience. Our rigorous testing process confirms that Issac handles user inputs effectively, manages external API interactions smoothly, and delivers accurate and relevant text outputs.

This focus on user experience and functionality underscores our commitment to creating a user-friendly AI Chat App that meets the needs of our users, providing them with a reliable and efficient tool for text generation.

6.2 TESTING THE RESULT

SL.NO	TEST INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	RESULT
1	Run the App on Android Phone	The App should Open without any error	Successfully runed the app and opened the home page	Pass
2	When click the App icon	That redirect to home page	Yes, when I click app icon that redirect to home page	Pass
3	In home page while clicked prompt interface	That allow you to provide prompt	It successfully takes the prompts	Pass
4	In home page while clicked menu icon	That provide a menu interface	Successfully allows to interact at home screen	Pass
5	When we prompt send icon	The app redirect to chat page	Yes, when click send icon that redirect to chat page	Pass
6	Generate text based on prompt	It should generate the text based on prompt	Successfully generating the text for the prompt	Pass

Fig: 6.2 Testing all kind of user case result

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

The "Issac" AI Chat App project demonstrates the potential of advanced technology to improve how we communicate with intelligent chat solutions. Developed using Kotlin and Android Studio, and powered by the Gemini API, Issac offers a seamless and engaging chat experience. Our main goal was to create a user-friendly app where users can interact with an AI chatbot for assistance, information retrieval, and casual conversations. By using the Gemini API, the app provides accurate and relevant responses, enhancing the overall user experience. The intuitive design and robust functionality ensure that Issac is accessible and enjoyable for everyone. This project highlights how smart apps like Issac can make technology easier and more enjoyable to use every day.

Future scope of the app is to keep users informed with the latest updates and data, we aim to enhance the app to offer real-time information. This will involve integrating live data sources and updating the information database continuously, ensuring that users receive the most current and relevant information. We also planned to implement user accounts linked to email addresses. This feature will allow users to create personal accounts, enabling them to maintain their recent conversations in a database. We intend to integrate an image generation tool within the app. This feature will enable users to create images from text prompts. This expansion of functionality will not only appeal to a broader audience but also provide users with creative and practical tools for their needs.

REFERENCE

1. Durall Gazulla, E., Martins, L., Fernández Ferrer, M. "Designing learning technology collaboratively: Analysis of a chatbot co-design." *International Journal of Research Publication and Reviews*, Published Date: 24 June 2022.
2. Raisian, K., Singh, P., Joshi, K. "Enhancement of ChatGPT using API Wrappers Techniques." *International Journal of Research Publication and Reviews*, Volume 34, Pages 82-86, June 2023.
3. Wiberg, M., Stolterman Bergqvist, E. "Automation of interaction design at the crossroads of user experience and AI." *Springer's Journal of Science*.
4. Koplin, J. J. "Dual use implications of AI text generation." *International Journal of Research Publication and Reviews*, Published Date: 29 May 2023.
5. "Building an AI-powered BigQuery Data Exploration App using Function Calling in Gemini." Available at: https://shorturl.at/Px2Wu
6. "Function calling: A native framework to connect Gemini to external systems, data, and APIs." Available at: https://shorturl.at/qZy30