AUK Admission Enquiry Chatbot

A Capstone Project Submitted to the Faculty of the Computer Science and Information Systems ProgramCollege of Engineering and Applied Sciences American University of Kuwait

In Partial Fulfillment of the Requirements for the degreeBachelor of Science in [Information Science]

AUK Admission ChatBot

By

Yasmeen Alsaad s00050398

Sarah alajmi s00051687

Advisor

[Dr. Marwa Mostafa]

Course Coordinator

[Dr. Marwa Mostafa]

30th of November, 20Fall 2024

Contents

Table of Figures

Abstract

The research provides a helpful tool for the admission and registration department in the college. This tool should help the students to know more about the admission process and the registration requirements. The tool is mobile application chatbot that can respond to the students' enquiries and questions. The chatbot app allow the students to ask the application using a voice note or text input, then the system can respond with the required data. The chatbot app also provides the student with more features like program suggestions and recommendations depending on some criteria provided by the end user. By applying this application, the student can have a better user experience and explore the admission and registration services easily and effectively. The application also can save the time and effort of the admission staff.

Chapter1: Introduction

As a result, to the rapid advancement in technology and the evolution concepts about the chatbots, it become more familiar to be integrated in the different fields like the businesses, education, or governments departments. The chatbots can interact with users and respond to enquiries in different output types. The output could be text, voice, images, or combination of the previous outputs.

One of the interesting fields that chatbots can work effectively is the education and learning field [1]. So, the chatbots can be deployed in schools or universities to help the students with their questions and enquiries. This will facilitate the process of accessing information related to students' inquiries towards admissions.

In this research, we are proposing chatbot android application for the admission enquiries. It can interact with prospective students or applicants who are interested in applying to a university or college especially for international students. The primary purpose of such a chatbot is to provide information and assistance to potential students regarding the admission process, programs offered, eligibility criteria, application deadlines, and any other relevant information related to the university's admissions.

Project Objectives:

The application should achieve the following objectives in order to cover the features and functionalities provided by the AUK Admission department:

- Information Provision: The chatbot can provide detailed information about the university, its academic programs, faculty, campus facilities, and admission requirements.
- Application Guidance: It can guide applicants through the application process, helping them understand what documents are needed, how to fill out forms, and where to submit their applications.
- Eligibility Check: The chatbot can assess an applicant's eligibility based on their academic background, test scores, and other relevant criteria.
- Deadlines and Notifications: It can remind applicants of important deadlines for submitting applications, paying fees, and providing necessary documents.

- Program Recommendations: The chatbot can suggest academic programs or majors that align with the applicant's interests and qualifications.
- FAQ Handling: It can answer frequently asked questions related to admissions, financial aid, scholarships, and campus life.

Significance of the Project

The proposed chatbot Android application for admission enquiries holds significant value in the context of enhancing user experience and improving the efficiency of the admission process in educational institutions, particularly in the case of the admission department.

The introduction of a chatbot in the admission and registration process introduces a user-friendly interface for prospective students. By allowing voice and text inputs, the application caters to diverse user preferences, making information retrieval more accessible and convenient.

The chatbot serves as a valuable tool for providing comprehensive information about the university, academic programs, faculty, and campus facilities.

The chatbot acts as a guide for applicants, assisting them in navigating through the intricacies of the admission process.

The application's capability to assess eligibility in real-time based on academic backgrounds and test scores adds efficiency to the admission process.

This feature assists applicants in determining their suitability for specific programs.

Through the analysis of user interests and qualifications, the chatbot offers personalized program recommendations.

This feature aids applicants in making informed decisions aligned with their academic and career aspirations.

The chatbot addresses frequently asked questions related to admissions, financial aid, scholarships, and campus life. This functionality minimizes the need for human intervention in handling routine queries, allowing admission staff to focus on more complex tasks.

Chapter 2: Review of Related Literature

Theoretical Background

Chatbots:

A chatbot is a computer program that simulates human conversations or chat through AI. Artificial Intelligence (AI) increasingly Integrates our daily lives with the creation and analysis of intelligent software and hardware [2]. chatbot do a variety of tasks and responding in different types of outputs. A chatbots a typical example of an AI system and one of the most elementary and widespread examples of intelligent Human-Computer Interaction. It is a computer program, which responds like a smart entity when conversed with through text or voice and understands one or more human languages by Natural Language Processing. Chatbots are developed and became so popular due to the increased use of smart devices and IoT technology.

Types of chatbots:

Based on [3] chatbots are categorized based on their primary objectives, with distinct functionalities and tasks associated with each classification:

1. **Informative Bots:**
    - Objective: Designed to provide users with pre-stored or fixed-source information.
    - Characteristics: Typically reliant on information retrieval algorithms, these bots fetch data from databases or perform string matching. Commonly referencing static sources like a site's FAQ page or an inventory database, they excel at tasks such as string matching, and response generation.
    - Example: FAQ bots.

2. **Chat-based/Conversational Bots:**
    - Objective: Mimic human interaction by responding to user input.
    - Characteristics: Engineered to engage in dynamic conversations with users, employing techniques such questioning-answering, information retrieval, string matching, and relevance detection.
    - Examples: Siri, Alexa.

3. **Task-based Bots:**
    - Objective: Execute specific tasks like booking a flight or assisting in online shopping.
    - Characteristics: Predetermined actions are defined for task execution, encompassing the flow of events and exceptions. These bots deploy intelligence in seeking information and analyzing user inputs.
    - Examples: Restaurant booking bots.

Android:

Android is the name of the mobile operating system owned by Google [4]. It most commonly comes installed on a variety of smartphones and tablets from a host of manufacturers offering users access to Google's own services like Search, YouTube, Maps, Gmail and more. There are multiple stores to publish applications on. The official place for publishing and downloading the applications is google play store while there are a lot of places that work almost the same work of google play store. In order to develop android application, we need to use one of the environments that allow us to produce android APK file. Android Studio is Android's official IDE. It is purpose built for Android to accelerate your development and help you build the highest-quality apps for every Android device. All updates by google on android will be present by android studio so there is no need to track the new updates and take care a lot about them. Android Studio provides a suitable environment where you can build apps for Android phones, tablets, Android Wear, Android TV, and Android Auto.

Related projects

Staffordshire university [5] introduced a chatbot called Beacon. Beacon developed to be used as a digital coach for students. It allows the students to interact using voice or text input. It deals mainly with the most frequent questions.

Also, st john's university [6] provides a chatbot called Johnny Chat. It is a web-based service that can interact with the students and reply to their requests. It provides college links for the students so they can explore the needed subjects.

Issues with Prior Projects

Chatbots depends on text-based interactions and does not support voice note inputs. Voice interactions can enhance accessibility for users who prefer verbal communication.

We noticed that most chatbots don't provide additional features like program suggestions and recommendations based on user criteria. This feature can help the students to select the best matching major.

Also, the previous projects doesn't assess an applicant's eligibility based on academic backgrounds and other criteria. This might be a limitation in assisting prospective students in determining their suitability for specific programs.

Chapter 3: Technical Background

Technicality of the project

The users can register for a new account in the application. The student can login to the app using the credentials. The students can have 2 main options to choose from. Recording a voice note, or providing a text input. +In case the student provided a voice note, the app applies speech-to-text conversion process. Then the converted text should be analyzed and processed to get the keywords. depending on the extracted keywords, the system either retrieves the data directly or provides the user with more options to choose from. The application then displays the results to the end user in a user-friendly interface. If the student provided a text, then the same process is applied. In addition to replying to student enquiries, the chatbot also gives the students to build a profile where they can provide the system with their academic background, test scores, interests and qualifications and other relevant criteria. Using these criteria, the chatbot can suggest academic programs or majors that align with the applicant's interests and qualifications. It also can assess an applicant's eligibility based on their academic background, test scores.

Details of the technologies to be used.

Software Applications:

Operating systems:

Windows: The development environment will be based on the Windows operating system. Windows provides a familiar and stable environment for software development, offering a wide range of development tools, libraries, and resources. Developers can leverage Windows' compatibility with various software packages and hardware configurations to create and test Android applications efficiently.

Programming languages

Java: Official programming language for android development [7]
Certainly! Java is the primary programming language used for Android development, and it offers several advantages for building mobile applications:

1. Official Support: Java is the official programming language for Android development, as designated by Google. This means it is well-supported by the Android SDK (Software Development Kit) and other development tools provided by Google.

2. Large Developer Community: Java has been around for decades and has a vast developer community. This means there are numerous resources, tutorials, forums, and libraries available for Android developers using Java.

3. Platform Independence: Java applications are designed to be platform independent. Android's Java implementation allows developers to write code once and run it on any device that supports Android, regardless of hardware architecture.

4. Object-Oriented Programming (OOP): Java is an object-oriented programming language, which aligns well with Android's application structure. OOP principles such as encapsulation, inheritance, and polymorphism facilitate building modular and maintainable code.

5. Memory Management: Java features automatic memory management through garbage collection. This helps developers avoid memory leaks and ensures efficient memory usage in Android applications.

6. Rich Ecosystem: Java has a rich ecosystem of libraries and frameworks that streamline Android development. For example, libraries like Retrofit for network requests, Gson for JSON parsing, and ButterKnife for view binding simplify common tasks in Android development.

7. Performance: While Java's performance may not match that of lower-level languages like C or C++, it still offers good performance for most Android applications. Additionally, optimizations provided by the Android runtime (ART) further improve Java's performance on the Android platform.

8. Compatibility: Java is compatible with older Android versions, allowing developers to target a broad range of devices. It also ensures backward compatibility with existing Android applications, reducing the risk of compatibility issues when updating apps.

Overall, Java's official status, extensive developer community, platform independence, and rich ecosystem make it a suitable choice for building robust and scalable Android applications, including university admission applications.

Database

Firebase Database

Firebase Database is a cloud-hosted NoSQL database provided by Google as part of the Firebase platform [8]. It offers real-time data synchronization and offline support, making it well-suited for mobile and web applications, including Android apps like the university admission application. More detailed explanation of Firebase Database:

1. Real-time Data Synchronization: Firebase Database synchronizes data across devices in real-time. When data changes on one device, it is automatically updated on all connected devices without the need for manual refreshes. This feature is particularly useful for collaborative applications or applications where multiple users need to access and modify the same data simultaneously.

2. NoSQL Database: Firebase Database is a NoSQL database, which means it does not require a fixed schema and allows for flexible data structures. This makes it easy to adapt to changing data requirements and iterate on application features without extensive database migrations.

3. JSON-based Data Model: Data in Firebase Database is organized as JSON (JavaScript Object Notation) documents, which are easy to read, write, and manipulate. JSON's hierarchical structure allows for nested data and relationships between different entities, enabling developers to model complex data structures efficiently.

4. Offline Support: Firebase Database provides offline support through local caching. When a device goes offline, the application can continue to read and write data locally. Once the device reconnects to the internet, Firebase automatically synchronizes the local changes with the server, ensuring data consistency across devices.

5. Security Rules: Firebase Database allows developers to define security rules that govern access to data. These rules can be customized to restrict access based on user authentication, IP addresses, or specific data conditions, ensuring that only authorized users can read or write to the database.

6. Scalability: Firebase Database is built on Google's infrastructure, which offers high scalability and reliability. It can handle a large volume of concurrent connections and data transactions, making it suitable for applications with growing user bases.

7. Integration with Other Firebase Services: Firebase Database seamlessly integrates with other Firebase services, such as Firebase Authentication, Firebase Storage, and Firebase Cloud Messaging. This allows developers to build end-to-end solutions with features like user authentication, cloud storage, and push notifications, all within the Firebase ecosystem.

Overall, Firebase Database provides a powerful and flexible solution for storing and synchronizing application data in real-time, making it an excellent choice for building modern, collaborative Android applications like the university admission application.

Authentication

Firebase Authentication

Firebase Authentication is a service provided by Google as part of the Firebase platform, designed to handle user authentication for mobile and web applications. It offers a secure and easy-to-implement authentication solution for Android apps, including the university admission application. More detailed explanation of Firebase Authentication:

1. User Authentication Methods: Firebase Authentication supports various authentication methods, including:

- Email/Password: Users can sign up and sign in using their email addresses and passwords. Firebase securely manages user credentials and handles password hashing to protect sensitive information.

- Phone Number: Users can verify their phone numbers via SMS for authentication. Firebase Authentication handles the verification process and provides a seamless experience for users.

- Social Authentication: Users can sign in using third-party providers such as Google, Facebook, Twitter, GitHub, and others. Firebase Authentication integrates with these providers to authenticate users securely without requiring additional passwords.

- Custom Authentication: Developers can implement custom authentication systems and integrate them with Firebase Authentication. This allows for flexibility in supporting different authentication methods or integrating with existing authentication systems.

2. Security: Firebase Authentication provides built-in security features to protect user accounts and sensitive data:

- Secure Authentication Tokens: Firebase issues authentication tokens to verified users, which are securely signed and encrypted. These tokens are used to authenticate users and authorize access to resources.

- OAuth 2.0 and OpenID Connect: Firebase Authentication follows industry standards for secure authentication, ensuring compatibility with OAuth 2.0 and OpenID Connect specifications.

- Security Rules: Firebase Authentication integrates with Firebase Realtime Database and Firebase Storage security rules, allowing developers to enforce access controls based on user authentication status, user roles, or specific data conditions.

3. Integration with Firebase Services: Firebase Authentication seamlessly integrates with other Firebase services, enabling developers to build end-to-end solutions with features like cloud storage, real-time database, and push notifications. For example:

- Firebase Realtime Database: Developers can define security rules that restrict access to data based on the user's authentication status or specific user attributes.

- Firebase Cloud Firestore: Firebase Authentication can be used to control access to Firestore documents and collections based on user authentication and authorization rules.

- Firebase Storage: Developers can secure access to storage buckets based on user authentication and authorization, ensuring that only authorized users can upload or download files.

4. Identity Providers: Firebase Authentication supports a wide range of identity providers, allowing users to sign in with accounts they already use and trust. This simplifies the sign-in process for users and improves user acquisition and retention for the application.

Overall, Firebase Authentication provides a secure and reliable authentication solution for Android applications like the university admission app, allowing developers to focus on building great user experiences without worrying about the complexities of user authentication and security.

Storage:

Firebase Storage

Firebase Storage is a cloud storage solution provided by Google as part of the Firebase platform. It offers secure and scalable storage for user-generated content, such as application documents, images, videos, and other media files. Here's a more detailed explanation of Firebase Storage:

1. Cloud-based Storage: Firebase Storage provides a cloud-based storage solution, allowing developers to store and serve user-generated content from Google's infrastructure. This eliminates the need for managing physical storage hardware and provides high availability and reliability for storing application data.

2. Scalability: Firebase Storage scales automatically to accommodate growing amounts of data and user traffic. It can handle large volumes of media files, making it suitable for applications with a high number of users or large amounts of user-generated content.

3. Integration with Firebase: Firebase Storage seamlessly integrates with other Firebase services, such as Firebase Authentication and Firebase Realtime Database. This allows developers to build end-to-end solutions with features like user authentication, real-time data synchronization, and cloud storage within the Firebase ecosystem.

4. Secure Storage: Firebase Storage ensures data security through various measures:

- Access Control: Developers can define access rules to restrict access to storage buckets based on user authentication, user roles, or specific conditions. This ensures that only authorized users can upload, download, or modify files.

- HTTPS Encryption: All data transfers between the client and Firebase Storage are encrypted using HTTPS, ensuring data privacy and integrity.

- Firebase Authentication Integration: Firebase Storage integrates seamlessly with Firebase Authentication, allowing developers to control access to storage based on user authentication status or specific user attributes.

5. File Metadata: Firebase Storage allows developers to store additional metadata along with files, such as file names, file types, timestamps, and custom metadata. This metadata can be used to organize and manage files efficiently and provide additional context for application features.

6. Offline Access: Firebase Storage provides offline support through local caching. Users can access and interact with stored files even when offline, with changes automatically synchronized with the cloud when connectivity is restored.

7. Direct File Uploads: Firebase Storage supports direct uploads from clients, allowing users to upload files directly from their devices to Firebase Storage without routing through a server. This reduces server load and improves upload performance for users.

8. CDN Integration: Firebase Storage integrates with Google Cloud CDN (Content Delivery Network), allowing developers to serve stored files with low latency and high reliability to users around the world.

Overall, Firebase Storage offers a reliable, scalable, and secure cloud storage solution for Android applications like the university admission app, enabling developers to store and serve user-generated content efficiently while focusing on building great user experiences.

Natural Language Processing (NLP)

In a university admission application, Natural Language Processing (NLP) can enhance the user experience by enabling the application to understand and process natural language inputs from applicants [9]. Here's how NLP could be implemented in such an application:

1. Application Form Interaction:

- Intent Recognition: NLP can be used to understand the intent behind applicant queries or responses. For example, an applicant might ask "What documents are required for admission?" NLP can recognize the intent to inquire about admission requirements and provide relevant information.
- Entity Extraction: NLP can extract key information from applicant responses. For instance, when an applicant submits their academic history, NLP can extract entities such as GPA, test scores, and course names.

2. Virtual Assistants and Chatbots:

- FAQs and Guidance: NLP-powered chatbots or virtual assistants can answer frequently asked questions about the admission process, deadlines, required documents, and program details.
- Application Assistance: Applicants can interact with the chatbot to receive guidance on completing the application form, uploading documents, and understanding specific requirements.

3. Document Analysis:

- Transcript Evaluation: NLP can analyze transcripts and academic documents to extract relevant information such as courses, grades, and credits. This information can be used for preliminary evaluation or to suggest courses of action to the applicant.
- Recommendation Letters: NLP can extract key information from recommendation letters, such as the recommender's credentials, the applicant's strengths, and specific achievements.

4. Personal Statement Analysis:

- Content Analysis: NLP techniques like sentiment analysis can evaluate the tone and sentiment of a personal statement, helping admissions officers assess the applicant's passion, personality, and fit for the program.
- Keyword Extraction: NLP can extract keywords and topics from personal statements, allowing admissions officers to quickly identify relevant themes and evaluate the applicant's interests and experiences.

5. Admissions Decision Support:

- Profile Matching: NLP can compare applicant profiles with program requirements and criteria to identify suitable matches. For example, NLP can analyze an applicant's academic background, interests, and experiences to determine their fit for specific programs.
- Risk Assessment: NLP can help identify potential risks or red flags in applications, such as inconsistencies in academic records or gaps in employment history.

6. Language Proficiency Evaluation:

- Language Assessment: NLP can evaluate an applicant's language proficiency based on written responses, personal statements, or test scores. This assessment can help determine if additional language support is needed for international applicants.

7. Feedback and Communication:

- Feedback Analysis: NLP can analyze applicant feedback and communication to identify common issues, concerns, or areas for improvement in the admission process. This

feedback can be used to refine application procedures and enhance the applicant experience.

Overall, integrating NLP into a university admission application can streamline the admission process, improve communication with applicants, and provide valuable insights for admissions officers, ultimately leading to a more efficient and effective admissions process.

Dialogflow

Dialog flow is a powerful natural language understanding (NLU) platform provided by Google Cloud for building conversational interfaces, including chatbots, voice assistants, and interactive voice response (IVR) systems [10]. Here's a more detailed explanation of Dialog flow:

1. **Natural Language Understanding (NLU):** Dialog flow uses advanced natural language processing techniques to understand and interpret user input in natural language. It can parse and analyze text or voice inputs to extract intents, entities, and context, allowing developers to build conversational experiences that understand user queries and respond appropriately.

2. Intent Detection: Dialog flow allows developers to define intents, which represent the actions or tasks users want to perform. Intents are mapped to specific user queries or phrases, and Dialog flow’s machine learning algorithms analyze incoming messages to match them to the appropriate intents.

3.Entity Recognition: Entities are parameters within user queries that provide additional context or information. Dialog flow can recognize predefined entities such as dates, locations, and numbers, as well as custom entities defined by developers. This allows for more precise understanding of user input and enables richer conversational interactions.

4. Context Management: Dialog flow maintains conversation context to understand the flow of conversation and maintain state between interactions. Contexts help Dialog flow remember previous user inputs, allowing for more natural and coherent conversations over time.

5. Integration with Messaging Platforms: Dialog flow integrates seamlessly with various messaging platforms, including Google Assistant, Facebook Messenger, Slack, and more. This allows developers to deploy conversational agents across multiple channels without significant additional effort.

6. Multi-language Support: Dialog flow supports multiple languages, allowing developers to build conversational experiences in languages other than English. It provides built-in language support for over 20 languages, making it accessible to a global audience.

7. Rich Responses: Dialog flow enables developers to generate rich responses, including text, images, cards, buttons, and even interactive elements like carousels and quick replies. This allows for engaging and interactive conversational experiences that go beyond simple text responses.

8. Integration with Other Google Cloud Services: Dialog flow integrates with other Google Cloud services, such as Firebase, Google Cloud Functions, and Google Cloud Speech-to-Text, allowing developers to leverage additional capabilities for building powerful and scalable conversational applications.

9. Analytics and Insights: Dialog flow provides analytics and insights into user interactions, allowing developers to track usage metrics, identify trends, and optimize conversational experiences over time.

Overall, Dialog flow empowers developers to create natural and intuitive conversational interfaces for Android applications like the university admission app. By leveraging Dialog flow's capabilities, developers can provide users with a seamless and interactive

experience, allowing them to ask questions, receive assistance, and complete tasks using natural language. Dialog flow also can be implemented in different teaching and learning fields. [11] describes how they developed a prototype system that utilizes Dialogflow's natural language processing capabilities to facilitate interactions between teachers and the system. This likely includes features such as data collection, analysis, and reporting functionalities tailored to the needs of classroom research.

Applications: Android Studio: the official IDE platform for android development

Android Studio is the official integrated development environment (IDE) for Android app development. It provides a comprehensive set of tools and features tailored specifically for building Android applications. Here's a more detailed explanation of Android Studio:

1. User Interface: Android Studio offers a user-friendly and intuitive interface designed to streamline the app development process. It provides easy access to various tools and features, allowing developers to focus on writing code and designing user interfaces.

2. Code Editor: Android Studio includes a powerful code editor with features such as syntax highlighting, code completion, and refactoring tools. It supports multiple programming languages, including Java and Kotlin, the two official languages for Android development.

3. Layout Editor: Android Studio's layout editor allows developers to visually design user interfaces for their apps. It provides drag-and-drop functionality for adding UI components, as well as tools for arranging and customizing layouts.

4. Resource Manager: Android Studio includes a resource manager for managing app resources such as images, strings, and colors. It provides a centralized location for organizing and accessing resources, making it easier to maintain consistency across the app.

5. Emulator: Android Studio includes a built-in emulator that allows developers to test their apps on virtual Android devices. The emulator supports various device configurations and Android versions, allowing developers to simulate real-world conditions and ensure compatibility with different devices.

6. Debugger: Android Studio's debugger allows developers to debug their apps and analyze runtime behavior. It provides features such as breakpoints, variable inspection, and call stack navigation, helping developers identify and fix issues in their code.

7. Performance Profiler: Android Studio includes performance profiling tools for monitoring and optimizing app performance. Developers can analyze CPU, memory, and network usage, identify performance bottlenecks, and optimize their apps for better efficiency and responsiveness.

8. Version Control Integration: Android Studio seamlessly integrates with version control systems such as Git, allowing developers to manage and collaborate on their projects more effectively. It provides features for committing, branching, merging, and resolving conflicts directly within the IDE.

9. Build System: Android Studio uses Gradle as its build system, which automates the process of compiling, packaging, and deploying Android apps. Gradle allows developers to customize their build configurations and manage dependencies efficiently.

10. Support for Kotlin: Android Studio fully supports Kotlin, a modern programming language that offers concise syntax, null safety, and interoperability with existing Java code. Developers can write Android apps entirely in Kotlin or mix Kotlin and Java code within the same project.

Overall, Android Studio provides a comprehensive development environment for building high-quality Android applications. Its rich set of features, including code editing, layout

design, debugging, and performance profiling, enables developers to create powerful and user-friendly apps efficiently.

Chapter 4: Methodology

Feasibility study:

a) Operational:

The application is designed to provide the suitable solutions for the problems mentioned in an early part of this paper. In addition, we didn't forget to take in consideration running application after development and deployment on the google play store. Operational problems that we may face after developing the application are performance, control, and services

Fishbone Diagram:

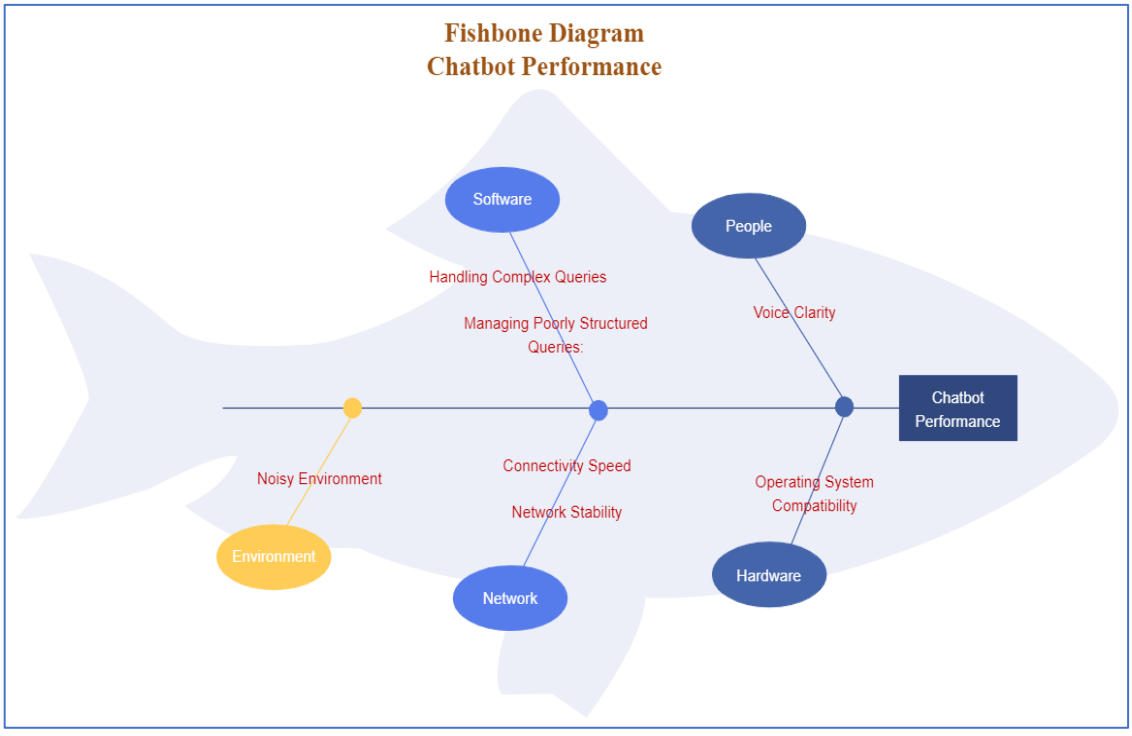


FIGURE 1: FISHBONE DIAGRAM

Functional Decomposition Diagram

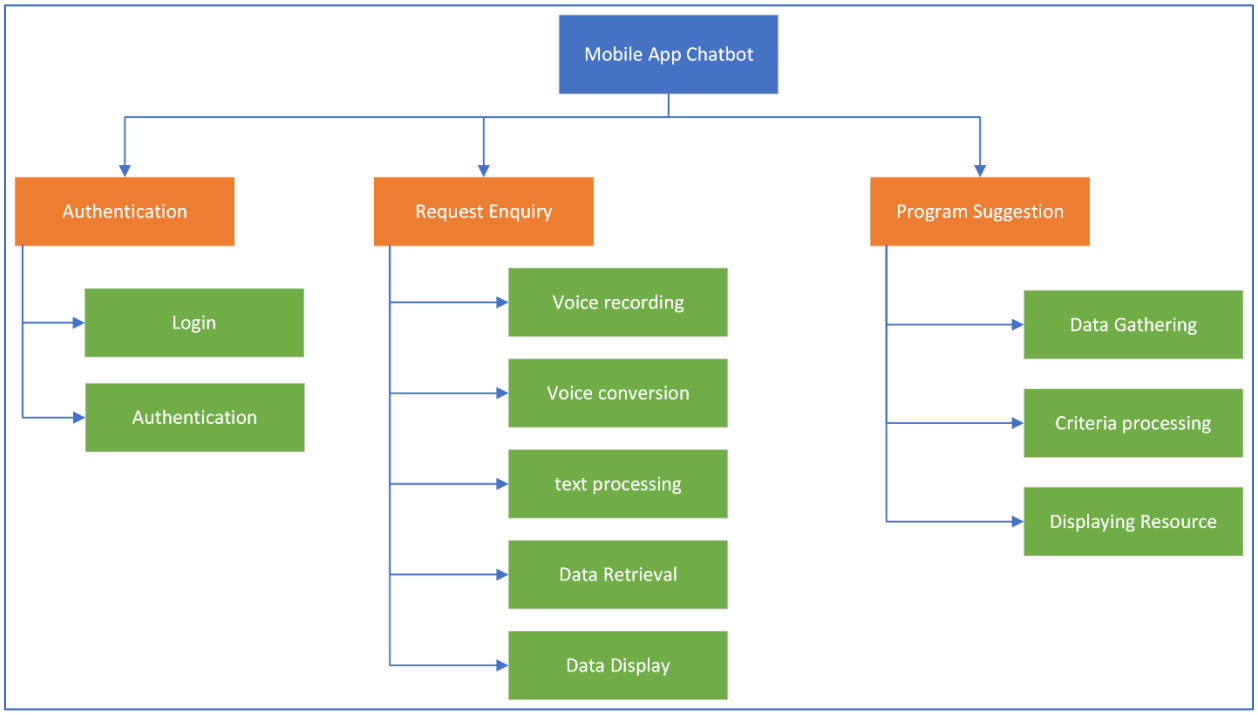


FIGURE 2: FUNCTIONAL DECOMPOSITION DIAGRAM

b) Technical:

We are going to develop the application using android technology, android technology supports enterprise and huge types of mobile applications for organizations beginning from small to large size of business. Android applications are developed using Java programming language and connected to Firebase database to save the whole system data inside. The official environment for developing android applications is android studio application which using Java language for programming. Android applications are reliable and give high security for the users. The database used for saving the application data is Realtime Firebase database. The resources for android learning and development are widely available for developers. Technical risk by using android technology is very low. The users of android application need to have android smartphone to run the application.

c) Economic:

The expected expenses begin with the development environment, which will be the personal Mac or windows computers and will cost around 400 KD but while we already have our laptops then it will cost nothing implicitly. The registration at Google play store, which is 25 USD paid once, In case of using Firebase database, then it charges 0.06 USD per verification if the number users exceed 10,000 users per month. For the storage, it will cost 5 USD for each GB after exceeding 1 GB per month. For NLP, while we are using Dialog Flow, then the pricing for the text analyzing is $0.007 per request. Then we may pay for marketing the idea and on the research, which is estimated by 200 KWD and may be increased since we are planning to use the social media to market for the application. if the college decided to use AUK database system, then the cost of the hosting and authentication will be near to zero.

For application revenues, of course the direct application revenue will be 0 KD while the application download from the store will be free of charge, there are no in-app purchases, and there are no ads included. But the revenue of the application is indirect. The app helps the college to attract more students to the college and gives an interesting user experience.

Requirements Modeling

Input:

- User queries submitted via text or voice note.
- Voice notes are processed using the Recognizer Intent class in the Android SDK to convert speech to text.
- Text queries are directly inputted by the user.
- Input includes user authentication data (email, password) for registration and login via Firebase AUTH.

Process:

- User authentication process using Firebase AUTH for registration and login.
- Voice notes converted to text using Recognizer Intent class.
- Text queries directly sent to Dialog Flow ES for natural language processing.
- Dialog Flow ES processes the queries, extracts intent and parameters, and returns results.
- Results from Dialog Flow ES used to query Firebase database for relevant information.
- Data retrieved from Firebase database is formatted and displayed to the users via the chatbot interface.

Output:

- Responses generated by the chatbot based on the user queries.
- Displayed information retrieved from Firebase database, such as answers to inquiries or relevant data.
- Authentication status messages for registration and login processes via Firebase AUTH.

Control:

- Utilization of Firebase AUTH for secure user authentication and access control.
- Proper handling and validation of user input to ensure data integrity and prevent errors.
- Implementation of error handling mechanisms to address any issues that arise during authentication, query processing, or database retrieval.

Data and Process Modelling

Context Diagram:

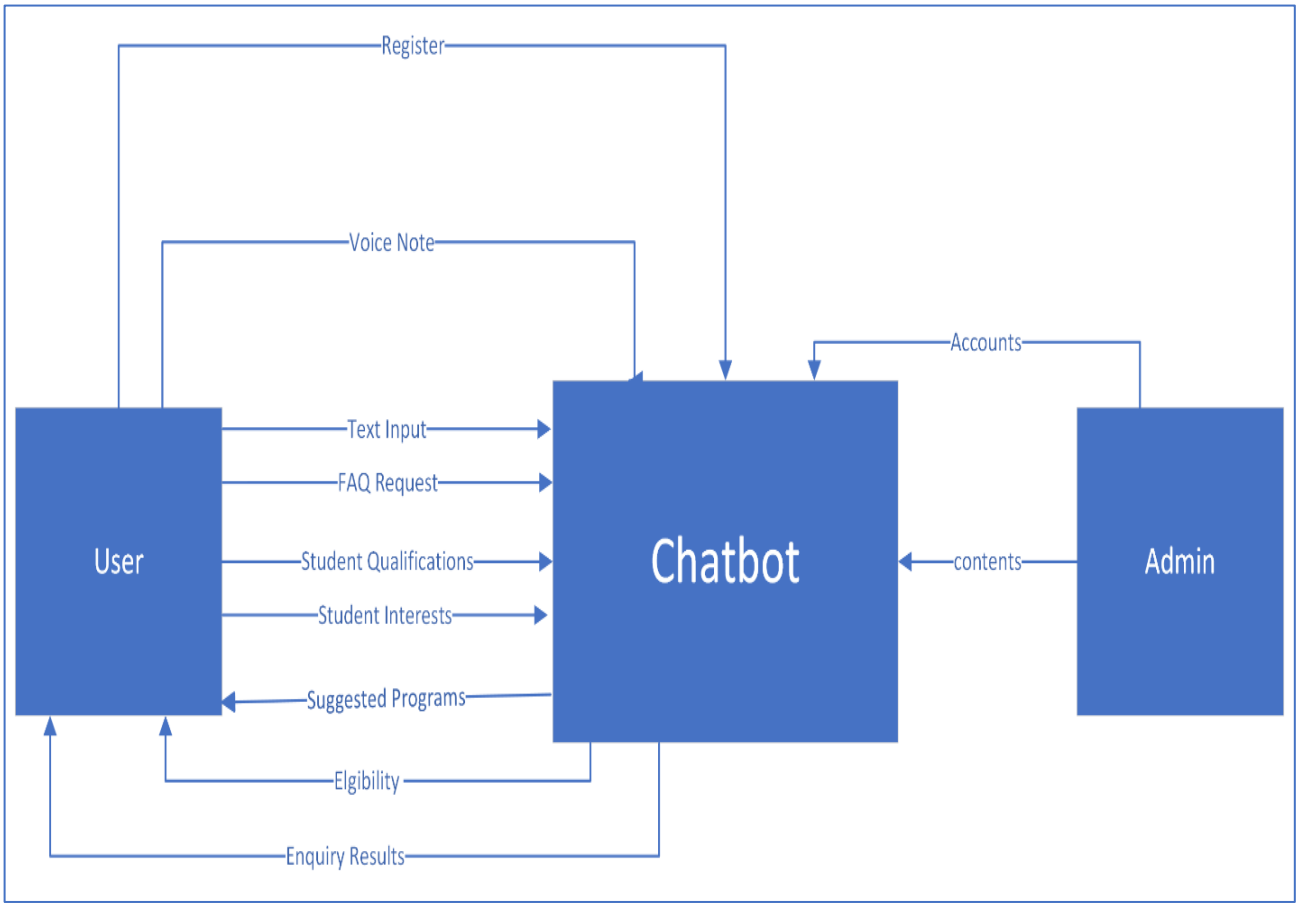


FIGURE 3: CONTEXT DIAGRAM

Data Flow Diagram (DFD):

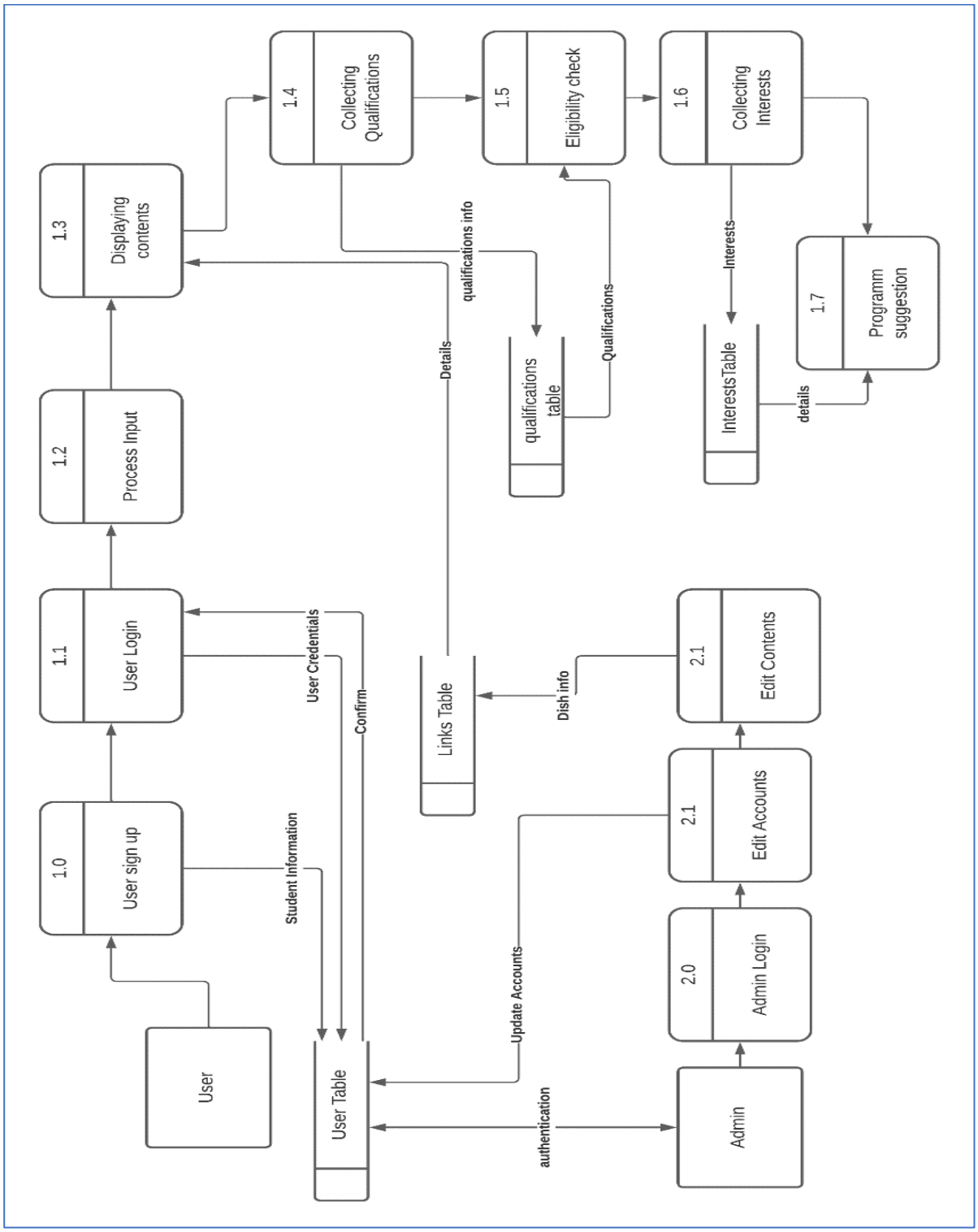


FIGURE 4: DATA FLOW DIAGRAM (DFD)

Flowchart:

It shows how the main process works in the application. the user login to the app, then the user can browse and choose one of the features provided by the application. The application read the input and check if the input was a voice note, text, or a selection. If the input was a selection, for example the user chooses to browse the FAQ, then it will show the list of questions provided by the admission department. If the app provides more options in the FAQ, then user can select one of the option then the system will show the full details and information about the required topic. But if the input was a text input, then the system should process the text, analyze it, then retrieve the data and display it to the user. If the retrieved data show more options to the user, then the user can insert his option as a selection, text, or voice note. If the user provided the application with a voice note, then the system converts the speech to text and repeat the same process applied on the text input

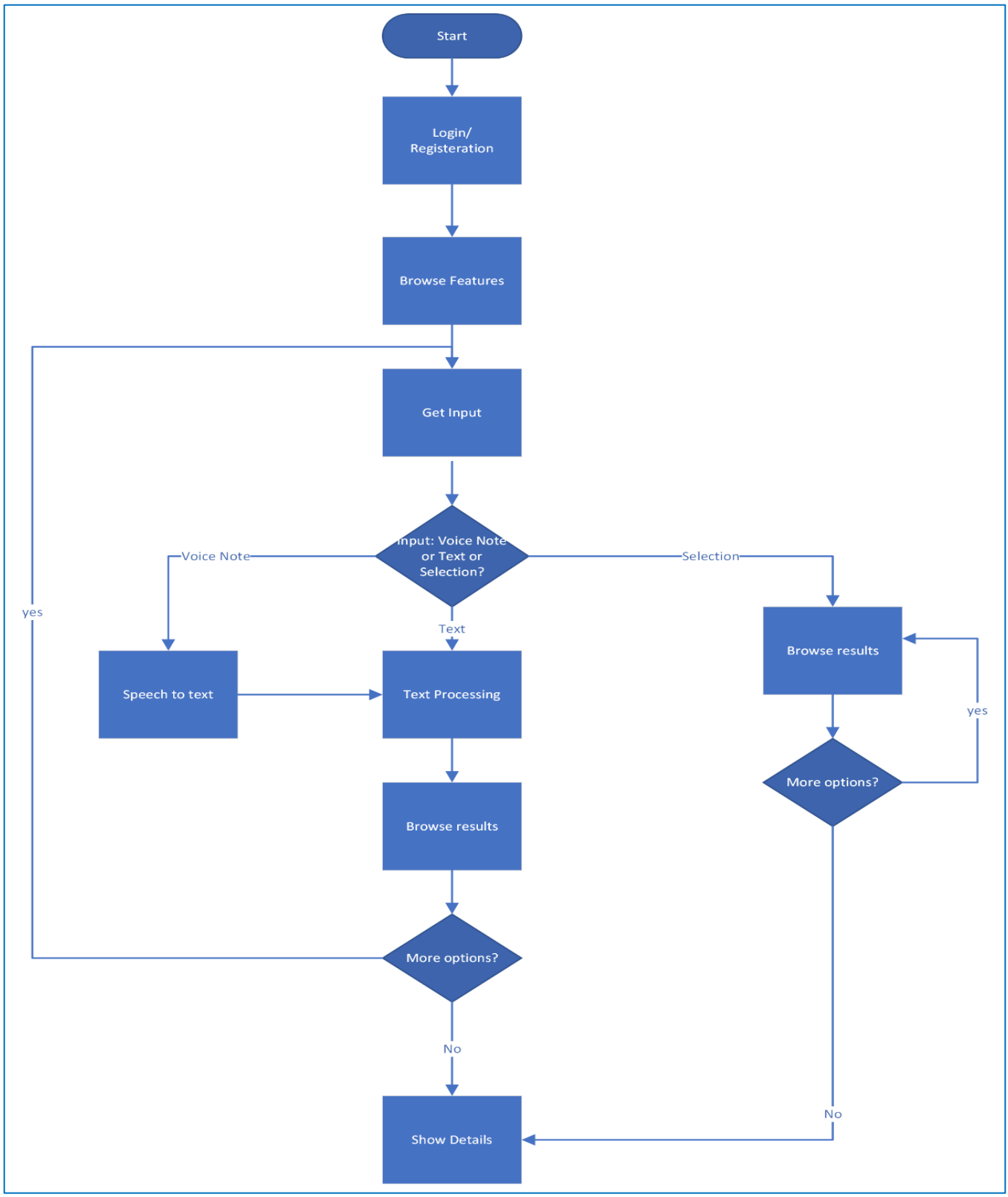


FIGURE 5: FLOWCHART

Object Modeling

Use case Diagram:

The use case diagram shows the main components of the proposed solution. It shows that the user can create an account or login using the registered account. The user has the ability to view and browse the main features provided by the app. The features can be browsing the FAQ section to view the most asked questions, or providing a text or a voice note about the user inquiry. The user can add the inquiry as a voice note, or a text to be processed and provide the users with the needed information. The application gives the users the option to add their academic background, test scores, interests and qualifications, and other relevant criteria in order to be processed and prepared. based on their academic background, test scores, and other relevant criteria, the app can assess an applicant's eligibility. Also, the chatbot can suggest academic programs or majors that align with the applicant's interests and qualifications. It can remind applicants of important deadlines for submitting applications, paying fees, and providing necessary documents.

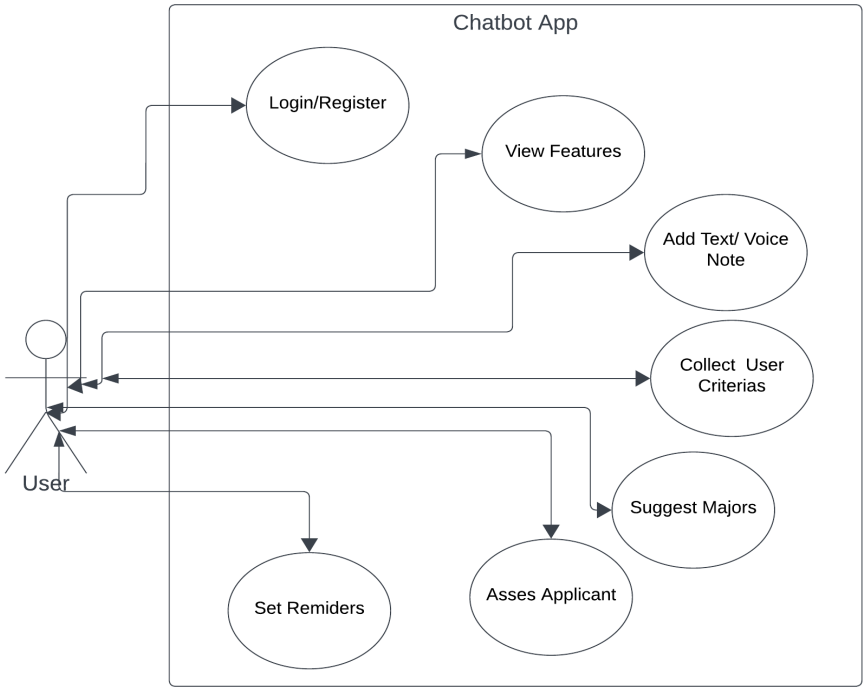


FIGURE 6: USE CASE

Activity Diagram:

An activity diagram for a university admission application illustrates the sequential steps involved in the application process, showing the flow of activities from start to finish [12]. Here's what each component typically represents:

Start/End:
The beginning and end points of the process.
Activity (Action):
Represents a specific action or task in the process, such as filling out an application form, submitting the application, reviewing, accepting, or rejecting the application.
Decision (Diamond):
Represents a point in the process where a decision needs to be made based on certain conditions. The flow can branch into different paths depending on the outcome of the decision.
Flow (Arrow):
Represents the direction of the process flow from one activity to another.
Merge (Diamond):
Represents the merging of different flows back into one flow after a decision point.
Here's a breakdown of the key activities in the context of a university admission application:

Start: The beginning of the application process.
Applicant fills out application form: The applicant completes the necessary forms with personal information, academic history, and other required details.
Submit application: The applicant submits the completed application.
Application complete? the app evaluates the application based on various criteria.
Accepted? Determines if the application is accepted in a certain branch.
App evaluate the submitted application, it shows the student the eligible branches that he can be enrolled in.

Query processing: The user adds the request. If the query was text? The application send the query to Dialog Flow to be processed and returns a keyword. If the query was voice? The voice is processed and converted to speech. Then the text is sent to Dialog Flow to be processed and returns a keyword.

Data retrieval: the app uses the keyword to retrieve the data from the Firebase database and display it to the users.

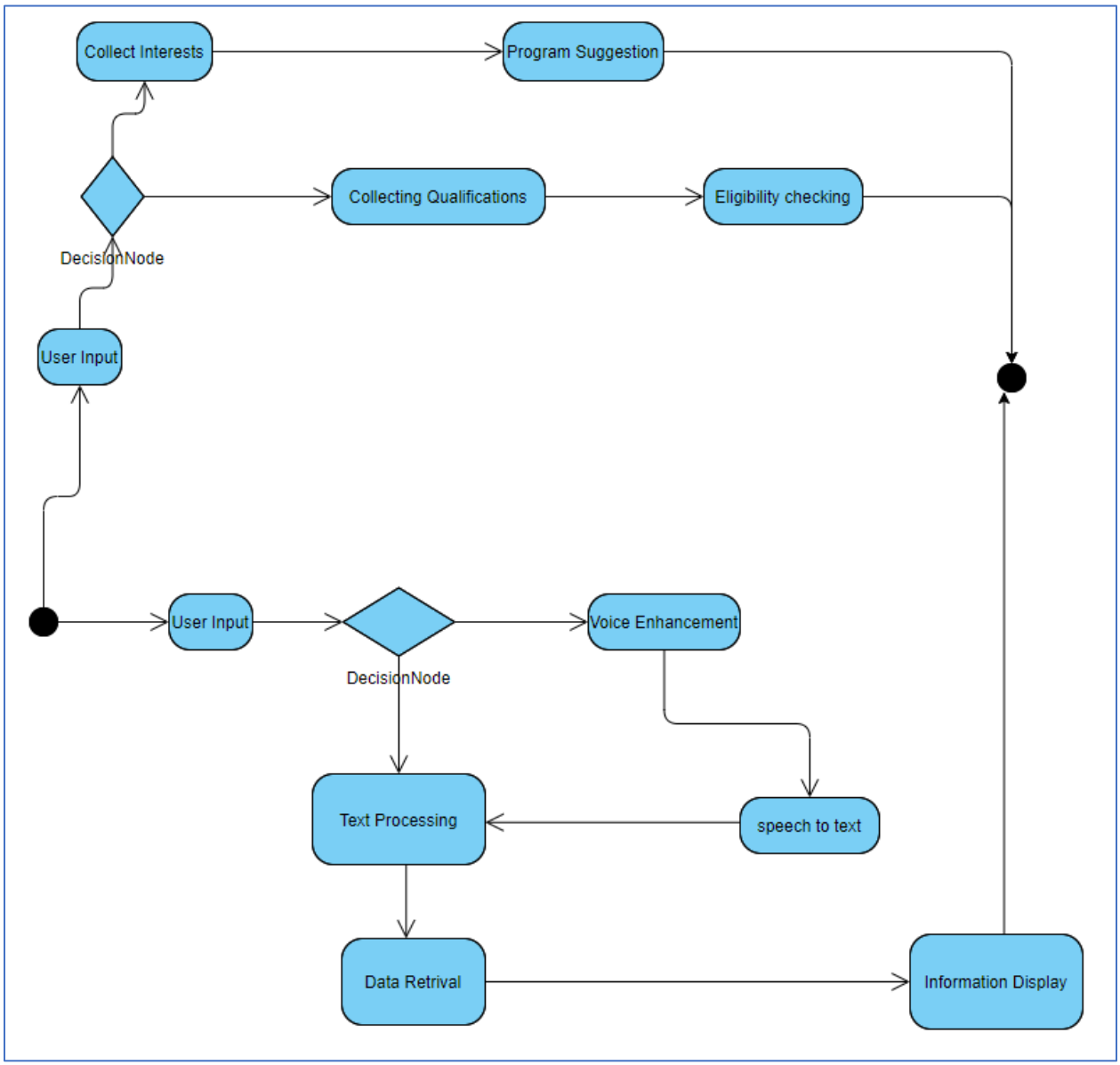


FIGURE 7: ACTIVITY DIAGRAM

Risk Assessment/Analysis

When conducting a risk assessment or analysis for a university admission application, several factors should be considered to ensure the integrity of the process and minimize potential risks. Here's an outline of the key areas to assess:

Data Security and Privacy:

Ensure that personal data provided by applicants is stored securely and complies with data protection regulations (e.g., GDPR).

Assess the security measures in place to prevent unauthorized access to sensitive information, such as academic records and personal details.

Fraud Prevention:

Identify potential risks of fraudulent applications, such as falsified documents or identity theft.

Implement verification processes to validate the authenticity of application materials, such as academic transcripts and recommendation letters.

Fairness and Bias:

Evaluate the application process for potential biases based on factors like race, gender, or socioeconomic background.

Implement measures to mitigate bias, such as blind application reviews or standardized testing.

Admissions Policy Compliance:

Ensure that the admission process adheres to the university's policies and regulations.

Identify any areas where the application process may deviate from established guidelines and address them accordingly.

System Reliability:

Assess the reliability of the application system to handle a large volume of submissions without errors or downtime.

Implement backup systems and contingency plans to mitigate the risk of system failure during critical periods, such as application deadlines.

Communication and Transparency:

Ensure clear communication of admission criteria and requirements to applicants to minimize confusion and misunderstandings.

Establish channels for applicants to seek clarification or assistance during the application process.

Ethical Considerations:

Evaluate the ethical implications of the admission process, such as the use of quotas or preferences for certain groups.

Ensure that the process promotes fairness, diversity, and equal opportunity for all applicants.

Feedback and Continuous Improvement:

Collect feedback from applicants, staff, and stakeholders to identify areas for improvement in the admission process.

Continuously review and update risk assessment measures to adapt to changing circumstances and emerging threats.

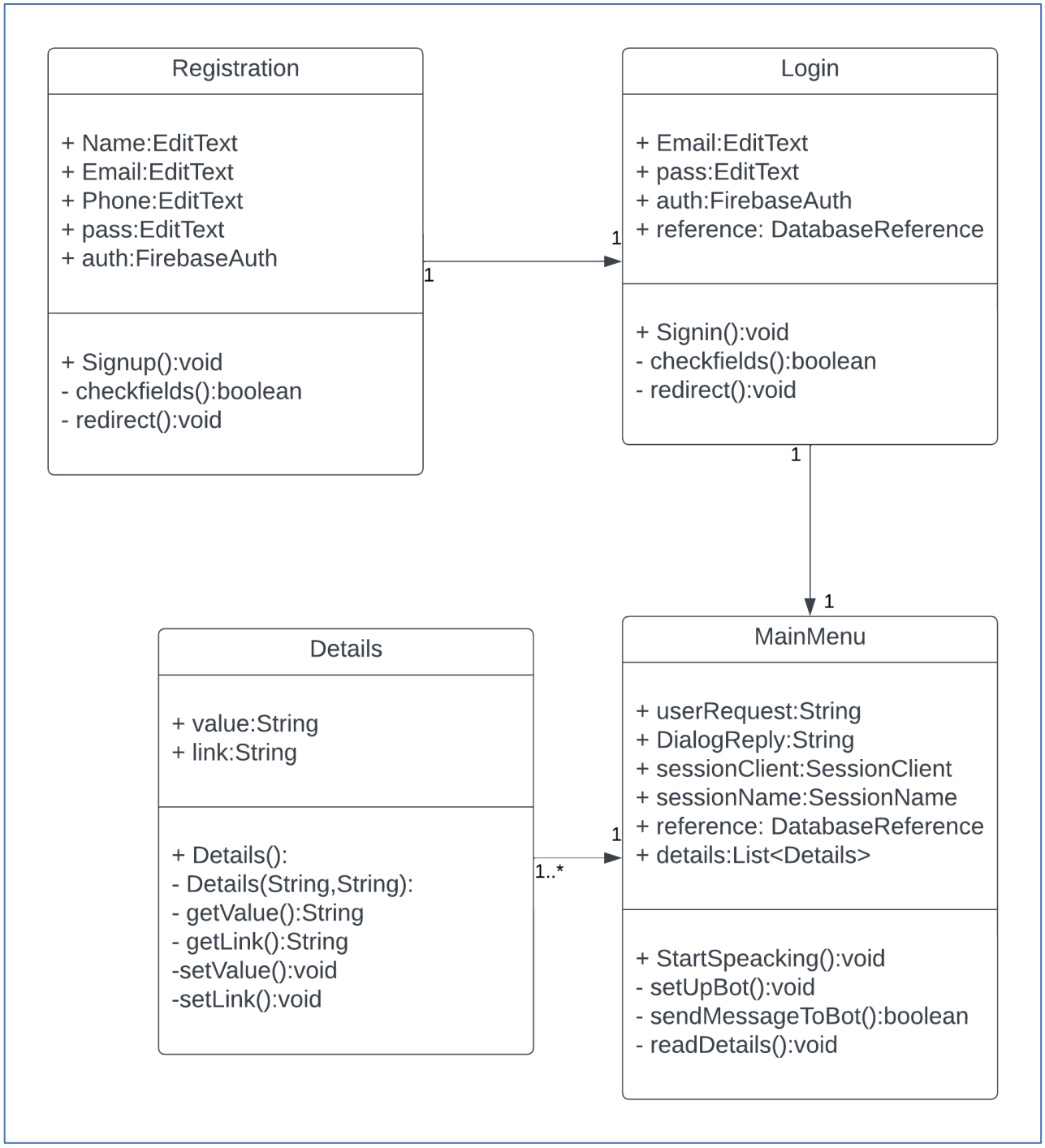
By systematically assessing these areas, universities can identify and mitigate risks associated with the admission process, ensuring fairness, security, and compliance while maintaining the integrity of their programs.

Risk	Description	Solution
Authentication and Security Risks	Unauthorized access to user accounts or sensitive data due to security vulnerabilities in the authentication process.	Implement robust security measures such as encryption of user credentials, secure token-based authentication with FirebaseAuth
Data Privacy Risks	Unauthorized access to user data stored in Firebase database, leading to privacy violations	Implement strict access controls and user permissions within Firebase, encrypt sensitive data
Speech-to-Text Conversion Risks	Inaccurate or unreliable conversion of voice notes	Conduct thorough testing of RecognizerIntent

	to text, leading to misinterpretation of user queries	functionality across different devices and environments to ensure accurate speech recognition
Natural Language Processing Risks	Inaccurate interpretation of user queries by DialogFlow ES, resulting in incorrect responses or failure to address user inquiries	Train DialogFlow ES with a diverse range of sample queries to improve its natural language understanding
Database Integrity Risks	Data inconsistencies or corruption in Firebase database due to errors in data retrieval	Implement data validation checks, transactional operations, and error handling mechanisms

Chapter 5: Development

Design: Detailed OO Design



Sequence Diagram

A sequence diagram for a university admission application depicts the interactions between various components involved in the process, showing the sequence of messages exchanged among them [13]. Here's what each component typically represents:

Actor:

Represents a user or external system interacting with the application. In the context of university admission, actors could include applicants, admission staff, or the application system itself.

Lifeline:

Represents the lifespan of an actor or component in the sequence diagram. It extends vertically from the actor or component's activation to its deactivation.

Message:

Represents communication between actors or components. Messages can be synchronous (denoted by solid arrows) or asynchronous (denoted by dashed arrows).

Activation:

Indicates when an actor or component is actively engaged in processing or responding to a message.

A breakdown of the key components in the context of a university admission application:

Applicant: Represents the person applying for admission to the university.

Admission System: Represents the system responsible for handling the admission process, including receiving applications, reviewing them, and sending notifications.

University Staff: Represents the staff members involved in the admission process, such as admissions officers or administrative personnel.

Sequence of Interactions:

Applicant submits application: The applicant initiates the process by submitting their application to the admission system.

Application received: The admission system acknowledges the receipt of the application.

Application review: The admission system reviews the application, which may involve verifying documents, checking qualifications, and assessing eligibility.

Send acceptance/rejection notification: Depending on the outcome of the review, the admission system sends an acceptance or rejection notification to the applicant.

Enrollment process: If accepted, the applicant may proceed with the enrollment process, which involves selecting courses, paying fees, and confirming enrollment.

Appeal process (optional): If the applicant is rejected and there is an appeal process available, the applicant may initiate an appeal.

Review appeal: University staff review the appeal, which may involve reconsidering the application or providing additional information.

Send appeal decision: The admission system sends the decision on the appeal to the applicant.

End: The completion of the admission process.

Sequence diagrams help visualize the flow of interactions and messages between different components, aiding in understanding the logic and timing of events in the admission application process.

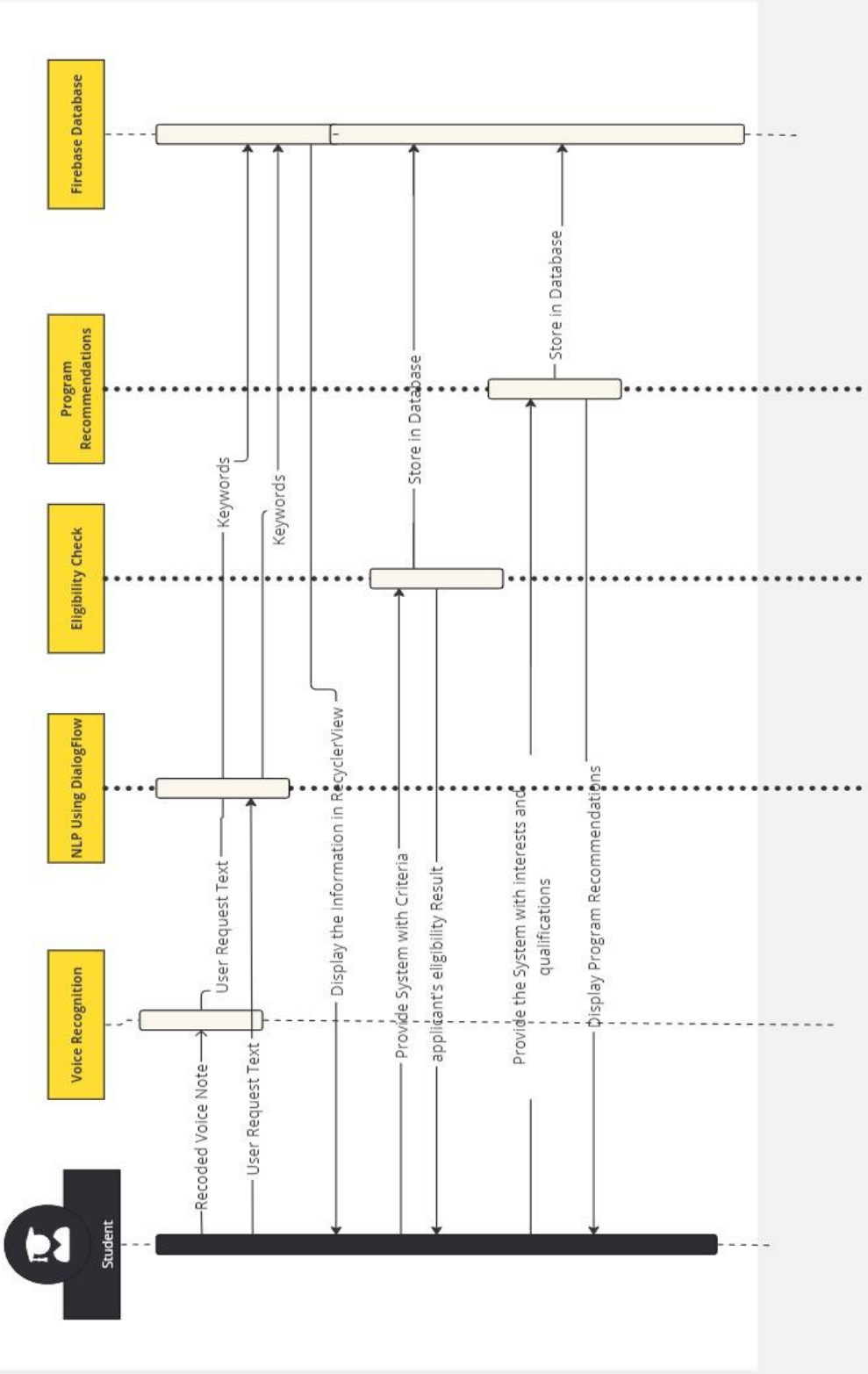


FIGURE 8: SEQUENCE DIAGRAM

Output and User-Interface Design

Registration Screen:

The users can create new accounts by entering their name, email, password and phone number as shown in figure10. A new record for the user will be saved in the database.

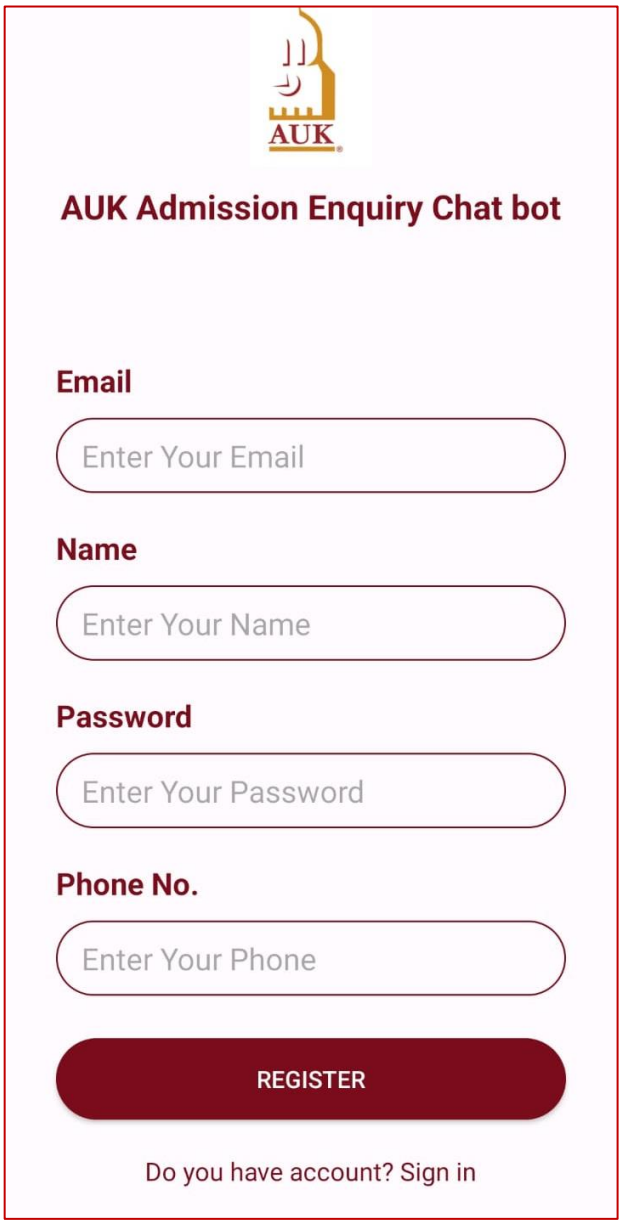


FIGURE 9: REGISTRATION SCREEN

Login Screen:

The users can access the application using the registered credentials. The users can use the email and password for the authentication process.

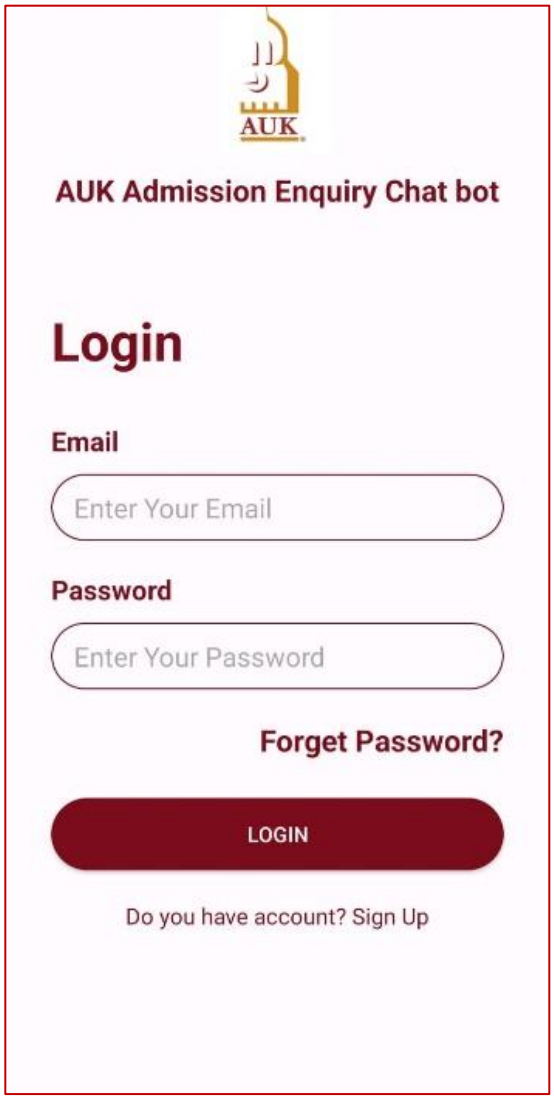


FIGURE 10: LOGIN SCREEN

Main Menu Screen:

In the main menu, the app gives the users the option to record a voice note as shown in figure12. The voice is converted to a text then the application responds with the result. The user also can write the request and send it to the application to respond with the requested information. The results are displayed in a recycler view as shown in figure13

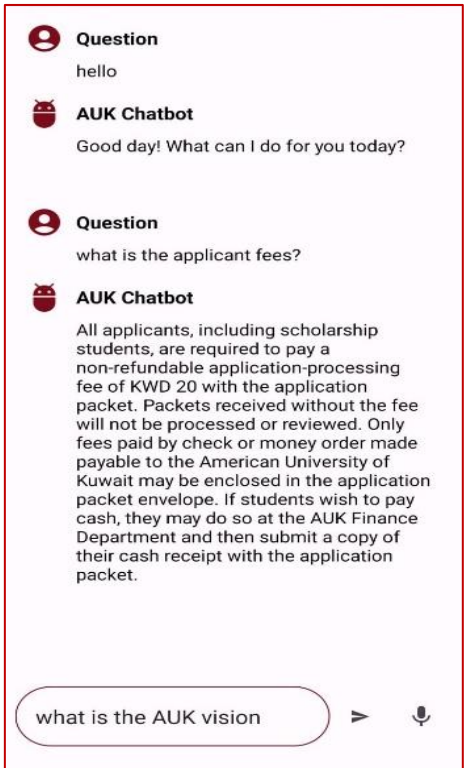


FIGURE 11: RECORDING VOICE NOTE

FIGURE 12: CHATBOT RESULT SCREEN

Data Design

Entity Relationship Diagram

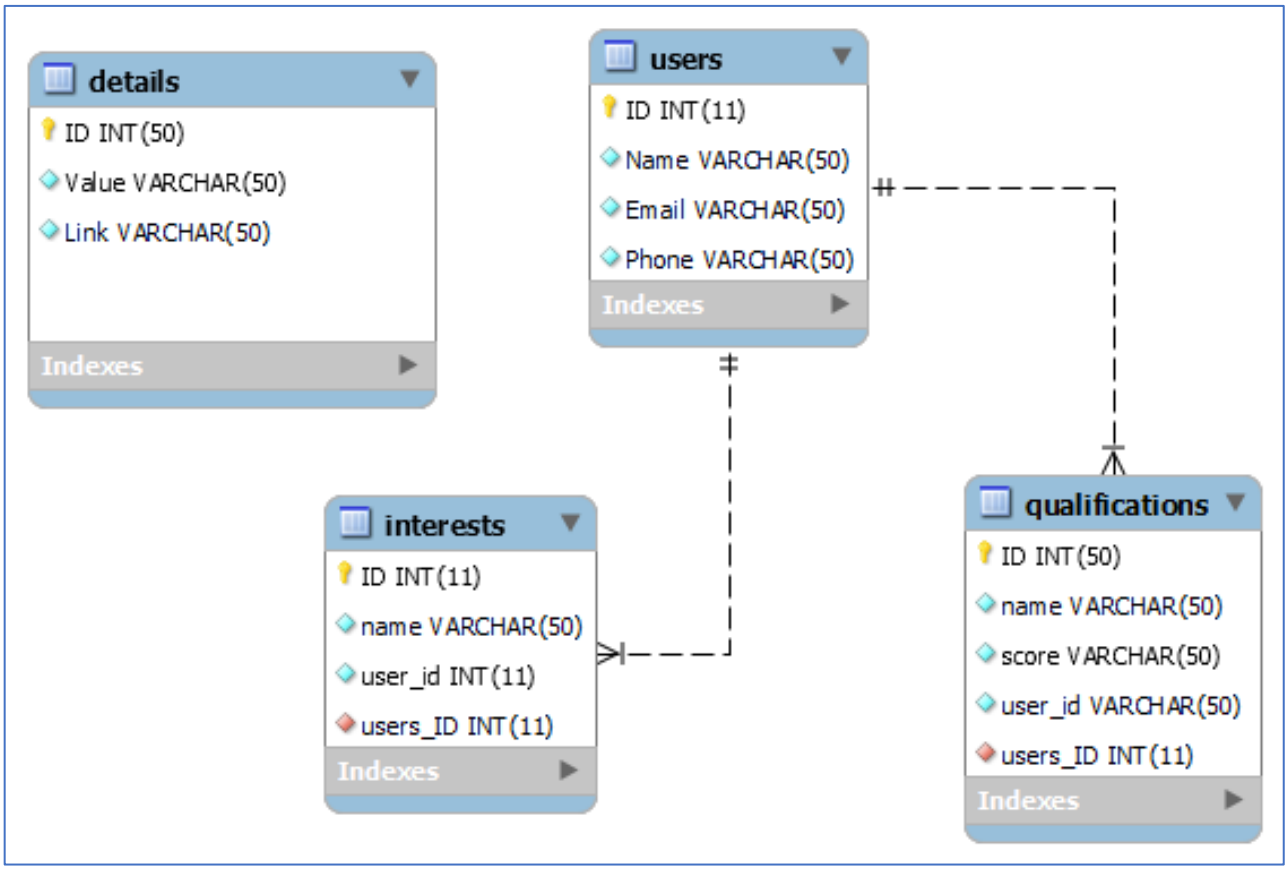


FIGURE 13:ENTITY RELATIONSHIP DIAGRAM

Data Dictionary

Firebase Realtime Database is a cloud-hosted NoSQL database provided by Google as part of the Firebase suite of services. It is designed to store and synchronize data in real-time across multiple clients. The Realtime Database is often used in mobile and web applications to build responsive features. Data is organized as a hierarchical JSON-like tree structure. This allows for flexible and dynamic data modeling. Firebase Realtime Database is a cloud-hosted NoSQL database provided by Google as part of the Firebase suite of services. It is designed to store and synchronize data in real-time across multiple clients.

The Realtime Database is often used in mobile and web applications to build responsive features.

Data is organized as a hierarchical JSON-like tree structure. This allows for flexible and dynamic data modeling.

Users Node

```
"users": {
  "5ok6xKGkX9ZsDUiqUvF34xwxRde2": {
    "name": “Ahmad",
    "email": “ahmad@auk.edu.kw",
    // ... other user data
  },
  // ... other users
}
```

Details Node

```
  “Vision": {
    “value": “Vision details … etc ",
    “link": “",
},
  // ... other links
}
```

Interests Node

```
“interests": {
  “interest1": {
```

"name": “Ahmad",

“user_id": “5ok6xKGkX9ZsDUiqUvF34xwxRde2 ",

},

// ... other interests

}

qualifications Node

“qualifications": {

“qulification1": {

"name": “Tofel",

“score": “6.4",

“user_id": “5ok6xKGkX9ZsDUiqUvF34xwxRde2 ",

},

// ... other qualifications

}

Development

Software Specification

- Android Studio: Android Studio is Android's official IDE. It is purpose built for Android to accelerate your development and help you build the highest-quality apps for every Android device. All updates by google on android will be present by android studio so there is no need to track the new updates and take care a lot about them. Android Studio provides a suitable environment where you can build apps for Android phones, tablets, Android Wear, Android TV, and Android Auto.
- Android Emulator: is a software application that helps developers to test and run Android applications on their PCs or laptops. It simulates the actions of a real Android device, it gives the developers the ability to test their apps across various screen sizes, hardware

configurations, and different Android versions without needing physical devices for each configuration.

- Database: The database used for saving the application data is Realtime Firebase database. It saves the users, details and user's interests.

Hardware Specification

- Computer: The developer needs a computer with 4 GB ram memory to run android studio application for the development. The user needs 8 GB ram to run both Android studio and Android emulator on the PC.
- Android Device: android devices are used to test the application on a real environment. It helps the developers to know how the application works and also it helps them in case of the limited resources on the computer to run emulator.

Program Specification

- The application is developed using the android studio with Java programming Language.
- The application uses 'Firebase AUTH' and 'signInWithEmailAndPassword' functions for the login process.
- The application also uses 'createUserWithEmailAndPassword' function for the registration and creating a new account.
- To detect the user voice and facilitate speech recognition functionality, the application used 'Recognizer Intent' class in the Android SDK.
- Dialog Flow ES is used for the natural language processing and understanding. It is a platform provided by Google Cloud. The primary job is to process user input in the form of text then extract the user's intent and any relevant parameters or entities from that input. Dialog flow ES then uses this information to generate an appropriate response or trigger a corresponding action.
- Retrieving Details: this feature is using the firebase database services to retrieve the details from Firebase database.

Programming Environment

Front End:

- In Android Studio, the user interface is designed using XML (Extensible Markup Language). XML is used for defining the UI elements, layouts, and other resources within Android apps. XML files in Android Studio serve as a asserting way in order to describe the general structure and appearance of user interfaces, as well as various other aspects of an Android app's settings and functionality.

Back End:

- Firebase Integration: in the chatbot app, we are depending mainly on Google Firebase services. We are using Firebase Authentication for the login and registration process. We also use Real-time Firebase database to save the information related to the chatbot. Firebase storage is used to save the files. The app also uses the Firebase Notification service to push notifications to the users if needed.
- Data Processing: The app uses Dialog Flow Essential service to process the user request and extract the user's intent from the input. We created multiple intents that can help the application in recognizing the user’s request.

Brief Manual:

Unit Testing:

Unit testing involves testing individual components or units of code to ensure they function correctly in isolation [14]. Here's how unit testing could be implemented in such an application:

1. User Interface Components:

- Login Screen: Unit tests could ensure that the login screen displays the correct UI elements, such as text fields for entering username and password, a login button, and any error messages.

- Application Form: Unit tests could verify that the application form displays the required fields for entering personal information, academic history, and supporting documents.

2. Business Logic Components:

- **User Authentication:** Unit tests could verify that the authentication logic correctly handles valid and invalid user credentials, including scenarios such as empty fields, incorrect passwords, and successful login attempts.

- **Application Validation:** Unit tests could ensure that the application validation logic correctly validates the input data, such as checking for required fields, valid email addresses, and acceptable document formats.

- **Application Submission:** Unit tests could verify that the submission logic correctly sends the application data to the server and handles success and failure responses.

3. Data Handling Components:

- **Data Retrieval:** Unit tests could ensure that data retrieval functions correctly retrieve data from the database, such as retrieving user information, application details, and admission criteria.

- **Data Manipulation:** Unit tests could verify that data manipulation functions correctly handle data transformations and calculations, such as calculating GPA or converting data formats.

4. Integration with External Services:

- **Firebase Authentication:** Unit tests could verify that the authentication integration with Firebase works as expected, including user registration, login, and logout functionality.

- **Firebase Database:** Unit tests could ensure that data is stored and retrieved correctly from the Firebase Database, including reading and writing application data.

5. Error Handling:

- **Validation Errors:** Unit tests could check that appropriate error messages are displayed when users enter invalid data or omit required fields.

- **Server Errors:** Unit tests could verify that the application gracefully handles errors returned by the server, such as network timeouts or server-side validation errors.

6. Edge Cases and Boundary Conditions:

- **Minimum and Maximum Values:** Unit tests could validate that the application handles edge cases, such as minimum and maximum values for input fields (e.g., minimum and maximum GPA).

- **Unsupported Input:** Unit tests could verify that the application gracefully handles unsupported input or unexpected user actions, such as entering non-numeric characters in a GPA field.

7. Performance Testing:

- **Response Time:** Unit tests could measure the response time of critical functions or API calls to ensure they meet performance requirements.

- **Resource Usage:** Unit tests could monitor resource usage, such as memory and CPU usage, to identify any performance bottlenecks.

Overall, unit testing in a university admission application helps ensure the reliability, functionality, and performance of individual components, leading to a more robust and user-friendly application.

Functionality	Description	Steps for testing	Actual Result
Test Case 1: Valid Login Credentials	Ensure that users can successfully log in with valid credentials	1. Provide valid email and password. 2. Attempt to log in. 3. Verify that the login is successful	Login Successfully and move to the main screen
Test Case 2: Invalid Login Credentials	Verify that the application handles invalid login credentials appropriately	1. Provide invalid email or password. 2. Attempt to log in. 3. Verify that the login fails and	Error message and ask the user to login again

		appropriate error message is displayed.	
Test Case 3: Voice to Text Conversion	Ensure that voice notes are accurately converted to text	1. Provide a voice note as input. 2. Convert the voice note to text. 3. Verify that the text output matches the content of the voice note	Most cases it converts it correctly. There are some cases where the voice conversion is not accurate
Test Case 4: Intent Recognition	Verify that DialogFlow correctly recognizes the intent of the user query.	1. Send a user query to DialogFlow. 2. Verify that DialogFlow identifies the correct intent	The system retrieve the correct results if the intent is available in the DialogFlow
Test Case 1: Data Retrieval	Verify that the application retrieves relevant data from Firebase based on user queries	4. Send a query to DialogFlow. 5. Retrieve data from Firebase based on DialogFlow's response. 6. Verify that the retrieved data matches the expected result.	The data is retrieved correctly from the firebase database

Integration Testing

Compatibility Testing

1. **Operating Systems:**
   - **Android Devices:**
     - Test the application on different Android devices running various versions of the Android operating system (e.g., Android 11, Android 12, Android13, Android 14), it worked perfectly.
2. **Network Environments:**
   - **Mobile Data and Wi-Fi:**
     - Test the application's functionality and performance under stable Wi-Fi connection and mobile data networks (4G). It worked perfectly.

Performance Testing:

Scenario Tested	Average Time (based on 10 times)
Login Response Time	2.3 seconds
Conversion and data retrieval time	5.4 second

Stress Testing

Stress testing evaluates the chatbot application's performance under extreme conditions to determine its stability, reliability, and scalability. The stress tests that can be done are concurrent login attempts, concurrent query processing, data retrieval load. Unfortunately, these tests can’t be applied efficiently now while we still don’t have enough number of users. The app was tested using 3 concurrent users which is very small number to put the app under stress.

Load Testing

Load testing evaluates the performance of the chatbot application under expected and peak loads to ensure it can handle concurrent user interactions without degradation in responsiveness or stability. The same situation of the stress testing also applied to load testing.

System Testing:

- **Test Scenario #1:** Application Launch
    - **Test Steps:**
        - Launch the Chatbot app.
        - Check the application startup and responsiveness.
        - Check for errors or crashes during startup.
    - **Test Result:** The Chatbot application launched without errors or crashes.
- **Test Scenario #2:** Voice Conversion and recognition
    - **Test Steps:**
        - Record the voice.
        - Convert the voice to text.
        - Send the text to Dialgflow.
        - Retrieve the parameters
    - **Test Result:** The Chatbot application converted the voice correctly and send appropriate response .
- **Test Scenario #3:** Data Retrieval
    - **Test Steps:**
        - Send the response to Firebase Database.
        - Retrieve the corresponding data
    - **Test Result:** The Chatbot application receive the data and display it correctly to the users

References:

1. Kuhail, M.A., Alturki, N., Alramlawi, S. et al. Interacting with educational chatbots: A systematic review. Educ Inf Technol 28, 973–1018 (2023).
2. Li, C., & Wang, Y. (2018). Design and Implementation of an Android Chatbot System based on AI.
3. Nimavat, Ketakee & Champaneria, Tushar. (2017). Chatbots: An overview. Types, Architecture, Tools and Future Possibilities.
4. Google LLC. Android Developer Documentation. Retrieved from https://developer.android.com/develop
5. Staffordshire University. (n.d.). Beacon. Retrieved November 27, 2023, from https://beacon.staffs.ac.uk/
6. St. John's University. (n.d.). Johnny Chat. Retrieved December 02, 2022, from https://www.stjohns.edu/office-information-technology/technology-labs-and-resources/johnny-chat https://doi.org/10.1007/s10639-022-11177-3
7. Oracle Corporation. Java Documentation. Retrieved from https://docs.oracle.com/en/java/
8. Google LLC. Firebase Documentation. Retrieved from https://firebase.google.com/docs
9. H. Abdulla, A. M. Eltahir, S. Alwahaishi, K. Saghair, J. Platos and V. Snasel, "Chatbots Development Using Natural Language Processing: A Review," 2022 26th International Conference on Circuits, Systems, Communications and Computers (CSCC), Crete, Greece, 2022, pp. 122-128, doi: 10.1109/CSCC55931.2022.00030
10. Google LLC. Dialogflow Documentation. Retrieved from https://cloud.google.com/dialogflow/es/docs
11. Sajjapanroj, Suthiporn & Longpradit, Panchit & Polanunt, Kaanwarin. (2020). A Prototype of Google Dialog Flow for School Teachers' Uses in Conducting Classroom Research. Asian Journal of Education. 15. 2020.
12. Seidl, Martina & Scholz, Marion & Huemer, Christian. (2015). The Activity Diagram. 10.1007/978-3-319-12742-2_7.
13. Al-Fedaghi, Sabah. (2021). UML Sequence Diagram: An Alternative Model.
14. Runeson, Per. (2006). A Survey of Unit Testing Practices. IEEE Software. 23. 10.1109/MS.2006.91.