|  |  |
| --- | --- |
| Quaid-i-Azam University - Crunchbase School Profile & Alumni | **Institute of Information Technology**  **Quaid-i-Azam University Islamabad.** |

**AI Student Advisor**

***By***

**Ahmad Latif (04162113032)**

***Supervisor*Dr. Basharat Mehmood**

***Bachelor of Science in Information Technology (2021-2025)***

**The candidate confirms that the work submitted is their own and appropriate  
 credit has been given where reference has been made to the work of others**.

|  |  |
| --- | --- |
| Quaid-i-Azam University - Crunchbase School Profile & Alumni | **Institute of Information Technology**  **Quaid-i-Azam University Islamabad.** |

**Ai Student Advisor**

**A project presented to**

**Institute of Information Technology**

**Quaid-i-Azam University, Islamabad**

**In partial fulfillment**

**of the requirement for the degree of**

***Bachelor of Science in Information Technology (2021-2025)***

**By**

**Ahmad Latif (04162113032)**

**DECLARATION**

We hereby declare that this software, neither whole nor as a part has been copied out from any source. It is further declared that we have developed this software and accompanied report entirely on the basis of our personal efforts. If any part of this project is proved to be copied out from any source or found to be reproduction of some other, we will stand by the consequences. No Portion of the work presented has been submitted of any application for any other degree or qualification of this or any other university or institute of learning.

Ahmad Latif

---------------------------

**CERTIFICATE OF APPROVAL**

It is to certify that the final year project of BS (CS) “AI Student Advisor” was developed by   
**Ahmad Latif (04162113032)** under the supervision of “Dr Basharat” and co supervisor “Ms. Sidra Batool Kazmi” and that in their opinion; it is fully adequate, in scope and quality for the degree of Bachelor of Information Technology.

---------------------------------------

**Supervisor**

---------------------------------------

**External Examiner**

---------------------------------------

**Head of Department**

**(Institute of Information Technology)**

**Executive Summary**

In the academic journey of students, timely guidance for course selection, learning support, and career preparation is crucial. Traditionally, this guidance is provided through human advisors or separate tools, but these methods face several limitations such as availability constraints, inconsistent advice, lack of personalization, and absence of a centralized system. Moreover, students often find it difficult to get immediate help for doubts, smart course planning, or realistic interview preparation especially when human advisors are unavailable.

To overcome these challenges, the **AI Student Advisor** is developed. It is a hybrid mobile application that intelligently supports students throughout their academic and pre-professional life. The system offers round-the-clock AI-based assistance, tailored according to the user’s semester, interests, and availability. It reduces dependency on manual advising while ensuring consistent, scalable, and context-aware support for every student. **AI Student Advisor** is a multi-functional app that includes:

An **AI Tutor** that answers academic queries using real-time responses powered by Gemini API.

A **Course Selection Assistant** that processes the student's timetable and recommends conflict-free courses based on their interests.

A **Mock Interview Using AI** that conducts voice-based job interviews with automatic feedback and scoring.

A **Student Profile Management** to manage courses and view profile details.

A **Todo List** where student can manage everyday tasks.

The system is built using Flutter for cross-platform compatibility and is powered by **Firebase** for secure login and data storage. Speech-to-text and text-to-speech APIs enable interactive voice communication. It is designed for ease of use and can be accessed anytime from any device, offering students an all-in-one academic and career assistant. By centralizing these essential academic services into one AI-driven platform, the AI Student Advisor ensures that no student is left behind due to lack of timely support.

**Acknowledgement**

All praise is to Almighty Allah who bestowed upon us a minute portion of His boundless knowledge by virtue of which we were able to accomplish this challenging task.

We are greatly indebted to our project supervisor “Dr Basharat” and our Co-Supervisor “Ms. Sidra Batool Kazmi”. Without their personal supervision, advice and valuable guidance, completion of this project would have been doubtful. We are grateful to them for their encouragement and continual help during this work.

And we are also thankful to our parents and family who have been a constant source of encouragement for us and brought us with the values of honesty & hard work.

Ahmad Latif

---------------------------

**Abbreviations**

|  |  |
| --- | --- |
| **SRS** | Software Require Specification |
| **PC** | Personal Computer |
| **UI** | User Interface |
| **UX** | User Experience |
| **SDK** | Software Development Kit |
| **API** | Application Programming Interface |
| **PDF** | Portable Document Format |
| **CSV** | Comma-Separated Values |
| **IDE** | Integrated Development Environment |
| **TTS** | Text To Speech |
| **STT** | Speech To text |
| **CI/CD** | Continuous Integration/ Continuous Development |
| **JSON** | JavaScript Object Notation |
| **AI** | Artificial Intelligence |

**Table of Contents**

[1 Introduction 10](#_Toc203913253)

[1.1 Vision Statement 10](#_Toc203913254)

[1.2 Related System Analysis/Literature Review 10](#_Toc203913255)

[1.3 Project Deliverables 11](#_Toc203913256)

[1.4 System Limitations/Constraints 12](#_Toc203913257)

[1.5 Tools and Technologies 12](#_Toc203913258)

[1.6 Relevance to Course Modules 13](#_Toc203913259)

[2 Problem Definition 14](#_Toc203913260)

[2.1 Problem Statement 14](#_Toc203913263)

[2.2 Problem Solution 14](#_Toc203913264)

[2.3 Objectives of the Proposed System 15](#_Toc203913265)

[2.4 Scope 15](#_Toc203913266)

[2.5 Modules 16](#_Toc203913267)

[2.5.1 Module 1: Course Selection Assistant 16](#_Toc203913268)

[2.5.2 Module 2: AI Tutor 16](#_Toc203913269)

[2.5.3 Module 3: Mock Interview Using Ai 16](#_Toc203913270)

[2.5.4 Module 4: User Profile & Authentication 17](#_Toc203913271)

[3 Requirement Analysis 17](#_Toc203913272)

[3.1 User classes and characteristics 17](#_Toc203913273)

[3.2 Requirement Identifying Technique 18](#_Toc203913274)

[3.2.1 Use Case Diagram 18](#_Toc203913275)

[3.2.2 Event Response Table 18](#_Toc203913276)

[3.3 Functional Requirements 19](#_Toc203913277)

[3.3.1 Functional Requirements 19](#_Toc203913278)

[3.4 Non-Functional Requirements 22](#_Toc203913279)

[3.4.1 Reliability 22](#_Toc203913280)

[3.4.2 Usability 23](#_Toc203913281)

[3.4.3 Performance 23](#_Toc203913282)

[3.4.4 Security 23](#_Toc203913283)

[3.5 External Interface Requirements 23](#_Toc203913284)

[3.5.1 User Interfaces Requirements 24](#_Toc203913285)

[3.5.2 Software interfaces 24](#_Toc203913286)

[3.5.3 Hardware interfaces 25](#_Toc203913287)

[3.5.4 Communications interfaces 26](#_Toc203913288)

[4 Design and Architecture 27](#_Toc203913289)

[4.1 Architectural Design 27](#_Toc203913294)

[4.2 Box and Line Diagram 28](#_Toc203913295)

[4.3 Design Models 29](#_Toc203913296)

[4.3.1 Activity Diagram 29](#_Toc203913297)

[4.3.2 Class Diagram 30](#_Toc203913298)

[4.3.3 Sequence Diagram 30](#_Toc203913299)

[4.4 Data Design 33](#_Toc203913300)

[4.4.1 Data Dictionary 33](#_Toc203913301)

[4.5 Human Interface Design 34](#_Toc203913302)

[4.5.1 Screen Images 34](#_Toc203913303)

[4.5.2 Screen Objects and Actions 34](#_Toc203913304)

[5 Implementation 36](#_Toc203913305)

[5.1 Algorithm 36](#_Toc203913307)

[5.2 External APIs/SDKs 38](#_Toc203913308)

[5.3 User Interface 38](#_Toc203913309)

[5.3.1 Splash Screen 5.3.2 Login Screen 39](#_Toc203913310)

[5.3.3 Signup Screen 5.3.4 Home Screen 40](#_Toc203913311)

[5.3.5 Course Selection Assistant 5.3.6 Ai Tutor 41](#_Toc203913312)

[5.3.7 Mock Interviews using Ai 5.3.8 Profile Page 42](#_Toc203913313)

[6 Testing and Evaluation 43](#_Toc203913314)

[6.1 Unit Testing 43](#_Toc203913317)

[6.2 Functional Testing 43](#_Toc203913318)

[6.3 Business Rules Testing 45](#_Toc203913319)

[6.3.1 Business Rule 1 45](#_Toc203913320)

[6.3.2 Business Rule 2 45](#_Toc203913321)

[6.3.3 Business Rule 3 46](#_Toc203913322)

[6.4 Integration Testing 46](#_Toc203913323)

[6.4.1 Integration Testing 1: Course Selection & Timetable Parsing 47](#_Toc203913324)

[6.4.2 Integration Testing 2: AI Tutor + Chat History 47](#_Toc203913325)

[6.4.3 Integration Testing 3: Mock Interview + Voice Input + Feedback 47](#_Toc203913326)

[7 Conclusion and Future Work 48](#_Toc203913327)

[7.1 Conclusion 48](#_Toc203913329)

[7.2 Future Work 49](#_Toc203913330)

[8 References 50](#_Toc203913331)

# Introduction

This chapter provides an overview of the **AI Student Advisor** project, outlining its purpose, scope, significance, target users, and core functionalities. It also briefly explains the motivation behind the project and the problems it aims to solve through intelligent academic assistance.

In today’s academic environment, students often face challenges in selecting appropriate courses, preparing for exams, and planning their career paths due to the lack of personalized guidance. Traditional advising systems are either time-consuming, generic, or unable to adapt to each student's individual needs. This gap in effective support has led to poor course choices, underperformance, and unpreparedness for career opportunities.

The **AI Student Advisor** is a smart, hybrid mobile application designed to bridge this gap. By leveraging artificial intelligence, the system offers tailored course recommendations, AI-powered tutoring and mock interview training based on the student’s academic performance, interests, and schedule. The application aims to empower students from all semesters particularly those in higher semesters making elective or career-related decisions with real-time, data-driven insights.

This project is developed with the intent to enhance student learning outcomes, improve academic decision-making, and equip students for their professional journeys. It ultimately contributes to building a smarter, more efficient, and student-centric educational support system.

## Vision Statement

1. ***For*** *university students* ***who*** *need personalized academic guidance and career planning support.* ***The*** *AI Student Advisor* ***is*** *an intelligent, hybrid mobile application* ***that*** *offers course selection recommendations, AI-powered tutoring and mock interview training tailored to each student’s interests and schedule.* ***Unlike*** *traditional static advising systems or generic guidance tools,* ***our product*** *provides dynamic, personalized, and AI-driven support that helps students make informed academic decisions, improve their learning outcomes, and prepare confidently for their future careers.*

## Related System Analysis/Literature Review

There are a number of academic advising and learning support systems that provide rudimentary course selection or tutoring capabilities. Yet, most of them do not provide personalization, incorporation of career planning tools, or responsiveness to students' academic calendars and achievement. Following are short descriptions of some applicable systems.

**1. DegreeWorks**

DegreeWorks is a web-based degree auditing and academic advising software used in the majority of universities. It helps students keep track of their academic standing and plan their upcoming courses. However, it does not offer AI-based customized course suggestions or career advice.

**2. Khan Academy**

Khan Academy has a strong library of free study materials and practice problems. While it is highly available and useful for general education, it has no curriculum mapping and individualized academic advising based on student schedules or university needs.

**3. Coursera Career Academy**

Coursera Career Academy offers career-oriented courses and certification. Although useful for skill building, it is not part of a student's current academic record and lacks mock interviews and university-specific course recommendations.

**4. ChatGPT-based Bots**

There are some learning environments that are testing out ChatGPT-driven bots to help students with homework-type questions. These are interactive and responsive but do not mesh well with institutional information such as course prerequisites, scheduling conflicts, or formal mock interview drills.

Table 1 Related System Analysis with proposed project solution

|  |  |  |
| --- | --- | --- |
| **Application Name** | **Weakness** | **Proposed Project Solution** |
| DegreeWorks | No AI-powered recommendations; lacks mock interview or career guidance | AI Student Advisor provides intelligent course suggestions and includes mock interview prep |
| Khan Academy | Generic learning support; no university course alignment or planning tools | Offers course selection based on university curriculum and provides focused tutoring support |
| Coursera Career Academy | Not tailored to current university programs; lacks academic integration | Integrates with student profiles and semester schedules to align career support with academics |
| ChatGPT-based Bots | Limited institutional data access; not structured for academic advising | AI Student Advisor accesses structured academic data to provide tailored, actionable guidance |

## Project Deliverables

**Mobile Application (Flutter-based Hybrid App)**  
A responsive cross-platform mobile application compatible with both Android and iOS devices.

**User Authentication**   
Secure login and signup features using Firebase Authentication for different user roles (e.g., students, admin).

**AI Tutor**   
An integrated AI-powered tutoring interface using Gemini API that answers academic queries through a chat interface.

**Course Selection Assistant**  
A module that helps students choose improvement and elective courses based on their semester, interests, and timetable, using PDF parsing and AI analysis.

**Mock Interview Assistant**  
An interactive feature that simulates real-world interviews and evaluates student responses using AI.

**Cloud Firestore Database Integration**  
For persistent data storage of user credentials, queries, chat history, course details, and interview sessions.

**Thesis**  
Complete SRS, architecture diagrams, testing results, and user manual.

**Presentation Slides & Report**  
A professional report (including this document) and PowerPoint slides for academic/project defense.

## System Limitations/Constraints

*LI1: The system relies heavily on internet connectivity for accessing Firebase services and AI APIs.*

*LI2: The AI tutor and interview assistant currently support only English and may struggle with local languages, slang, or highly domain-specific academic content.*

*LI3: The system does not include live human advisors or real-time mentoring. All suggestions are AI-generated and may lack the emotional intelligence or contextual awareness of a human mentor.*

*LI4****:****Speech-to-text accuracy may vary depending on the user's accent, background noise, and microphone quality, which can affect mock interview responses.*

*LI5****:****The timetable parsing feature depends on the format and clarity of the uploaded file. Poorly structured or handwritten files may result in parsing errors or incorrect recommendations.*

## Tools and Technologies

Table 2 Tools and Technologies for Proposed Project

|  |  |  |  |
| --- | --- | --- | --- |
| **Tools**  **And**  **Technologies** | **Tools** | **Version** | **Rationale** |
| Android Studio | 2024.1.1 Patch1 | IDE |
| Firebase | 2025 | Cloud-based backend for authentication and DB |
| Figma | Online | UI/UX design and wireframing |
| **Technology** | **Version** | **Rationale** |
| Flutter | 3.22.3 | Cross platform for front-end and back-end Development |
| Firebase Firestore | Latest | No SQL DB for Back-end Development |
| Gemini API | Flash 2.0 | API |

## Relevance to Course Modules

The **AI Student Advisor** project is a practical application that integrates multiple concepts and skills learned throughout the Bachelor of Computer Science (BCS) program. The following courses have directly contributed to the design, development, and management of this project:

* **Machine Learning (ML)**  
  Used for implementing AI-powered features such as personalized course recommendations and mock interview evaluations through Gemini API or other AI models.
* **Software Engineering (SE)**  
  Applied to follow software development life cycle (SDLC) phases, including requirement analysis, system design, coding standards, testing, and documentation.
* **Database Management Systems (DBMS)**  
  Essential for designing and managing the Firebase Firestore database that stores student data, course info, and interaction logs.
* **Introduction to Web**   
  Helped in understanding web-based technologies and backend integration, which is useful when connecting the app to Firebase and APIs.
* **Object-Oriented Programming (OOP)**  
  Used in structuring the Flutter (Dart) code with reusable components and logic through classes, inheritance, and encapsulation.
* **Software Project Management**  
  Guided the planning, scheduling, resource allocation, risk management, and progress tracking of the overall project.

# Problem Definition



## Problem Statement

University students often face difficulties in selecting the right courses, managing academic workloads,

and preparing for future careers due to the lack of personalized academic guidance. Traditional advising

systems are either overloaded, generic, or inaccessible, leading many students to make uninformed

decisions that can result in poor academic performance or delayed graduation. Moreover, students in

higher semesters struggle with choosing suitable electives and aligning their coursework with career

goals. The absence of tailored tutoring support and structured mock interview preparation further limits

students’ ability to perform well academically and succeed in job placements. Most existing systems do

not consider a student's individual interests, past academic record, or semester-specific constraints like

timetable conflicts. Manual course selection processes are often time-consuming and prone to human

error. There is also limited availability of platforms that combine academic advising, AI-based tutoring,

and career preparation in one accessible solution. As education becomes more data-driven and

competitive, students require intelligent tools that adapt to their academic journey and provide real-time,

relevant support. Therefore, there is a critical need for a mobile-based, AI-powered system that helps

students make smarter academic choices, improve learning outcomes, and build confidence for their

future careers.

## Problem Solution

To address the academic challenges faced by university students, the proposed **AI Student Advisor** is a hybrid mobile application that provides personalized academic guidance, AI-powered tutoring, and career preparation using Mock Interviews in a single platform. The system will analyze each student’s academic background, semester schedule, and preferences to recommend the most suitable courses, helping them avoid timetable conflicts and poor choices. The integration of artificial intelligence will enable real-time, dynamic interactions where students can receive tailored advice similar to what a human advisor might offer. The system features a mock interview module that simulates real job interview scenarios using AI, allowing students to practice and gain confidence before applying for internships or jobs.

The system’s mobile-first design ensures accessibility from anywhere, making it easier for students to get support without the need for physical appointments. Firebase will be used for backend services such as user authentication, database storage, and secure data handling. The application will maintain a user-friendly interface developed using Flutter, ensuring a smooth experience across Android and iOS devices.

By automating and personalizing academic advising, the AI Student Advisor reduces the load on human advisors while improving the quality of support available to students. The system will be scalable to support integration with university portals and course databases in the future. It empowers students to make smarter academic decisions, improve their academic performance, and prepare for their future careers effectively.

*.*

## Objectives of the Proposed System

***BO-1:****Provide personalized course selection recommendations based on student profiles, semester plans, and academic history.*

***BO-2:****Integrate AI-based tutoring to assist students in preparing for exams and understanding complex topics.*

***BO-3:****Offer mock interview training sessions using AI to simulate real interview environments.*

***BO-4:****Reduce dependency on human academic advisors by at least 50% through 24/7 AI assistance.*

***BO-5:****Improve student decision-making related to course enrollment and career planning.*

***BO-6:****Enable students to identify course conflicts and elective options early in the registration process.*

***BO-7:****Increase accessibility of academic advising tools via a user-friendly, cross-platform mobile app.*

***BO-8:****Enhance student preparedness for internships and job interviews through intelligent feedback systems.*

***BO-9:****Securely manage and store user data using Firebase Authentication and Firestore..*

## Scope

The scope of the **AI Student Advisor** project is to design and develop a hybrid mobile application that serves as an intelligent academic assistant for university students. The system will focus on delivering personalized course recommendations based on a student's academic history, interests, and semester schedule. It will allow students to upload or input their timetables to detect course conflicts and suggest improvement or elective courses accordingly. The application is intended for students from all semesters, particularly those in their 7th and 8th semesters who must make critical decisions regarding electives and career preparation.

One of the core functionalities includes an AI-powered chatbot that acts as a virtual tutor, helping students with exam preparation by providing explanations, summaries, and interactive Q&A support. Additionally, the system will offer a mock interview module where students can simulate job interview scenarios and receive intelligent feedback to enhance their communication and confidence. These features will help students make better academic decisions, prepare for exams more effectively, and build career readiness.

The backend services will be handled using Firebase, including real-time data storage, user authentication, and secure cloud access. The frontend will be developed using Flutter, ensuring compatibility across Android and iOS platforms. The interface will be designed to be intuitive and user-friendly, making the app accessible even for non-technical users. The system will maintain user data privacy and ensure secure interactions through Firebase’s authentication and database rules.

The current version will not include integration with university portals or live human advising. It will also be limited to English language support. Automated grading or plagiarism detection is outside the scope of this application. However, the system will be built modularly to allow future enhancements such as university integration, multi-language support, and performance tracking dashboards.

The main focus of the project is to simplify academic planning, reduce student reliance on manual or static advising methods, and improve academic performance through personalized AI support. This project does not aim to replace human advisors but to serve as a reliable 24/7 assistant that complements existing support systems. Overall, the AI Student Advisor will empower students to make informed academic choices, strengthen their preparation for assessments, and enhance their career opportunities.

## Modules

Following are the modules of proposed project

* Course Selection Assistant
* Ai Tutor
* Mock Interviews using Ai
* User Profile & Authentication

### ****Module 1: Course Selection Assistant****

* **FE-1:** Allow students to input or upload semester-wise academic timetable.
* **FE-2:** Recommend improvement or elective courses based on previous performance and current availability.
* **FE-3:** Detect and avoid timetable conflicts using AI logic.
* **FE-4:** Filter course suggestions based on interest, department, or credit hours.
* **FE-5:** Provide summary of selected courses and allow confirmation or removal.

### ****Module 2: AI Tutor****

* **FE-1:** Provide AI-powered academic support through a conversational chatbot.
* **FE-2:** Answer student queries related to course topics, definitions, and explanations.
* **FE-3:** Summarize long topics or generate quiz-style practice questions.
* **FE-4:** Support basic natural language input to maintain user-friendly interaction.
* **FE-5:** Log previous interactions for reference or review.

### ****Module 3: Mock Interview Using Ai****

* **FE-1:** Allow students to choose a domain/role (e.g., Software Developer, Data Analyst).
* **FE-2:** Simulate job interview sessions with AI-generated questions.
* **FE-3:** Capture and analyze user responses to provide feedback.
* **FE-4:** Offer guidance on improving communication and answering techniques.
* **FE-5:** Maintain a history of attempted mock interviews for progress tracking.

### ****Module 4: User Profile & Authentication****

* **FE-1:** Register and authenticate users using Firebase email/password login.
* **FE-2:** Maintain individual profiles including name, roll number, semester, and department.
* **FE-3:** Save academic history and AI preferences for personalized interaction.
* **FE-4:** Secure login/logout with Firebase Authentication.

# Requirement Analysis

This chapter outlines the requirement analysis for the **AI Student Advisor** application. It begins by identifying different user classes and their characteristics, followed by the techniques used to identify functional and non-functional requirements. The goal is to ensure that the system aligns with the expectations and needs of the end users, delivering an effective and personalized academic support experience.

## User classes and characteristics

|  |  |
| --- | --- |
| **User Classes** | **Description** |
| **Undergraduate Students** | These are the primary users ranging from semester 1 to 8. They use the app to receive course recommendations, get AI-based tutoring help, and prepare for interviews. They are generally tech-savvy and expect personalized, quick, and mobile-friendly solutions for academic decisions. |
| **Improvement Grade Students** | Students who are retaking one or more courses to improve their grades. They need specific help identifying conflict-free schedules and prefer guidance focused on academic performance and course load balancing. |
| **Graduating Students** | Final year students who are planning for jobs or internships. They use the system mainly for elective selection aligned with career paths and to practice mock interviews using AI. They are goal-oriented and expect job-preparation support. |

## Requirement Identifying Technique

### Use Case Diagram

Use Case of Ai student advisor


### Event Response Table

|  |  |  |
| --- | --- | --- |
| **Event** | **System State** | **Response** |
| Student opens app | App not authenticated | Display login screen |
| Student submits correct credentials | Login screen | Authenticate user and show home screen |
| Student submits incorrect credentials | Login screen | Show error message and stay on login screen |
| User clicks "AI Tutor" | Home screen loaded | Load AI chat history and show AI Tutor screen |
| User sends message to AI Tutor | AI Tutor screen active | Send message to Gemini API, display AI response |
| User uploads timetable PDF | Course Selector active | Parse timetable and extract schedule data |
| User taps “Get Course Suggestions” | Timetable parsed | Generate course suggestions and display them |
| User selects a course | Suggestions displayed | Save selected course to Firestore |
| User starts mock interview | Interview screen loaded | Load questions and begin voice interaction |
| User answers a question | Interview in progress | Convert voice to text and record answer |
| Interview ends | All questions answered | Generate feedback and show results |
| User taps “Logout” | Any screen | Clear session and return to login screen |

## Functional Requirements

This section outlines the core functional requirements for the AI Student Advisor system. These requirements describe what the system must do to fulfill the needs of its users. The requirements are organized by **key features**. Each requirement is uniquely identified, described, and labeled with its source, rationale, and priority.

### Functional Requirements

Following are the functional Requirements of this project:

Table 3 Description of FR-1

|  |  |
| --- | --- |
| **Identifier** | FR-1 |
| **Title** | Upload Timetable |
| **Requirement** | The student shall be able to upload a .txt, .csv, or .pdf file containing their academic timetable. |
| **Source** | Student need for conflict-aware course selection |
| **Rationale** | Helps the system identify free time slots and suggest courses accordingly. |
| **Business Rule** | File must not exceed 5MB |
| **Dependencies** | FR-2 |
| **Priority** | High |

Table 4 Description of FR-2

|  |  |
| --- | --- |
| **Identifier** | FR-1 |
| **Title** | AI-based Timetable Parsing |
| **Requirement** | Upon receiving the timetable file, the system shall use the AI to extract structured course data. |
| **Source** | Functional design specification |
| **Rationale** | Allows intelligent course matching by schedule |
| **Dependencies** | FR-1 |
| **Priority** | High |

Table 5 Description of FR-3

|  |  |
| --- | --- |
| **Identifier** | FR-3 |
| **Title** | Course Recommendations |
| **Requirement** | The system shall suggest elective or improvement courses that fit the student's interests and available time slots. |
| **Source** | Student need for conflict-aware course selection |
| **Rationale** | Helps students make informed course choices |
| **Dependencies** | FR-2 |
| **Priority** | High |

Table 6 Description of FR-4

|  |  |
| --- | --- |
| **Identifier** | FR-4 |
| **Title** | Ai Tutor |
| **Requirement** | The system shall allow students to send academic questions and receive contextual responses using Gemini AI. |
| **Source** | AI Tutor Module |
| **Rationale** | Enables real-time, intelligent tutoring |
| **Business Rule** | Queries should be under 200 characters |
| **Dependencies** | Firebase Authentication |
| **Priority** | High |

Table 7 Description of FR-5

|  |  |
| --- | --- |
| **Identifier** | FR-5 |
| **Title** | Save AI Tutor History |
| **Requirement** | The system shall save each tutoring session in Firestore linked to the user profile. |
| **Source** | Personalization requirement |
| **Rationale** | Allows students to revisit and reflect on previous sessions |
| **Dependencies** | FR-4 |
| **Priority** | Medium |

Table 8 Description of FR-6

|  |  |
| --- | --- |
| **Identifier** | FR-6 |
| **Title** | Starts Interview |
| **Requirement** | The student shall be able to begin a mock interview by selecting a topic, job title, and difficulty level. |
| **Source** | Mock Interview Module |
| **Rationale** | Personalizes the interview experience |
| **Dependencies** | None |
| **Priority** | High |

Table 9 Description of FR-7

|  |  |
| --- | --- |
| **Identifier** | FR-7 |
| **Title** | Generates Interview Questions |
| **Requirement** | The system shall generate questions based on the provided job role and difficulty. |
| **Source** | Mock Interview Trainer functionality |
| **Rationale** | Enables dynamic and realistic interview scenarios |
| **Dependencies** | FR-6 |
| **Priority** | High |

Table 10 Description of FR-8

|  |  |
| --- | --- |
| **Identifier** | FR-8 |
| **Title** | Capture Interview Answers Using Voice |
| **Requirement** | The student shall respond to interview questions via microphone. The system shall convert it to text. |
| **Source** | Mock Interview Module |
| **Rationale** | Simulates real interview environment |
| **Dependencies** | Microphone permissions |
| **Priority** | High |

Table 11 Description of FR-9

|  |  |
| --- | --- |
| **Identifier** | FR-9 |
| **Title** | Generates interview feedback |
| **Requirement** | After completion, the system shall use AI to generate a performance report with score and feedback. |
| **Source** | Mock Interview Report functionality |
| **Rationale** | Helps students identify strengths and weaknesses |
| **Dependencies** | FR-7, FR-8 |
| **Priority** | High |

Table 12 Description of FR-10

|  |  |
| --- | --- |
| **Identifier** | FR-10 |
| **Title** | Displays Interview Questions |
| **Requirement** | The system displays the interview questions and their respective answers so that user can see How did they replied. |
| **Source** | Mock Interview Feedback |
| **Rationale** | Enables students to review their performance. |
| **Dependencies** | FR-8 |
| **Priority** | Medium |

## Non-Functional Requirements

This section outlines the non-functional requirements of the **AI Student Advisor** system. These requirements define the system's expected quality attributes including reliability, usability, performance, and security. Each attribute ensures that the system operates efficiently, securely, and is user-friendly while meeting the expectations of stakeholders.

### ****Reliability****

**REL-1:** The system shall maintain a Mean Time Between Failures of at least 300 hours during peak academic periods (e.g., course registration or exams).  
**REL-2:** In the event of a failure (e.g., API failure or data fetch error), the system shall log the error and present the user with an informative fallback message within 3 seconds.  
**REL-3:** The system shall automatically attempt to reconnect to Firebase or Gemini API within 5 seconds of disconnection.  
**REL-4:** All critical user actions (e.g., course selections, mock interview results) shall be backed up to Firebase to prevent data loss in case of failure.

### ****Usability****

**USE-1:** The system shall allow a new user to complete initial setup (profile creation, login, and basic usage) within 5 minutes without requiring external help.  
**USE-2:** All chatbot modules shall support predefined quick-reply suggestions to guide new users in interaction.  
**USE-3:** The application shall display friendly error messages and provide users with clear suggestions for resolving invalid inputs (e.g., file upload errors or unsupported questions).  
**USE-4:** The user interface shall support accessibility standards including readable font sizes, sufficient contrast, and screen-reader compatibility for visually impaired users.  
**USE-5:** The system shall log and allow retrieval of previous tutoring sessions with a single tap.

### ****Performance****

**PER-1:** 95% of AI chatbot responses shall be returned within 4 seconds under standard network conditions.  
**PER-2:** File uploads (up to 5MB) shall be processed and parsed within 6 seconds.  
**PER-3:** The system shall support concurrent usage by up to 10,000 students without degradation in performance.   
**PER-4:** Voice-to-text responses during mock interviews shall be transcribed within 3 seconds on devices with average processing power.

### ****Security****

**SEC-1:** All user authentications shall be performed via secure, encrypted Firebase Authentication.  
**SEC-2:** The system shall not store plaintext passwords or sensitive information locally on the device.   
**SEC-3:** Communication between client and Firebase/AI APIs shall be encrypted using HTTPS/TLS 1.2 or higher.  
**SEC-4:** The system shall log unauthorized access attempts and alert administrators via Firebase monitoring.

## External Interface Requirements

### User Interfaces Requirements

The mobile application's user interface is developed using **Flutter**, adhering to Material Design guidelines to ensure consistency, usability, and accessibility. The interface is designed to provide an intuitive experience for students interacting with AI-powered features.

#### Interface Guidelines:

* The system shall follow **Google Material Design** standards across all screens.
* Standard font family shall be **Roboto**, with base size 14sp and larger font for headers.
* **Button labels** shall be concise, using clear action verbs (e.g., "Upload", "Submit", "Start Interview").
* Consistent **color scheme**: Primary theme color is deep blue (#0D47A1), with white backgrounds and accent highlights in cyan or light green.
* All screens shall maintain consistent **tabbing order** and logical keyboard focus for accessibility.
* **Snackbars** and **dialogs** are used to display confirmations, errors, and loading messages.
* The interface shall maintain **responsiveness** and adjust properly for different screen sizes (min width 320dp, max 1080dp).
* Voice interactions (e.g., in mock interview) shall display transcriptions and microphone animation for clarity.
* Accessibility accommodations include **screen reader compatibility** and sufficient contrast ratios for visually impaired users.

### Software interfaces

**SI-1: Firebase Authentication**  
**SI-1.1:** The AI Student Advisor shall authenticate users (students) using Firebase Authentication through email and password via Flutter’s FirebaseAuth plugin.  
**SI-1.2:** The system shall restrict access to user-specific data using Firebase Security Rules and UID-based matching.

**SI-2: Firebase Firestore Database**  
**SI-2.1:** The AI Student Advisor shall store and retrieve student profiles, uploaded timetable data, enrolled courses, tutoring conversations, and interview results using Firestore NoSQL database.  
**SI-2.2:** The application shall maintain real-time synchronization between client data and Firestore through secure API endpoints.

**SI-3: Gemini AI API**   
**SI-3.1:** The AI Student Advisor shall send prompt-based queries (course selection, tutoring, mock interview) to the Gemini API via HTTPS and receive dynamic AI-generated responses.  
**SI-3.2:** The system shall use Gemini for generating feedback reports, personalized recommendations, and question-answer content.

**SI-4: Speech-to-Text Plugin (Flutter speech to text)**  
**SI-4.1:** The AI Student Advisor shall convert voice responses from students during mock interviews into text using speech recognition plugins.  
**SI-4.2:** The app shall request microphone permissions from the OS before activating this service.

**SI-5: Text-to-Speech Plugin(Flutter tts)**  
**SI-5.1:** The AI Student Advisor shall use text-to-speech to read out interview questions or tutor messages when voice interaction is enabled.  
**SI-5.2:** The system shall allow users to toggle this setting as per their preference.

**SI-6: Flutter Markdown Renderer**  
**SI-6.1:** The AI Student Advisor shall render AI chatbot messages with markdown formatting using Flutter's markdown rendering library for improved readability.

**SI-7: Operating System Interfaces**  
**SI-7.1:** The application shall be compatible with Android 11.0+ and iOS 15+ using Flutter 3.22.3 for cross-platform mobile deployment.  
**SI-7.2:** The system shall rely on OS-level permissions for file access, microphone input, and internet connectivity.

### Hardware interfaces

**HI-1: Microphone**

* The system shall use the mobile device's built-in microphone to capture user responses in the Mock Interviewmodule.
* Audio data shall be streamed to a Speech-to-Text for real-time conversion into text.
* The app shall request permission from the operating system before accessing the microphone.
* Microphone access shall be activated only during the mock interview session and automatically disabled afterward.

**HI-2: Speaker**

* The system will use the device’s speaker for Text-to-Speech (TTS) output.
* In Mock Interview, the app will vocalize questions using TTS.
* Users can control speaker output through in-app toggle options.

**HI-3: Touch Screen**

* The mobile app will rely on the touchscreen for all user interactions including navigation, text input, and file selection.
* Standard touch gestures shall be supported for scrolling, tapping, typing, and long-press actions.

**HI-4: File Storage Access**

* The app shall access the mobile device’s internal storage or cloud drive (Google Drive/iCloud) to allow uploading of academic timetable files (.txt, .csv, .pdf).
* The system shall restrict file selection to supported formats and sizes (max 5 MB).
* OS-level permissions shall be used to request read-only access to user-selected files.

### Communications interfaces

The AI Student Advisor application relies on internet-based communication to interact with cloud services, APIs, and real-time data storage systems. All communication is handled using secure network protocols, and the application does not require email or SMS in its current version. Below are the defined communication interface requirements:

CI-1: The AI Student Advisor shall communicate with Firebase and Gemini AI APIs using secure HTTPS protocol over TLS 1.2 or higher to ensure data integrity and encryption. All requests and responses between the client and cloud services shall be transmitted via RESTful API calls.

CI-2: The application shall use Firebase Firestore's real-time data sync protocol to ensure instant updates of user data such as course selections and chatbot interactions. The sync shall be bidirectional, allowing both the client and the server to update shared data.

CI-3: The system shall require an active internet connection to perform AI-based functions such as course recommendations, tutoring conversations, and mock interviews. In the absence of connectivity, the system shall display a message: “You are offline. Some features may not be available.”

CI-4: The system shall transmit voice data from the Mock Interview Trainer module to a speech-to-text engine over the network in real-time. Voice inputs shall be processed securely without being stored permanently on the device.

# Design and Architecture



## Architectural Design

The AI Student Advisor system is designed using a layered architecture pattern to promote separation of concerns, scalability, and maintainability. The system is decomposed into four logical layers:

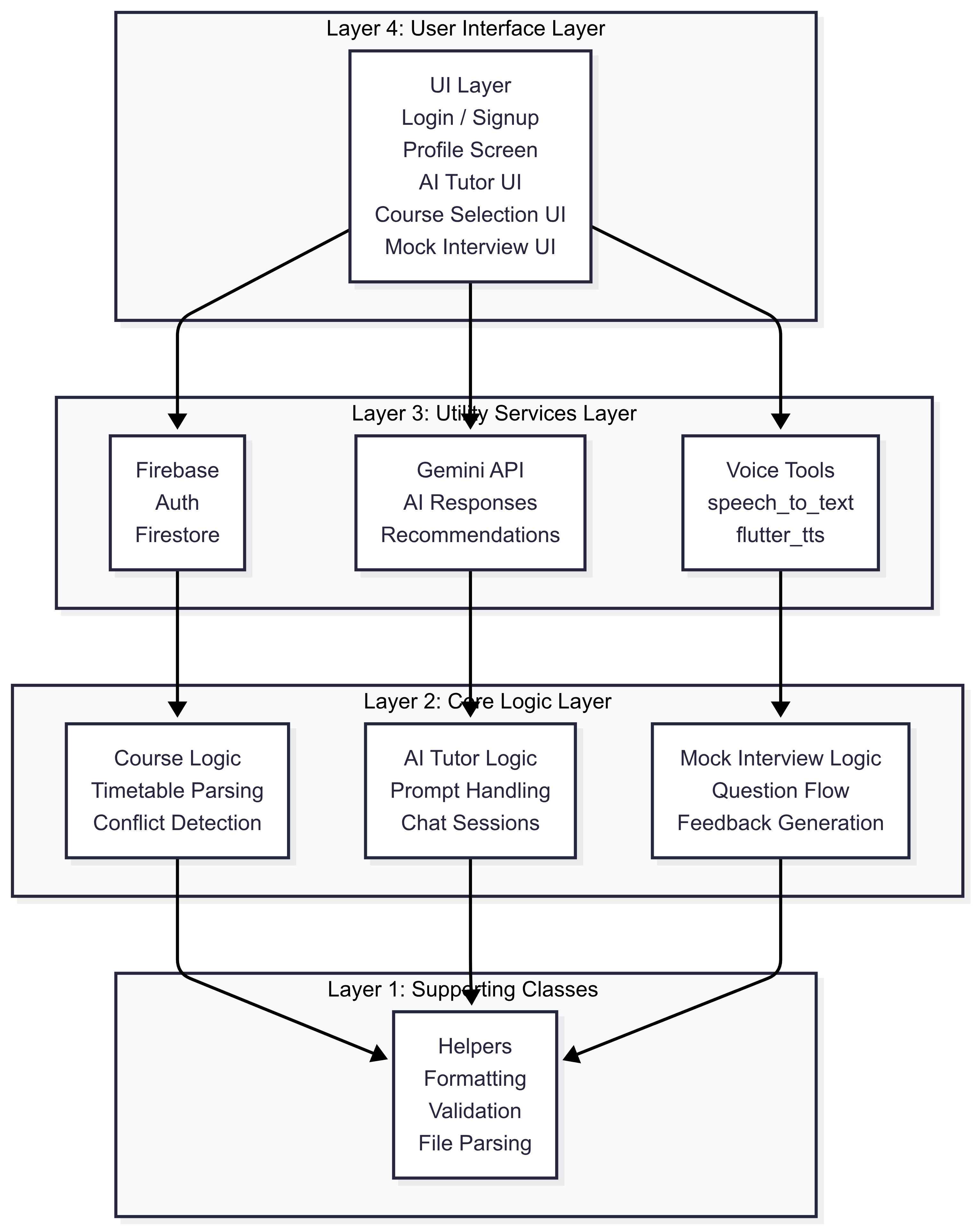
1. **Layer 1: Supporting Classes Layer** – This layer consists of all base utility classes and helper functions that provide core support to the system (e.g., file handling, formatting, validation utilities).
2. **Layer 2: Core Layer** – This layer contains the business logic and models such as course selection logic, timetable conflict detection, AI request formatting, and data preparation for Firestore and Gemini API.
3. **Layer 3: Utility Layer** – This contains services for interfacing with Firebase, Gemini AI, speech-to-text, and text-to-speech services. It abstracts complex external API calls from the core logic.
4. **Layer 4: User Interface Layer** – This is the topmost layer and handles all user-facing components developed in Flutter. It includes screens, widgets, forms, and interactive components such as the chatbot and mock interview screens.

Each layer communicates only with its immediate lower layer via well-defined interfaces, enforcing strong encapsulation. For example, the UI layer calls the utility services, which in turn call the core logic or API modules, while the core logic uses supporting classes from Layer 1 for base functionality.

This structure ensures that:

* UI changes do not impact core logic
* API service changes are isolated to the utility layer
* Support classes can be reused across layers

## Box and Line Diagram



## Design Models

The system is designed using the Object-Oriented Development approach, making use of UML (Unified Modeling Language) to describe the structure, behavior, and interaction between components. The following models illustrate the static and dynamic aspects of the AI Student Advisor system.

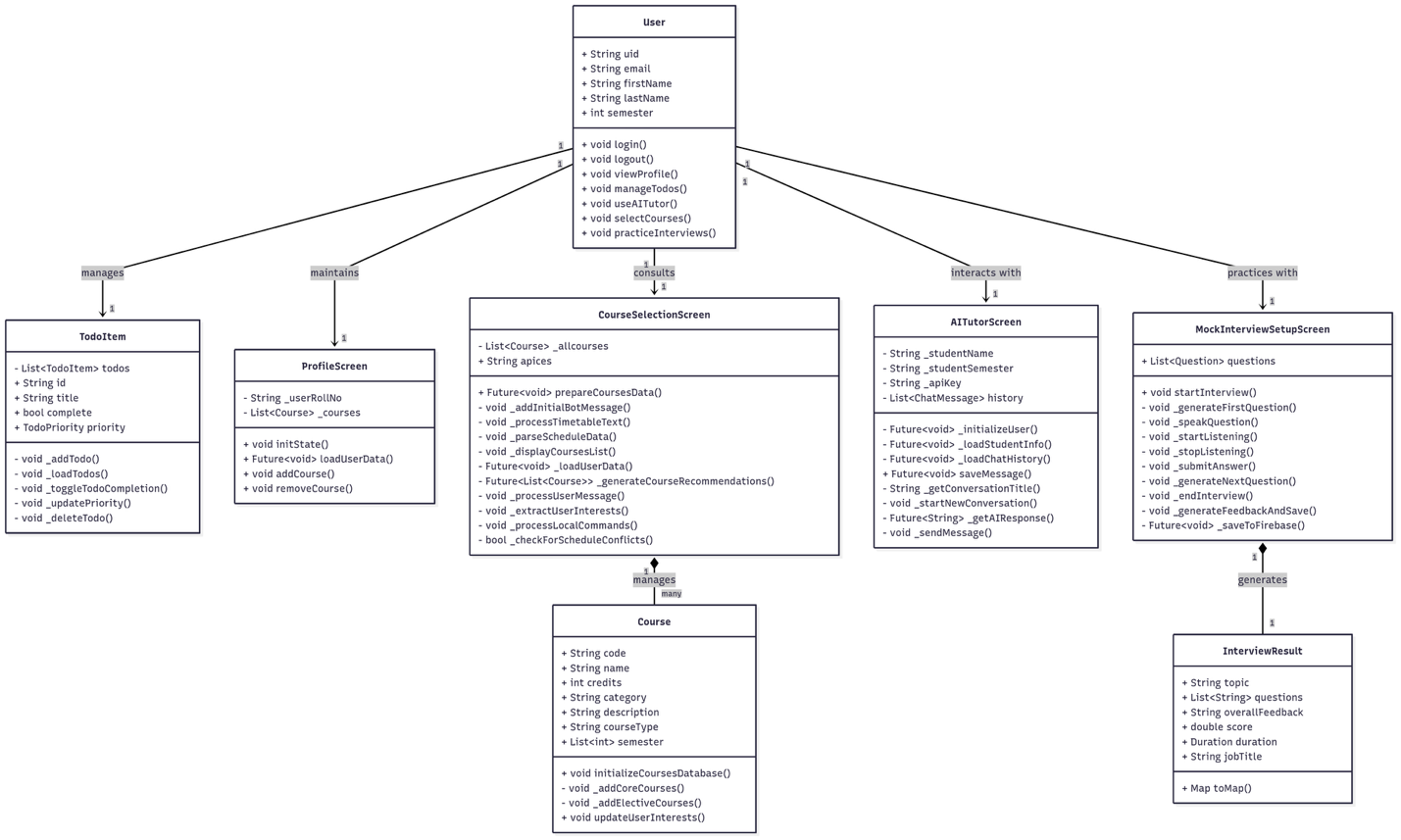
***Design Models for Object Oriented Development Approach***

### Activity Diagram

A diagram of a process

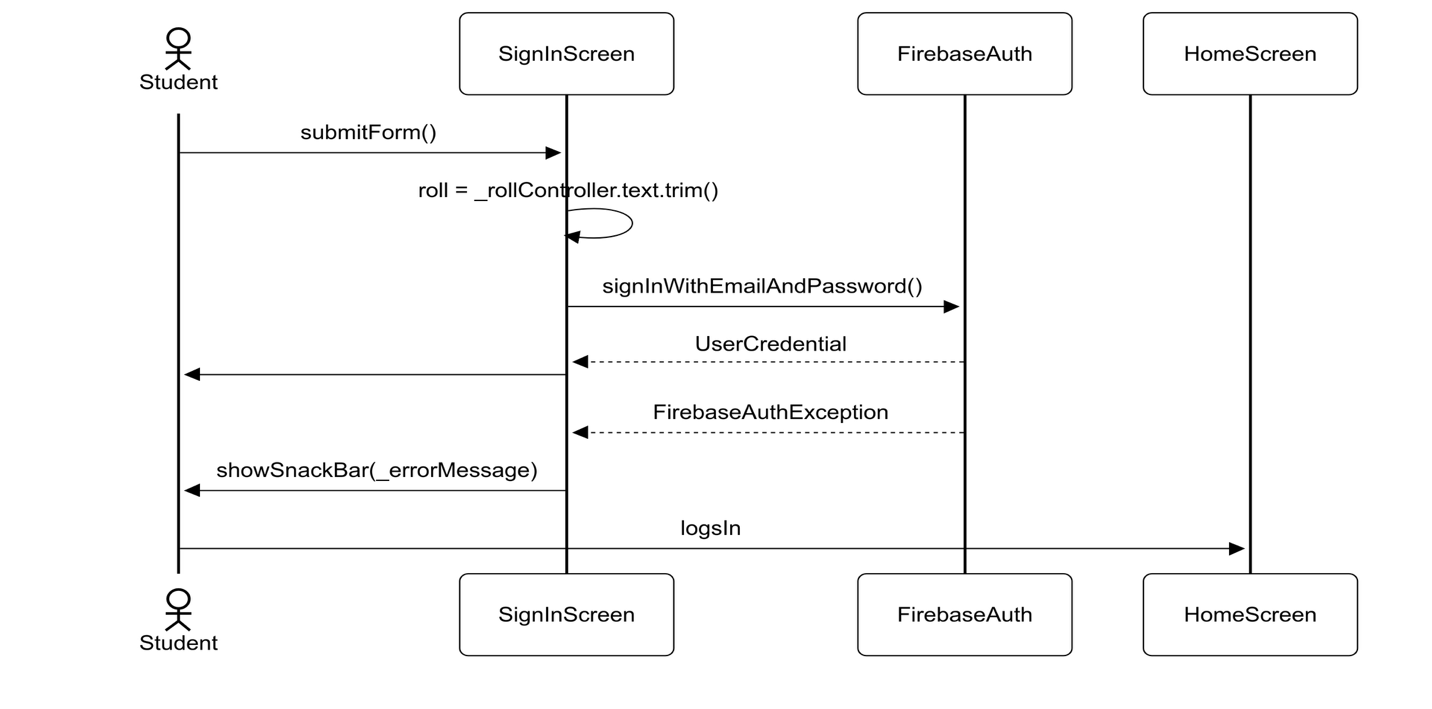
AI-generated content may be incorrect.

### Class Diagram

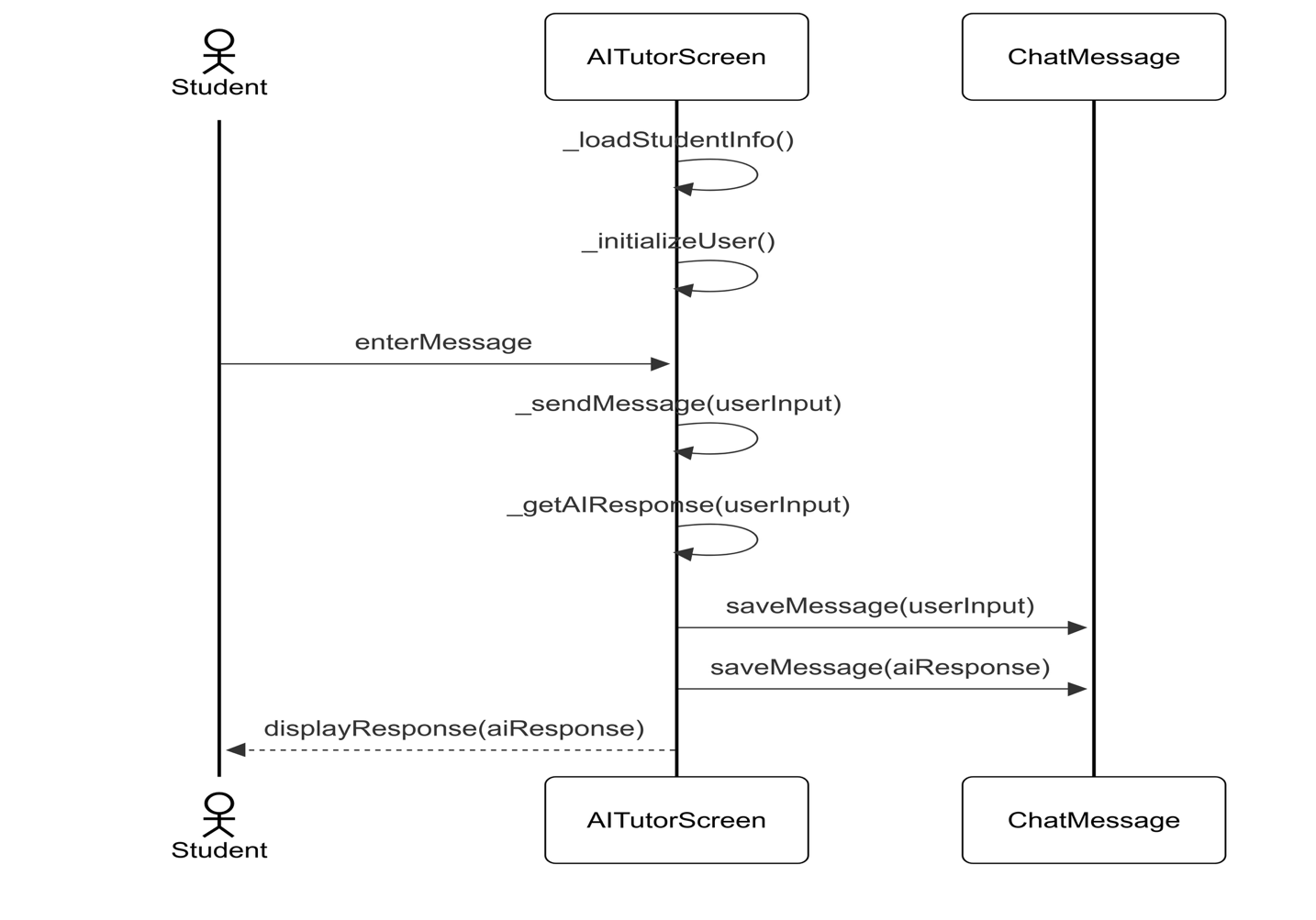


### Sequence Diagram

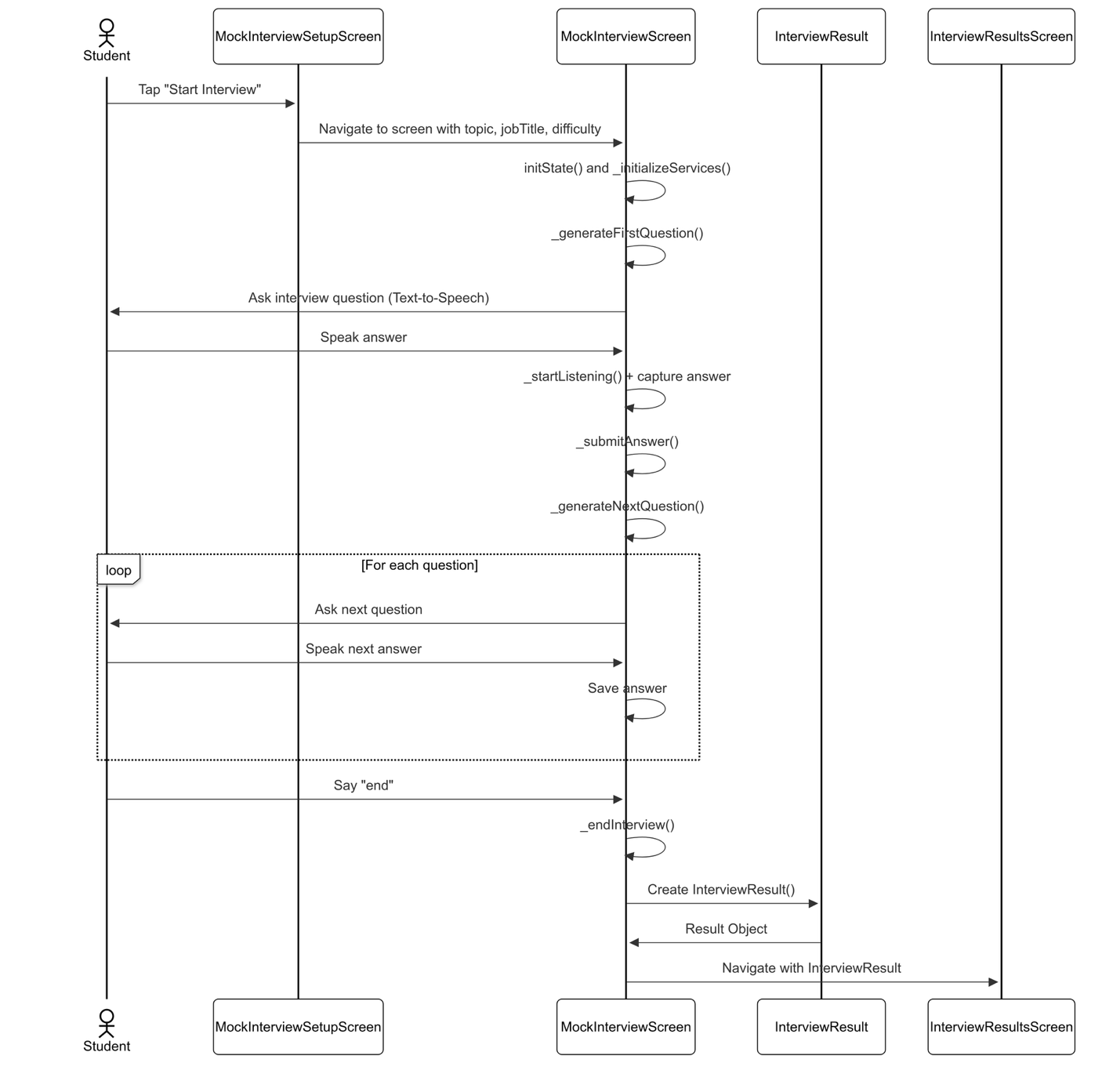
#### Login Sequence



#### Ai Tutor Sequence



#### Mock Interview Sequence



## Data Design

The data design of the **AI Student Advisor** system defines how the system's information domain (e.g., users, courses, sessions, tasks, and results) is stored, accessed, and processed. The application uses **Firebase Firestore**, a cloud-hosted NoSQL database, for all dynamic and user-specific data. Firestore collections and documents are organized based on user IDs and module responsibilities.

Each user has a unique document ID (roll number), under which personalized subcollections such asselected **courses, chat sessions, student data,** and**Todo tasks** are stored.

In addition, **Gemini AI**and**speech services** handle temporary runtime data such as messages, questions, and responses, which do not persist unless stored by design. Following db are used:

* **Firebase Authentication**: Stores secure login credentials and manages user sessions.
* **Firebase Firestore**: Stores user profile data, chat history, course selections, and to-do tasks.

### Data Dictionary

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity / Collection** | **Field** | **Type** | **Description** |
| **Users** | rollNumber | String | Unique identifier for each student (also used as document ID). |
|  | name | String | Full name of the student. |
|  | semester | String | Current semester (e.g., "7th", "6th"). |
|  | email | String | Email used for Firebase login. |
| **SelectedCourses** | courseName | String | Name of the selected course. |
|  | creditHours | int | Number of credit hours for this course. |
|  | type | String | Either "Core", "Elective”. |
| **ChatSessions** | sessionId | String | Unique ID for a chat session. |
|  | messages | List<Map> | List of messages exchanged with AI Tutor. |
|  | timestamp | Timestamp | Date/time of the session. |
| **MockInterviews** | interviewId | String | Unique ID for each interview session. |
|  | jobTitle | String | Role selected for the interview (e.g., “Web Developer”). |
|  | difficulty | String | Difficulty level (Easy/Medium/Hard). |
|  | score | double | Score generated by Gemini AI. |
|  | feedback | String | AI-generated summary of performance. |
|  | questionsAndAnswers | List<Map> | Each entry contains question, answer, timestamp. |
| **TodoTasks** | taskId | String | Unique ID of the task. |
|  | title | String | Task title/description. |
|  | priority | String | Priority level (High/Medium/Low). |
|  | isDone | Boolean | Completion status of the task. |
|  | createdAt | Timestamp | Task creation time. |

## Human Interface Design

The **AI Student Advisor** mobile application is designed with a **user-first approach**, ensuring that all academic features are accessible, interactive, and visually appealing. The system offers personalized interactions through a clean, modern interface built with Flutter, following **Material Design principles**. From the home dashboard, users can easily navigate to Course Selection, AI Tutor, and Mock Interview modules using visual cards and navigation icons.

The application ensures that:

* All main functions are **reachable within two taps** from the homepage.
* User feedback is provided via **Snackbars, loading indicators, confirmation dialogs,** and**AI responses.**
* Students receive real-time validation when uploading files, selecting courses, or engaging in AI-based activities.

Each screen is responsive, visually consistent, and supports basic accessibility (e.g., sufficient contrast, large touch targets, and voice interaction in interviews).

### Screen Images

Display screenshots showing the interface from the user’s perspective. These can be hand-drawn, or you can use an automated drawing tool. Just make them as accurate as possible. (Graph paper works well.)

### Screen Objects and Actions

|  |  |  |
| --- | --- | --- |
| **Screen Object** | **User Action** | **System Response / Feedback** |
| **TextField (RollNo/Password)** | Enter text | Stores input and perform validation on focus loss |
| **Login Button** | Tap | Authenticates user via Firebase and navigates to Home Page |
| **Card: AI Course Selection** | Tap | Navigates to Course Selection screen |
| **Card: AI Tutor** | Tap | Navigates to AI Tutor screen |
| **Card: Mock Interview** | Tap | Navigates to Interview Setup screen |
| **File Upload Button** | Select .txt/.csv/.pdf file from device | Uploads file to Firebase Storage and parses using Gemini API |
| **Course Checkbox** | Tap to select/unselect course | Adds or removes course from Firestore under user’s profile |
| **Send Button (AI Tutor)** | Enter text & tap send | Sends query to AI and shows AI response in chat |
| **Mic Icon (Mock Interview)** | Tap and speak | Transcribes answer using speech-to-text and shows real-time transcript |
| **Start Interview Button** | Tap | Loads and asks questions based on job title and difficulty |
| **Logout Icon (Top right)** | Tap | Displays confirmation dialog and logs user out on confirmation |
| **Add Task Button (Todo)** | Tap | Opens dialog to input task title and priority |
| **Checkbox (Task Item)** | Tap | Marks task as complete/incomplete and updates Firestore |
| **Delete Task Icon** | Tap | Removes the task from Firestore |

# Implementation

This chapter presents the implementation details of the AI Student Advisor system. The application is built using Flutter SDK for cross-platform mobile development, Firebase for backend services, and GeminiAPI for AI-based functionalities such as tutoring, course recommendations, and interview simulation. The source code follows proper coding standards and modular design principles, but only pseudocode is included here to describe core functionalities without exposing implementation specifics.

The major modules implemented in the system are:

* Login and Authentication
* Course Selection Assistant
* AI Tutor
* Mock Interviews using AI
* To-Do List
* Profile Management.



## Algorithm

Mention the algorithm(s) used in your project to get the work done with regards to major modules. Provide a pseudocode explanation regarding the functioning of the core features. Be sure to use the correct syntax and semantics for algorithm representations. Following are few examples of algorithms/pseudocode:

Table 13

|  |  |
| --- | --- |
| **Algorithm 1 Login and Authentication** | |
| **Input:** Roll Number and Password | |
| **Output:** it will log In to Home Screen | |
| 1:Function loginUser(RollNo, password):  2: If RollNo or password is empty:  3: Show error "Please fill in all fields"  4: Return  5: Try:  6: Authenticate using FirebaseAuth.signInWithRollNoAndPassword(RollNo, password)  7: If success:  8: Navigate to HomePage  9: Else:  10: Show "Invalid credentials" error  11: Catch error:  12: Show error message from Firebase | |
| **Algorithm 2 Course Selection Assistant** | |
| **Input:** Timetable and interests | |
| **Output:** courses acc to students interests | |
| 1:Function uploadTimetable(file):  2: Validate file type (.txt, .csv, .pdf)  3: Upload file to Firebase Storage  4: Extract text using OCR or file reader  5: Send parsed text to Gemini API for time slot extraction  6:Function recommendCourses(parsedSchedule):  7: Load all offered courses from Firestore  8: Filter courses based on:  9: - No timing conflict with parsed schedule  10: - Elective or improvement criteria  11: Return list of recommended courses | |
| **Algorithm 3 AI Tutor** |  |
| **Input:** Any data |  |
| **Output:** Results of that data |  |
| 1:Function askQuestion(userInput):  2: If userInput is empty:  3: Show error "Please enter a question"  4: Return  5: Show loading spinner  6: Send userInput to Gemini API  7: Receive AI response  8: Save question and response to Firestore  9: Display response in chat interface |  |
| **Algorithm 4 Mock Interviews using AI** | |
| **Input:** Interview Topic, Job title and Difficulty | |
| **Output:** Interview Questions, Feedback, Score | |
| 1:Function startMockInterview(jobTitle, difficultyLevel):  2: Send request to Gemini API for relevant interview questions  3: Display questions one by one  4:Function captureAnswerWithVoice():  5: Activate speech-to-text engine  6: Transcribe spoken answer  7: Display and store transcribed text  8:Function generateFeedback(allAnswers):  9: Send answers to Gemini API for evaluation  10: Receive feedback and score  11: Store results in Firestore  12: Display results to user | |
| **Algorithm 5 To-do List** | |
| **Input:** Task and its priority | |
| **Output:** saved to database | |
| 1:Function addTask(title, priority):  2: If title is empty:  3: Show error  4: Return  5: Create new task object  6: Store task in Firestore under current user  7:Function markTaskAsDone(taskId):  8: Update task’s 'isDone' field to true in Firestore  9:Function deleteTask(taskId):  10: Remove task document from Firestore | |
| **Algorithm 6 Profile Management** | |
| **Input:** Change Courses | |
| **Output:** Display enrolled courses nd student details | |
| 1: Function loadProfile(userId):  2: Fetch profile data from Firestore  3: Display on UI  4:Function updateProfile(userId, newData):  5: Validate newData  6: Update Firestore with new values  7: Show confirmation message | |

## External APIs/SDKs

Describe the third-party APIs/SDKs used in the project implementation in the following table. Few examples of APIs are provided in the table.

Table 14‎4 Details of APIs used in the project

|  |  |  |  |
| --- | --- | --- | --- |
| **Name of API and version** | **Description of API** | **Purpose of usage** | **List down the API endpoint/function/class in which it is used** |
| Gemini(2.0-Flash) | Ai Chatbots | Used for AI based Responses | class MockInterviewSetupScreen,  class CourseSelectionScreen,  class AITutorChat |
| Flutter SDK | Building cross-platform UI (Android & iOS) | For developing hybrid applications | It is used in overall project |

## User Interface

Details about user interface with descriptions. Provide the User Interface for each sub-system (such as Mobile App, Web App, Client App, Admin App). Provide description of each User Interface explaining the details.

When inserting User Interfaces, use appropriate size of the image, for example, for mobile app, 2-4 screens can be placed on a single page.

Following are few examples of User Interfaces:

### 5.3.1 Splash Screen 5.3.2 Login Screen

Splash Screen appears when student opens If the account is already made then student

the app. Just enter roll no. and password of the acc.

A screen shot of a logo

AI-generated content may be incorrect. A screenshot of a login screen

AI-generated content may be incorrect. Figure 1 Splash Screen Figure 2 login Screen

### 5.3.3 Signup Screen 5.3.4 Home Screen

Student create his account and then logins When the student signs in then the home

to the app. Screen appears where different modules are.

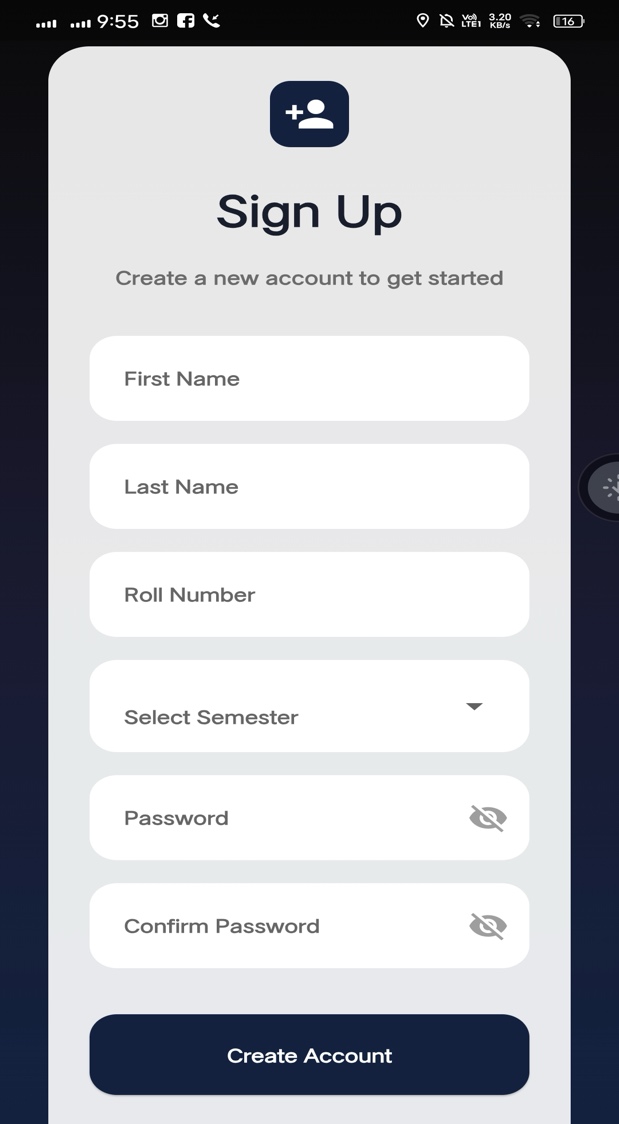
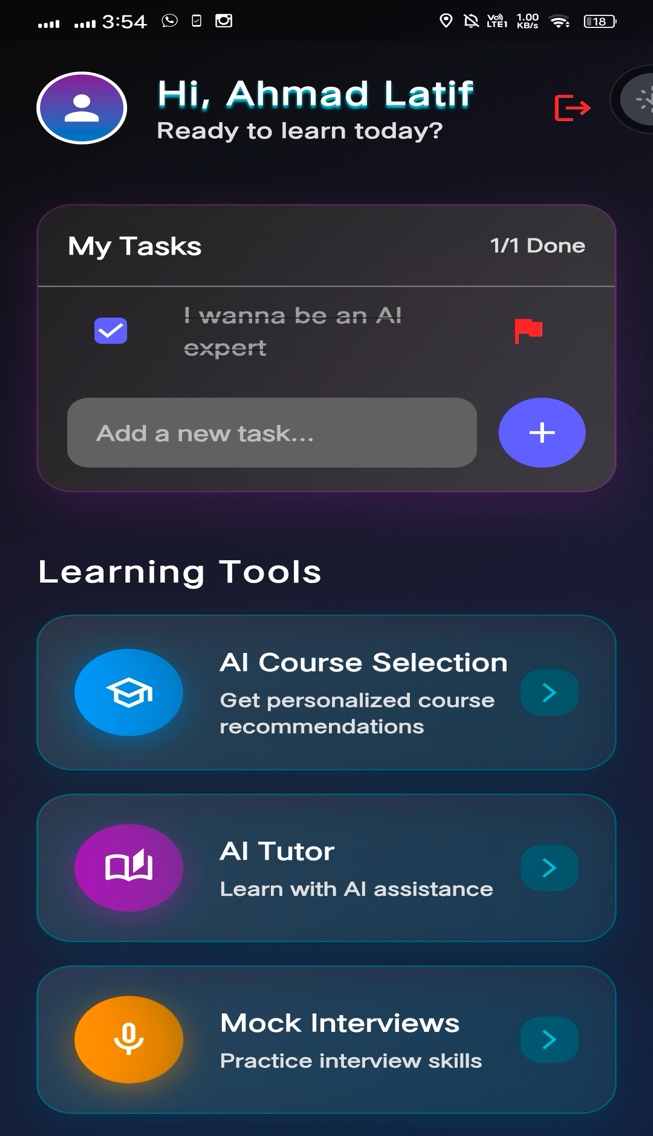
 

Figure 4 Signup Screen Figure 5 Home Screen

### 5.3.5 Course Selection Assistant 5.3.6 Ai Tutor

This module helps student to select This module help students to study

courses based on interest. effectively.

A screenshot of a computer

AI-generated content may be incorrect. 

Figure 6 Course Selection Assistant Figure 7 Ai Tutor

### 5.3.7 Mock Interviews using Ai 5.3.8 Profile Page

This module helps student to prepare This page displays the student details and

for the future by interview sessions. The courses in which he/she is enrolled.

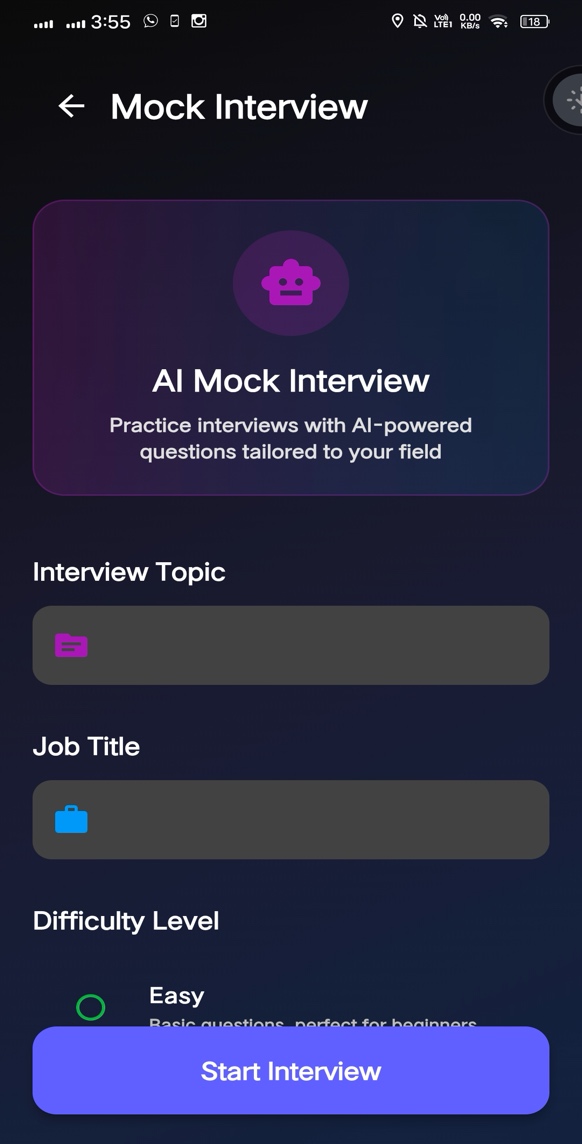
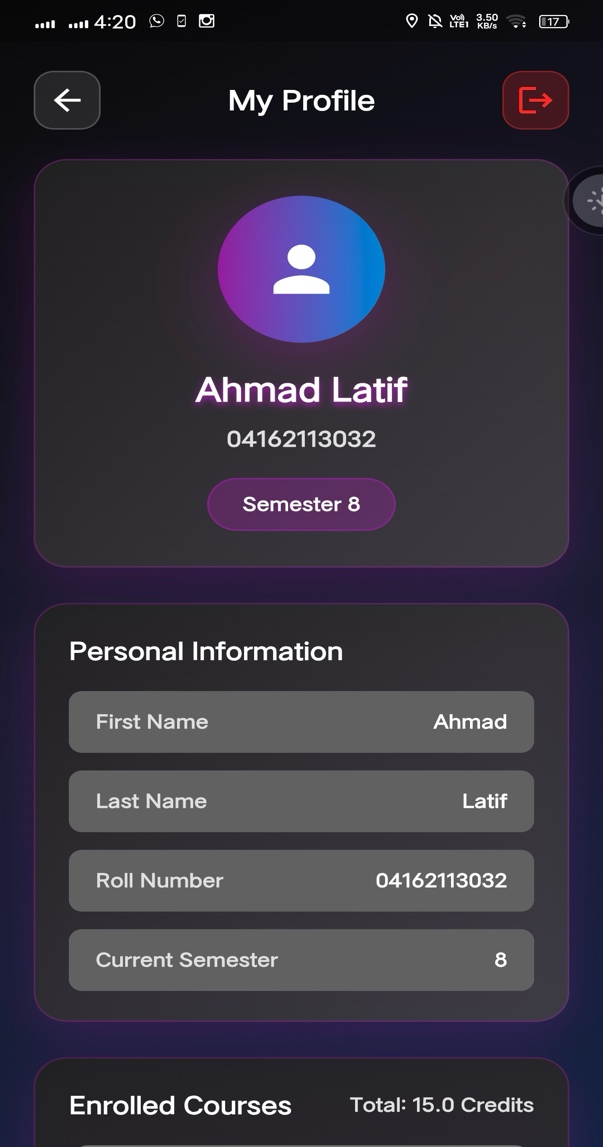
 

Figure 8 Mock Interviews using Ai Figure 9 Home Screen

# Testing and Evaluation



After the successful development of the **AI Student Advisor**, a series of rigorous tests were conducted to ensure that each module performs correctly and meets the functional and non-functional requirements. Testing helps identify and eliminate bugs, validate individual module behavior, and verify proper integration of the system components before final deployment..

## Unit Testing

It’s a level of software testing where individual units of a software/component are tested. The purpose is to validate that each unit of the software performs as designed.

**Unit Testing 1:** Login as Student with valid and invalid credentials

**Testing Objective:** To ensure the login form is working correctly with valid and invalid credentials/inputs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1 | Check the Roll Number field of login to validate that it takes proper email | Roll Number:  04162113032 | Validates Roll Number address and moves cursor to next textbox | Pass |
| 2 | Check the Roll Number field of login to validate that it displays error message. | Roll Number: 03162113032 | Highlights field and displays error message | Pass |
| 3 | Submit form with empty fields | “” (Empty) | Displays error message: “Please fill all fields” | Pass |
| 4 | Login with wrong password | Roll Number: 04162113032  Password: “123” | Shows: “Invalid password” | Pass |
| 5 | Login with correct password | Roll Number: 04162113032  Password: “12345678” | Redirects to Home Page | Pass |

## Functional Testing

Functional testing was performed after the unit testing phase to verify that the complete system performs according to its defined specifications. Each module’s functionality was validated using real inputs and interaction scenarios. This ensures the system meets the user requirements and delivers the intended behavior.

**Functional Testing 1:** Course Selection Assistant

**Objective**: To ensure the timetable upload, parsing, and course recommendation functionalities behave as expected.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Test Case / Script** | **Input / Action** | **Expected Result** | **Actual Result** | **Status** |
| 1 | Load Course Selection screen | Tap “Course Selection” on dashboard | Screen loads with upload interface and title | As expected | Pass |
| 2 | Upload a valid timetable file | Upload .pdf timetable | File uploads successfully, filename displayed | As expected | Pass |
| 3 | Upload invalid file | Upload .jpg or unsupported format | Show error: “Invalid file type” | Error shown correctly | Pass |
| 4 | Parse uploaded timetable | Tap parse button | Schedule extracted and displayed | Schedule shown with slots | Pass |
| 5 | Recommend courses | Tap “Recommend” | Display non-conflicting elective or improvement courses | List of valid courses shown | Pass |
| 6 | Add course to selection | Tap on checkbox beside course | Course marked and stored in Firestore | Course added to database | Pass |

**Functional Testing 2:** Ai Tutor

**Objective**: To verify if students can send academic queries and receive meaningful AI responses.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Test Case / Script** | **Input / Action** | **Expected Result** | **Actual Result** | **Status** |
| 1 | Access AI Tutor screen | Tap “AI Tutor” on homepage | Loads chat UI with input field and send icon | Chat UI loaded | Pass |
| 2 | Send empty question | Leave input empty, tap send | Error message: “Please enter a question” | Message shown | Pass |
| 3 | Send valid question | “What is recursion?” | Gemini responds with accurate explanation | AI responded in chat | Pass |
| 4 | Submit long technical query | Input complex ML concept | Gemini processes and provides a clear breakdown | Response accurate | Pass |
| 5 | Session saved | Ask 2–3 questions in a session | All Q&A saved to Firestore for future reference | Data stored | Pass |

**Functional Testing 2:** Mock Interviews Using AI

**Objective**: To ensure job-specific mock interviews can be conducted and feedback generated using AI.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Test Case / Script** | **Input / Action** | **Expected Result** | **Actual Result** | **Status** |
| 1 | Navigate to Mock Interview | Tap on “Mock Interview” on dashboard | Setup screen loads with job title and difficulty dropdown | Screen loaded | Pass |
| 2 | Start Interview | Job: “Mobile App Dev”, Level: “Medium” | Gemini AI begins asking questions via TTS | Interview started | Pass |
| 3 | Answer via voice | Speak into mic | Answer transcribed and displayed on screen | Accurate transcription | Pass |
| 4 | Complete interview | Tap submit | Gemini provides score and feedback summary | Score and review displayed | Pass |

## Business Rules Testing

Business rule testing validates that system decisions follow predefined logic based on inputs and conditions. This testing uses **decision table-based testing**, where multiple combinations of input conditions are mapped to expected outcomes (actions). These rules are defined earlier in the Functional Requirements (FRs) and Use Cases.

Below is a **decision table** for the course recommendation feature, which follows a business rule:

### Business Rule 1

**"Recommend only those courses that do not conflict with the student's existing schedule and belong to elective/improvement types."**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Conditions** | **Rule 1** | **Rule 2** | **Rule 3** | **Rule 4** |
| Course timing overlaps with student's timetable | No | Yes | Yes | No |
| Course is marked as "elective" or "improvement" | Yes | Yes | No | No |
| Student has credit hour capacity left | Yes | Yes | Yes | No |
|  |  |  |  |  |
| **Actions** |  |  |  |  |
| Recommend course | Yes | No | No | No |
| Show conflict warning message | No | Yes | Yes | Yes |

### Business Rule 2

"A task can be created only if it has a title and a selected priority (High/Medium/Low).”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Conditions** | **Rule 1** | **Rule 2** | **Rule 3** | **Rule 4** |
| Task title is not empty | Yes | Yes | No | No |
| Priority is selected | Yes | No | Yes | No |
|  |  |  |  |  |
| **Actions** |  |  |  |  |
| Save task to Firestore | Yes | No | No | No |
| Show error "Priority required" | No | Yes | No | Yes |
| Show error "Task title required" | No | No | Yes | Yes |

### Business Rule 3

“The AI Tutor should only respond if the input is valid and internet connection is available.”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Conditions** | **Rule 1** | **Rule 2** | **Rule 3** | **Rule 4** |
| Input text is not empty | Yes | Yes | No | No |
| Internet connection is available | Yes | No | Yes | No |
|  |  |  |  |  |
| **Actions** |  |  |  |  |
| Send query to Gemini API | Yes | No | No | No |
| Display AI response | Yes | No | No | No |
| Show "No internet" error | No | Yes | No | Yes |
| Show "Empty input" warning | No | No | Yes | Yes |

## Integration Testing

Integration tests assess whether a set of classes that must work together do so without error. They

ensure that the interfaces and linkages between different parts of the system work properly. At this point, the classes have passed their individual unit tests, so the focus now is on the flow of control among the classes and on the data exchanged among them. Integration testing follows the same general procedures as unit testing: The tester develops a test plan that has a series of tests, which, in turn, have a test. Integration testing is often done by a set of programmers and/or systems analysts.

### ****Integration Testing 1: Course Selection & Timetable Parsing****

**Objective:** To ensure that the uploaded timetable is correctly parsed and that the AI-based course recommendation engine interacts properly with Firestore and the Gemini API.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Test Case / Script** | **Attribute & Value** | **Expected Result** | **Actual Result** | **Status** |
| 1 | Upload and Parse Timetable | File: sem7\_schedule.pdf | Timetable parsed and displayed on screen | File parsed, schedule shown | Pass |
| 2 | Recommend Courses based on Timetable | Parsed schedule object | Recommended elective courses with no timing conflicts shown | Course list shown correctly | Pass |
| 3 | Add a Recommended Course | Course: AI Ethics - Wed 11AM | Course stored under user profile in Firestore | Stored and displayed in user dashboard | Pass |

### ****Integration Testing 2: AI Tutor + Chat History****

**Objective:** To ensure that the AI Tutor module integrates properly with the API and stores the conversation in Firestore.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Test Case / Script** | **Attribute & Value** | **Expected Result** | **Actual Result** | **Status** |
| 1 | Send a tutoring query | Question: "Explain backpropagation" | Gemini API responds and displays response in chat | Response received and shown | Pass |
| 2 | Auto-save conversation to Firestore | Session ID: UID12345 | Q&A saved in Firestore under user chat history | Data saved correctly | Pass |
| 3 | Retrieve saved chat | Login and access AI Tutor | Previously asked questions are shown in session history | Past conversation displayed | Pass |

### ****Integration Testing 3: Mock Interview + Voice Input + Feedback****

**Objective:** To verify that the voice recording, AI question evaluation, and score generation work seamlessly together.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Test Case / Script** | **Attribute & Value** | **Expected Result** | **Actual Result** | **Status** |
| 1 | Start Mock Interview | Role: Mobile App Dev, Level: Medium | Interview starts and first question is read aloud | TTS works, question spoken aloud | Pass |
| 2 | Respond with voice | Answer: "I use Flutter for cross-platform apps" | Voice recorded and transcribed correctly | Answer displayed as text | Pass |
| 3 | Submit all answers | 5 questions completed | Gemini API evaluates and generates score + feedback | Score and feedback shown | Pass |
| 4 | Save results to Firestore | User: 04162113032 | Interview result saved under profile | Saved successfully | Pass |

# Conclusion and Future Work

This chapter summarizes the project achievements and outlines possible enhancements for future development. The goal of the AI Student Advisor project was to create an intelligent academic assistant that helps university students with course selection, tutoring, and interview preparation. The system integrates AI capabilities with Firebase backend services to provide personalized, efficient, and scalable support to students throughout their academic journey.



## Conclusion

The **AI Student Advisor**application successfully demonstrates how artificial intelligence can be utilized to enhance the student experience in higher education. The core features developed include:

* **Course Selection Assistant** – helps students choose elective or improvement courses based on their timetable availability.
* **AI Tutor** – provides intelligent, instant academic assistance using natural language input.
* **Mock Interviews using Ai** – simulates job interviews and offers feedback to improve student performance.
* **To-Do Task Manager** – helps students manage their academic tasks and priorities efficiently.

Throughout the project, modern tools such as **Flutter SDK**, **Firebase**,and **Google's Gemini API** were used to ensure a seamless cross-platform experience. The system was rigorously tested through unit, functional, integration, and business rule testing, ensuring its stability and usability.

This project provides a solid foundation for a fully scalable academic advisor platform. The feedback collected during development also helped validate its usefulness and revealed areas of potential improvement.

## Future Work

Though the initial version of the AI Student Advisor meets its fundamental objectives, there is significant potential for further enhancement and expansion. Some of the areas identified for future work include:

* **Multilingual Support:** Adding support for local languages and dialects to improve accessibility for non-English speakers.
* **Human-AI Hybrid Counseling:** Integrating a human advisor dashboard to allow collaborative advising with real counselors.
* **Voice-Enabled Navigation:** Improving accessibility for differently-abled users by enabling full voice interaction and screen reading.
* **Improved AI Feedback:** Enhancing mock interview scoring by training a domain-specific evaluation model.
* **Analytics Dashboard:** Adding dashboards for admin and faculty to monitor student engagement, progress, and trends.
* **Role-Based Control Panel:** Adding dedicated panels for different users like Admins, Advisors, or Faculty.

The project will be further developed by Muzamil, who will take ownership of the next phase. He intends to enhance the application's functionality, improve user experience, and possibly integrate it with institutional learning systems for broader deployment.

# References

1. Google. “Firebase Documentation.”  
   Internet: https://firebase.google.com/docs, [Accessed: July 10, 2025].
2. OpenAI. “Gemini AI API.”  
   Internet: [https://ai.google.dev](https://ai.google.dev/), [Accessed: July 8, 2025].
3. Flutter. “Flutter Official Documentation.”  
   Internet: https://docs.flutter.dev, [Accessed: July 7, 2025].
4. W.K. Chen. Linear Networks and Systems. Belmont, CA: Wadsworth, 1993, pp. 123–135.
5. G. Pevere. “Infrared Nation.” The International Journal of Infrared Design, vol. 33, pp. 56–99, Jan. 1979.
6. W.K. Chen. *Linear Networks and Systems*. Belmont, CA: Wadsworth, 1993, pp. 123–135. *(If included for course relevance or format requirement)*