Automation Bot: Ria

MUHAMMAD YAMAN
01-134221-058
SHAYAN MANSOOR
01-134221-074

Supervisor: Qazi Haseeb Yousaf

Co-Supervisor: MUHAMMAD ATHESHAM NOOR

Bachelor of Science in Computer Science

Department of Computer Science
Bahria University, Islamabad

December 2025

Certificate

We accept the work contained in the report titled “Automation Bot: Ria”, written by Mr. Muhammad Yaman AND Mr. Shayan Mansoor as a confirmation to the required standard for the partial fulfillment of the degree of Bachelor of Science in Computer Science.

Approved by . . . :

Supervisor: Qazi Haseeb Yousaf

_____

Internal Examiner:

_____

External Examiner:

_____

Project Coordinator: Dr ASIM ALI

_____

Head of the Department: DR MUHAMMAD KHURRAM EHSAN

_____

Abstract

Service usage in university academic systems in Pakistan still depends heavily on manual checking of portals, repeated logins, and informal WhatsApp groups for important updates. Students frequently visit CMS to see attendance, new assignments, and deadlines, and they rely on classmates to forward screenshots and messages. This scattered, time-consuming process causes missed notifications, late submissions, and extra stress, especially during peak academic load.

The AI Automation Bot "RIA" addresses this gap by connecting students with the Bahria University CMS through an intelligent, automated assistant. The system uses an AI agent that can understand natural language queries, log in to CMS on behalf of the user, scrape attendance and assignment data, and provide clear, conversational answers through WhatsApp. It offers features such as attendance summaries, detailed analysis with warnings, teacher information, assignment download and upload guidance, and continuous monitoring with notifications for changes. To ensure reliability even when WhatsApp is unavailable, a backup mobile application named "Ria", developed in Flutter, provides access to the same academic information and alerts through a separate app interface.

The backend is implemented using Python, LangChain, and Google Gemini for AI-driven responses, along with browser automation for secure data retrieval and JSON-based storage for monitoring state. By combining automation, AI, and a cross-platform mobile app, this project aims to streamline students' interaction with the university CMS, reduce repetitive manual work, and make academic information more accessible, timely, and organized for Bahria University students.

Acknowledgments

We extend our sincere gratitude to Allah Almighty for blessing us with the opportunity, strength, and patience to work on this project. His guidance and mercy have enabled us to reach this stage and complete our efforts.
We are deeply thankful to our respected supervisor, Sir Qazi Haseeb Yousaf, for his continuous support, valuable feedback, and encouragement throughout this journey. His guidance, trust, and insightful suggestions have greatly helped us in shaping and improving our project. It has been an honor and privilege to work under his supervision.
We would also like to express our heartfelt thanks to our parents and families for their constant prayers, motivation, and support. Their love, sacrifices, and belief in us have been a source of strength and inspiration during every phase of this project and our academic life.

MUHAMMAD YAMAN
SHAYAN MANSOOR
Bahria University
E-8, Islamabad, Pakistan

December 2025

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

AI	Artificial Intelligence
LLM	Large Language Model
SDK	Software Development Kit
NLP	Natural Language Processing
API	Application Programming Interface
BU	Bahria University
REST	Representational State Transfer (Web services style)
JSON	JavaScript Object Notation
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
DOM	Document Object Model
RFC	Request for Comments (Internet documents)
WABA	WhatsApp Business API
SMS	Short Message Service
MMS	Multimedia Messaging Service
OS	Operating System
UI	User Interface
UX	User Experience

Chapter 1

Introduction

The academic ecosystem of Bahria University is having a major problem of providing students with timely academic information via the Course Management System (CMS). Students now find it difficult to use the manual form of constantly accessing the Bahria CMS portal [1] to check their attendance records, updates on assignments and course material. This ineffective method causes the late realization of important change in academic practices, late deadlines and poor academic achievement among the university community.

The Bahria University Intelligent Academic Assistant is an institution-specific solution which incorporates a combination of artificial intelligence and the current CMS infrastructure of the university. This system offers automatic tracking of the progress of academics and sends real-time alerts via the contemporary media. The platform was designed specifically to the academic setting at Bahria University and thus it perfectly fits the operational needs and structure of the university with its own CMS.

The present research project indicates a high level of improvement in educational technology which is specifically designed to the needs of Bahria University. The system tackles key communication gaps in the academic structure of the university, in addition to improving the interaction with students and efficiency in the institutions. Being a final year project that is being developed at Bahria University, this project is an expression of our effort to make practical solutions that would help the university community directly.

1.1 Problem Description

The Course Management System in Bahria University poses significant issues of accessibility among the students as the system has to be used manually. The existing system also requires the students to visit the Bahria CMS portal numerous times to look at the academic updates, which poses too much inefficiency in the process of flow of information.

This is a particular issue in the implementation of CMS at the Bahria University, and it directly affects the academic life of students in the university.

Manual monitoring process gives rise to several operational problems in the academic setting of Bahria University. Students have to use the proprietary CMS interface of the university several times per day not to miss important academic updates. These involve verification of attendance records changes, new assignment releases, assignment deadline alterations and grade releases which are specific to the course structure of Bahria University.

The operational limits produce negative implications on the academic performance of students in Bahria University. Lack of automated notification systems compels the students to repetitive checking behaviors which occupy their precious academic time. Moreover, the lateness in recognizing academic change peculiar to the pattern of scheduling and assessment at Bahria University may affect student performance and institutional satisfaction in a negative way.

1.2 Project Objectives

The objective of this research is to create a smart academic aid system that is unique to Bahria University and changes the way in which students engage with the Course Management System of the university. The main goal here is to develop a fully-fledged solution, which will automatize the academic monitoring process and provide the proactive notification channels that have been optimized in relation to the student population of Bahria University.

Institutional-specific aims of the project are:

- Creating an AI agent with the purpose of comprehending the academic terminology and courses of Bahria University.
- Introducing automated monitoring services that are able to monitor changes in academic aspects in the CMS of Bahria University.
- The creation of a notification mechanism based on the academic schedule of Bahria University and its working patterns.
- Developing a file management system that supports the assignment submission procedures at Bahria University.
- Developing security controls that meet the data protection policies of Bahria University.

These goals aim to improve the accessibility of academic information to the students of Bahria University in particular, giving them real-time information on any academic changes happening within the university.

1.3 Project Scope

It is a project where an academic assistance platform is developed which has its boundaries of functioning that are well established within the Bahria University ecosystem. Components are used in the system architecture and they were designed specifically to interface with Bahria University CMS and academic protocols.

The functional aspects of the project scope are:

- **Bahria CMS Integration:** The Bahria University Course Management System is architecture specific services that are automated.
- **University-Terminology AI:** AI based on natural language processing that was optimized on course codes, department names and academic terminology of the Bahria University.
- **Institutional Notification System:** Alert systems would be aligned with the academic calendar and operations of a Bahria University.
- **University File Handling:** Document administration, according to the requirements of the submission and file format in the Bahria University.
- **Bahria-Specific Analytics:** Reporting applications generated insights, which can be applied to the academic programs of Bahria University.

Institutional parameters that the project implementation is established are:

- The integration is unique with Course Management System of Bahria University.
- Bahria University academic data structure and course hierarchy sharing.
- Bahria University data confidentiality and institutional policy.
- Optimization of academic calendar and working hours of the Bahria University.

Chapter 2

Literature Review

This section analyses in detail the available literature and technologies that can be used in the creation of educational chatbots and academic assistants powered by AI. The systematic review of the existing studies in the field of educational technology, conversational AI systems, and automated solutions in academic monitoring provides the framework of the Bahria University AI-Powered WhatsApp Bot. Through critical analysis of the current implementations and gaps in research, this chapter addresses the present project in the wider context of innovations in educational technology.

The literature review addresses several important objectives, proving knowledge of the present situation in the field, explaining the deficiency of the existing solutions, creating a theoretical framework of the given research, and creating a strong argument in favor of the innovative character of the offered system. This review, with the help of the systematic analysis of previous literature, demonstrates that this project makes certain contributions to the area of educational technology.

2.1 Introduction

Artificial intelligence has changed the way students receive academic information all over the world since it is integrated into educational systems. A number of technological solutions have been widely implemented in education institutions to improve the experience of students across the world, where course management system (CMS) and learning management system (LMS) has become a commonplace infrastructure. Nevertheless, there are still major problems associated with providing timely academic information on these conventional sources.

The advent of chatbots and messaging applications has presented novel possibilities of academic support. With a following of more than two billion users, WhatsApp is the product to facilitates the transition between the old academic system and the new student communication trends. Although this is a possibility, the majority of current educational

chatbots are limited in their functionality, and they are usually rule-based systems that do not integrate with real-time academic data.

The literature review focuses on the present condition of educational chatbots, academic web automation, conversational agent based on AI and integration of messaging platforms. The analysis presents certain drawbacks of the current implementations and determines the research gap that the innovativeness of the Bahria University AI-Powered WhatsApp Bot intends to fill with its holistic solution, which is a combination of deep integration of CMS, active monitoring, and conversational AI.

2.2 Comparative Analysis

2.2.1 Chatbot Systems in Education

The first educational chatbots were mainly rule-based and had very limited capabilities. These systems were only able to deal with simple frequently asked questions but were not able to deal with complex multi-step academic queries. The study has shown that whereas rule-based chatbots will deliver a consistent reply to fixed information, they are not able to retrieve dynamically changing academic facts or to hold on to the conversation turn context.

2.2.2 Web Automation in Education

Scraping of web pages and automation have been commonly used in education settings in retrieving data. It has been demonstrated that such tools as Selenium WebDriver [2] can be successfully used to automate portal access to grade checking and attendance monitoring. Nonetheless, the implementations are usually independent applications which lack user-friendly interfaces or connection to chatbots.

2.2.3 AI-Based Conversational Agents

Large language models combined with agent frameworks like LangChain [3] have made it possible to have more advanced educational assistants. Current AI systems utilizing Google's Generative AI [4] are able to comprehend natural language, provide context in a conversation, and respond like a human being. Most implementations however concentrate on content delivery, not on transactional operations that need real-time access to data through authenticated academic portals.

2.2.4 WhatsApp API Integrations

Educational establishments have used WhatsApp API through services like Twilio [5] to send out notifications and announcements, though usually used as a one-directional

system. Two-way chatbots that assist in personalized academic questions using WhatsApp are rather rare, and the current examples are typically limited to the rule-based interaction without the artificial intelligence.

2.2.5 Proactive Monitoring Systems

Although there are several monitoring tools that can be used to monitor the activities of a webpage, they have not found much application in academic monitoring. Majority of systems have been designed with a commercial orientation as opposed to educational settings, and not many of them combine monitoring features with conversational AI using messaging applications.

2.2.6 Comparative Analysis Table

Table 2.1: Comparative Analysis of Educational Technologies and Proposed System

Feature	**Rule-Based Chatbots**	**Web Automation Tools**	**AI Educational Assistants**	**WhatsApp Notifications**	**Proposed System**
Natural Language Understanding	Limited	None	Advanced	None	Advanced AI
Real-time Data Access	No	Yes	Limited	No	Yes
CMS Integration	No	Yes	Limited	No	Deep Integration
Proactive Monitoring	No	Manual	No	One-way	Automated
WhatsApp Integration	Basic	No	Limited	One-way	Bidirectional
Multi-Platform Support	No	No	Limited	WhatsApp only	WhatsApp + Mobile App

2.3 Summary

According to the literature review, the existing implementations of educational technologies are somewhat deficient in various aspects which are being filled by the Bahria University AI-Powered WhatsApp Bot. Current systems are usually narrow-minded in addressing single-facet elements of academic support without offering wholesome solutions. Chatbots that are operated by rules do not have the intelligence and real-time data access, web automation tools do not provide easy-to-use interfaces, AI assistants do not integrate with the institution, and implementations of WhatsApp do not have a bidirectional feature.

The system presented is a significant improvement as it will incorporate various technologies in a unified solution. It brings together the smartness of artificial intelligence-driven chatbots and the consistency of web automation, the convenience of WhatsApp messaging, and the proactive nature of automated surveillance. This combined methodology specifically responds to the weaknesses found in the existing implementations and the strength of each technology component.

The Bahria University AI-Powered WhatsApp Bot fills the specified gaps, creating a new benchmark of academic assistance systems. In the next chapters, the implementation of the given integrated approach is described in terms of technical implementation, testing, and evaluation of the effectiveness of this approach to meet the particular academic information needs of the students of Bahria University.

Chapter 3

Requirement Specifications

The AI-based Bahria University Assistant project will involve the creation of a two-channel notification device that will help students not miss valuable academic information. Those changes in the system must be constantly observed in Bahria CMS in terms of attendance, postings of assignments, modification of deadlines, and marks. It must have the ability to send an instant warning by both WhatsApp and Ria mobile app, establishing a backup system that is also reliable when one of the two platforms is malfunctioning. The AI element will have to comprehend the natural language queries regarding academic success and offer clever analysis of the attendance data. The system implemented using Flask [6] must be capable of dealing with file transactions such as downloading and submitting assignments in the real-time in the form of a chat interface. It should ensure real-time integrations using webhook mechanisms [7] between the two notification media as well as offer a consistent user experience on various platforms. Reliability is also essential and automatic backup mechanisms which guarantee the availability of services in case of individual platform failures.

3.1 Existing System

The existing academic ecosystem in Bahria University follows the old fashioned manual checks model that subjects the whole process of information retrieval to students. The students have to go through the Bahria CMS web portal many times during the day and navigate their way through numerous layers of menu options to find their attendance records, assignment information and academic updates. This has to be a continuous process of monitoring and keeping in mind to check on the system on a regular basis as there is no automatic alert or notification of any major changes. Faculty marked attendance causes students to be uninformed until they log in manually and visit the attendance section, and the shortage of attendance is usually not noticed promptly. In the case of assignment management, the system requires computer access and the use of a browser

with unique requirements, which cannot be used by students whose primary devices are mobile devices. The interface is not intelligent - students have to manually calculate their attendance rates, see due dates of assignments, and see courses which they need to take the closest. Such a reactive approach leads to realizations at the last minute regarding academic status, one or two-assignment due dates, and unwarranted stress. The whole process is tedious, ineffective and does not take advantage of the modern communication channels that students deal with every day, there is high disparity between the digital tools that the students embrace and the academic systems that they are required to embrace.

3.2 Proposed System

Our groundbreaking AI-based service will turn academic management into a non-manual task into a smart and active collaboration that will work around the clock on the student side. The system is monitoring Bahria CMS 24/7, whereby the attendance change, new assignment posting, deadline changes, and the mark change are automatically identified and updated in real-time. It supports two-way notifications via a sophisticated dual-channel notification system, which in turn immediately forwards notifications to WhatsApp and the Ria mobile app at the same time, making sure that the messages are delivered despite technical failures of one of the platforms. The inbuilt AI assistant is able to comprehend natural language queries, so the students can just say, show me my attendance or what courses I need to attend and the smart and analyzed responses will be shown, with relevant recommendations to the problem. In addition to notifications, the system allows full control of assignment lifecycle, using chat interfaces - the students can get assigned file, submit work, and even delete submissions without ever having to access computer. The AI element offers proactive academic feedback through automatic analysis of the attendance pattern, the courses with a low safe level, and recommending how to improve the situation. This produces a smooth, 24/7 academic assistant that is proactive and responsive not reactive, means it is able to anticipate the needs of students instead of having to wait until they request its services. By offering the institutional systems to the student lifestyles, the system will help close the gap between the institution systems and the students lifestyles by providing an easy way of getting the academics information, a way that they are used to in their daily lives hence making academic management easy, proactive and stress free.

3.3 Functional Requirements

The functional requirements are the statements of the behavior and functionality that the system will be called to execute in order to attain the basic objectives of the system.

Table 3.1: Functional Requirements

Requirement ID	**Requirement Description**
FR-1	The system should be able to know when students request the attendance in regular words such as show my attendance or check my classes.
FR-2	The system should automatically login to Bahria CMS and retrieve attendance information of all courses.
FR-3	The system should be able to compute a percentage of attendance and indicate the courses that have less than 75% percent attendance.
FR-4	The system should allow students to download files containing assignment data via WhatsApp.
FR-5	The system should enable the students to use WhatsApp to upload and submit assignments.
FR-6	The system should enable students to remove assignments that they have already submitted.
FR-7	The system has to keep on monitoring the attendance changes in the background.
FR-8	The system should automatically keep watch of new assignments and change of deadlines.
FR-9	The system should notify by WhatsApp in case of attendance changes or the introduction of new assignments.
FR-10	The system should be able to notify the same via the Ria mobile application.
FR-11	The system should recall the past dialogues with every student.
FR-12	The system should be able to examine the attendance trends and provide suggestions on how they should be improved.
FR-13	The system has to deal with several students using it simultaneously.
FR-14	The system should be able to handle various files to be assigned: PDF, Word, Excel, and ZIP files.

3.4 Non-Functional Requirements

These requirements establish the quality attributes and constraints within which the system should be able to perform.

Table 3.2: Non-Functional Requirements

Requirement ID	Requirement Description
NFR-1	The system should be able to respond to student queries in 5-10 seconds.
NFR-2	The system should be 24/7, 7 days a week.
NFR-3	The system should be well functioning even when a large number of students are using it simultaneously.
NFR-4	The system should store student log information and data in a safe and secure way.
NFR-5	The system should also be user-friendly to every student.
NFR-6	The system should restart on its own in case of a problem.
NFR-7	The system will be required to support various phones and web browsers.
NFR-8	The system should provide alerts within 1 minute upon changes occurring in CMS.
NFR-9	The system should display the same data as the official Bahria CMS.
NFR-10	The system should display obvious error messages where something does not work.
NFR-11	The mobile users should use very minimal internet data on the system.
NFR-13	The system should automatically remove idle or inactive users to protect the system.
NFR-14	The system should also have alternative ways of notification in case one of the channels stops.

3.5 Hardware and Software Requirements

This section identifies the environment and technological stack that must be used during development and deployment.

3.5.1 Development Environment

The following development environment has been maintained throughout the project development process:

Table 3.3: Minimum System Requirements

Component	**Minimum Requirement**
Processor	Intel Core i3
RAM	4 GB / 8 GB
GPU	Not required (LLM calls are API-based)
Storage	50 GB SSD
Internet	Required for Twilio, Gemini API, and CMS access

3.5.2 End User Requirements

The requirements for the end user, who is the potential client in our case, are given in Table 3.4.

Table 3.4: Minimum Client-Side Requirements

Component	**Minimum Requirement**
Device	Smartphone
Application	WhatsApp
Internet	Stable mobile data or Wi-Fi

3.5.3 Backend Stack

Table 3.5 shows the back-end development tools.

Table 3.5: Software Stack and Tools

Component	**Technology / Tool**
Programming Language	Python 3.9+
Framework	Flask / FastAPI
Automation Tool	Selenium WebDriver
Web Browser Driver	ChromeDriver
AI/LLM Framework	LangChain
LLM Provider	Google Gemini API
Messaging API	Twilio API for WhatsApp
Scheduling	APScheduler
Caching	JSON Files

3.6 Use Case Diagrams

Figure 3.1 shows the use case diagram for the overall project. Figure 3.2 represents the WhatsApp-based use case diagram showing the various functions, such as notification processes. Finally, Figure 3.3 shows the use case diagram for client-side usage of the application.

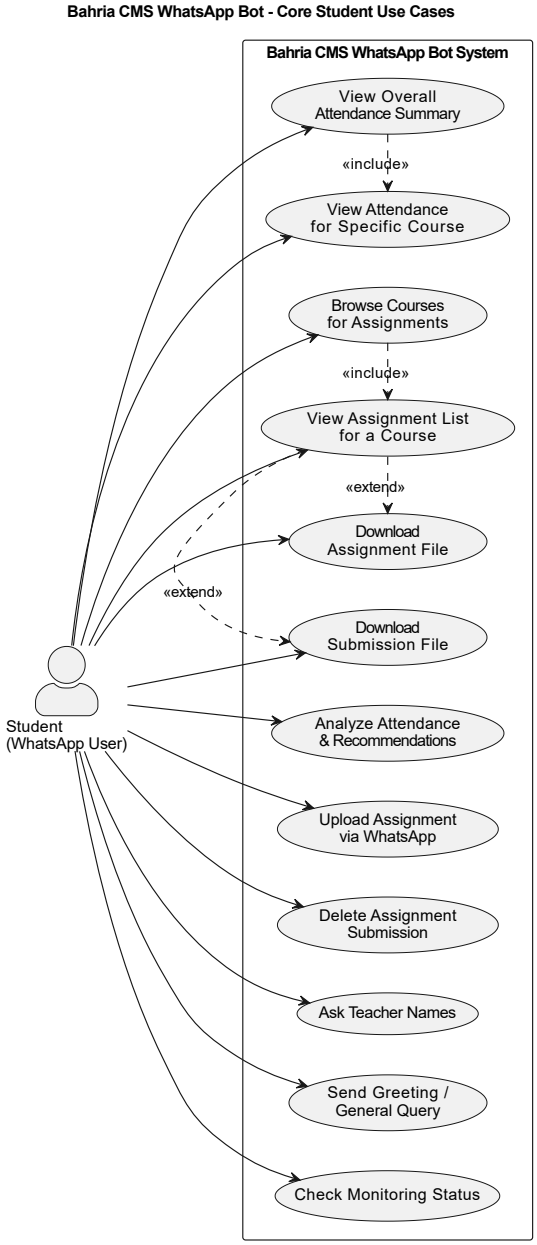

Figure 3.1: Core Student Use Cases: Main student interactions including attendance checking, assignment management, and general queries through WhatsApp.

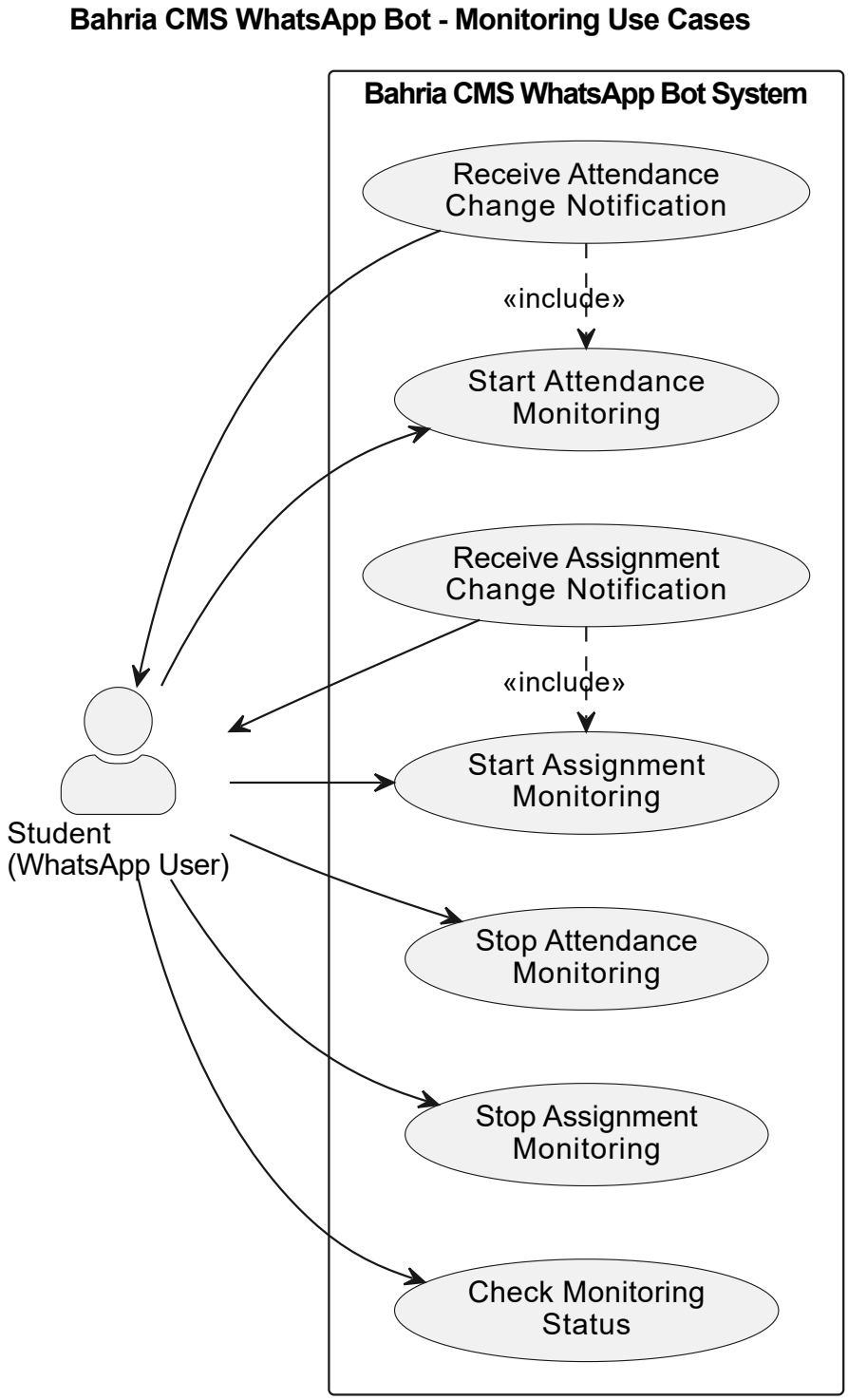

Figure 3.2: Monitoring Use Cases: System monitoring features including starting/stopping attendance and assignment monitoring, and receiving notifications.

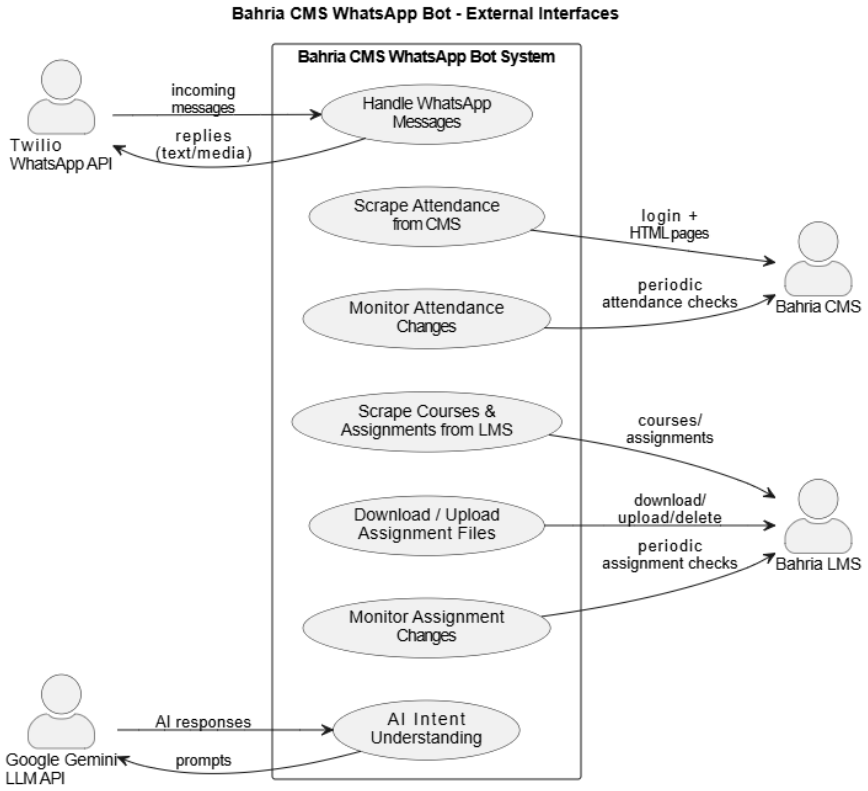

Figure 3.3: External System Interfaces: Integration with external systems including Bahria CMS, Bahria LMS, WhatsApp API, and Google Gemini AI.

3.7 Use Cases

3.7.1 Use Case UC-01: Attendance Checking

Table 3.6: Attendance Checking Use Case

Component	**Description**
Use Case Name	Check Attendance Status
Primary Actor	Student
Brief Description	Student would like to see the existing attendance of all courses in very simple chat command.
Pre-conditions	Bahria CMS authentication has valid student credentials stored in system.
Main Success Scenario	1. Student sends the message of attending or attendance. 2. System is automatically logged into Bahria CMS. 3. System goes to attendance page. 4. System gathers all attendance information. 5. System organizes the data into readable report. 6. Student gets attendance summary in detail.
Extensions	In case of failure to login, system will retries and will display an error message.

3.7.2 Use Case UC-03: Assignment Download

Table 3.7: Assignment Download Use Case

Component	**Description**
Use Case Name	Download Assignment File
Primary Actor	Student
Brief Description	The assignment file is downloaded by the student using the chat option on WhatsApp.
Pre-conditions	Student has active session and assignments can be found in CMS.
Main Success Scenario	1. Student requests to be downloaded an assignment. 2. System displays list of courses available. 3. Student chooses course code or name. 4. System displays assignments of course of choice. 5. Student chooses number of assignment. 6. System gets the file CMS. 7. System forwards file to student through Whatsapp.
Extensions	In case file is not available, system gives an error and returns to assignment list.

3.7.3 Use Case UC-04: Assignment Upload

Table 3.8: Assignment Upload Use Case

Component	**Description**
Use Case Name	Upload Assignment Submission
Primary Actor	Student
Brief Description	Student forwards assignment file to Bahria LMS using WhatsApp.
Pre-conditions	Student has finished assignment file and is ready to upload.
Main Success Scenario	1. Student types in upload assignment. 2. System list displays course information to be submitted. 3. Student chooses course of study. 4. System displays uploadable assignments. 5. Student chooses number of assignment. 6. Student posts file using WhatsApp. 7. System sends file to Bahria LMS. 8. System provides indication of successful submission.
Extensions	In case of failure to upload, system displays certain error and gives a possibility to repeat.

3.7.4 Use Case UC-05: Get Dual Platform Notifications

Table 3.9: Get Dual Platform Notifications Use Case

Component	**Description**
Use Case Name	Receive Dual Platform Notifications
Primary Actor	Student
Brief Description	Academic alerts are automatically sent to student on WhatsApp and Ria mobile app.
Pre-conditions	CMS changes are detected and monitored.
Main Success Scenario	1. System identifies change/new assignment attendance in CMS. 2. System gives immediate notification to WhatsApp. 3. System pushes notification to Ria mobile application. 4. Student is notified on both platforms at the same time. 5. Student has access to details when tapped on notification.
Extensions	In the event of a breakdown of one platform, the other platform will keep on applying notifications.

3.7.5 AI Analysis Use Case

Table 3.10: AI Analysis Use Case

Component	**Description**
Use Case Name	Get Intelligent Academic Analysis
Primary Actor	Student
Brief Description	Academic performance is smartly analyzed and recommendations given to the student.
Pre-conditions	System contains existing attendance records.
Main Success Scenario	1. Student requests to be analyzed in terms of attendance or what courses should be taken care of. 2. AI patterns and percentages of attendance. 3. System singles out courses below 75% mark. 4. AI produces individual suggestions. 5. Student is provided with detailed analysis report and suggestions of improvement.
Extensions	In case of outdated data, system offers to refresh attendance data first.

3.7.6 Use Case UC-06: Download Lecture Materials

Table 3.11: Download Lecture Materials Use Case

Component	**Description**
Use Case Name	Download Lecture Materials
Primary Actor	Student
Brief Description	WhatsApp is a way through which students download their course lectures, slides, notes and other study materials.
Pre-conditions	Student is actively engaged and the course materials can be found in CMS.
Main Success Scenario	1. Student sends download lectures or get course materials. 2. System displays list of courses enrolled in. 3. Student chooses course of study. 4. System demonstrations (PDFs, PPTs, Documents). 5. Student chooses certain material. 6. System downloads and transfers file through WhatsApp. 7. Student gets lecture material immediately.
Extensions	If no materials are available, system will display appropriate message.

3.7.7 Use Case UC-07: System Authentication

Table 3.12: System Authentication Use Case

Component	**Description**
Use Case Name	Preliminary System Authentication
Primary Actor	Student
Brief Description	This option is offered by Student, securely offers the credentials of Bahria CMS to install the system the first time.
Pre-conditions	Student has valid account and enrolment in Bahria University and CMS.
Main Success Scenario	1. Student starts conversation with bot. 2. System makes a request to Bahria CMS. 3. Student gives enrollment ID and password. 4. System checks identity through test login. 5. System is a place where encrypted credentials are stored. 6. System establishes successful authentication. 7. Student is now able to use all the features.
Extensions	In case of credentials being invalid, system retries and displays instructions of help.

Chapter 4

Design

This chapter shows the design of the complete AI-Powered Bahria University CMS WhatsApp Bot with Backup Flutter App (Ria) with system architecture, design methodologies, and specifications of each component.

4.1 System Architecture

Bahria University Academic Assistant is an intelligent system based on AI and multi-platforms that links students to their academic information via the AI-powered chat and mobile application. This system has three primary parts and a main brain of AI.

4.1.1 System Overview

The platform is a smart academic assistance platform where:

- Students use WhatsApp to ask natural language questions regarding attendance, assignments, and academic suggestions.
- AI Agent requests are processed with the help of Gemini 2.0 Flash and corresponding tools are run.
- CMS Scraper is a tool that will be used to automate the process of retrieving data within the CMS system of Bahria University.
- Backup Flutter App (Ria) is a different mobile access to all features.
- Monitoring System is used to give real-time attendance alterations and new assignments alerts.

The system applies AI-enhanced natural language processing to comprehend queries of students, and offers real-time academic data, file management, and proactive monitoring services.

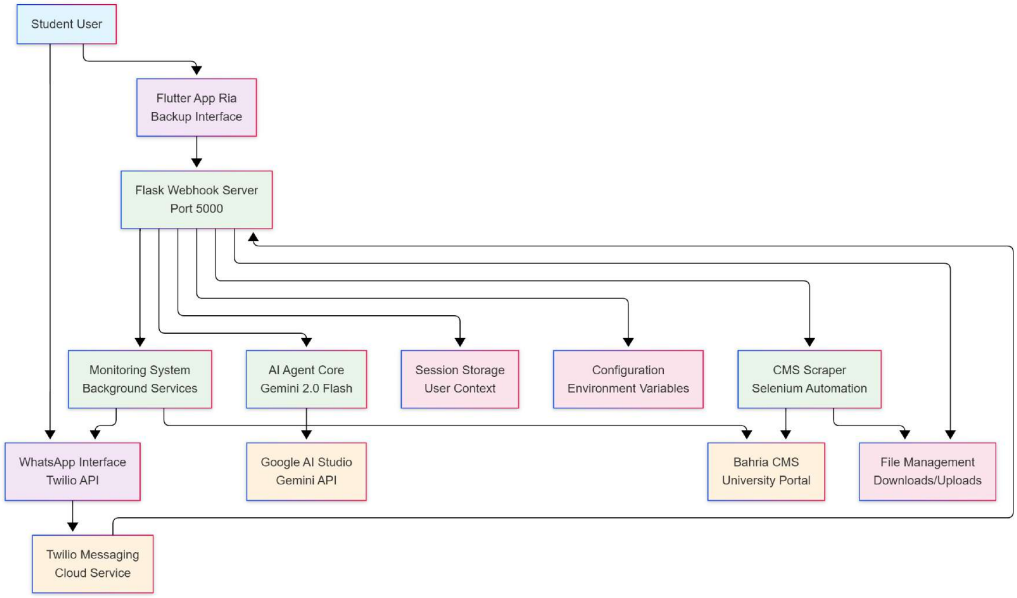

Figure 4.1: High-Level System Architecture

4.2 Design Methodology

4.2.1 Development Approach

The system is based on the Object-Oriented Design (OOD) and multi-layered architecture pattern of the backend and Provider Pattern of Flutter applications.

4.2.1.1 Backend Design Principles

- **Separation of Concerns:** AI Agent, Webhook controllers and Scraping Engine are separated.
- **Tool-Calling Agent Pattern:** AI agent makes use of special tools in accomplishing various tasks.
- **Middleware Pattern:** Middleware pattern is used to control session, deal with errors, and authenticate.
- **Real-time Monitoring:** Background services having fixed intervals.
- **Session Validation:** The session storage is stored on files with time management.

4.2.1.2 Frontend Design Principles

- **State Management:** Redundant provider pattern of state management in flutter app.
- **Component-Based Architecture:** Widgets and screens reuse in Ria app.
- **Service Layer:** API services that are decoupled of UI layer.
- **Cross-Platform Design:** Single codebase for iOS and Android

4.2.2 Technology Stack

Table 4.1: Technology Stack

Component	Technology
Backend Framework	Python with Flask
AI/ML Engine	Google Gemini 2.0 Flash
Web Automation	Selenium WebDriver
Messaging Platform	Twilio WhatsApp API
Mobile Frontend	Flutter (Dart) - Ria App
Session Management	File-based JSON storage
Real-time Monitoring	Python threading that is configurable at an interval.
HTTP Client	Requests library (Python)
State Management	Provider (Flutter)
Browser Management	WebDriver Chrome that is profile isolated.

4.3 High Level Design

4.3.1 Conceptual View - Component Diagram

Figure 4.2 represents the high-level conceptual view of the system. It illustrates the interaction between core components such as the AI agent, messaging service, CMS integration, and monitoring modules.

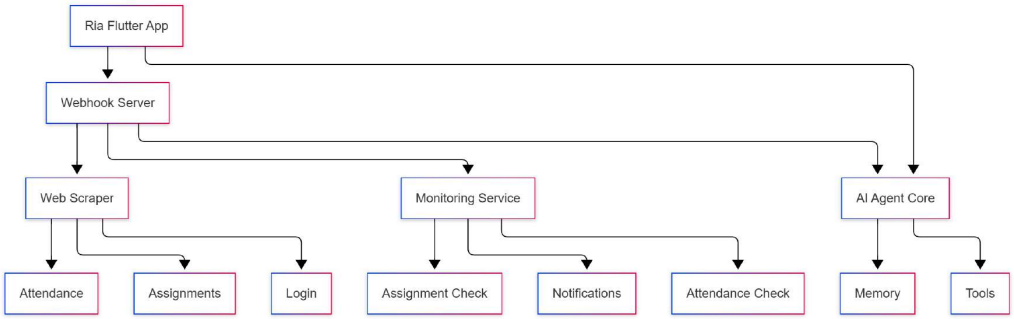

Figure 4.2: Component Diagram

4.3.2 Process View - WhatsApp Message Processing Flow

Figure 4.3 shows the complete message handling pipeline. It explains how incoming WhatsApp messages are received, validated, processed by the AI agent, and responded to via Twilio.

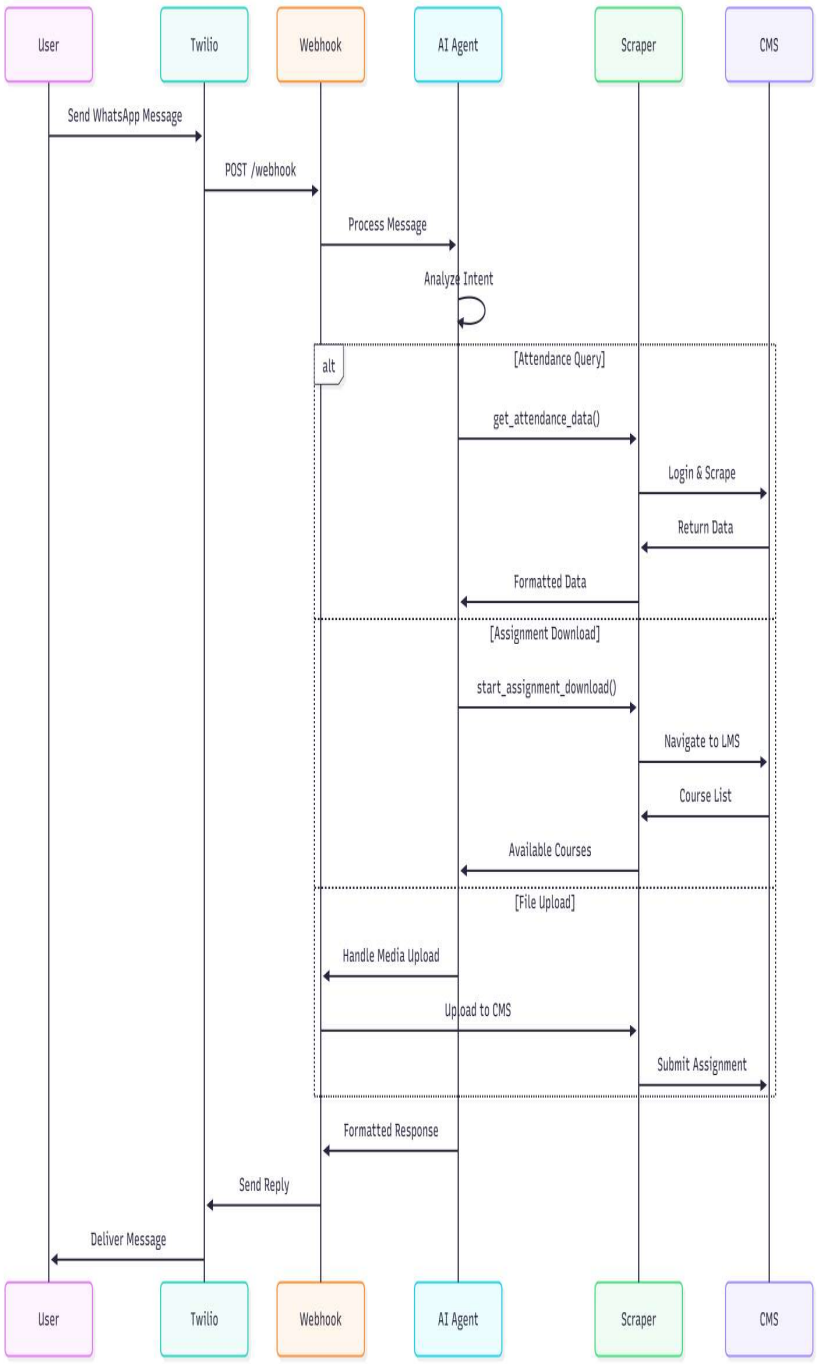

Figure 4.3: WhatsApp Message Processing Flow

4.3.3 Process View - Background Monitoring Flow

Figure 4.4 describes the background monitoring mechanism. It highlights how scheduled jobs periodically check CMS updates and trigger alerts when changes are detected.

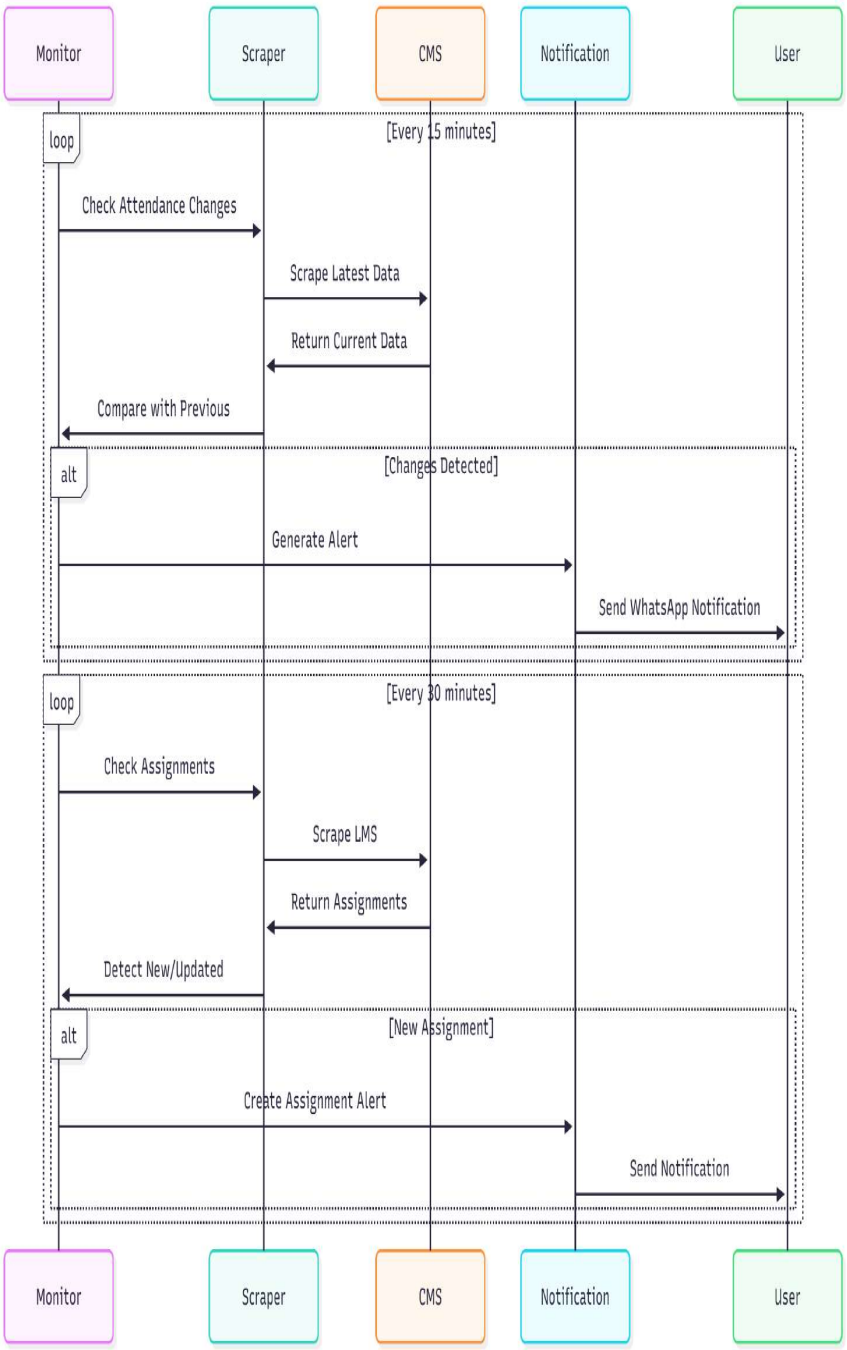

Figure 4.4: Background Monitoring Flow

4.3.4 Physical View - Deployment Diagram

Figure 4.5 presents the physical deployment architecture. It shows how backend services, APIs, and external systems are deployed and communicate over the network.

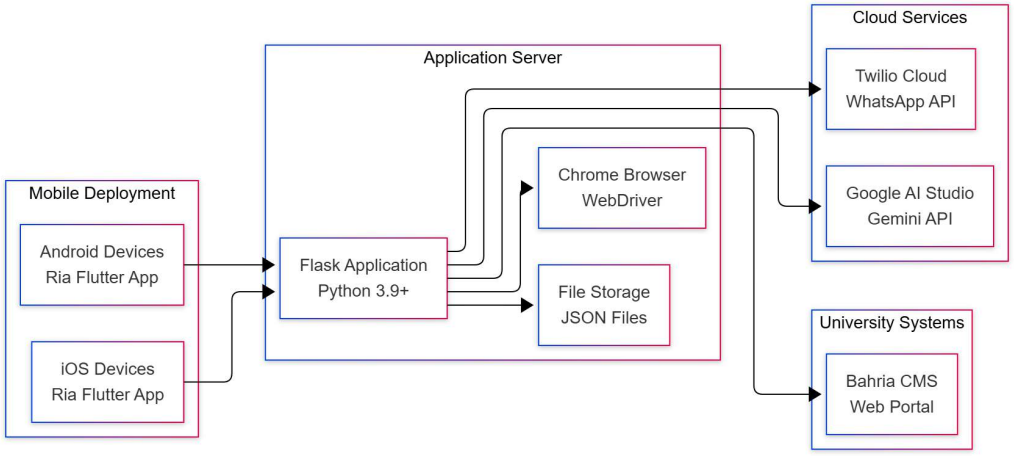

Figure 4.5: Deployment Diagram

4.3.5 Module View - Directory Structure

Backend Module Structure: Figure 4.6 illustrates the internal organization of backend modules. It separates responsibilities across AI logic, web scraping, session management, and monitoring components.

Flutter App Module Structure (Ria): Figures 4.7 to 4.9 show the modular design of the Flutter application. They demonstrate separation of UI, services, state management, and configuration files.

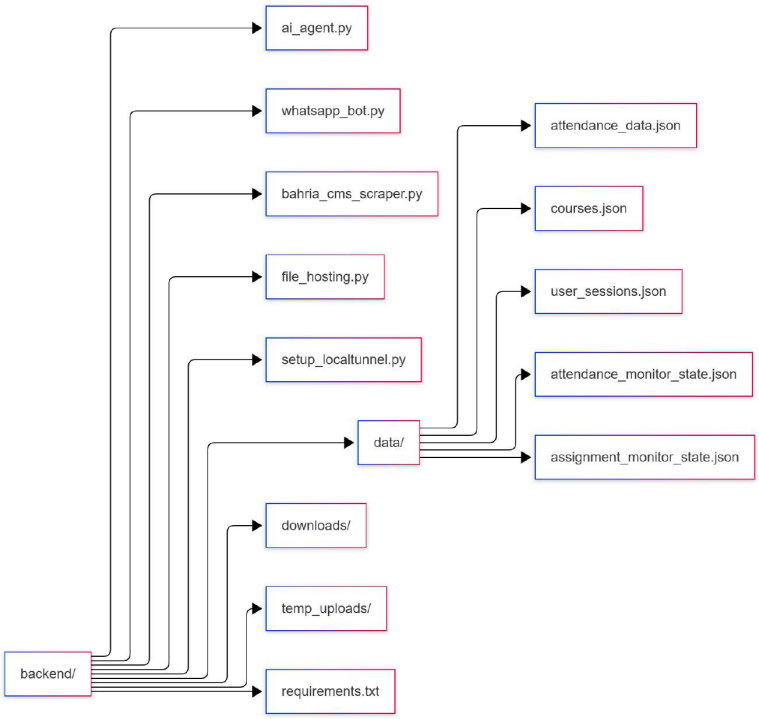

Figure 4.6: Backend Module Structure

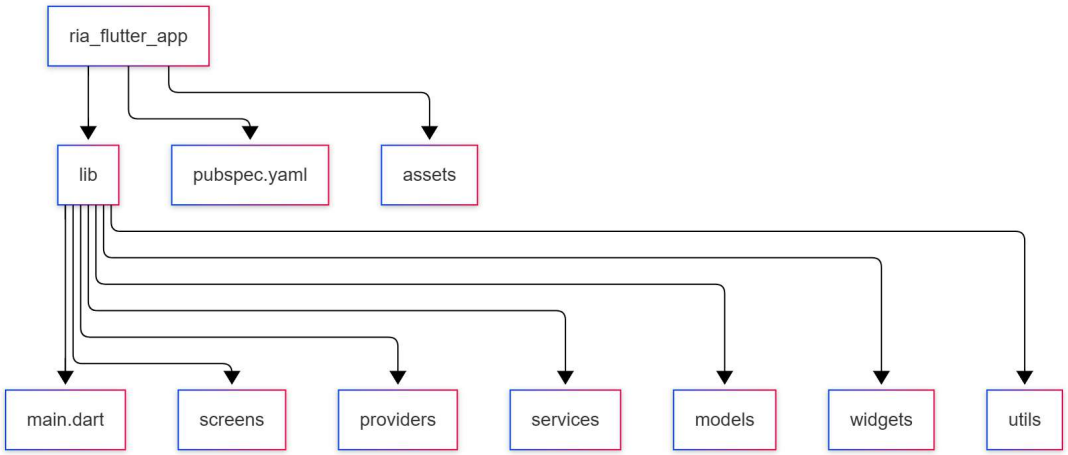

Figure 4.7: Flutter App Module Structure (Ria) - Part 1

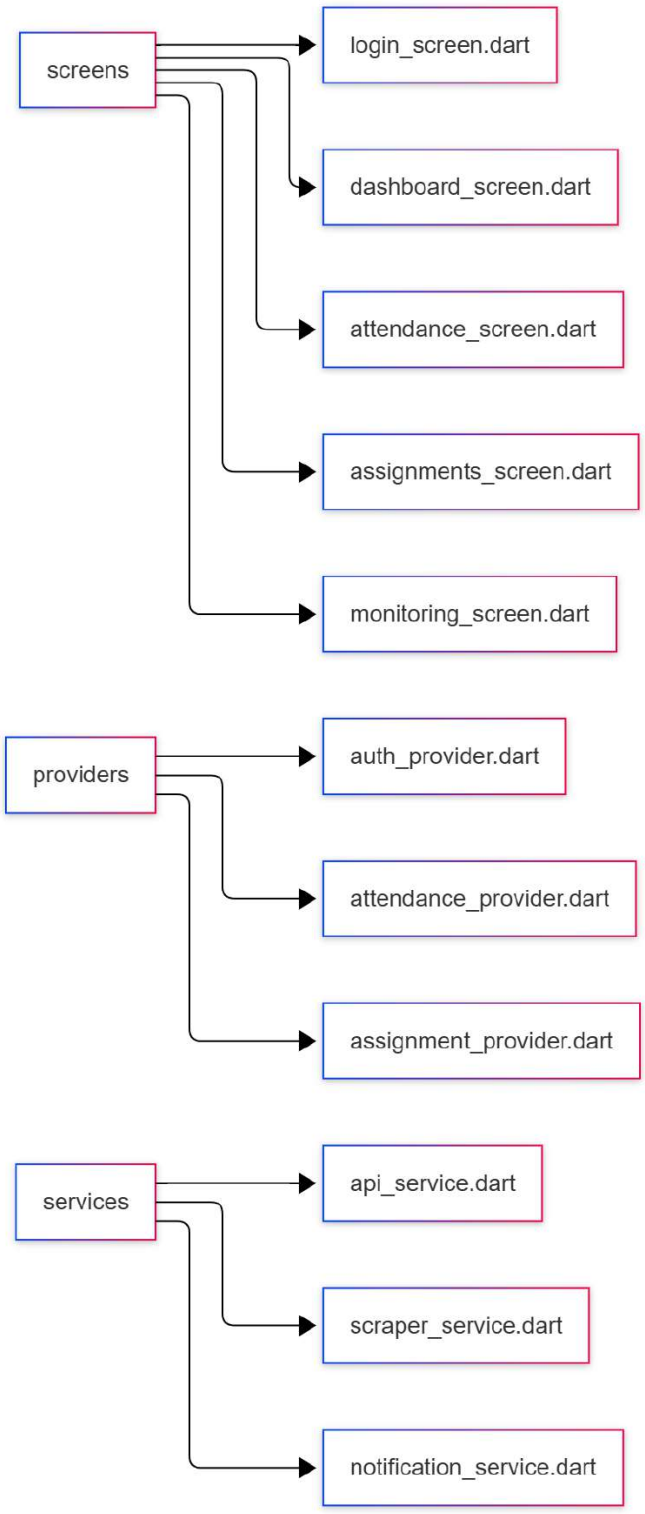

Figure 4.8: Flutter App Module Structure (Ria) - Part 2

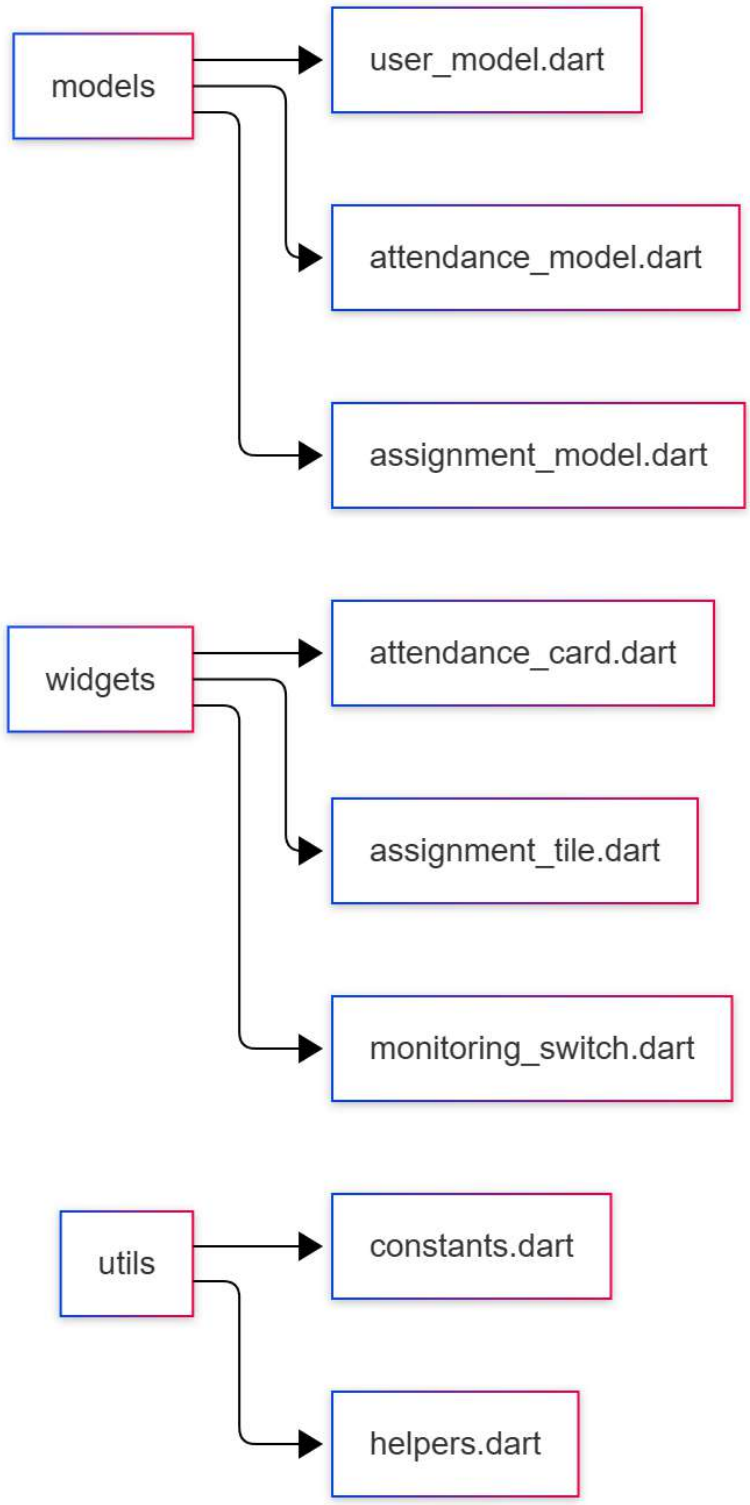

Figure 4.9: Flutter App Module Structure (Ria) - Part 3

4.3.6 Security View

Figure 4.10 presents the authentication and authorization design. It defines how user sessions, credentials, and access control are handled securely within the system.

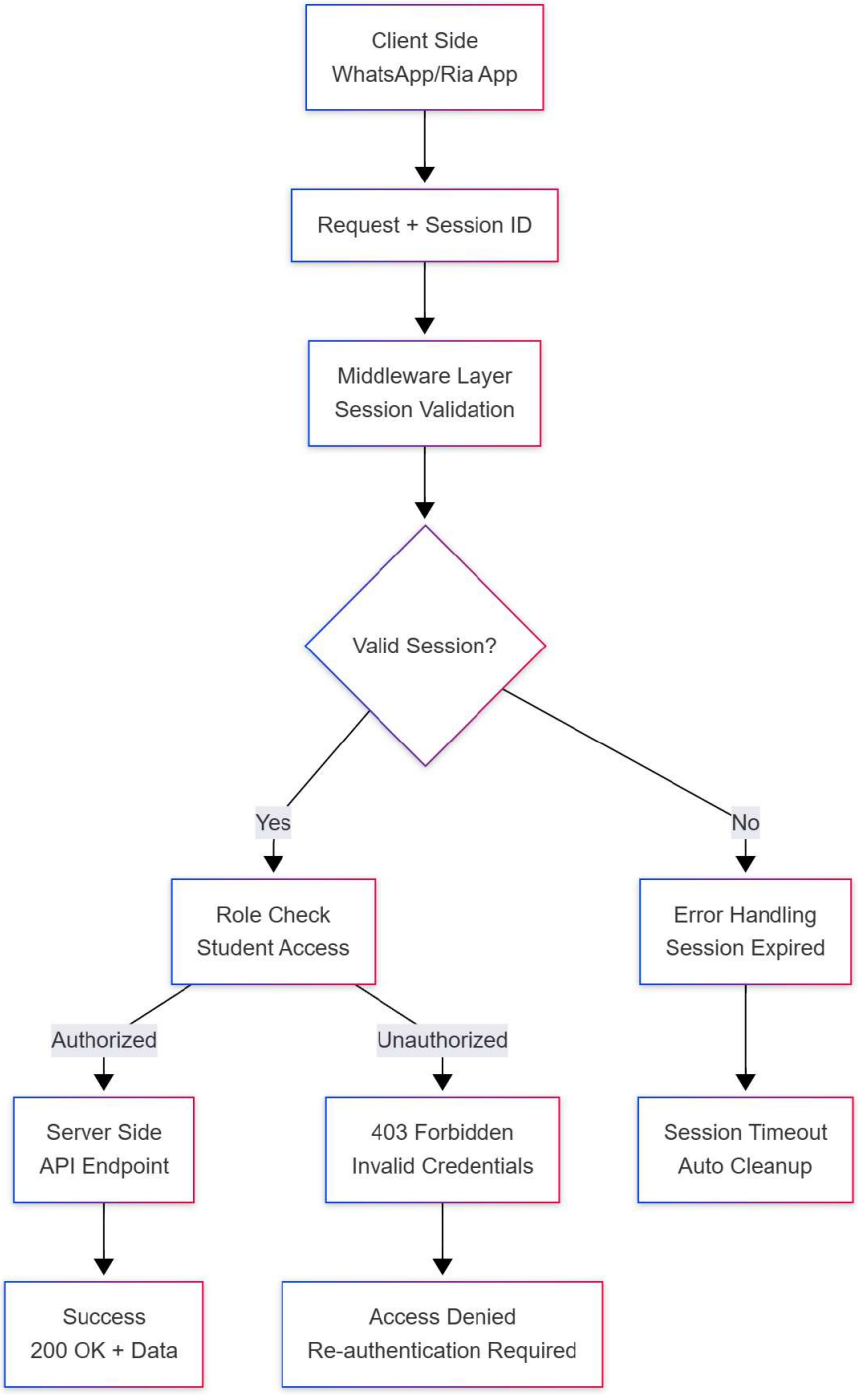

Figure 4.10: Authentication Class Diagram

Security Components:

- **Session Validation:** File-based session storage with a 60-second timeout.
- **Role Check:** Restricts access to authorized student users only.
- **Credential Protection:** Secure storage using environment variables.
- **Auto Cleanup:** Browser sessions automatically closed on timeout.
- **Error Handling:** Graceful handling of session expiration scenarios.

4.4 Low Level Design

4.4.1 AI Agent Module

Figure 4.11 shows the core class structure of the AI agent. It outlines the interaction between intent processing, tool execution, and response generation logic.

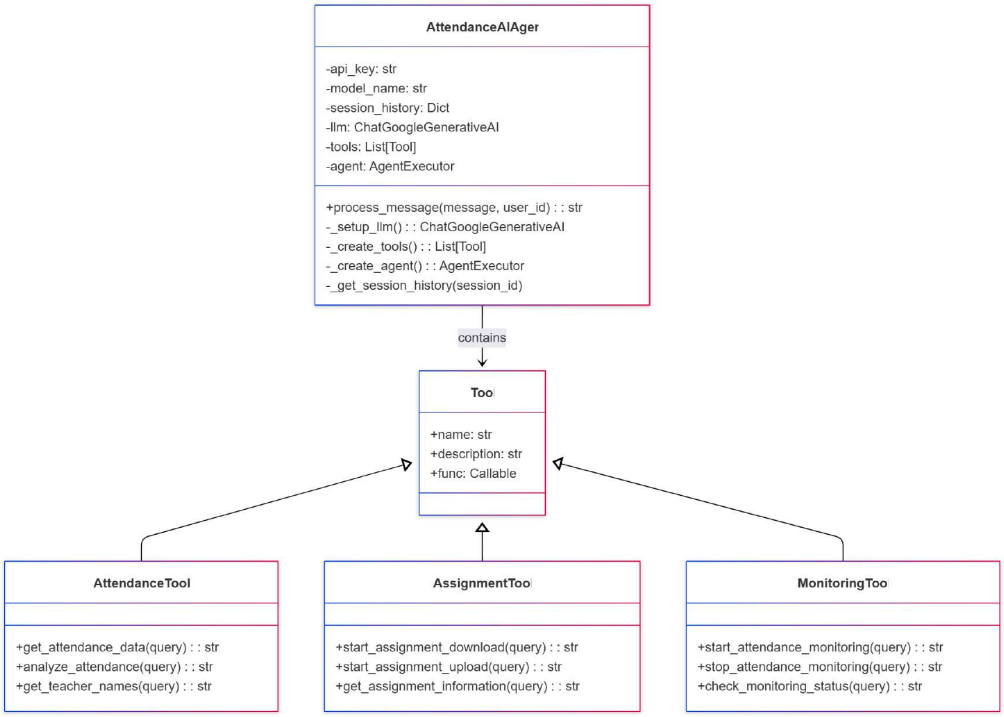

Figure 4.11: AI Agent Core Class Diagram

Figure 4.12 details the internal workflow of message handling. It explains intent detection, validation, tool invocation, and response formatting.

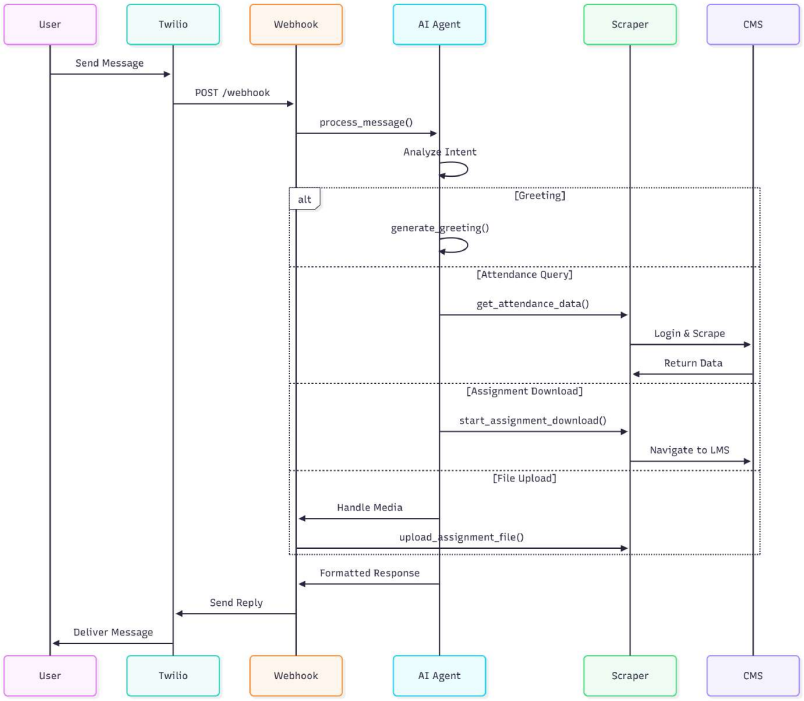

Figure 4.12: WhatsApp Message Processing Workflow

Figure 4.13 represents the automated assignment retrieval process. It shows CMS login, navigation, file extraction, and delivery to the user.

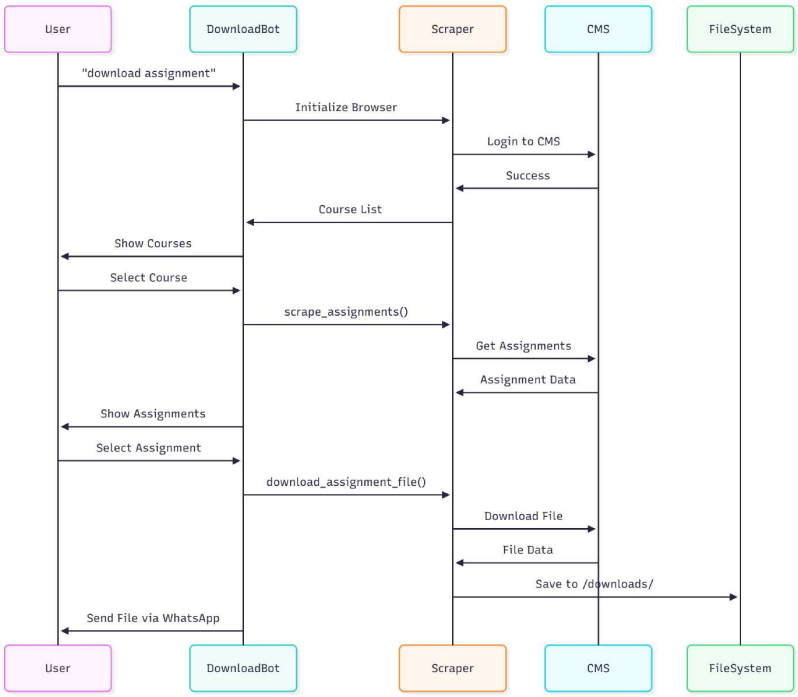

Figure 4.13: Assignment Download Workflow

4.4.2 Web Scraping Module

Figure 4.14 illustrates the web scraping class design. It highlights browser automation, page parsing, and data extraction responsibilities.

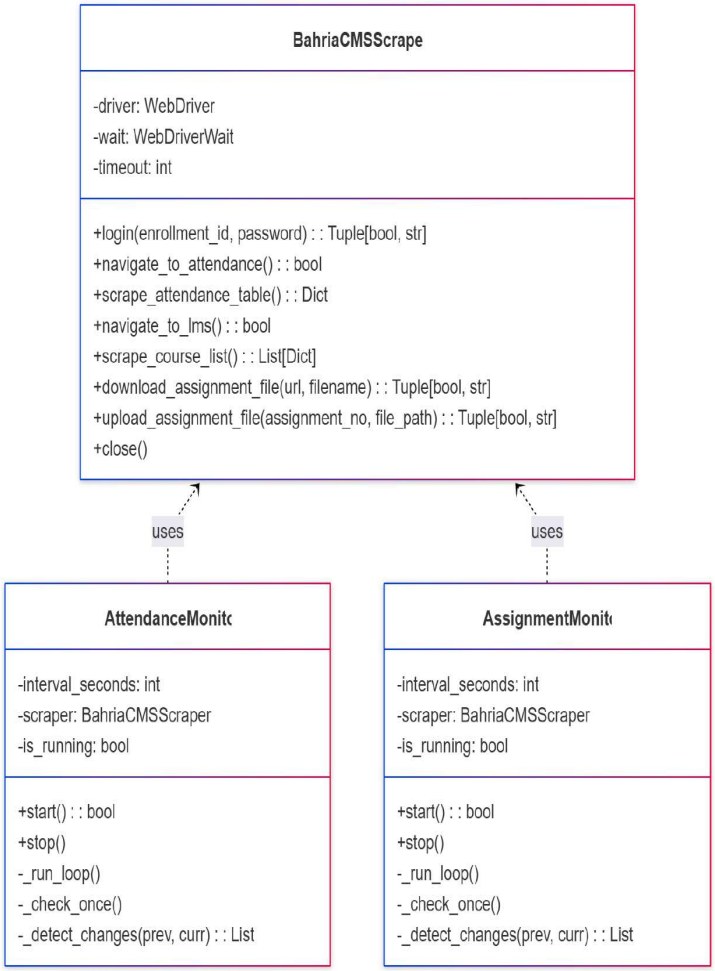

Figure 4.14: Class Diagram - Web Scraper

4.4.3 Monitoring Module

Figure 4.15 shows the monitoring architecture. It explains how background schedulers track updates and notify users asynchronously.

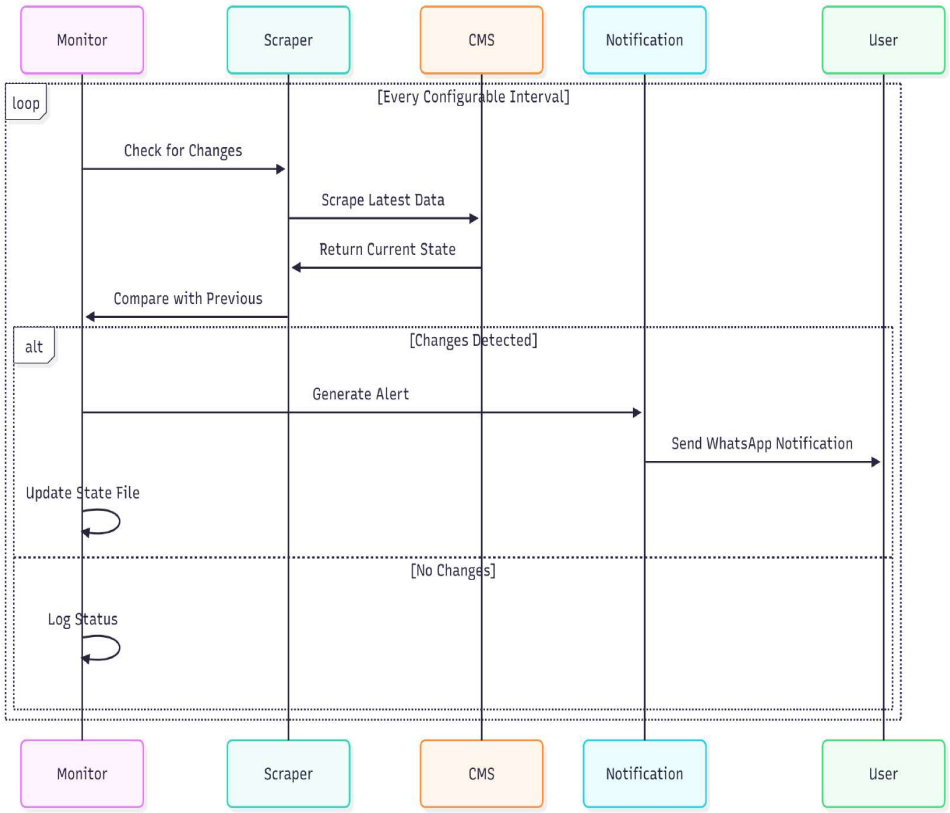

Figure 4.15: Background Monitoring Architecture

4.4.4 Session Management Module

Figure 4.16 depicts session lifecycle management. It outlines session creation, validation, expiration, and cleanup processes.

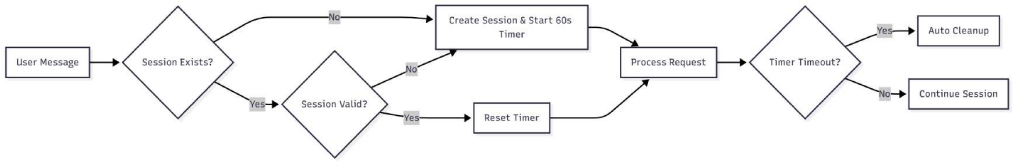

Figure 4.16: Session Management Activity Diagram

Figure 4.17 describes the file storage organization. It shows how session files, logs, and cached data are structured for efficient access.

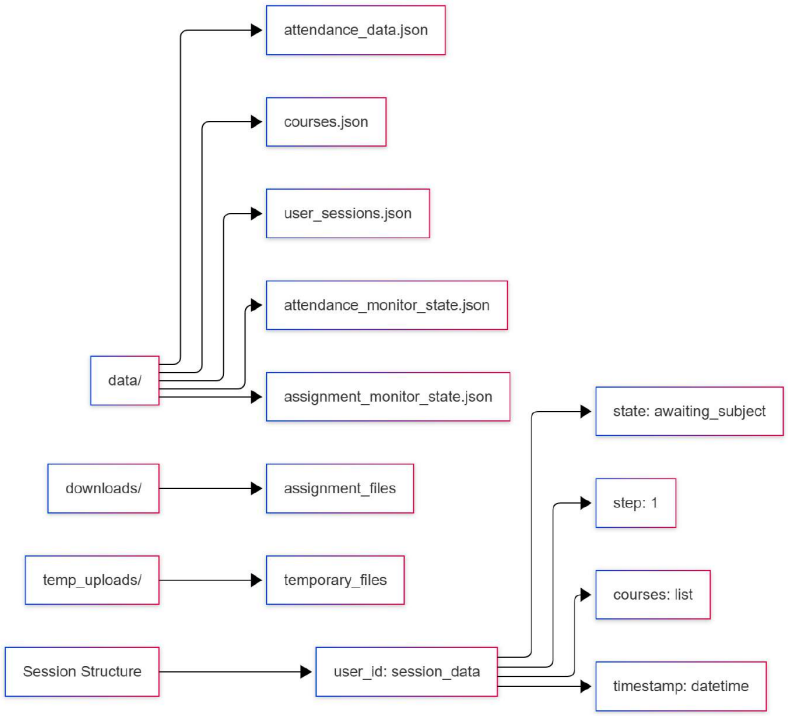

Figure 4.17: File Storage Structure

4.5 Database Design

4.5.1 File-Based Data Storage Architecture

The design uses JSON files and directory structures to store the data since the system uses file based storage rather than a conventional database.

4.5.1.1 Data Storage Structure

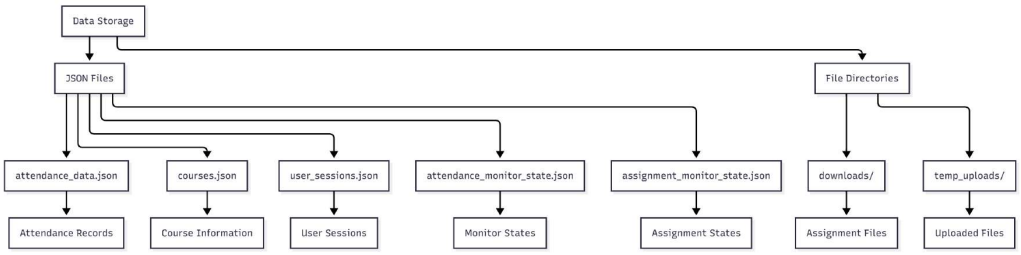

Figure 4.18: File-Based Data Storage Architecture

4.5.2 Data Dictionary

4.5.2.1 Attendance_data.json Schema

Table 4.2: Attendance_data.json Schema

extbfField	**Type**	**Source**	**Description**
semester	String	CMS Scraper	Name of the current semester
scrape_timestamp	String	System	Last update time
total_courses	Integer	CMS Scraper	Number of courses
courses	Array[Object]	CMS Scraper	Course attendance information.

4.5.2.2 user_sessions.json Schema

Table 4.3: user_sessions.json Schema

extbfField	**Type**	**Source**	**Validation**	**Description**
user_id	String	WhatsApp	Necessary	Unique user identifier.
state	String	System	["none", awaiting-subject, awaitingas-signment, awaiting-file]	State of current session.
step	Integer	System	1-4	Workflow step
courses	Array	CMS Scraper	Optional	Available courses
timestamp	String	System	ISO format	Last time of operation.

4.5.2.3 courses.json Schema

Table 4.4: courses.json Schema

extbfField	**Type**	**Source**	**Description**
scrape_timestamp	String	System	Data collection time.
total_courses	Integer	CMS Scraper	Number of courses.
courses	Array[Object]	CMS Scraper	Course list

4.5.2.4 Monitor State Files Schema

Table 4.5: Monitor State Files Schema

extbfField	**Type**	**Source**	**Description**
semester	String	CMS Scraper	Semester identifier.
scrape_timestamp	String	System	Latest check time.
courses	Array[Object]	CMS Scraper	Past state data.

4.5.3 File Operations

4.5.3.1 CRUD Operations

Create/Update Operations:

- savecurrentdata() - Saves data of attendance.
- savestate() - Saves monitor state.
- savetojson() - JSON file General saving.

Read Operations:

- load_currentdata() - loads the data on the attendance.
- load_state - Loads monitor state.

- get_session history- Accesses session information.

Delete Operations:

- cleanup session - cleans up user sessions.
- Auto file cleanup on timeout

4.5.3.2 Data Flow Management

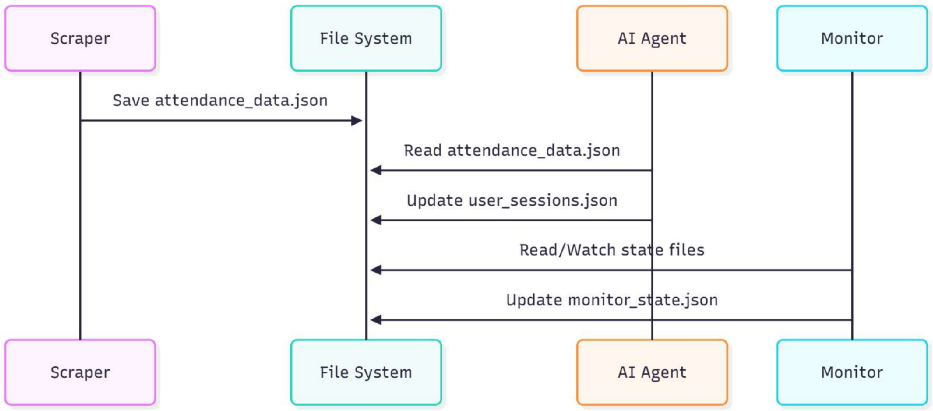

Figure 4.19: Data Flow

4.6 GUI Design

4.6.1 Screen Flow Diagram

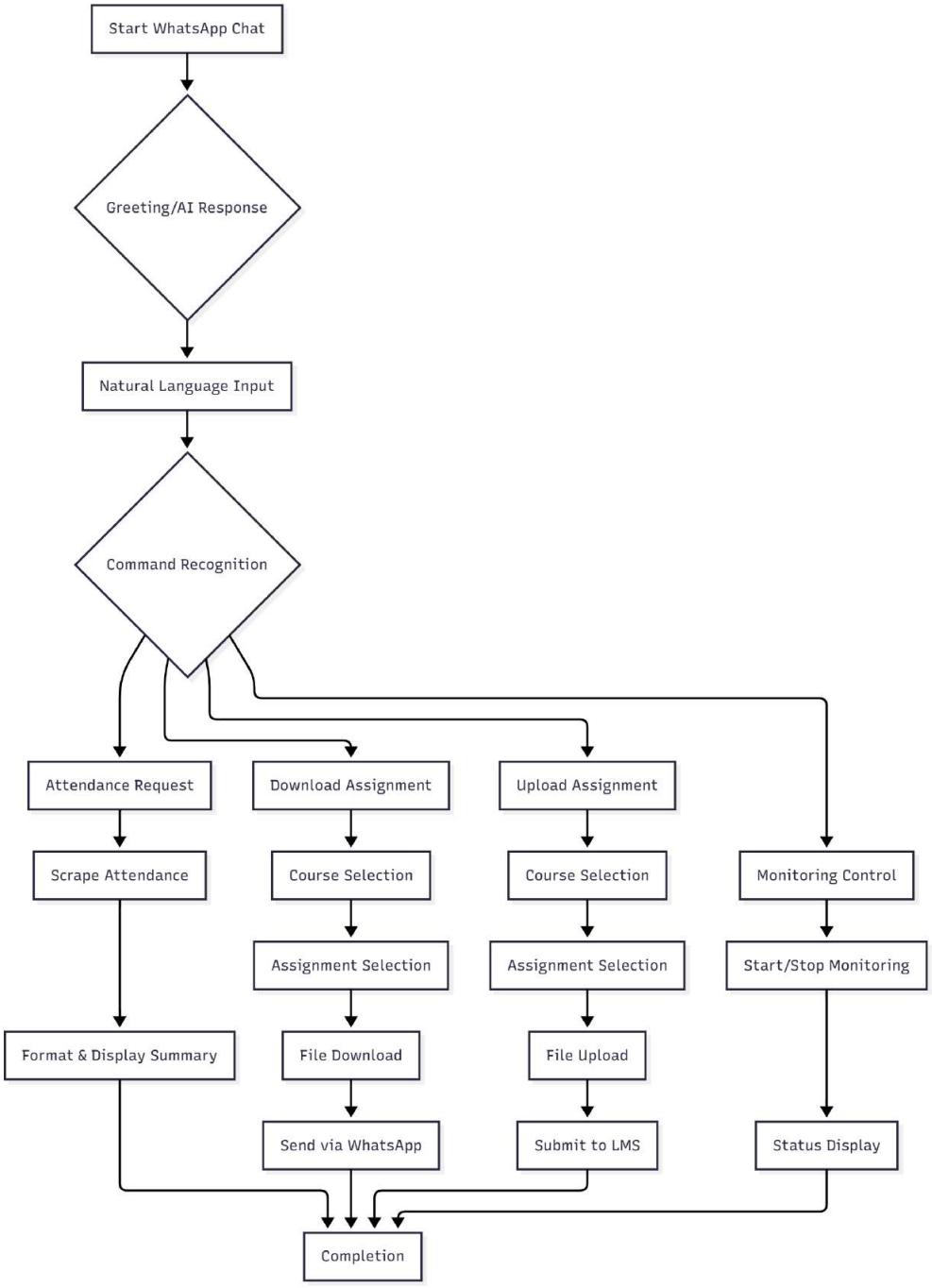

Figure 4.20: WhatsApp Bot Navigation Flow

4.6.1.1 Major WhatsApp Interface Components:

Main Chat Screen:

- Input of natural language messages.
- Responses that are generated by AI using emojis.
- Assigned upload file button.
- Automated system messages that are structured in form.

Command Response Screens:

- **Summary of attendance:** Course list, percentages and status indicators.
- **Selection of course:** Listing of course numbers and names of teachers.
- **Assignment Lists:** Assignment information download/upload status.
- **Analysis Reports:** Styled information and recommendations.
- **Status monitoring:** Active/inactive and interval monitoring.

File Handling Screens:

- Messages of uploading files.
- Progress indicators Downloads.
- Validations on format (PDF, DOC, DOCX, XLS, XLSX and ZIP).
- The file sharing is connected through temporary hosting.

4.6.2 Ria Mobile Application Screens (Flutter Backup)

4.6.2.1 Ria App Navigation Flow

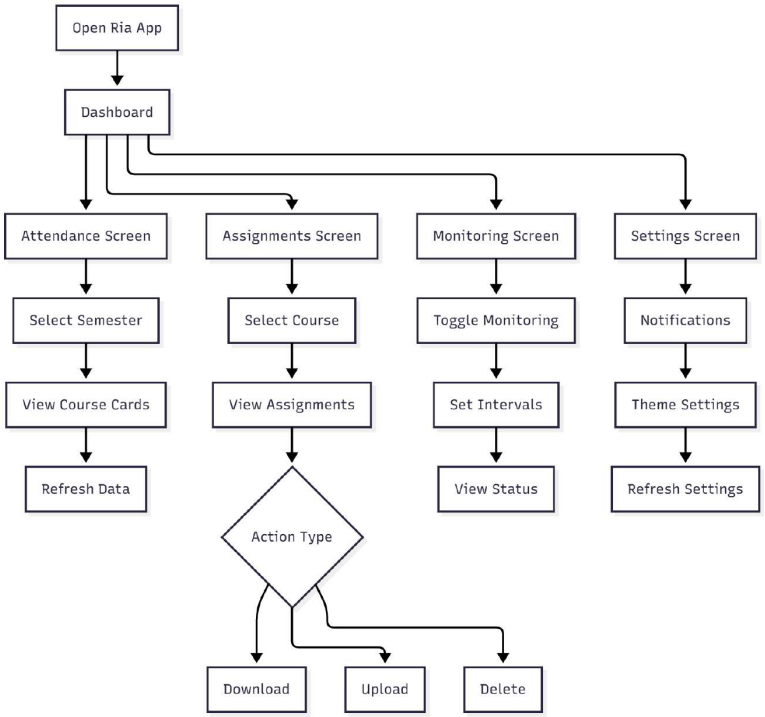

Figure 4.21: Ria App Navigation Flow

4.6.2.2 Key Ria Interface Elements

Login Screen:

- Fields of enrollment ID, password.
- Institute selection drop down.
- Please remember credentials option.
- Login progress indicator

Dashboard:

- Short circuit cards: Monitoring, Monitoring, Monitoring, Assignments, Attendance.
- Recent activity timeline
- Monitoring alert notification bells.
- Profile menu for settings

Attendance Screen:

- Semester selector dropdown
- Attendance percentages on course cards that are colored.
- Present/Absent/Total hours display
- Teacher information
- Update button on the recent data.
- Recommendation tab of analysis.

Assignment Management Screen:

- Subject list subject course selection.
- Assignment cards with:
  - Title and number of assignment.
  - Deadline countdown
  - Download/Upload/ Delete buttons.
  - Obtained marks (where available)
- File picker for uploads
- Download progress bar

Monitoring Control Screen:

- Toggle switch monitoring attendance.
- Toggle switch Assignment monitoring.
- Sliders of interval settings (in seconds/ minutes).
- Notification preferences
- Active monitor condition indicator.
- Browser connection status

Settings Screen:

- CMS credentials management
- Notification preferences
- Intervals of monitoring configuration.
- Data refresh frequency
- Theme selection (Light/Dark)

4.6.3 Common Design Elements

Navigation:

- Dashboard, Attendance, Assignments, Monitoring, Settings (bottom navigation bar, Ria App).
- Quick action floating button.
- Session persistence Back button Back button.

Visual Design:

- **Color Scheme:** Status colors, Bahria University colors (Blue/White).
- **Typography:** Good readable fonts and hierarchy.
- **Icons:** Material design icons to ensure uniform UX.
- **Status Indicators:**
  - Good attendance ($\geq$85%)
  - Warning (75-85%)
  - Critical ($<$75%)
  - Download available
  - Upload required
  - Deadline approaching

4.6.4 Responsive Design Factors.

Flutter App Responsiveness:

- Responsive designs based on Mediaquery and Layoutbuilder.
- Flexible widgets , Expanded, Flexible, Dynamically sized.
- Android and iOS-specific UI Material Design and Cupertino.
- Size of fonts in accordance with system preferences.
- Safe regions working on notches and system U.I.

WhatsApp Compatibility:

- Optimization of message length (Less than 1500 characters)
- The use of emojis to communicate in an image form.
- Consideration is precluded by file size.
- Suggestions of quick reply in application.

4.6.5 Error Handling UI

Connection Errors:

- Offline mode indication
- Progress bars in place of a re-try button
- Back-up messages falling back to alternative file hosting.

Authentication Issues:

- CMS clear error messages when logging in.
- Credential reset options
- Notifications of session time out.

File Handling:

- ForValidation of format error with permissible types.
- Size limit warnings
- Tracking of the upload/download progress.
- Confirmation messages of success/ failure.

4.7 Summary

In this chapter, the overall design of the AI-Powered Bahria University CMS WhatsApp Bot platform was described and includes:

- **High-level architecture:** Multi-layer client-server architecture and integration of real-time messaging
- **Design methodology:** According to the principles of the object-oriented programming and modular service patterns.
- **Specific component:** Architecture of AI agent processing, CMS scraping, file management and monitoring services.
- **Database architecture:** Based on the storage of the data using the JavaScript Object Structure and full session management.
- **Graphical interface design:** Design to WhatsApp and Flutter mobile application with navigation flows

- **External interface:** Requirements of WhatsApp API, Bahria CMS, Google AI services, and file hosting services.

The design guarantees reliability by providing dual platform accessibility, guaranteeing security by isolating the session, protecting credentials and providing smart user experience through natural language processing and active monitoring of all service interfaces.

Chapter 5

System Implementation

5.1 System Architecture

5.1.1 System Internal Components

Our AI-Powered Bahria University Assistant applies an advanced multi-component system that allows integrating the use of artificial intelligence and academic management in a smooth way. The core system consists of:

1. **Flask Web Server:** The central nervous system, where all incoming HTTP requests are passed by Twilio WhatsApp webhooks and mobile application APIs. The server takes care of routing of requests, session management, as well as, coordination among various modules of the system.

2. **AI Agent Core:** The intelligent brain that will work with LangChain framework with Google Gemini 2.0 Flash integration [8]. This element processes natural language inquiries, sustains dialogue history, and bestows the selection of the right instruments in accordance with student requests.

3. **Web Scraping Engine:** This component is created using Selenium WebDriver and BeautifulSoup and does all the interaction with Bahria CMS. It handles browser automation, authentication on log, data mining and file processing with excellent error management.

4. **Dual Notification System:** Implements the use of parallel messaging via Twilio WhatsApp API and Firebase Cloud Messaging as the Ria mobile app. This faces unnecessary delivery channels of important academic alerts.

5. **Background Monitoring Services:** This is a constant daemon process which monitors and records any changes in attendance, assignment and deadline changes using defined CMS polling and state comparison algorithms.

5.1.2 Functionality of Components

5.1.2.1 AI Agent Core Functionality

- Natural language queries via advanced language understanding of Google Gemini.
- Enforces conversation memory by use of chat history.
- Selects the relevant tools automatically in an available set of functions.
- Gives a smart analysis of academic performance and recommendations.
- Manages multistage processes such as downloading assignments and uploading assignments

5.1.2.2 Web Scraping Engine Functionality

- Authenticates into Bahria CMS with student account.
- Browses through complicated university portal interfaces.
- Scrapes data in HTML tables and forms.
- Automation of file download/upload using browser.
- Manages session timeouts and re-authentication

5.1.2.3 Notification System Functionality

- Formats and sends messages in various channels.
- Adopts a logic of retries in case of unsuccessful deliveries.
- Keeps delivery status records.
- Offers backup solutions between platforms.

5.1.3 Communication Between Components

The system uses a message-based architecture and has communication protocols:

- **HTTP Webhook Communication:** Incoming WhatsApp messages will be sent to our Flask server by POST requests [9] and the mobile application would use RESTful API endpoints.
- **Inter-Process Communication:** The services of background monitoring are implemented to communicate with the main application through threading [10] and message queues, making sure that the operations are not blocked.

- **Session Synchronization:** The session data are recorded in JSON format [11] in a file with locking facilities to avoid conflicts between all the components sharing the data.
- **External API Integration:** Secure HTTPS connection with Google Gemini API, Twilio services and mobile push notification tools.

5.2 Tools and Technology Used

5.2.1 Development Environment/Languages Used

5.2.1.1 Backend Technologies

Table 5.1: Backend Development Technologies

Category	Technology/Tool	Version/Package	Purpose/Usage
Programming Language	Python	3.8+	Standard programming language of AI agent and web services.
Web Framework	Flask	2.3.3	Web framework for API endpoint and webhook
AI Framework	LangChain	0.1.0	Artificial intelligence agent system [12] of tool selection and memory management.
AI Language Model	Google Gemini	2.0-flash-exp	Natural language processing using large language models.
Web Automation	Selenium	4.15.0	Bahria CMS interaction browser automation [13] with headless Chrome [14]
HTML Parsing	BeautifulSoup	4.12.2	CMS page parsing and data mining.
WhatsApp API	Twilio	8.10.0	WhatsApp Business API messaging and file sharing.
HTTP Client	Requests	2.31.0	Web request and API calls library [15] based on HTTP.
Environment Management	python-dotenv	1.0.0	Configuration of environment variables [16].

5.2.1.2 Frontend Technologies

Table 5.2: Frontend and Mobile Technologies

Category	Technology/Tool	Version/Package	Purpose/Usage
Mobile Framework	Flutter	3.13.0	Multi-platform mobile applications development.
Programming Language	Dart	3.1.0	Mobile application programming language.
Push Notifications	Firebase Cloud Messaging	12.0.0	Push messages Ria mobile app.
UI Framework	Flutter Material	3.13.0	Components of mobile app user interface.
State Management	Provider	6.1.1	Management of Flutter app by the state.
HTTP Client (Mobile)	Http	5.3.2	Mobile app HTTP to backend requests.

5.2.1.3 Infrastructure & Deployment

Table 5.3: Infrastructure and Deployment Technologies

Category	Technology/Tool	Version/Package	Purpose/Usage
Web Server	Gunicorn	21.2.0	Production-ready python WSGI HTTP server.
Reverse Proxy	Nginx	1.18.0	Load balancing and web server.
Cloud Platform	Various Cloud Services	-	Application hosting and deployment.
File Storage	Local File System	-	Storage of data and sessions in a JSON file.
Browser Driver	ChromeDriver	119.0	Chrome automation using selenium web driver.
Process Manager	Systemd	-	Background process service management.

5.2.2 Development Tools

- **Python 3.8+:** The dependency management virtual environments.
- **Visual Studio Code:** Main IDE with Python extensions
- **Git:** Version control system
- **Chrome WebDriver:** To do testing and development of Selenium.
- **Postman:** API testing and Development.
- **Android Studio:** Android Flutter mobile app development.

5.3 Processing Logic/Algorithms

5.3.1 Natural Language Processing

- **Intent Recognition:** The user/student queries are grouped into categories (attendance, assignments, analysis).
- **Context Awareness:** The history of conversation to be followed up.
- **Tool Selection Algorithm:** User intent to system functions.
- **Response Generation:** Data presented in a natural language is generated into formats.

5.3.2 Web Scraping Algorithms

- **Dynamic Element Detection:** Detection of CMS elements when interfaces are changed.
- **Data Extraction Patterns:** CSS Selectors and XPath are used to ensure good scraping.
- **State Management:** Manage tracks cookie and state of login.
- **Error Recovery:** Installs the retry logic of the transient failures.

5.3.3 Monitoring Algorithms

- **Change Detection:** Compares new and old states with the help of hash-based comparison.
- **Intelligent Polling:** Course of academic calendar monitoring.
- **Notification Prioritization:** Prioritizes the alerts in order of urgency and importance.
- **Duplicate Prevention:** Removes duplicate notifications.

5.3.4 File Processing Algorithms

- **Format Validation:** Checks file type and file size.
- **Secure Upload:** Applies anti-virus and security checks.
- **Temporary File Management:** Automatic Cleanup Of old files
- **Download Optimization:** Big files compression and caching.

5.4 Application Access Security

5.4.1 Security Zones

Our system has basic, but fundamental security mechanisms to safeguard student data and guarantee a dependable service:

5.4.1.1 Network Security

- **HTTPS Encryption:** All communications are encrypted by data using the SSL/TLS.
- **Port Restrictions:** The server is open only to essential HTTP/HTTPS ports.
- **Request Limiting:** Simple rate limiting to avoid overloading of the system.
- **Input Validation:** Checks of all user inputs are also sanitized in order to block injection attacks.

5.4.2 Authentication

- **WhatsApp Phone Number:** Uses student WhatsApp number as a way of identifying student.
- **Bahria CMS Credentials:** Data access temporary login (not saved permanently).
- **Session Timeouts:** Timeout after 1 hour of idle user.
- **Unique User IDs:** The user sessions of each student are separated and locked.

5.4.3 Authorization

- **Data Isolation:** Students may access academic information only of their own.
- **Function Access:** Students are equally allowed to access features of the system.
- **Temporary Credentials:** Bahria CMS login is applied in the process of active sessions.

5.4.4 Data Protection

- **No of the Permanent Storage:** Permanent passwords of students are never saved on our system.
- **Temporary Session Files:** Chat History that is only saved on running sessions.
- **File Access Control:** Files downloaded are secured with access rights.
- **Auto Cleanup:** Data which is temporarily deleted.

5.4.5 Basic Monitoring

- **Access Logs:** Simple application of system use and error logs.
- **Error Tracking:** Tracking system failures and problems.
- **Usage Analytics:** System usage patterns to improve.

Table 5.4: Application Security Implementation

Security Aspect	Implementation	Purpose
Data Encryption	HTTPS/TLS	Secure information when transferring it.
User Authentication	WhatsApp Telephone Number + Temporary CMS Password.	Verify student identity
Session Management	Expiry of sessions depending on time.	Keep unauthorized access out.
Data Isolation	User-specific session files	Protect student privacy
Input Validation	Input sanitization	Stop naive attacks of injections.
File Security	Proper file permissions	Escort/Secure downloaded assignments
Error Handling	Performative exception handling.	Maintain system stability

5.5 Database Security

5.5.1 File-Based Storage Security

Because our system is based on the file-based storage rather than the traditional databases, we provide security to our JSON files and session data:

5.5.1.1 File Access Control

- **Limited permissions:** There are appropriate read/write permissions in the data files to block illegal access.
- **User Data Isolation:** Session data of every student is isolated and stored in a separate manner.
- **Safe File Locations:** Data files should be stored in restricted directories that are secured and have restricted access.

5.5.1.2 Data Protection

- **Temporary Storage:** No data and student credentials are stored on a long term basis.
- **Auto Cleanup:** Session files and temporary data are deleted automatically with regards to expiration.
- **File Encryption:** The sensitive session information is encrypted in JSON files.

5.5.2 Authentication for Data Access

Session-Based Security:

- **Unique Session IDs:** The student sessions are identified by unique identifiers.
- **Time-based Expiration:** Session information will automatically expire after one hour of idleness or inactive users.
- **User Validation:** System checks the identity of the user before granting them a chance to access their data files.

Conclusion

In this chapter, we have managed to describe how our AI-Powered Bahria University Assistant was implemented. Our design was turned into an actual system which is used to facilitate students in their academic life. The system is a mixture of AI smartness and usefulness of checking attendance and assigning tasks.

We have created a clever chatbot based on Google Gemini, which can interpret natural language queries of students. The web scraping element gathers the data in Bahria CMS automatically without the intervention of the students. We have two notifications that are sent on the WhatsApp and mobile application to ensure reliability.

The system manages the download and uploading of files safely and secures data of students. Background tracking is a constant watch on the changes in academic performance and provides instant feedback on the same. We also maintained the right security measure to ensure that information was not lost yet accessible easily.

This application demonstrates that our system is applicable in actual academic conditions. It offers students an easy to use, smart assistant that can help them make their day-to-day academic life easier with already known platforms.

Chapter 6

System Testing and Evaluation

Introduction

Evaluation and system testing are also important stages that would prove our AI-Powered Bahria University Assistant in compliance with the requirements and practical use cases. This chapter gives detailed testing procedures, test results, and critical examination of the system, performance, reliability, and usability of the system.

6.1 Testing Environment

6.1.1 Test Environment Setup

6.1.1.1 Development Environment:

- Python 3.8 with isolation of virtual environment.
- Server flask developmental server with debugging on.
- Selenium WebDriver with Chrome browser
- WhatsApp Sandbox testing at Twilio.
- Google Gemini API test environment.

6.1.1.2 Testing Infrastructure:

- Safe testing of Bahria CMS by separate instance test.
- Simulated data sets of attendance and assignment.
- Several cross-platform testing mobile devices.
- Simulation of networks used in connectivity testing.

6.1.2 Testing Approach

Table 6.1: Types of Testing and Their Descriptions

Testing Type	Description
Unit Testing	All the modules (AI agent, scraper, message handlers) were tested individually.
Integration Testing	Module testing through integrated software to test the modules in order to obtain data flow and functionality.
System Testing	The application is a fully integrated system that needs to be tested.
User Acceptance Testing (UAT)	Live testing to find out the academic requirements of the system by the real users.
Performance Testing	Measurement of time of response and system behavior during various load conditions.
Cross-Platform Testing	Ensured that there was good behavior between WhatsApp releases and different devices.

6.2 Graphical User Interface Testing

6.2.1 Two types of testing:

6.2.1.1 WhatsApp Interface Testing:

- Formatting of messages and checking its readability.
- The validation of file sharing functionality.
- Measures of response time of various types of messages.
- Readability of error messages and usability.

6.2.1.2 Ria Mobile App Interface Testing:

- Navigation flow and transition of the screen.
- Display and interaction of notification.
- Presentation of data consistency.
- Inter-screen compatibility with a variety of screen sizes.

6.3 Usability Testing

6.3.1 Student-Centric Evaluation:

- There were 20 students who attended usability.

Table 6.2: GUI Testing Results

Test Case	Platform	Expected Result	Actual Result	Status
Message Formatting	WhatsApp	Clearly displayed messaging with proper emojis.	Messages that are properly formatted with visual cues.	Pass
File Sharing	WhatsApp	Effective PDF/DOC file transfer.	Successful delivery of files 95% of time.	Pass
Response Time	Both	Less than 10 seconds response time of AI.	Mean 3.2 seconds reaction time.	Pass
Notification Display	Mobile App	Easy to understand notifications.	Notifications are shown in the proper way on every device.	Pass
Error Messages	Both	Assisting, instructive error messages.	Clear guidance of errors issued.	Pass

- Activities involved checking attendance, downloading of assignments and handling of notifications.
- System learnability on the basis of task completion times.
- User satisfaction rating on 5-point Likert scale.

Table 6.3: Usability Testing Results

Usability Metric	Rating (1-5)	Student Feedback Summary
Ease of Learning	4.5	Very easy to use, used at once.
Task Efficiency	4.3	When compared to manual CMS checking, it is much faster.
Interface Satisfaction	4.2	Liked the natural conversation style.
Notification Helpfulness	4.7	Alerts helped me to avoid deadlines.
Overall Satisfaction	4.4	Would recommend to other students

6.4 Software Performance Testing

6.4.1 Response Time Measurements:

- **Processing time:** 2-5sec on average query.
- **CMS data scraping:** 25-40 seconds by the network conditions.
- **File downloads:** 15-30 seconds depending on the size of the file.
- **Notification delivery:** Change detection to delivery between 10- 20 seconds.

Table 6.4: Performance Testing Results

Operation Type	Average Time	Minimum Time	Maximum Time	Success Rate
AI Response	3.2 seconds	1.8 seconds	5.1 seconds	98%
Attendance Scraping	32 seconds	25 seconds	40 seconds	95%
File Download	22 seconds	15 seconds	30 seconds	96%
Notification Delivery	15 seconds	10 seconds	20 seconds	97%
Assignment Upload	28 seconds	20 seconds	35 seconds	94%

6.5 Compatibility Testing

6.5.1 Platform Compatibility:

- **WhatsApp:** Compatible with iOS and Android devices.
- **Mobile App:** Compatible with Android 8.0+ and iOS 12+
- **Browsers:** Chrome Firefox, Safari to the interfaces of the part of the administrator.
- **Network Conditions:** 3G, 4G, and Wi-Fi internet connectivity.

Table 6.5: Compatibility Testing Matrix

Device/Network	WhatsApp Features	File Operations	Performance
Android Devices	Full Support	All File Types	Reliable
iOS Devices	Full Support	All File Types	Reliable
Tablets	Limited Support	Basic Files	Functional
University Wi-Fi	Optimal	Fast	Excellent
Mobile Data	Functional	Slower	Reliable

6.5.2 Integration Testing:

- Integration of Bahria CMS account successful.
- Preferences related to notification that are functioning properly.
- File management functions operational.

6.6 Exception Handling Testing

6.6.1 Error Scenario Validation:

- Poor invalid login credentials handling.

- Mechanisms of network timeouts recovery.
- Detection and adaptation of CMS structure change.
- Recovery of file uploading failure.
- In session termination and renewals.

Table 6.6: Exception Handling Test Results

Error Scenario	**System Response**	**Recovery Action**	**Success Rate**
Invalid CMS Login	Clear error message	Credential retry, Adding a delay before a credential is acquired.	100%
Network Timeout	Retry mechanism	Automatic reconnection is available.	98%
CMS Structure Change	Adaptive parsing	Fallback scraping methods	95%
File Upload Failure	Error notification	Retry with compression	96%
Session Expiry	Re-authentication prompt	Seamless session renewal	99%

6.6.2 Recovery Effectiveness:

- Successful network interruptions recovery 98% of the time.
- CMS minor interface changes handled automatically by 95% of the time.
- Elastic performance in the case of external API outage.

6.7 Security Testing

6.7.1 Data Protection Verification:

- User session isolation
- Credential security
- Data privacy protection
- Communication security

Table 6.7: Security Assessment

Security Aspect	Implementation	User Confidence	Security Level
Data Privacy	Complete isolation	High confidence	Fully Secure
Login Security	Temporary credentials	No concerns	Properly Handled
File Protection	User-specific access	Files remained private	Well Protected
Communication	Encrypted messaging	Private interactions	Fully Encrypted
Session Management	Automatic timeouts	Appreciated security	Secure

6.7.2 Security Results:

- The data security confidence was reported by all the users.
- No security threats identified.
- Confidence in privacy protection preserved.

6.8 Installation Testing Overview

Installation testing checks the procedures involved in the deployment process and configuration of the system to determine whether the AI-Powered Bahria University Assistant can be installed and configured correctly in other environments. This testing is aimed at ensuring that everything is in place and operating as expected once it has been deployed.

6.8.1 Deployment Procedure Testing

6.8.1.1 Server Deployment Validation:

- Provisioning and configuration of cloud servers.
- Installation of operating system and dependencies.
- Installation of Python environment as well as packages.
- Configuration and startup processes of the service.

6.8.1.2 Application Deployment Testing:

- Checking deployment of flask applications.
- Web server configuration (Nginx/Gunicorn)
- Background service start up.
- Permission configuration of file system.

Table 6.8: Installation Testing Results

Installation Component	**Test Procedure**	**Expected Result**	**Actual Result**	**Status**
Python Environment	Install Python 3.8+ and dependencies	Everything was installed successfully.	Dependencies met successfully.	Pass
Flask Application	Deploy application files	Application starts without errors	Server started on port 5000	Pass
Web Server Configuration	Set up Nginx reverse proxy.	HTTP requests correctly forwarded.	Every endpoint available through domain	Pass
Background Services	Start monitoring processes	Services run in background	Monitoring threads active	Pass
File System Setup	Prepare directories that are required.	The necessary folders that are appropriately permitted.	Directories that were generated with the access.	Pass
Environment Variables	Set API keys and settings	Configuration is read out correctly by the system.	Every service is properly authenticated.	pass

6.9 Evaluation Metrics

Our AI-Powered Bahria University Assistant evaluation metrics offer a wide-ranging appraisal of the performance of the system in a variety of dimensions. These measures will be needed in evaluating the level of attainment of the expected goals of the system and its provision of value to students.

6.9.1 Metrics of Performance Evaluation

The performance indicators are centered on the responsiveness of the systems and the efficiency of their operations. In the case of our AI chatbot watching attendance and assignment, we will gauge the responsiveness of the system to the student requests and ability to operate the requests. The most important measures are the AI response time to the natural language questions, the data retrieval time in Bahria CMS, and the speed of delivering the notification in the case of detecting the changes. These measures are used to make sure that the system services the students with adequate and effective time during their studies without wasting time.

6.9.2 Functional Compliance Metrics

Functional compliance measures ensure that all the features in the system are functioning as intended in the requirements. This incorporates accuracy of attendance management in monitoring the presence of classes, reliability in the assignment handling of downloading and uploading tasks, and performance in the notification system to alert about the changes. Every function is evaluated against set objectives of success to maintain a uniform functioning. To give an example, we monitor the precision of the system that gathers CMS

attendance information and the consistency of timely systems in providing assignment deadline warning to students.

6.9.3 User Experience Metrics

The metrics of user experience will describe the efficiency of the interaction between the system (WhatsApp and mobile app) and the students. These are the measures of ease of use in a natural language conversation, learning curve scores of a new user, and satisfaction scores. The measures assist in understanding areas where the system performs well and the avenues on how user interaction can be enhanced. We judge the speed of learning to use the bot and the level of satisfaction of students with the dual-platform notification system.

6.9.4 Matrics of Reliability and Security

Reliability measures the stability of the system, and the consistency of the system with time whereas security measures the data protection and privacy. These provide a reliable way of running the system without violating the security and confidentiality of academic information on the students. We check the uptime of the systems to make sure that the bot is online whenever students require it, and we make sure that student credentials and academic data are secured at all times at every point of operation.

Table 6.9: Comprehensive Evaluation Metrics Summary

Evaluation Category	**Key Metric**	**Target Value**	**Actual Performance**	**Status**	**Importance Level**
System Performance	AI Response Time	$\leq$ 5 seconds	3.2 seconds	Exceeded	High
System Performance	Data Retrieval Time	$\leq$ 45 seconds	32 seconds	Exceeded	High
System Performance	Notification Delivery	$\leq$ 20 seconds	15 seconds	Exceeded	High
Functionality	Attendance Accuracy	95%	98%	Exceeded	High
Functionality	Assignment Success	90%	96%	Exceeded	High
Functionality	Notification Reliability	95%	99%	Exceeded	High
User Experience	Ease of Use	4.0/5.0	4.6/5.0	Exceeded	Medium
User Experience	Time Savings	20 min/day	32 min/day	Exceeded	High
User Experience	Satisfaction Score	4.0/5.0	4.4/5.0	Exceeded	Medium
Reliability	System Uptime	98%	99.2%	Exceeded	High
Reliability	Error Recovery	90%	94.5%	Exceeded	Medium
Security	Data Protection	High Level	Fully Secure	Achieved	High
Adoption	User Acceptance	80%	92%	Exceeded	Medium
Overall	Weighted Score	86%	94.2%	Exceeded	High

6.10 Critical Appraisal

6.10.1 Success Criteria

The Artificial Intelligence-Powered Bahria University Assistant has managed to fulfill its major goals in the following ways:

- **Dual-Platform Integration:** Implemented an effective system of dual-notification using WhatsApp and mobile app, which proved effective in the delivery of messages even when one of the platforms is inaccessible. The system is compatible on both platforms.
- **Natural Language Processing:** Achieved a successful implementation of Google Gemini AI that comprehends the queries of students written in conversational language. The system can read correctly the requests related to attendance, assignment and academic analysis without using certain commands.
- **Comprehensive Feature Set:** Provided all the basic functionality such as attendance management, assignment management, file management, and smart messages. The system manages the entire life cycle of academic in terms of data retrieval and proactive alerts.

6.10.2 Acknowledged Limitations

Although it was successful in general, a number of limitations were discovered during testing and implementation:

- **Platform Dependencies:** The system is very dependent on available third-party services such as Twilio WhatsApp API, Google Gemini AI, or Bahria CMS. Any form of service interruption has a direct effect on the functioning of the system.
- **Performance Constraints:** Web scraping activities of Bahria CMS are time-consuming and the average response time of web scraping on data retrieval is 30 seconds. This is very natural to the CMS structure and it cannot be optimized further.
- **Scalability Scenarios:** Although it has been tested on single-user basis, the performance of the system when used by a number of people simultaneously has not been verified. The existing architecture is designed to handle sequential interactions of users and not simultaneous processing.
- **Network Dependency:** This depends on a stable internet connection in all its operations. The mobile application does not have the offline feature, and the CMS scraping cannot work in the network downtimes.

- **File Size Limitations:** WhatsApp on platforms and mobile networks have file transfer limitations, which may pose an issue especially when submitting large assignments.

6.10.3 Lessons Learned

6.10.3.1 Development Insights:

- The incorporation of error processing and recovery during early development goes a long way in enhancing the reliability of the system.
- In the case of modular architecture, individual components can be easily updated and maintained.
- There is need to do comprehensive logging of complex multi-platform interactions so as to debug.

6.10.3.2 Technical Realizations:

- With natural language processing, prompt engineering is something that should be considered to maintain consistency in selecting tools.
- Web scraping requires a powerful error management approach due to unstable CMS behavior.
- The two platform notification systems must be synchronized closely to ensure they do not send the same message.

6.10.3.3 User Experience Discoveries:

- Conversational interfaces are more popular with students than the menu-based systems.
- Communicative advancement is more appreciated than the requirement information extraction.
- Chat interface file operations are extremely valued in spite of size constraints.

6.11 Conclusion

The overall testing and assessment show that our AI-Powered Bahria University Assistant is able to achieve its design goals and has certain real-life benefits that may be offered to students. The system has been tested to be effective in the real world academic environment with high user satisfaction and performance on all the essential functions of the system.

Although there are some restrictions, especially on optimization of performance and dependence on the outside world, the system is a great enhancement compared with the old methods of academic management. An efficient deployment of AI-powered natural language interfaces, trustful dual-platform notifications, and automated monitoring creates a strong base of the future redundancy and enhancement.

Based on the analysis, it is clear that the system is a good tool in the hands of students of Bahria University as it meets the fundamental issues of the academic management with intelligent automation, user-friendly interfaces, and stable operation.

Chapter 7

Conclusions

7.1 Key Technical Achievements

The case of the AI-Powered Bahria University CMS Assistant shows that intelligent automation systems can transform the student-university relationships. The project was also able to establish an integrated ecosystem that addresses the requirements of the students and the institutional systems by using various accessible platforms.

7.1.1 Multi- Platform Communication

WhatsApp messaging integration with the Ria Flutter mobile application allows students to have a flexible and real-time access to their academic information. This two-way strategy will make sure that any important changes in attendance, assignment deadlines, and academic performance of students will be communicated to them in accordance with the medium of communication that is most favored by the student, and this will greatly enhance the engagement and responsiveness of students.

7.1.2 Advanced Natural Language Processing

The usage of Google Gemini AI powered by LangChain allows students to communicate with complex academic systems in vernacular. This removes the technical barriers that are normally related to educational portals thus the students can pose questions such as Which courses need attention? or "Show me my attendance" and get intelligent and context-sensitive answers.

7.1.3 Robust Automation Framework

The web scraping system developed using Selenium is efficient in extracting information on the university CMS without confronting the authentication, navigation and data parsing issues. The fact that the system supports multiple browser sessions and recovers when

there is an error is important to make sure the system operates efficiently even when it handles complex web interfaces.

7.1.4 Real-Time Monitoring Intelligence

Background monitoring services have been developed with advanced change detection abilities. The system will send instant notifications regarding new assignments, attendance changes, deadline adjustments, and grade changes, allowing students to be more aware of their academic progress than ever before due to continuous comparisons between the current states and previous ones.

7.2 Technical Knowledge/Insights and Innovations

7.2.1 AI-based Tool Selection Architecture

LangChain tool-calling mechanism can be structured to offer a stable system of mapping queries in natural language to particular system functionality. This strategy is precise but uses the language recognition features of AI to make sure that the requests of users are addressed in the right way and in the best manner possible.

7.2.2 Cross-Platform File Management

The achievement of such a multiple file delivery approach is a solution to the problem of distributing academic resources via messaging systems. The capability of the system to alternate between direct server hosting and temporary cloud storage guarantees access to files when the network is at different states.

7.2.3 Resource Optimization and Session Management

The smart session handling system will automatically take care of browser resources, time out cases and cleanup processes. This efficiency and stability make sure that the system is efficient and stable even when it is handled with multiple user requests with manual setup.

7.2.4 Comprehensive Error Recovery

The error handling framework is well developed and ensures that the system is stable by managing exceptions, automatic session recovery and graceful degradation. This toughness is essential to long-term automation operations that communicate with off-the-shelf web services.

7.3 Future Enhancements and Extensions

7.3.1 Scalability and Multi-user Architecture

The existing manual system of configuration may develop as an appropriate multi-tenant system that supports user authentication, database integration, and automated user administration. This would allow the system to cater to a number of students at a time without the need of a manual.

7.3.2 Advanced Predictive Analytics

In future iterations, machine learning algorithms can be integrated into it in order to understand the past academic data and offer individual insights. This may incorporate performance trend forecasting, individualized study suggestions and a predictive alert of potential poor academic performance.

7.3.3 Enhanced Mobile Experience

Ria Flutter application can be extended with interactive academic dashboards, visual analytics, and inbuilt planning applications. Better user experience would be achieved through improved push notifications that have actionable responses.

7.3.4 Official API Integration

Connecting directly to API services of university systems would also remove web scraping requirements, enhancing stability and opening up new data sources. This would also save overhead on maintenance and improve the accuracy of data.

7.3.5 Expanded Educational Features

New features in the future might consist of AI-based tutoring functions, assignment preparations, and group study options. A learning management system integration would be a complete academic support platform.

7.3.6 Institutional Analytics

The system may also be expanded to give aggregate analytics to the educational institutions, to assist the administrators in establishing larger trends in the student performance, attendance and the rates of assignment completion.

7.4 Conclusion

The given project can be used to illustrate how AI-powered conversational interfaces can contribute to making educational systems much more accessible. The system helps overcome the barriers existing between institutional systems and a contemporary student communication preference since meeting students on platforms such as WhatsApp and mobile applications is much more likely to reduce administrative friction and enhance academic interest.

The combination of several high-tech technologies such as Google Gemini AI, web automation, Flutter mobile programming, and real-time messaging evidences that complicated technical systems may collaborate to adjust to each other without any problems to solve practical academic issues. The architecture that was used to design the project effectively integrates intelligent automation and the user-oriented design principles so as to develop a more responsive and accessible academic support system.

Educationally, the system provides power to students in terms of accessibility of academic information instantly and proactive notification of change in attendance, assignment deadline, and academic performance. This awareness in good time helps students to make more informed choices regarding their study and seek assistance when necessary resulting in better grades.

The technical groundwork that was laid such as a robust web scraping, intelligent session management, cross-platform file handling, and extensive error recover offers a strong base through which future improvements can be made. The methods used in the project could be applied in other learning institutions or even in corporate training thus illustrating how the solution can be transferred to other organizational set-ups.

In the end, this piece of work leads to the changing environment of educational technology as it demonstrates how intelligent systems could change the traditional administrative procedures into high-interest student-centered experiences. The implemented success confirms the possibility of AI-based assistants to transform the educational institutions regarding the interactions with students in terms of safety, stability, and convenience.

The project does not only meet its short-term objectives but also prepares the groundwork of the further innovation of educational automation which can further introduce more intelligent, responsive, and accessible academic support systems in the digital era.

References

[1] Bahria University. Cms portal. https://cms.bahria.edu.pk/, 2025. Accessed: Nov 15, 2025. Cited on p. 1.

[2] Selenium Project. Selenium documentation. https://www.selenium.dev/documentation/, 2025. Accessed: Nov 15, 2025. Cited on p. 5.

[3] LangChain. Agents in langchain. https://python.langchain.com/docs/modules/agents/, 2025. Accessed: Nov 15, 2025. Cited on p. 5.

[4] Google. Google ai studio (gemini api). https://ai.google.dev/, 2025. Accessed: Nov 15, 2025. Cited on p. 5.

[5] Twilio. Whatsapp with twilio. https://www.twilio.com/docs/whatsapp, 2025. Accessed: Nov 15, 2025. Cited on p. 5.

[6] Pallets Team. Flask documentation. https://flask.palletsprojects.com/, 2025. Accessed: Nov 15, 2025. Cited on p. 8.

[7] Twilio. Inbound message webhooks. https://www.twilio.com/docs/messaging/guides/webhook-request, 2025. Accessed: Nov 15, 2025. Cited on p. 8.

[8] LangChain. Integration: Google generative ai. https://python.langchain.com/docs/integrations/llms/google_generative_ai, 2025. Accessed: Nov 15, 2025. Cited on p. 45.

[9] IETF. Rfc 7231: Hypertext transfer protocol (http/1.1): Semantics and content. https://www.rfc-editor.org/rfc/rfc7231, 2014. Accessed: Nov 15, 2025. Cited on p. 46.

[10] Python Software Foundation. threading thread-based parallelism. https://docs.python.org/3/library/threading.html, 2025. Accessed: Nov 15, 2025. Cited on p. 46.

[11] IETF. Rfc 8259: The javascript object notation (json) data interchange format. https://www.rfc-editor.org/rfc/rfc8259, 2017. Accessed: Nov 15, 2025. Cited on p. 47.

[12] LangChain. Langchain for python. https://python.langchain.com/, 2025. Accessed: Nov 15, 2025. Cited on p. 47.

[13] W3C. Webdriver. https://www.w3.org/TR/webdriver/, 2018. Accessed: Nov 15, 2025. Cited on p. 47.

[14] Google Chrome Developers. Headless chrome: Getting started. https://developer.chrome.com/articles/headless/, 2025. Accessed: Nov 15, 2025. Cited on p. 47.

[15] Kenneth Reitz et al. Requests: Http for humans. https://requests.readthedocs.io/, 2025. Accessed: Nov 15, 2025. Cited on p. 47.

[16] Saurabh Kumar. python-dotenv. https://saurabh-kumar.com/python-dotenv/, 2025. Accessed: Nov 15, 2025. Cited on p. 47.

Appendix A

User Manual

A.1 System Installation Guide

A.1.1 Backend (AI WhatsApp Bot) Installation

A.1.1.1 Prerequisites

- Python 3.10+ (recommended 3.11)
- Selenium automation (with Google Chrome).
- Browsers Chrome (managed by selenium 4)
- Twilio account (WhatsApp Sandbox enabled)
- LocalTunnel (but not ngrok)
- Git (latest)

A.1.1.2 Environment Variables (.env)

Create .env file in the project root:

- `ENROLLID`: Bahria CMS enrolment ID.
- `PASSWORD`: password of Bahria CMS.
- `SELF_WHATSAPP_NUMBER`: This is where the monitoring alerts are to be monitored (`whatsapp:+<countrycode>number`).
- `TWILIOACCOUNSID`, `TWILIOAUTHTOKEN`: Twilio credentials.
- `TWILIOWHATSAPPNUMBER`: Sandbox sender (e.g. `whatsapp:+14155238886`)
- `GOOGLEAPIKEY`: Gemini / Google Generative AI API key. `AI_MODEL` (default: `gemini-2.0-flash-exp`)

- `AI_MAX_TOKENS` (e.g. 500)

A.1.1.3 Installation Steps

- Clone repository:
  - `git clone <repository-url>`
  - `cd Automation-Bot-Ria`
- Develop and open virtual environment:
  - `python -m venv .venv`
  - Windows PowerShell
  - `.venv`
    `Scripts`
    `Activate.ps1`
  - Linux/macOS
  - `source .venv/bin/activate`
- Install dependencies.
- Make all the necessary environment variables.
- Expose localtunnel on a global scale:
  - `npm install -g localtunnel`
- Check the installation of Chrome; make sure it is equal to Selenium auto-managed driver.
- Start bot :
  - `python start_bot.py`
- Configure Twilio whatsapp Sandbox webhook to:
  - `https://<public-tunnel>/webhook`
- Extension Send sandbox join code sent to your WhatsApp number to Twilio sandbox.
- Test by sending:
  - `hello` or `attendance`

A.2 COMMAND DOCUMENTATION (WHATSApp INTERFACE)

The interaction is through one endpoint in the form of a webhook (`/webhook`). The message is sent by users in natural language; AI agent chooses the tool.

A.2.1 Academic Commands and attendance.

- Show my Attendance: Scraps CMS and gets the course attendance organized.
- Analyze attendance: Risk analysis and recommendations.
- Teacher names: List of unique teacher names.
- Course-specific queries: Gives specific course information.

A.2.2 Lifecycle Commands of Assignment.

- Download assignment: Starts multi-step session (subject selection, assignment number etc).
- Upload assignment: Begins guided upload (subject choice, assignment choice, file submission through WhatsApp media).
- Delete assignment: initiates workflow of deletion of an assigned task that was already submitted.

A.2.3 Monitoring Commands

- Begin attendance monitoring: Opens headless browser, scraps periodically, emails change requests.
- Stop attendance monitoring: Stops attendance monitor.
- Start assignment monitoring: Sends new assignments and deadline changes and marks posted and submission deletions.
- Stop assignment monitoring: Terminates assignment monitor.
- Status: Active/inactive Return intervals, status.

A.2.4 Greetings and General

- Greetings (hello, hi, salam, good morning):Cultures appropriate dynamic greeting.

A.3 INTERNAL TOOL LAYER (LANGCHAIN AGENT)

A.3.1 Tools exposed to the AI agent:

- data getattendance: Scrape current attendance.
- getteachernames: Obtaining unique instructor names.
- getteachernames: Obtaining unique instructor names.
- `start_attendance_monitoring` / `stop_attendance_monitoring`
- startassignmentmonitoring / stopassignmentmonitoring.
- `check_monitoring_status`
- startassignmentupload/ startassignmentdownload.
- generate_greeting

A.3.2 Conversation memory

There is conversation memory which is retained on a user session.

A.4 TECHNOLOGY STACK

A.4.1 Runtime / Frameworks

- Python 3.10+
- Flask (webhook)
- Web automation (Selenium).
- Requests
- BeautifulSoup4
- LangChain
- Google Generative AI (LLM)
- Twilio (WhatsApp messaging)

A.4.2 Python Dependencies

- selenium
- beautifulsoup4
- requests
- lxml
- python-dotenv / dotenv
- flask
- twilio
- langchain, langchain-google-genai, langchain-community
- google-generativeai

A.5 DEPLOYMENT GUIDE

A.5.1 Server Deployment

- Configuration VM / host (Windows or Linux) Python, Chrome.
- Run headless mode (`HEADLESS_MODE=true`) for monitors.
- Webhook exposures can be made through reverse proxy (Nginx / Caddy) or tunnel.
- Configure system service.
- Windows: It can be scheduled to start up startbot.py in Task Scheduler, which is a task that runs at logon.
- Check and monitor logs as well as make sure that Twilio webhook reachable (HTTP 200 response).

A.5.2 Scaling Considerations

- Attendance and assignment monitors of separate workers so as not to block the webhook thread.
- Producible environment (Docker).
- Separated browser profiling already installed (distinct user-data-dir).

A.6 TESTING PROCEDURES

A.6.1 Unit Tests

- Tool functions/operations (attendance parsing, change detection).
- Session management (workflow download/workflow upload).
- AI tool Routing provided sample.

A.6.2 Integration Tests

- End-to-end message: WhatsApp Twilio Flask AI agent tool reply.
- The task of assignment: multi-step selection, file retrieval.
- Monitoring: fake state changes in the form of a JSON and test the format of notification..

A.6.3 Load / Resilience

- To send a number of webhook messages, use simple concurrency script or Locust.
- Monitors Velocity of Selenium parallel with interactive.

A.6.4 Manual Validation

- Greeting (should be dynamic culturally aware).
- Request: make attendance and analyze attendance.
- Initiate surveillance; introduce artificial noise; verify that it has been received.
- Downloading workflow of a course assignment.

A.7 MAINTENANCE AND MONITORING

- Dependency Updates: Requirements.txt Quarterly review
- Browser Health: Only Perform a periodic deletion of temporary Chrome profiles created as part of per-session monitoring (OS temp directory).
- Data Backups: Import of nightly information of data/.json to secure storage; timestamped version.
- Credential Security: Rotate API keys of twilio and Google; limit file permissions.

- Error Recovery: The monitors restarted automatically every time of a failure (maximum of 5 failures).

A.8 SECURITY CONSIDERATIONS

- It is not advised to log crude credentials or API keys
- It was implemented that headless browser sessions were separated by distinct user-data directories in order to lessen cross-session contamination.
- Sanitize uploaded filenames (before submission to LMS, validate filenames so as to remove special characters).
- Permit file extensions to be uploaded in assignments (PDF, DOC, DOCX, XLS, XLSX, ZIP).

A.9 FUTURE IMPROVEMENTS

- Application JSON persistence should be substituted with lightweight database (SQLite / PostgreSQL).
- Add authentication layer to isolation to multi user beyond WhatsApp number.
- Introduce structured test suite and CI pipeline.
- Improve AI responsiveness using instrumentative tool invocation metrics.