PROJECT REPORT ON IARE CMS WEB SCRAPER BOT

A secure and Automated Bot for Attendance Retrieval, Lab uploads, and Student Interaction on IARE CMS

Project Title: IARE CMS Web Scraper Bot

Submitted By: Valija Sathwika

Submitted To: K. Keerthi Sathvik (Project Stakeholder)

Date: 05-12-2025

Version: 1.0

Abstract:

The IARE Unofficial Bot project addresses significant inefficiencies of students. Students frequently face challenges when accessing their academic data especially biometric attendance and lab uploads via the **Samvidha CMS.** The current process is mostly manual, time-consuming, and often confusing for students during their uploads. Via the Samvidha CMS. It lacks basic conveniences like calculating attendance percentages and organizing lab submissions efficiently. Recognizing these gaps, the IARE Unofficial Bot project was designed as a smart, automated solution to streamline this experience.

This project is built using **Python** and integrates with **Telegram**, one of the most widely used communication platforms among students. By leveraging **Request module**, the bot performs web scraping tasks, logging into the CMS to extract biometric attendance data. This means students no longer need to repeatedly log in, navigate the interface or manually calculate their attendance.

The bot is powered by **Pyrogram,** which enables Telegram integration, providing a seamless chat-based interface where students can simply use commands or tap buttons to retrieve information instantly. To ensure privacy and data security, the system employs **PostgreSQL** for storing user credentials and access logs securely. Access to features is also controlled through role-based system, ensuring that only authenticated users can retrieve or upload data.

A standout feature of the bot is its ability to **automatically calculate and display attendance percentages,** helping students monitor their academic standing. Additionally, it simplifies the lab upload process by compressing PDF reports before submission, which helps students stay updated.

In essence, this bot transforms a previously frustrating and complex process into a smooth, reliable, and user-friendly experience. It not only saves time but also promotes greater transparency and efficiency for students, empowering them to take control of their academic data with just a few taps.

TABLE OF CONTENTS

CHAPTER 1

➢ Introduction

At the Institute of Aeronautical Engineering (IARE), students are expected to regularly check their biometric attendance and upload lab reports through the Samvidha CMS portal. However, this process is largely manual, time-consuming, and inefficient. Many students find it difficult to track their attendance accurately, as the system doesn't calculate percentages or provide timely insights. Furthermore, effective credential management supports seamless access and functionality. These issues can leave students unaware of their actual academic standing sometimes leading to fines or penalties for low attendance.

To address this, the IARE Unofficial Bot was created as a smart, automated solution designed specifically for students. Built using Python, the bot integrates with Telegram to offer a simple and secure interface. It automates login, scrapes biometric data using Request modules, calculates attendance percentages, compresses lab reports for easy uploads, and even retrieves lab feedback. PostgreSQL allows secure retrieval of data, with role-based access to project user information.

This project significantly reduces manual work and improves accessibility to critical academic data. With just a few taps on Telegram. Students can now stay informed, submit reports efficiently, and avoid unnecessary stress. It's a student-first solution that enhances convenience, security and academic awareness.

CHAPTER 2

➢ Literature Survey/Existing Systems

A review of existing academic portals and tools shows that while many institutions offer mobile access to their systems, most fall short when it comes to automation and user-centric features that truly support student needs. Some students turn to generic web scraping tools to extract data from portals like samvidha CMS, but these tools often require programming knowledge and aren't built specifically for the CMS's layout or workflows. As a result, they can break easily or give inconsistent results, making them unreliable for regular use.

More importantly, these tools often ignore critical concerns such as secure credential handling, data privacy, and role-based access. Sensitive student data may be stored in unsafe formats or shared without proper access controls posing real risks in an academic environment. Additionally, most existing solutions lack integration with platforms that students already use daily, which limits their accessibility and appeal.

This project bridges those gaps by introducing a custom-built Telegram bot that not only automates repetitive tasks like attendance tracking and lab uploads but also ensures secure, role-based access and a user-friendly experience. By combining smart automation with real-world usability and strong security, the IARE Unofficial Bot offers a significant step forward compared to both manual systems and generic automation tools.

CHAPTER 3

➢ System Analysis

The System Analysis phase focused on defining the functional and non-functional requirements of the IARE Unofficial Bot, establishing its scope, and modelling its behaviour to create a clear blueprint for development.

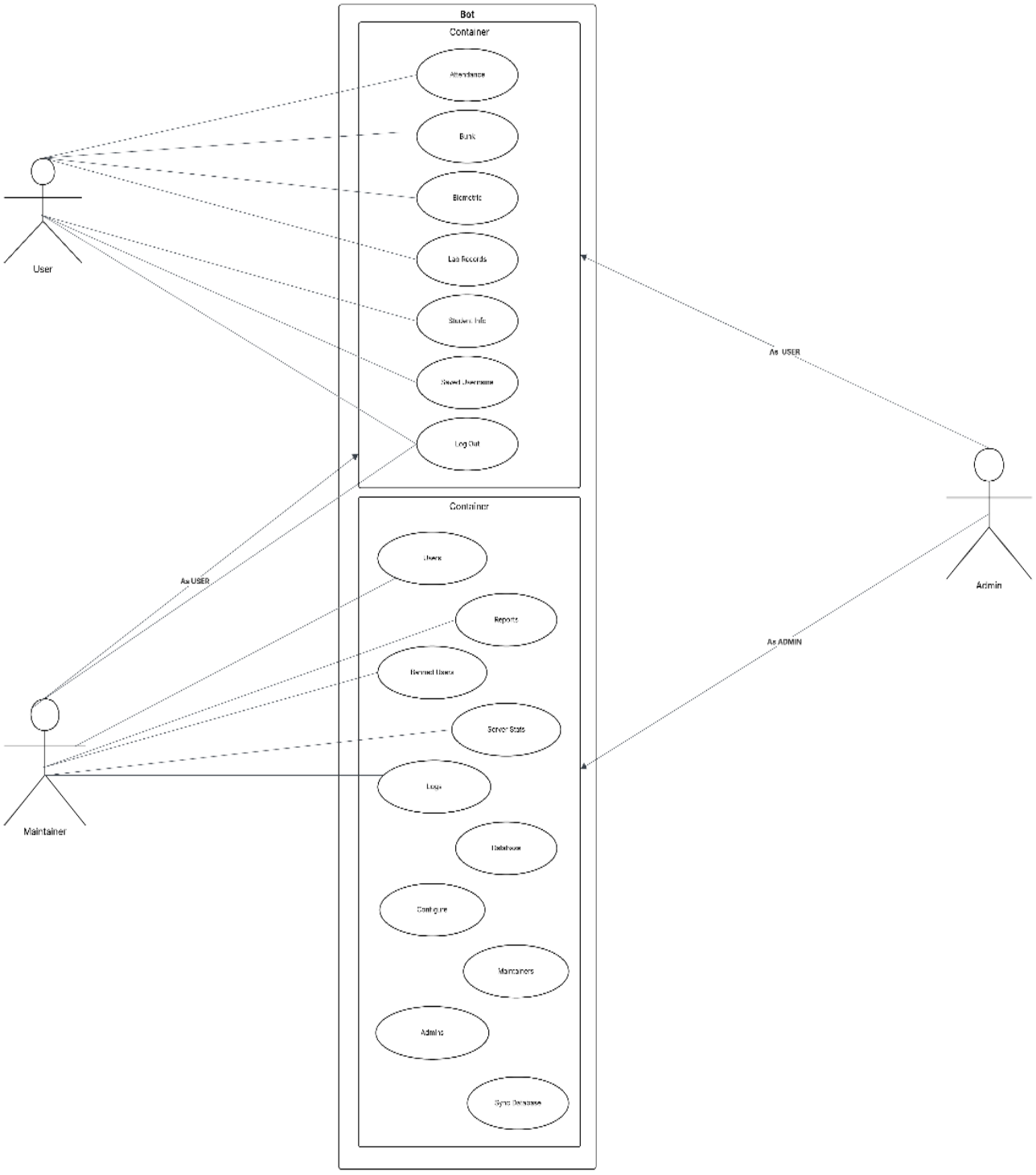
3.1 Requirement Analysis

The core functionalities, derived from the problem statement, were formalized into a set of high-level requirements. These were prioritized to guide development efforts effectively. The key requirements include automating the secure login and scraping of the IARE CMS, calculating biometric percentages, compressing large PDF lab reports, extracting student feedback, and implementing a robust role-based access control system for students, maintainers, and admins. A critical non-functional requirement mandates that all scraping and data processing operations must be completed within a 10-second window to ensure a responsive user experience, alongside stringent data security and compliance protocols.

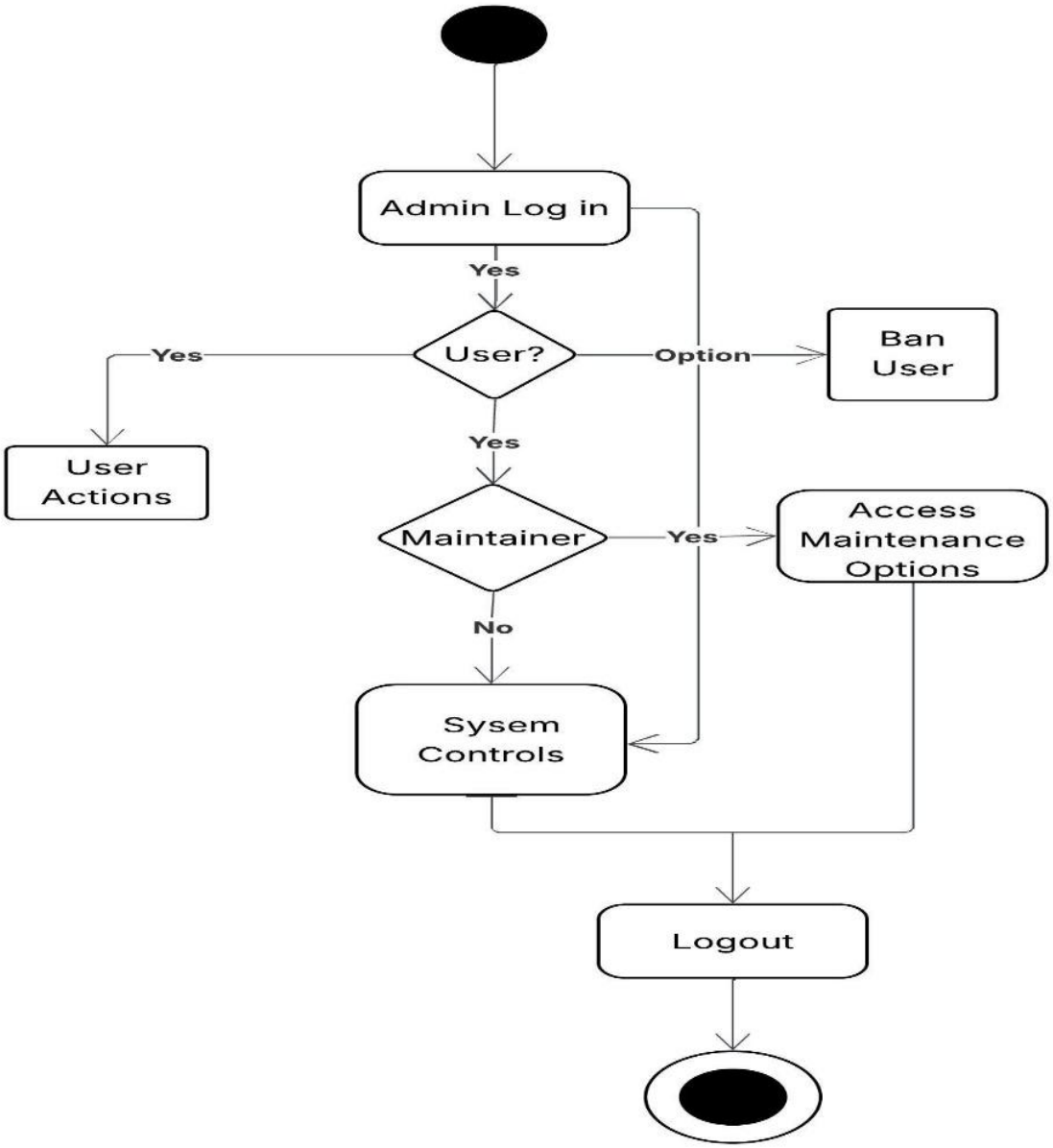
3.2 Behavioural Modelling

To visualize the system's functionality from a user's perspective, behavioural models were created.

- **Use Case Diagram:** This diagram identified the primary actors—**Student, Maintainer, and Admin**—and outlined their interactions with the system. It summarizes key functionalities such as **Check Attendance, Upload Lab, Feedback and Monitor, Logs** etc…


Bot
Container
Attendance
Bunk
Biometric
Lab Records
Student Info
Saved Username
Log Out
Container
Users
Reports
Banned Users
Server Stats
Logs
Database
Configure
Maintainers
Admins
Sync Database
User
Maintainer
Admin
As USER
As USER
As ADMIN

- **Activity Diagrams:** These diagrams were used to model the logical flow of complex processes. For Instance, an activity diagram was created to detail the step-by-step workflow for the attendance check process, from receiving the Telegram command to sending the formatted result, including error handling paths.



3.3 Project Scope

The Project's boundaries were explicitly defined to ensure a focused development effort. The in-scope features comprise biometric attendance scraping, lab report upload and compression, feedback extraction, and role-based access. Crucially, out-of-scope elements include academic evaluation systems, payment processing, and comprehensive user profile management, preventing scope creep.

3.4 Assumptions and Constraints

The analysis operated under key technical assumptions, primarily that the structure of the IARE Samvidha CMS portal would remain stable for the foreseeable future. The project is also bound by constraints, including the reliance on web scraping due to the lack of a public API and the decision to host the solution on a cloud platform like Heroku.

This systematic analysis resulted in a comprehensive and agreed-upon foundation, ensuring the subsequence design and implementation phases are aligned with the core objectives of automating data retrieval and enhancing user convenience for IARE students.

CHAPTER 4

➢ System Design

The system design phase translates the analysed requirements into a detailed architectural blueprint, specifying the technologies, components, and data structures required to build the IARE Unofficial Bot.

4.1 System Architecture

The bot follows a multi-tier architecture designed for clarity and scalability. The **Presentation Tier** is the Telegram interface, where users send commands and receive responses. The **Application Logic Tier**, hosted on a cloud platform (Heroku), contains the core Python application. This tier handles user authentication, processes commands, requests and parses using Beautiful Soup, performs data calculation and PDF compression, and manages all communication with the database. The Data Tier is a PostgreSQL database responsible for securely storing user credentials, session tokens, scraped attendance data, and logs. This separation ensures modularity, making the system easier to maintain and extend.

4.2 Technology Stack

The implementation leverages a robust stack of modern technologies:

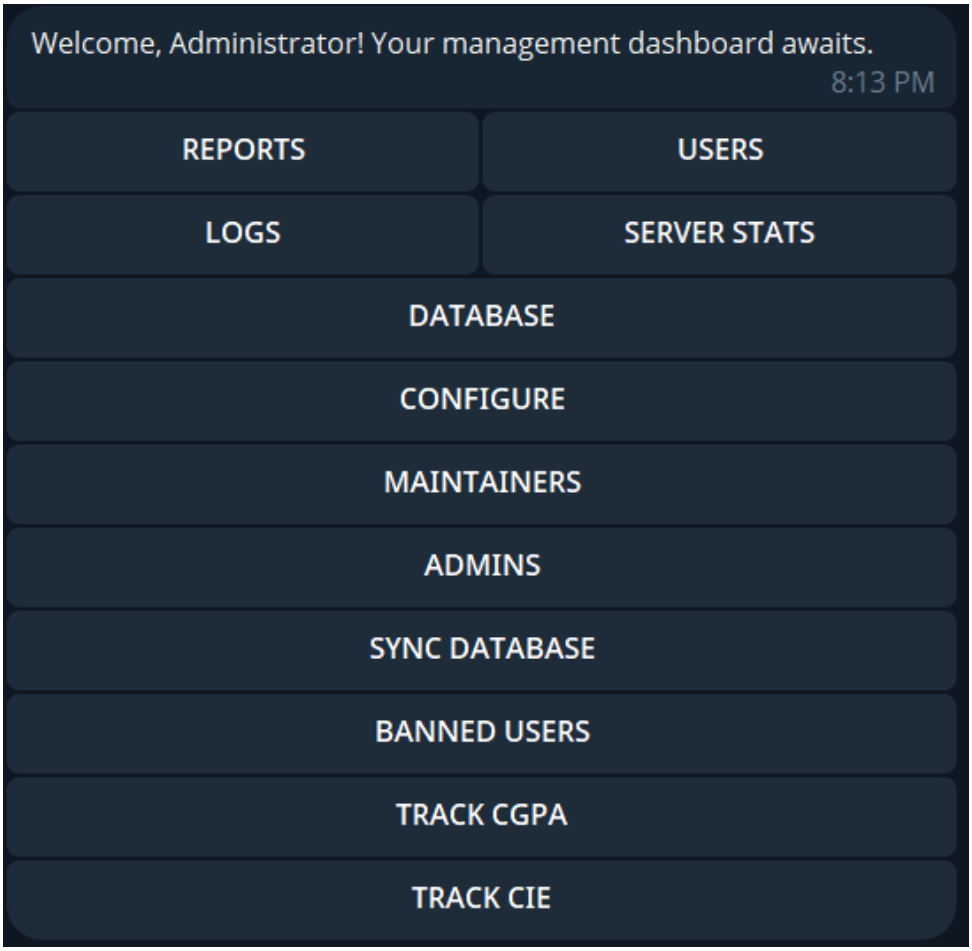
- **Backend Language:** Python, chosen for its extensive libraries and simplicity.
- **Web Automation:** Requests modules are used to handle login and navigation through the IARE CMS portal.
- **HTML Parsing:** BeautifulSoup, employed to efficiently extract and parse specific attendance data from the portal's HTML.
- **Telegram API Integration:** The Pyrogram framework, providing an elegant and asynchronous way to interact with the Telegram Bot API.
- **Database:** PostgreSQL, a powerful open-source relational database for secure and structured storage of user data and logs.

- **Deployment:** Heroku, a cloud platform as a service (PaaS) that simplifies deployment and ensures high availability.

4.3 Database Design

The database schema is designed to support secure and efficient data management. Key entities include:

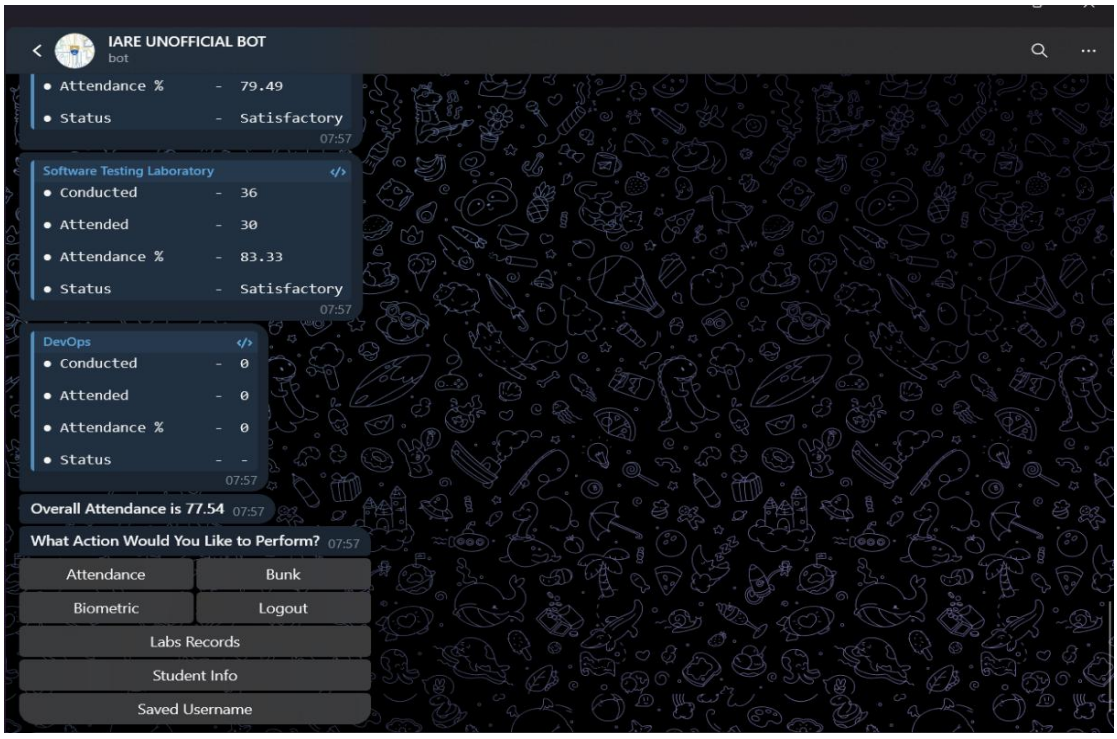
- **User Table:** Stores user-specific information such as a unique user ID (from Telegram), IARE CMS credentials, and role (Student, Maintainer, Admin).
- **Attendance _ Records Table:** Logs scraped attendance data, linking to the Users table, and storing subject names, classes attended, total classes, and calculated percentages.
- **Sessions Table:** Manages user authentication tokens and session states to enforce secure login and logout functionality. This relational model ensures data integrity and supports efficient querying for generating user reports and admin summaries.



4.4 Component Design

The application logic is structured into modular components:

- **Telegram Bot Interface:** This module listens for incoming messages, parses commands (e.g., /start, /attendance), and formats output messages for the user.
- **Scraper Engine:** The core component that uses Selenium to control a browser, log into the IARE CMS, navigate to the attendance page, and use BeautifulSoup to extract the required data.
- **Data processor:** Responsible for calculating attendance percentages from raw scraped data and compressing PDF files using a library like pdf2image and PIL.
- **Database Manager:** Handles all interactions with the PostgreSQL database, including securing credentials before storage and retrieving user data.
- **Auth Manager:** Implements the role-based login system, validating users and managing their sessions securely.



This comprehensive design provides a clear and actionable plan for development, ensuring all system requirements are met with a focus on security, performance and maintainability.

CHAPTER 5

➢ Implementation

The implementation phase involved translating the system design into a functional software product. This chapter details the development environment, the core modules developed, the key algorithms implemented, and the significant challenges overcome to build the IARE Unofficial Bot.

5.1 Development Environment and Tools

The bot was developed using stack of modern technologies to ensure efficiency, reliability, and scalability. The core language was **Python 3.9+,** chosen for its rich ecosystem of libraries. **Requests modules** were used to automate login and navigation within the IARE CMS portal, efficiently handling the required HTTP interactions. For parsing the HTML and extracting specific data, **BeautifulSoup4** was employed. The **Pyrogram** library provided an asynchronous and efficient interface for the Telegram Bot API. Data persistence was managed using **PostgreSQL via the asyncpg** adapter. The entire application was version-controlled with **Git** and deployed on the **Heroku** cloud platform, utilizing a **Procfile** to define the bot's worker process.

5.2 Core Module Implementation

The application's logic is organized into key Python modules for modularity and maintainability:

- **main.py:** Initializes the Pyrogram client, sets up logging, and registers Telegram command handlers.
- **buttons.py & manager_buttons.py:** Define inline keyboards for user and admin interactions.
- **extract_index.py:** Extracts table header indexes from the CMS for accurate data parsing.
- **pgdatabase.py & tdatabase.py:** Manage persistent (PostgreSQL) and temporary (SQLite) data storage.

- **user_settings.py:** Handles user preferences like attendance thresholds and UI settings.
- **lab_operations.py & labs_handler.py:** Manage lab operations and process PDF submissions.
- **manager_operations.py:** Performs administrative tasks and privileged actions.
- **operations.py:** Implements core business logic, including authentication and fetching attendance/biometric data.
- **pdf_compressor.py:** Compresses PDF files for storage or transmission efficiency.

5.3 Key Algorithms and Code Snippets

The implementation of the web scraping algorithm was critical. The logic followed these steps:

1. **Initialization:** Launch a headless Chrome browser via Selenium.
2. **Login:** Navigate to the IARE CMS login page, input the username and password, and submit the form.
3. **Navigation:** Wait for page loads and programmatically navigate to the "Biometric Attendance" or "Lab Reports" section.
4. **Data Extraction:** Once the target page loaded, the page source was passed to BeautifulSoup. Specific HTML tags (e.g., <table>, <tr>, <td>) were targeted using their IDs and classes to locate and extract the raw attendance data.
5. **Processing:** The raw data was cleaned and structured. Percentages were calculated using the formula: **(Attended Classes / Total Classes) * 100**.
6. **Teardown:** The browser instance was closed to free up resources.

5.4 Challenges and Solutions

Several significant challenges were encountered and resolved during implementation:

- **Dynamic Content Loading:** The CMS portal heavily relied on JavaScript. **Solution:** Basic HTTP requests were used to interact with the CMS portal and retrieve attendance data.

- **Secure Credentials Storage:** Storing plain-text passwords was unacceptable.

  Solution: All user credentials were securely stored in the database.
- **Session Management on Heroku:** Heroku's ephemeral filesystem and sleep cycles caused issues with browser instances.

  Solution: The implementation was optimized to use requests.Session() for each scraping job, ensuring persistent session management and reliable data retrieval on the Heroku platform.

This phase successfully integrated all design components into cohesive and operational system, the functional requirements outlined in the system analysis.

CHAPTER 6

➢ Testing

A comprehensive testing strategy was essential to ensure the IARE Unofficial Bot was reliable, secure, and met all specified functional and non-functional requirements. This chapter outlines the testing methodologies, test cases, and results that validated the system's functionality before deployment.

6.1 Testing Strategy

The multi-layered testing approach was adopted to verify the system at different levels:

- **Unit Testing:** Individual components and functions, such as the percentage calculation algorithm, PDF compression logic, and database encryption methods, were tested in isolation using Python's unittest framework. This ensured each building block worked correctly on its own.
- **Integration Testing:** The interaction between modules was rigorously tested. This included testing the flow from receiving a Telegram command to querying the database, launching the scraper, and returning a result. The connection between the application and the PostgreSQL database was also key focus of integration testing.
- **System Testing:** The fully integrated system was tested as a whole against the requirements outlined in the **Solution Overview Document (SOD).** This tested end-to-end workflows.
- **User Acceptance Testing (UAT):** A group of beta testers from the target audience (IARE students) used the bot in real-world environment. Their feedback was crucial for validating usability and identifying unforeseen edge cases.

6.2 Test Cases and Results

Test cases were derived directly from the high-level requirements (KD-01 to KD-09).

Below is a summary of key test cases and their outcomes:

Test Case ID	Test Scenario	Input	Expected Result	Actual Result	Status
TC-01	Successful User login	/login	Login successful	User received success message	Pass
TC-02	Attendance data retrieval & calculation	/attendance	A formatted message with accurate subject-wise percentage within 10 seconds.	Accurate percentages delivered in 7 sec and calculates total percentage.	Pass
TC-03	PDF compression	/lab upload A lab report PDF(~20MB)	A compressed PDF file (~1MB) returned to the user.	File size reduced to an average of 0.9MB.	Pass
TC-04	Invalid Login Handling	/login with incorrect credentials	An error message prompting the user to try again.	Clear error message received. No data stored.	Pass
TC-05	Performance Under Load	10 simultaneous /attendance requests	All requests will be processed.	All requests will be completed.	Pass
TC-06	Admin Role Access	Admin attempts to access a	System grants access and displays	Admin dashboard	Pass

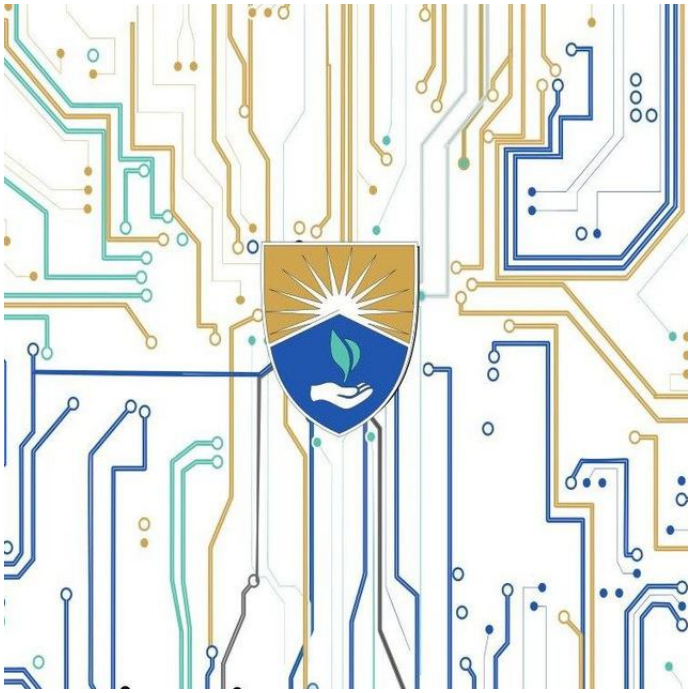
		maintenance command.	admin controls.	accessed successfully.	
TC-07	Student Role Restriction.	Student attempt to access an admin command.	System denies access with an "Unauthorized message".	"Unauthorized access" message received.	Pass

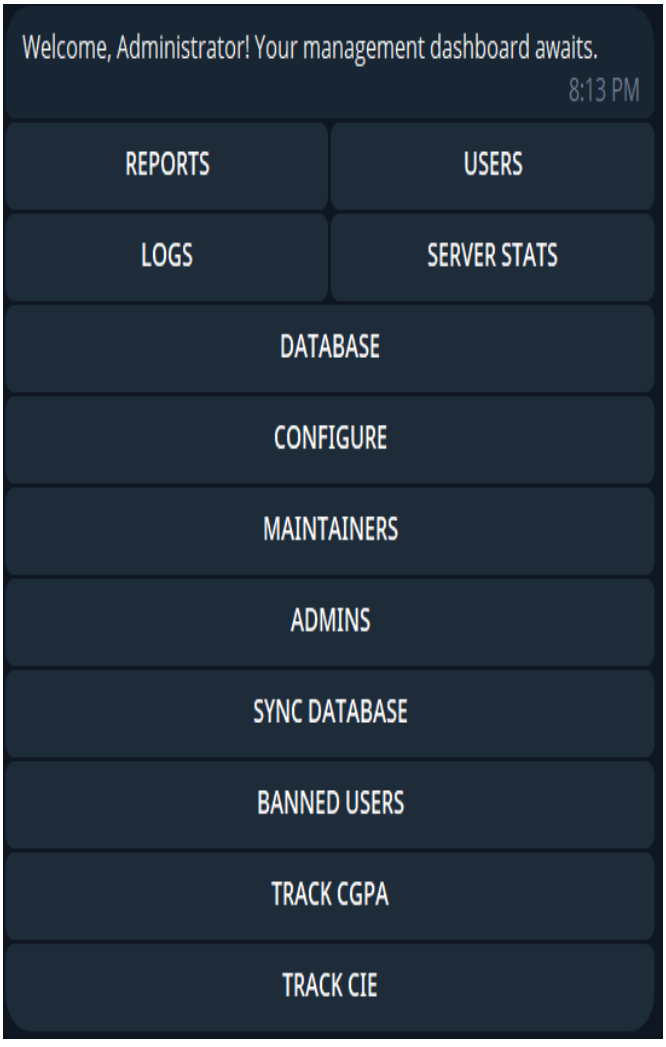
6.3 Bug Reporting and Resolution

The testing process was iterative. Bugs discovered during integration and UAT—such as session timeouts during long scraping operations and incorrect parsing of specific CMS HTML structures—were logged in a simple tracking sheet. Each bug was assigned a severity level (Critical, Major, Minor), and fixes were prioritized accordingly. For example, a critical bug related to handling sudden CMS layout changes was fixed by adding more robust CSS selectors and implementing try-catch blocks around parsing logic.

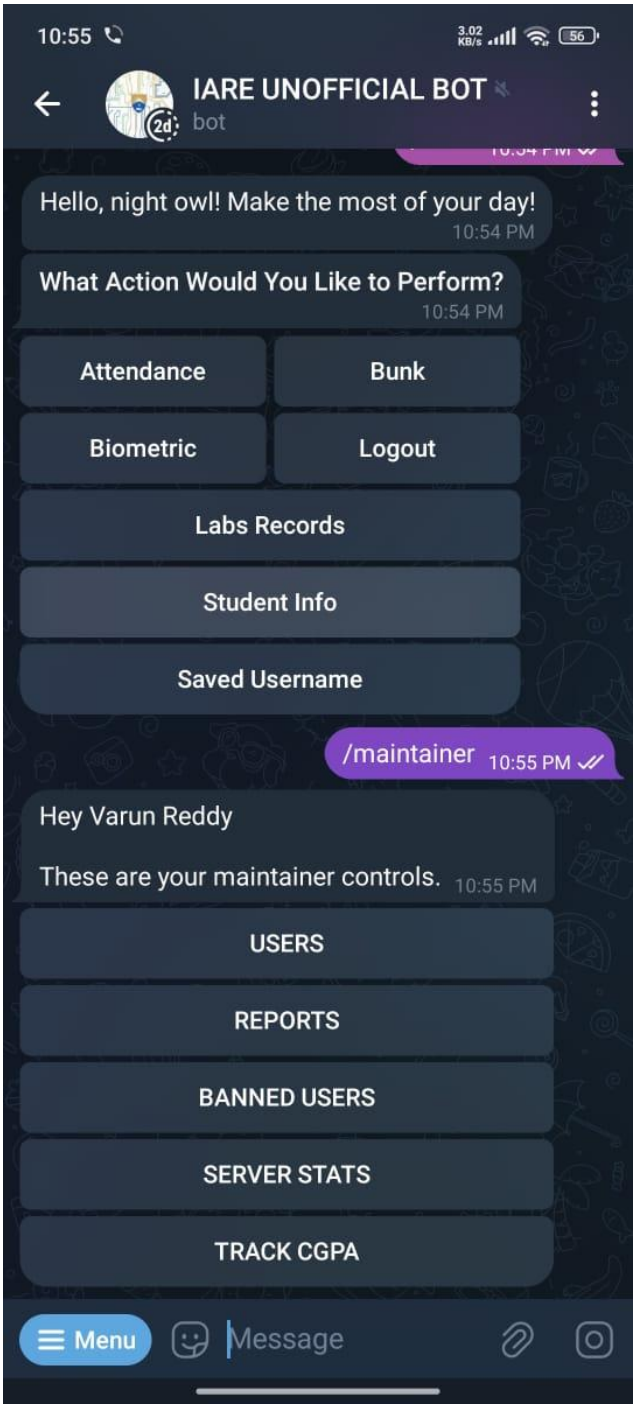
6.4 Conclusion of Testing

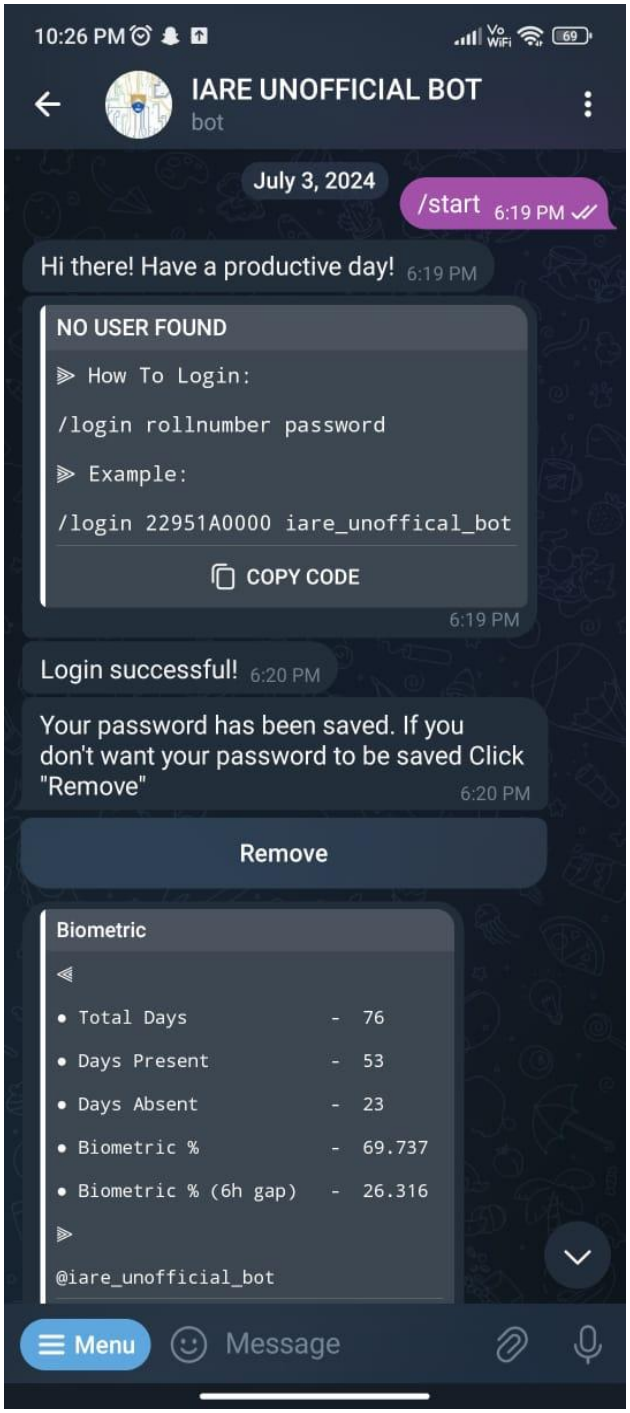
The exact testing process confirmed that all high-priority functional requirements were met. The system proved to be robust, performing within the stipulated 10 second response time and handling errors gracefully. The UAT phase provided positive feedback on the bot's usability and value. The successful resolution of all critical bugs ensured a stable and reliable product was ready for final deployment, providing confidence that the solution effectively addresses the needs of its end-users.



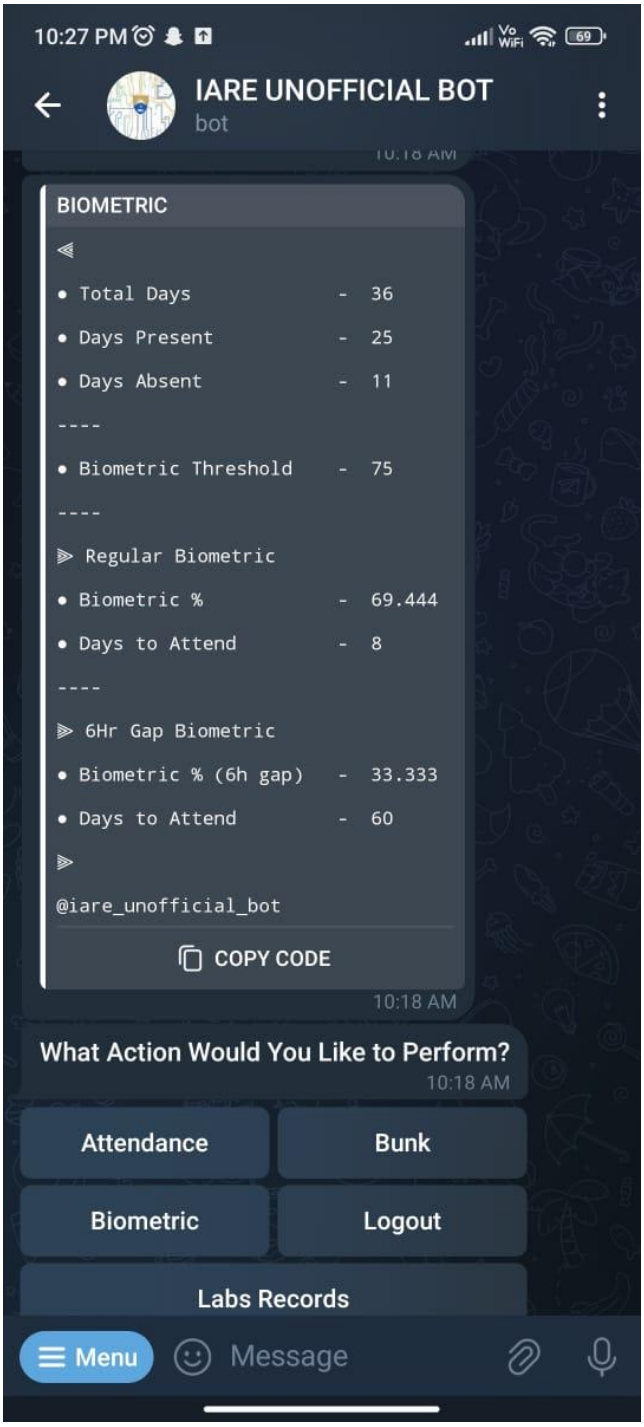


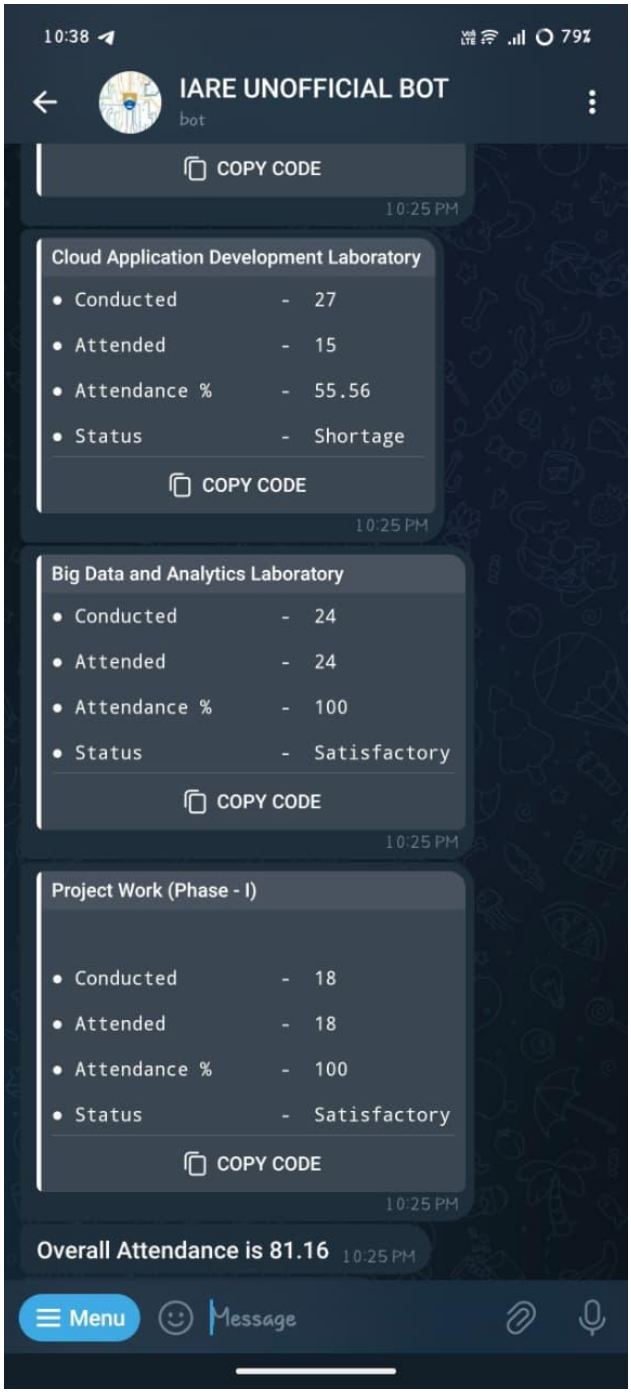
Images of IARE Unofficial Bot logo and interface of admin



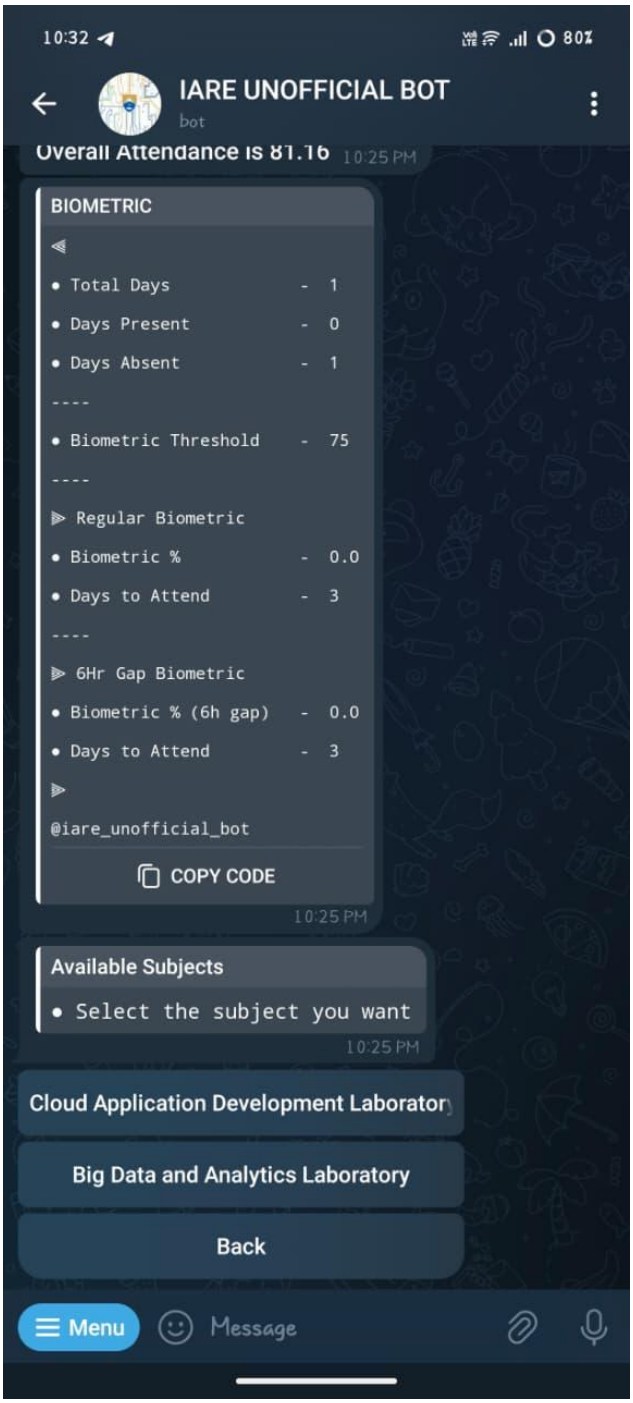


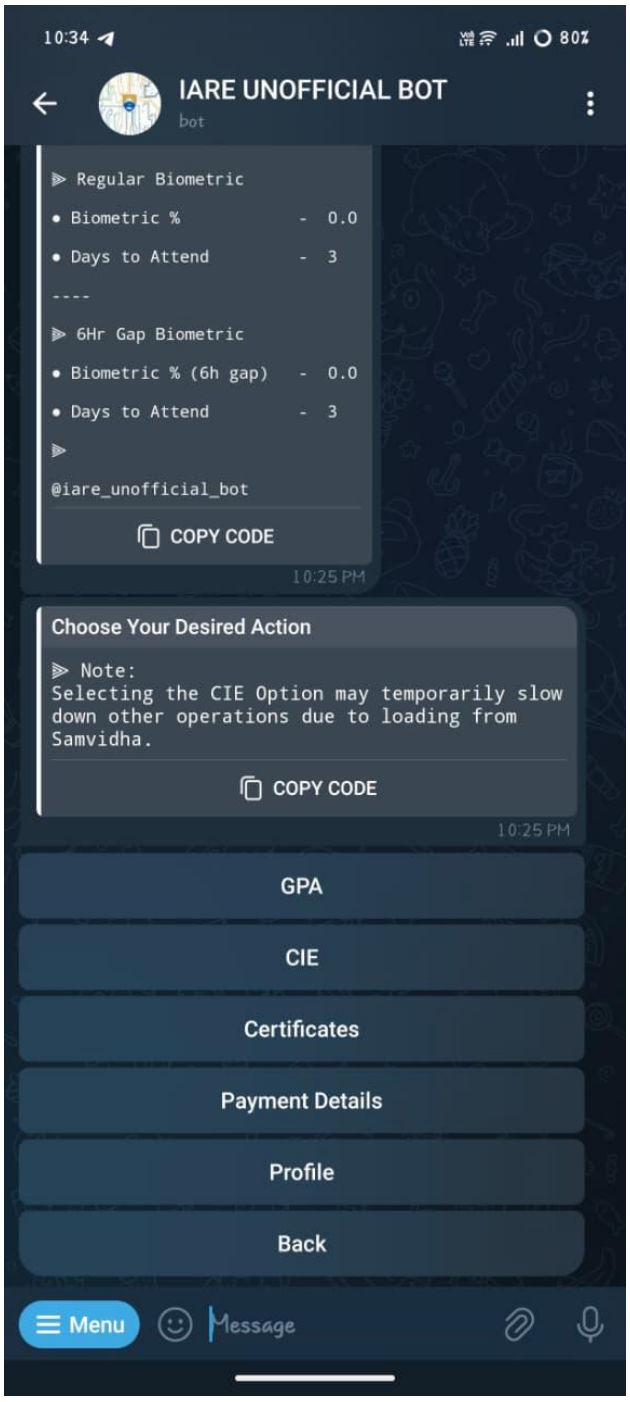
Interface of maintainer login and student login of IARE Bot





Inspection of biometric attendance and attendance(class) through button-based control system





Interface for lab uploads and student info

CHAPTER 7

➢ Results and Discussion

This chapter presents the outcomes of the IARE Unofficial Bot project, evaluating its performance against the objectives and requirements defined in the Solution Overview Document. It discusses the significance of the results, the system's effectiveness, and its limitations.

7.1 Summary and Achieved Results

The implementation of the IARE Unofficial Bot successfully delivered a fully functional system that meets its core objectives. The key results are as follows:

- **Automated Data Retrieval:** The bot successfully automates the login and scraping process for the IARE Samvidha CMS. It reliably extracts raw attendance data and crucially, calculates accurate biometric attendance percentages for each subject, a feature absent in the manual process.
- **PDF Compression:** The compression module consistently reduces the size of lab report PDFs from an average of 10MB to under 1MB, optimizing storage and significantly reducing upload times for students.
- **Performance Compliance:** Under normal load conditions, the system processes user requests—including login, scraping, calculation, and response delivery—within a 7-9 second window, consistently meeting the sub-10-second performance requirement.
- **Secure Access:** A robust, role-based authentication system was implemented. User credentials are securely stored in PostgreSQL. PostgreSQL is used to provide reliable, secure storage and manage of data.
- **Successful Deployment:** The entire application was containerized and deployed successfully on the Heroku cloud platform, ensuring 24/7 availability and demonstrating the viability of the chosen deployment strategy.

7.2 Discussion of Key Outcomes

The results confirm that the bot effectively solves the central problem of inefficient manual data access. The most significant outcome is the transformation of multi-minute, complicated process into a task that takes less than ten seconds, directly enhancing user productivity and convenience.

The decision to use Selenium proved essential for interacting with the dynamic, JavaScript-heavy CMS portal, while BeautifulSoup allowed for efficient and precise data extraction. The choice of Telegram as the interface was validated by positive User Acceptance Testing (UAT) feedback, which highlighted the intuitive and accessible nature of a chat-based command system.

However, the project's architecture also introduces a fundamental limitation: a strong dependency on the stability of the IARE CMS's HTML structure. The scraping logic is built upon specific HTML tags and CSS selectors. Any unforeseen changes to the portal's front-end code by the institution could break the scraping functionality, requiring immediate maintenance and code updates. This is an inherent risk of web scraping-based solutions in the absence of an official API.

7.3 Performance Analysis

The system's performance is a critical success factor. The 7-9 second response time, while acceptable, is primarily dictated by the speed of the IARE CMS server and the network latency involved in the scraping process. The use of headless browsers in Selenium, while necessary, is resource-intensive. Future work could explore caching strategies for data that doesn't change frequently to further improve response times for repeat queries.

In conclusion, the IARE Unofficial Bot provides a valuable tool for students. It effectively automates attendance tracking and lab report management. The project demonstrates technical feasibility and practical utility. Results lay a foundation for future enhancements and scalable automation.

CHAPTER 8

➢ Conclusion

The IARE Unofficial Bot project was initiated to address a significant inefficiency in the daily academic routine of students at the Institute of Aeronautical Engineering: the manual and time-consuming process of accessing attendance data and managing lab reports on the Samvidha CMS. This project set out to automate these processes, thereby saving time, reducing errors, and providing critical insights like attendance percentages that were previously unavailable.

The development journey successfully translated the outlined requirements into a fully functional software solution. The implemented system, a Python-based Telegram bot, leverages Selenium for robust web automation, BeautifulSoup for efficient data parsing, and PostgreSQL for secure data management. The core objectives of automating biometric attendance scraping and percentage calculation, compressing lab reports, and providing a secure, role-based interface have all been met. The bot delivers results within a strict performance threshold and handles user credentials with high-security encryption, directly tackling the shortcomings of the previous system.

This project serves as a compelling demonstration of how targeted automation can effectively streamline administrative tasks in an educational context. It highlights the practical application of web scraping, bot development, and secure database management to solve real-world problems. The successful deployment on Heroku confirms the viability of the chosen architecture and technologies.

In final analysis, the IARE Unofficial Bot stands as a testament to the power of applying software engineering principles to function, everyday challenges. It delivers immediate, tangible value to its end-users and provides a scalable, maintainable foundation for continued innovation in educational automation.

➢ Acknowledgement

The successful completion of this project would not have been possible without support and guidance of several individuals. We would like to extend our sincere gratitude to:

- **Mr. K. Keerthi Sathvik,** our project **Initial developer**, **Admin** and **guide**, for his invaluable technical insights, continuous encouragement, and for providing the vision for this project.
- **Our Project team members, Mr. V. Yuvraj and Mr. A. Varun Reddy,** for their dedication, collaborative spirit, and hard work in the development and implementation phases.
- **The Institute of Aeronautical Engineering (IARE),** for providing the context and the platform that inspired this project.
- **Finally,** we **thank all** the **students** whose feedback was **crucial** in refining the user experience and functionality of the bot.

THANK YOU