Software Requirements Specification

for

JUNO 2.0

Version 1.0

Group Name: UniDevs

Aashi Gupta	SE22UARI003	se22uari003@mahindrauniversity.edu.in
Chadagonda Harshitha Reddy	SE22UARI031	se22uari031@mahindrauniversity.edu.in
D Sri Sanjana Reddy	SE22UARI040	se22uari040@mahindrauniversity.edu.in
Goka Roshan Sai	SE22UARI052	se22uari052@mahindrauniversity.edu.in
Yaswitha Kolla	SE22UARI074	se22uari074@mahindrauniversity.edu.in
Madhulika Munnangi	SE22UARI089	se22uari089@mahindrauniversity.edu.in
P Komal Varma	SE22UARI117	se22uari117@mahindrauniversity.edu.in

Instructor:	*Dr. Avinash Arun Chauhan*
Course:	Software Engineering (CS3201)
Lab Section:	*IT-2 FF*
Teaching Assistant:	Murali Krishna Bukkasamudram
Date:	March 10, 2025

Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
Preliminary V 1.0	UniDevs	Information about the revision. This table does not need to be filled in whenever a document is touched, only when the version is being upgraded.	10/03/25

1 1 Introduction

This document outlines the **Software Requirements Specification (SRS)** for the **JUNO 2.0.** JUNO 2.0 is a campus management system designed to streamline daily university operations. It provides real-time updates on classroom availability, mess balance tracking, faculty consultation scheduling, and a lost-and-found feature. By integrating with existing university systems, the website enhances accessibility, reduces administrative burdens, and improves the overall student and faculty experience. This document details the system's functionalities, external interfaces, and design constraints. It serves as a guide for developers, testers, and stakeholders to ensure the project meets the intended requirements.

1.1 Document Purpose

This Software Requirements Specification (SRS) document defines the functional and non-functional requirements for the JUNO 2.0, which serves as an enhanced campus management system for university students, faculty, and staff. The website aims to address common challenges such as classroom availability, mess balance tracking, professor consultation scheduling, and lost-and-found management. This document provides a detailed outline of the system’s objectives, scope, features, and performance expectations to guide the development process.

This SRS covers the first release (Version 1.0) of the website, detailing all core functionalities planned for initial deployment. It specifies requirements for the frontend, backend, database interactions, and security measures, ensuring alignment with university policies. Future versions or additional modules may be addressed in separate documents.

1.2 Product Scope

The JUNO 2.0 website is a campus management system designed to streamline daily university operations for students, faculty, and staff. It provides a centralized platform to access essential services such as real-time classroom availability, mess balance tracking, professor consultation scheduling, and lost-and-found management. The website replaces traditional manual processes with a digital, user-friendly, and efficient solution, reducing administrative burdens and improving campus life.

By offering real-time updates, seamless booking, and secure access, the website enhances convenience, transparency, and efficiency for all users. Students can quickly find available classrooms, check meal balances, and schedule professor meetings, while faculty and staff can better manage their schedules. The website's secure authentication system ensures that only authorized university members can access relevant features, making it a reliable and secure tool for campus management.

1.3 Intended Audience and Document Overview

This **SRS** document is intended for the following readers:

- **Client (University Administration) –** To understand the website's purpose, scope, and expected functionalities, and ensure alignment with university policies.
- **Professor (Project Guide) –** To review technical feasibility and provide feedback.
- **Developers –** To use detailed functional requirements, software interfaces, and system design as a reference for implementation.
- **Testers –** To verify if the system meets specified requirements and functions as expected.
- **Future Maintainers:** To ensure continuity and scalability in future updates.

1.3.1 Overview

- **Introduction –** Covers the document's purpose, product scope, intended audience, definitions, conventions, and references.
- **Overall Description** – Provides an overview of the product, its functionality, constraints, assumptions, and dependencies.
- **Specific Requirements –** Details external interface requirements, functional requirements, and use case models.
- **Other Non-Functional Requirements –** Specifies performance requirements, security and safety measures, and software quality attributes.
- **Other Requirements –** Lists additional project considerations.
- **Appendices** – Contains supplementary materials, including wireframes of UI design and formulae for non-functional requirements.

This document ensures that all stakeholders have a comprehensive understanding of the system's goals, design, and implementation details.

1.3.2 Document Structure and Reading Guide

1. **Introduction & Scope (Sections 1) –** Recommended for all readers to get an overview of the project.
2. **Functional & External Interface Requirements (Sections 3 & 4) –** Essential for developers and testers.
3. **Assumptions, Constraints, and Integration Details (Sections 2 & 4) –** Useful for the client and professor to ensure project feasibility.

1.4 Definitions, Acronyms and Abbreviations

1. **API** – Application Programming Interface
2. **GUI** – Graphical User Interface
3. **MU** – Mahindra University
4. **SRS** – Software Requirements Specification
5. **TLS 1.2** – Transport Layer Security version 1.2
6. **UI/UX** – User Interface / User Experience

1.4.1 Technical Definitions (Used in Non-Functional Requirements):

1. **AES-256** (Advanced Encryption Standard - 256-bit) – A secure encryption algorithm used to protect sensitive data by encoding it with a 256-bit key.
2. **Authentication Mechanism –** The process of verifying the identity of users, ensuring secure access control.
3. **Encryption** – The method of encoding data to protect sensitive information from unauthorized access.
4. **Latency** – The delay between a user action and the system's response, measured in milliseconds (ms).
5. **Load Balancing** – A technique used to distribute network traffic efficiently to prevent server overload.
6. **Response Time –** The time taken by the system to respond to a user request.
7. **Scalability –** The ability of the system to handle increased workload by upgrading resources or optimizing performance.
8. **Throughput –** The number of successful transactions processed by the system per second.
9. **TLS 1.2** (Transport Layer Security 1.2) – A cryptographic protocol that ensures secure communication over networks, protecting data from unauthorized access.

This glossary ensures a clear understanding of key terms used throughout the document, particularly in technical sections.

1.5 Document Conventions

This document follows the IEEE formatting requirements. Arial font size 11 throughout the document for text, Arial Bold font size 14 for headings, Arial Bold size 12 for Sub-headings and Headings under Sub-headings are underlined. Italics for comments. Document text is 1.15 spaced and maintains the 1" margins.

2 Overall Description

2.1 Product Overview

The JUNO 2.0 is a follow-on product that builds on the earlier version of the college utility website. It is designed to enhance campus management by providing a centralized, real-time, and user-friendly solution for students, faculty, and staff. This system is not a direct replacement for any existing website but rather an enhancement to manual processes currently used for classroom availability, mess balance tracking, professor scheduling, and lost-and-found management. By digitizing these tasks, the website aims to reduce administrative burdens, increase transparency, and offer real-time updates to its users.

2.2 Product Functionality

The website will function as a subsystem within the university's digital ecosystem, interacting with dummy database and authentication services. It will interface with:

- **Classroom Management System** – to fetch classroom availability data.
- **Mess Management System –** to retrieve and update mess balances.
- **Scheduling Module** – for faculty to set available consultation hours.
- **Lost-and-Found System** – to allow users to report and track lost items.
- **Real-Time Updates –** Push notifications and live data feeds for all core functionalities.
- **Web View Rendering** – Ensure a seamless experience across web platforms.

2.3 Design and Implementation Constraints

The development of the JUNO 2.0 Website is subject to several constraints that affect the design and implementation process. These constraints ensure the project remains within technical, security, and organizational requirements while maintaining a user-friendly experience.

Hardware Limitations

- The website must operate efficiently on **mid-range smartphones** with a minimum of **2GB RAM and 1.5 GHz processor speed**.
- The website should be optimized for **low battery consumption** to enhance usability.
- The UI/UX should be responsive across various screen sizes (mobile and tablet).

Software Dependencies & Interfaces

- The website must integrate seamlessly with the **existing university portal**.

- Interfaces will be developed for **authentication class scheduling, and mess balance tracking, professor meeting and lost and found**.
- The system will follow a **client-server architecture**, with the backend exposing **RESTful APIs** for modular development.

Required Technologies and Tools

- **Frontend:** React with TypeScript.
- **Backend:** Express.js.
- **Database:** PostgreSQL (relational data).
- **Development Tools:** Visual Studio Code, Express Server, and JWT based Authentication, Multer.

Security Considerations

- **End-to-End Encryption:** All sensitive data (user credentials, mess balances, scheduling data) must be encrypted using **AES-256** encryption.
- **Role-Based Access Control (RBAC):** Faculty and students will have **different permission levels** to prevent unauthorized access.
- **Session Management:** Secure session handling techniques will be implemented to prevent **session hijacking**.

2.4 Assumptions and Dependencies

The project's success depends on several key assumptions and external factors:

- **Authentication System:**
    - It is assumed that a robust, secure authentication system exists, that allows only authorized university members to access the website.
- **User Participation:**
    - Faculty, students, and staff are assumed to actively participate in the testing phase and provide valuable feedback to guide refinements.
- **Data Availability:**
    - While actual university data is not provided, the system is expected to interface with such data post-deployment; initial development will rely on dummy data.
- **Policy and Compliance:**
    - The website's features must strictly adhere to university administrative policies and guidelines.
- **External Dependencies:**
    - The product depends on external university backend systems for real-time data and authentication.

- Reliance on third-party services (e.g., for notifications or data synchronization) will be managed to ensure consistent performance.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The JUNO 2.0 Website will provide an intuitive and user-friendly interface that allows university students, faculty, and staff to seamlessly access and manage campus-related activities.

Main User Interface Features*:

1. **Dashboard** – Provides quick access to various sections of the website, including:
   a. **Classroom Allotment (Search for and book available rooms)**
   b. **Mess Balance** (View meal credits)
   c. **Contact Faculty** (Check professor availability and book slots)
   d. **Lost & Found** (Report or retrieve lost items)

2. **Interactive UI Elements**
   a. **Touch-based navigation** with **buttons, dropdown menus, and list views**
   b. **Search functionality** for quick access to classrooms, professors, and items
   c. **Real-time notifications** for important updates

User Interaction

1. **Touchscreen Controls –** Users will interact through taps, swipes, and long presses.

2. **Real-time Status Updates –** Users will receive notifications on room availability, mess balances, and lost-and-found reports.

**Refer Appendix A for wireframes of the features.*

3.1.2 Hardware Interfaces

1. **Smartphones & Tablets (Android & iOS)** – Primary user interface.
2. **University Servers –** Hosts authentication, room availability, mess balance, and other data.
3. **Campus Wi-Fi & Mobile Networks** – Enables real-time data exchange.
4. **RFID/Smart Card Readers –** Used for mess balance tracking.

3.1.3 Software Interfaces

- **University Server & Database –** Handles user authentication, classroom availability, mess balance, and lost-and-found data.
- **API Services** – Facilitates communication between the website and the backend for real-time updates.
- **Push Notification System** – Sends alerts for room bookings, mess balance updates, and lost-and-found reports.
- **RFID Middleware –** Processes mess balance transactions when integrated with smart card readers.

3.2 Functional Requirements

3.2.1 The system shall fetch classroom availability data from the Classroom Management System and allow students to book empty rooms.

3.2.2 The system shall retrieve and update mess balances by interacting with the Mess Management System and display balances to students.

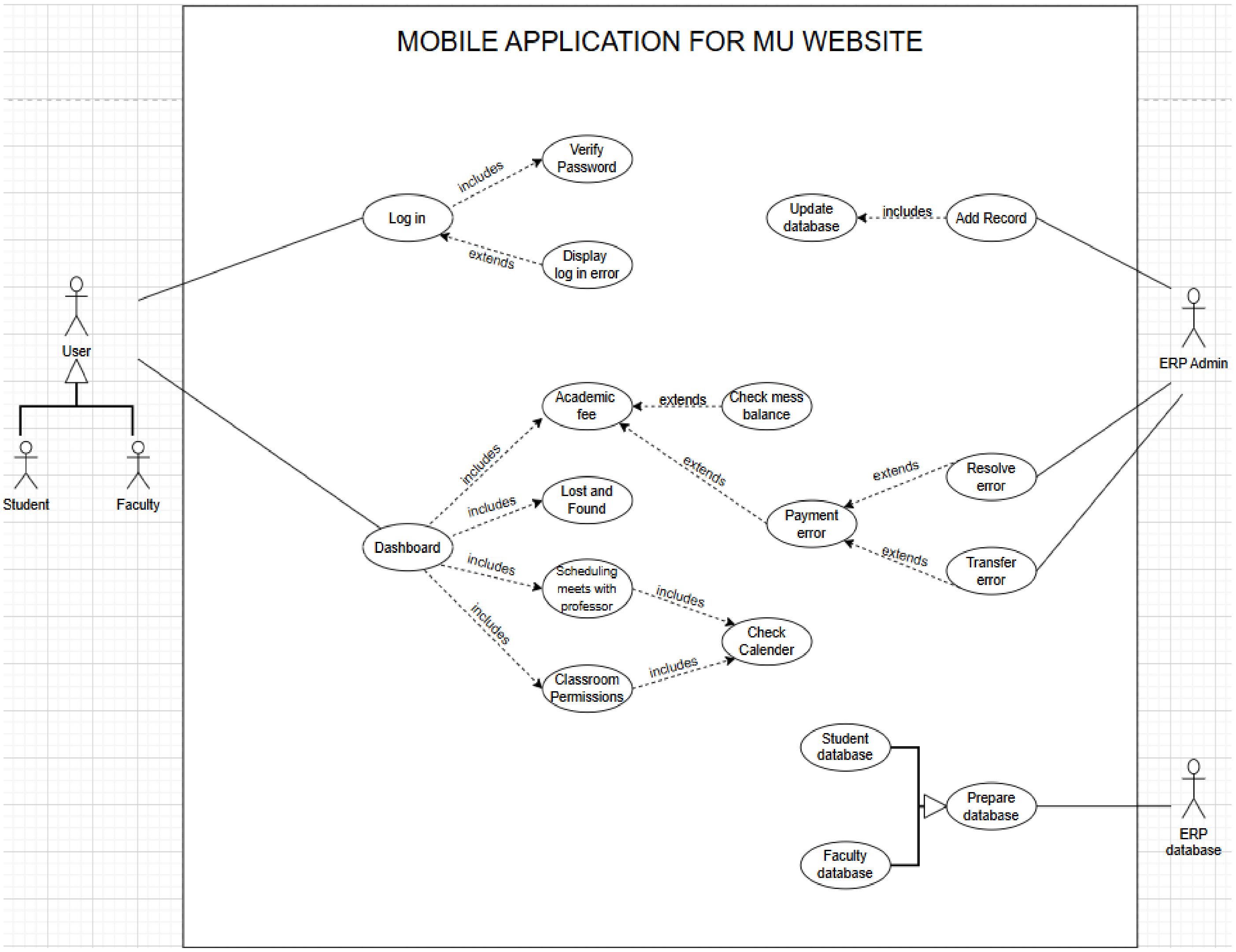
3.2.3 The system shall allow faculty to set and update their available consultation hours through the Scheduling Module, enabling students to book appointments.

3.2.4 The system shall enable users to report and track lost items via the Lost-and-Found System for easy recovery.

3.2.5 The system shall send real-time push notifications for classroom bookings, scheduled appointments, and lost-and-found reports.

3.2.6 The system shall provide a Web View Rendering feature to ensure seamless usability across web platforms.

3.3 Use Case Model



The use cases below are the 4 main high priority features in the website.

3.3.1 Use Case #1 - Check mess balance: U1

Author – Aashi, Harshitha, Madhulika

Purpose - To allow day scholars to track and manage their mess balance in the wallet system, ensuring that meal deductions are recorded in real time.

Requirements Traceability –

- The system must deduct a fixed amount when an RFID card is scanned.
- The transaction should display a timestamp and updated balance.
- Day-wise transactions should be removed from the database monthly.
- The semester-wise balance should be displayed separately under "Student Misc."

Priority - High.

Preconditions -

- The student must be a day scholar.
- The RFID card should be registered in the system.

Post conditions -

- The student's wallet balance is updated after each meal transaction.
- Transactions older than a month are deleted.
- The semester-wise balance is available for review.

Actors – Day Scholar Students

Extends – Academic fee

Flow of Events

1. **Basic Flow** -
   - Student scans their RFID card at the mess.
   - The system deducts a fixed meal amount.
   - The transaction is logged with a timestamp.
   - The updated daily and overall balance is displayed.
2. **Alternative Flow** -
   - If the student has insufficient funds, they are notified with a message.
3. **Exceptions** -
   - The RFID card is not recognized.
   - System fails to deduct the appropriate amount.

3.3.2 Use Case #2 – Lost and Found: U2

Author: Yaswitha, Komal

Purpose: To help students and faculty report lost and found items and track notifications about submitted items.

Requirements Traceability:

- Users can report lost items with descriptions and optional images.

- Users can report found items with a short description and location of submission.
- A bulletin board displays lost/found notifications for 30 days.

Priority: High
Preconditions:

- User must be logged in.

Postconditions:

- Lost item reports are recorded in the system.
- Found item reports are posted with submission locations.
- Notifications older than 30 days are deleted.

Actors: Student and Faculty

Flow of Events:

1. **Basic Flow:**
   - User selects "Lost and Found" from the dashboard.
   - User chooses one of the three options: Lost, Found, or Forum.
   - User enters the required details and submits the form.
   - The system updates the bulletin board with the new entry.
2. **Alternative Flow:**
   - If no image is uploaded for a lost item, the system still records the description.
3. **Exceptions:**
   - User enters invalid data (e.g., too long description for "Found").

Includes: Bulletin Board Update

3.3.3 Use Case #3 – Scheduling meets with professor: U3

Author: Sanjana, Roshan

Purpose: To allow students to schedule meetings with professors based on their availability.

Requirements Traceability:

- The system must display the student's registered subjects.
- It should show available professors and teaching assistants.
- The professor/TA should approve, decline, or reschedule the meeting.
- Approved meetings should be added to the professor's calendar.

Priority: High

Preconditions:

- The student must be logged in.
- The professor's free slots should be available in the system.

Postconditions:

- A meeting request is sent to the professor/TA.
- The student is notified of the approval, rejection, or rescheduling.

Actors: Students, Professors and Teaching Assistants

Extends: Check Calendar

Flow of Events:

1. **Basic Flow:**
   - Student selects "Contact Faculty" from the dashboard.
   - The system displays the subjects registered by the student.
   - The student selects a subject and chooses a professor/TA.
   - The professor's free slots are displayed.
   - The student selects a time slot and submits a meeting request.
   - The professor approves or rejects the meeting.
2. **Alternative Flow:**
   - If the professor is unavailable, the student can select another slot.
3. **Exceptions:**
   - The professor is unavailable for the entire selected week.

Includes: Calendar Management

3.3.4 Use Case #4 Classroom Allotment: U4

Author: Aashi, Harshitha, Madhulika

Purpose: To allow students to request permission for classroom use.

Requirements Traceability:

- Users should be able to request classroom access.
- The request should go to the concerned authority for approval.

Priority: Medium

Preconditions:

- User must be logged in.

Postconditions:

- The request is either approved or denied.

Actors: Student, Faculty and Admin

Flow of Events:

1. **Basic Flow:**
   - User selects "Classroom Allotment" from the dashboard.
   - User fills out the request form.
   - The request is sent to the admin for approval.
   - The admin approves or denies the request.
2. **Alternative Flow:**
   - If the classroom is unavailable, the system suggests alternate rooms.
3. **Exceptions:**
   - The admin does not respond in time.

4 Other Non-functional Requirements

4.1 Performance Requirements

1. **Response Time**: The website should respond to user actions within **2 seconds** for all core features under normal load conditions (e.g., booking a classroom, checking mess balance, scheduling professor meetings, and lost-and-found queries).

2. **Concurrent Users**: The system should support at least **500 concurrent users*** without noticeable lag or downtime.

3. **Data Processing Time**: Backend processing for fetching and updating real-time data (e.g., classroom availability, mess balance) should not exceed **1 second*** under normal condition.

4**. Classroom Booking Updates**: The website must update the availability of classrooms in **real time** (within **3 seconds***) when a booking is made or a room becomes available.

5**. Mess Balance Updates**: Users should see their latest meal balance within **2 seconds** of requesting an update.

6. **Professor Availability Updates**: Updates made by professors should reflect in the system within **5 seconds** after submission.

7. **Lost-and-Found Query Time**: Searching for lost items in the system should return results in under **2 seconds** for a dataset of at least **10,000 records**.

8. **Network Conditions**: The website should remain functional on a **3G connection** with minor delays (up to **5 seconds**) in fetching updates but should not crash or timeout.

9. **System Downtime**: The website should have a **99.9% uptime**, meaning total downtime should not exceed **8.76 hours per year***.

10. **Automatic Recovery**: In case of a system failure, the website should recover within **10 seconds*** and restore the last known state.

**Refer to Appendix B for details on the above-mentioned values.*

4.2 Safety and Security Requirements

1. **Role-Based Access Control (RBAC)**: Different user roles (students, faculty, staff) will have access only to relevant features and data.

2. **Data Encryption**:
- All sensitive user data (login credentials, mess balance, bookings) must be encrypted using **AES-256** before storage.
- Data transmission must be secured using **TLS 1.2** or higher to prevent interception.

3**. Data Retention Policy**: Personal data should be stored only for as long as necessary and automatically deleted after a set period (e.g., one semester for booking history).

4**. Secure API Communication**: The website should only communicate with university servers via **HTTPS**, preventing man-in-the-middle (MITM) attacks.

5**. Automatic Logout**: Users should be logged out after 15 minutes of inactivity to prevent unauthorized access.

6**. Classroom Booking Protection**: A user should not be able to book multiple classrooms at the same time or book rooms beyond allowed hours.

7**. Mess Balance Restrictions**: Users cannot transfer or modify their mess balance directly to prevent fraud.

8**. Lost-and-Found Moderation**: All submitted lost-and-found reports must be reviewed before becoming public to prevent misuse.

9**. University IT Policies**: The system must comply with MU's IT and data protection policies.

4.3 Software Quality Attributes

4.3.1 Reliability

4.2.1.1 **Uptime Guarantee:** The system must achieve 99.9% uptime, ensuring it is available for students and faculty throughout the academic year.

4.2.1.2 **Fault Tolerance:** In case of a failure (e.g., server crash), the system must automatically recover and resume operations within 10 seconds.

4.2.1.3 **Data Integrity:** All user transactions (classroom bookings, mess balance updates) must be fully committed to the database before confirming actions to prevent data loss or inconsistencies.

4.2.1.4 **Load Handling:** The website should maintain performance while handling up to 1,000 requests per minute, ensuring stability during peak usage times.

4.3.2 Maintainability

4.2.1.5 **Modular Architecture**: The codebase will be structured in a modular format, allowing developers to update or add features without affecting other components.

4.2.1.6 **Code Documentation**: Every function and module must have inline documentation, and a developer guide will be maintained for easy onboarding of new contributors.

4.2.1.7 **Version Control:** All development will be tracked using Git, ensuring seamless rollback in case of issues and easier collaboration.

4.2.1.8 **Configurable Settings:** Key settings (such as session timeout duration, user role permissions) should be stored in a configuration file, allowing changes without modifying the core code.

4.3.3 Usability

4.2.1.9 I**ntuitive UI Design:** The website must follow a clean and simple UI/UX design, ensuring that users can navigate features with minimal effort.

4.2.1.10 **Mobile Responsiveness:** The interface must be fully roid and iOS devices, adjusting dynamically for different screen sizes.

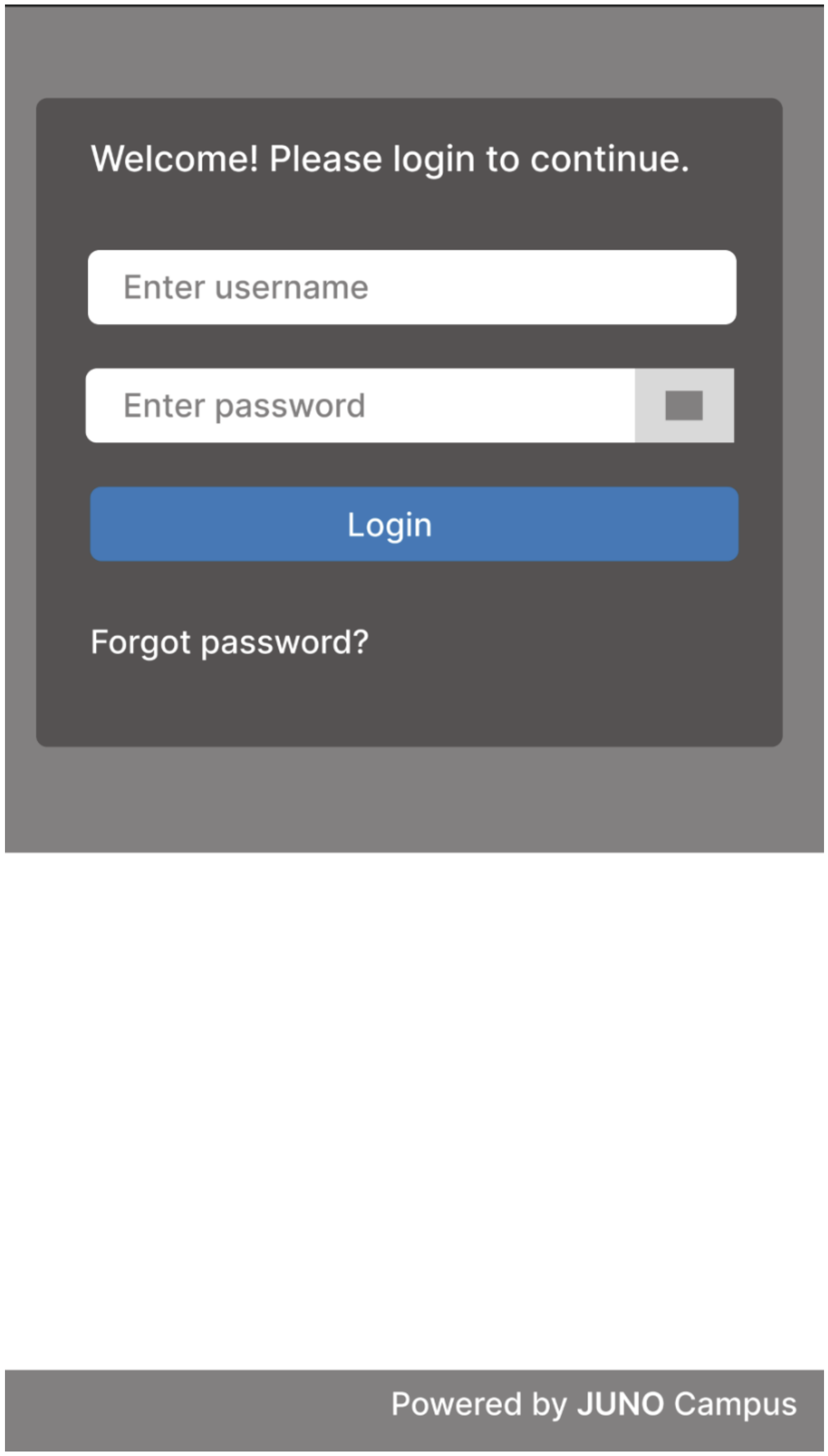
4.2.1.11 **Error Handling & Feedback:** All user actions should provide instant feedback (e.g., error messages, confirmations), and form inputs must validate incorrect entries before submission.

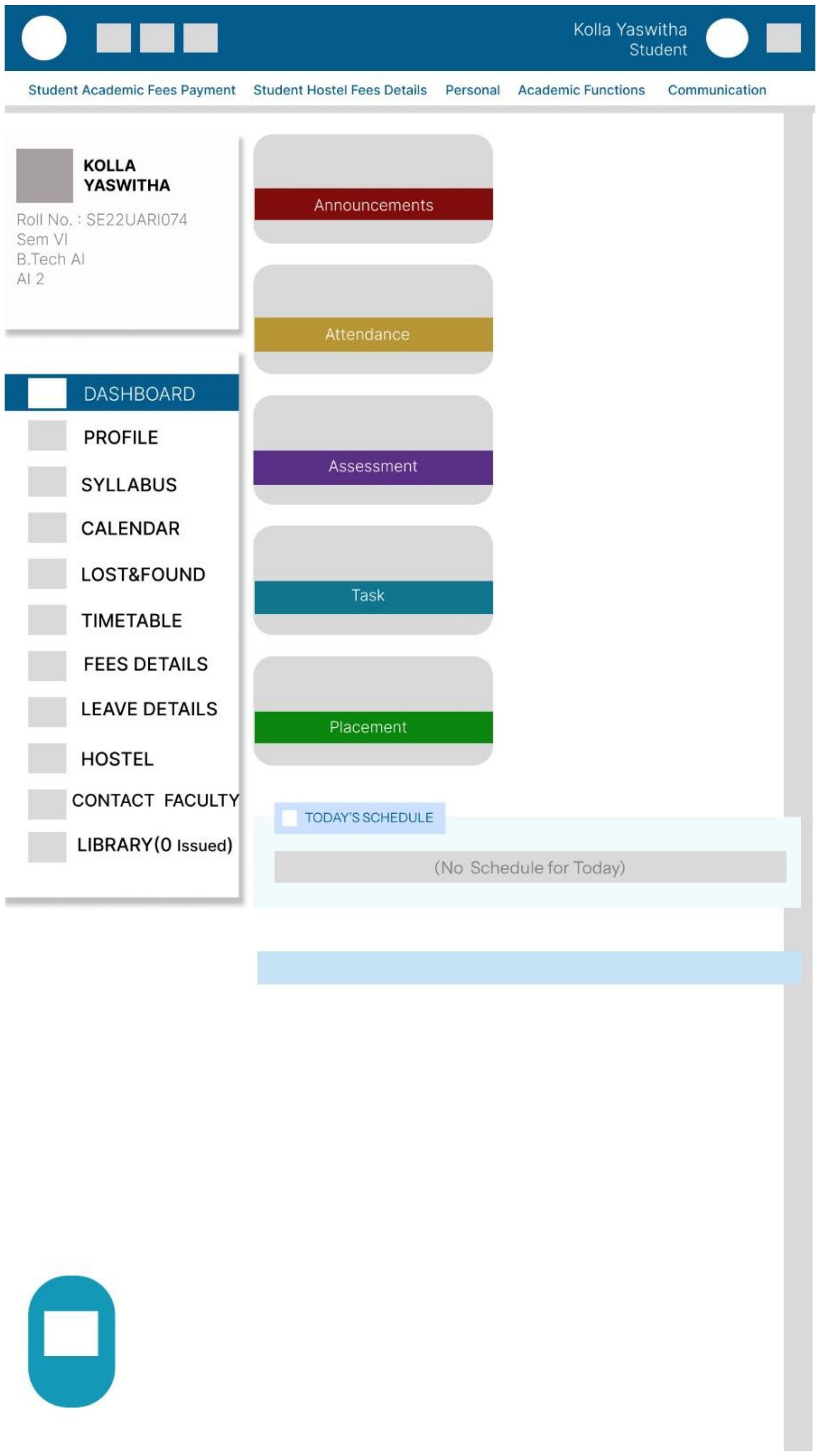
4.3.4 Interoperability

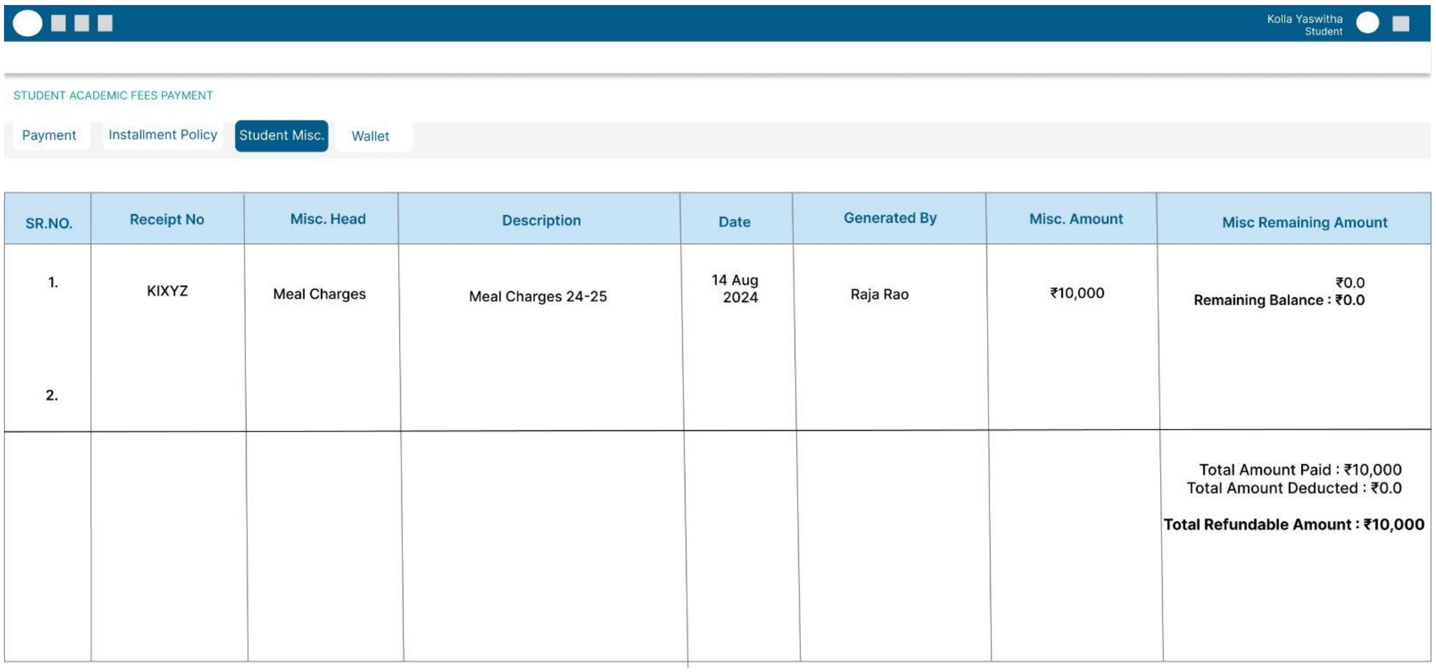
4.2.1.12 **Compatibility with University Systems:** The website must be able to integrate with the university's existing authentication system and student database to avoid duplicate logins.

4.2.1.13 **API-Based Communication:** All external system connections (e.g., timetable data, mess balances) should use REST APIs, ensuring easy updates and scalability.

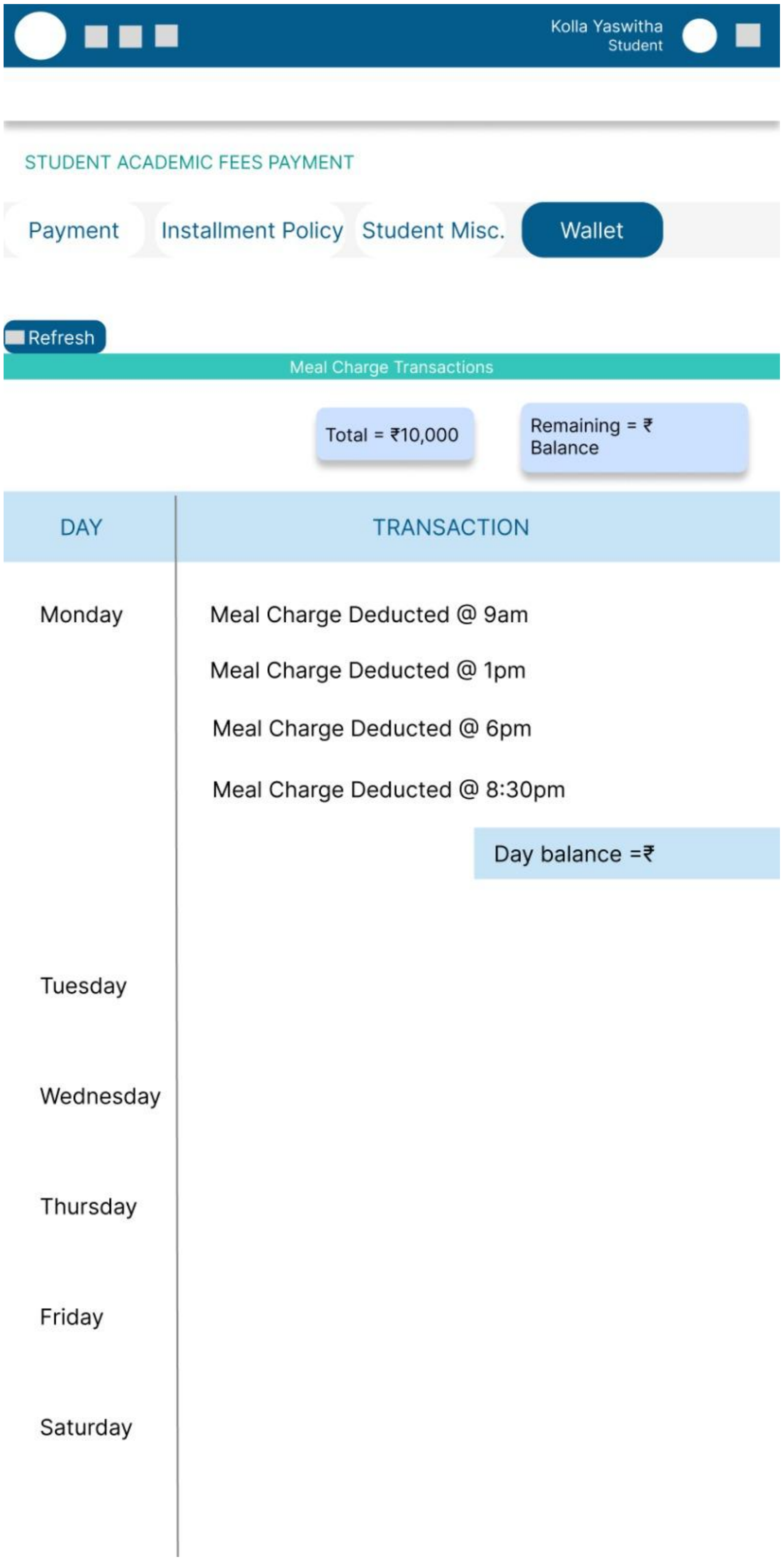
Appendix A – Data Dictionary


Welcome! Please login to continue.

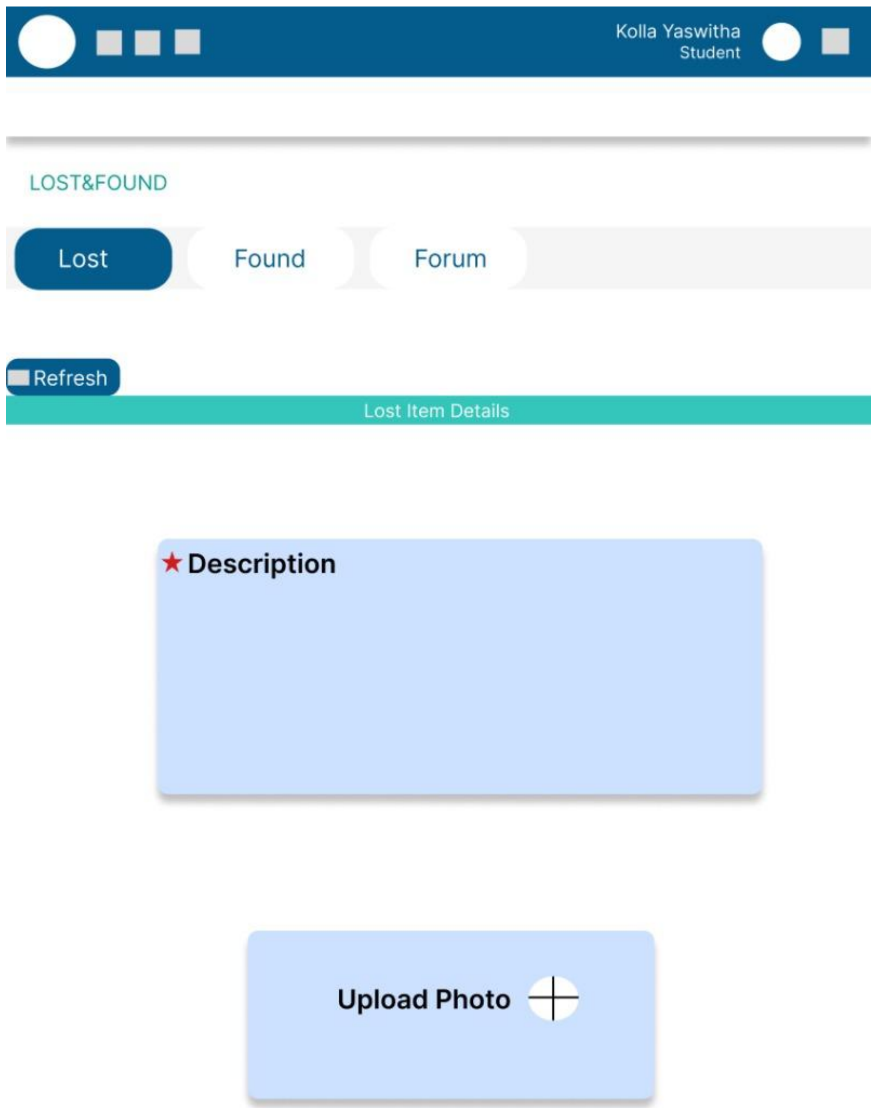

Kolla Yaswitha
Student
Student Academic Fees Payment
Student Hostel Fees Details
Personal
Academic Functions
Communication
KOLLA YASWITHA
Roll No. : SE22UARI074
Sem VI
B.Tech AI
AI 2
DASHBOARD
PROFILE
SYLLABUS
CALENDAR
LOST&FOUND
TIMETABLE
FEES DETAILS
LEAVE DETAILS
HOSTEL
CONTACT FACULTY
LIBRARY(0 Issued)
Announcements
Attendance
Assessment
Task
Placement
TODAY'S SCHEDULE
(No Schedule for Today)

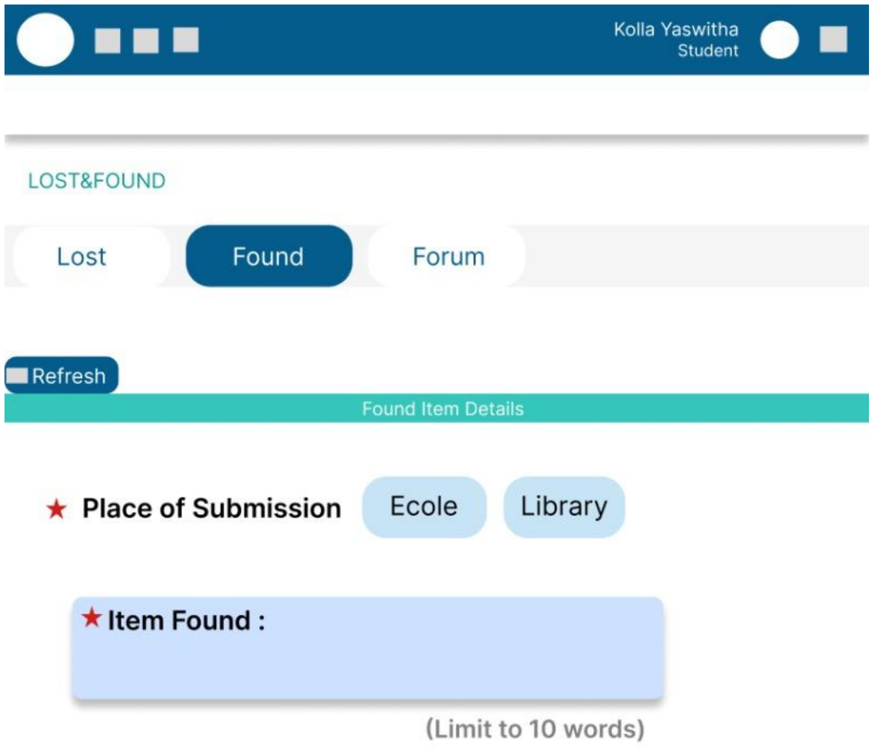


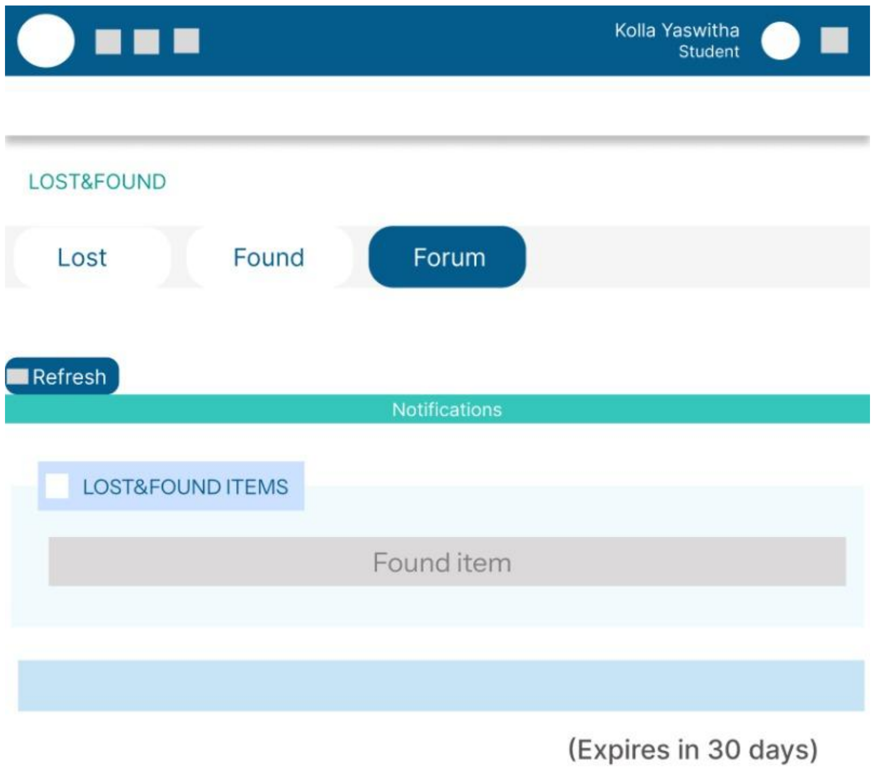
SR.NO.	Receipt No	Misc. Head	Description	Date	Generated By	Misc. Amount	Misc Remaining Amount
1.	KIXYZ	Meal Charges	Meal Charges 24-25	14 Aug 2024	Raja Rao	₹10,000	₹0.0 **Remaining Balance : ₹0.0**
2.							
							Total Amount Paid : ₹10,000 Total Amount Deducted : ₹0.0 **Total Refundable Amount : ₹10,000**

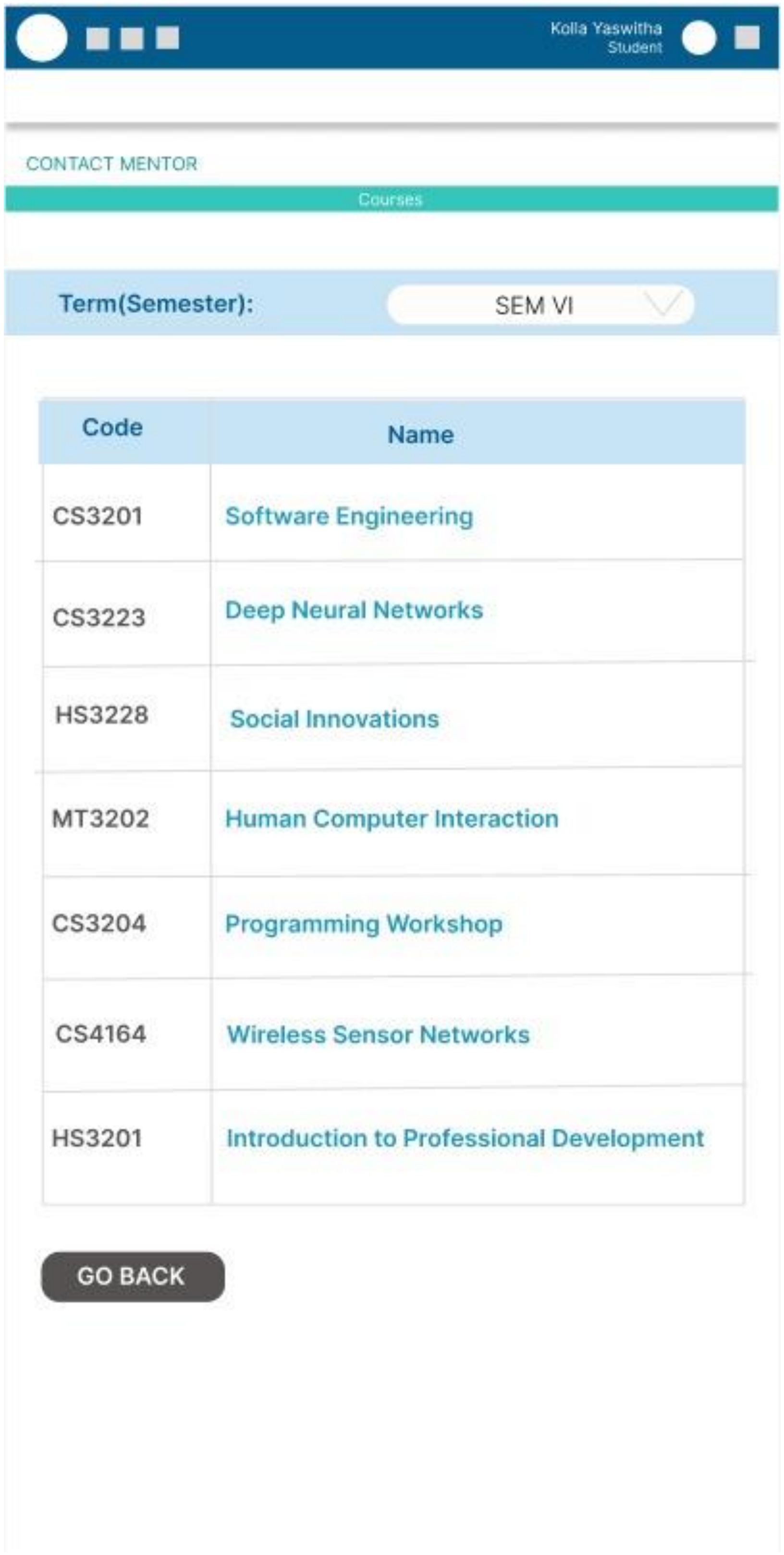


DAY	TRANSACTION
Monday	Meal Charge Deducted @ 9am Meal Charge Deducted @ 1pm Meal Charge Deducted @ 6pm Meal Charge Deducted @ 8:30pm Day balance =₹
Tuesday	
Wednesday	
Thursday	
Friday	
Saturday	

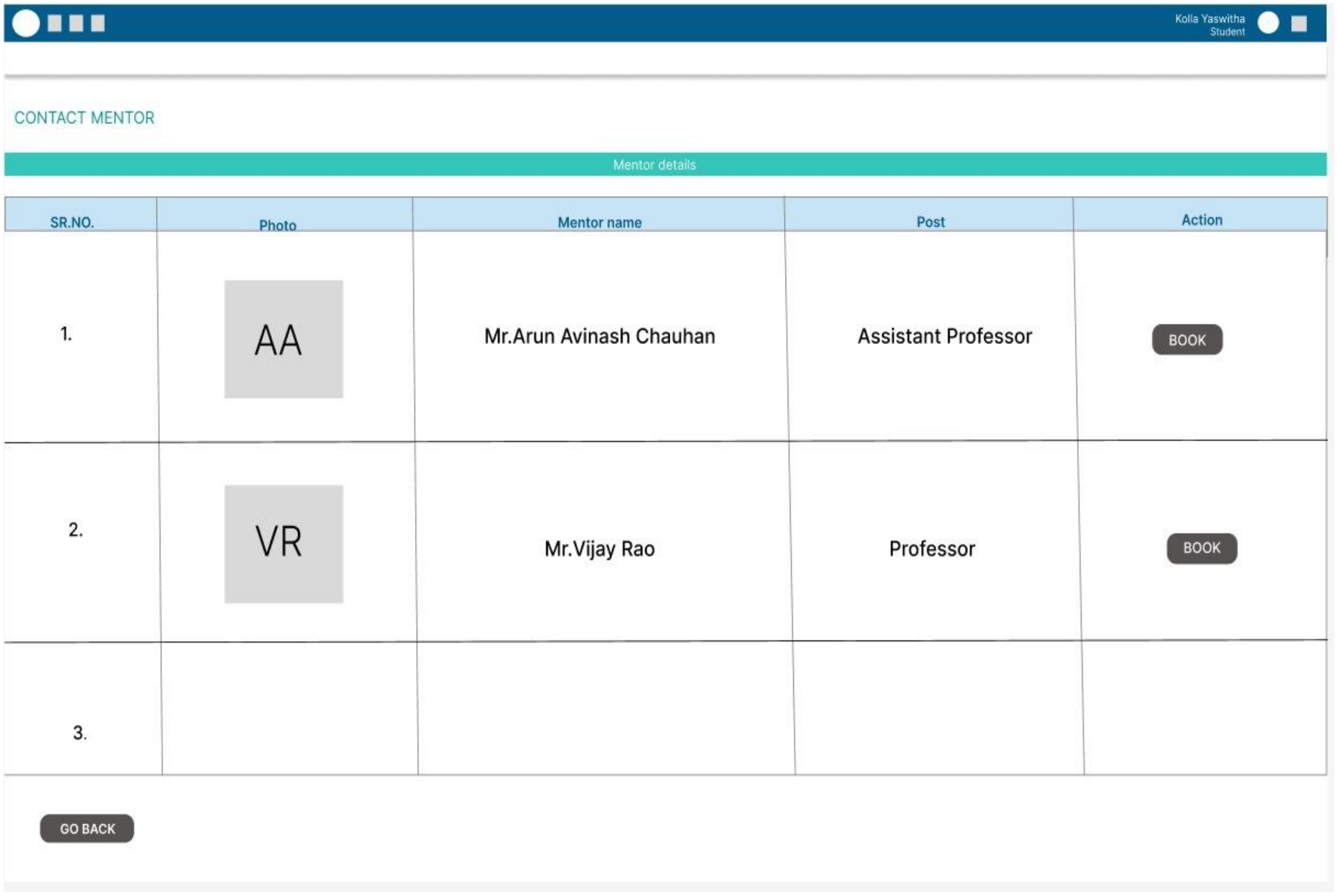

Kolla Yaswitha
Student
LOST&FOUND
Lost
Found
Forum
Refresh
Lost Item Details
★ Description
Upload Photo


Kolla Yaswitha
Student
LOST&FOUND
Lost
Found
Forum
Refresh
Found Item Details
★ Place of Submission
Ecole
Library
★ Item Found :
(Limit to 10 words)


Kolla Yaswitha
Student
LOST&FOUND
Lost
Found
Forum
Refresh
Notifications
LOST&FOUND ITEMS
Found item
(Expires in 30 days)



Code	Name
CS3201	Software Engineering
CS3223	Deep Neural Networks
HS3228	Social Innovations
MT3202	Human Computer Interaction
CS3204	Programming Workshop
CS4164	Wireless Sensor Networks
HS3201	Introduction to Professional Development

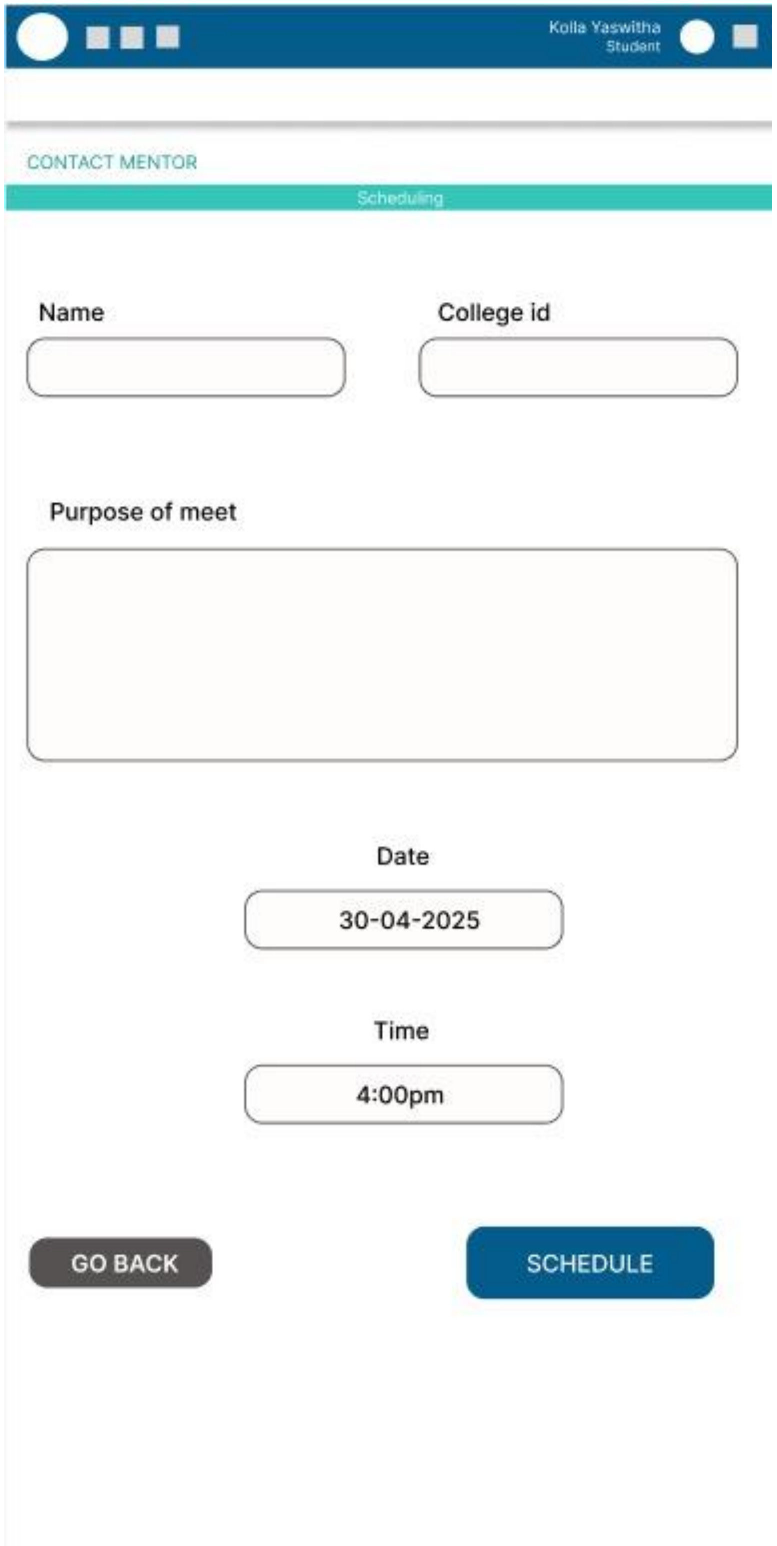

Kolla Yaswitha
Student

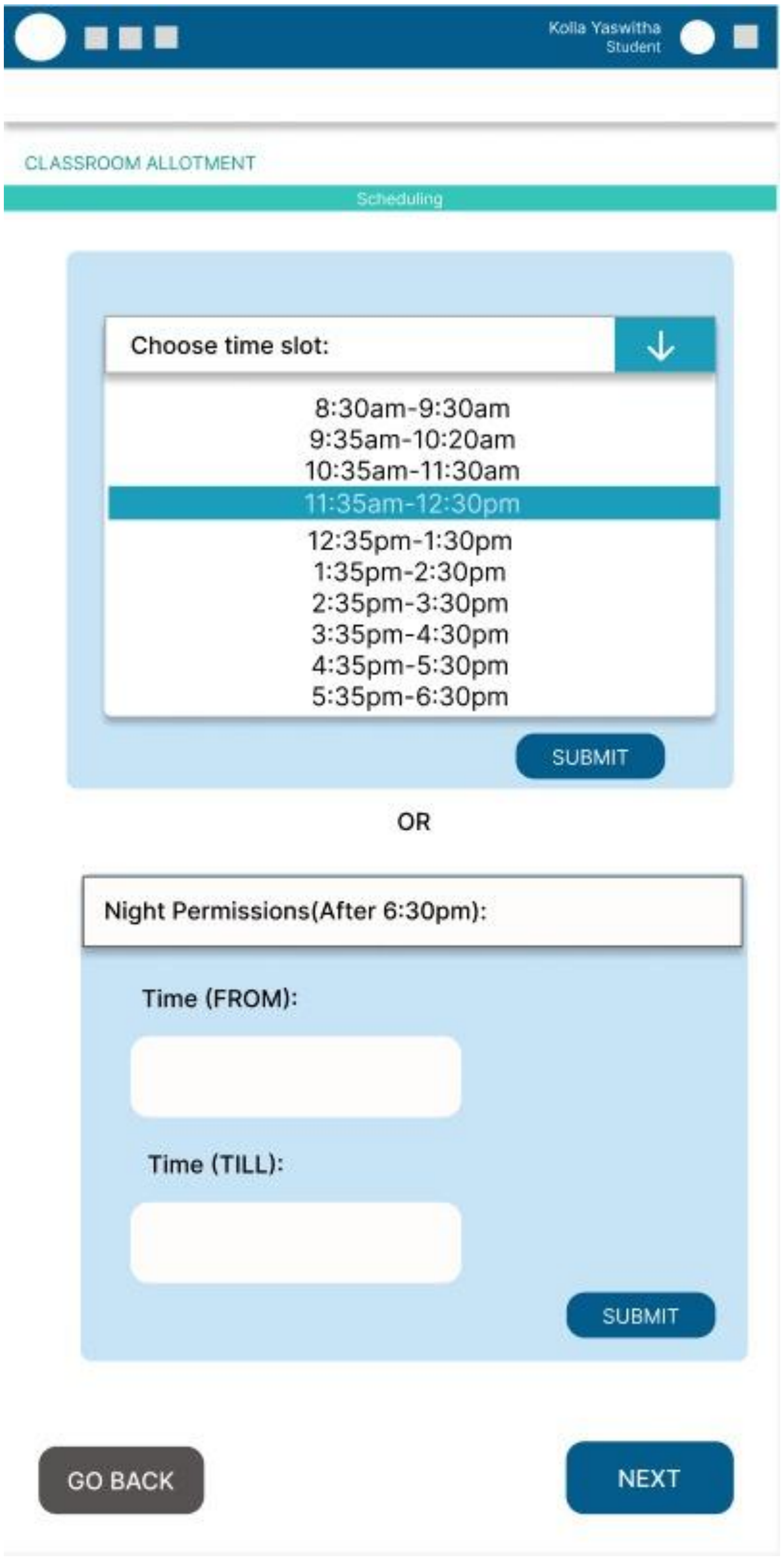
CONTACT MENTOR

Mentor details

SR.NO.	Photo	Mentor name	Post	Action
1.	AA	Mr.Arun Avinash Chauhan	Assistant Professor	BOOK
2.	VR	Mr.Vijay Rao	Professor	BOOK
3.				

GO BACK


Kolla Yaswitha
Student
CONTACT MENTOR
Scheduling
Name
College id
Purpose of meet
Date
30-04-2025
Time
4:00pm
GO BACK
SCHEDULE



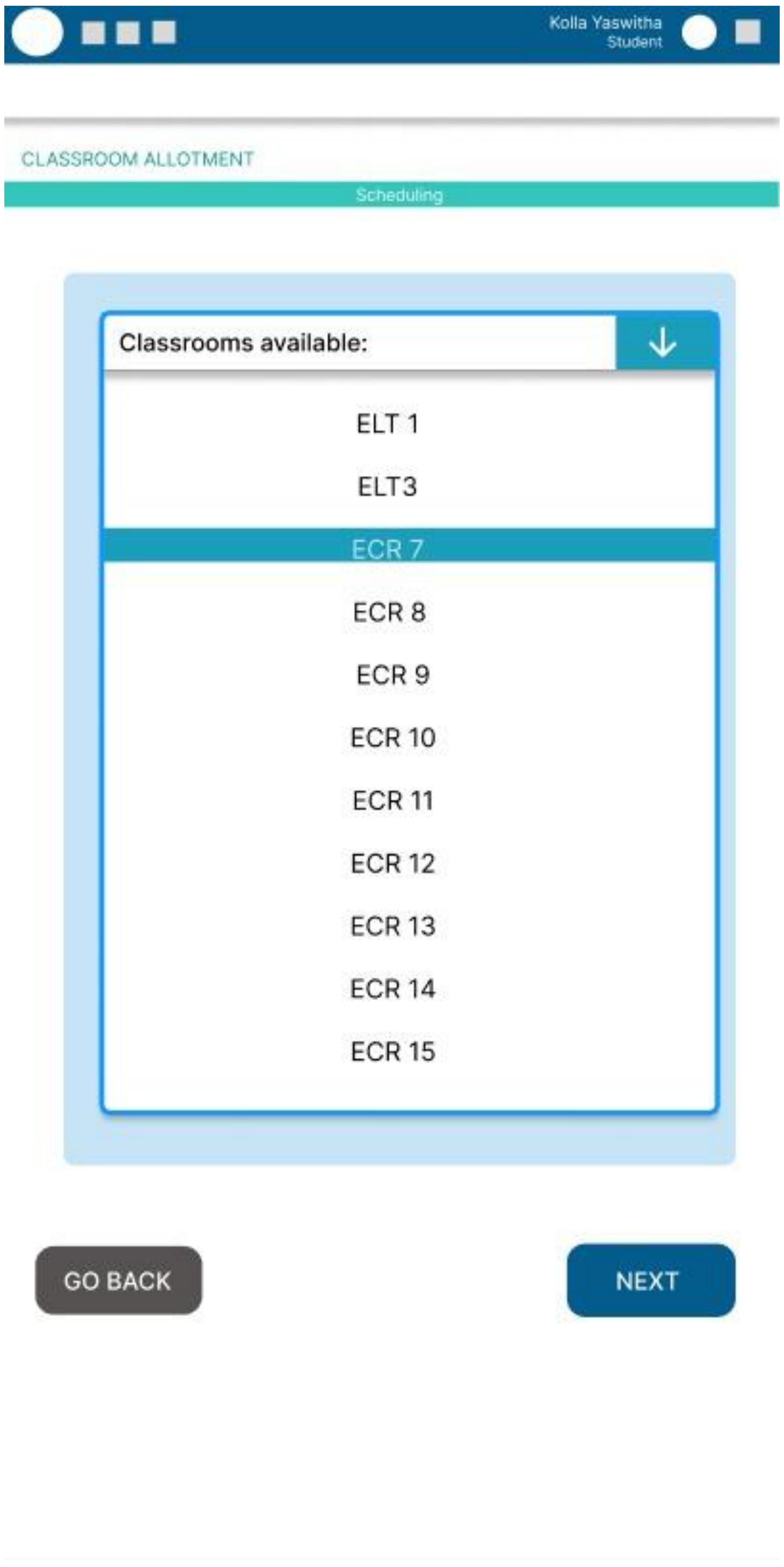
Choose time slot:

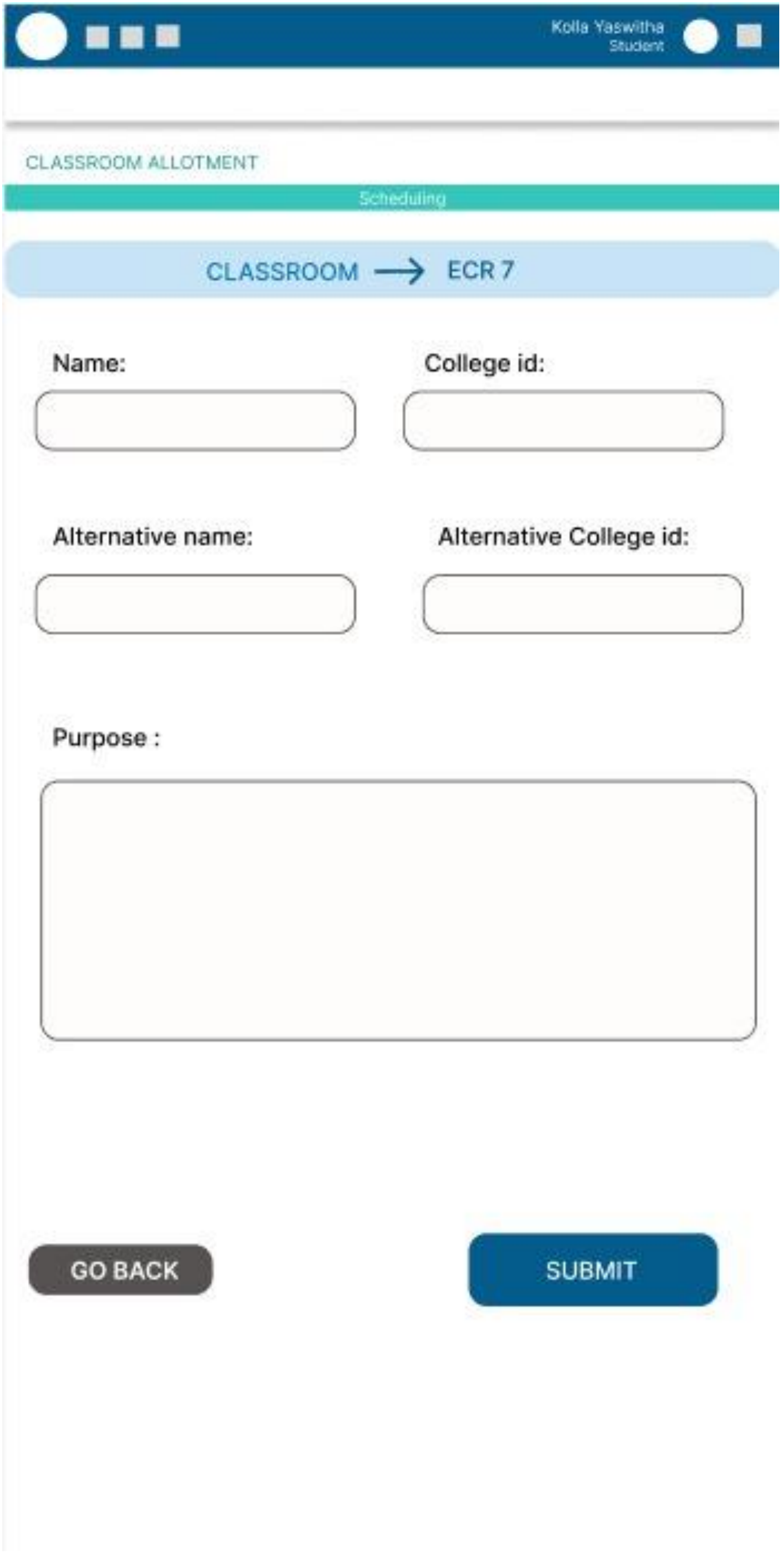
8:30am-9:30am
9:35am-10:20am
10:35am-11:30am
11:35am-12:30pm
12:35pm-1:30pm
1:35pm-2:30pm
2:35pm-3:30pm
3:35pm-4:30pm
4:35pm-5:30pm
5:35pm-6:30pm

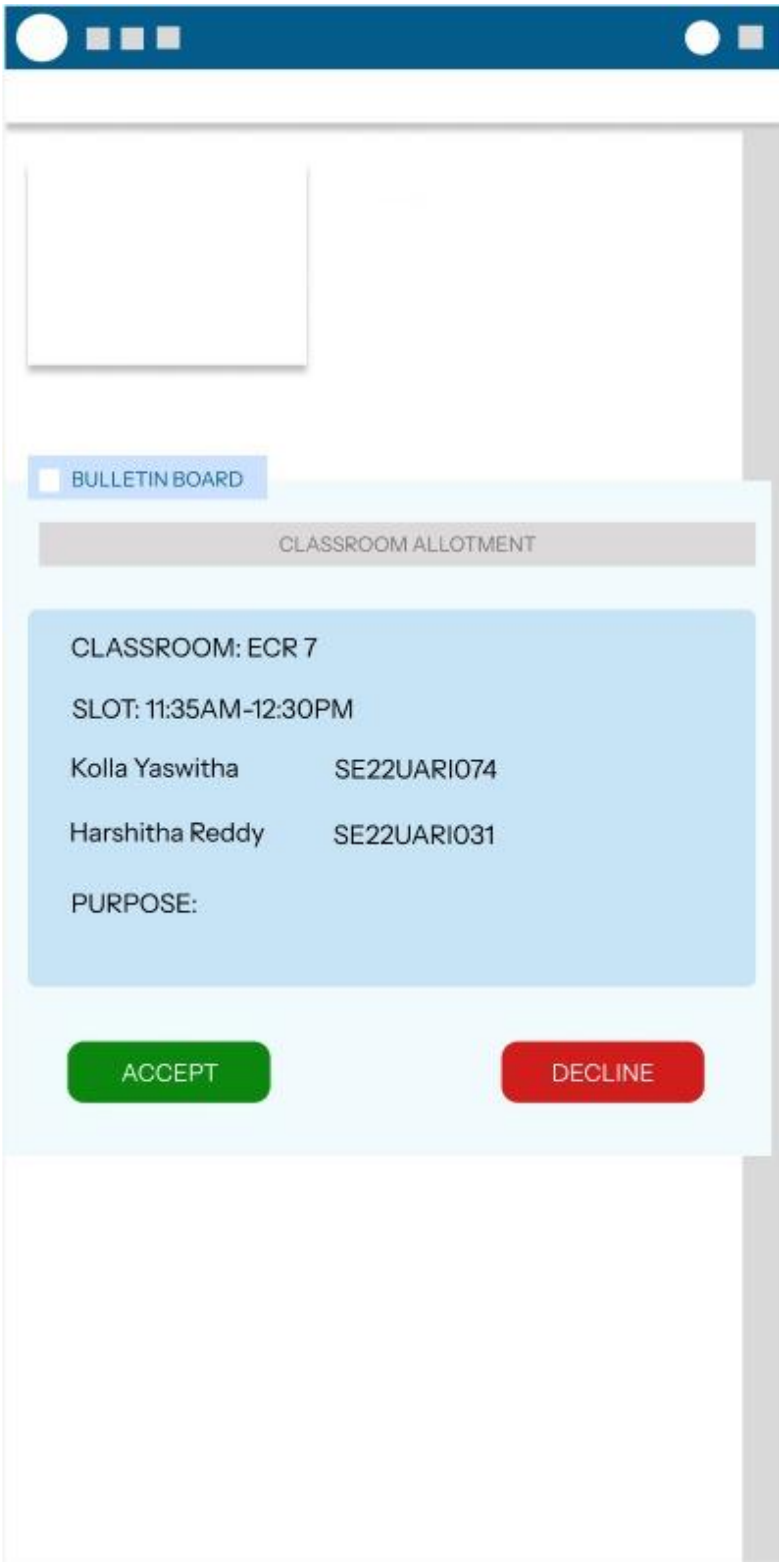
Night Permissions(After 6:30pm):

Time (FROM):

Time (TILL):


Kolla Yaswitha
Student
CLASSROOM ALLOTMENT
Scheduling
Classrooms available:
ELT 1
ELT3
ECR 7
ECR 8
ECR 9
ECR 10
ECR 11
ECR 12
ECR 13
ECR 14
ECR 15
GO BACK
NEXT


Kolla Yaswitha
Student
CLASSROOM ALLOTMENT
Scheduling
CLASSROOM → ECR 7
Name:
College id:
Alternative name:
Alternative College id:
Purpose :
GO BACK
SUBMIT



CLASSROOM: ECR 7

SLOT: 11:35AM-12:30PM

Kolla Yaswitha	SE22UARI074
Harshitha Reddy	SE22UARI031

PURPOSE:

Appendix B - Group Log

The performance benchmarks are based on industry standards, expected university usage, and reasonable assumptions.

1. **System Uptime:** 99.9% uptime allows for 8.76 hours of downtime per year, calculated using the formula:
   Allowed Downtime = (1−Uptime Percentage) ×Total Hours in a Year

2. **Concurrent Users:** Estimated at 500 users, assuming 10% of 5,000 students are active simultaneously.

3. **Peak Load**: The system should handle 1,000 requests per minute, assuming each active user performs two actions per minute.

4. **Booking & Updates:** Classroom availability and mess balance updates should process within 3 seconds to ensure real-time accuracy.

5. **A 3-second window** accounts for:
   i. Querying the database
   ii. Checking room availability
   iii. Saving the booking

   Real-time booking systems (like hotel reservations) aim for **sub-3-second responses.**

6. **System Recovery:** The website should auto-recover from failures within 10 seconds using fault-tolerant mechanisms.