Software Design Specifications

for

Mobile Application for MU Website

Version 1.0

Prepared by: 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
G 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

Document Information

Title: Mobile Application for MU Website	
	Document Version No:1.0
	Document Version Date:9th April, 2025
Prepared By: UniDevs	Preparation Date:9th April, 2025

Version History

Ver. No.	Ver. Date	Revised By	Description	Filename
1.0	9-04-2025	UniDevs	Initial Draft	SDD.doc

Table of Contents

1 Introduction

This document outlines the Software Design Specification for the Mobile Application for the MU Website. The Mobile Application for the MU Website 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 app enhances accessibility, reduces administrative burdens, and improves the overall student and faculty experience. This document serves to elaborate the system's software design, including its architecture, major modules, external interfaces, design constraints, and data models. It acts as a foundational reference for developers, testers, and stakeholders to ensure that all features are designed and implemented in accordance with the specified requirements and system goals.

1.1 Purpose

This Software Design Specification (SDS) defines the design and architecture of the Mobile Application for MU Website. It is intended to guide developers during the construction phase and to provide a comprehensive overview to stakeholders including testers, project managers, and users. The SDS details functional modules, their interfaces, and behaviours.

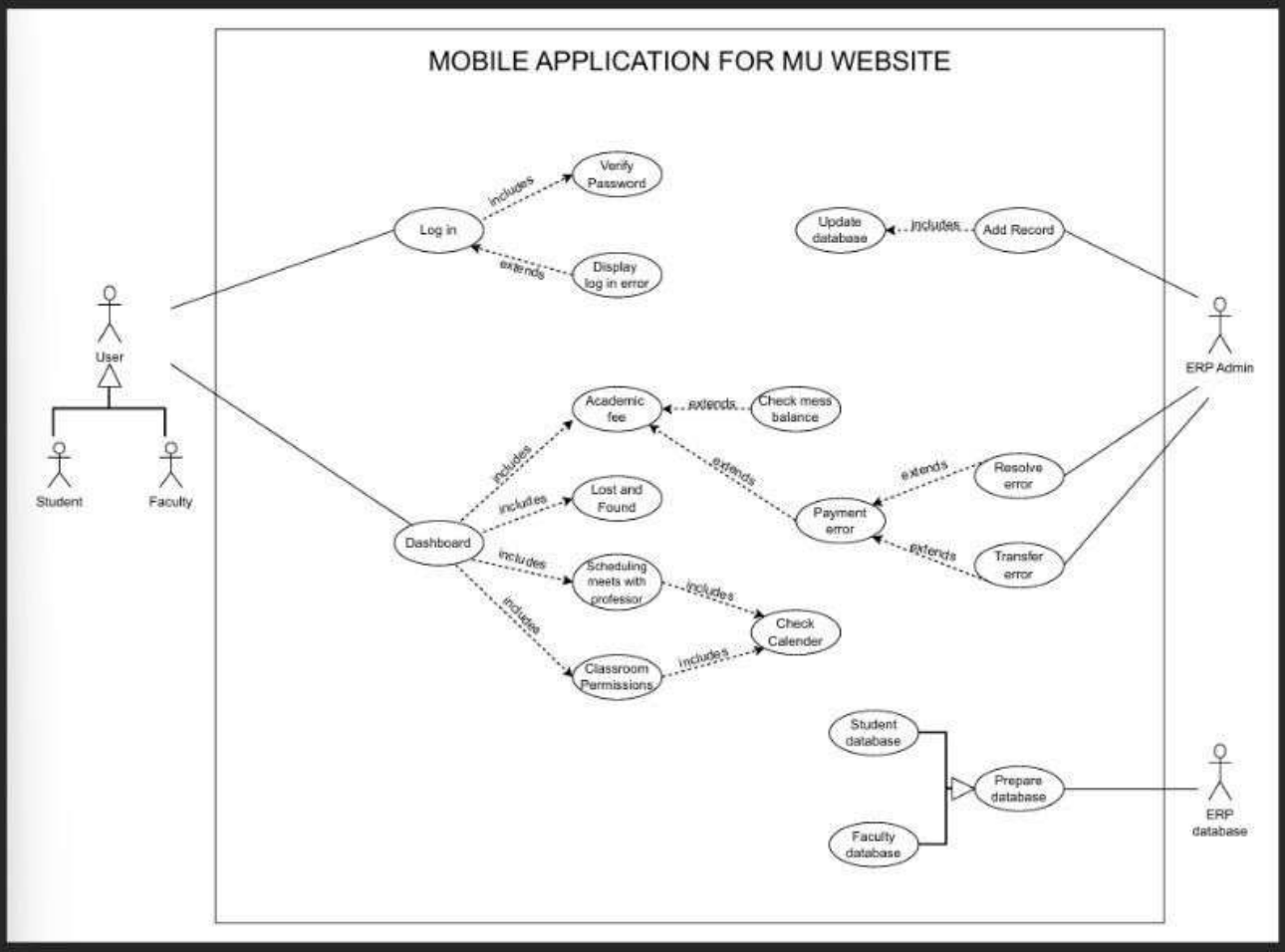
1.2 Scope

The application is designed for students, faculty, and staff of MU University to manage essential campus-related services. The scope of this version includes login, wallet tracking for day scholars, faculty contact and meeting scheduling, and a lost and found management system. The Classroom Permissions module is reserved for future development.

1.3 Definitions, Acronyms, and Abbreviations

- MU: Mahindra University
- RFID: Radio Frequency Identification
- TA: Teaching Assistant
- UI: User Interface
- API: Application Programming Interface
- DB: Database
- ERP: Enterprise Resource Planning

2 Use Case View



2.1 Use Case

2.1.1 Log In

1. **Actors**: User (Student/Faculty)
2. **Description**: The user logs in using university credentials. The system verifies the password and either grants access to the dashboard or displays a login error message.
3. **Includes**: Verify Password, Display login error

2.1.2 Dashboard Access

4. **Actors**: User (Student/Faculty)

5. **Description**: Upon successful login, the user is directed to the dashboard, which serves as a navigation hub for all features.
6. **Includes**: Academic Fee, Lost and Found, Scheduling Meetings with Professor, Classroom Permissions

2.1.3 Academic Fee

7. **Actors**: Student
8. **Description**: The user accesses the academic fee section to view payment details.
9. **Includes**: Check mess balance

2.1.4 Lost and Found

10. **Actors**: User
11. **Description**: Allows users to add a record of a lost or found item. Admin updates the database and handles records.
12. **Includes**: Add Record, Update Database

2.1.5 Scheduling Meetings with Professor

13. **Actors**: Student
14. **Description**: Enables a student to view a calendar and schedule meetings with professors or TAs.
15. **Includes**: Check Calendar

2.1.6 Classroom Permissions

16. **Actors**: User
17. **Description**: Users can request permission for classroom usage.

2.1.7 Error Handling

18. **Actors**: ERP Admin
19. **Description**: The ERP admin is responsible for resolving various system errors.
20. **Includes**: Resolve Error

21. **Extends**: Payment Error, Transfer Error

2.1.8 Backend Database Management

22. **Actors**: ERP Admin
23. **Description**: Admin handles updates to the ERP, student, and faculty databases.
24. **Includes**: Prepare Database

3 Design Overview

The software design for the MU Mobile Application is based on a modular architecture that aligns with best practices in scalable system design. Each module has a clearly defined responsibility and interacts with other components through standardized interfaces.

The application is built to comply with the design contracts derived from functional requirements, providing strong integration with the university's authentication system, RFID-based meal deduction system, and faculty scheduling database. The modular structure separates the responsibilities of user authentication, financial tracking, appointment scheduling, lost and found management, and administrative controls.

Each feature is encapsulated within a dedicated module that communicates with backend services via APIs.

3.1 Design Goals and Constraints

- Ensure responsive UI/UX on mobile
- High availability and real-time interaction
- Secure authentication and role-based access
- Monthly auto-clear for wallet transactions
- Future support for module extension

3.2 Design Assumptions

- All students have valid university credentials
- RFID readers are available at meal points
- Course and faculty data is preloaded
- Students and faculty have mobile access

3.3 Significant Design Packages

- **Auth Module**: Handles login and credential validation
- **Wallet Module**: Tracks daily and monthly transactions
- **Faculty Scheduler Module**: Manages faculty schedules and student meeting requests
- **Lost & Found Module**: Enables lost item reports and notifications
- **Notification Module**: Sends alerts and confirmations

3.4 Dependent External Interfaces

External Application and Interface Name	Module Using the Interface	Functionality/ Description
University API	LoginService	Used to validate credentials
RFID System	WalletRFID	Used to deduct meal charges from wallet
CourseDB	SubjectAPI	Fetches registered subjects and faculty

3.5 Implemented Application External Interfaces (and SOA web services)

Interface Name	Module Implementing the Interface	Functionality/ Description
CalendarScheduler	Faculty Scheduler	Integrates meeting time selection with availability
WalletTransactionAPI	Wallet Module	Handles deductions and logs

4 Logical View

4.0.1 Presentation Layer (User Interface Layer)

This is the topmost layer responsible for interactions with end users. It captures user input and displays results and messages.

- **Modules in this Layer:**
  - **Login Page** – Collects university mail and password input.
  - **Dashboard** – Shows main features like Academic Fees (Wallet), Lost and Found, Faculty Scheduling, etc.
  - **Lost & Found Interface** – UI for reporting lost/found items and viewing the bulletin.
  - **Calendar Interface** – Shows scheduled appointments with professors.
  - **Wallet Tracker** – Displays balance, deductions, and daily breakdown.

4.0.2 Application Layer

This layer acts as a bridge between the presentation and business logic layers. It coordinates user actions and requests, managing navigation and data passing.

- **Modules in this Layer:**
  - **Login Controller** – Verifies credentials and handles login errors.
  - **Navigation Controller** – Routes user to selected dashboard features.
  - **Faculty Meeting Controller** – Manages scheduling and calendar updates.
  - **Lost and Found Controller** – Validates and processes entries and updates the bulletin board.

4.0.3 Business Logic Layer

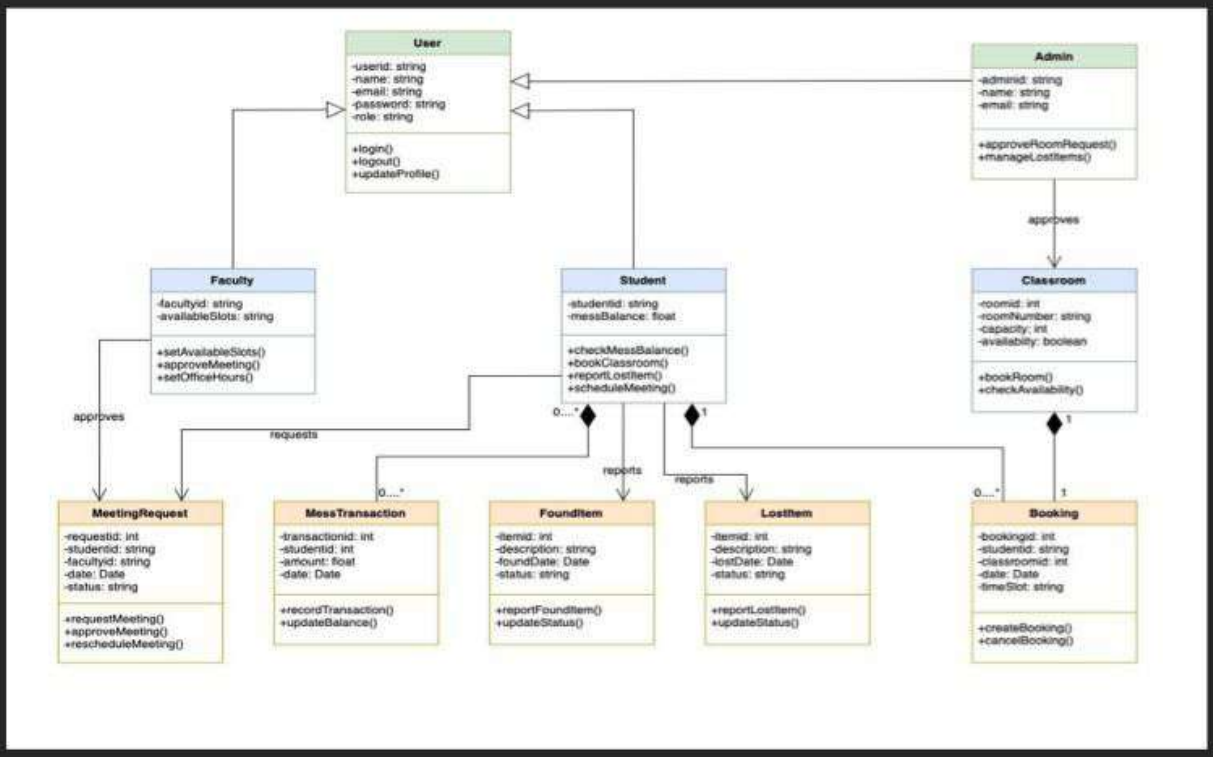
This layer encapsulates the core functionality of the system. It processes data from the application layer, applies business rules, and communicates with the data layer.

- **Modules in this Layer:**
  - **Wallet Service** – Handles RFID deductions, balance updates, and semester summaries.
  - **Faculty Scheduling Service** – Handles availability checking, meeting requests, and updates.
  - **Lost and Found Service** – Handles storage of posts, auto-deletion of entries, and location mapping.

4.0.4 Data Access Layer

- **Databases/Entities:**
  - **Student Database** – Contains user credentials, course registrations, wallet info.
  - **Faculty Database** – Contains schedules, availability slots, and course assignments.
  - **ERP Database** – Payment and transaction logs.
  - **Lost and Found Bulletin DB** – Stores entries with timestamps for auto-deletion after 30 days.

4.1 Design Model



4.1.1 User Module

Class: User

- **Responsibilities**: Acts as a base class for all types of users including Student, Faculty, and Admin.
- **Attributes**:
  - userId: string
  - name: string
  - email: string
  - password: string
  - role: string
- **Operations**:
  - login()
  - logout()
  - updateProfile()
- **Relationships**: Superclass of Student, Faculty, and Admin.

4.1.2 Student Module

Class: Student (Inherits from User)

- **Responsibilities**: Represents a student who can check mess balance, book classrooms, report lost items, and schedule meetings.
- **Attributes**:
  - studentId: string
  - messBalance: float
- **Operations**:
  - checkMessBalance()
  - bookClassroom()
  - reportLostItem()
  - scheduleMeeting()
- **Relationships**:
  - Has many MessTransaction
  - Can report LostItem and FoundItem
  - Can create MeetingRequest
  - Can create Booking

4.1.3 Faculty Module

Class: Faculty (Inherits from User)

- **Responsibilities**: Represents a faculty member who manages office hours, sets available slots, and approves meeting requests.
- **Attributes**:
  - facultyId: string
  - availableSlots: string
- **Operations**:
  - setAvailableSlots()
  - approveMeeting()
  - setOfficeHours()

- **Relationships**:
  - Approves MeetingRequest

4.1.4 Admin Module

Class: Admin (Inherits from User)

- **Responsibilities**: Manages room requests and lost item reports.
- **Attributes**:
  - adminId: string
  - name: string
  - email: string
- **Operations**:
  - approveRoomRequest()
  - manageLostItems()
- **Relationships**:
  - Approves Booking and manages LostItem

4.1.5 Meeting Management Module

Class: MeetingRequest

- **Responsibilities**: Manages student-faculty meeting requests.
- **Attributes**:
  - requestId: int
  - studentId: string
  - facultyId: string
  - date: Date
  - status: string
- **Operations**:
  - requestMeeting()
  - approveMeeting()
  - rescheduleMeeting()
- **Relationships**:
  - Created by Student
  - Approved by Faculty

4.1.6 Mess Management Module

Class: MessTransaction

- **Responsibilities**: Tracks daily mess charges and transactions for day scholars.
- **Attributes**:
  - transactionId: int
  - studentId: int
  - amount: float
  - date: Date
- **Operations**:
  - recordTransaction()
  - updateBalance()
- **Relationships**:
  - Linked to Student

4.1.7 Lost and Found Module

Class: FoundItem

- **Responsibilities**: Represents a found item reported by a student.
- **Attributes**:
  - itemId: int
  - description: string
  - foundDate: Date
  - status: string
- **Operations**:
  - reportFoundItem()
  - updateStatus()
- **Relationships**:
  - Reported by Student

Class: LostItem

- **Responsibilities**: Represents a lost item reported by a student.
- **Attributes**:

    - itemId: int
    - description: string
    - lostDate: Date
    - status: string
- **Operations**:
    - reportLostItem()
    - updateStatus()
- **Relationships**:
    - Reported by Student
    - Managed by Admin

4.1.8 Classroom Booking Module

Class: Classroom

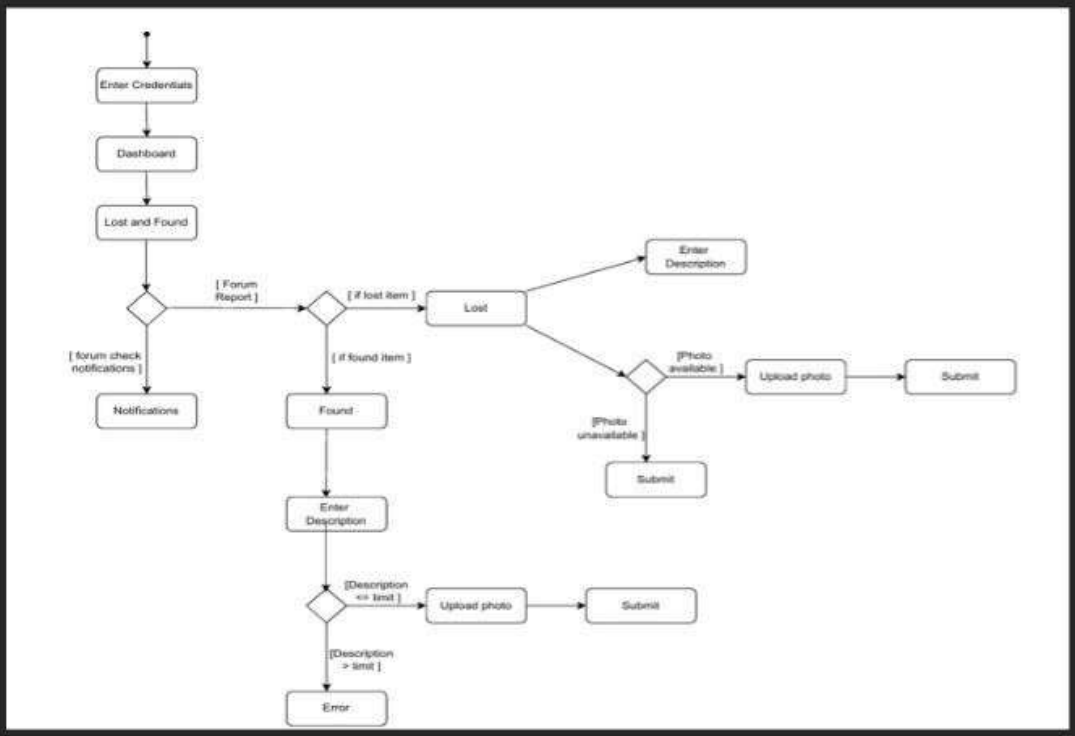
- **Responsibilities**: Represents a classroom that can be booked.
- **Attributes**:
    - roomId: int
    - roomNumber: string
    - capacity: int
    - availability: boolean
- **Operations**:
    - bookRoom()
    - checkAvailability()
- **Relationships**:
    - 1-to-many with Booking

Class: Booking

- **Responsibilities**: Represents a booking request for a classroom.
- **Attributes**:
    - bookingId: int
    - studentId: string
    - classroomId: int
    - date: Date
    - timeSlot: string
- **Operations**:
    - createBooking()
    - cancelBooking()

- **Relationships:**
  - Created by Student
  - Linked to Classroom

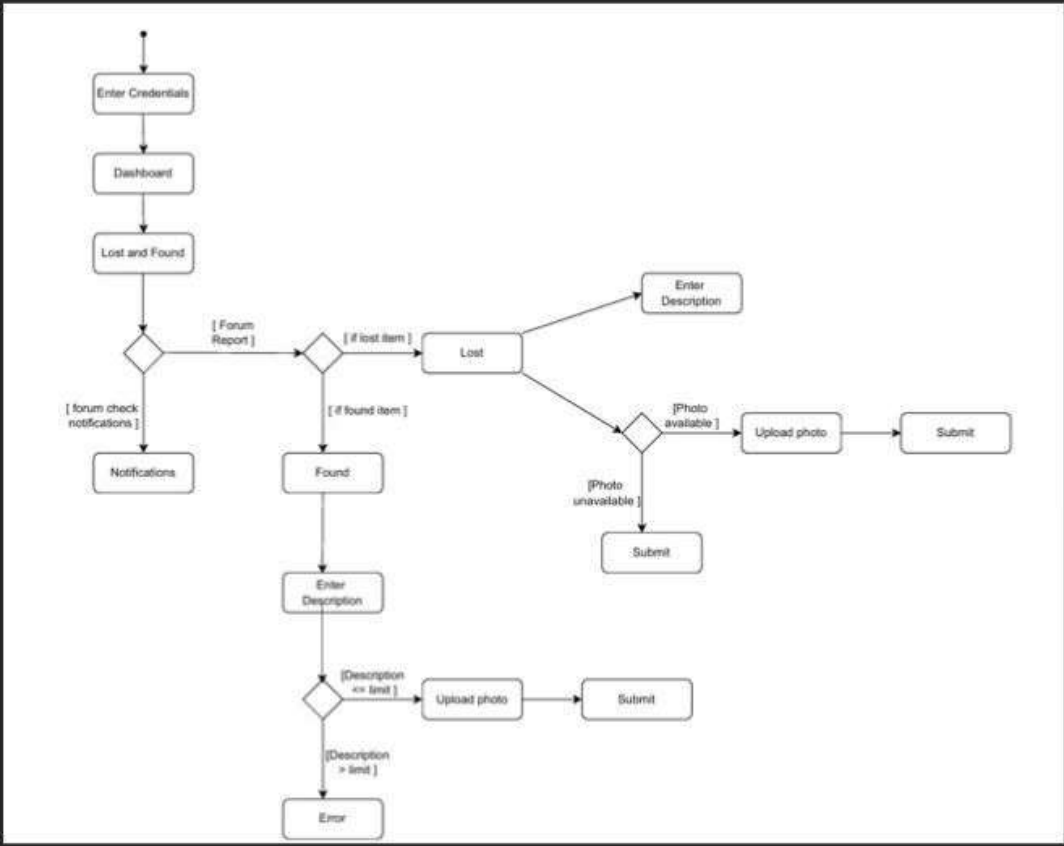
4.2 Use Case Realization



4.2.1 Use Case: Lost and Found

1. **Login & Dashboard Navigation**
   1. The user logs in with credentials.
   2. Navigates to the **Lost and Found** section from the dashboard.
2. **Lost Option**
   1. User selects "Lost".
   2. Enters description (mandatory).
   3. If a photo is available → Upload → Submit.
   4. If photo not available → Directly submit.
3. **Found Option**
   1. User selects "Found".
   2. Enters description (max 10 words).
   3. If description is valid → Submit (with or without photo).
   4. If description exceeds limit → Error displayed.

4. **Forum Notifications**
    1. User selects "Forum" to view lost/found item notifications.
    2. Items auto-expire after 30 days (display message on screen).



4.2.2 Use Case: Contact Faculty

1. **Login & Navigation**
    1. User logs in → Accesses **Contact Faculty** from dashboard.
2. **Course and Instructor Selection**
    1. Chooses registered course → Lists professors/TAs.
    2. Selects either professor or TA.
3. **Calendar & Slot Booking**
    1. Displays available slots in calendar.
    2. User books slot → Enters name, roll number, purpose of meeting.
4. **Meeting Request Flow**
    1. Request sent → Faculty/TA can approve, reject, or ask to reschedule.
    2. If approved → Slot marked busy in calendar.
    3. Notification sent to student.

5 Data View

5.1 Domain Model

Entities:

- User
- Wallet
- MeetingRequest
- LostItem
- FoundItem

Relationships:

- One User → One Wallet
- One Subject → Many Faculty
- One MeetingRequest ↔ One Calendar Event
- Many Lost/Found Items → One Forum Board

5.2 Data Model (persistent data view)

Tables:

- users
- wallet_transactions
- meeting_requests
- lost_items
- found_items

5.2.1 Data Dictionary

- `users`: (id, name, email, password, role)
- `wallet_transactions`: (id, user_id, date, time, amount_deducted, balance_after)
- `meeting_requests`: (id, user_id, faculty_id, subject, time, reason, status)
- `lost_items`: (id, user_id, description, image_path, created_at)
- `found_items`: (id, user_id, description, location, created_at)

6 Exception Handling

- **Invalid Login**: Shown when credentials are wrong
- **Insufficient Wallet Balance**: RFID scan denied, notify user
- **Meeting Conflict**: Alert if selected slot is occupied
- **Description Too Long**: Enforce 10-word limit on Found items
- **System Failure**: Log error, notify admin

7 Configurable Parameters

Configuration Parameter Name	Definition and Usage	Dynamic?
SessionTimeout	Minutes before auto logout	Yes
MaxLostDescLength	Word limit for found item descriptions	Yes
WalletPurgeInterval	Monthly clearing of transaction history	No
AutoDeleteForumItems	Duration to retain lost/found records	Yes

8 Quality of Service

8.1 Availability

The Mobile Application for the MU Website is designed for high availability to ensure uninterrupted access to key features like mess balance tracking, faculty meetings, and lost-and-found services. Modular architecture, cloud deployment, and backup strategies support system resilience. Scheduled housekeeping tasks, such as monthly transaction cleanups and semester-end data archival, are planned during off-peak hours to minimize impact. Error handling, real-time monitoring, and failover mechanisms further enhance reliability. While occasional downtime may occur during maintenance or bulk data uploads, these are communicated in advance. Overall, the system ensures a smooth and consistent experience for students, faculty, and administrators.

8.2 Security and Authorization

8.2.1 Authentication

Authentication ensures that only registered users can access the system. Role-based credentials are assigned at registration by the admin or during system onboarding.

8.2.2 Access Design and Control

- A student cannot access other students' meeting requests.
- A faculty member can only modify their own calendar availability.
- Admins have universal read/write access but cannot impersonate users.

8.2.3 User Access Setup & Management

- **Admin Dashboard:** Admin users can:
  - Onboard new users (students/faculty)
  - Assign or modify roles
  - Reset user credentials
  - Revoke access if needed
- **User Profile Module:** Users can:
  - Update personal information
  - Reset their own passwords (with email/OTP validation)

8.3 Load and Performance Implications

- Support for concurrent logins: 5000+
- Wallet and RFID expected to handle 50+ scans/min
- Async handling of faculty meeting requests

8.4 Monitoring and Control

The **Notification Handler Daemon** runs every 10 seconds to check for event triggers such as booking approvals, item matches, or mess balance updates, sending real-time push notifications and in-app alerts.

The **Booking Expiry Checker**, running every 30 minutes, identifies and removes expired classroom bookings, ensuring room availability is up to date.

The **Lost and Found Matcher** operates every 5 minutes to automatically compare new lost item reports with found items using keyword and metadata analysis, notifying both students and admins of potential matches.

The **Mess Balance Auditor** executes daily at midnight, auditing transaction logs for anomalies like duplicate deductions or unusual balance drops, and alerts administrators when discrepancies are found.