Software Design Specifications
for

Interactive Website for Mahindra University (Version 1.0)

Prepared by:
Elite Executors
Mahindra University

Document Information

Title: Software Design Specification for Intranet Portal for MU	
Project Manager: Avinash Arun Chauhan	**Document Version No: version 1**
	Document Version Date: 09-04-2025
Prepared By: Elite Executors	**Preparation Date:09-04-2025**

Table of Contents

1. Introduction

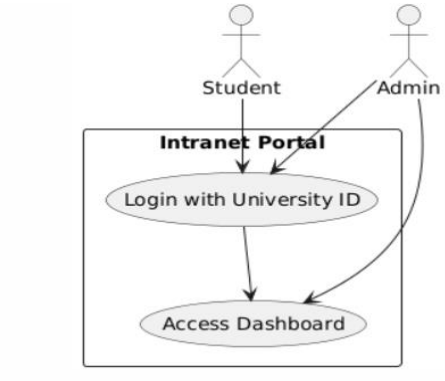
1.1 Purpose
This Software Design Specification (SDS) provides an architectural and detailed design description for the "Interactive Website for Mahindra University". It is intended for use by the developers, testers, and stakeholders to understand how the system will be constructed to meet the requirements outlined in the SRS.

1.2 Scope
This design document covers all components of the intranet portal, including attendance, student analytics, course materials, scheduling, appointments, announcements, hostel management, club activities, and complaint tracking.

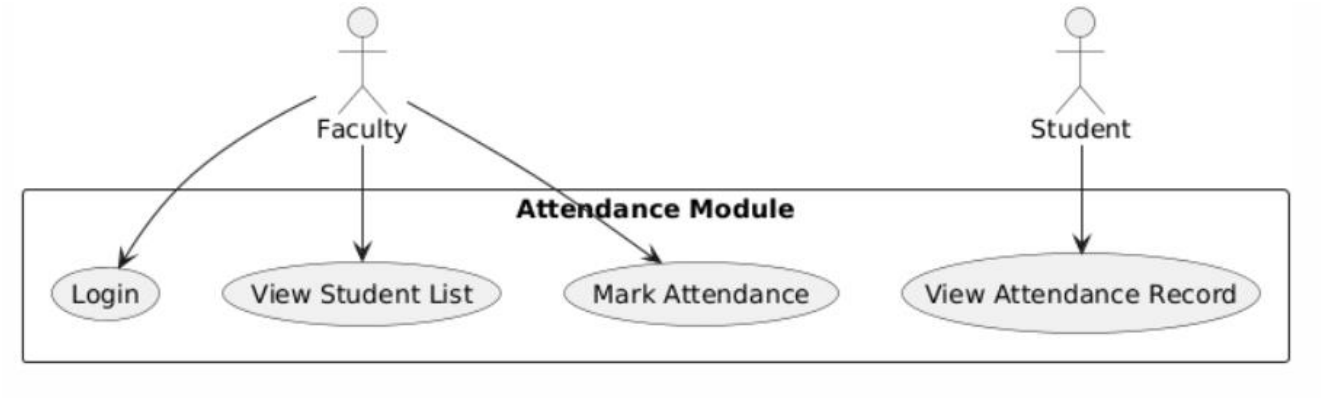
1.3 Definitions, Acronyms, and Abbreviations
- **MERN Stack**: MongoDB, Express.js, React.js, Node.js
- **RBAC**: Role-Based Access Control
- **JWT**: JSON Web Token
- **UI**: User Interface
- **DBMS**: Database Management System

1.4 References
- SRS for Intranet Portal for MU
- IEEE SRS Documentation Standards
- MongoDB, Express.js, React.js, Node.js Documentation

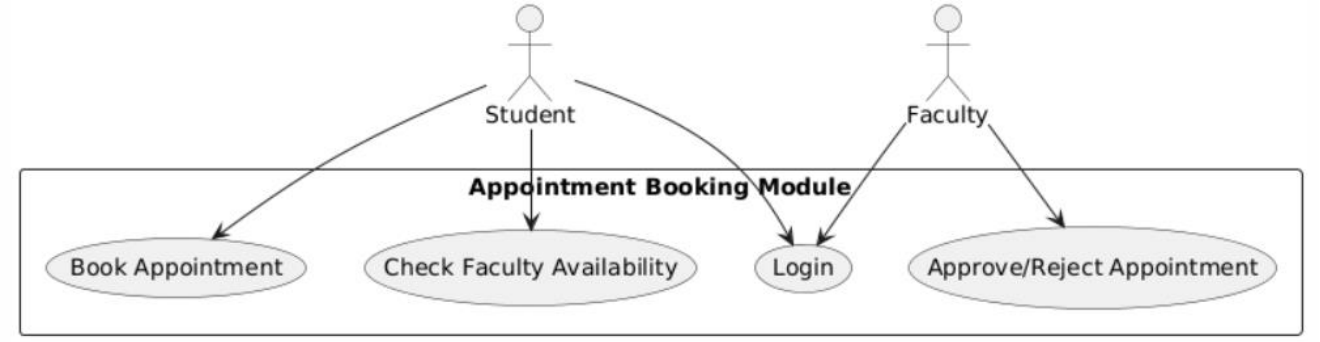
2 Use Case View
2.1 Use Case
Key use cases include:
- U1: Student Login and Dashboard Access



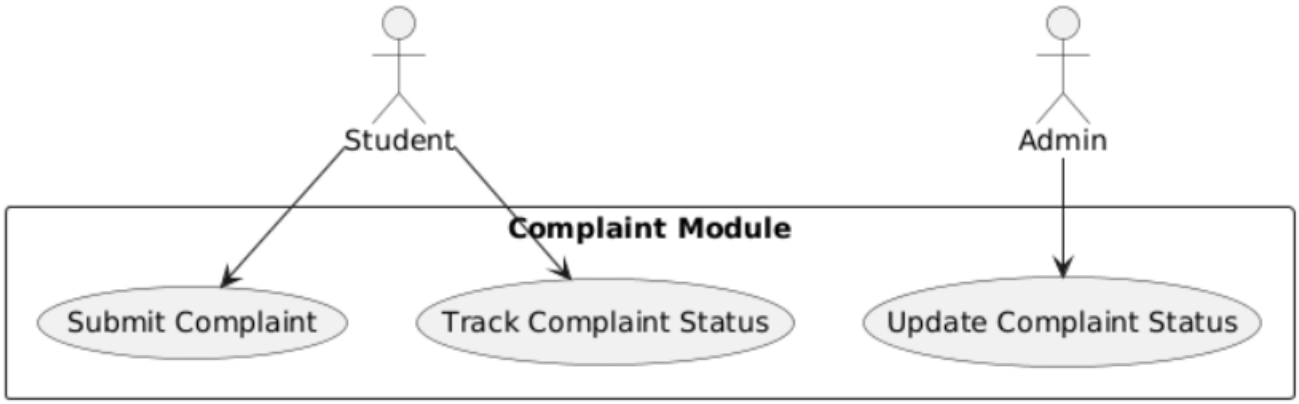
- U2: Faculty Attendance Submission



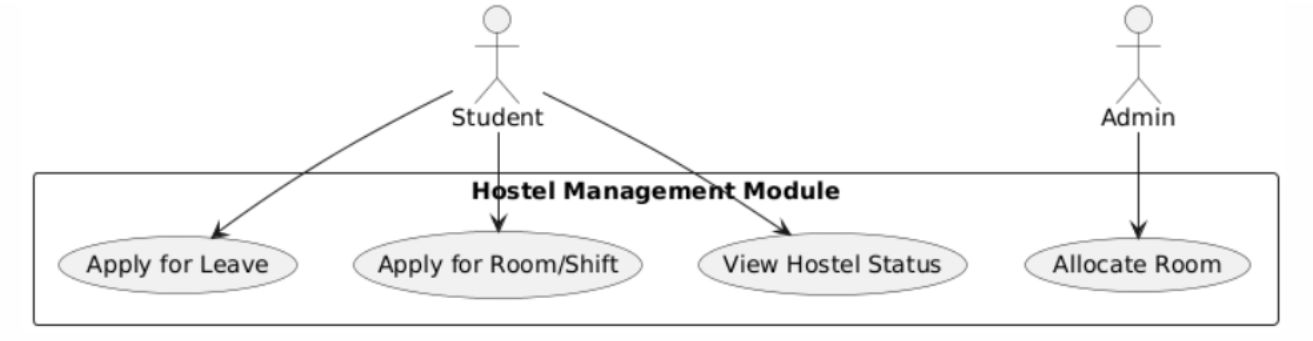
- U3: Appointment Booking with Faculty



- U4: Complaint Submission and Tracking



- U5: Hostel Room Allocation



3 Design Overview

3.1 Design Goals and Constraints

Design Goals:

- Create a user-friendly web interface for students, faculty, and admin.
- Provide role-based dashboards with only relevant features.
- Ensure secure login using university credentials.
- Allow modular growth for future enhancements (e.g., exam schedules, results).

Design Constraints:

- The system must be developed using MERN stack only.
- Authentication is limited to valid university credentials.
- System should support up to 5000 concurrent users.
- Deployment and testing time is limited due to academic schedule.
- Designed for internal intranet use, but accessible over the internet.
- No legacy code involved; system is built from scratch.

3.2 Design Assumptions

- The university IT department will provide access to a basic authentication service or allow local credential validation.
- All users (students/faculty/admin) have unique university IDs and valid email addresses.
- Users will access the portal via modern browsers (e.g., Chrome, Edge, Firefox).
- The system assumes internet availability during use.
- All modules will use MongoDB Atlas or a local MongoDB server for storing persistent data.

3.3 Significant Design Packages

- Auth Module
- Dashboard Module (Student/Faculty/Admin)
- Attendance Management
- Course Content Module
- Scheduling and Calendar
- Complaint and Hostel Manage

3.4 Dependent External Interfaces

The table below lists the public interfaces this design requires from other modules or applications.

External Application and Interface Name	Module Using the Interface	Functionality/ Description
University Auth System	SSO Login	Used for secure login and session management

3.5 Implemented Application External Interfaces (and SOA web services)

The table below lists the implementation of public interfaces this design makes available for other applications.

Interface Name	Module Implementing the Interface	Functionality Description
`/api/auth/login`	`auth`	Accepts user ID and password, returns JWT if valid
`/api/attendance/mark`	`attendance`	Faculty can submit attendance for a class
`/api/appointments/book`	`appointments`	Students can book a new appointment
`/api/complaints/submit`	`complaints`	Students can submit a new complaint
`/api/hostel/apply-room`	`hostel`	Allows students to apply for a room
`/api/dashboard/data`	`dashboard`	Returns dashboard data based on role

4 Logical View

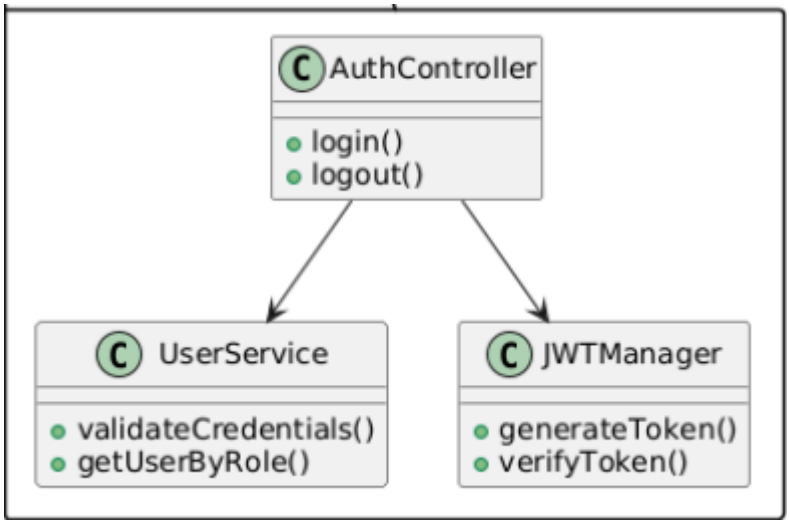
4.1 Design Model

Module 1: Authentication Module
Main Classes:

- AuthController: Handles login, registration, and logout.
- UserService: Manages user data and authentication logic.
- JWTManager: Generates and validates tokens.

Responsibilities:

- Authenticate users using university credentials.
- Generate JWT for session handling.
- Enforce RBAC (Role-Based Access Control).

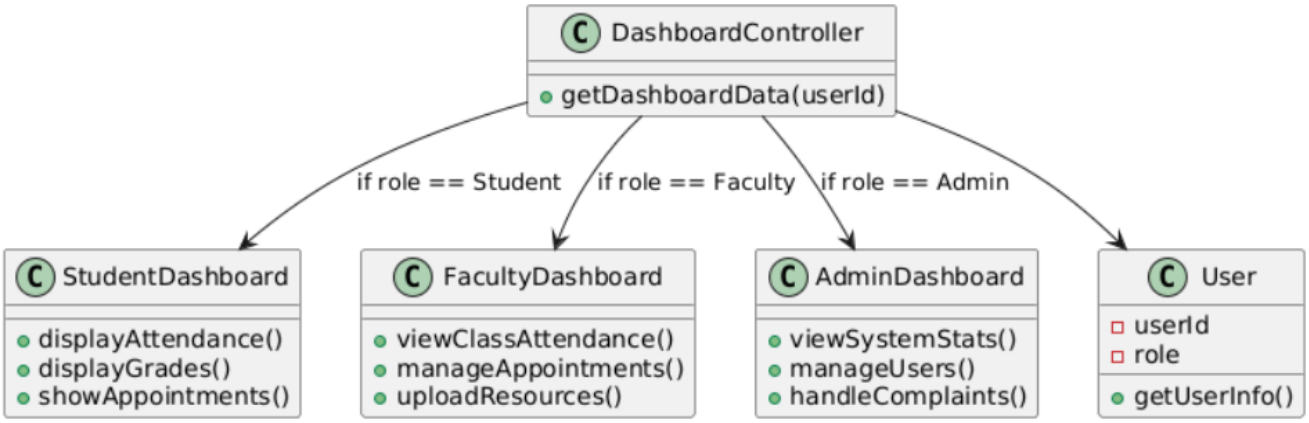


Module 2: Dashboard Module
Main Classes:

- DashboardController: Aggregates data for user-specific dashboards.
- StudentDashboard, FacultyDashboard, AdminDashboard: Render role-based views.

Responsibilities:

- Display personalized information based on role.
- Provide access links to relevant modules.



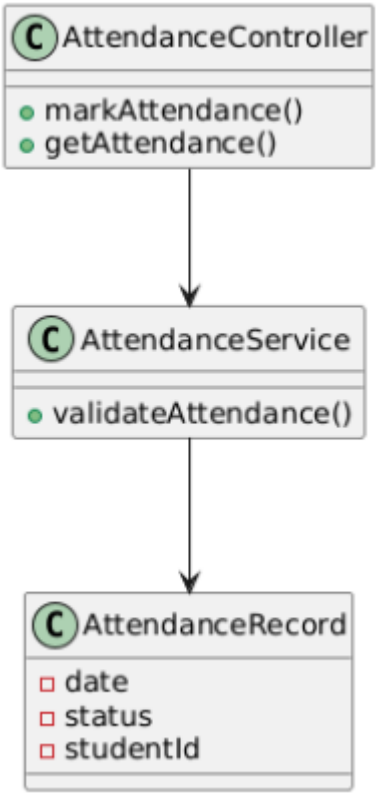
Module 3: Attendance Management Module

Main Classes:

- AttendanceController: Handles marking and retrieving attendance.
- AttendanceService: Validates logic and business rules.
- AttendanceRecord: Data model for attendance.

Responsibilities:

- Allow faculty to mark attendance.
- Enable students to view their records.

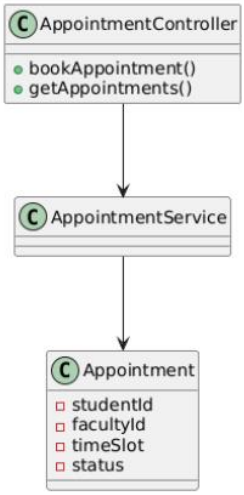


Module 4: Appointment Module
Main Classes:

- AppointmentController
- AppointmentService
- Appointment: Model storing appointment data.

Responsibilities:

- Students can book appointments.
- Faculty can approve/reject requests.

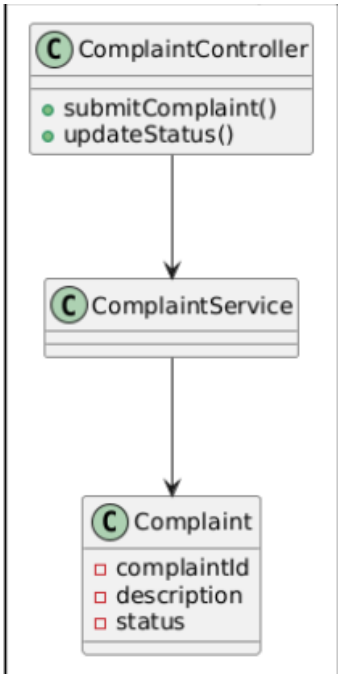


Module 5: Complaint Module
Main Classes:

- ComplaintController
- ComplaintService
- Complaint: Model for complaints with status updates.

Responsibilities:

- Log, update, and resolve complaints.

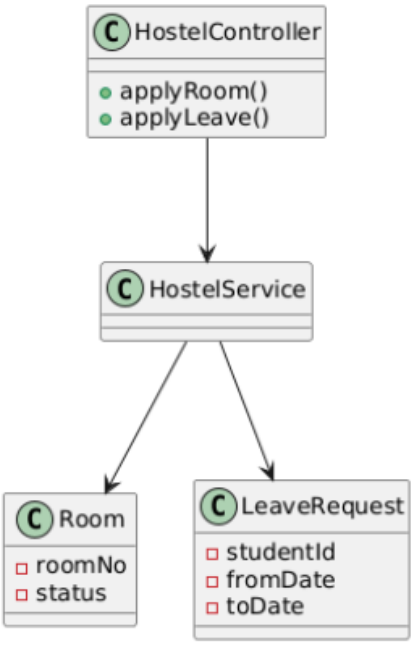


Module 6: Hostel Management Module
Main Classes:

- HostelController
- HostelService
- Room, LeaveRequest

Responsibilities:

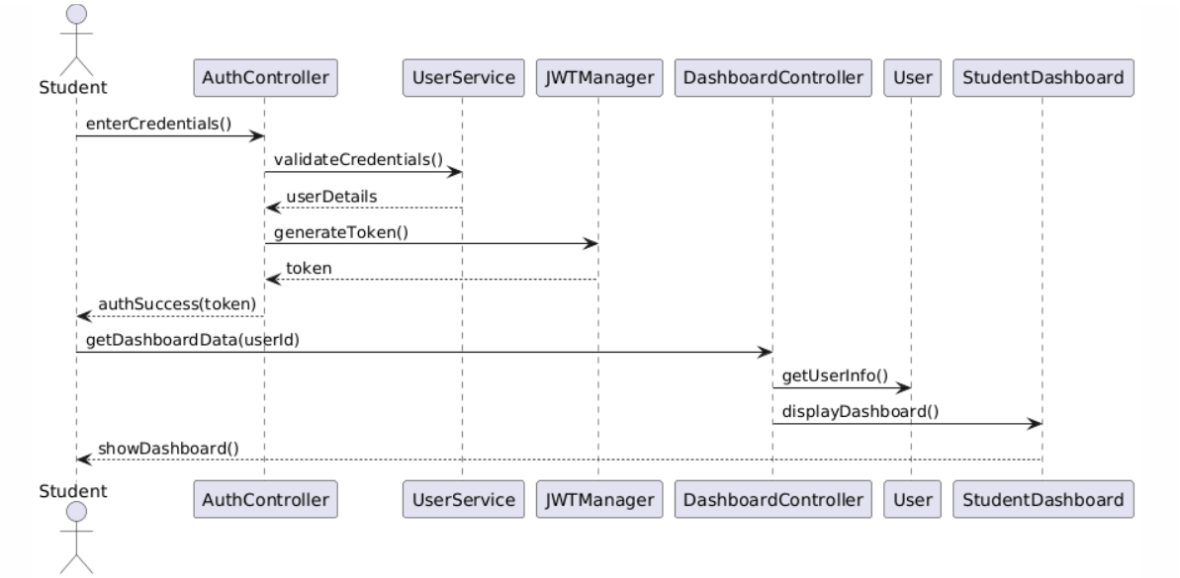
- Room allocation and shifting.
- Leave request approval workflow.



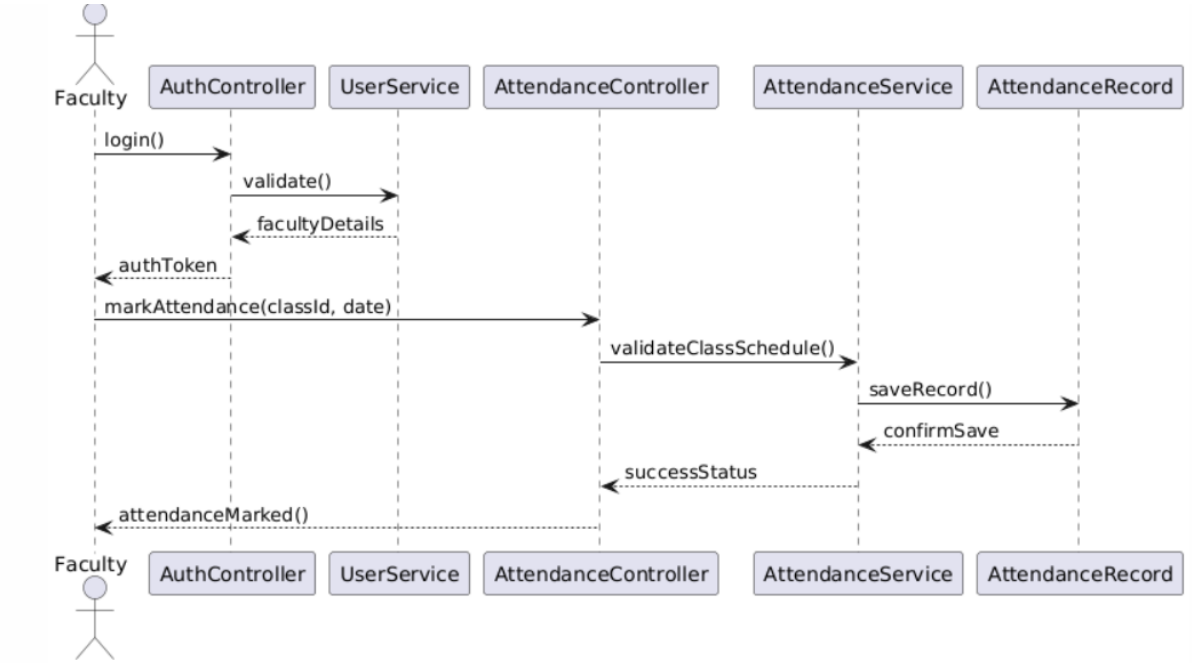
4.2 Use Case Realization

Use activity and sequence diagrams to show interactions (e.g., booking an appointment includes checking availability, selecting time, and sending confirmation)

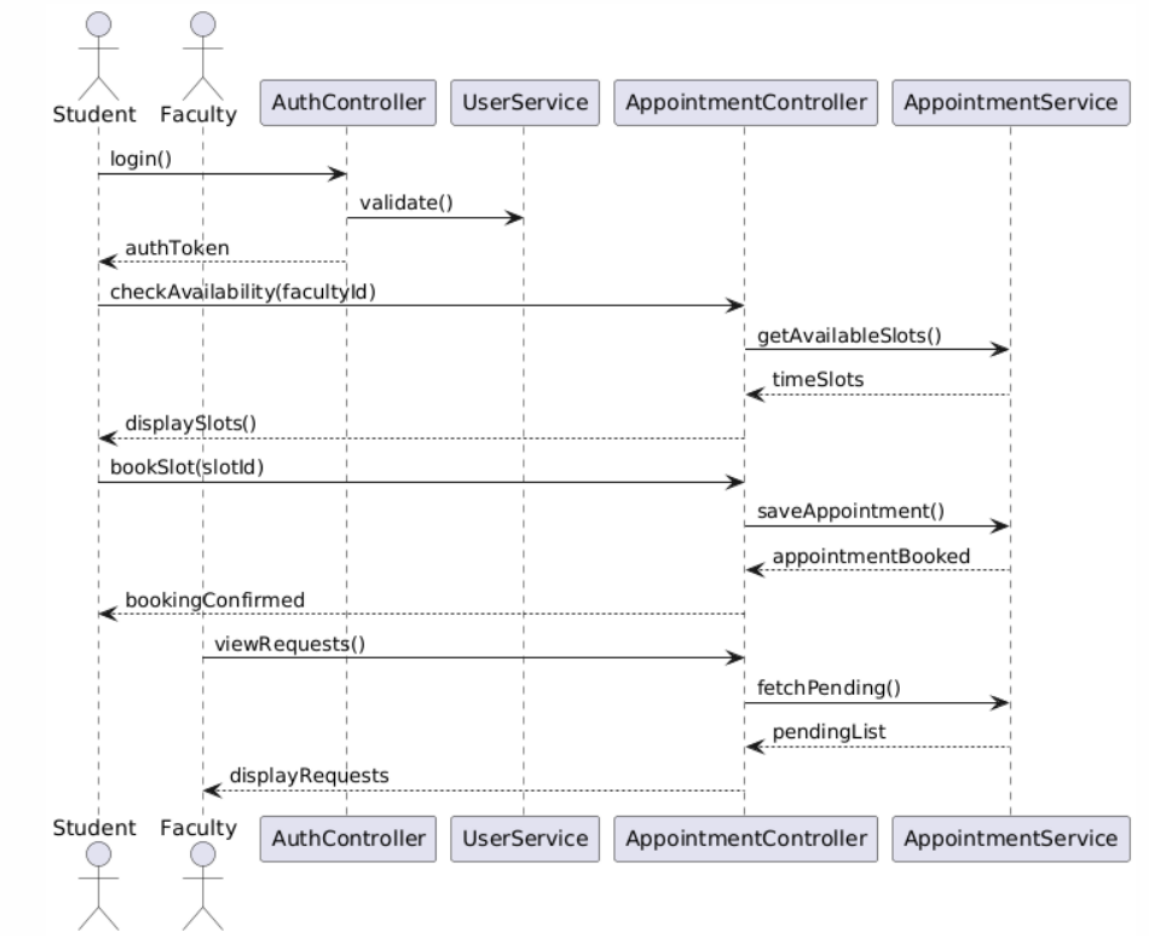
U1: Student Login and Dashboard Access



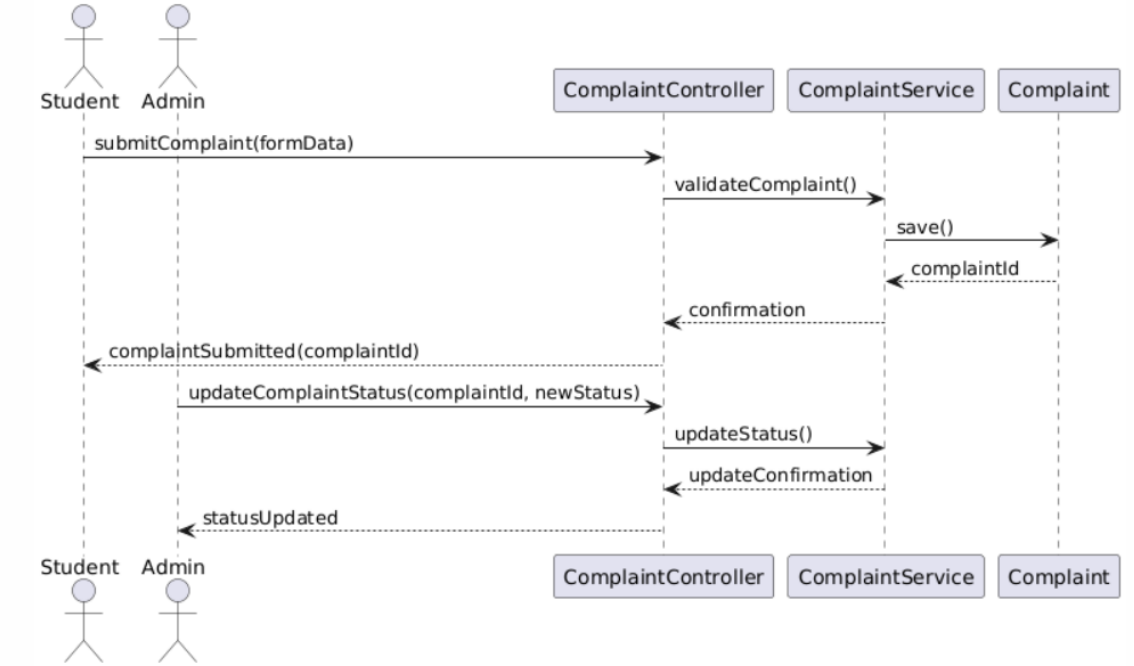
U2: Faculty Attendance Submission



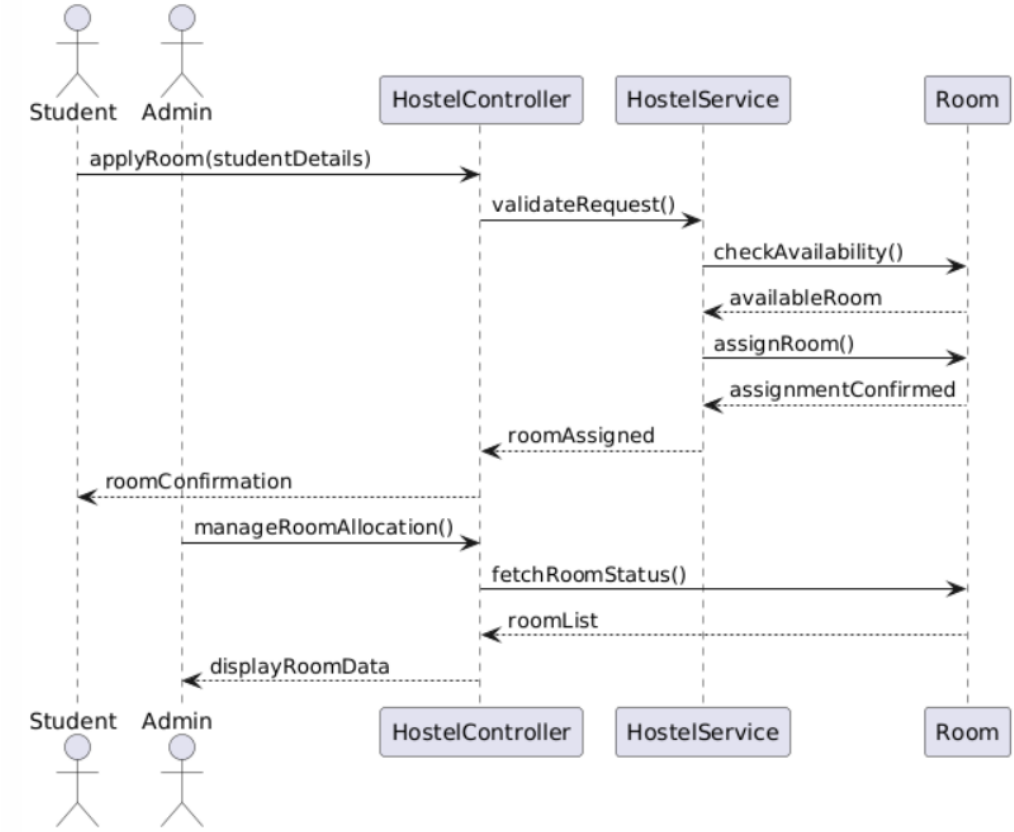
U3: Appointment Booking with Faculty



U4: Complaint Submission and Tracking



U5: Hostel Room Allocation



5 Data View

5.1 Domain Model

The domain model consists of the primary entities in the system and the relationships between them. Each entity corresponds to a MongoDB collection and represents a distinct business object.

Entities and Relationships:

- **User** *(Supertype for Student, Faculty, Admin)*
- **AttendanceRecord** *(related to User and Course)*
- **Appointment** *(between Student and Faculty)*
- **Complaint** *(filed by Student, handled by Admin)*
- **Room** *(allocated to Students)*
- **LeaveRequest** *(by Student, approved by Admin)*
- **Course** *(mapped to Faculty and Student)*
- **Event / ClubActivity**

ER-style Relationship Summary:

- A Student can have many AttendanceRecords, Appointments, Complaints, LeaveRequests.
- A Faculty teaches many Courses and receives many Appointments.
- Room is assigned to one Student.
- Admin oversees Complaints, Room Allocations, and Approves Leave.

5.2 Data Model (Persistent Data View)

The system uses a NoSQL (MongoDB) document schema. Data is stored in collections, with documents representing objects.

Primary Collections:

- users → Common schema with role-based differentiation
- attendance_records → Stores date-wise attendance
- appointments → Tracks meetings between students and faculty
- complaints → Stores complaints with status
- rooms → Hostel room assignments
- leave_requests → Student leave applications
- courses → Faculty-created courses with enrolled students
- events → Club or university events

5.2.1 Data Dictionary

Entity	Attribute	Type	Description	Example / Notes
`users`	`_id`	ObjectId	Unique identifier	MongoDB auto-generated
	`name`	String	Full name of the user	"John Doe"
	`email`	String	Official university email	"john@mu.edu.in"
	`role`	Enum	`Student`, `Faculty`, `Admin`	Defines access level
	`passwordHash`	String	Encrypted password	Stored securely
	`profileImage`	String	Profile image URL/path	Optional
`attendance_records`	`_id`	ObjectId	Record ID	
	`studentId`	ObjectId	Linked to `users`	Student foreign key
	`courseId`	ObjectId	Linked to `courses`	Course foreign key
	`date`	Date	Attendance date	"2025-04-09"
	`status`	Enum	`Present`, `Absent`	Attendance mark
`appointments`	`_id`	ObjectId	Appointment ID	
	`studentId`	ObjectId	Student reference	
	`facultyId`	ObjectId	Faculty reference	
	`dateTime`	DateTime	Appointment slot	"2025-04-12 14:00"
	`status`	Enum	`Pending`, `Approved`, `Rejected`	

`complaints`	`_id`	ObjectId	Complaint ID	
	`studentId`	ObjectId	Submitted by student	
	`description`	String	Complaint details	"Wi-Fi not working"
	`status`	Enum	`Open`, `In Progress`, `Resolved`	Default: Open
	`timestamp`	DateTime	Submission time	"2025-04-09 10:15"
`rooms`	`_id`	ObjectId	Room ID	
	`roomNo`	String	Room number	"B-103"
	`studentId`	ObjectId	Assigned student	Nullable
	`status`	Enum	`Occupied`, `Vacant`	
`leave_requests`	`_id`	ObjectId	Leave request ID	
	`studentId`	ObjectId	Requesting student	
	`fromDate`	Date	Start of leave	"2025-04-14"
	`toDate`	Date	End of leave	"2025-04-17"
	`reason`	String	Reason for leave	"Medical"
	`status`	Enum	`Pending`, `Approved`, `Denied`	Admin-reviewed
`courses`	`_id`	ObjectId	Course ID	
	`courseId`	String	Course code	"CS101"
	`name`	String	Course name	"Data Structures"
	`facultyId`	ObjectId	Faculty in charge	
	`enrolledStudents`	Array<ObjectId>	List of student IDs	
`events`	`_id`	ObjectId	Event ID	
	`name`	String	Event title	"Hackathon 2025"
	`description`	String	Event info	
	`organizerId`	ObjectId	Organizer (student/faculty)	
	`participants`	Array<ObjectId>	Users participating	

6 Exception Handling

Exception Type	Circumstances	Handling Strategy	Logging	Follow-Up Action
AuthenticationError	Invalid login credentials, token expired	Return 401 Unauthorized with message	Log user ID and IP address	Prompt user to login again
AuthorizationError	Accessing a resource without permission	Return 403 Forbidden	Log role and attempted resource	Redirect to home/dashboard
ValidationError	Invalid or missing request body fields	Return 400 Bad Request with field-specific errors	Log endpoint and request body	Notify user with validation hints
DatabaseError	DB connectivity failure, query timeout, write conflict	Return 500 Internal Server Error	Log error stack, query, and user session	Retry logic or contact support
ResourceNotFoundError	Requested entity (e.g., user, complaint) doesn't exist	Return 404 Not Found	Log resource ID and request path	Show "not found" page/message
FileUploadError	Failure in uploading or saving user file	Return 400 with reason	Log file type, size, and path	Ask user to re-upload
AppointmentConflictError	Booking clashes with an already taken slot	Return 409 Conflict	Log student ID, faculty ID, slot	Prompt for alternate slots
UnhandledException	Any other uncaught runtime error	Return 500 Internal Server Error	Global logger catches and stores full trace	Notify dev team; display fallback message

Frontend Error Handling

- React **Error Boundaries** to catch rendering issues in UI components.
- **Toast Notifications** and inline messages for recoverable issues (e.g., form errors).
- **Fallback UIs** for key pages (e.g., "Something went wrong. Try again later.")

Monitoring and Alerts

- Alerts triggered for repeated exceptions or critical backend failures.
- Admin dashboard includes a basic monitoring panel showing error logs and system health.

7 Configurable Parameters

The following table lists parameters that can be configured by the system administrator or during deployment. These control the system's basic behavior and settings.

Configuration Parameter Name	Definition and Usage	Dynamic? (Can be changed without restarting)
SESSION_TIMEOUT_MINUTES	Time in minutes before a user is automatically logged out due to inactivity	Yes
MAX_APPOINTMENTS_PER_DAY	Maximum number of appointments a student can book in one day	Yes
HOSTEL_MAX_CAPACITY	Number of students that can be assigned to each hostel block	No
ANNOUNCEMENT_DISPLAY_DAYS	Number of days an announcement remains visible	Yes
ALLOWED_FILE_SIZE_MB	Maximum file upload size for course materials or complaints (in MB)	No
DEFAULT_USER_ROLE	Role assigned to new users if not specified (e.g., “Student”)	No
COMPLAINT_RESPONSE_DAYS	Expected number of days to resolve a complaint	Yes
LEAVE_MAX_DAYS	Maximum number of days allowed per leave request	Yes

8 Quality of Service

8.1 Availability

The system should be available to students, faculty, and admins **24/7**, except during scheduled maintenance.

Design choices that support availability:

- Built using the MERN stack (React + Node.js + MongoDB) to allow quick recovery from crashes.
- Uses modular code, so one feature failing doesn't crash the whole system.
- Data backups can be scheduled during off-hours (e.g., midnight).
- Maintenance like database cleanup and logs archiving will be planned during **semester breaks or weekends** to reduce impact.

8.2 Security and Authorization

Security Requirements:

- All users must log in using **University ID and password**.
- Passwords are stored as **hashed values**, not in plain text.
- Each user has a role: `Student`, `Faculty`, or `Admin`.

Authorization Features:

- **Role-Based Access Control (RBAC)** is implemented.
  - Students can only access their own data (attendance, appointments, complaints, etc.).
  - Faculty can view/manage data for the courses they teach.
  - Admins can access all user and system data.

Additional Security Notes:

- Only admins can manage user roles or approve room/leave requests.
- Input validation is performed on both frontend and backend to prevent harmful data (e.g., injection attacks).

8.3 Load and Performance Implications

Expected Usage:

- ~3000–5000 users from Mahindra University may use the system.

Design Responses to Load Requirements:

- Database queries are optimized with indexes (e.g., on `userId`, `date`, `courseId`).
- Pagination is used for displaying large lists (like complaints or student records).
- Heavy operations like report generation can be performed in the background (future feature).
- React ensures that only updated components are re-rendered, improving performance.

Target Performance:

- System should respond within **2 seconds** for most actions.
- Should support **100+ concurrent users** during peak hours (e.g., before exams or hostel allocation time).

8.4 Monitoring and Control

Even for a beginner-level project, basic monitoring is helpful.

Monitoring Features:

- Errors (like login failure or server crash) are logged to a file or console.
- Admin dashboard shows:
    - Number of active complaints
    - Number of appointments today
    - System health (basic: status OK/Error)

Future Scope (Optional):

- Add admin email alerts for frequent system errors or failed login attempts.
- Implement uptime monitoring tools (like UptimeRobot) for live status checking.