OD MANAGEMENT SYSTEM

LAB RECORD

CS19442-SOFTWARE ENGINEERING CONCEPTS

Submitted by:

Laavanya P	220701139
Lakshanya D	220701140
Lalit Prasanna	220701142
Lithan	220701143
Logesh	220701144
Logeshwaran	220701145

Software Concepts & Engineering - Lab Manual

Overview of the Project

Business Architecture Diagram

Requirements as User Stories

Architecture Diagram

Test Strategy

Deployment Architecture of the application

ABSTRACT

The OD (On Duty) Management System is a comprehensive web application designed to streamline and automate the process of managing and approving On Duty requests within an educational institution. The system facilitates seamless communication between faculty members and students, ensuring efficient handling of OD requests for various events.

The system features two distinct user interfaces, one for faculty members and another for students. Faculty members can log in to the system to input details of events requiring On Duty approval. Students, on the other hand, can access the system to view the list of events and submit OD requests to their respective Class Incharge, academic head, and Head of Department (HOD). Students are required to attach relevant registration proofs while submitting their OD requests.

Upon submission, the OD requests are sent to the relevant faculty members for verification. The faculty can review the attached registration proof, ensuring the authenticity of the request. Faculty members have the authority to either approve the request by generating an OD letter or reject it based on the provided information.

Students can track the status of their OD requests through the web application, receiving notifications upon approval or rejection. Additionally, once the event concludes, students are allowed to upload their certificates directly into the system.

The OD Management System utilizes modern web technologies and platforms to ensure a user-friendly and efficient experience. The system is developed using a robust web development framework and may incorporate technologies such as HTML, CSS, JavaScript for the front-end, and a back-end framework python with flask for server-side functionality. The

system's database is implemented using a reliable database management system like MySQL. Security measures, such as user authentication and authorization protocols, are integrated to protect sensitive information.

Overall, the OD Management System enhances the efficiency of OD request processing, providing a centralized platform for faculty and students to manage, track, and communicate regarding On Duty approvals within the academic institution

OVERVIEW

Introduction

The Duty Management System for a college is designed to streamline and automate the various administrative and academic processes related to managing duties, such as tracking students' on-duty (OD) requests, updating event participation by faculty, and monitoring students' progress in their OS (Operating System) application projects. This system aims to enhance efficiency, reduce paperwork, and provide a centralized platform for all stakeholders—students, faculty, and administrators.

Key Features

1.On-Duty (OD) History Management

Student Requests : Students can submit OD requests online, specifying the reason, date, and duration of their absence.

Approval Workflow : Faculty can review, approve, or reject OD requests. Notifications are sent to students regarding the status of their requests.

History Tracking : Both students and faculty can view a detailed history of past OD requests, including dates, reasons, and approval statuses.

Analytics : The system can generate reports on OD trends, helping administrators identify patterns and take necessary actions.

2. Event Updation by Faculty

Event Creation: Faculty can create events, such as seminars, workshops, and extracurricular activities, and update details like date, time, and venue.

Attendance Management: Faculty can mark attendance for events, and the system can automatically update students' OD records if they participated.

Notification System: Students receive notifications about upcoming events and any changes to event details.

3. Progress Tracking of OD Requests

Status Monitoring: Students can track the progress of their OD requests in real-time, seeing whether their requests are pending, approved, or rejected.

Detailed Updates: The system provides detailed updates at each stage of the approval process, including comments from faculty or administrators.

Automated Notifications: Students receive automated notifications via email or SMS when the status of their OD request changes.

History Logs: All status changes and comments are logged, providing a comprehensive history of the OD request process.

4. User Roles and Permissions

Role-Based Access Control : Different user roles (students, faculty, administrators) have varying levels of access and permissions, ensuring data security and privacy.

Customizable Permissions : Administrators can customize permissions for specific users or groups, allowing flexibility in managing access.

5. Notifications and Alerts

Automated Notifications: The system sends automated email and SMS notifications for important updates, such as OD approval status, upcoming events, and project deadlines.

Alert Management: Users can set up custom alerts for specific events or actions, ensuring they stay informed about critical updates.

6. User-Friendly Interface

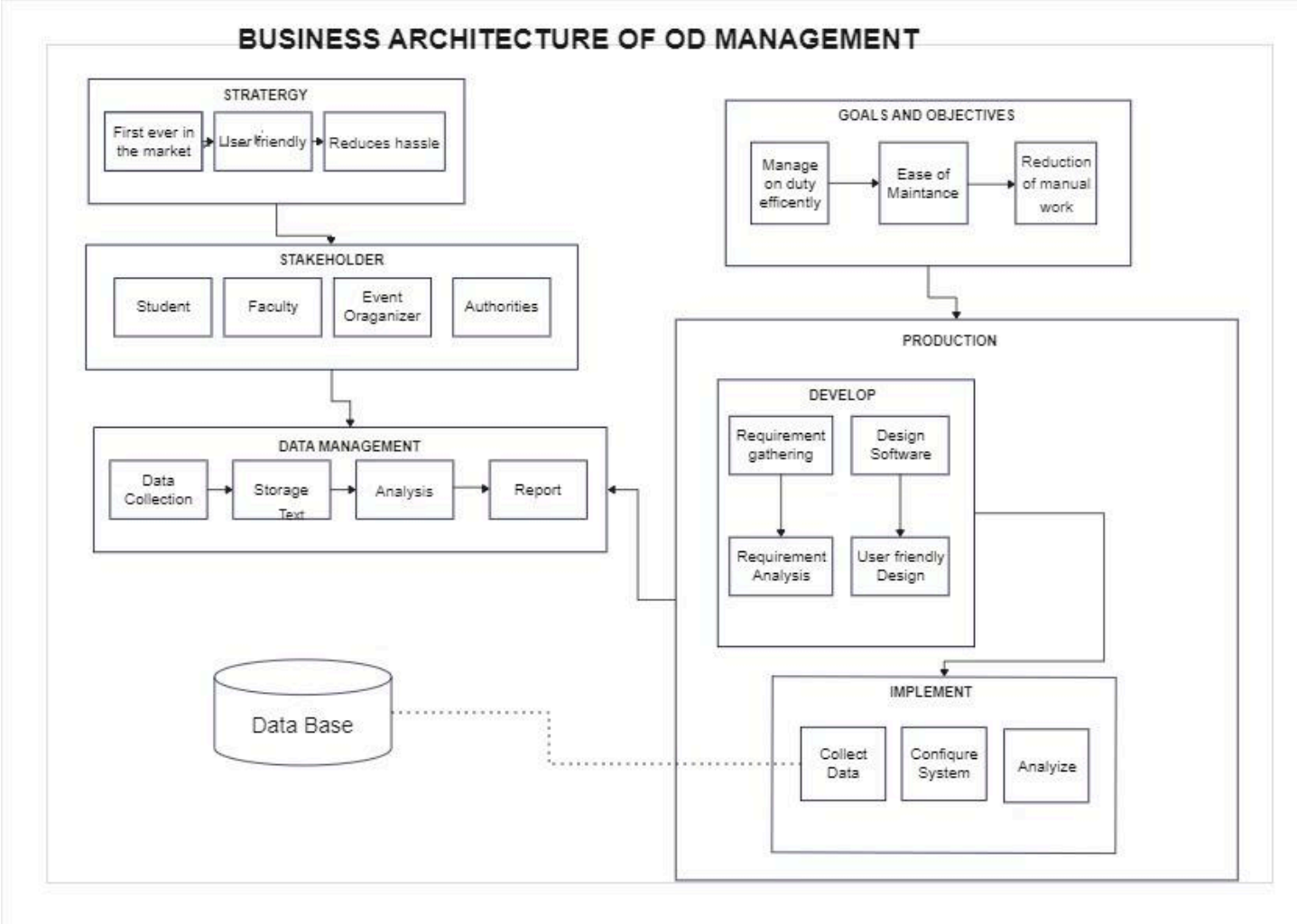
Dashboard: A centralized dashboard provides an overview of all activities, including pending OD requests, upcoming events, and project progress.

Responsive Design: The system is accessible from various devices, including smartphones, tablets, and desktops, ensuring ease of use.

Conclusion

The On Duty Management System for colleges offers a robust solution to manage various administrative and academic tasks efficiently. By automating OD requests, event updates, and project progress tracking, the system enhances productivity, ensures transparency, and fosters better communication among students, faculty, and administrators. With its user-friendly interface and powerful features, the system is poised to become an indispensable tool in the modern educational environment.

BUSINESS ARCHITECTURE



Strategy

- **First Over in the Market**: The system aims to be a pioneer in the market, providing a unique and comprehensive solution for duty management in colleges.
- **User-Friendly**: Emphasizes ease of use to ensure that all stakeholders, including students, faculty, and administrators, can navigate the system effortlessly.

- **Reduces Hassle**: Aims to simplify and streamline processes to reduce the manual effort and administrative burden associated with managing on-duty requests.

Goals and Objectives

- **Manage On-Duty Efficiently**: The system is designed to handle on-duty requests efficiently, ensuring quick processing and clear communication of approval status.
- **Ease of Maintenance**: Built with ease of maintenance in mind, the system ensures that updates and modifications can be made with minimal disruption.
- **Reduction of Manual Work**: By automating various processes, the system significantly reduces the need for manual intervention, thereby saving time and reducing errors.

Stakeholders

- **Student**: Primary users who submit on-duty requests and track their status.
- **Faculty**: Review and approve on-duty requests and update event participation.
- **Event Organizer**: Create and manage events, track attendance, and update relevant data.
- **Authorities**: Administrators who oversee the entire process and generate reports for analysis.

Data Management

- **Data Collection**: Gather data from various inputs such as OD requests, event participation, and project updates.

- **Storage & Test**: Securely store data and perform regular testing to ensure data integrity and system reliability.
- **Analysis**: Analyze collected data to identify trends, patterns, and areas for improvement.
- **Report**: Generate detailed reports to provide insights into OD requests, event participation, and overall system performance.

Database

- **Central Repository**: A central database stores all the data related to OD management, ensuring that information is easily accessible and well-organized.

Production

- **Develop**:
  - **Requirement Gathering**: Collect requirements from stakeholders to ensure the system meets their needs.
  - **Requirement Analysis**: Analyze the gathered requirements to design a solution that addresses all identified needs.
  - **Design Software**: Develop a user-friendly design that facilitates ease of use and efficient workflow.
  - **User-Friendly Design**: Focus on creating an intuitive interface that enhances user experience.
- **Implement**:
  - **Collect Data**: Gather and input initial data into the system to ensure it is fully operational.
  - **Configure System**: Set up and configure the system according to the specific needs of the institution.
  - **Analyze**: Continuously monitor and analyze system performance to identify and address any issues.

Requirements as user stories

As a **student**, I should be able to log in to the website using my university credentials so that I can access the website

As a **student**, I must be able to reset my password so that I can log in even if I forget the password

As a **faculty**, I should be able to log in to the website using my credentials so that I can access the website

As a **faculty**, I must be able to add or remove events from the list so that students can avail OD for those events

As a **student**, I must be able to select events from the list and attach proof so that the faculty can verify and grant me od

As a **student**, I must be able to track my OD so that I can check the progress of my OD request

As a **faculty**, I must be able to view the students' OD request with proof so that I can verify and provide OD for the students

As a **faculty**, I must be able to send an od letter as a pdf to the students so that they can use it for proof to other faculty members

As a **student,** I must be able to view if my OD request is approved or denied so that I can take appropriate action

As a **student**, I should receive notifications about the status of my OD request so that I am informed of any updates.

Poker Planning Estimation For User stories:

Student login: 3 story points

Password reset: 5 story points

Faculty login: 3 story points

Add/remove events: 8 story points

Select events and attach proof: 5 story points

Track OD request: 5 story points

View OD requests with proof: 5 story points

Send OD letter as PDF: 8 story points

View OD request status: 3 story points

Non Functional Requiremnts:

Performance:

The system should respond to user requests within 2 seconds for 95% of the requests.

Usability:

User Interface: The system should have an intuitive and user-friendly interface, with clear navigation and helpful tooltips.

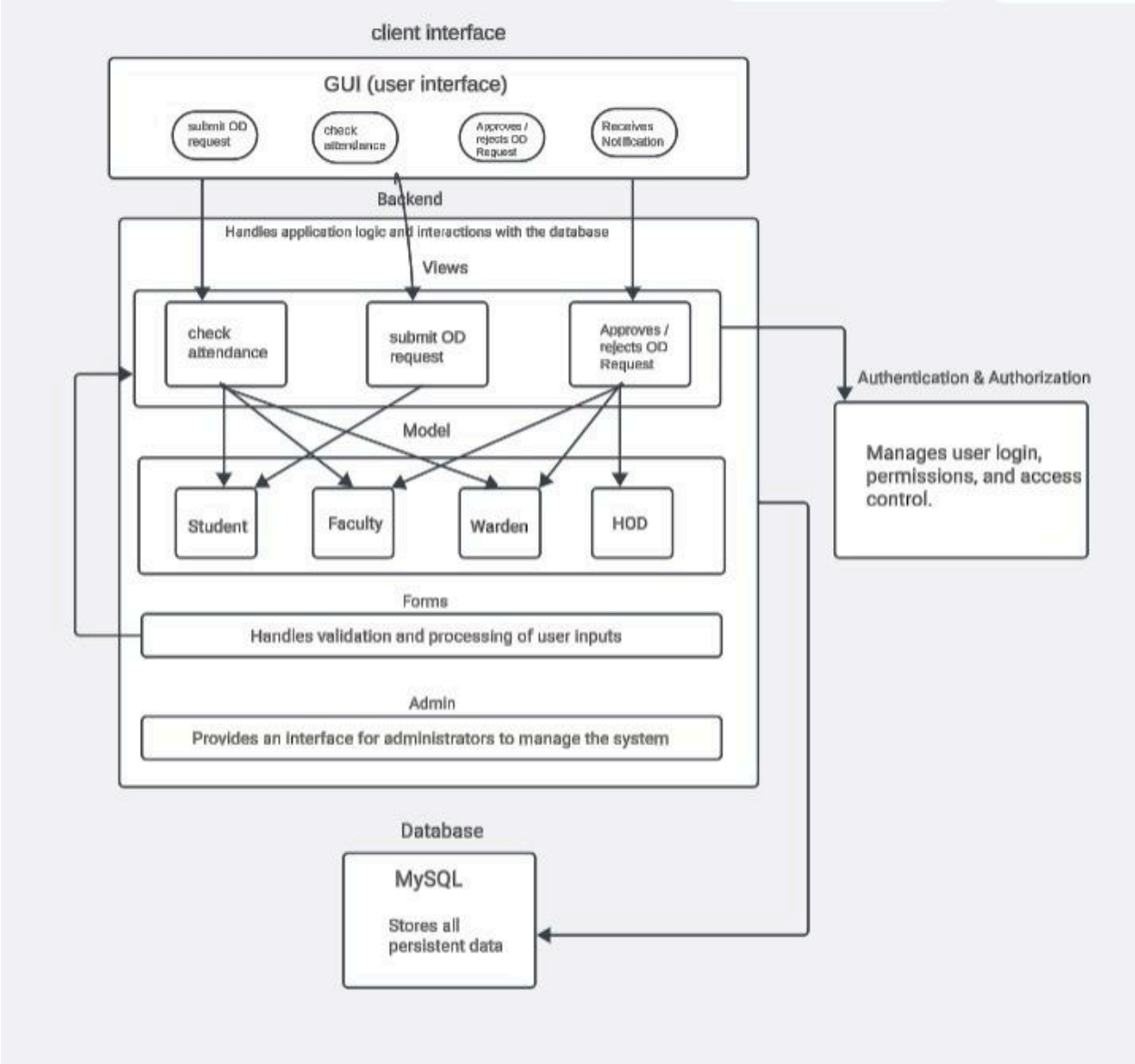
Reliability:

Error Handling: The system should handle errors gracefully and provide meaningful error messages to users.

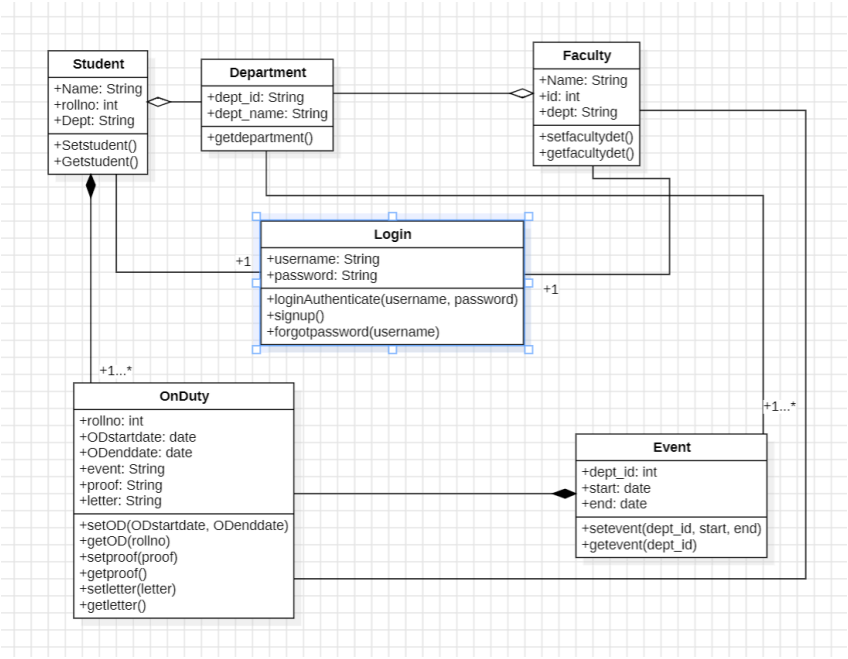
Security:

Data Encryption: All sensitive data, such as login credentials and OD request details, should be encrypted both in transit and at rest.

ARCHITECTURE DIAGRAM

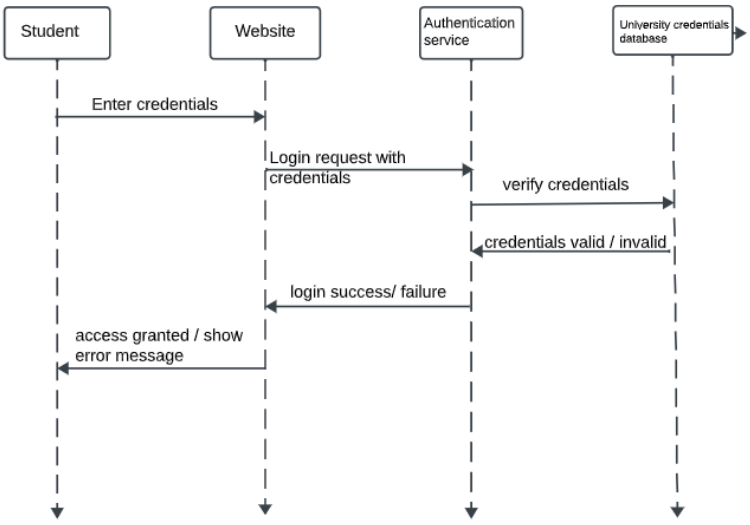


Class Diagram

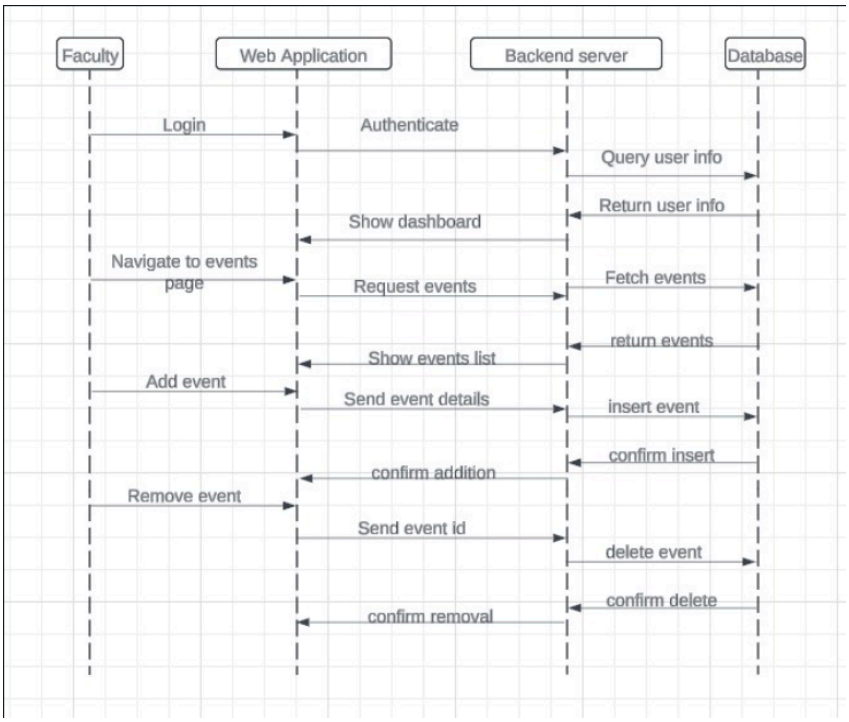


Sequence Diagram

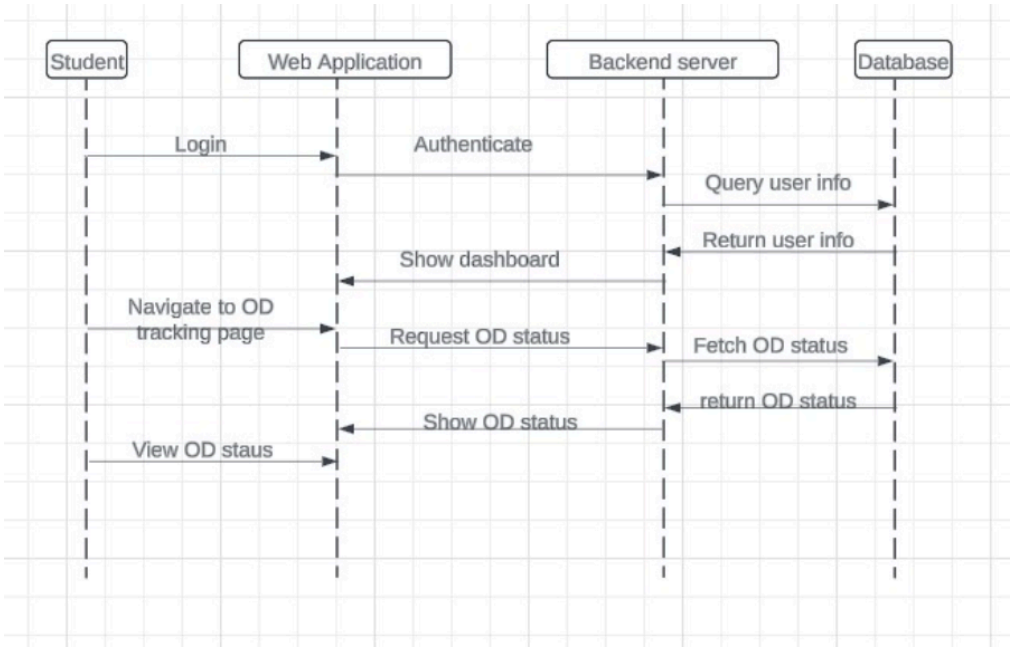
1)User story:As a student, I should be able to log in to the website using my university credentials so that I can access the website



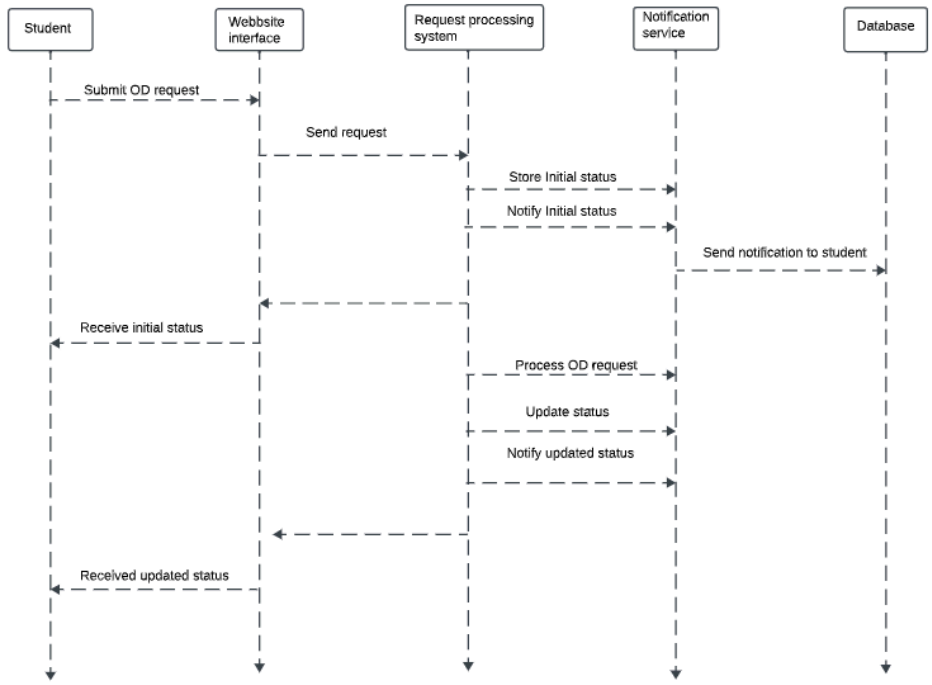
2) User story :As a Faculty, I must be able to add or remove events from the list so that students can avail for OD for those events



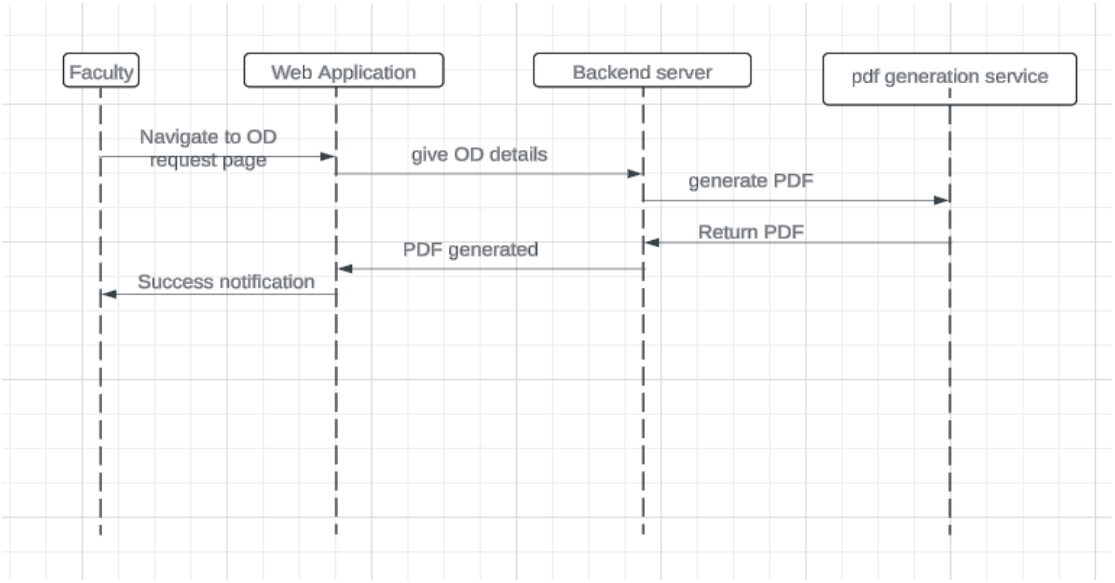
3)User story:As a student, I must be able to track my OD so that I can check the progress of my OD request



4)User story:As a student, I should receive notifications about the status of my OD request so that I am informed of any updates.



5)User story:As a faculty, I must be able to send an od letter as a pdf to the students so that they can use it for proof to other faculty members



TEST STRATEGY

1. Test Scope and Objectives:

Define the scope of testing, including functional areas (e.g., duty scheduling, assignment, reporting).

Specify the objectives: validate system functionality, ensure data accuracy, and verify compliance with business rules.

2. Test Levels:

Unit Testing: Validate individual components (e.g., On duty assignment algorithms,event management modules).

Integration Testing: Verify interactions between modules (e.g., shift handovers,).

System Testing: Validate end-to-end scenarios (e.g., duty creation, assignment, and reporting).

3. Test Data Strategy:

Generate synthetic data for duties, shifts, personnel, and roles.
Cover various scenarios (e.g., overlapping shifts, different duty types).

4. Test Environment:

Set up test environments mirroring production (e.g., databases, servers).
Ensure data privacy and security, especially when handling sensitive information.

5. Functional Testing:

Test the following scenarios:

- Duty creation and assignment.
- Shift handovers and continuity.
- Duty swaps or replacements.
- Reporting and analytics.
- Compliance with labor laws (e.g., maximum working hours).
- User roles (e.g., admin, supervisor, staff).

6. Non-Functional Testing:

Performance Testing: Evaluate system responsiveness under varying loads.

Security Testing: Verify access controls, data encryption, and user authentication.

Usability Testing: Assess user-friendliness and navigation.

7.Regression Testing:

Ensure that new features or bug fixes do not impact existing functionality.

SAMPLE TEST CASES:

1. User Story: Create a New Duty:

Happy Path:

- Verify that a new duty can be created with valid details (date, time, location).
- Confirm that the duty appears in the system after creation.

Error Scenarios:

- Attempt to create a duty with missing or invalid information (e.g., blank date).
- Check error messages for proper validation.

2. User Story: Assign Duty to Staff:

Happy Path:

- Assign a duty to a staff member.
- Validate that the staff member's schedule reflects the assigned duty.

Error Scenarios:
Try to assign a duty to an unavailable staff member (already assigned elsewhere).
Verify error handling for conflicting assignments.

3. User Story: Generate Duty Reports:
Happy Path:
Generate duty reports for a specific date range.
Ensure accurate representation of assigned duties.
Error Scenarios:
Attempt to generate a report with invalid date ranges.
Check for missing or incorrect data in the report.

4. User Story: Swap Duties:
Happy Path:
Allow staff members to request duty swaps.
Validate that swapped duties are reflected correctly.
Error Scenarios:
Verify handling of invalid swap requests (e.g., requesting a swap with an incompatible duty).

5. User Story: Compliance with Working Hours:
Happy Path:
Assign duties while adhering to maximum working hours per day/week.
Confirm that the system prevents overloading staff.
Error Scenarios:
Assign duties that exceed legal working hour limits.
Check for warnings or violations.

DEPLOYMENT ARCHITECTURE