RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

CS19442 SOFTWARE ENGINEERING CONCEPTS LAB

Laboratory Record
Note Book

Name :……SANJAY.S……. ……...

Year / Branch / Section : ……..II/CSE/D …….

University Register No. : ……2116220701248…....……

College Roll No. : ……...220701248..……

Semester : ……IV…….……

Academic Year : ……2023-2024…….

RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

BONAFIDE CERTIFICATE

Name: SANJAY.S

Academic Year: 2023-2024 Semester: IV Branch: CSE

Register No: 2116220701248

Certified that this is the bonafide record of work done by the above student in the CS19442-Software Engineering Concepts Laboratory during the year 2023- 2024

Signature of Faculty-in-charge

Submitted for the Practical Examination held on ----------

Internal Examiner External Examiner

HOSTEL
MANAGEMENT SYSTEM

INDEX

EXP NO:1

OVERVIEW

Problem Being Addressed:

The hostel management system is designed to address various challenges faced in managing hostel facilities within a college campus. These challenges include:

Manual Processes:

Traditional methods of managing hostel related tasks, such as room allocation, attendance tracking, and complaint handling, are often manual and time-consuming.

Lack of Transparency:

The absence of a centralized system leads to a lack of transparency in hostel operations, making it difficult for students to access information and lodge complaints effectively.

Inefficient Communication:

Communication between students and hostel authorities regarding issues such as room maintenance, food quality, and out pass requests is often inefficient and prone to delays.

Solution Overview:

The hostel management system provides a comprehensive solution to these challenges by offering the following features:

User Module:

Login:

Students can log in to the system using their credentials to access various functionalities.

Attendance Monitoring:

Students can view their attendance records, providing transparency and accountability in attendance tracking.

Complaint Management:

Students can raise complaints about food quality, room maintenance, or other hostel-related issues through the system, ensuring timely resolution.

Room Booking:

Students can request room bookings through the website, simplifying the process of securing accommodation within the hostel.

Outpass Application:

Students can apply for out passes through the system, streamlining the process of seeking permission for leaving the college premises.

Warden Module:

Complaint Resolution:

Wardens can view and address complaints raised by students through the system, ensuring efficient resolution of hostel-related issues.

Out pass Verification:

Wardens can verify and approve out pass requests submitted by students, maintaining control over student movements.

Attendance Management:

Wardens can monitor and update student attendance records through the system, facilitating accurate attendance tracking.

Room Booking Approval:

Wardens can review and approve room booking requests, ensuring proper allocation of hostel rooms based on availability and eligibility criteria.

User Benefits:

Convenience:

The system offers a user-friendly interface for students and wardens, making hostel-related tasks more convenient and accessible.

Transparency:

By centralizing hostel operations, the system promotes transparency and accountability, enhancing trust between students and hostel authorities.

Efficiency:

Automation of various processes reduces the administrative burden on hostel staff and ensures timely resolution of issues.

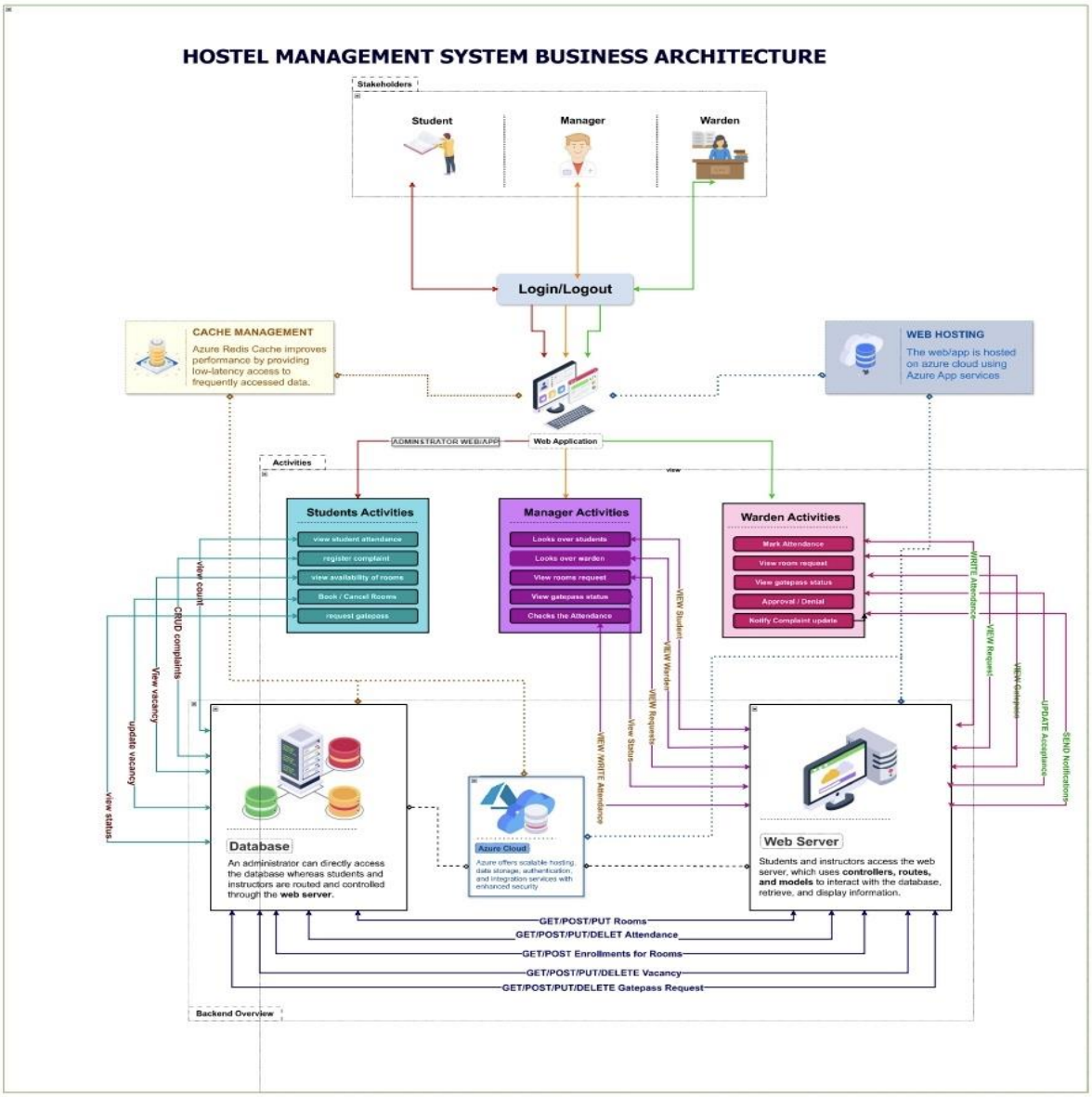
Improved Communication:

The system facilitates seamless communication between students and hostel authorities, fostering better coordination and collaboration.

Overall, the hostel management system significantly improves the hostel experience for students and enhances the efficiency of hostel operations, leading to a more streamlined and organized management process within the college campus.

EX NO:2

BUISNESS ARCHITECTURE



The diagram depicts the business architecture of a Hostel Management System.

1. Users: Students, Managers, and Wardens, who access the system via login/logout functionality.

2. **Activities:**

Each user type has specific activities:

Students: View details, lodge complaints, view rooms, check financial forms, and print gate passes.

Managers: Look at new students, view requests, approve requests, and check attendance.

Wardens: Mark attendance, view requests, approve/deny requests, and notify about complaints.

3. **Cache Management***: Auto Redis Cache improves performance by storing frequently accessed data.*

4. Web Hosting*: The application is hosted on Azure App Services.*

5. Database*: Stores student and hostel-related data accessible through the web server.*

6. Web Server*: Manages communication between users, database, and other components using various HTTP methods.*

7.APIs*:Used for CRUD operations on rooms, attendance, vacancies, and gate pass requests.*

8. Backend Overview*:Illustrates the flow and interaction between the web server and the database.*

***9. Technologies**: Depicts usage of web hosting, cache management, database, and API communication.*

*10. **Overall Function** :Ensures efficient management and operation of hostel activities via a web application.*

EX NO:3

FUNCTIONAL AND NON FUNCTIONAL REQUIREMENT

1.Functional Requirements User Stories

Room Allocation

- As a student, I want to view available room categories (AC, Non-AC) with pictures so that I can apply for hostel accommodation.
- As a warden, I want to review room requests, check room availability, and allocate rooms to students so that I can manage hostel occupancy efficiently.

Attendance Management

- As a warden, I want to mark student attendance by selecting rooms and marking students as present (P) so that I can maintain accurate attendance records.
- As a student, I want to view my attendance history so that I can track my attendance record.

Complaint Submission and Management

- As a student, I want to register complaints about hostel facilities so that I can report issues for resolution.
- As a student, I want to view all complaints publicly with their status (unresolved or resolved) so that I can stay informed about ongoing issues.

Complaint Resolution Verification

- As a student, I want to verify the resolution of my complaints by accepting or rejecting the status marked by the warden so that I can ensure my issue has been properly resolved.

Notifications

- ***1**.As a student, I want to receive notifications about the status of my room requests, complaints, and gate pass requests so that I stay informed about my submissions.*
- *2.As a warden, I want to receive notifications about new room requests, complaints, and other relevant updates so that I can take timely action.*

Gate Pass Management

- As a student, I want to apply for gate passes for outings during holidays so that I can get permission to leave the hostel.

- As a warden, I want to review and approve or reject gate pass requests so that I can manage student outings

User Roles and Permissions

- As a student, I want access to features relevant to me so that I can interact with the system appropriately.

- As a warden, I want access to features relevant to me so that I can manage student activities.
- As a hostel manager, I want to view and manage warden actions and unresolved issues so that I
- can oversee hostel operations effectively.

Data Management

- As a warden, I want to update and manage student profiles and their respective data so that I can keep records accurate and up-to-date.

Frontend Interface

- As a student, I want a responsive user interface developed using Bootstrap and JavaScript so that I can interact with the system easily.
- As a warden, I want a responsive user interface developed using Bootstrap and JavaScript so that I can manage my tasks efficiently.

2.Non-Functional Requirements

Performance

- As a user, I want the system to support up to 1000 concurrent users without performance degradation so that I can use the system reliably during peak times.

- As a user, I want pages to load within 2 seconds under normal load conditions so that my interaction with the system is smooth.

Security

- As a user, I want sensitive data such as passwords and personal information to be encrypted so that my data is protected.
- As a user, I want secure login mechanisms, including email and password verification, so that my account remains secure.

Reliability

- As a user, I want the system to have 99.9% uptime so that it is continuously available.

Usability

- As a user, I want an intuitive interface designed for ease of use so that I require minimal training to interact with the system.
- As a warden, I want a dashboard view summarizing key information such as attendance and complaints so that I can quickly access important data.

Compliance

- As a user, I want the system to comply with relevant data protection regulations so that my data is handled legally.
- As an admin, I want audit trails for actions taken within the system so that accountability is ensured.

Scalability

- As an admin, I want the system to handle increased numbers of users and additional hostels in the future without significant redesign so that it can grow with the institution.

Maintainability

- As a developer, I want the system to be developed using modular components so that updates and maintenance are easier to perform.

Accessibility

- As a user, I want the system to be accessible on various devices, including smartphones, tablets, and desktops so that I can use it conveniently from anywhere.

Data Integrity

- As a user, I want the system to ensure data accuracy and consistency across all modules so that I can trust the information provided.

User Experience

- As a user, I want a seamless and responsive user experience so that my interactions with the system are smooth and feedback is timely.

EXP NO:4

ARCHITECTURAL PATTERN

MVC ARCHITECTURE

Certainly! Let's implement the Model-View-Controller (MVC) architecture for the hostel management system:

1. Model:

User Model:

Manages data related to student accounts, including login credentials, attendance records, complaints, room bookings, and out pass requests.

Warden Model:

Handles data concerning warden accounts, such as their roles, responsibilities, and actions taken on student requests and complaints.

Database Access Layer:

Provides methods for interacting with the database, including querying, updating, and deleting records.

2. View:

User Interface (UI):

Represents the user-facing components of the system, including web pages or interfaces for student login, attendance viewing, complaint submission, room booking, and outpass application.

Warden Dashboard:

Displays relevant information and functionalities for wardens, such as complaint resolution, out pass verification, attendance management, and room booking approval.

3. Controller:

User Controller:

Handles user requests and actions initiated by students, such as logging in, viewing attendance, submitting complaints, booking rooms, and applying for out passes.

Warden Controller:

Manages actions and requests initiated by wardens, including resolving complaints, verifying out passes, updating attendance records, and approving room bookings.

Routing Layer:

Maps incoming requests from users and wardens to appropriate controller actions based on URL routes.

4.Interactions:

- When a user or warden interacts with the system through the UI, the corresponding controller receives the request.
- The controller processes the request, invokes necessary operations on the model layer (e.g., retrieving data from or updating data in the database), and prepares the response.
- The response is then passed to the appropriate view, which generates the corresponding output (e.g., rendering HTML pages or JSON responses).
- Finally, the generated output is sent back to the user's browser or client device for display.

5.Error Handling:

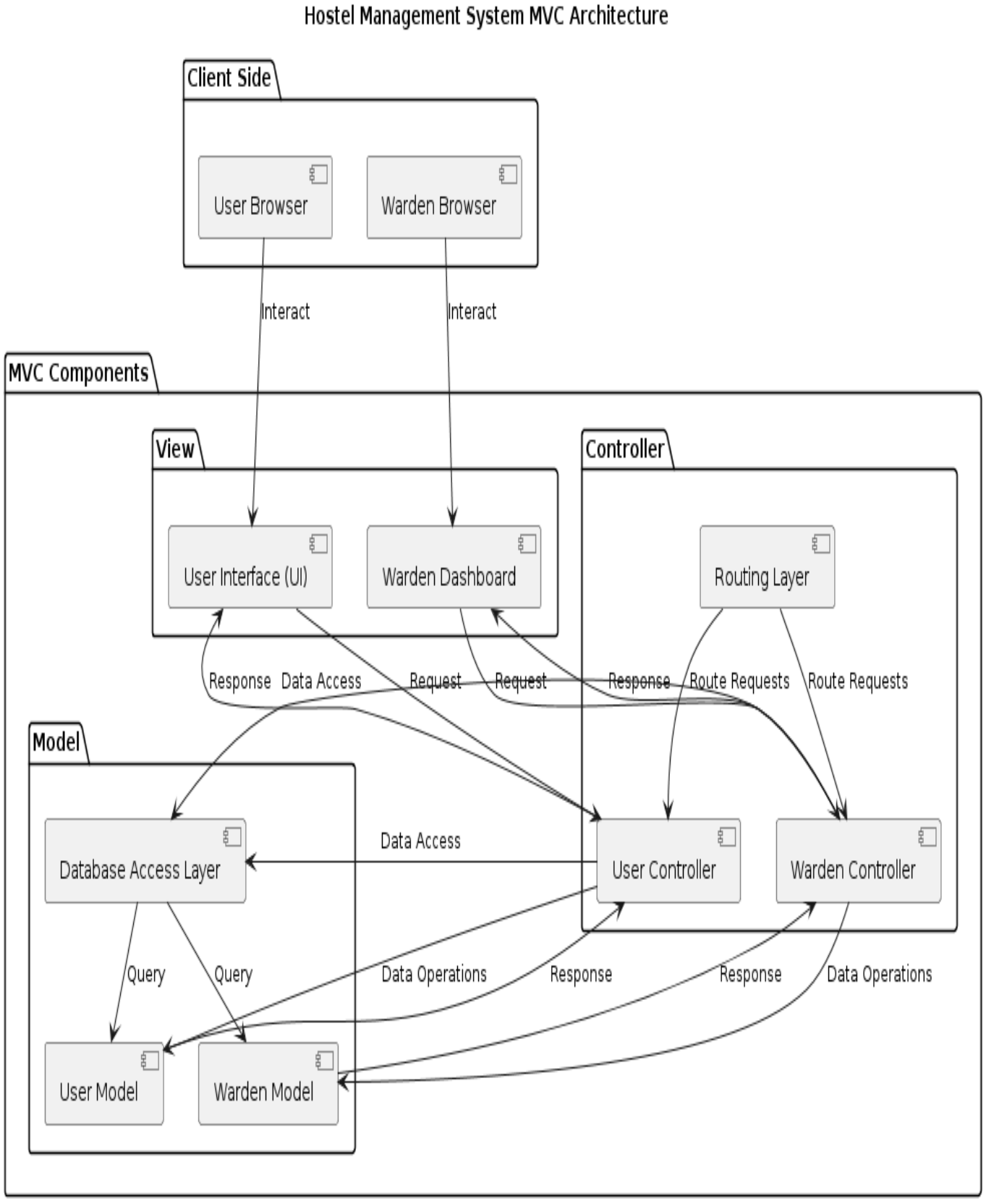
Each controller implements error handling mechanisms to detect and handle exceptions or errors that occur during request processing. - Error messages are displayed to users in a user-friendly format, providing guidance on how to resolve issues or proceed with the request.

Logging:

- The system incorporates logging mechanisms within controllers and model components to record significant events, errors, and transactions.
- Logs capture details such as user actions, system responses, database transactions, and error messages, facilitating auditing, monitoring, and debugging.

Data Storage:

The model layer interacts with the database access layer to store and retrieve data from the underlying database management system (e.g., MySQL, PostgreSQL)



EXP NO 5

DESIGN PRINCIPLES

3. Design Principles Used and Why:

. Design Principles:

Single Responsibility Principle (SRP):

Each module, class, or function is designed to have a single responsibility, promoting clarity, maintainability, and reusability.

Open/Closed Principle (OCP):

Modules are open for extension but closed for modification, allowing for the addition of new features or functionalities without altering existing code.

Dependency Inversion Principle (DIP):

High-level modules depend on abstractions, not concrete implementations, facilitating loose coupling and enabling easier integration of components.

Don't Repeat Yourself (DRY):

Common functionalities are encapsulated into reusable components, reducing redundancy and ensuring consistency throughout the system.

Reasons for Using Design Principles

Modularity:

Design principles help in breaking down the system into smaller, manageable components, making it easier to develop, test, and maintain.

Flexibility:

By adhering to design principles, the system becomes more adaptable to changes in requirements, technology, and business rules.

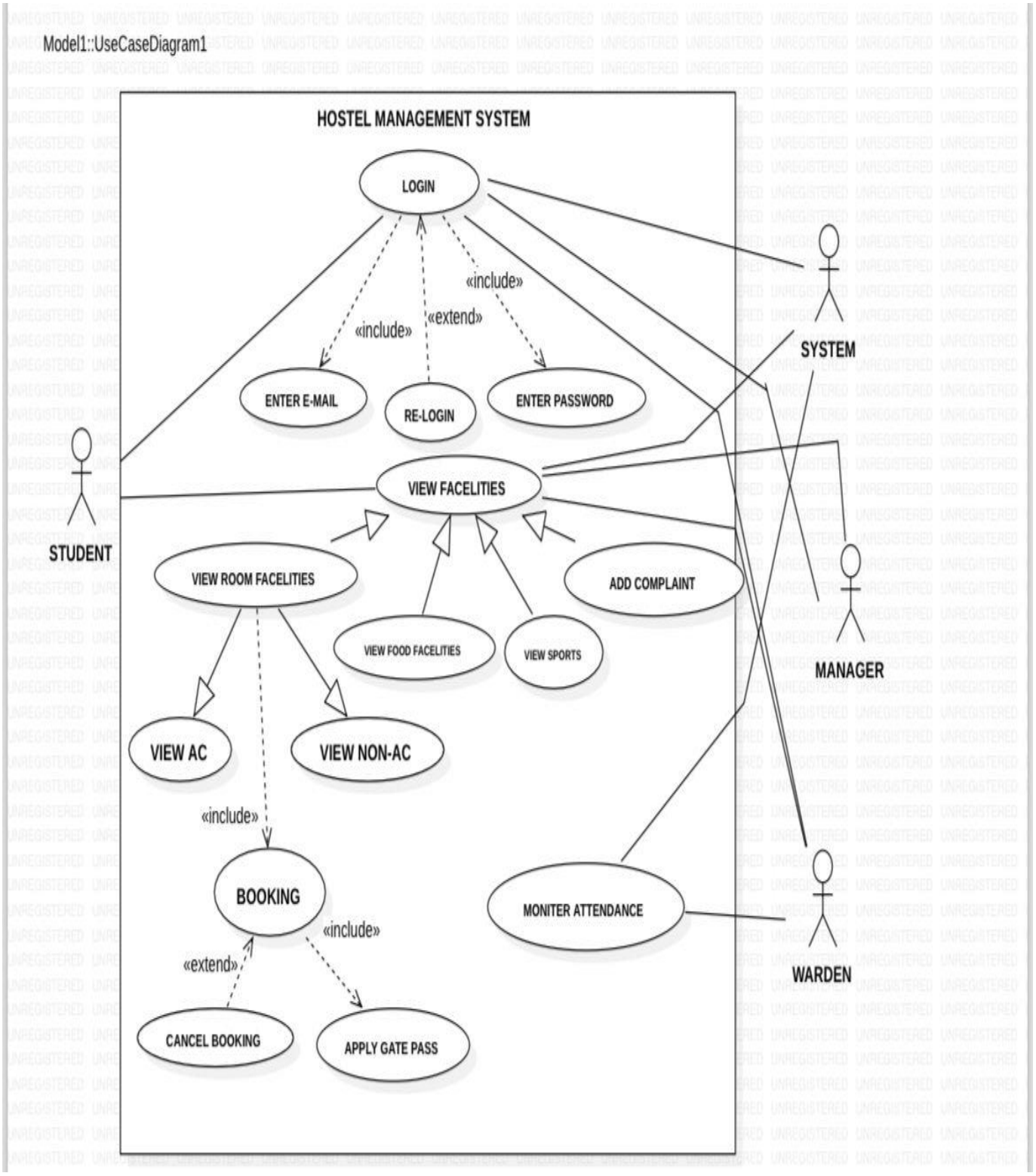
Readability and Maintainability:

Following design principles enhances code readability and maintainability, making it easier for developers to understand, modify, and extend the system over time.

By incorporating these elements into the design and architecture of the hostel management system, it becomes more robust, scalable, and maintainable, meeting the requirements

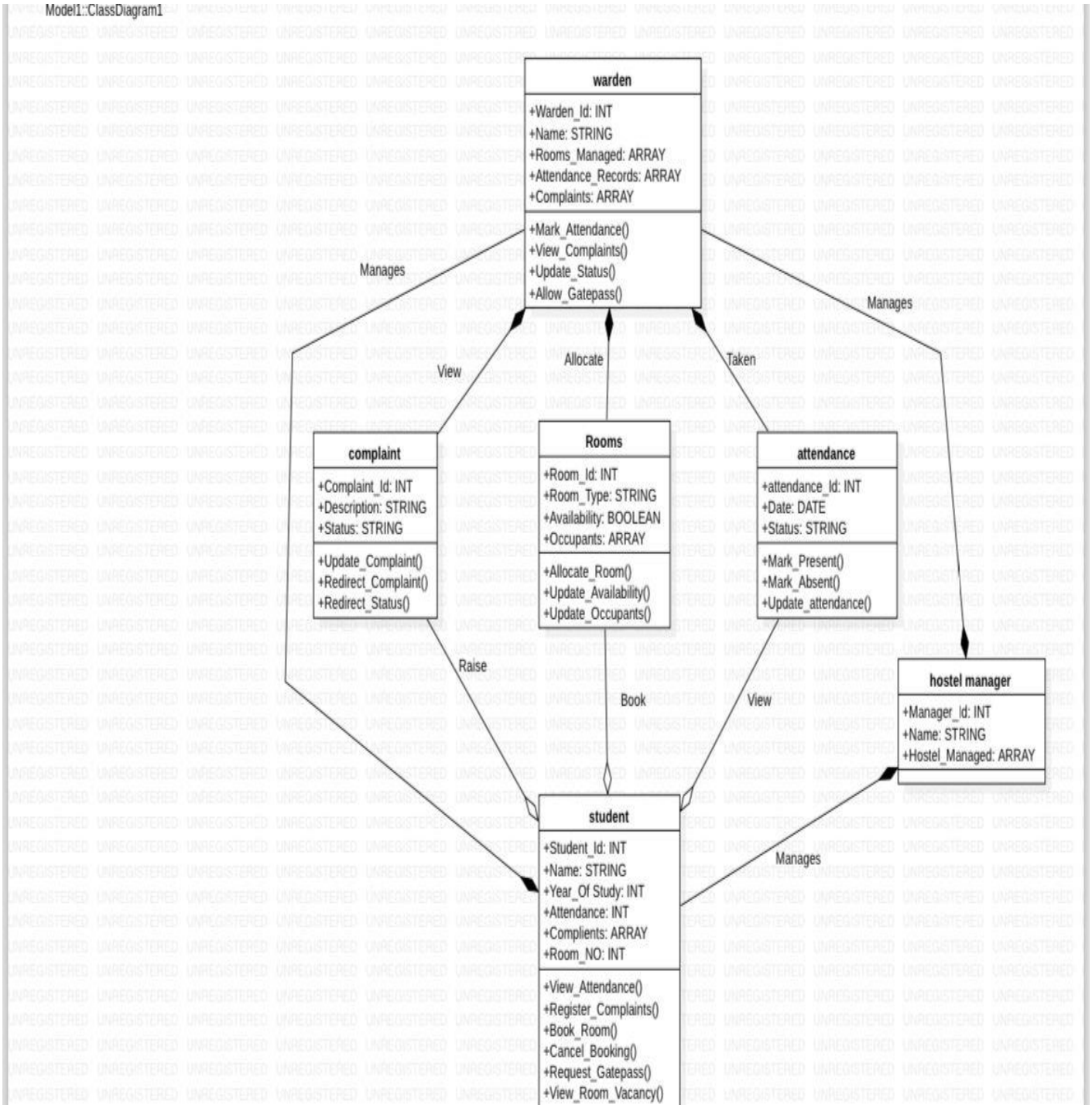
EXP NO:6

USE CASE DIAGRAM



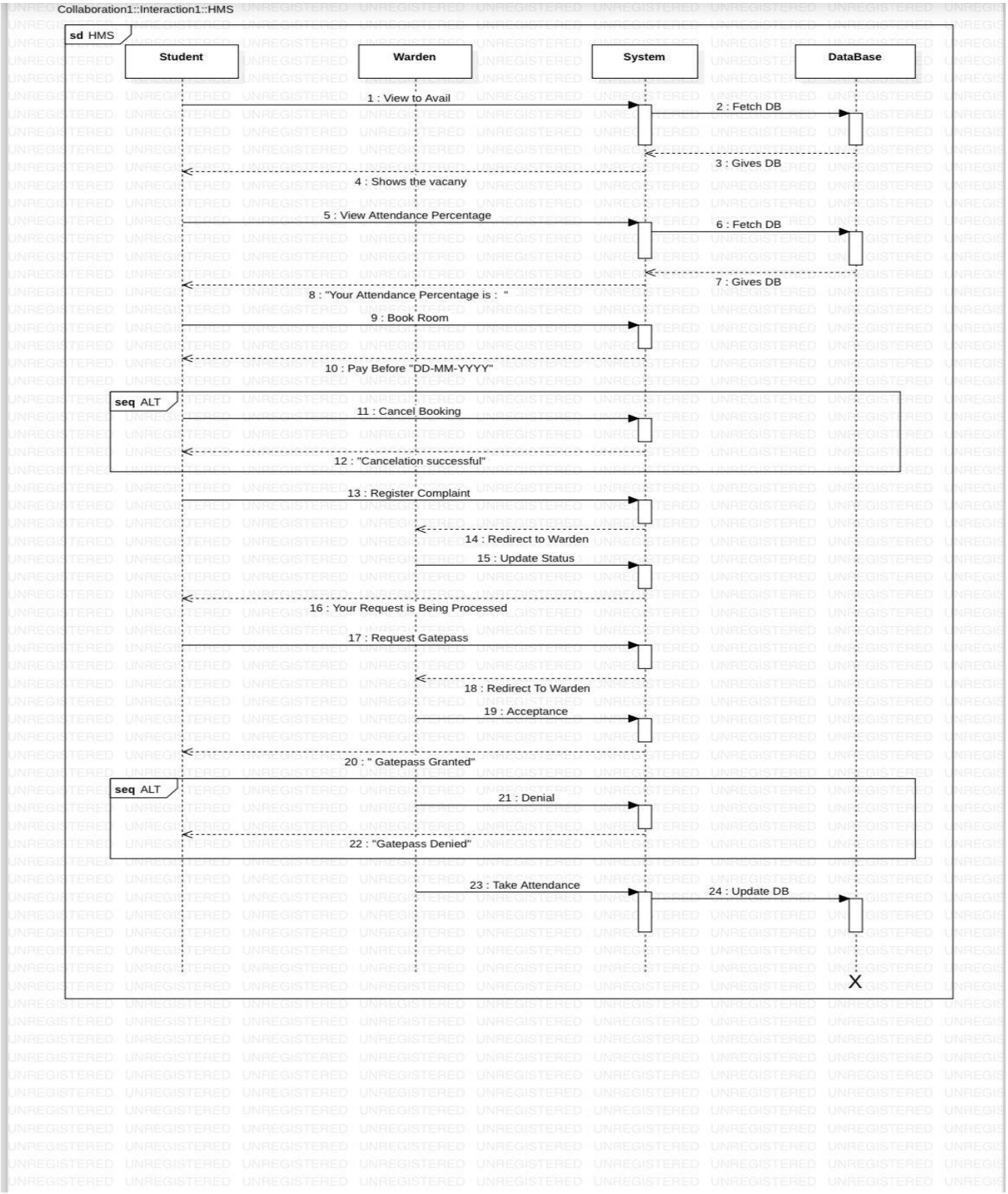
EXP NO:7

CLASS DIAGRAM



EXP NO 8

SEQUENCE DIAGRAM



EXP NO :9

TEST CASES

Certainly! Here are four test cases based on the provided user stories:

Test Case 1: Viewing Available Room Categories

Test Case ID: TC001

Test Case Description: As a student, I navigate to the room allocation section and view available room categories (AC, Non-AC) with pictures.

Actual Result: Both AC and Non-AC room categories are displayed with corresponding images.

Expected Result: The student interface accurately shows the available room categories along with their respective images.

Test Case 2: Marking Student Attendance

Test Case ID: TC002

Test Case Description: As a warden, I mark student attendance by selecting rooms and marking students as present (P).

Actual Result: I select rooms and mark students as present; the attendance records are updated accordingly.

Expected Result: The attendance management system correctly records the attendance of students marked as present by the warden.

Test Case 3: Complaint Submission and Viewing

Test Case ID: TC003

Test Case Description: As a student, I register a complaint about hostel facilities and then view all complaints publicly with their status (unresolved or resolved).

Actual Result: I submit a complaint and can view it along with other complaints, each showing their status.

Expected Result: The complaint submission system accepts my complaint and displays it publicly with its corresponding status, either unresolved or resolved.

Test Case 4: Gate Pass Request Approval

Test Case ID:TC004

Test Case Description: As a warden, I review and either approve or reject a gate pass request submitted by a student.

Actual Result: I review a gate pass request and either approve or reject it, which updates the status accordingly.

Expected Result: The gate pass management system accurately updates the status of the gate pass request based on the warden's decision, reflecting either approval or rejection.