SOFTWARE ENGINEERING

SOFTWARE DESIGN SPECIFICATION

GROUP MEMBERS-

- Anand Raj (IIT2022047)
- Rohan Phutke (IIT2022048)
- Avanish Gurjar (IIT2022049)
- Aman Raj (IIT2022050)
- Piyush Priyadarshi (IIT2022051)
- Saurav Gitte (IIT2022066)

TABLE OF CONTENT

The Software Design Specification

1) INTRODUCTION

The Software Design Document is a document intended to provide comprehensive documentation to guide software development by detailing how the software should be constructed. This document includes narrative and graphical representations of the software design, encompassing use case models, sequence diagrams, collaboration models, object behaviour models, and other pertinent requirement information.

1.1 Purpose of this document

This document serves to outline the design of the hostel management system. It provides specific details regarding expected inputs, outputs, classes, and functions. The interactions among these classes to fulfil the desired requirements are elucidated through detailed figures included at the conclusion of this document.

1.2 Scope of the development project

This section delineates the features included within the scope of the software to be developed, as well as those excluded from consideration.

In Scope:

a. Room allocation management

b. Help desk for student complaints

c. Student dashboard for information retrieval

Out of Scope:

a. Integration with external access control systems (e.g., RFID card readers)

b. Online payment processing for hostel fees

c. Integration with external transportation booking systems for students' travel arrangements

1.3 Definitions, acronyms, and abbreviations

IEEE: Institute of Electrical and Electronics Engineers

SDS: Software Design Specification

HMS: Hostel Management System

1.4 References

1.4.1 R. S. Pressman, Software Engineering: A Practitioner's Approach, 5th Ed, McGraw-Hill, 2001.

1.4.2 IEEE SDS template

1.5 Overview of document

This Software Design Document is structured into several sections with corresponding sub-sections, as follows:

1. **Introduction**: describes about the document, purpose, scope of development project definitions and abbreviations used in the document.

2. **Conceptual Architecture/Architecture Diagram**: describes the overview of components, modules, structure and relationships and user interface issues.

3. **Logical Architecture**: describes Logical Architecture Description and Components.

4. **Execution Architecture**: defines the runtime environment, processes, deployment view.

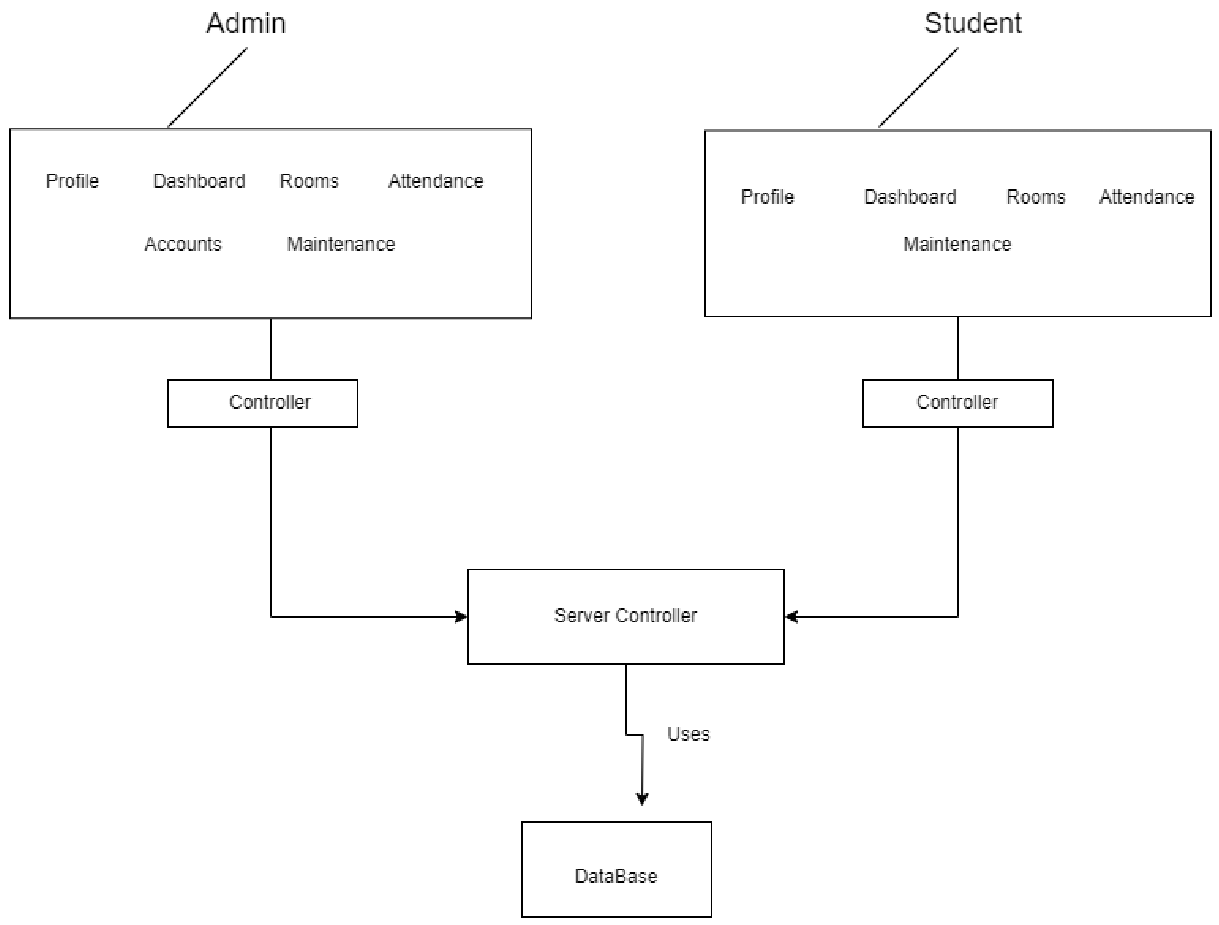
5. **Design Decisions and Trade-offs:** describes the decisions taken along with the reason as to why they were chosen over other alternatives.

6. **Pseudocode for components**: describes pseudocode, as the name indicates.

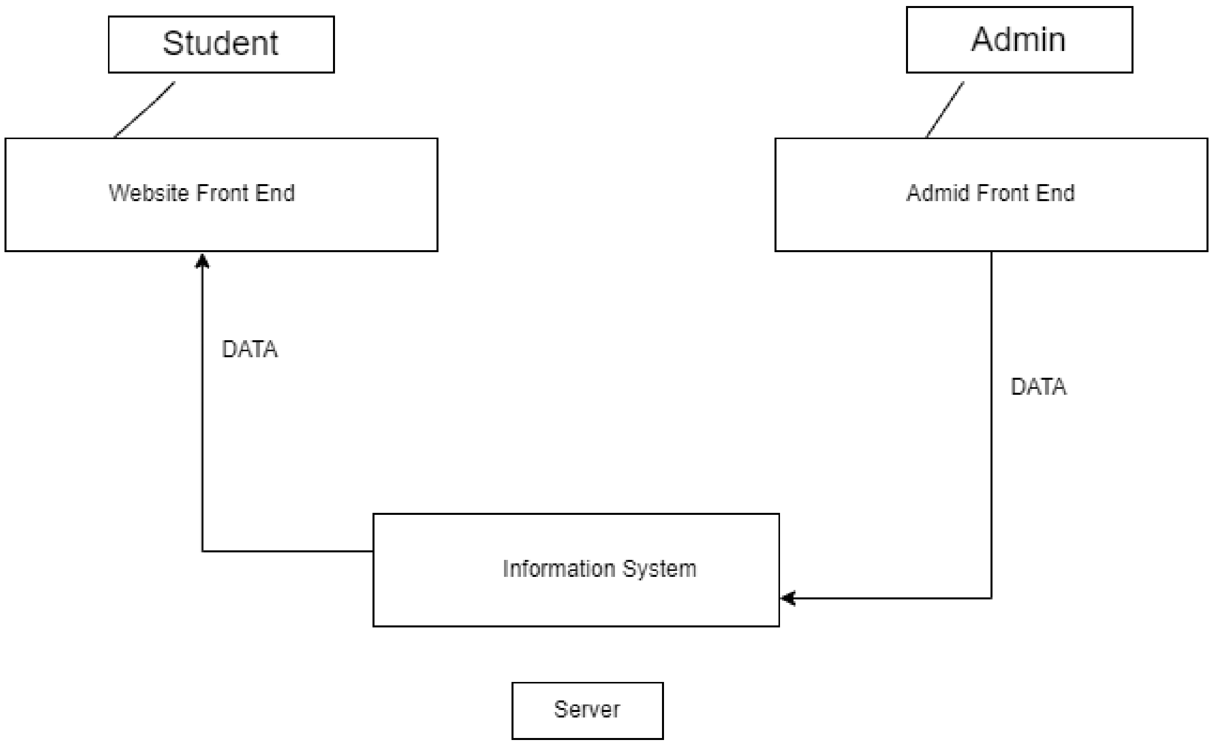
7. **Appendices**: describes subsidiary matter if any.

2. Conceptual Architecture/Architecture Diagram

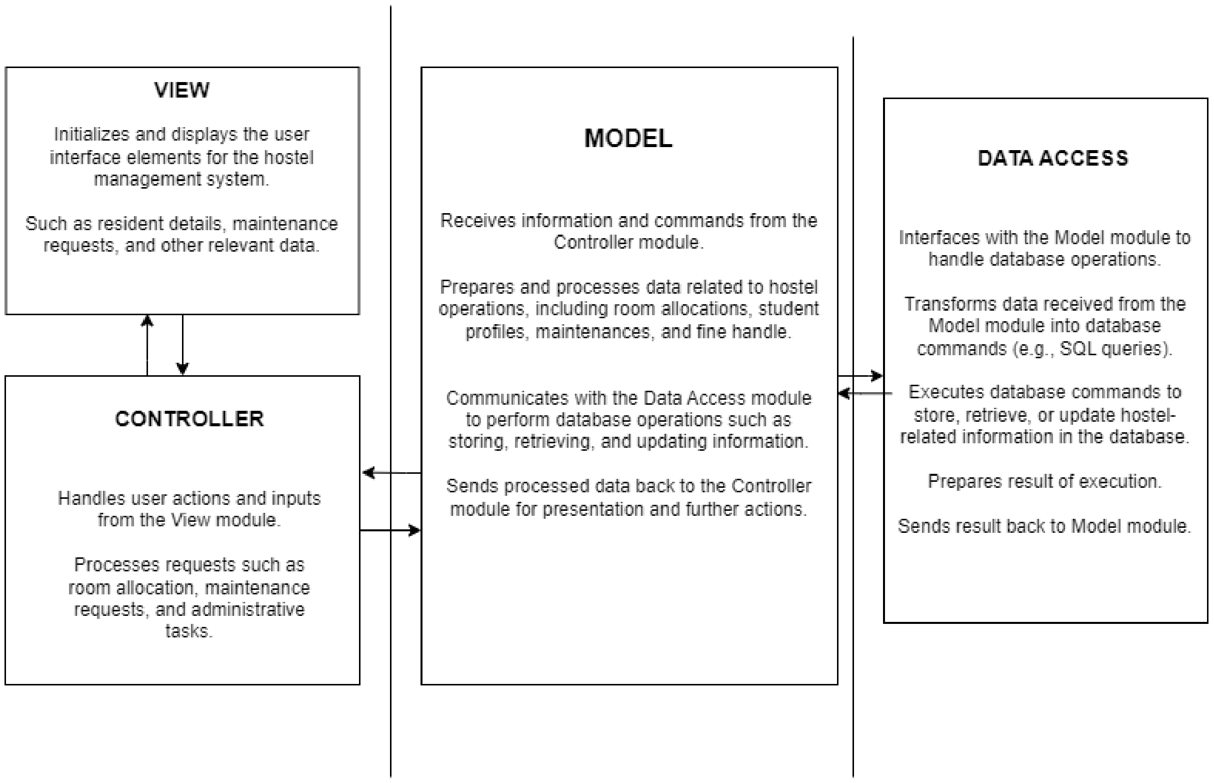
Architecture Diagram 1:



Architecture Diagram 2:

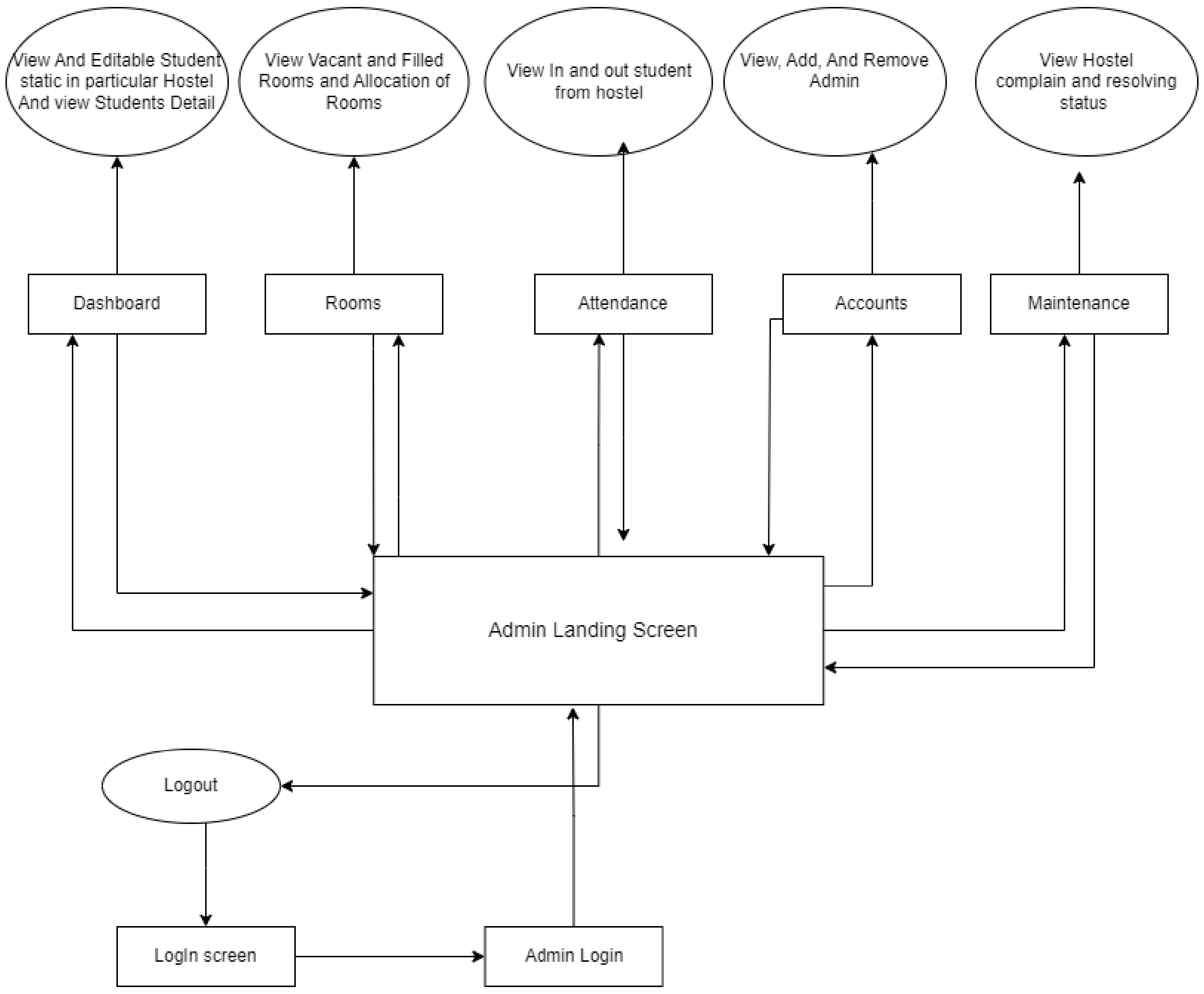


2.1 Overview of modules / components

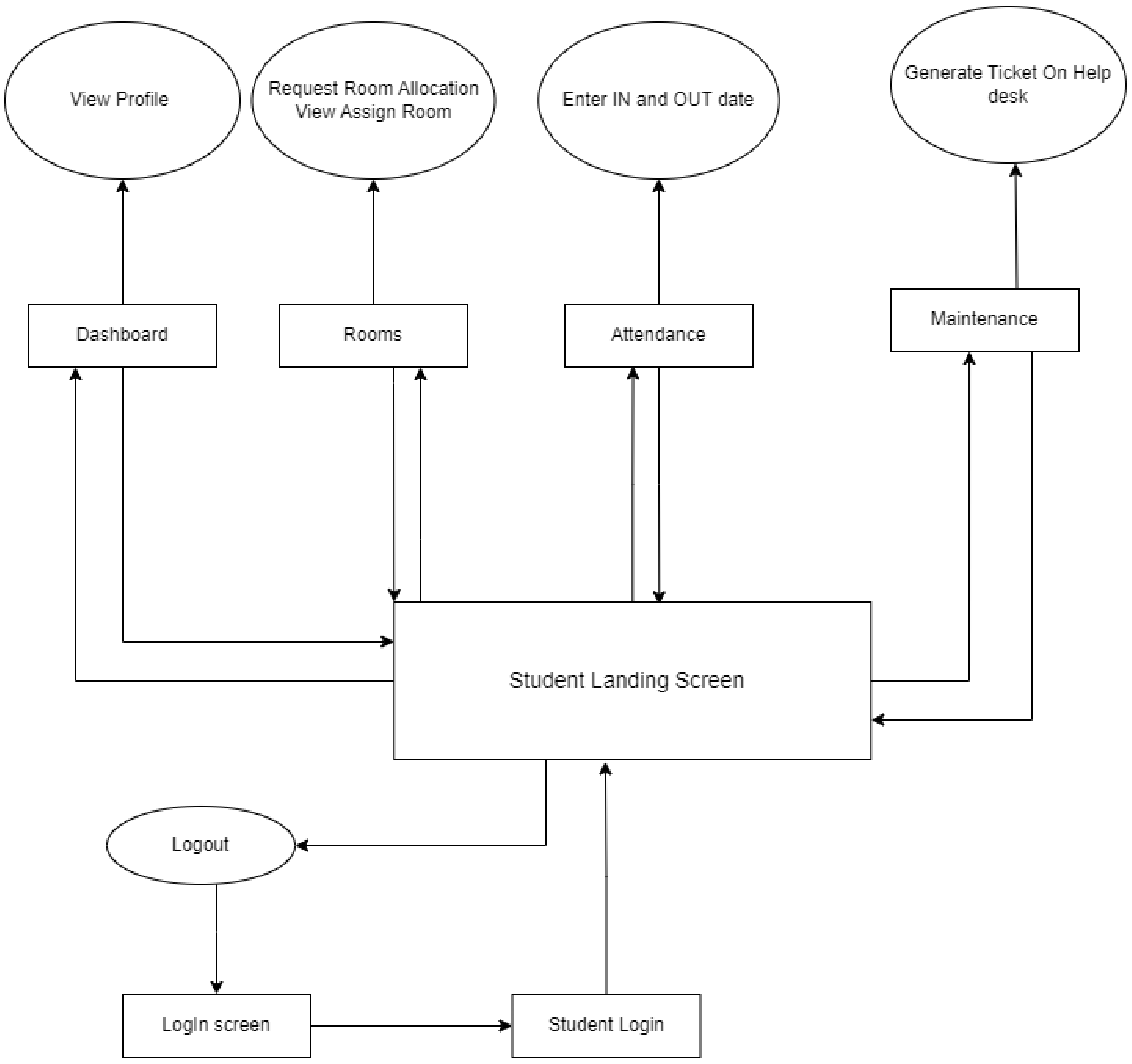

VIEW
Initializes and displays the user interface elements for the hostel management system.
Such as resident details, maintenance requests, and other relevant data.
CONTROLLER
Handles user actions and inputs from the View module.
Processes requests such as room allocation, maintenance requests, and administrative tasks.
MODEL
Receives information and commands from the Controller module.
Prepares and processes data related to hostel operations, including room allocations, student profiles, maintenances, and fine handle.
Communicates with the Data Access module to perform database operations such as storing, retrieving, and updating information.
Sends processed data back to the Controller module for presentation and further actions.
DATA ACCESS
Interfaces with the Model module to handle database operations.
Transforms data received from the Model module into database commands (e.g., SQL queries).
Executes database commands to store, retrieve, or update hostel-related information in the database.
Prepares result of execution.
Sends result back to Model module.

2.2 Structure and relationships

2.2.1 Admin's Side


View And Editable Student static in particular Hostel And view Students Detail
View Vacant and Filled Rooms and Allocation of Rooms
View In and out student from hostel
View, Add, And Remove Admin
View Hostel complain and resolving status
Dashboard
Rooms
Attendance
Accounts
Maintenance
Admin Landing Screen
Logout
LogIn screen
Admin Login

2.2.2 Student's Side



NOTE:

- The boxes represent individual screens.
- The circles represent actions that do not have screens.
- The arrows represent navigation between screens.

2.3 User interface issues

User interface considerations for a Hostel Management System would vary slightly from those of a Campus Recruitment System like Mieten. Here's how the user interface issues might be addressed for different types of users in the context of a Hostel Management System:

Student Residents

User Profile: A typical user of a Hostel Management System is a student residing in the hostel.

Interface Design: The interface should be intuitive and easy to navigate, following common design patterns seen in web applications. This includes clear menu options, easy-to-understand icons, and a clean layout.

Key Features: Students should be able to easily access features like room allocation, maintenance requests, IN-OUT system and fine options.

Communication: Integration with messaging systems can be useful for announcements, feedback, and resolving issues.

Hostel Management/Authorities:

User Profile: Higher-level authorities such as hostel managers, university administrators, or stakeholders overseeing multiple hostels.

Interface Design: The interface should provide a holistic view of hostel operations, allowing for data analysis, strategic planning, and decision-making.

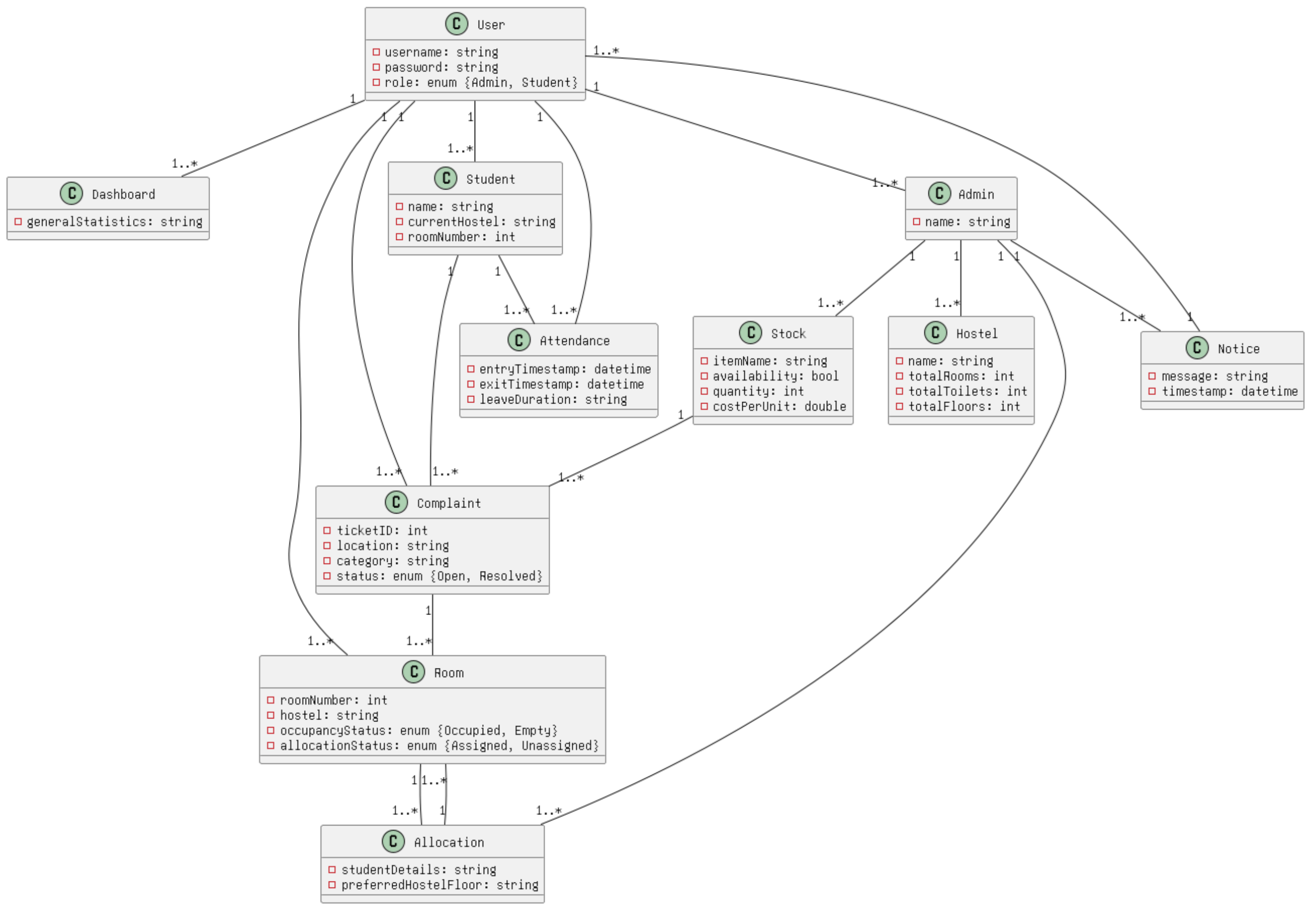
Task Management : Tools for managing room allocations, tracking maintenance tasks, generating reports, and communicating with residents should be readily accessible.

Analytics and Reporting: Tools for generating detailed reports and managing room allocations, tracking maintenance tasks, generating reports, and communicating with residents should be readily accessible.

Integration: Integration with university systems (e.g., student databases, financial systems) for seamless data exchange and coordination.

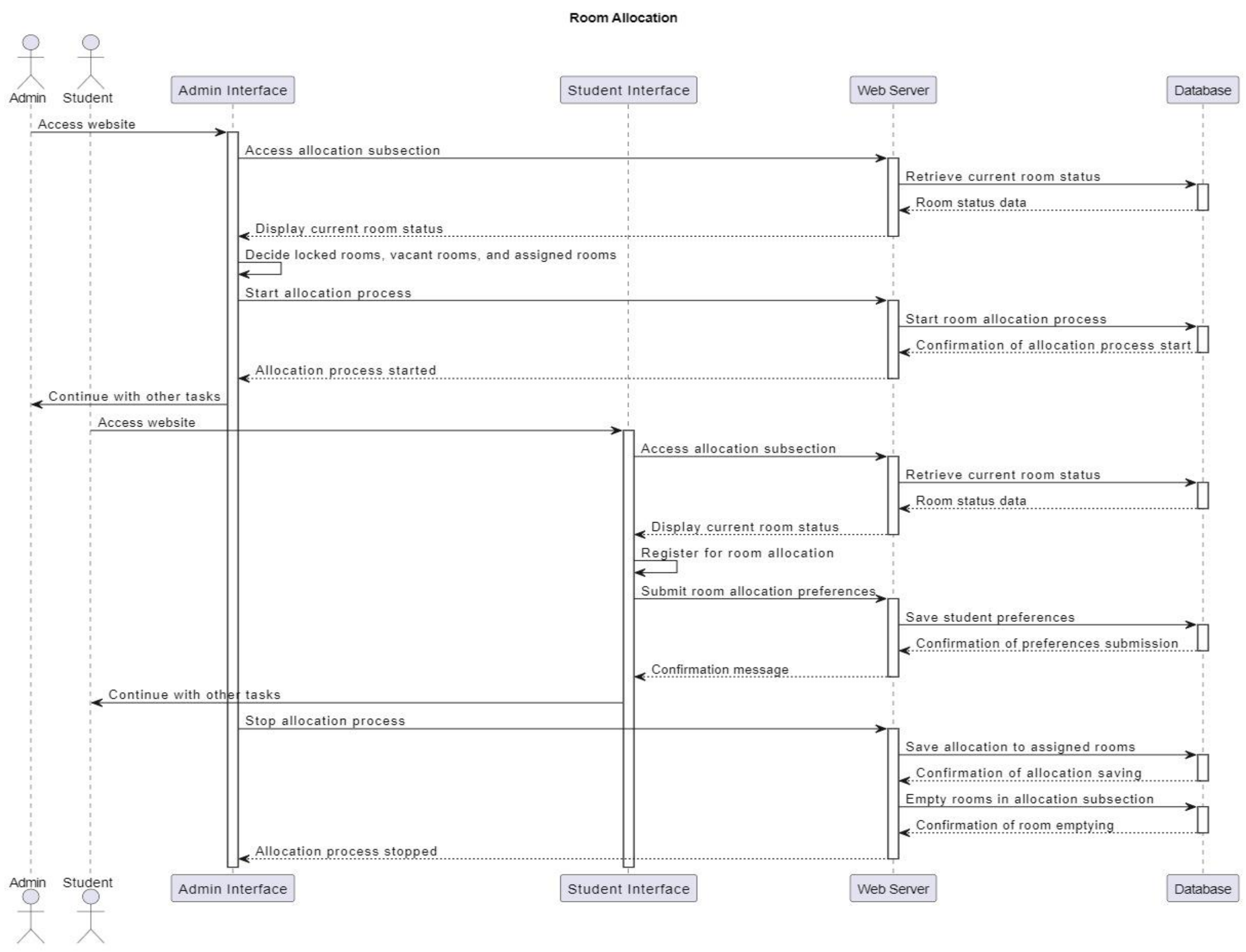
3. Logical Architecture (Class Diagram, Sequence Diagram, State Diagram)

Class Diagram:

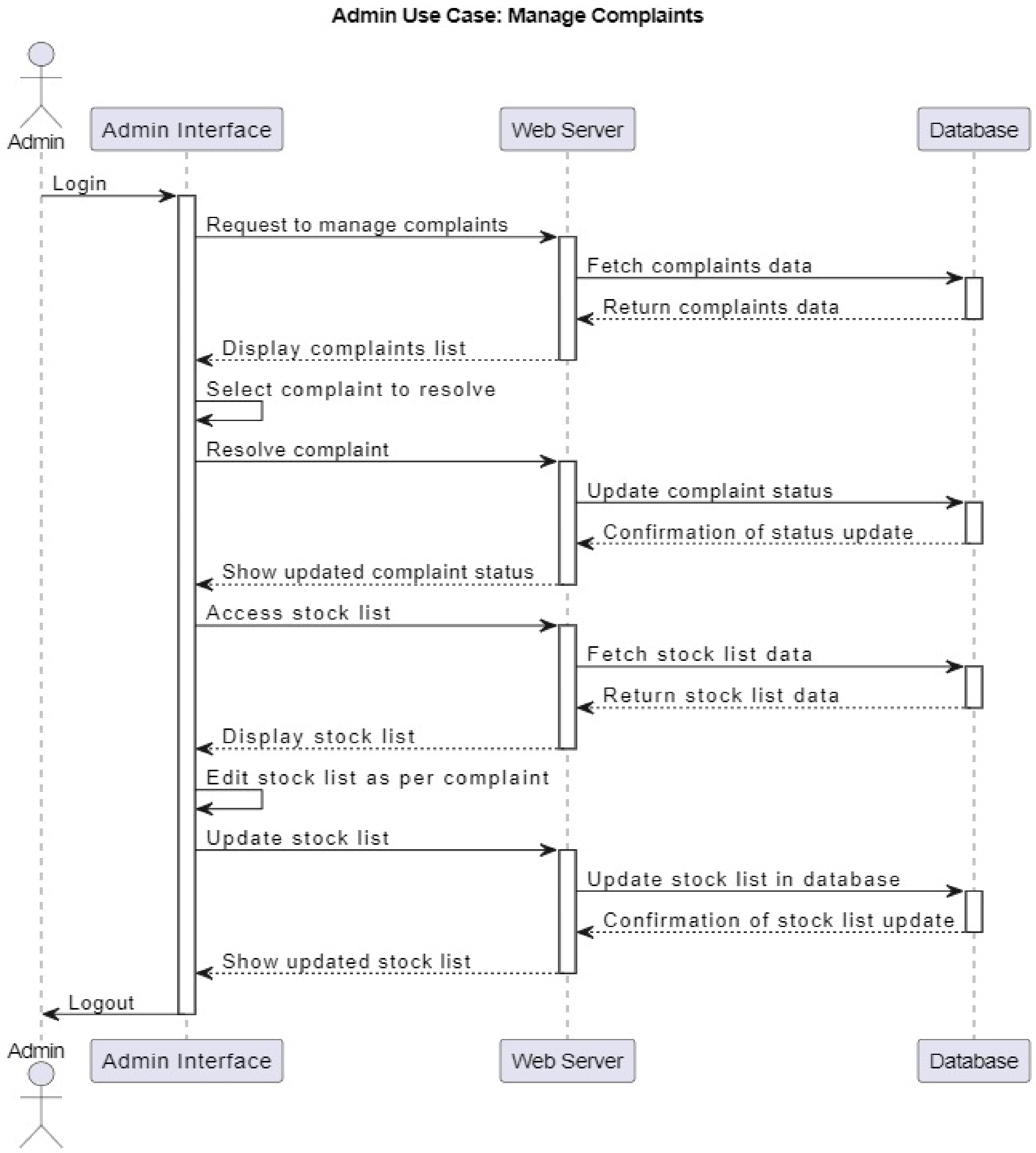


Sequence Diagrams:

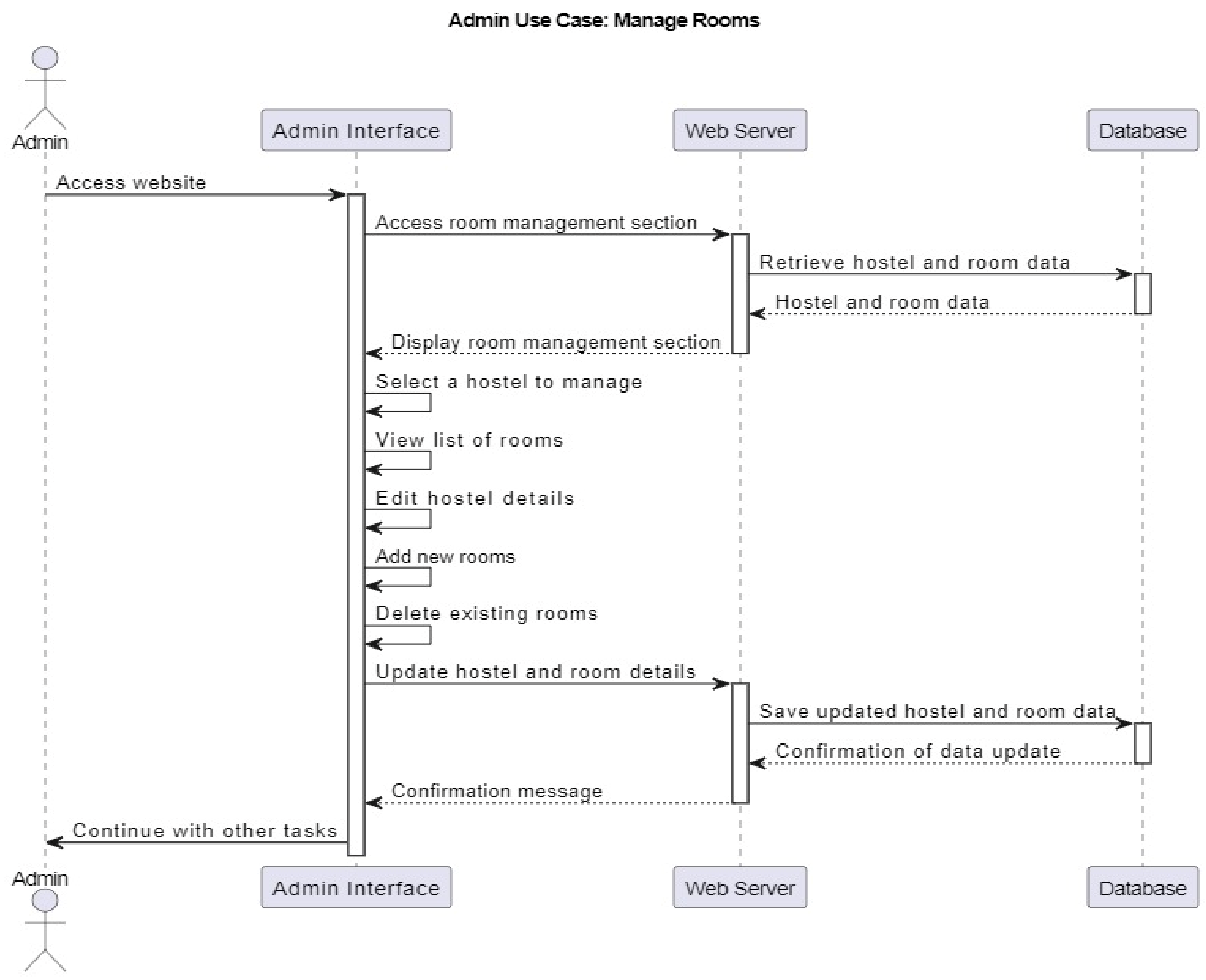
Sequence Diagram: Student Room Allocation


Room Allocation
Admin
Student
Admin Interface
Student Interface
Web Server
Database
Access website
Access allocation subsection
Retrieve current room status
Room status data
Display current room status
Decide locked rooms, vacant rooms, and assigned rooms
Start allocation process
Start room allocation process
Confirmation of allocation process start
Allocation process started
Continue with other tasks
Access website
Access allocation subsection
Retrieve current room status
Room status data
Display current room status
Register for room allocation
Submit room allocation preferences
Save student preferences
Confirmation of preferences submission
Confirmation message
Continue with other tasks
Stop allocation process
Save allocation to assigned rooms
Confirmation of allocation saving
Empty rooms in allocation subsection
Confirmation of room emptying
Allocation process stopped

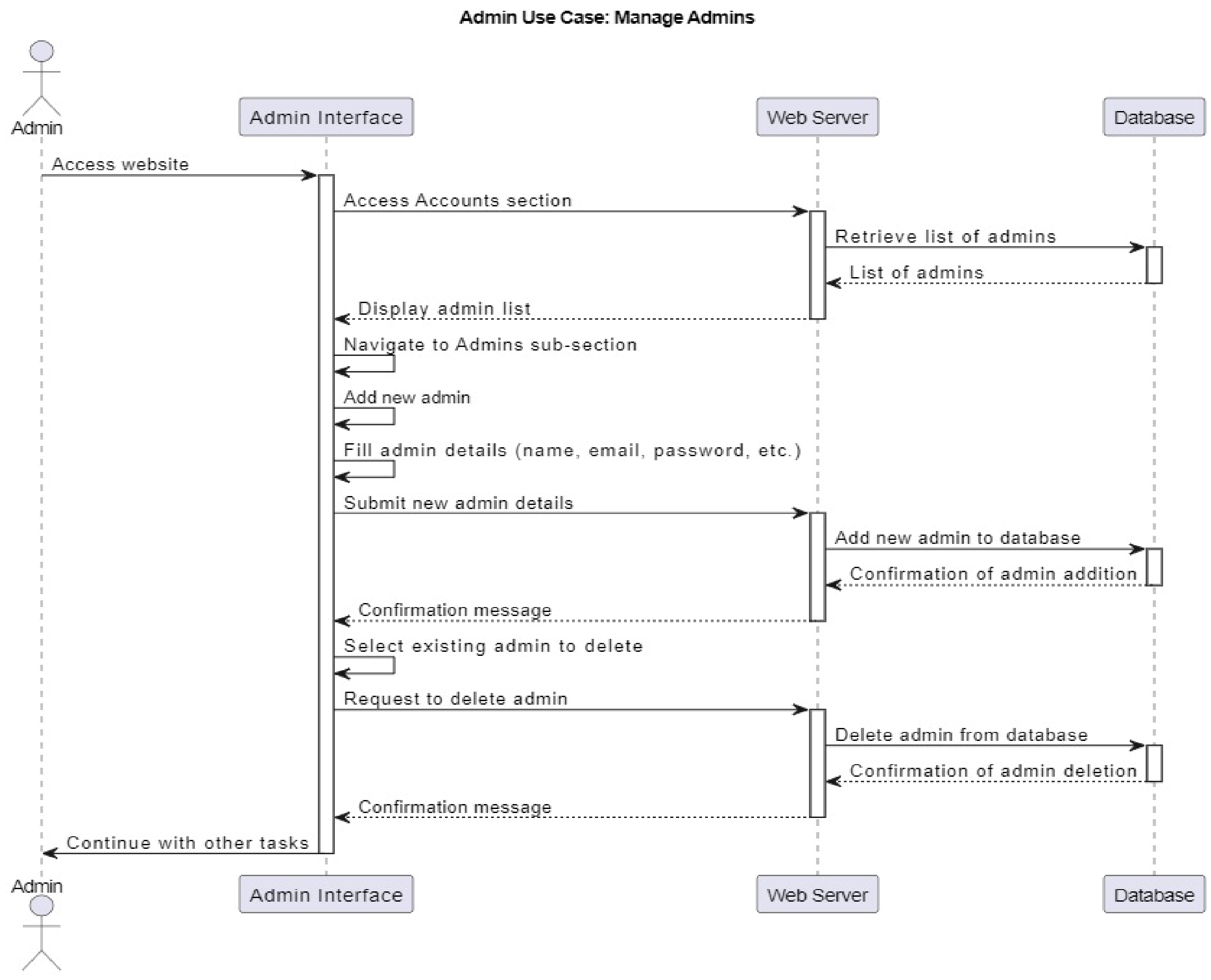
Sequence Diagram: Manage Complaint



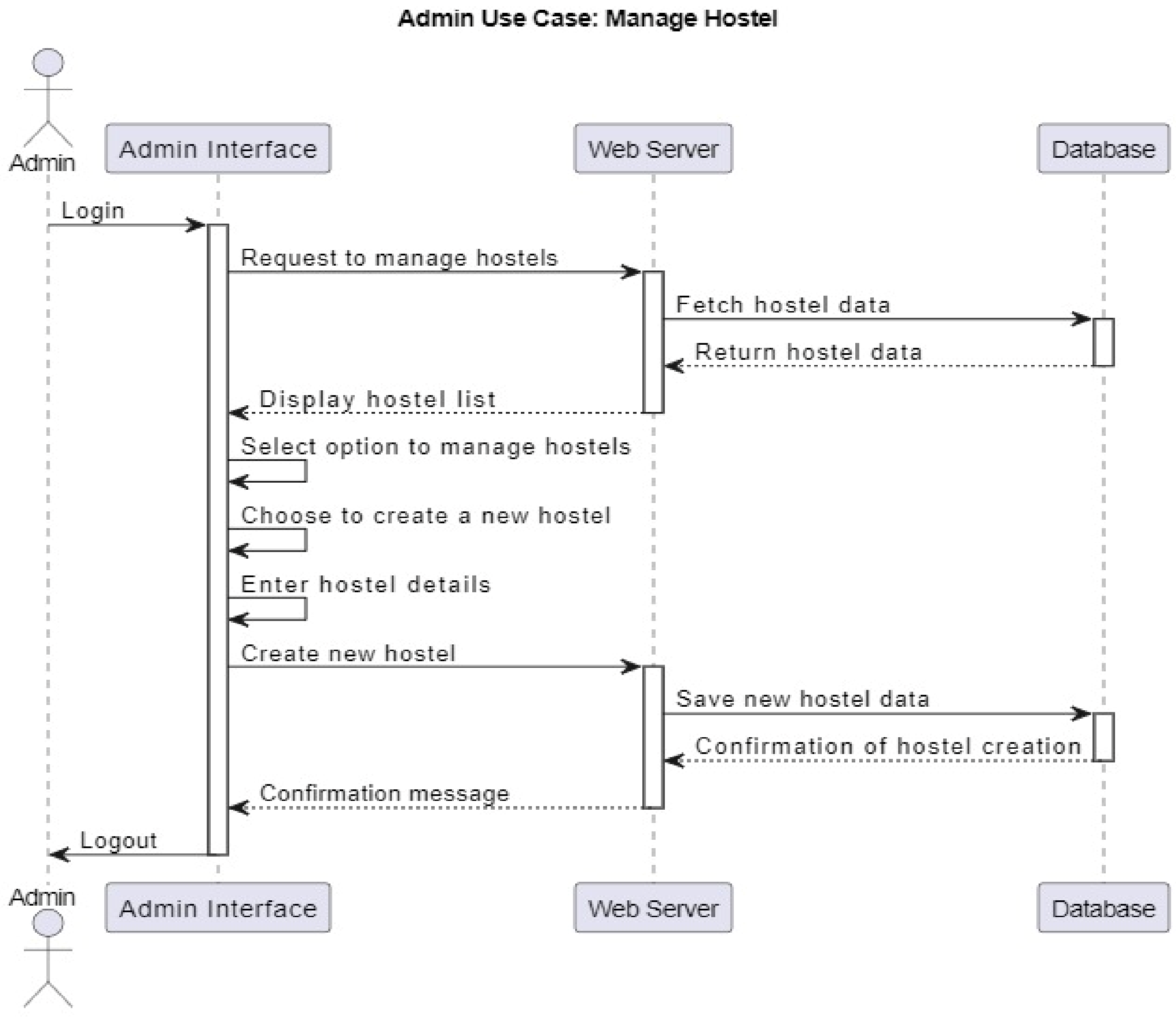
Sequence Diagram: Manage Rooms


Admin Use Case: Manage Rooms
Admin
Admin Interface
Web Server
Database
Access website
Access room management section
Retrieve hostel and room data
Hostel and room data
Display room management section
Select a hostel to manage
View list of rooms
Edit hostel details
Add new rooms
Delete existing rooms
Update hostel and room details
Save updated hostel and room data
Confirmation of data update
Confirmation message
Continue with other tasks
Admin
Admin Interface
Web Server
Database

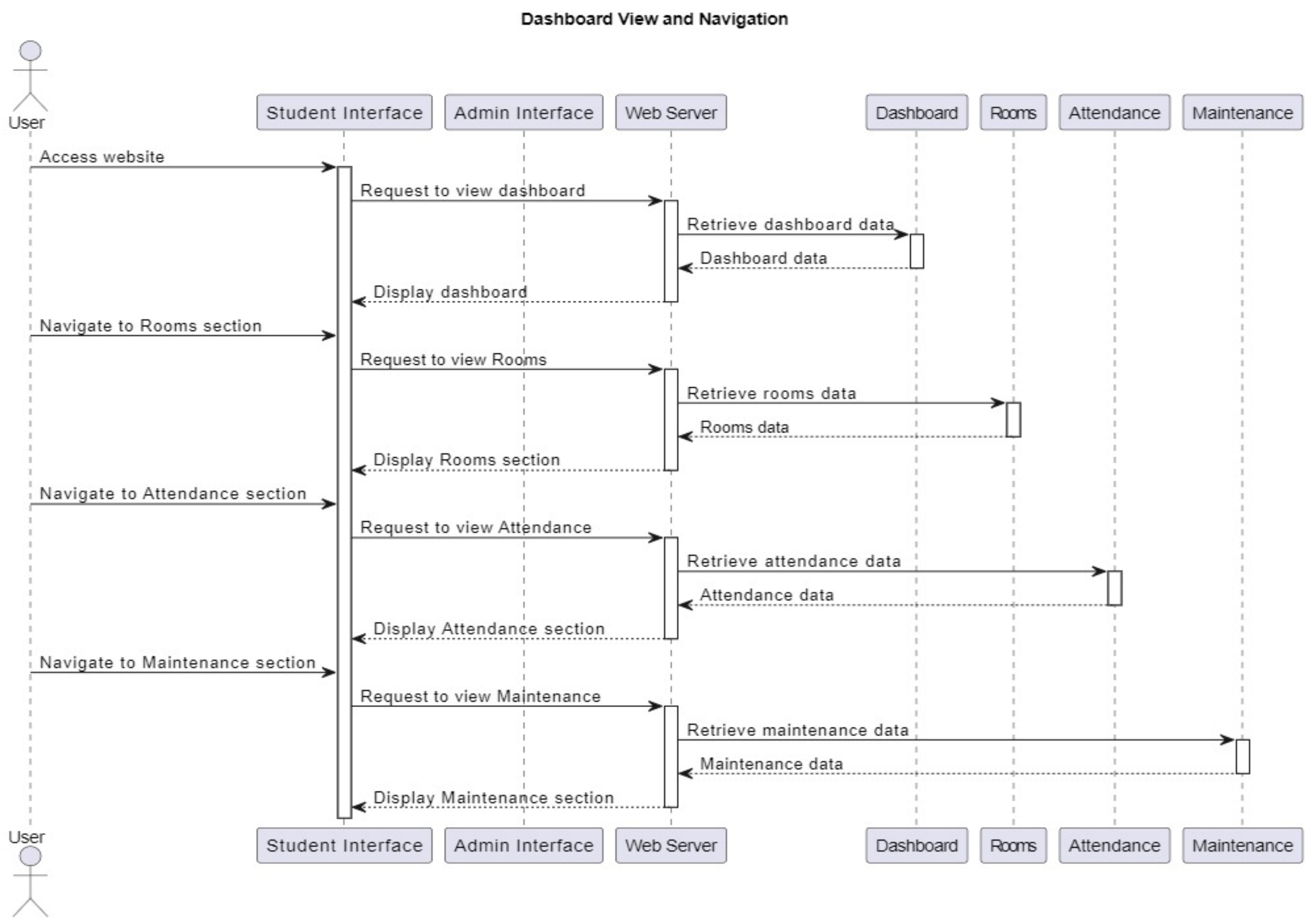
Sequence Diagram: Manage Admin


Admin Use Case: Manage Admins
Admin
Admin Interface
Web Server
Database
Access website
Access Accounts section
Retrieve list of admins
List of admins
Display admin list
Navigate to Admins sub-section
Add new admin
Fill admin details (name, email, password, etc.)
Submit new admin details
Add new admin to database
Confirmation of admin addition
Confirmation message
Select existing admin to delete
Request to delete admin
Delete admin from database
Confirmation of admin deletion
Confirmation message
Continue with other tasks
Admin
Admin Interface
Web Server
Database

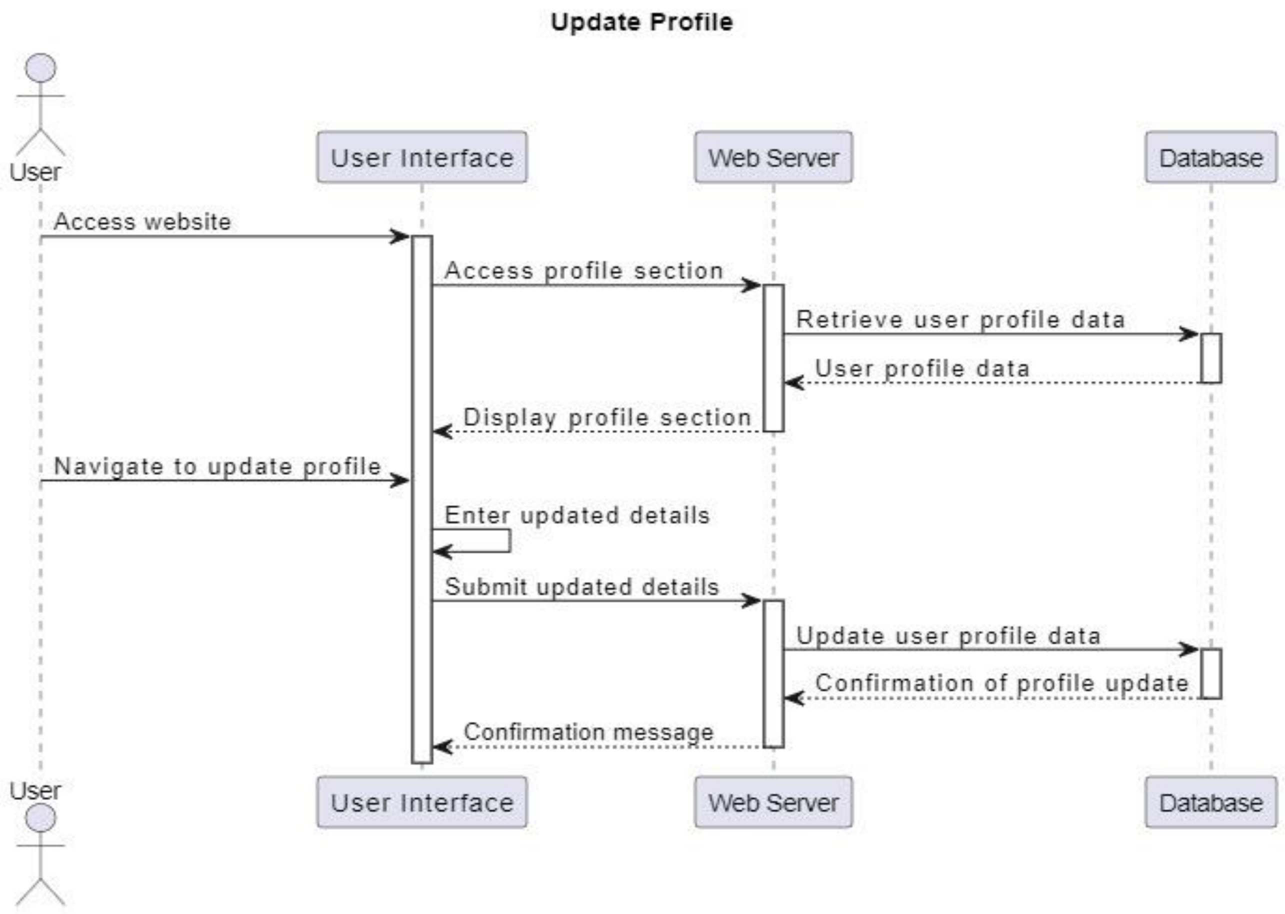
Sequence Diagram: Manage Hostel



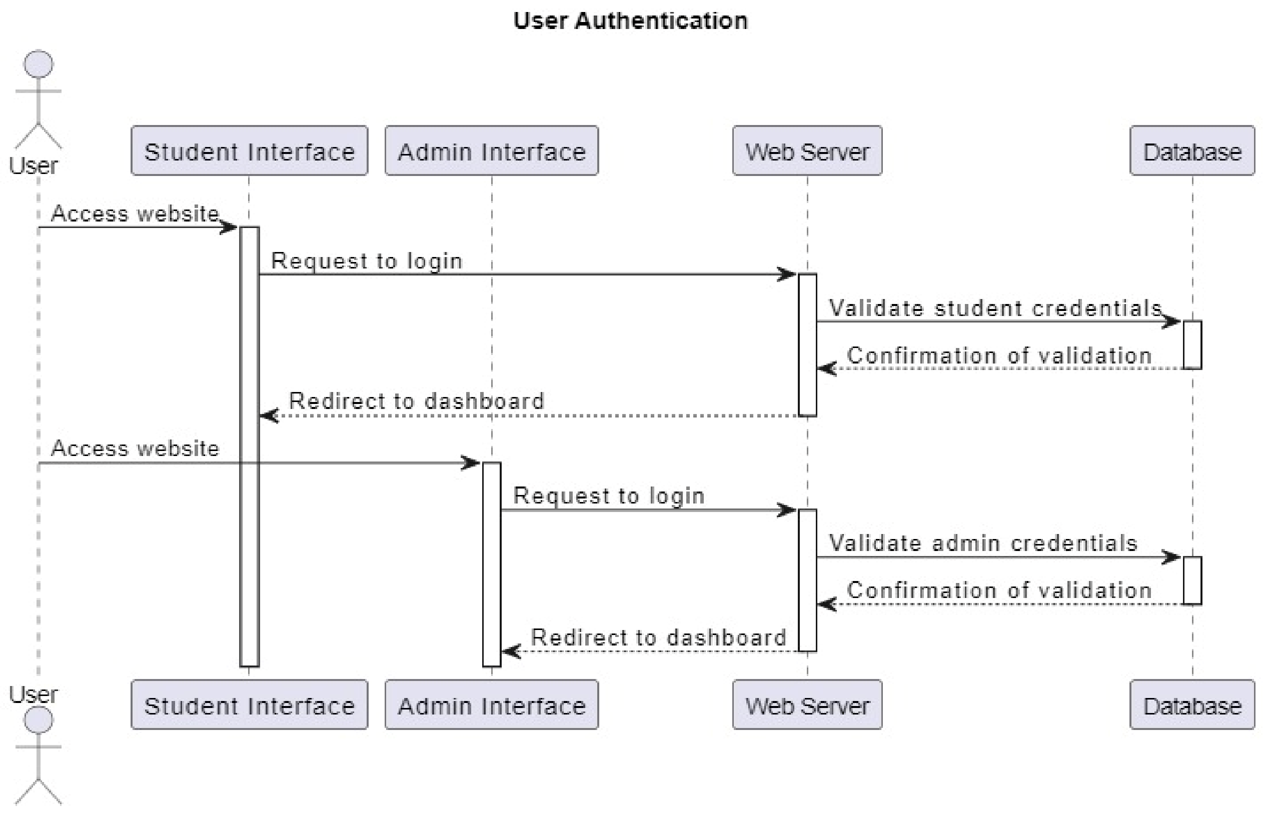
Sequence Diagram: Dashboard View and Navigation



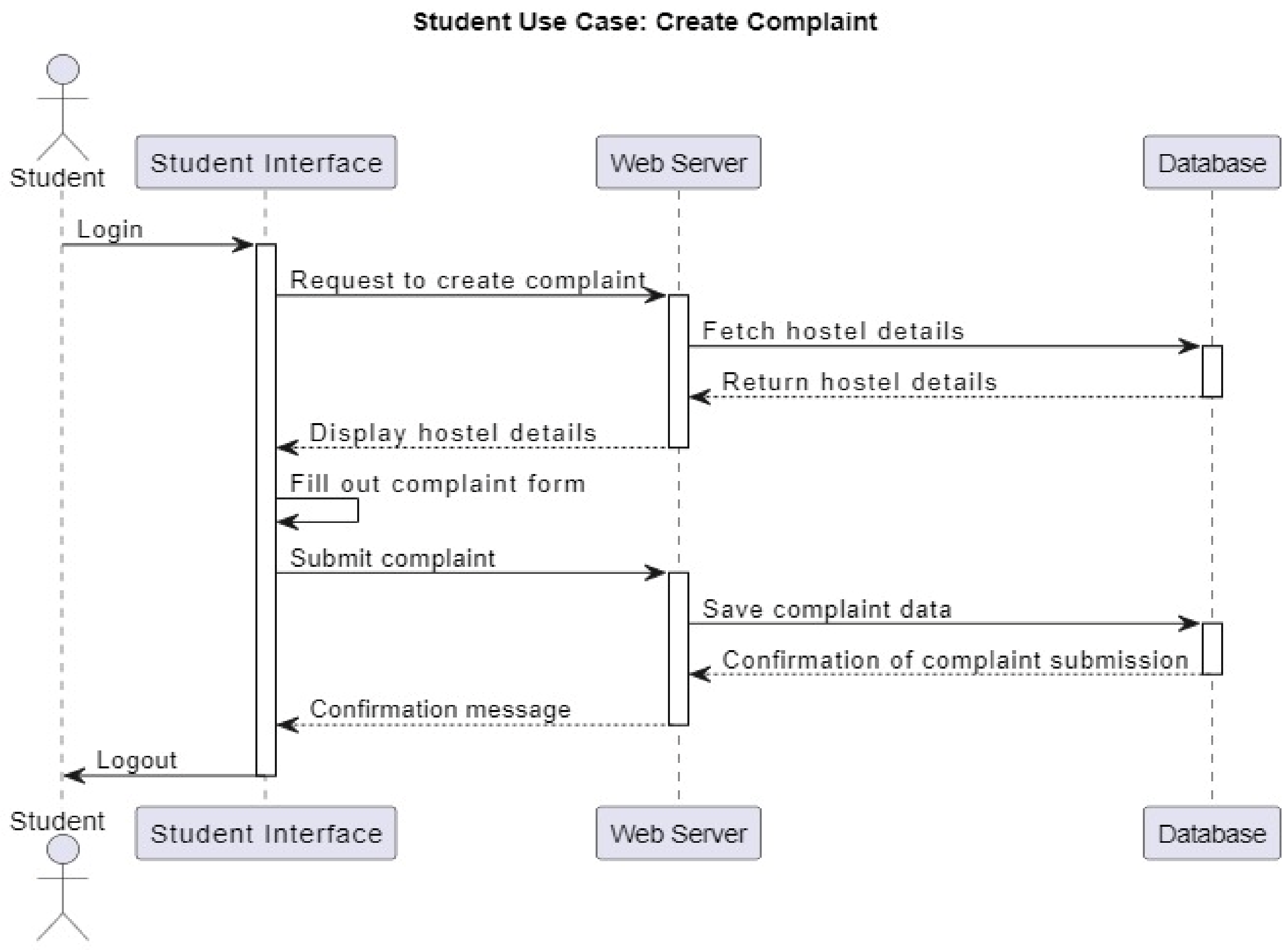
Sequence Diagram: Update Profile



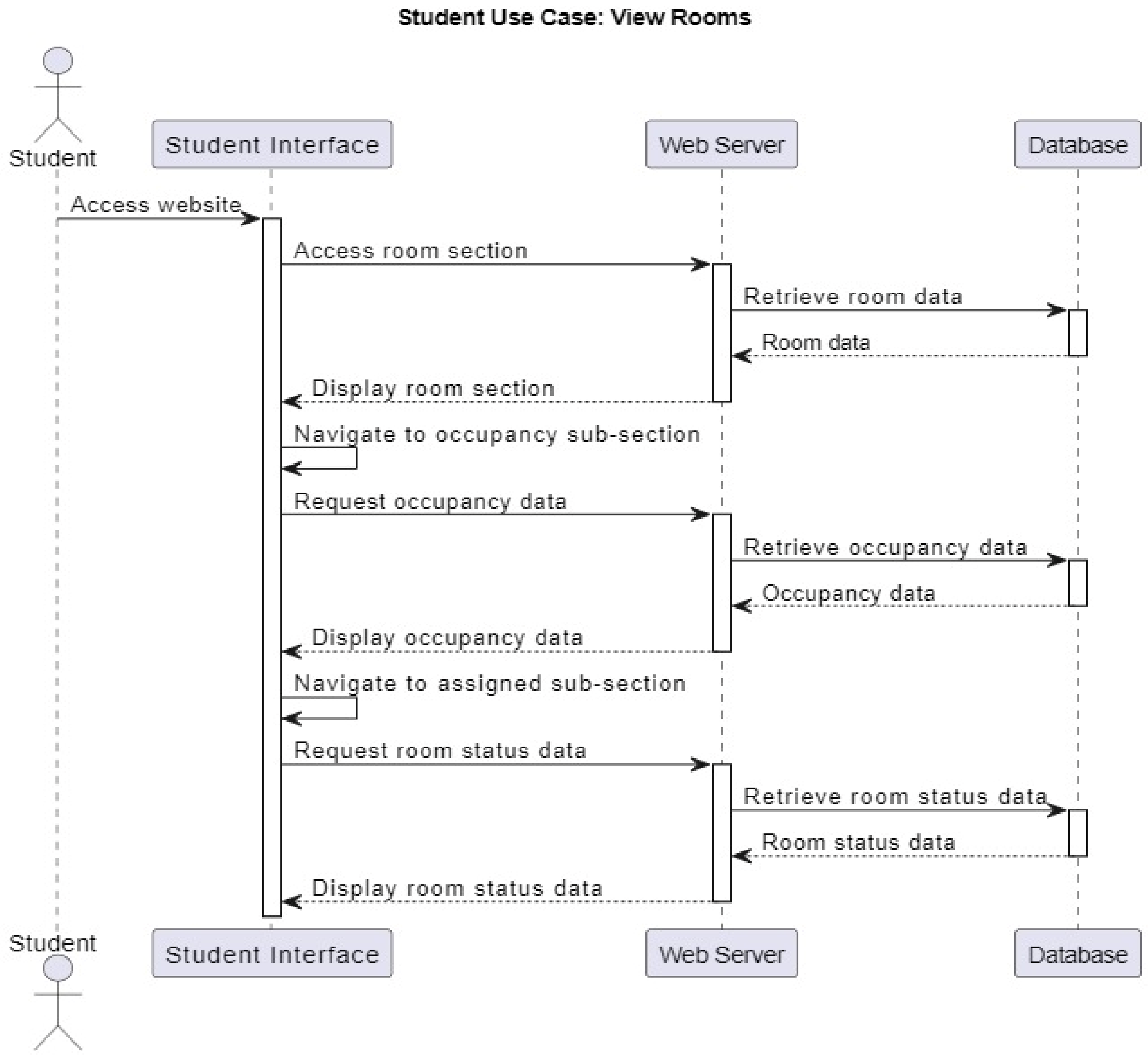
Sequence Diagram: User Authentication



Sequence Diagram: Student Complaint

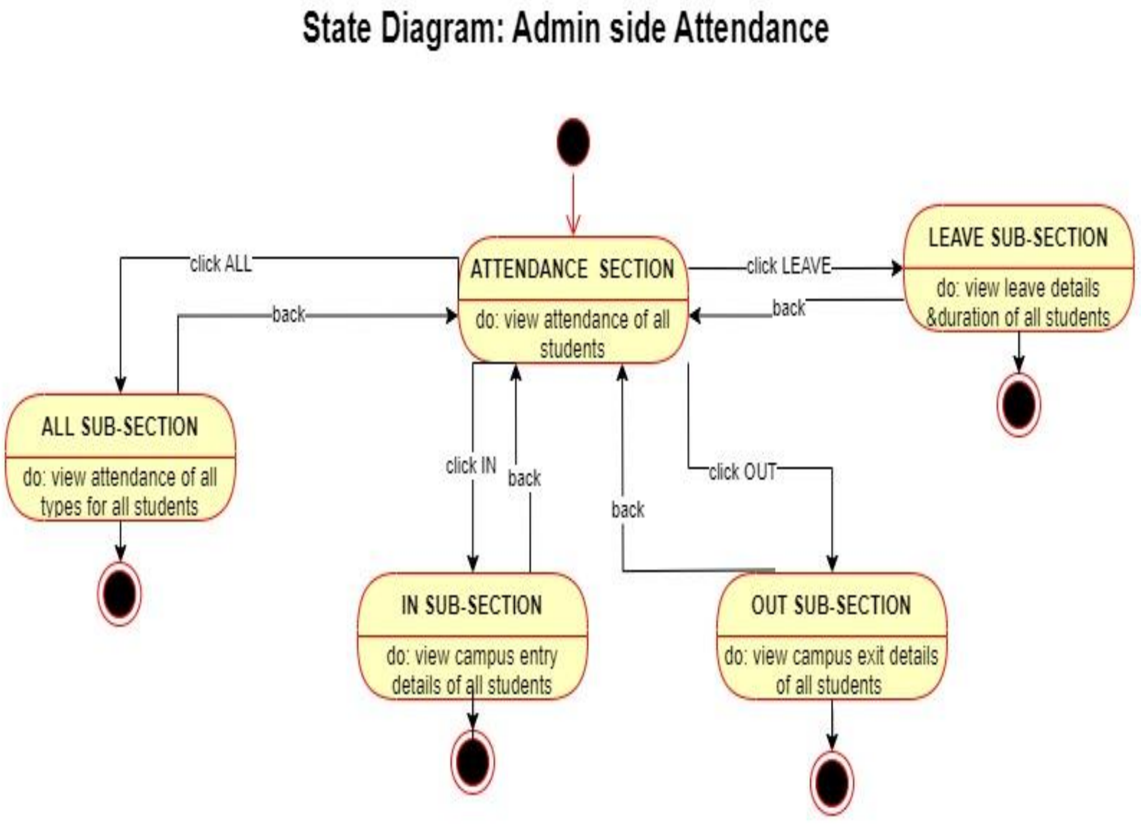


Sequence Diagram: View Rooms



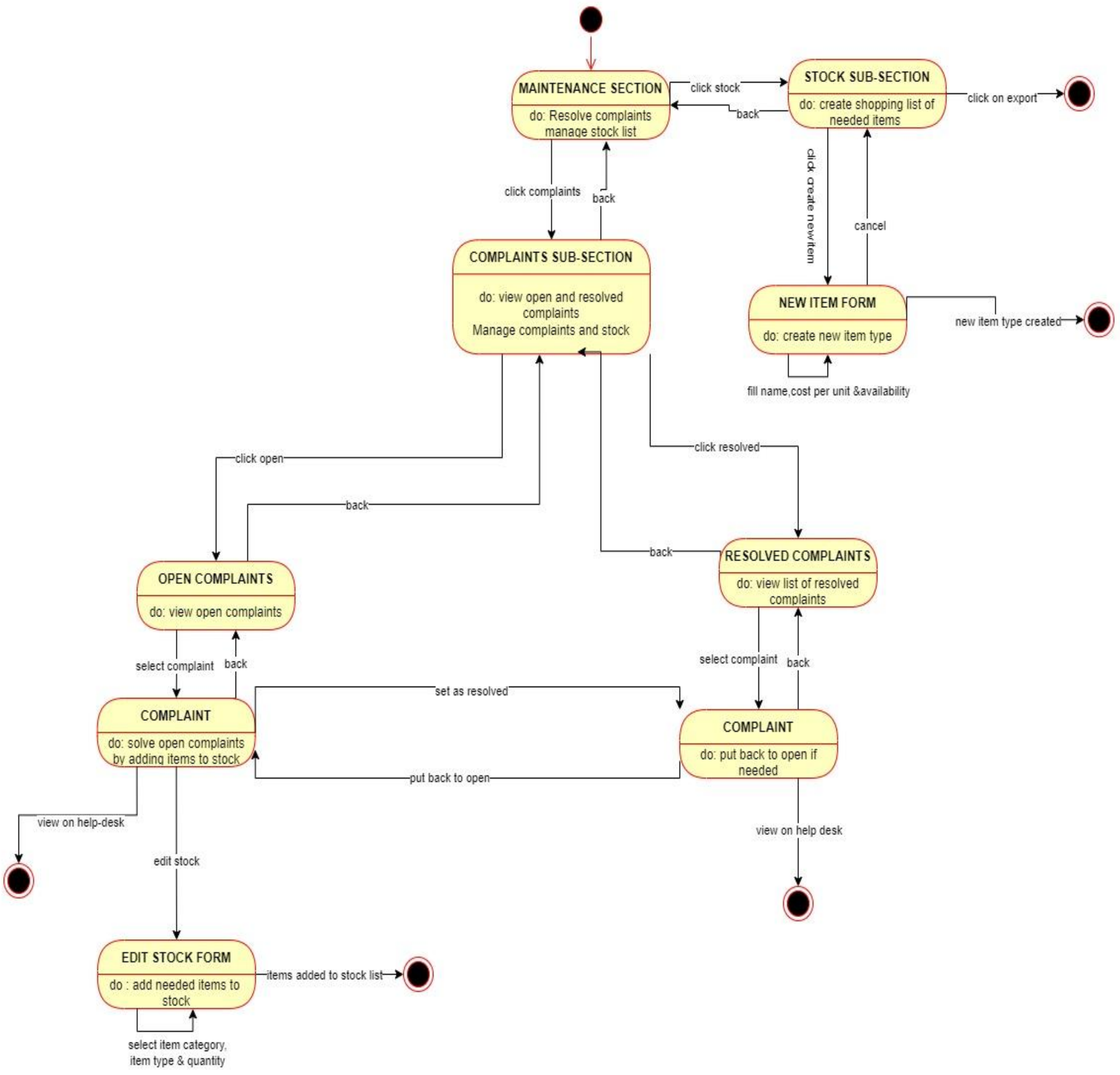
State Diagrams:

State Diagram: Admin Side Attendance


State Diagram: Admin side Attendance
ATTENDANCE SECTION
do: view attendance of all students
click ALL
back
click LEAVE
back
LEAVE SUB-SECTION
do: view leave details &duration of all students
ALL SUB-SECTION
do: view attendance of all types for all students
click IN
back
back
click OUT
IN SUB-SECTION
do: view campus entry details of all students
OUT SUB-SECTION
do: view campus exit details of all students

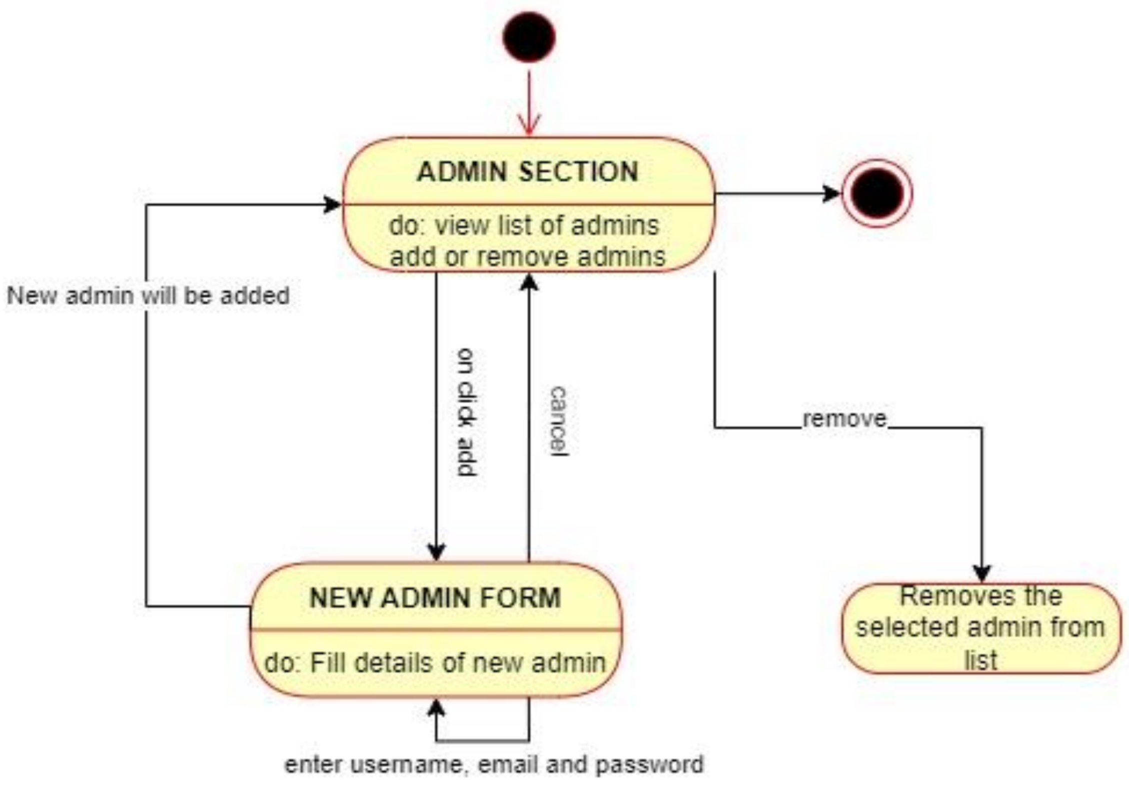
State Diagram: Admin Side Manage Complaint

State Diagram: Admin Side Manage Complaints

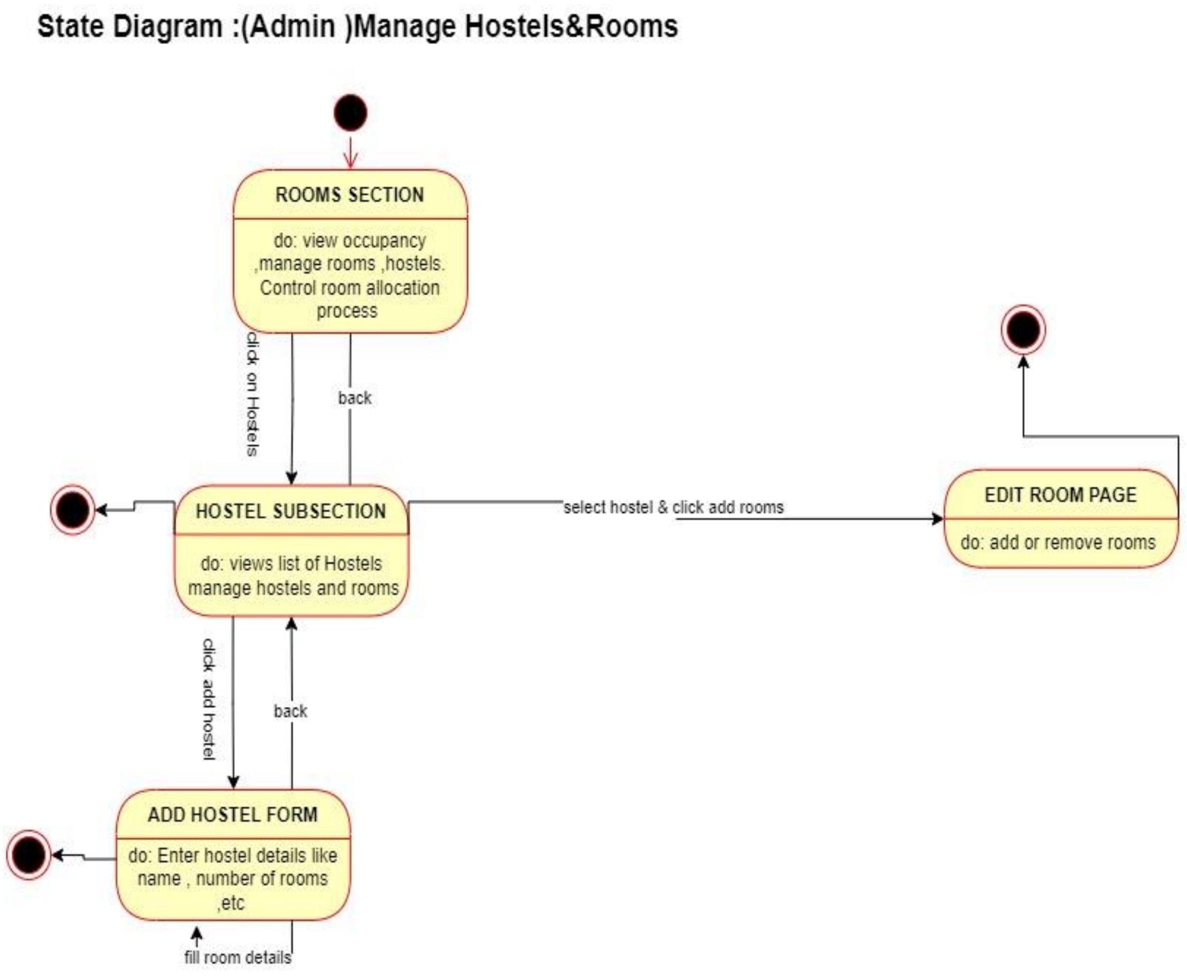


State Diagram: Manage Admin

State Diagram: Manage Admins

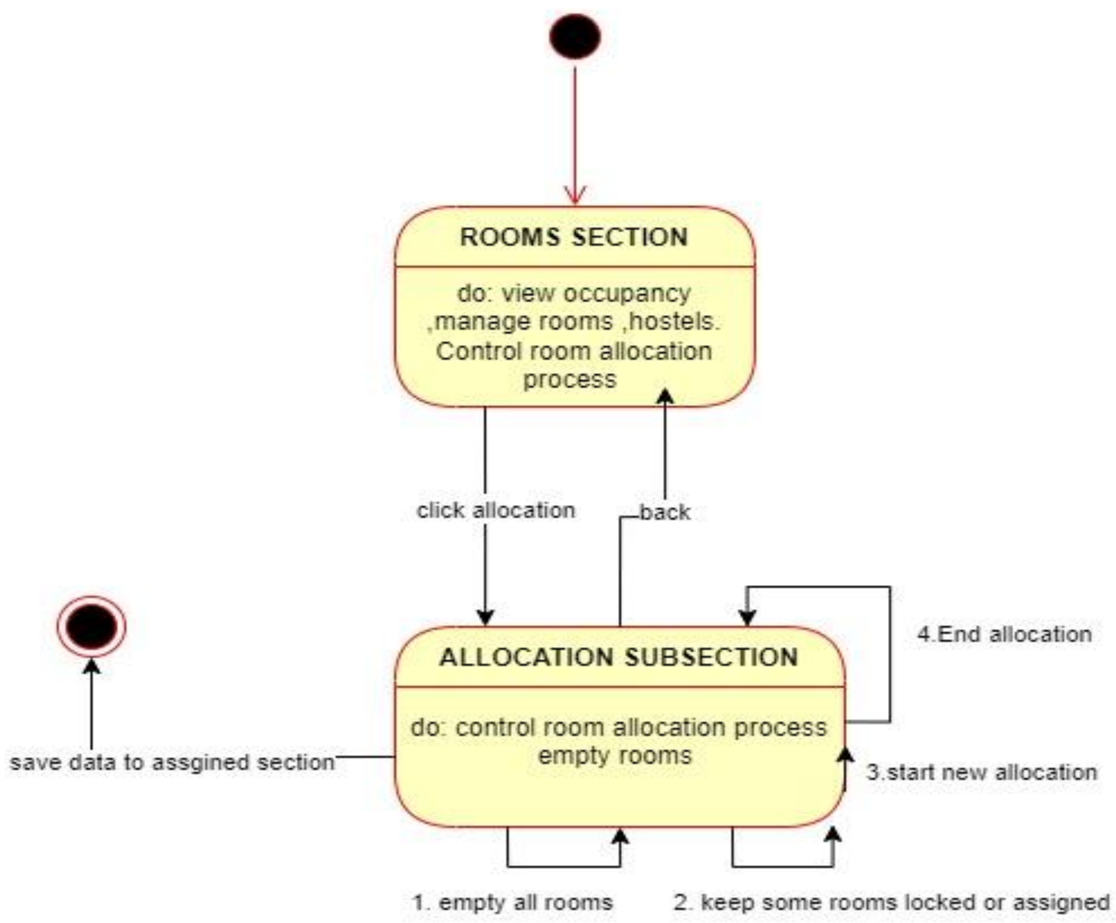


State Diagram: Manage Hostels & Rooms

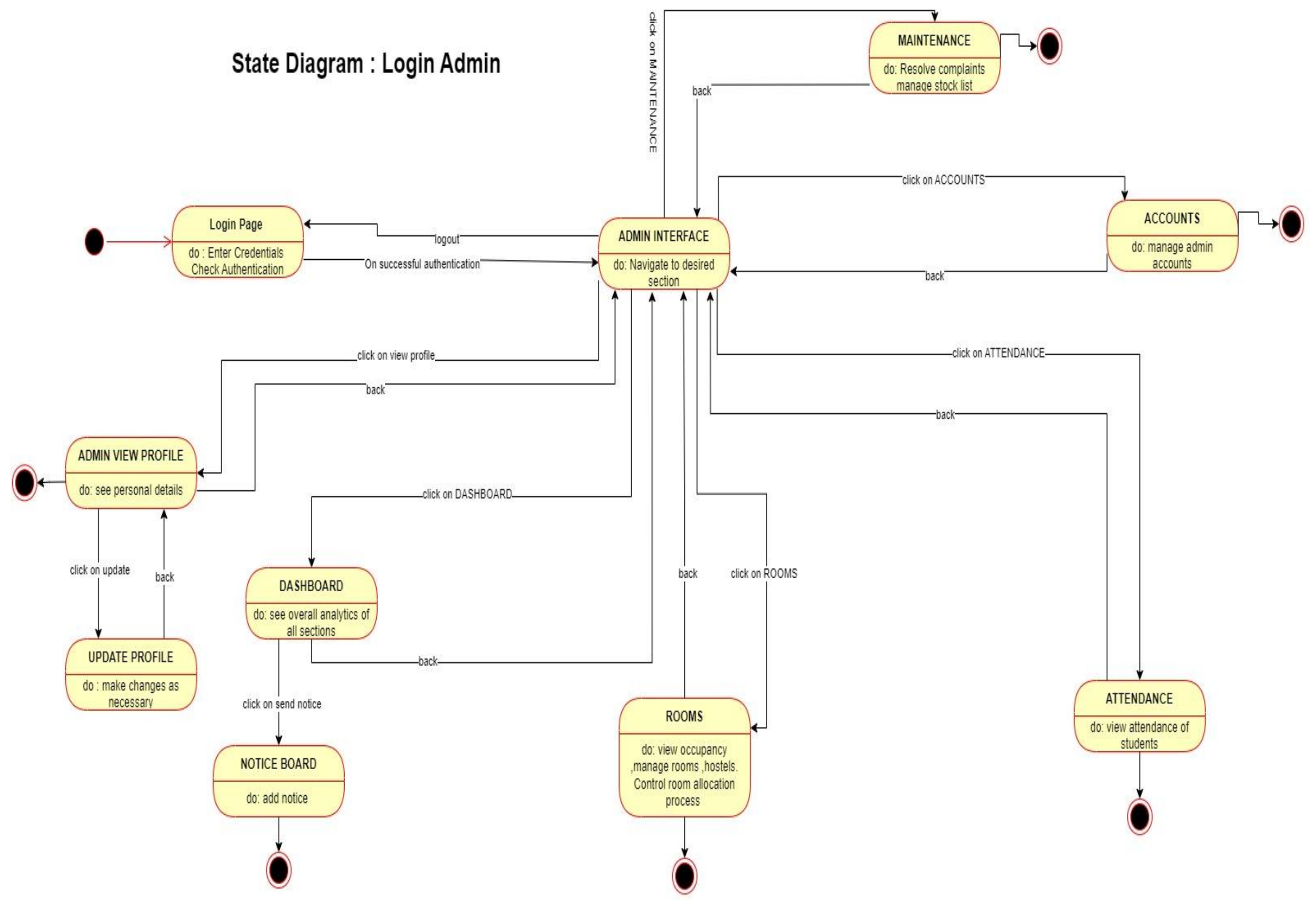

State Diagram :(Admin)Manage Hostels&Rooms
ROOMS SECTION
do: view occupancy ,manage rooms ,hostels. Control room allocation process
click on Hostels
back
HOSTEL SUBSECTION
do: views list of Hostels manage hostels and rooms
select hostel & click add rooms
EDIT ROOM PAGE
do: add or remove rooms
click add hostel
back
ADD HOSTEL FORM
do: Enter hostel details like name , number of rooms ,etc
fill room details

State Diagram: Admin Side Room Allocation

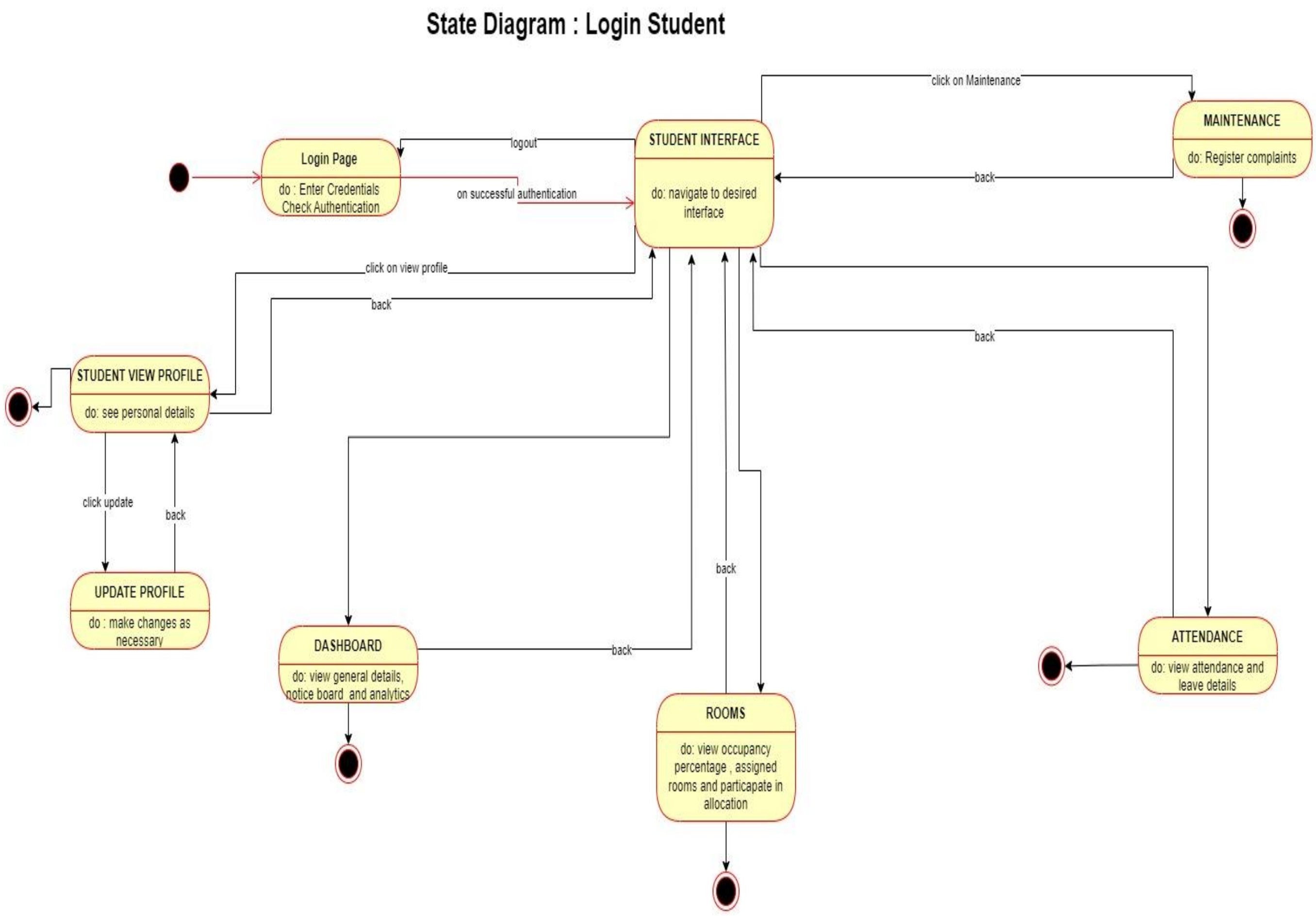
State Diagram: Admin Side Room Allocation


ROOMS SECTION
do: view occupancy ,manage rooms ,hostels. Control room allocation process
click allocation
back
ALLOCATION SUBSECTION
do: control room allocation process empty rooms
4.End allocation
3.start new allocation
save data to assgined section
1. empty all rooms
2. keep some rooms locked or assigned

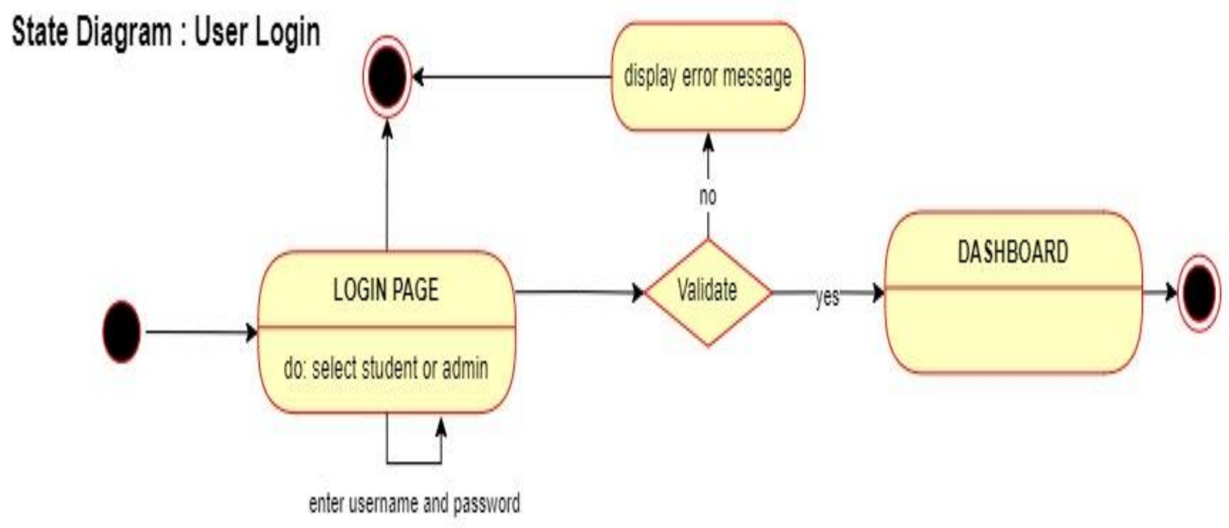
State Diagram: Admin Login


State Diagram : Login Admin
Login Page
do : Enter Credentials Check Authentication
logout
On successful authentication
ADMIN INTERFACE
do: Navigate to desired section
click on MAINTENANCE
MAINTENANCE
do: Resolve complaints manage stock list
back
click on ACCOUNTS
ACCOUNTS
do: manage admin accounts
back
click on view profile
back
ADMIN VIEW PROFILE
do: see personal details
click on update
back
UPDATE PROFILE
do : make changes as necessary
click on DASHBOARD
DASHBOARD
do: see overall analytics of all sections
back
click on send notice
NOTICE BOARD
do: add notice
click on ATTENDANCE
back
ATTENDANCE
do: view attendance of students
back
click on ROOMS
ROOMS
do: view occupancy ,manage rooms ,hostels. Control room allocation process

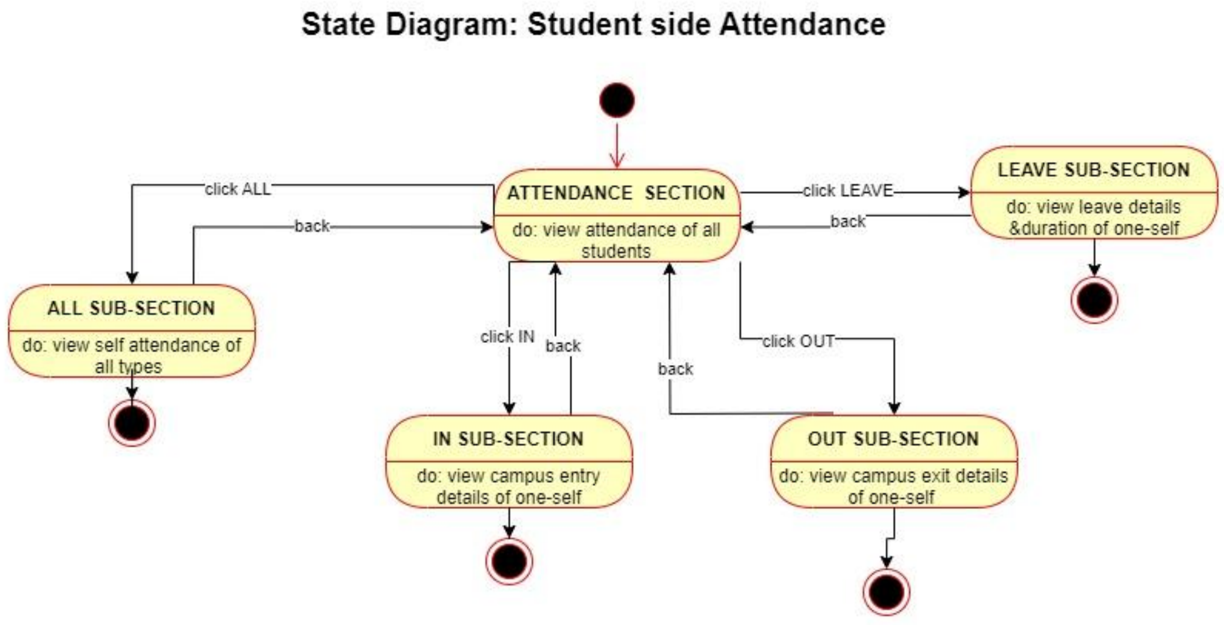
State Diagram: Student Login


State Diagram : Login Student
Login Page
do : Enter Credentials Check Authentication
on successful authentication
logout
STUDENT INTERFACE
do: navigate to desired interface
click on Maintenance
MAINTENANCE
do: Register complaints
back
click on view profile
back
STUDENT VIEW PROFILE
do: see personal details
click update
back
UPDATE PROFILE
do : make changes as necessary
back
DASHBOARD
do: view general details, notice board and analytics
back
ROOMS
do: view occupancy percentage , assigned rooms and particapate in allocation
back
ATTENDANCE
do: view attendance and leave details

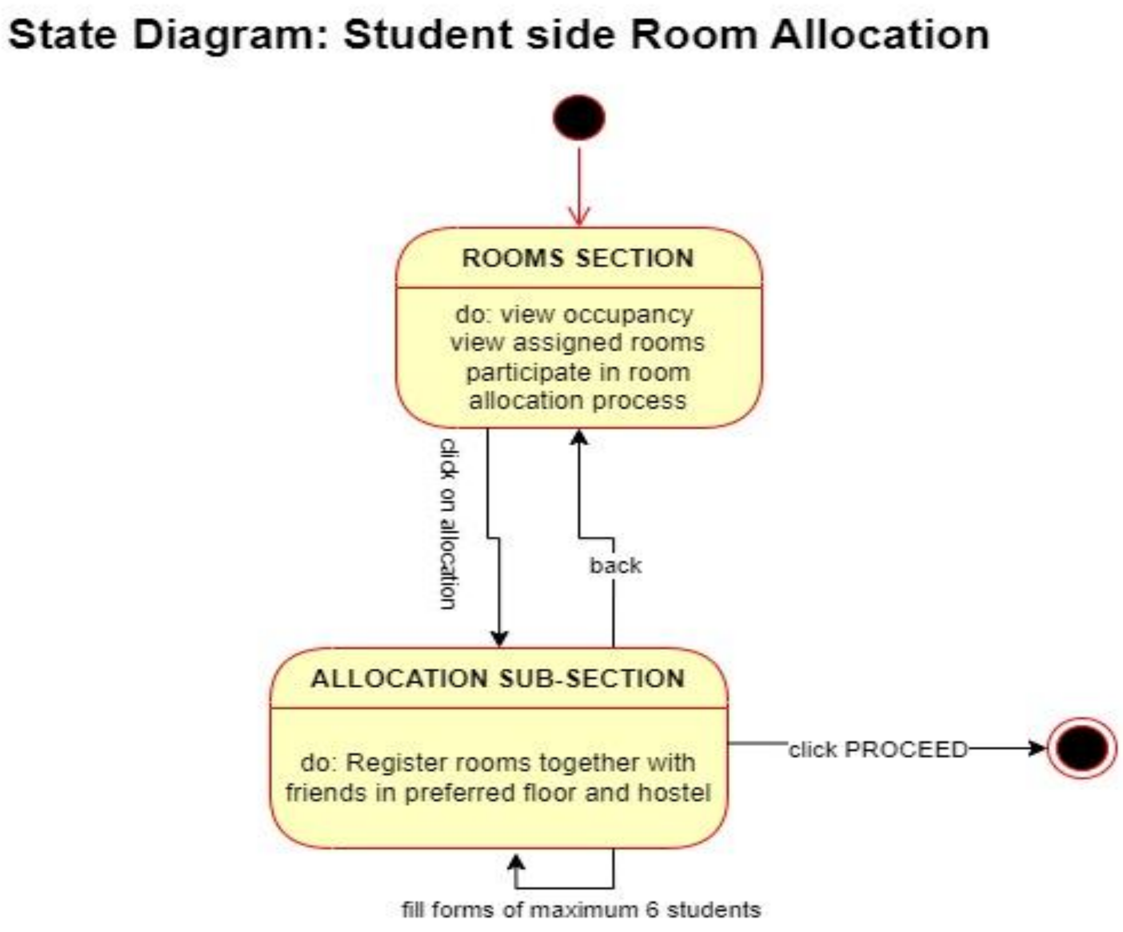
State Diagram: User Login


State Diagram : User Login
display error message
no
LOGIN PAGE
do: select student or admin
Validate
yes
DASHBOARD
enter username and password

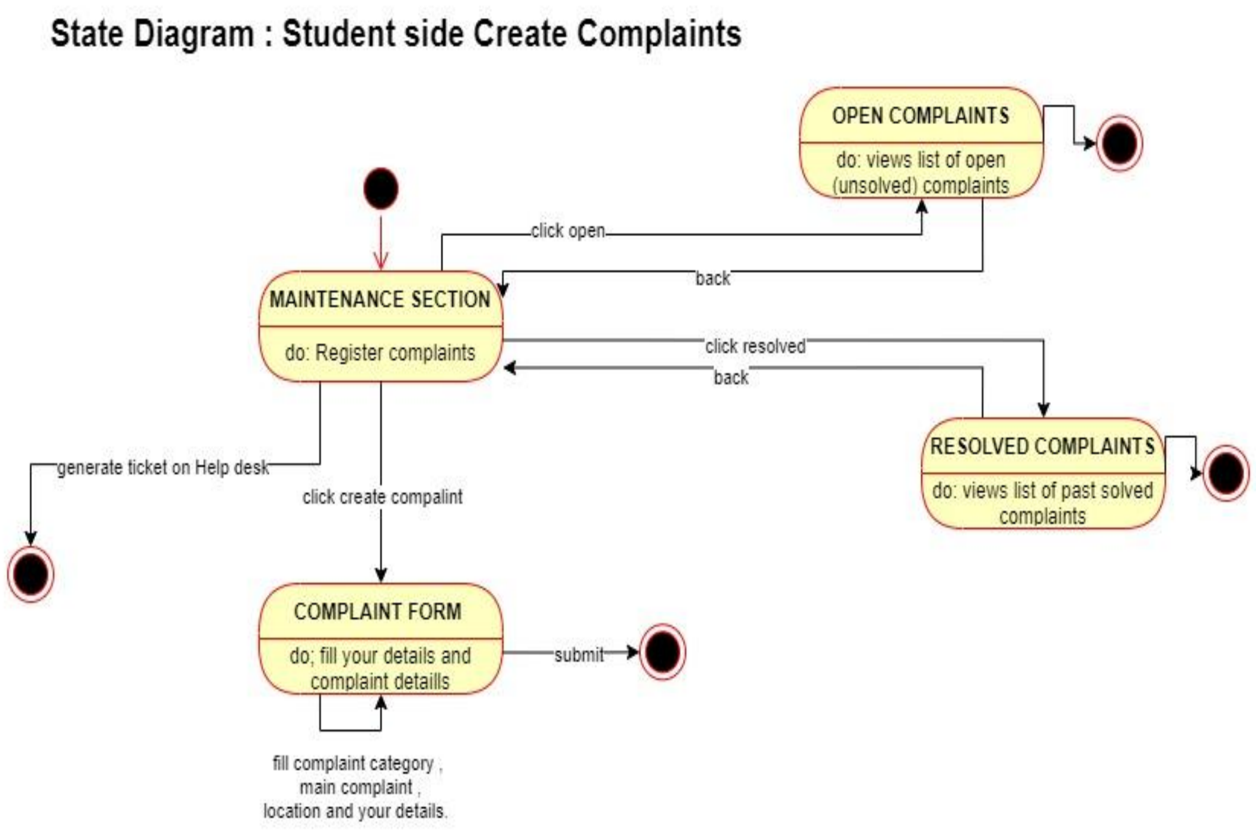
State Diagram: Student Side Attendance


State Diagram: Student side Attendance
click ALL
back
ATTENDANCE SECTION
do: view attendance of all students
click LEAVE
back
LEAVE SUB-SECTION
do: view leave details &duration of one-self
ALL SUB-SECTION
do: view self attendance of all types
click IN
back
back
click OUT
IN SUB-SECTION
do: view campus entry details of one-self
OUT SUB-SECTION
do: view campus exit details of one-self

State Diagram: Student Side Room Allocation



State Diagram: Student Side Complaint


State Diagram : Student side Create Complaints
OPEN COMPLAINTS
do: views list of open (unsolved) complaints
click open
back
MAINTENANCE SECTION
do: Register complaints
click resolved
back
RESOLVED COMPLAINTS
do: views list of past solved complaints
generate ticket on Help desk
click create compalint
COMPLAINT FORM
do; fill your details and complaint detaills
submit
fill complaint category , main complaint , location and your details.

3.1 Logical Architecture Description

3.1.1 Class Diagram explanation:

1. Classes: The diagram depicts various classes, including “User,” “Student,” “Attendance,” “Stock,” “Hostel,” “Notices,” “Complaint,” and “Location.”

2. Attributes: Each class has specific attributes associated with it. For example, the “User” class may have attributes like “username” and “email.”

3. Relationships: The lines connecting the classes represent relationships. For instance, an association between “User” and “Student” indicates that a user can be associated with a student.

4. System Overview: The diagram provides a visual overview of how different elements within the system interact with each other.

3.1.2 Sequence Diagram:

Sequence Diagram: Student Room Allocation

Input:

- Saved Instance State: The state of the activity, which may include data such as user input or UI state.

- Email: User's email address for authentication.

- Password: User's password for authentication.

Output:

- Launch the Activity: The activity is successfully launched upon authentication.

- Sign In: Opens the landing page if login is successful.

Method Description:

1. On Create(Bundle saved Instance State):

   - This method is invoked when the activity is first launched.

   - It initializes the activity and retrieves the state of the activity from the saved Instance State parameter.

   - The UI elements are populated using set Content View method.

2. Authentication Process:

   - Users are prompted to input their email and password.

   - Upon submission, the system verifies the credentials.

3. Successful Login:

   - If the login is successful, the activity proceeds to launch the landing page.

4. Admin Functionality:

   - Room Data: Admin can access details of all rooms, including room number, capacity, and occupancy status.

- Confirmation Status: Admin can view the confirmation status of room allocations.
- Assigned Rooms: Admin can retrieve a list of rooms allocated to students.
- Empty Rooms: Admin can generate a list of unoccupied rooms.
- Starting of Rooms Allocation Process: Admin initiates the room allocation process.

5. Student Functionality:
   - Room Preferences: Students specify their room preferences.
   - Confirmation Status: Students check the status of their room allocation requests.
   - Assigned Room Details: Students view details of their assigned room.
   - Available Rooms: Students access a list of unoccupied rooms.
   - Room Allocation Process: Students submit their preferences, and admin allocates rooms based on availability and preferences.

Manage Complaint

Input:

- complaint Request: Details of a new complaint lodged by a hostel resident.
- Resolved Complaint: Information about a resolved complaint.
- Complaint List: List of all current complaints in the hostel.
- Stock Details: Details of available stock items in the hostel.
- Update Stock Request: Request to update the stock, such as adding new items like desks, tables, fans, etc.

Output:

- Complaint Acknowledgement: Confirmation of receiving the complaint request.
- Complaint Resolved Confirmation: Notification indicating the successful resolution of a complaint.
- Current Complaint List: Updated list of current complaints.
- Stock Details Updated: Confirmation of successful stock update.

Method Description:

1. Complaint Request Handling:

- Upon receiving a Complaint Request, the system acknowledges the complaint and adds it to the list of current complaints.

- The complaint is assigned a unique identifier for tracking purposes.

2. Resolved Complaints:

- When a complaint is resolved, the system updates its status and notifies the concerned resident.

- A Complaint Resolved Confirmation is sent to acknowledge the resolution.

3. Viewing Complaint List:

- Hostel staff can access the Complaint List to view all pending complaints along with their details.

4. Stock Management:

- Stock Details provide information about available stock items in the hostel, including desks, tables, fans, etc.

- Upon receiving an Update Stock Request, the system updates the stock accordingly.

- This process involves adding new items to the inventory or updating existing ones.

- A confirmation message (Stock Details Updated) is sent after the stock update is completed successfully.

5. User Interaction:

- Hostel residents can submit complaints through a designated interface.

- Staff members can update the stock as needed based on user requests or inventory management requirements.

- Notifications and status updates ensure effective communication between residents and hostel management regarding complaints and stock availability.

Manage Rooms

Input:

- View Available Rooms: Request to view the list of available rooms in the hostel.

- Hostel Details: Details of the hostel, including room configurations and current occupancy.

- New Student Details: Information about new students, including their preferences and personal details.

- New Room Details: Information about newly added rooms, including room number, capacity, and amenities.

- Update Room Details : Request to update room details with student information and occupancy date.

Output:

- Available Rooms List: List of available rooms in the hostel.

- Updated Hostel Details: Hostel details updated with information about new students and room configurations.

- Room Added Confirmation: Confirmation message indicating successful addition of new rooms.

- Room Details Updated Confirmation: Confirmation message indicating successful update of room details with student information.

Method Description:

1. View Available Rooms:

   - Upon receiving a View Available Rooms request, the system retrieves and displays a list of rooms that are currently unoccupied.

2. Editing Hostel Details with New Students:

   - When new students join the hostel, their details are added to the system.

   - The Hostel Details are updated to reflect the new students' occupancy and preferences.

3. Adding New Rooms:

   - To accommodate more students, the hostel may need to add new rooms.

   - Upon receiving a New Room Details request, the system adds the new rooms to the hostel configuration.

- A Room Added Confirmation message is sent to confirm the successful addition of new rooms.

4. Updating Room Details with Student Information:

   - When rooms are allocated to students, their details along with the occupancy date are recorded.

   - The Update Room Details request triggers the system to update room details with student information.

   - This ensures accurate tracking of room occupancy and allocation history.

   - Upon successful update, a Room Details Updated Confirmation message is sent to acknowledge the update.

5. User Interaction:

   - Hostel staff can manage room allocation and availability through a user-friendly interface.

   - The system ensures that hostel details are up-to-date, reflecting changes in room occupancy and configuration.

   - Students can also access information about available rooms and their own room allocations, facilitating smooth hostel management.

Manage Admin

Input:

- View Existing Admin Details: Request to view details of all existing admins in the hostel management system.

- New Admin Details : Information about a new admin to be added to the system.

- Delete Admin Details: Request to delete existing admin details.

Output:

- Existing Admin Details List: List of details of all existing admins.

- Admin Added Confirmation: Confirmation message indicating successful addition of a new admin.

- Admin Deleted Confirmation: Confirmation message indicating successful deletion of an admin.

Method Description:

1. Viewing Existing Admin Details:

   - Upon receiving a View Existing Admin Details request, the system retrieves and displays details of all existing admins stored in the database.

   - This includes information such as admin names, contact details, and roles.

2. Adding New Admins:

   - When a new admin is appointed, their details are added to the system.

   - The New Admin Details input triggers the system to add the new admin to the database.

   - Upon successful addition, an Admin Added Confirmation message is sent to acknowledge the addition of the new admin.

3. Deleting Existing Admins:

   - If an admin is no longer associated with the hostel management system, their details can be deleted.

   - The Delete Admin Details request specifies the admin to be deleted.

   - Upon successful deletion, an Admin Deleted Confirmation message is sent to confirm the removal of the admin from the system.

4. User Interaction:

   - Hostel staff with appropriate privileges can access the admin management functionality through a dedicated interface.

   - They can view existing admin details, add new admins, and delete existing ones as necessary.

   - Confirmation messages ensure that actions such as addition or deletion of admin details are successfully executed and communicated to the user.

Manage Hostels

Input:

- View All Hostel Details: Request to view details of all existing hostels in the system.

- New Hostel Details: Information about a new hostel to be created.

- Add Hostel Details: Details of a hostel to be added to the system.

- Fetch Hostel Data: Request to fetch data of a specific hostel.

Output:

- All Hostel Details: List of details of all existing hostels.

- Hostel Created Confirmation: Confirmation message indicating successful creation of a new hostel.

- Hostel Details Added Confirmation: Confirmation message indicating successful addition of hostel details.

- Hostel Data: Details of the requested hostel.

Method Description:

1. Viewing All Hostel Details:

   - Upon receiving a View All Hostel Details request, the system retrieves and displays details of all existing hostels stored in the database.

   - This includes information such as hostel names, addresses, capacities, and any other relevant details.

2. Creating a New Hostel:

   - When a new hostel is established, its basic details are added to the system.

   - The New Hostel Details input triggers the system to create a new hostel entity in the database.

   - Upon successful creation, a Hostel Created Confirmation message is sent to acknowledge the establishment of the new hostel.

3. Adding Details of the Hostel:

   - Once a hostel is created, additional details such as address, capacity, facilities, etc., need to be added.

- The Add Hostel Details input provides these details to be associated with the respective hostel entity.

- Upon successful addition, a Hostel Details Added Confirmation message is sent to acknowledge the addition of hostel details.

4. Fetching Hostel Data:

- When specific details of a hostel are required, a fetch Hostel Data request is made, specifying the hostel of interest.

- The system retrieves the requested hostel data from the database and returns it as Hostel Data .

5. User Interaction:

- Hostel administrators or authorized staff can access the hostel management functionality through a user-friendly interface.

- They can view existing hostel details, create new hostels, add additional details, and fetch data as needed.

- Confirmation messages ensure that actions such as hostel creation or addition of details are successfully executed and communicated to the user.

Manage Dashboard View

Input:

- View Room Details: Request to view details of all rooms in the hostel.

- View Maintenance Section: Request to access the maintenance section of the hostel management system.

- View In Out Section: Request to display the in-out section of the hostel management system.

Output:

- Room Details: Details of all rooms in the hostel.

- Maintenance Section: Access to the maintenance section.

- In Out Section: Display of the in-out section.

Method Description:

1. Viewing Room Details:

- Upon receiving a View Room Details request, the system retrieves and displays details of all rooms in the hostel.

- Room details typically include room numbers, capacities, current occupancy status, and any other relevant information.

2. Accessing Maintenance Section:

- When a user requests to View Maintenance Section, the system grants access to the maintenance section of the hostel management system.

- This section allows staff to manage maintenance requests, schedule maintenance tasks, and track maintenance history.

3. Displaying In-Out Section:

- Upon receiving a View In Out Section request, the system displays the in-out section of the hostel management system.

- This section typically provides information about hostel residents' check-ins and check-outs, including dates and times.

- It may also include functionality for managing guest entries and exits.

4. User Interaction:

- Users interact with the dashboard interface to navigate through different sections of the hostel management system.

- They can view room details, access maintenance features, and monitor in-out activities efficiently.

- The dashboard provides a centralized hub for hostel management tasks, facilitating smooth operation and oversight.

Manage Profile

Input:

- View All Student Details: Request to view details of all students in a particular hostel.

- Update Student Information: Request to update information of a specific student.

- Student information to be updated, including details such as name, contact information, room number, etc.

Output:

- All Student Details: Details of all students in the hostel.

- Update Confirmation: Confirmation message indicating successful update of student information.

Method Description:

1. Viewing All Student Details:

   - Upon receiving a View All Student Details request, the system retrieves and displays details of all students currently residing in the particular hostel.

   - Student details typically include name, contact information, room number, check-in date, etc.

2. Updating Student Information:

   - When a request to update student information (Update Student Information) is received, the system modifies the specified details according to the provided input.

   - This could involve updating contact information, room allocation, or any other relevant data.

   - After successfully updating the information, the system sends an Update Confirmation message to acknowledge the completion of the update process.

3. User Interaction:

   - Hostel staff or administrators interact with the system interface to access student details and update information as needed.

   - The system ensures data accuracy and provides confirmation messages to inform users about the outcome of their update requests.

   - This functionality facilitates efficient management of student records and ensures that information is kept up-to-date in the hostel management system.

Manage User Authentication

Input:

- Login Request: Request to login with user credentials.

- User credentials: Email and password.

Output:

- Valid Credential Confirmation`: Confirmation message indicating that the entered credentials are valid.

- Redirect to the dashboard upon successful authentication.

Method Description:

1. Login Request:

   - When a user attempts to login, a Login Request is initiated, providing the user's email and password.

2. Entering Site with Valid Credential:

   - The system verifies the provided credentials against the stored user data.

   - If the credentials are valid, the user is granted access to the site.

3. Confirmation of Valid Credential:

   - Upon successful authentication, the system sends a Valid Credential Confirmation message to acknowledge that the entered credentials are valid.

4. Redirect to Dashboard:

   - After confirming the validity of the credentials, the user is redirected to the dashboard or the landing page of the hostel management system.

   - From here, the user can access various functionalities based on their role and permissions.

5. User Interaction:

- Users interact with the authentication system by providing their credentials through a login form.

- The system processes these credentials, authenticates the user, and grants access accordingly.

- Confirmation messages and redirection to the appropriate interface enhance the user experience and ensure secure access to the hostel management system.

Manage User Complaint

Input:

- Display Hostel Details Request: Request to display details of the hostel.
- Complaint Submission: Complaint submitted by the user, including details such as type of complaint, description, room number, etc.

Output:

- Hostel Details: Details of the hostel displayed to the user.
- Complaint Confirmation Message`: Confirmation message indicating successful submission of the complaint.
- Saving complaint data for further processing.

Method Description:

1. Display Hostel Details Request:

- Upon receiving a Display Hostel Details Request, the system retrieves and displays details of the hostel, including room configurations, amenities, etc.

- This information assists users in identifying and reporting issues within the hostel premises.

2. Submitting Complaint:

- Users submit complaints (Complaint Submission) through a designated interface, providing details such as the type of complaint (e.g., maintenance issue, noise disturbance), description of the problem, and relevant location details (e.g., room number).

- This allows users to report any issues they encounter in the hostel premises.

3. Confirmation Message:

- After successfully submitting the complaint, the system sends a Complaint Confirmation Message to acknowledge the receipt of the complaint.

- This confirmation assures users that their complaint has been registered and will be addressed by the hostel management.

4. Saving Complaint Data:

- Upon submission, the complaint data is saved in the system for further processing by hostel management staff.

- This includes storing details such as the type of complaint, description, room number, timestamp, etc.

- Saving complaint data ensures that hostel management can track and manage reported issues effectively.

5. User Interaction:

- Users interact with the complaint submission system by providing details of their complaints through an intuitive interface.

- The system processes these complaints, sends confirmation messages, and saves the complaint data for hostel management to address in a timely manner.

Manage Rooms View

Input:

- View Rooms Data Request: Request to view data about the rooms in the hostel.
- View Students In Room Request: Request to view details of students residing in a particular room.

Output:

- Rooms Data: Details of all rooms in the hostel, including occupancy status and room types.
- Filled Rooms Count: Number of rooms that are currently occupied.
- Vacant Rooms Count: Number of rooms that are currently unoccupied.
- Single Rooms Count : Number of single rooms that are vacant and filled.

- Double Rooms Count: Number of double rooms that are vacant and filled.

- Students In Room Details: Details of students residing in a particular room.

Method Description:

1. Viewing Rooms Data

   - Upon receiving a View Rooms Data Request, the system retrieves and provides details of all rooms in the hostel.

   - Room data includes room numbers, occupancy status (filled or vacant), and room types (single or double).

2. Counting Filled and Vacant Rooms:

   - The system calculates the number of filled and vacant rooms based on the occupancy status retrieved from the database.

   - This information is provided as Filled Rooms Count and Vacant Rooms Count respectively.

3. Counting Single and Double Rooms:

   - Similarly, the system calculates the number of single and double rooms that are vacant and filled.

   - This information is provided as Single Rooms Count and Double Rooms Count.

4. Viewing Students in a Particular Room:

   - Upon receiving a View Students In Room Request for a specific room, the system retrieves and displays details of students residing in that room.

   - This includes information such as student names, contact details, check-in dates, etc.

   - The details are provided as Students In Room Details.

5. User Interaction:

   - Hostel staff or administrators interact with the system to access room data and student details through user-friendly interfaces.

   - This functionality facilitates efficient management of room allocations and enables hostel staff to provide necessary assistance to students residing in the hostel.

3.1.3 State Diagram:

Admin side Attendance

Input:

- View All Student Attendance Request: Request to view attendance records of all students.

- View Campus Entry Exit Request: Request to view campus entry and exit records of all students.

- View All Student Leave Details Request: Request to view leave details and duration of all students.

Output:

- All Student Attendance Records: Attendance records of all students.

- Campus Entry Exit Records: Campus entry and exit records of all students.

- All Student Leave Details: Leave details and duration of all students.

Method Description:

1. Viewing Attendance of All Students:

   - Upon receiving a View All Student Attendance Request, the system retrieves and displays the attendance records of all students.

   - Attendance records typically include dates, attendance status (present or absent), and any remarks or notes.

2. Viewing Campus Entry and Exit Records:

   - When a View Campus Entry Exit Request is received, the system fetches and presents campus entry and exit records of all students.

   - Campus entry and exit records include timestamps indicating the time when a student entered or exited the hostel premises.

3. Viewing Leave Details and Duration:

- Upon receiving a View All Student Leave Details Request, the system retrieves and provides leave details and duration of all students.

- Leave details include leave types (such as medical leave, vacation leave), start and end dates of leave, and the duration of each leave period.

4. User Interaction:

- Hostel administrators or authorized staff access the admin side attendance functionality through a web interface.

- They can view attendance records, campus entry and exit details, and leave information of all students to monitor their activities and ensure compliance with hostel rules and regulations.

- The system provides comprehensive data presentation and filtering options to facilitate efficient management and oversight of student attendance and activities.

Admin side Complaint Management

Input:

- View Help Desk Complaints Request: Request to view complaints lodged in the help desk.
- Solve Complaint Request: Request to solve a specific complaint.
- Create Item List Request: Request to create a list of items needed based on the complaint.
- View Resolved Complaints Request: Request to view previously resolved complaints.

Output:

- Help Desk Complaints List: List of complaints lodged in the help desk.
- Complaint Solved Confirmation: Confirmation message indicating successful resolution of a complaint.
- Item List Created: List of items created based on the demand in the complaint.
- Resolved Complaints List: List of previously resolved complaints.

Method Description:

1. Viewing Complaints in Help Desk:

   - Upon receiving a View Help Desk Complaints Request, the system retrieves and displays complaints lodged in the help desk.

   - Complaint details include complaint ID, type of complaint, description, and current status.

2. Solving a Complaint:

   - When a Solve Complaint Request is received, the system marks the specified complaint as solved.

   - This action changes the status of the complaint and indicates that it has been successfully resolved.

   - The system sends a Complaint Solved Confirmation message to acknowledge the resolution of the complaint.

3. Creating a List of Items Needed:

   - Upon receiving a Create Item List Request, the system analyzes the complaint and creates a list of items needed based on the demand specified in the complaint.

   - For example, if a complaint mentions the need for a chair, desk, and bed, the system generates a list including these items.

   - The created item list is provided as Item List Created.

4. Viewing Resolved Complaints:

   - When a View Resolved Complaints Request is received, the system retrieves and displays a list of previously resolved complaints.

   - This allows administrators to track the history of complaints and their resolutions for reference.

5. User Interaction:

   - Hostel administrators or staff access the admin side complaint management functionality through a web interface.

   - They can view, solve, and create item lists based on complaints to address issues raised by hostel residents.

- The system provides confirmation messages and organized interfaces to facilitate efficient complaint resolution and management**.**

Manage Admins

Input:

- View Admins List Request: Request to view the list of admins currently present.
- Add Admin Request: Request to add a new admin.
- Remove Admin Request: Request to remove an existing admin.

Output:

- Admins List: List of admins currently present.
- Admin Added Confirmation: Confirmation message indicating successful addition of a new admin.
- Admin Removed Confirmation: Confirmation message indicating successful removal of an admin.

Method Description:

1. Viewing List of Admins:

   - Upon receiving a View Admins List Request, the system retrieves and displays the list of admins currently present in the hostel management system.
   - Admin details may include names, contact information, roles, etc.

2. Adding a New Admin:

   - When an add Admin Request is received, the system prompts the administrator to fill in details of the new admin.
   - Once the details are provided, the system adds the new admin to the list of admins.
   - The system sends an admin Added Confirmation message to acknowledge the successful addition of the new admin.

3. Removing an Admin:

- Upon receiving a remove Admin Request, the system identifies the admin to be removed based on the provided input.

- The system removes the specified admin from the list of admins.

- An admin Removed Confirmation message is sent to acknowledge the successful removal of the admin.

4. User Interaction:

- Hostel administrators or authorized staff access the admin management functionality through a web interface.

- They can view the list of admins, add new admins, and remove existing ones as necessary to manage access and roles effectively.

- The system provides confirmation messages to ensure that actions such as addition or removal of admins are successfully executed and communicated to the user.

Manage Hostels & Rooms

Input:

- View Hostels List Request: Request to view the list of hostels.
- Manage Rooms Request: Request to manage rooms within a hostel.
- Add Room Request: Request to add a new room to a hostel.
- Remove Room Request: Request to remove an existing room from a hostel.
- Enter Hostel Details Request: Request to enter details of a hostel, such as name and number of rooms.
- View Occupancy Request: Request to view occupancy status of rooms in a hostel.
- Control Room Allocation Request: Request to control the room allocation process.

Output:

- Hostels List: List of hostels currently managed in the system.
- Room Management Interface: Interface for managing rooms within a hostel.

- Room Added Confirmation: Confirmation message indicating successful addition of a new room.

- Room Removed Confirmation: Confirmation message indicating successful removal of a room.

- Hostel Details Entered Confirmation: Confirmation message indicating successful entry of hostel details.

- Occupancy Status: Occupancy status of rooms in a hostel.

- Control options for the room allocation process.

Method Description:

1. Viewing List of Hostels:

   - Upon receiving a View Hostels List Request, the system retrieves and displays the list of hostels currently managed in the hostel management system.

   - Hostel details may include names, locations, and other relevant information.

2. Managing Rooms:

   - When a Manage Rooms Request is received, the system provides an interface for managing rooms within a selected hostel.

   - This interface allows administrators to add or remove rooms, view occupancy status, and control room allocation processes.

3. Adding a New Room:

   - Upon receiving an Add Room Request, the system prompts the administrator to enter details of the new room, such as room number, capacity, and amenities.

   - Once the details are provided, the system adds the new room to the selected hostel.

   - A Room Added Confirmation message is sent to acknowledge the successful addition of the new room.

4. Removing an Existing Room:

   - When a Remove Room Request is received, the system identifies the room to be removed based on the provided input.

- The system removes the specified room from the selected hostel.

- A Room Removed Confirmation message is sent to acknowledge the successful removal of the room.

5. Entering Hostel Details:

- Upon receiving an Enter Hostel Details Request, the system prompts the administrator to enter details of the hostel, such as name and number of rooms.

- Once the details are provided, the system adds the hostel to the list of managed hostels.

- A Hostel Details Entered Confirmation message is sent to acknowledge the successful entry of hostel details.

6. Viewing Occupancy Status:

- When a View Occupancy Request` is received, the system retrieves and provides the occupancy status of rooms in the selected hostel.

- Occupancy status includes information about occupied and vacant rooms.

7. Controlling Room Allocation Process:

- Upon receiving a control Room Allocation Request, the system provides control options for managing the room allocation process.

- This may include functionalities such as starting or stopping the allocation process, adjusting allocation criteria, etc.

8. User Interaction:

- Hostel administrators or authorized staff interact with the hostel and room management functionalities through a web interface.

- They can view, add, and remove rooms, enter hostel details, view occupancy status, and control the room allocation process as necessary.

- The system provides confirmation messages to ensure that actions such as addition or removal of rooms, and hostel details entry are successfully executed and communicated to the user.

Admin Side Room Allocation

Input:

- Empty All Rooms Request`: Request to empty all rooms in the hostel.
- Lock Rooms Request: Request to keep certain rooms locked or assigned.
- Start Room Allocation Request: Request to start the room allocation process.
- End Room Allocation Request: Request to end the room allocation process.
- Save Data Request: Request to save data related to room allocation.
- Control options for room allocation.

Output:

- Confirmation messages for successful execution of each action.

Method Description:

1. Emptying All Rooms:

   - Upon receiving an Empty All Rooms Request, the system empties all rooms in the hostel, clearing them of any current occupants.
   - This action prepares the rooms for the allocation process.
   - A confirmation message is sent to acknowledge the successful emptying of all rooms.

2. Locking or Assigning Rooms:

   - When a Lock Rooms Request is received, the system keeps certain rooms locked or assigns them to specific occupants, if necessary.
   - This ensures that these rooms are not included in the allocation process or are reserved for particular purposes.
   - A confirmation message is sent to acknowledge the successful locking or assignment of rooms.

3. Starting Room Allocation:

- Upon receiving a Start Room Allocation Request, the system initiates the room allocation process.

- This process involves assigning available rooms to residents based on predefined criteria or user preferences.

- A confirmation message is sent to acknowledge the start of the room allocation process.

4. Ending Room Allocation:

- When an End Room Allocation Request is received, the system concludes the room allocation process.

- This action finalizes room assignments and updates the database with the allocated rooms.

- A confirmation message is sent to acknowledge the end of the room allocation process.

5. Saving Data:

- Upon receiving a Save Data Request, the system saves relevant data related to the room allocation process, ensuring that changes are persisted.

- This includes updating the database with allocated rooms, occupancy status, and any other relevant information.

- A confirmation message is sent to acknowledge the successful saving of data.

6. Control Options for Room Allocation:

- Control options allow administrators to manage the room allocation process effectively.

- These options may include starting or ending the allocation process, adjusting allocation criteria, or reviewing allocation results.

- Confirmation messages provide feedback to administrators regarding the execution of these control options.

7. User Interaction:

- Hostel administrators or authorized staff interact with the room allocation functionality through a web interface.

- They can initiate, monitor, and control the room allocation process as necessary to ensure efficient management of hostel accommodations.

- The system provides confirmation messages to communicate the outcomes of various actions and maintain transparency in the allocation process.

Admin Login

Input:

- view All Student Details Request : Request to view details of all students.
- view Room Occupancy Request : Request to view room occupancy status.
- control Room Allocation Request: Request to control room allocation process.
- manage Admin Account Request: Request to manage admin account.
- resolve Student Complaint Request: Request to resolve student complaints.
- Credentials for authentication and login.

Output:

- Details of all students.
- Room occupancy status.
- Control options for room allocation.
- Interface for managing admin account.
- Confirmation messages for successful resolution of student complaints or authentication.
- Updated admin account details.

Method Description:

1. Viewing Details of All Students:

   - Upon receiving a View All Student Details Request, the system retrieves and displays details of all students currently enrolled in the hostel.
   - Student details typically include names, contact information, room numbers, etc.

2. Viewing Room Occupancy:

   - When a view Room Occupancy Request is received, the system retrieves and presents the current occupancy status of rooms in the hostel.

- This includes information about occupied and vacant rooms.

3. Controlling Room Allocation:

   - Upon receiving a control Room Allocation Request, the system provides control options for managing the room allocation process.

   - These options may include starting or stopping the allocation process, adjusting allocation criteria, etc.

4. Managing Admin Account:

   - When a manage Admin Account Request is received, the system provides an interface for managing the admin account.

   - This interface allows the admin to update their account details, change passwords, or perform other account-related actions.

5. Resolving Student Complaints:

   - Upon receiving a resolve Student Complaint Request, the system allows the admin to view and resolve student complaints.

   - This may involve reviewing complaint details, taking necessary actions to address the issues raised, and updating the status of complaints accordingly.

6. Credential Check and Authentication:

   - The system checks the provided credentials for authentication during the login process.

   - If the credentials are valid, the admin is granted access to the admin panel.

   - If authentication fails, an appropriate error message is displayed.

7. Making Changes as Necessary:

   - After successful authentication, the admin can make changes to student details, room allocations, complaint resolutions, etc., as necessary.

   - The system processes these changes and updates the database accordingly.

8. User Interaction:

   - Admins interact with the hostel management system through a web interface.

- They can access various functionalities such as viewing student details, managing room allocations, resolving complaints, etc., by navigating through the admin panel.

- The system provides confirmation messages and feedback to ensure smooth operation and effective management of hostel affairs.

Student Login

Input:

- view Personal Details Request : Request to view personal details.
- register Complaint Request: Request to register a complaint.
- Navigation to desired interface.
- view Leave Details Request: Request to view leave details.

Output:

- Personal details of the student.
- Confirmation message for successful registration of a complaint.
- Navigation to the desired interface.
- Leave details of the student.

Method Description:

1. Viewing Personal Details:

   - Upon receiving a view Personal Details Request, the system retrieves and displays the personal details of the student, such as name, contact information, room number, etc.

   - These details provide the student with information about their profile within the hostel management system.

2. Registering a Complaint:

   - When a register Complaint Request is received, the system allows the student to register a complaint.

- The student can provide details of the complaint, such as type of issue, description, and relevant information.

- Upon successful registration, a confirmation message is displayed to acknowledge the submission of the complaint.

3. Navigating to Desired Interface:

- The student can navigate to the desired interface within the hostel management system to access various functionalities.

- This may include viewing room details, checking leave status, accessing maintenance requests, etc.

- The system facilitates smooth navigation to the desired interface based on the student's selection.

4. Viewing Leave Details:

- Upon receiving a view Leave Details Request, the system retrieves and displays leave details of the student.

- Leave details may include information about the types of leave (e.g., medical leave, vacation leave), leave duration, and any remarks or notes associated with the leave requests.

5. User Interaction:

- Students interact with the hostel management system through a web interface accessible via their login credentials.

- They can view personal details, register complaints, navigate to desired interfaces, and view leave details as necessary.

- The system provides a user-friendly experience with intuitive navigation and clear feedback messages to ensure efficient communication and interaction.

User Login

Input:

- user Type: Type of user (student or admin).

- `username`: User's username.

- `password`: User's password.

Output:

- Access to the dashboard for valid credentials.

- Error message for invalid credentials, prompting the user to re-enter their username and password.

Method Description:

1. Entering as Student or Admin:

   - The user selects whether they are logging in as a student or an admin.

   - This determines the functionalities and access levels available to the user upon successful login.

2. Entering Username and Password:

   - The user enters their username and password into the login form.

   - These credentials are used to authenticate the user and grant access to the dashboard.

3. Validating Credentials:

   - The system validates the entered username and password against the stored credentials in the database.

   - If the credentials are valid, the user is granted access to the dashboard corresponding to their role (student or admin).

   - If the credentials are invalid, an error message is displayed, prompting the user to re-enter their username and password.

4. Accessing the Dashboard:

   - Upon successful validation of credentials, the user gains access to the dashboard.

   - The dashboard provides access to various functionalities and features based on the user's role and permissions.

5. Error Handling:

- If the entered credentials are incorrect, the system displays an error message indicating that the login attempt failed.

- The user is prompted to re-enter their username and password to try logging in again.

6. User Interaction:

- Users interact with the login system through a web interface accessible via a browser.

- They enter their credentials into the login form and navigate through the authentication process to gain access to the dashboard.

- The system provides feedback messages to guide users through the login process and ensure secure access to the hostel management system.

Student Side Attendance

Input:

- view Own Attendance Request: Request to view attendance of oneself.
- view Own Entry Exit Records Request`: Request to view campus entry and exit records of oneself.
- view Own Leave Details Request`: Request to view leave details and duration of oneself.
- view All Students Attendance Request : Request to view attendance of all students.

Output:

- Attendance record of oneself.
- Campus entry and exit records of oneself.
- Leave details and duration of oneself.
- Attendance records of all students.

Method Description:

1.Viewing Own Attendance:

- Upon receiving a view Own Attendance Request, the system retrieves and displays the attendance record of the requesting student.

- The attendance record typically includes dates, attendance status (present or absent), and any remarks or notes.

2. Viewing Own Entry and Exit Records:

- When a view Own Entry Exit Records Request is received, the system fetches and presents the campus entry and exit records of the requesting student.

- Entry and exit records include timestamps indicating the time when the student entered or exited the hostel premises.

3.Viewing Own Leave Details:

- Upon receiving a view Own Leave Details Request, the system retrieves and provides leave details and duration of the requesting student.

- Leave details include leave types (such as medical leave, vacation leave), start and end dates of leave, and the duration of each leave period.

4. Viewing Attendance of All Students:

- When a view All Students Attendance Request is received, the system retrieves and displays the attendance records of all students.

- This allows the requesting student to view the attendance status of their peers and compare it with their own.

5. User Interaction:

- Students interact with the attendance system through a web interface accessible via their login credentials.

- They can view their own attendance, entry and exit records, leave details, and attendance of all students as necessary.

- The system provides a user-friendly interface with intuitive navigation and clear presentation of data to facilitate efficient monitoring of attendance-related information.

Student Side Room Allocation

Input:

- Register Room Request: Request to register for a room allocation, along with preferred floor and hostel.

- Friends List: List of friends to be accommodated in the same room (maximum of 6).

- Participation in the room allocation process.

Output:

- Confirmation message for successful registration of room allocation.

- Participation in the room allocation process.

Method Description:

1. Registering for Room Allocation:

   - Upon receiving a Register Room Request, the student provides their preferences for room allocation, including preferred floor and hostel.

   - The student also specifies a list of friends to be accommodated in the same room, with a maximum limit of 6 friends.

   - This information is submitted to the system for processing.

2. Participating in Room Allocation Process:

   - The student participates in the room allocation process, during which the system considers their preferences and attempts to assign them to a suitable room.

   - The system takes into account factors such as floor preference, availability of rooms, and accommodation of friends within the same room.

   - The student may receive notifications or updates regarding their room allocation status during the process.

3. User Interaction:

   - Students interact with the room allocation system through a web interface accessible via their login credentials.

   - They provide their preferences for room allocation and participate in the allocation process to secure accommodation according to their preferences.

- The system provides feedback and updates to the students regarding their room allocation status, ensuring transparency and communication throughout the process.

Student Side Complaint

Input:

- view Unresolved Complaints Request request to view the list of unresolved complaints.
- Details of the student and their complaint.
- view Past Solved Complaints Request: Request to view the list of past solved complaints.
- generate Ticket Request: Request to generate a ticket in the help desk.

Output:

- List of unresolved complaints.
- List of past solved complaints.
- Confirmation message for successful submission of a complaint.
- Confirmation message for successful generation of a ticket.

Method Description:

1. Viewing List of Unresolved Complaints:

   - Upon receiving a view Unresolved Complaints Request, the system retrieves and displays the list of unresolved complaints that require attention.

   - This list includes details of each complaint such as complaint ID, type, description, and status.

2. Filling Details & Complaint:

   - The student fills in their personal details and provides details of the complaint they wish to register.

- This may include the type of issue, description, and any relevant information to help address the complaint effectively.

3. Viewing List of Past Solved Complaints:

   - When a view Past Solved Complaints Request is received, the system retrieves and presents the list of past complaints that have been successfully resolved.

   - Students can review these past complaints for reference and to track the resolution process.

4. Generating Ticket in Help Desk:

   - Upon receiving a generate Ticket Request, the system generates a ticket in the help desk to register the student's complaint formally.

   - The ticket includes all the details provided by the student and is assigned a unique identifier for tracking purposes.

   - A confirmation message is sent to the student to acknowledge the successful generation of the ticket.

5. User Interaction:

   - Students interact with the complaint system through a web interface accessible via their login credentials.

   - They can view unresolved complaints, submit their own complaints, view past solved complaints, and generate tickets in the help desk as necessary.

   - The system provides confirmation messages and feedback to ensure that complaints are registered and addressed efficiently.

Execution architecture

Client-Side Interface:

The user interacts with the system through a web browser.

HTML, CSS, and JavaScript are used to create the user interface.

Web Server:

Hosts the hostel management website and serves web pages to clients.

Apache HTTP Server or Nginx can be used as the web server.

Application Logic:

Implemented using server-side scripting languages like HTML, CSS, React and JavaScript (Node.js).

Handles user requests, processes data, and generates dynamic content.

Interacts with the database to retrieve and store information.

Database:

Stores data related to students, rooms, bookings, payments, etc.

MySQL or PostgreSQL can be used as the relational database management system.

Authentication and Authorization:

Manages user authentication and authorization.

Ensures secure access to the system's functionalities.

Sessions or tokens can be used for authentication.

External Services Integration (Optional):

Integration with external services like payment gateways, email services, or SMS gateways can be implemented.

APIs provided by these services can be utilized for integration.

Caching (Optional):

Caching frequently accessed data to improve performance. Simple caching mechanisms using in-memory storage like Redis can be employed.

Security Measures:

Implement security best practices such as input validation, data sanitization, and protection against common web vulnerabilities.

This architecture is more focused on the website aspect of the hostel management system, providing a basic framework for its development and deployment.

Design decisions and trade-offs

Design decisions and trade-offs play a crucial role in developing a hostel management system website. Here are some key design decisions and associated trade-offs:

Choice of Programming Language and Framework:

Decision: Choosing the programming language and framework for building the website's backend logic.

Trade-offs:

HTML and CSS is commonly used for web development due to its ease of use and widespread support. However, it may lack certain modern features and performance optimizations compared to newer languages like JavaScript (Node.js).

It provides a robust framework with built-in security features and scalability options, but it may have a steeper learning curve for developers.

Node.js offers non-blocking I/O, making it suitable for handling concurrent requests efficiently, but it may require more effort to manage asynchronous code.

Database Selection:

Decision: Choosing the type of database (SQL vs. NoSQL) and specific database management system (e.g., MySQL).

Trade-offs:

SQL databases offer strong consistency, ACID compliance, and better support for complex queries, making them suitable for relational data models typically found in hostel management systems. However, they may face scalability challenges with high-volume read and write operations.

NoSQL databases provide greater flexibility, scalability, and performance for certain use cases but may sacrifice consistency and transaction support.

User Authentication and Authorization:

Decision: Determining the method of user authentication and authorization.

Trade-offs:

Session-based authentication provides simplicity and is suitable for smaller-scale applications but may require server-side storage and management of session data.

Token-based authentication (e.g., JWT) eliminates the need for server-side session storage and enables stateless authentication, improving scalability.

However, it may require additional effort to handle token expiration and refresh mechanisms.

Integration with External Services:

Decision: Integrating external services such as payment gateways, email services, or SMS gateways.

Trade-offs:

Integrating third-party services can enhance functionality and user experience but may introduce dependencies and potential points of failure.

Custom-built solutions provide more control and flexibility but require additional development effort and ongoing maintenance.

Caching Strategy:

Decision: Implementing caching mechanisms to improve performance.

Trade-offs:

Caching frequently accessed data can reduce database load and improve response times but may introduce complexity and overhead in managing cache invalidation and consistency.

Over-caching or caching stale data can lead to inconsistencies and potential security vulnerabilities.

Security Measures:

Decision: Implementing security measures to protect against common web vulnerabilities.

Trade-offs:

Strong security measures such as input validation, encryption, and access controls are essential for protecting sensitive data and preventing attacks.

However, overly restrictive security measures may impact user experience or system performance if they introduce excessive validation overhead or authentication barriers.

Scalability and Performance Optimization:

Decision: Architecting the system for scalability and optimizing performance.

Trade-offs:

Implementing scalable architecture with load balancing, horizontal scaling, and distributed caching can ensure the system can handle increased traffic and growing data volumes.

However, designing for scalability may add complexity and overhead, requiring careful planning and ongoing monitoring to ensure optimal performance.

Overall, design decisions should be made based on the specific requirements, constraints, and priorities of the hostel management system website, considering factors such as functionality, usability, performance, scalability, and security. It's essential to weigh the trade-offs carefully to achieve the right balance for the project

6.0 Pseudocode for components

1) Class User (Method—login)

Pseudocode:

```
 Method login(username: string, password: string) -> bool
// This method will check if the provided username and password are valid.
// If valid, it returns true indicating successful login, otherwise false.

// Check if username exists in the database
IF username exists in database THEN
   // Retrieve stored password associated with username
   Stored Password = getStoredPassword(username)

   // Compare provided password with stored password
   IF password == stored Password THEN
      RETURN true  // Login successful
   ELSE
      RETURN false  // Password incorrect
ELSE
   RETURN false  // Username does not exist

 End Method
```

2) Class Admin (Method manage Hostels)

Pseudocode: Class Admin:

```
// Represents an administrator in the hostel management system
```

```
Method manage Hostels():
    // Allows the admin to perform actions related to hostels
    Display "Welcome, Admin! Choose an action:"
    Display "1. View all hostels"
    Display "2. Add a new hostel"
    Display "3. Update hostel details"
    Display "4. Delete a hostel"
    Display "5. Exit"

    Input choice
    SWITCH choice:
        CASE 1:
            // View all hostels
            Display All Hostels()
            BREAK
        CASE 2:
            // Add a new hostel
            Input hostel Name, total Rooms, total Floors, total Toilets
            Create Hostel (hostel Name, total Rooms, total Floors, total Toilets)
            Display "Hostel added successfully!"
            BREAK
        CASE 3:
            // Update hostel details
            Input hostel Id
            IF Hostel Exists(hostel Id):
                Input updated Name, updated Rooms, updated Floors, updated Toilets
                Update Hostel Details(hostel Id, updated Name, updated Rooms, updated Floors,
updated Toilets)
                Display "Hostel details updated!"
            ELSE:
                Display "Hostel not found."
            BREAK
        CASE 4:
            // Delete a hostel
            Input hostel Id
            IF Hostel Exists(hostel Id):
                Delete Hostel(hostel Id)
                Display "Hostel deleted successfully!"
            ELSE:
                Display "Hostel not found."
            BREAK
        CASE 5:
            // Exit
            Display "Exiting admin panel."
            BREAK
        DEFAULT:
            Display "Invalid choice. Please select a valid option."
            BREAK

Method Display All Hostels():
    // Retrieves and displays information about all hostels
    // Implementation details: Fetch data from the database and show it

Method Create Hostel(name, rooms, floors, toilets):
    // Adds a new hostel to the system
    // Implementation details: Insert new hostel data into the database

Method Update Hostel Details(id, name, rooms, floors, toilets):
    // Updates details of an existing hostel
    // Implementation details: Update hostel data in the database

  Method Delete Hostel(id):
    // Deletes a hostel from the system
    // Implementation details: Remove hostel data from the database

  Method Hostel Exists(id):
    // Checks if a hostel with the given ID exists
    // Implementation details: Query the database for hostel existence
    RETURN true or false
```

3)class admin (Method Manage Rooms):

Pseudocode: Class Admin:

```
  // Represents an administrator in the hostel management system

  Method manage Rooms():
    // Allows the admin to perform actions related to rooms
    Display "Welcome, Admin! Choose an action:"
    Display "1. View all rooms"
    Display "2. Add a new room"
    Display "3. Update room details"
    Display "4. Delete a room"
    Display "5. Exit"

    Input choice
    SWITCH choice:
      CASE 1:
        // View all rooms
        Display All Rooms()
        BREAK
    CASE 2:
        // Add a new room
        Input hostel Id, room Number, capacity, floor
        Create Room(hostel Id, room Number, capacity, floor)
        Display "Room added successfully!"
        BREAK
    CASE 3:
        // Update room details
        Input room Id
        IF Room Exists(room Id):
            Input updated Number, updated Capacity, updated Floor
            Update Room Details(room Id, updated Number, updated Capacity, updated
Floor)
            Display "Room details updated!"
        ELSE:
            Display "Room not found."
        BREAK
    CASE 4:
        // Delete a room
        Input room Id
        IF Room Exists(room Id):
            Delete Room(room Id)
            Display "Room deleted successfully!"
        ELSE:
            Display "Room not found."
        BREAK
    CASE 5:
        // Exit
```

```
            Display "Exiting admin panel."
            BREAK
        DEFAULT:
            Display "Invalid choice. Please select a valid option."
            BREAK

Method Display All Rooms():
    // Retrieves and displays information about all rooms
    // Implementation details: Fetch data from the database and show it

Method Create Room(hostel Id, room Number, capacity, floor):
    // Adds a new room to the system
    // Implementation details: Insert new room data into the database

Method Update Room Details(room Id, room Number, capacity, floor):
    // Updates details of an existing room
    // Implementation details: Update room data in the database

Method Delete Room(room Id):
    // Deletes a room from the system
    // Implementation details: Remove room data from the database

Method Room Exists(room Id):
    // Checks if a room with the given ID exists
    // Implementation details: Query the database for room existence
    RETURN true or false
```

4) Class admin (Method manage complains)

Pseudocode: Class Admin:

```
// Represents an administrator in the hostel management system

Method manage Complaints():
  // Allows the admin to handle complaints
  Display "Welcome, Admin! Choose an action:"
  Display "1. View all complaints"
  Display "2. Mark a complaint as resolved"
  Display "3. Delete a complaint"
  Display "4. Exit"

  Input choice
  SWITCH choice:
    CASE 1:
      // View all complaints
      Display All Complaints()
      BREAK
    CASE 2:
      // Mark a complaint as resolved
      Input complaint Id
      IF Complaint Exists(complaint Id):
        Mark Complaint Resolved(complaint Id)
        Display "Complaint marked as resolved!"
      ELSE:
        Display "Complaint not found."
      BREAK
    CASE 3:
      // Delete a complaint
      Input complaint Id
      IF Complaint Exists(complaint Id):
        Delete Complaint(complaint Id)
        Display "Complaint deleted successfully!"
      ELSE:
        Display "Complaint not found."
      BREAK
    CASE 4:
      // Exit
      Display "Exiting admin panel."
      BREAK
    DEFAULT:
      Display "Invalid choice. Please select a valid option."
      BREAK

Method Display All Complaints():
  // Retrieves and displays information about all complaints
```

```
    // Implementation details: Fetch data from the database and show it

  Method Mark Complaint Resolved(complaint Id):
    // Marks a complaint as resolved
    // Implementation details: Update complaint status in the database

  Method Delete Complaint(complaint Id):
    // Deletes a complaint from the system
    // Implementation details: Remove complaint data from the database

  Method Complaint Exists(complaint Id):
    // Checks if a complaint with the given ID exists
    // Implementation details: Query the database for complaint existence
    RETURN true or false
```

5) Class student (method viewStatus)

Pseudocode: Class Student:

```
  // Represents a student in the hostel management system

  Method view Status():
    // Displays the status of the student's room and other relevant information
    Display "Welcome, Student! Here is your room status:"
    Display "Hostel Name: ABC Hostel"
    Display "Room Number: 203"
    Display "Floor: 2"
    Display "Capacity: 4"
    Display "Occupancy: 3 (1 vacant bed)"
    Display "Complaints: 2 pending"
    Display "Attendance: 85%"

    // Additional logic to retrieve and display other relevant data
    // Implementation details: Fetch data from the database and show it
```

6) Class Student (Method view Rooms)

Pseudocode: Class Student:

```
  // Represents a student in the hostel management system

  Method view Rooms():
    // Displays information about available rooms
    Display "Available Rooms:"
    FOR EACH room IN Hostel.getAvailableRooms():
      Display "Room Number:", room.roomNumber
      Display "Floor:", room.floor
      Display "Capacity:", room.capacity
      Display "Vacant Beds:", room.vacantBeds
      Display "------------------------"

    // Additional logic to retrieve and display other relevant data
    // Implementation details: Fetch data from the database and show it
```

7) Class student (Method create Complain)

Pseudocode:

```
Class Student:
  // Represents a student in the hostel management system

  Method create Complaint(category: string, description: string):
    // Allows the student to create a new complaint
    // Parameters: category (type of complaint), description (details of the issue)
    Create New Complaint(category, description)
    Display "Complaint submitted successfully!"
    // Additional logic to save the complaint in the database
    // Implementation details: Insert new complaint data into the database
```

8) Class Dashboard (Method display Status)

Pseudo Code:

```
Class Dashboard:
  // Represents the dashboard in the hostel management system

  Method display Status():
    // Displays overall system status and statistics
    Display "Hostel Management System Dashboard"
    Display "---------------------------------"
    Display "Total Hostels: 5"
    Display "Total Rooms: 150"
    Display "Occupied Rooms: 120"
    Display "Vacant Rooms: 30"
    Display "Total Students: 300"
    Display "Complaints Pending: 10"
    Display "Maintenance Requests: 5"
    Display "Attendance Rate: 85%"
    // Additional logic to retrieve and display other relevant data
    // Implementation details: Fetch data from the database and show it
```

9) Class rooms (Method View Occupancy)

Pseudocode:

```
Class Room:
   // Represents a room in the hostel management system

   Method view Occupancy():
      // Displays information about room occupancy
      Display "Room Number: 203"
      Display "Floor: 2"
      Display "Capacity: 4"
      Display "Occupied Beds: 3"
      Display "Vacant Beds: 1"
      // Additional logic to retrieve and display other relevant data
      // Implementation details: Fetch data from the database and show it
```

10) Class Maintenance (Method create Complaint)

Pseudo Code:

```
Class Maintenance:
   // Represents maintenance staff in the hostel management system


Method create  Complaint(room Number: int, category: string, description:    string):

      // Allows Student to create a new complaint
      // Parameters: room Number (room where the issue occurred),    category (type
of complaint), description (details of the issue)
      Create New Complaint(room  Number, category, description)
      Display "Complaint submitted successfully!"
      // Additional logic to save the complaint in the database
      // Implementation details: Insert new complaint data into the database
```

11) Class Admin:

```
    // Represents an administrator in the hostel management system

   Method allocate Room(student Id: int, room Number: int):
      // Allocates a room to a student
      // Parameters: student Id (unique identifier for the student), room Number
(room to be allocated)

      IF StudentExists(student Id) AND Room Exists(room Number) THEN
         IF Room Is Vacant(room Number) THEN
            Allocate Room To Student(student Id, room Number)
            Display "Room allocated successfully!"
```

```
        ELSE
            Display "Room is already occupied."
        END IF
    ELSE
        Display "Invalid student ID or room number."
    END IF

Method StudentExists(student Id: int) -> bool:
    // Checks if a student with the given ID exists
    // Implementation details: Query the database for student existence
    RETURN true or false

Method Room Exists(room Number: int) -> bool:
    // Checks if a room with the given number exists
    // Implementation details: Query the database for room existence
    RETURN true or false

Method Room Is Vacant(room  Number: int) -> bool:
    // Checks if the specified room is vacant
    // Implementation details: Query the database to verify room occupancy
    RETURN true or false

Method Allocate Room To Student(student Id: int, room Number: int):
    // Updates the database to allocate the specified room to the student
    // Implementation details: Update student-room
```