

Darshan University

A Project Report on

**“Hostel Management System”**

Under the subject

**Software Engineering (2301CS405)**

B. Tech, Semester – 4th

Computer Science & Engineering Department

|  |  |
| --- | --- |
| Submitted By | |
| Student Name: Jay Vadariya | Enrolment No : 24010101702 |
| Academic Year  (2024-2025) | |
| Internal Guide  Prof. Rajkumar Gondaliya Darshan University | Dean-DIET Dr. Gopi Sanghani  Darshan University |

|  |  |
| --- | --- |
|  | **Computer Science & Engineering Department**  **Darshan University** |

# DECLARATION

We hereby declare that the SRS, submitted along with the **Software Engineering (2301CS405)** for entitled **“Hostel Management System”** submitted in partial fulfilment for the Semester-4 of **Bachelor Technology (B. Tech)** in **Computer Science and Engineering (CSE)** Department to Darshan University, Rajkot, is a record of the work carried out at **Darshan University, Rajkot** under the supervision of **Rajkumar gondaliya** and that no part of any of report has been directly copied from any students’ reports, without providing due reference.

Jay Vadariya

Student’s Signature

Date:

|  |  |
| --- | --- |
|  | **Computer Science & Engineering Department**  **Darshan University** |

# CERTIFICATE

This is to certify that the SRS on **“Hostel Management System” has** been satisfactorily prepared by **Jay Vadariya** (**24010101702**) under my guidance in the fulfillment of the course **Software Engineering (2301CS405)** work during the academic year 2024-2025.

|  |  |
| --- | --- |
| Internal Guide  Prof., Rakumar Gondaliya Darshan University | Dean-DIET  Dr. Gopi Sanghani Darshan University |

# ACKNOWLEDGEMENT

I wish to express my sincere gratitude to my project guide Prof. **R.B.Gondaliya** and all the faculty members for helping me through my project by giving me the necessary suggestions and advices along with their valuable co-ordination in completing this work.

I also thank my parents, friends and all the members of the family for their precious support and encouragement which they had provided in completion of my work. In addition to that, I would also like to mention the Darshan University personals who gave me the permission to use and experience the valuable resources required for the project from the University premises.

Thus, in conclusion to the above said, I once again thank the faculties and members of

**Darshan University** for their valuable support in completion of the project.

Thanking You

**Jay Vadariya**

# ABSTRACT

A Hostel Management System (HMS) is a comprehensive software solution designed to streamline the administration and operations of hostels, including those associated with educational institutions, workplaces, and independent accommodations. This system aims to address various challenges in hostel management, such as room allocation, fee collection, maintenance scheduling, and resident monitoring, by automating and optimizing these processes.

The HMS integrates key functionalities such as student registration, room assignments, billing and payment tracking, and inventory management into a unified platform, ensuring efficient and transparent operations. Additionally, it incorporates features for handling grievances, generating reports, and maintaining detailed records of residents and staff.

Main purpose of this system is to reduce human efforts as much as possible.

**Table of Contents**

[List of Figures I](#_bookmark0)

[List of Tables II](#_bookmark1)

1. [Introduction 1](#_bookmark2)
   1. [Product perspective: 1](#_bookmark3)
   2. [Product features 1](#_bookmark4)
      1. [There are three different users who will be using this product: 1](#_bookmark5)
      2. [The features that are required for the Warden are: 1](#_bookmark6)
      3. [The features that are required for the admin are: 1](#_bookmark7)
   3. [Functional Requirement 2](#_bookmark8)
      1. [Student 2](#_bookmark9)
      2. [Warden 2](#_bookmark10)
      3. [Admin 3](#_bookmark11)
   4. [Non-Functional Requirement 4](#_bookmark12)
      1. [Usability 4](#_bookmark13)
      2. [Accuracy 4](#_bookmark14)
      3. [Availability 4](#_bookmark15)
      4. [Maintainability 4](#_bookmark16)
2. [Design and Implementation Constraints 5](#_bookmark17)
   1. [Use case diagram 5](#_bookmark18)
   2. [Activity diagram and Swimlane diagram 6](#_bookmark20)
   3. [Sequence diagram 8](#_bookmark22)
   4. [State diagram 9](#_bookmark24)
   5. [Class diagram 10](#_bookmark26)
   6. [Data flow diagram 11](#_bookmark28)
      1. [Context diagram (level-0) 11](#_bookmark29)
      2. [DFD Level-1 12](#_bookmark31)
3. [External interface requirement (Screens) 13](#_bookmark33)
   1. [Screen-1: Register Screen 13](#_bookmark34)
   2. [Screen-2: Login Form 14](#_bookmark37)
   3. [Screen-3: Attendance 15](#_bookmark40)
   4. [Screen-4: Guest Form 16](#_bookmark43)
   5. [Screen-5: Search Available Beds 17](#_bookmark44)
4. [Database Design 18](#_bookmark45)
   1. [List of Tables 18](#_bookmark46)
5. [Stories and Scenario 20](#_bookmark51)
   1. [Story-1: Pay Hostel Fees 20](#_bookmark52)
      1. [Scenario# S1.1 20](#_bookmark53)
      2. [Scenario# S1.2 20](#_bookmark54)
      3. [Scenario# S1.3 21](#_bookmark55)
   2. [Story-3: Room Management 22](#_bookmark56)
      1. [Scenario# S2.1 22](#_bookmark57)
   3. [Story-2: View Fees Payment History 23](#_bookmark58)
      1. [Scenario# S2.1 23](#_bookmark59)
6. [Test Cases 24](#_bookmark60)
7. [References 31](#_bookmark61)

## List of Figures

[Figure 2.1-1 Use case diagram for Hostel management system 5](#_bookmark19)

[Figure 2.2-1 Activity diagram for paying fees 6](#_bookmark21)

Figure 2.2-2 Swimlane diagram for Room allocation 7

[Figure 2.3-1 Sequence diagram for Room allocation 8](#_bookmark23)

[Figure 2.4-1 State diagram of Student 9](#_bookmark25)

Figure 2.4-2 State diagram of Hostel 9

[Figure 2.5-1 Class diagram for Hostel management system 10](#_bookmark27)

[Figure 2.6-1 Context diagram for Hostel management system 11](#_bookmark30)

[Figure 2.6-2 DFD level-1 for Hostel management system 12](#_bookmark32)

Figure 2.6-3 DFD level-2 for Hostel Management 12

[Figure 3.1-1 Screen-1: Registration Form 13](#_bookmark35)

[Figure 3.2-1 Screen-2: Login Form 14](#_bookmark38)

[Figure 3.3-1 Screen-3: Attendance 15](#_bookmark41)

[Figure 3.3-2 Screen-3: Guest Form](#_bookmark41) [15](#_bookmark41)

[Figure 3.3-3 Screen-3: Search Available Beds](#_bookmark41) [15](#_bookmark41)

## List of Tables

[Table 3.1-1 Screen element of Registration form 13](#_bookmark36)

[Table 3.2-1 Screen element of Login form 14](#_bookmark39)

[Table 3.3-1 Screen element of Attendance 15](#_bookmark42)

[Table 3.4-1 Screen element of Guest Form 15](#_bookmark42)

[Table 3.5-1 Screen element of Search Available Beds 15](#_bookmark42)

[Table 4.1-1 Table: Student 18](#_bookmark47)

[Table 4.1-2 Table: Room 18](#_bookmark48)

[Table 4.1-3 Table: Staff 18](#_bookmark49)

[Table 4.1-4 Table: Fees 19](#_bookmark50)

[Table 4.1-5 Table: Attendance](#_bookmark50) [19](#_bookmark50)

## Introduction

#### Product perspective:

**The Hostel Management System (HMS) project aims to develop an integrated software solution to manage the complex and dynamic operations of hostels. The project focuses on addressing the needs of administrators, staff, and residents by providing a user-friendly and efficient platform for managing various hostel-related activities.**

###### Product features

##### There are three different users who will be using this product:

* + - * View and update personal details, emergency contacts, and medical information.
      * Check room allocation status and request room changes.
      * Pay hostel fees and view payment history

##### The features that are required for the Warden are:

* + - * Overview of hostel status (occupancy, complaints, announcements).
      * Quick access to student, staff, and room details.
      * Approve or deny student room change requests.
      * Monitor room availability, allocation, and conditions.
      * View and manage student profiles, including personal details, health information, and emergency contacts.
      * Track student attendance and curfew compliance.
      * Handle student grievances and complaints.
      * Issue warnings for curfew violations or rule-breaking incidents.
      * View, assign, and track the resolution of student complaints (e.g., maintenance, roommate issues).
      * Escalate unresolved complaints to higher authorities.
      * Monitor and record daily student attendance.

##### The features that are required for the admin are:

* + - * Overview of hostel occupancy, student details, room availability, payments, etc.
      * Allocate, swap, or vacate rooms for students.
      * Track hostel fees, fines, payment history, and dues.
      * Manage student records, admission details, and grievances.
      * Assign duties to hostel staff and manage work schedules.

#### Functional Requirement

##### Student

* + - * **User Registration**: Allow students to register by providing necessary personal and academic information.
      * **Profile Management**: Enable students to view and update their profiles, including contact details, emergency contacts, and academic information.
      * **Room Application**: Facilitate the application process for room allotment, allowing students to specify preferences.
      * **Room Assignment Notification**: Notify students of their room assignment or any changes in room allocation.
      * **Room Change Request**: Allow students to request room changes based on availability and hostel policies.

*3. Billing and Payment*

* + - * **Fee Structure Information**: Provide details on fee structures, including room rent, utilities, and other charges.
      * **Invoice Generation**: Generate invoices for accommodation fees and other charges.
      * **Online Payment**: Allow students to make payments online using various payment methods (credit card, debit card, net banking, etc.).
      * **Payment History**: Enable students to view their payment history and track outstanding dues.
      * **Maintenance Requests**: Allow students to submit maintenance requests for issues in their rooms or common areas.

##### Warden

* + - * **User Authentication and Access control** : The Warden should have a secure login mechanism.
      * **Room Allocation control**: View and manage room availability and occupancy.
      * **Resident Information control** : Maintain detailed records of all residents, including personal information, contact details, and room assignments.
      * **Billing and Payment Tracking:** Generate invoices for accommodation fees and other charges.
      * **Maintenance and Housekeeping Requests:** Receive and manage maintenance and housekeeping requests from residents ,Schedule and monitor the progress of maintenance and housekeeping activities.
      * **Grievance and Feedback control:** Receive complaints and feedback from residents. Track the status of grievances and ensure they are addressed promptly.
      * **Communication and Announcements:** Send notifications and announcements to residents, such as important notices, event reminders, or policy changes.Maintain a record of all communications for future reference.
      * **Reporting and Analytics:** Generate reports on various aspects of hostel management, including occupancy rates, financial summaries, maintenance activities, and resident feedback.
      * **Attendance and Monitoring:** Track resident attendance and monitor their presence within the hostel. Implement check-in/check-out mechanisms for residents.
      * **Disciplinary Actions:** Record and manage disciplinary actions taken against residents. Monitor compliance with hostel rules and regulations.
      * **Inventory:** Maintain records of hostel inventory, including furniture, appliances, and other assets.
      * **Visitor Management:** Record details of visitors to the hostel. Ensure compliance with visitor policies and maintain security.
      * **Emergency Management:** Implement procedures for handling emergencies, such as medical incidents or security threats.
      * **Customization and Preferences:** Customize settings and preferences for different hostels, accommodating specific needs and policies.
      * **Training and Support:** Access to training resources and support for using the HMS effectively.

##### Admin

* + - * **Add/Remove Users**: Ability to add or remove users (students, staff, etc.).
      * **User Roles and Permissions**: Assign roles and permissions to different users (e.g., admin, student, staff).
      * **Profile**: Update and manage user profiles.
      * **Room Allocation**: Allocate and reallocate rooms to students.
      * **Room Availability**: Check and manage room availability.
      * **Room Maintenance**: Track and manage maintenance requests for rooms.
      * **Fee Structure Setup**: Define and update the fee structure.
      * **Fee Collection**: Record and track fee payments from students.
      * **Fee Due Reminders**: Send reminders for due or overdue fees.
      * **Attendance Tracking**: Track and manage student attendance.
      * **Leave Requests**: Approve or reject leave requests from students.
      * **Asset control**: hostel assets and inventory.
      * **Stock Tracking**: Track stock levels and reorder supplies as needed.
      * **Log Complaints**: Allow students to log complaints.
      * **Complaint Resolution**: Track and manage the resolution of complaints.

##### 1.4.5 Manager

* **Resident Registration:** Ability to register new residents, including their personal details, room allocation, and payment information.
* **Staff :** Register and manage staff details, assign roles and responsibilities.
* **User Profiles:** Create and update profiles for both residents and staff, including personal information, contact details, and roles.
* **Room Allocation:** Assign rooms and beds to residents based on availability and preferences.
* **Room Availability:** View and update room and bed availability status.
* **Room Transfer:** Facilitate room transfers for residents, updating the system accordingly.
* **Fee Structure:** Define and update the fee structure for different room types and services.
* **Payment Tracking:** Record and track payments made by residents, generate receipts, and manage due payments.
* **Invoice Generation:** Generate invoices for residents for room rent, utilities, and other services.
* **Attendance Tracking:** Track and manage daily attendance of residents.

#### Non-Functional Requirement

##### Usability:

* + - * The UI should be simple enough for everyone to understand and get the relevant information without any special training. Different languages can be provided based on the requirements.

##### Accuracy:

* + - * The data stored about the books and the fines calculated should be correct, consistent, and reliable.

##### Availability:

* + - * The System should be available for the duration when the library operates and must be recovered within an hour or less if it fails. The system should respond to the requests within two seconds or less.

##### Maintainability:

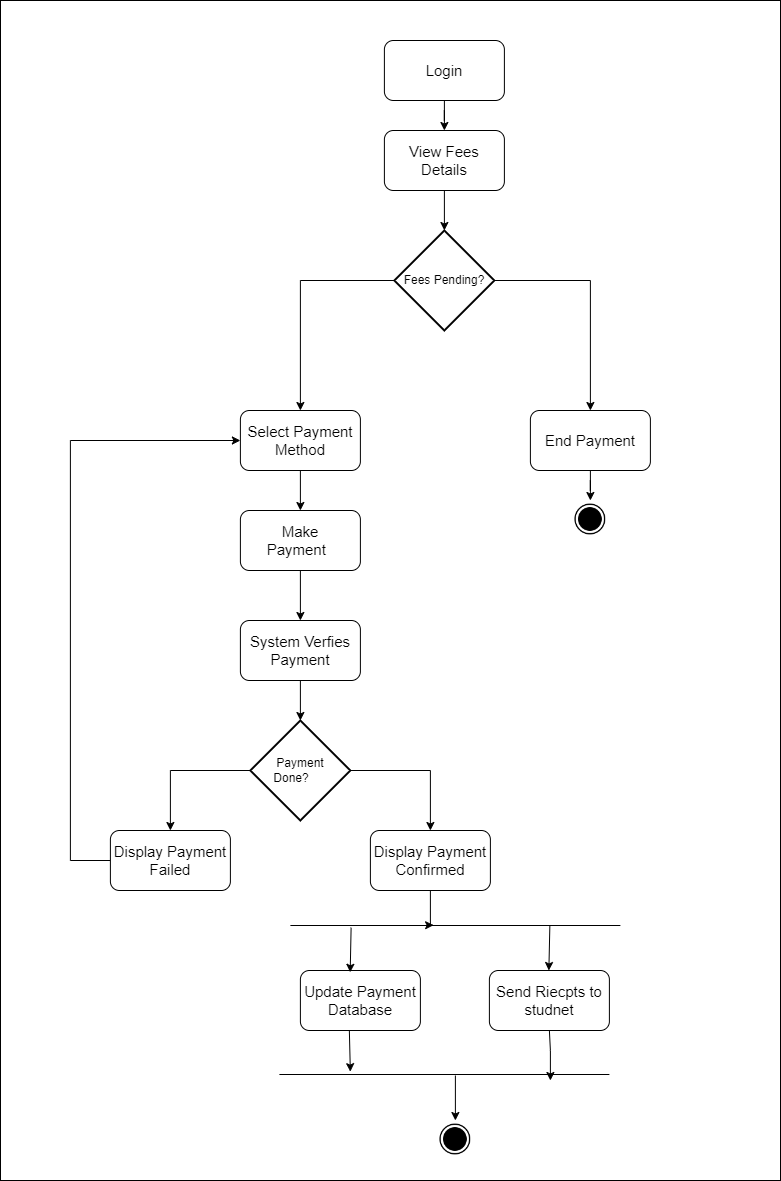
* + - * The software should be easily maintainable and adding new features and making changes o the software must be as simple as possible. In addition to this, the software must also be portable.

## Design and Implementation Constraints

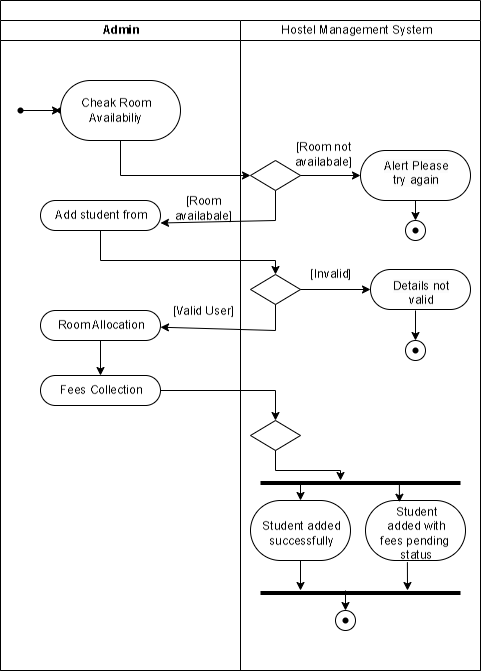
#### 

#### 2.1 Use case diagram

#### Activity diagram and Swimlane diagram

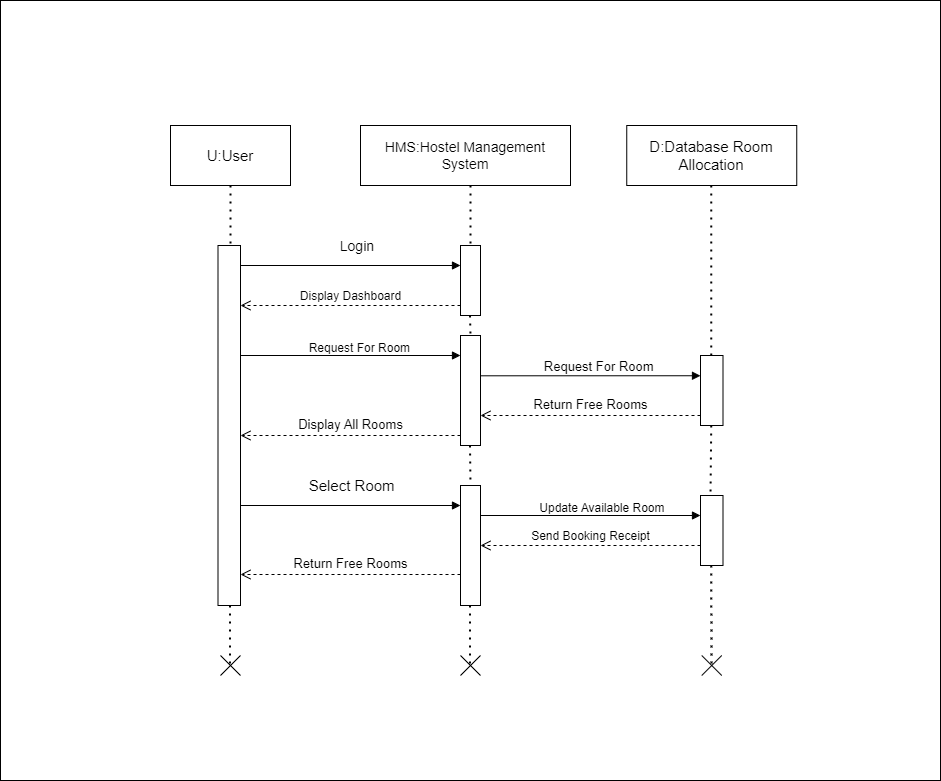


### Figure 2.2-1 Activity diagram for Paying Fees



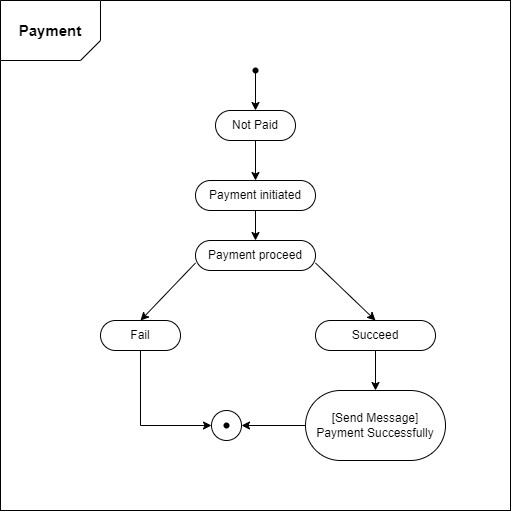
*Figure 2.2-3 Swim Len diagram for room allocation*

#### Sequence diagram



### Figure 2.3-1 Sequence diagram for Room Allocation

#### State diagram



### Figure 2.4-1 State diagram of Student

#### Class diagram

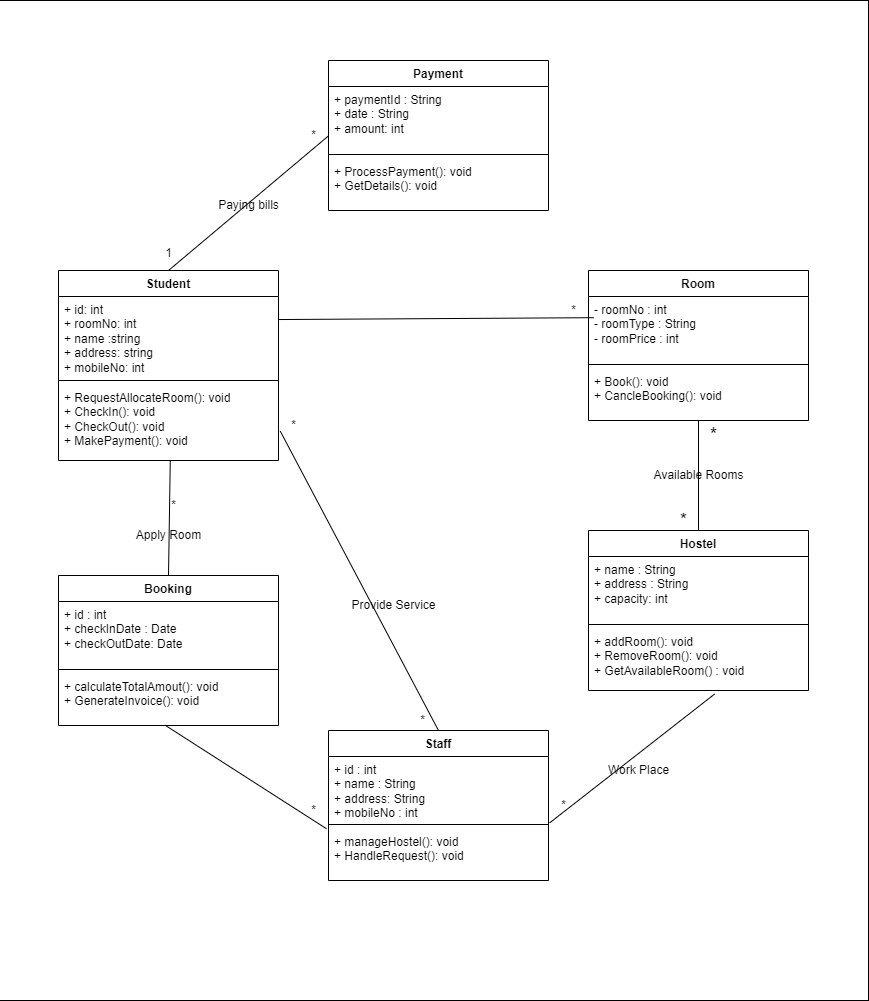
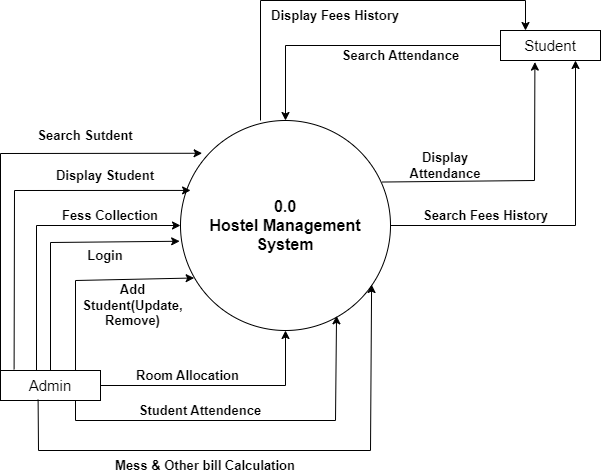


Figure 2.5-1 Class diagram for Hostel management system

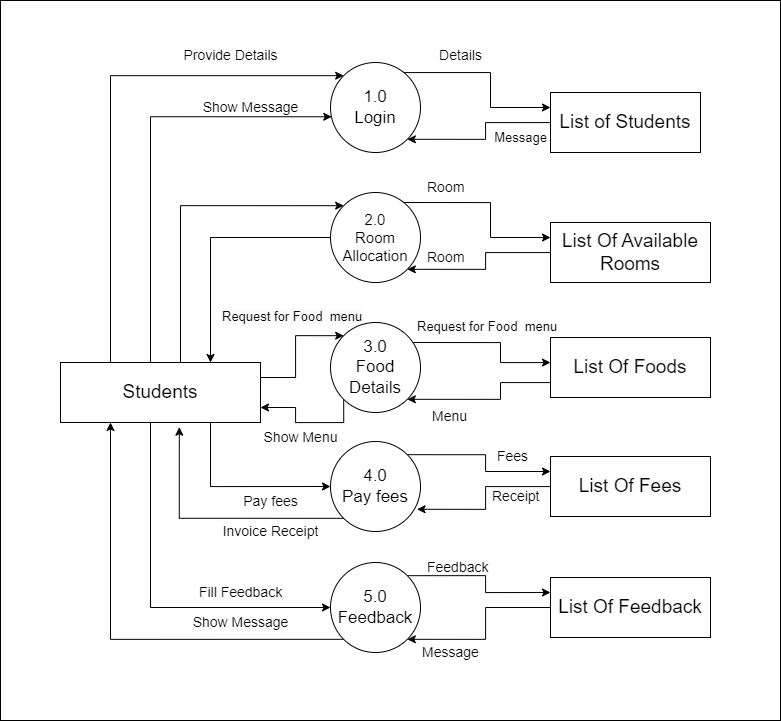
#### Data flow diagram

##### 2.5.1 Context diagram (level-0)



### Figure 2.6-1 Context diagram for Hostel management system

##### DFD Level-1



*Figure 2.6-2 DFD level-1 for Hostel management system*

## External interface requirement (Screens)

#### Screen-1: Room Application Screen

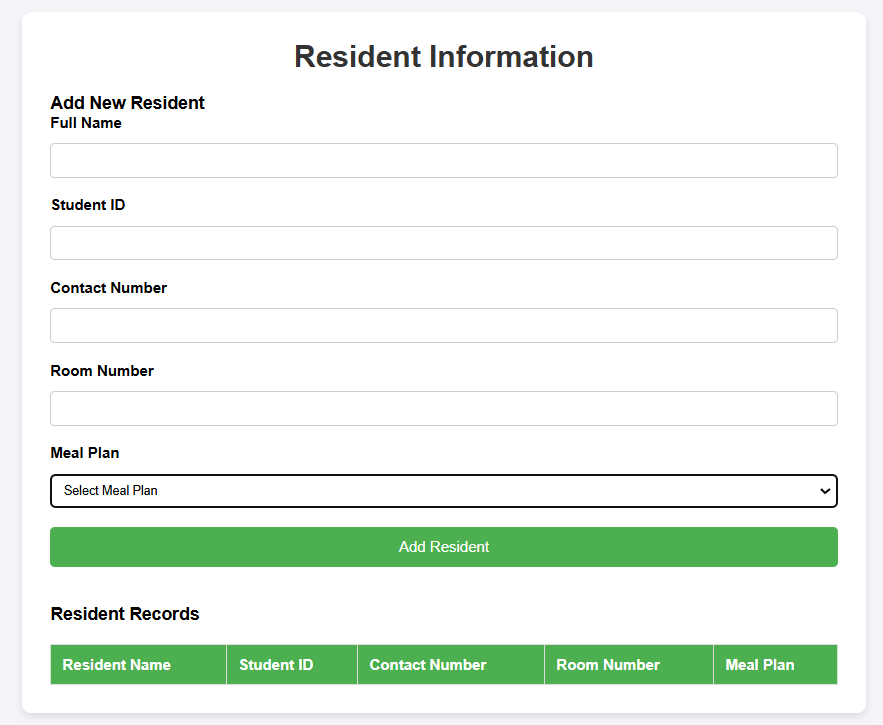
*Figure 3.1-1 Screen-1: Room Application*

**Purpose:** Allows students to apply for hostel accommodation.

*Table 3.1-1 Screen element of Room Application*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr.** | **Screen Element** | **Input Type** | **O/M** | **1/N** | **Description** |
| **1** | Full Name | Textbox | M | 1 | Enter the full name of the applicant. |
| **2** | Student ID | Textbox | M | 1 | Enter the student identification number. |
| **3** | Room Preference | Dropdown (Select) | M | 1 | Choose the preferred room type. |
| **4** | Meal Plan | Dropdown (Select) | M | 1 | Select the desired meal plan. |
| **5** | Special Requests | Text Area | M | 1 | Enter any special requests or preferences. |
| **6** | Submit Button | Button | M | 1 | Submit the form data for processing |

#### Screen-2: Resident Information

****

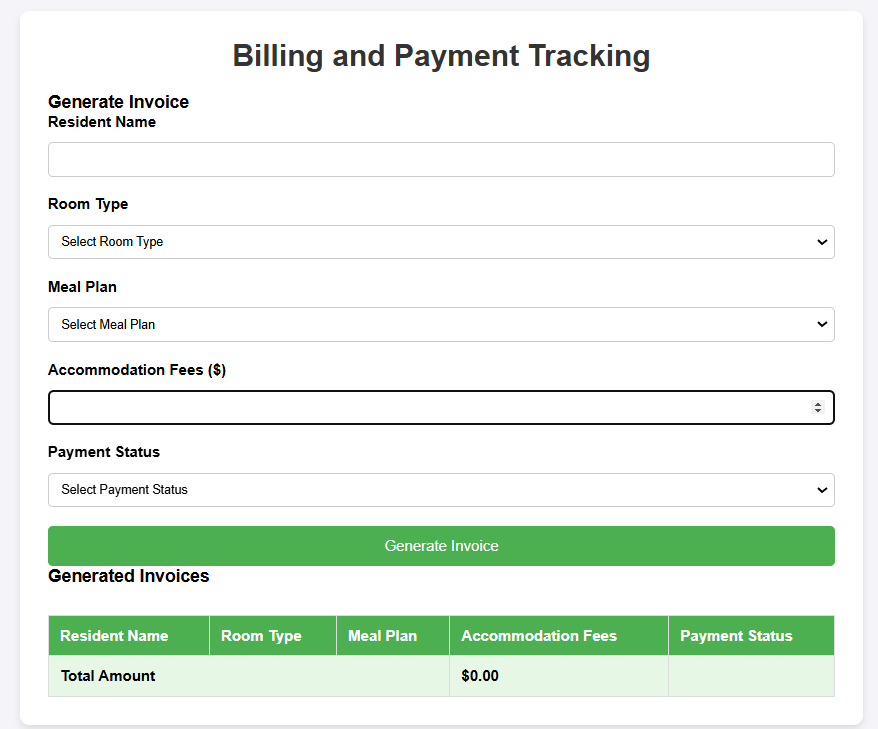
*Figure 3.2-1 Screen-2: Resident Information*

**Purpose:** Stores essential student information (name, ID, room number, contact details, guardian details, etc.).

*Table 3.2-1 Screen element of Resident Information*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr.** | **Screen Element** | **Input Type** | **O/M** | **1/N** | **Description** |
| **1** | Full Name | Textbox | M | 1 | Enter the full name of the resident. |
| **2** | Student ID | Text Input | M | 1 | Enter the unique student identification number. |
| **3** | Contact Number | Text Input | M | 1 | Enter the contact number of the resident.. |
| **4** | Room Number | Text Input | M | 1 | Required, numeric, must match room allocation. |
| **5** | Meal Plan | Dropdown (Select) | M | 1 | Select the resident's meal plan preference. |
| **6** | Add Resident Button | Button | M | 1 | Submit the form data to add a new resident record |

#### Screen-3: Billing and Payment Tracking



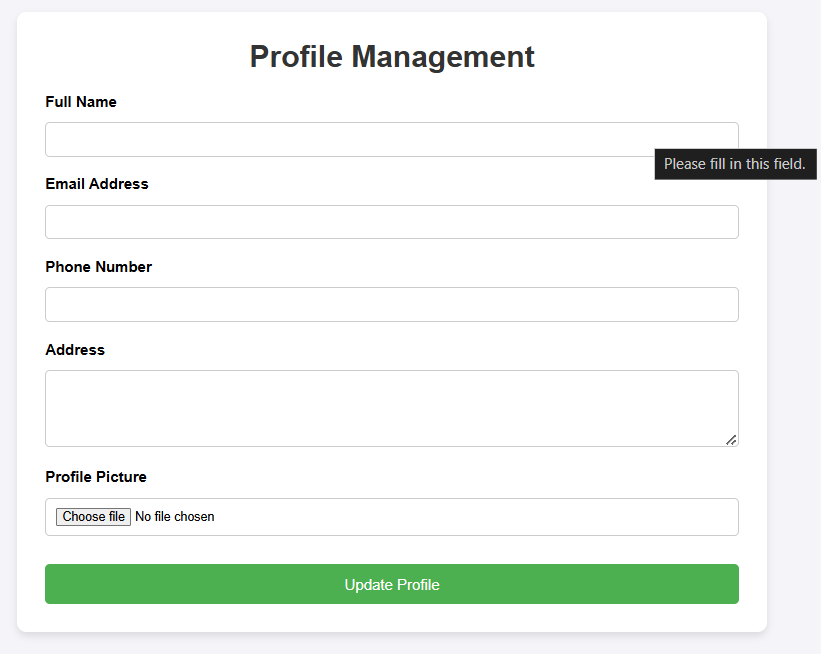
*Figure 3.3-1 Screen-3: Billing and payment*

**Purpose:** This module will allows students to view their hostel fees, due dates, and payment history.

*Table 3.3-1 Screen element of Billing and payment*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr.** | **Screen Element** | **Input Type** | **O/M** | **1/N** | **Description** |
| **1** | Resident Name | Text Input | M | 1 | Enter the name of the resident being billed |
| **2** | Room Type | Dropdown (Select) | M | 1 | Select the type of room for billing. |
| **3** | Meal Plan | Dropdown (Select) | M | N | Select the meal plan associated with billing. |
| **4** | Accommodation Fees ($) | Numeric Input | M | 1 | Enter the accommodation fees in dollars. |
| **5** | Payment Status | Dropdown (Select) | M | 1 | Select the payment status (e.g., Paid, Pending). |
| **6** | Generate Invoice Button | Button | M | 1 | Generate an invoice based on the entered details. |

#### Screen-4: Profile Management



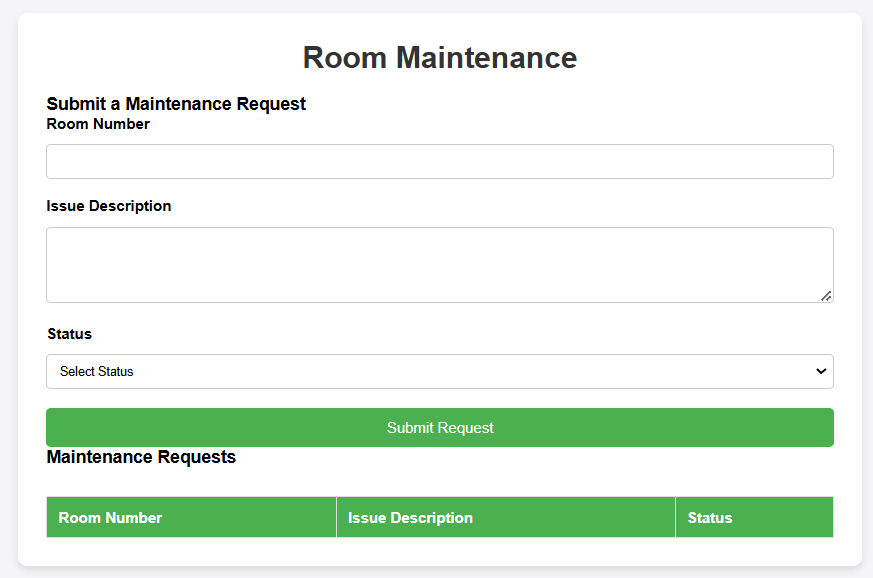
*Figure 3.4-1 Screen-4: Profile Management*

**Purpose:** This module Allows students, wardens, and admins to update and view their personal details (name, contact information, address, etc.)

*Table 3.4-1 Screen element of Profile management*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr.** | **Screen Element** | **Input Type** | **O/M** | **1/N** | **Description** |
| **1** | Cross Button | Link | M | 1 | Return to last page |
| **2** | Title | Text | M | 1 | Displays the name of the page |
| **3** | Guest Name | Textbox | M | 1 | Guest name |
| **4** | Number | Textbox | M | 1 | Phone number |
| **5** | Relation | Textbox | M | 1 | Relationship with student |
| **6** | Buttons | Link | M | N | Check in check out buttons |
| **7** | Host now | Button | M | 1 | Submit button to host |

#### Screen-5: Room Maintenance

****

*Figure 3.4-1 Screen-4: Room Maintenance*

**Purpose:** This module allows Assign rooms to students based on availability and preferences..

*Table 3.4-1 Screen element of Room maintenance*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr.** | **Screen Element** | **Input Type** | **O/M** | **1/N** | **Description** |
| **1** | Room Number | TextBox | M | 1 | Accepts room number input. |
| **2** | Issue Description | Textarea | M | 1 | Allows user to enter the issue details. |
| **3** | Status | Select Box | M | 1 | Provides options to select the issue status |
| **4** | Submit a Maintenance Request | Text Label | M | 1 | Section title for submitting a request. |
| **5** | Room Number | Textbox | M | 1 | Accepts room number input. |
| **6** | Issue Description | Textarea | M | 1 | Allows user to enter the issue details. |
| **7** | Status | Select Box | M | 1 | Provides options to select the issue status. |

## Database Design

#### List of Tables

* Student
* Room
* Staff
* Fees
* Attendance

*Table 4.1-1 Table: : Student*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column** | **Data Type** | **Null** | **Keys & Constrains** | **Default Value & Description** |
| **StudentID** | int | NN | PK (Auto Increment) | Autoincrement from 1 |
| **StudentName** | varchar(100) | NN | - | Name of the student |
| **RoomID** | Varchar(100) | NN | FK | Reference of Room Table |
| **DOB** | int | AN | - | Birth date of the student |
| **JoiningDate** | DateTime | AN | - | Date of joining |
| **Contact** | Number(10,0) | NN | - | Contact details of the student |
| **E-mail** | Varchar(50) | AN | - | Email address of the student |
| **PaymentID** | int | NN | FK | Reference of Payment Table |
| **AttandanceID** | int | AN | FK | Reference of Attendance Table |

*Table 4.1-2 Table: Room*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column** | **Data Type** | **Null** | **Keys & Constrains** | **Default Value &**  **Description** |
| **RoomID** | int | NN | PK (Auto Increment) | Autoincrement from 1 |
| **OccupacyStatus** | Boolean | NN | - | Available or not |
| **Feedback** | Varchar(50) | AN | - | Feedbacks of the room |
| **RoomType** | Varchar(50) | AN | - | Type of the room |
| **Rent** | int | AN | - | Amount of the room  selected |
| **StudentID** | int | NN | FK | Reference of Student Table |

*Table 4.1-3 Table: Staff*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column** | **Data Type** | **Null** | **Keys & Constrains** | **Default Value & Description** |
| **StaffID** | int | NN | PK (Auto Increment) | Autoincrement from 1 |
| **StaffName** | varchar(100) | AN | - | Name of the staff |
| **IsAdmin** | Boolean | AN | - | Admin or not |
| **Designation** | varchar(100) | AN | - | Designation of the staff |
| **E-mail** | Varchar(50) | AN | - | Email of the staff |
| **Conatct** | Number(10,0) | AN | - | Contact details of the staff |
| **Salary** | int | NN | - | Salary of the staff |
| **AttandanceID** | int | AN | FK | Reference of Attendance Table |

*Table 4.1-4 Table: : Fees*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column** | **Data Type** | **Null** | **Keys & Constrains** | **Default Value & Description** |
| **FeesID** | int | NN | PK (Auto Increment) | Autoincrement from 1 |
| **InvoiceNumber** | int | AN | - | Bill number |
| **FeesStatus** | Boolean | AN | - | Fees paid or pending |
| **InvoiceDate** | DateTime | AN | - | Date of the bill generated |
| **Amount** | int | NN | - | Amount of the room selected |
| **StudentID** | int | AN | FK | Reference of Student Table |

*Table 4.1-5 Table: Attendance*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column** | **Data Type** | **Null** | **Keys & Constrains** | **Default Value & Description** |
| **AttendanceID** | int | NN | PK (Auto Increment) | Autoincrement from 1 |
| **Date** | DateTime | AN | - |  |
| **Status** | Boolean | AN | - |  |
| **UpdatedAt** | DateTime | AN | - |  |
| **StudentID** | int | AN | FK | Reference of Student Table |

## Stories and Scenario

#### Story-1: Pay Hostel Fees

|  |  |  |
| --- | --- | --- |
| *Story #* ***S1*** | **:** | **As a Student**  **I want to pay my hostel fees**  **So that I can ensure my accommodation is secured and my account is up-to- date.** |
| **Priority** | **:** | High |
| **Estimate** | **:** | XL |
| **Reason** | **:** | Timely fees payment is essential for managing hostel operations and ensuring that students' accommodations are confirmed. |

##### Scenario# S1.1

|  |  |  |
| --- | --- | --- |
| *Scenario#* ***S1.1*** | **:** | Paying Hostel Fees with Valid Payment Information |
| **Prerequisite** | **:** | Student is logged in and has an outstanding fees balance. |
| **Acceptance Criteria** | **:** | **Given:** The student is on the fees payment page.Valid book information, including title, author, ISBN, and other relevant details is added.  **When:**  The student selects the "Pay Fees" option.  The student enters valid payment details (e.g., credit card number, expiration date, CVV).  The student clicks the "Submit Payment" button.  **Then:**  The system processes the payment successfully.  The student receives a confirmation message with a receipt for the payment. |

##### Scenario# S1.2

|  |  |  |
| --- | --- | --- |
| *Scenario#* ***S1.2*** | **:** | Paying Hostel Fees with Invalid Payment Information |
| **Prerequisite** | **:** | Student is logged in and has an outstanding fees balance. |
| **Acceptance Criteria** | **:** | **Given:** The student is on the fees payment page.  **When:** The student enters invalid or incomplete payment details (e.g., incorrect credit card number).  The student clicks the "Submit Payment" button.  **Then:**  The system displays an error message indicating that the payment could not be processed.  The payment is not completed**.** |

##### Scenario# S1.3

|  |  |  |
| --- | --- | --- |
| *Scenario#* ***S1.3*** | **:** | Paying Hostel Fees with Insufficient Funds |
| **Prerequisite** | **:** | Student is logged in and has an outstanding fees balance. |
| **Acceptance Criteria** | **:** | **Given**: The student is on the fees payment page.  **When**: The student enters payment details for an account with insufficient funds.  The student clicks the "Submit Payment" button.  **Then**:  The system displays an error message indicating insufficient funds. The payment is not completed. |

#### Story-2: Room Management

|  |  |  |
| --- | --- | --- |
| *Story #* ***S2*** | : | **As a hostel warden,**  **I want to allocate rooms to students based on availability and preferences, so that students have their accommodation sorted efficiently.** |
| **Priority** | **:** | High |
| **Estimate** | **:** | M |
| **Reason** | **:** | Viewing room management allows warden to manage their hostel and  students and ensure that all transactions are recorded correctly. |

##### Scenario# S2.1

|  |  |  |
| --- | --- | --- |
| *Scenario#* ***S2.1*** | **:** | New Student Room Allotment |
| **Prerequisite** | **:** | Student is logged in and has payment records. |
| **Acceptance Criteria** | **:** | **Given:** The warden is on the room allocation page.  **When:** The warden selects the room to allocate.  **Then:**  **The system displays a list of all available rooms , allocated rooms and payment of new student.** |

* + 1. Scenario# S2.2

|  |  |  |
| --- | --- | --- |
| *Scenario#* ***S2.2*** | **:** | Room Change Request |
| **Prerequisite** | **:** | Student is logged in and has no past payments. |
| **Acceptance Criteria** | **:** | **Given:** The student is on the room change page.  **When:** The student selects the option to change room.  **Then:**  The system displays available rooms and student’s previous room and other details related to student. |

#### Story-3: View Fees Payment History

|  |  |  |
| --- | --- | --- |
| *Story #* ***S2*** | : | **As a Student**  **I want to view my fees payment history**  **So that I can track past payments and verify that all fees are paid.** |
| **Priority** | **:** | High |
| **Estimate** | **:** | M |
| **Reason** | **:** | Viewing payment history allows students to manage their finances and ensure  that all transactions are recorded correctly. |

##### Scenario# S2.1

|  |  |  |
| --- | --- | --- |
| *Scenario#* ***S2.1*** | **:** | Viewing Fees Payment History with Valid Login |
| **Prerequisite** | **:** | Student is logged in and has payment records. |
| **Acceptance Criteria** | **:** | **Given:** The student is on the payment history page.  **When:** The student selects the option to view payment history..  **Then:**  The system displays a list of past payments with details including dates, amounts, and payment status. |

* + 1. Scenario# S2.2

|  |  |  |
| --- | --- | --- |
| *Scenario#* ***S2.2*** | **:** | Viewing Fees Payment History with No Past Transactions |
| **Prerequisite** | **:** | Student is logged in and has no past payments. |
| **Acceptance Criteria** | **:** | **Given:** The student is on the payment history page.  **When:** The student selects the option to view payment history.  **Then:**  The system displays a message indicating that there are no past transactions. |

## Test Cases

|  |  |  |  |
| --- | --- | --- | --- |
| **Project Name:** | **Hostel Management**  **System** | **Test Designed by:** | **Jay Vadariya** |
| **Module Name:** | **Room Allocation** | **Test Designed date:** | 28-02-2025 |
| **Release Version:** | **1.0** | **Test Executed by:** | **R. B. Gondaliya** |
|  |  | **Test Execution date:** | 15-01-2023 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Pre-condition**: **Web application should be accessible** | | | | |
| **Test Case ID** | **Test Title** | **Test Type** | **Description** | **Test Case ID** |
| **TC\_001** | Allocate Room to web application with  valid credential | Functional | Allocate a room in the Hostel Management System web  application using valid credentials. | TC\_001 |
| **TC\_002** | Allocate Room to web application with invalid credential | Functional | Attempt to allocate a room in the Hostel Management System web application using invalid  credentials. | TC\_002 |
| **TC\_003** | Verify Room  Allocation page elements | GUI | Verify that all necessary elements are available and correctly displayed on the Room Allocation  page. | TC\_003 |
| **TC\_004** | Verify Room Allocation data is  saved correctly | Database | Ensure that the room allocation data is correctly saved to the  database when a room is allocated. | TC\_004 |

|  |  |
| --- | --- |
| Test Case Title | Allocate Room to web application with valid credential |
| Test Type | Functional |
| Test Priority | High |
| Pre-condition | Web application should be accessible |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test Step | Test Case Description | Expected Result | Actual Result | Status | Comment | Data | BUG ID |
| 1 | Access Web  application URL | The site  launched properly | Site launched successfully | Pass |  | [https://accounts.](https://accounts.google.com/ServiceLogin) [google.com/Ser](https://accounts.google.com/ServiceLogin) [viceLogin](https://accounts.google.com/ServiceLogin) |  |
| 2 | Login with valid credentials | User should be able to log in and be  directed to the dashboard. | Logged in and navigated to dashboard. | Pass |  | Username: admin; Password: admin123 |  |
| 3 | Navigate to the "Room Allocation" page | The "Room Allocation" page should load, displaying available  options for | Room Allocation page displayed. | pass |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | room allocation. |  |  |  |  |  |
| 4 | Select a student from the student list | The system  should allow selecting a  student for room  allocation. | Student selected successfully. | Pass |  | Student ID: 202301 |  |
| 5 | Choose an available room from the room list | The system should display a list of available rooms and allow  selection. | Room selected successfully. | pass |  | Room Number: 101 |  |
| 6 | Enter the allocation start date | The date field should be  editable and  accept a valid start date. | Start date accepted. |  |  | Start Date: 2025-09-01 |  |
| 7 | Enter the allocation end date | The date field should be  editable and accept a valid end date. | End date accepted. |  |  | End Date: 2025-12-31 |  |

|  |  |
| --- | --- |
| Test Case Title | Reserve Room to web application with invalid credential |
| Test Type | Functional |
| Test Priority | Medium |
| Pre-condition | Web application should be accessible |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test Step | Test Case Description | Expected Result | Actual Result | Status | Comment | Data | BUG ID |
| 1 | Access Web application  URL | The site  launched properly | Site launched successfully | Pass |  | [https://accounts.](https://accounts.google.com/ServiceLogin) [google.com/Ser](https://accounts.google.com/ServiceLogin) [viceLogin](https://accounts.google.com/ServiceLogin) |  |
| 2 | Enter valid Username in username field | Username field should be  editable and  accept the Username | Username input accepted | Pass |  | Username:vadariyajay236@gmail.com |  |
| 3 | Enter valid Password in Password field | Password field should be  editable and  accept the password and display as star or dot | Password input displayed in dot and accepted | pass |  | Password: jay236 |  |
| 4 | Enter valid captcha code in captch field | Captch field should editable and accept captcha and captcha is case sensitive | Captcha input accepted | Pass | Step require d when human action validati on perfor  m | get captcha  from image which is near by captcha field |  |
| 5 | Click on login button | User should login into site and navigated to dashboard | User navigated to dashboard and username should br display in top  of the right side. | pass |  |  |  |

|  |  |
| --- | --- |
| Test Case Title | Verify Room Reservation Page Elements |
| Test Type | GUI / Functional |
| Test Priority | High |
| Pre-condition | User is logged in and navigates to the Room Reservation page. |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test Step | Test Case Description | Expected Result | Actual Result | Status | Comment | Data | Bug ID |
| 1 | Access the | The page | Room | Pass |  |  |  |
|  | Room | should | Reservation |  |
|  | Reservatio | load | page |  |
|  | n page | successful | loaded |  |
|  |  | ly with all | successfully |  |
|  |  | UI |  |  |
|  |  | compone |  |  |
|  |  | nts |  |  |
| 2 | Verify presence | "Room Type" | "Room Type" working | Pass | Dropdown | Room types: Deluxe, Suite, Standard |  |
|  | of "Room | dropdow |  |  |  |  |
|  | Type" | n should |  |  |  |  |
|  | dropdown | be visible |  |  |  |  |
|  |  | and |  |  |  |  |
|  |  | functional |  |  |  |  |
| 3 | Verify | "Check-in | "Check-in | Pass |  |  |  |
|  | "Check-in | Date" | Date" field |  |
|  | Date" field | field | visible and |  |
|  | is present | should be | functional |  |
|  |  | visible |  |  |
|  |  | and allow |  |  |
|  |  | date |  |  |
|  |  | selection |  |  |
| 4 | Verify | "Check- | "Check-out | Pass |  |  |  |
|  | "Check- | out Date" | Date" field |  |
|  | out Date" | field | visible and |  |
|  | field is | should be | functional |  |
|  | present | visible |  |  |
|  |  | and allow |  |  |
|  |  | date |  |  |
|  |  | selection |  |  |
| 5 | Verify | "Number | "Number of | Pass |  |  |  |
|  | "Number | of | Guests" |  |
|  | of Guests" | Guests" | field |  |
|  | field is | input field | present and |  |
|  | present | should be | functional |  |
|  |  | visible |  |  |
|  |  | and allow |  |  |
|  |  | user input |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 6 | Verify "Special Requests" text area is present | "Special Requests" text area should be visible and allow  input | "Special Requests" text area visible and working | Pass |  |  |  |
| 7 | Check the availability of the  "Submit Reservatio n" button | "Submit Reservati on" button should be visible and  clickable | "Submit Reservation " button is present and functional | Pass |  |  |  |
| 8 | Verify confirmati on message appears after submitting | A  confirmat ion message should appear after successful submissio n | Confirmatio n message displayed after submission | Pass |  |  |  |
| 9 | Ensure error messages are displayed for empty required fields | Relevant error messages should be shown for empty required fields when attemptin g to  submit | Error messages displayed as expected | Pass |  |  |  |

|  |  |
| --- | --- |
| Test Case Title | Verify Room Reservation Data is Saved Correctly |
| Test Type | Database / Functional |
| Test Priority | High |
| Pre-condition | User is logged in and has successfully made a room reservation |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test Step | Test Case Description | Expected Result | Actual Result | Status | Comment | Data | Bug ID |
| 1 | Access the reservatio n database | Database should be accessible and open for  querying | Database accessed successfully | Pass |  |  |  |
| 2 | Query the database for the reservatio n entry | Reservati on should be present with correct  details | Reservation details | pass | correct | Room Type: Deluxe Check-in Date: 2024-09-  01  Check-out Date: 2024-  09-05  Guest Name: John Doe |  |
| 3 | Verify the reservatio n details in the database | Reservati on details should match the data provided during  booking | Details match correctly | Pass |  |  |  |
| 4 | Check if the reservatio n status is updated correctly | Status should be updated to "Confirm ed" or  similar | Status updated to "Confirmed " | Pass |  |  |  |
| 5 | Verify reservatio n data is accessible in the user account | User should be able to view their reservatio n details under their  account | Reservation details visible under account | Pass |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 6 | Validate that no duplicate reservatio ns are  present | Only one reservatio n should be present for the same  room and dates | No duplicate reservation s found | Pass |  |  |  |

## References

<http://www.w3schools.com/html/html_intro.asp>

* https://[www.w3schools.com/php/default.asp](http://www.w3schools.com/php/default.asp)
* https://[www.javatpoint.com/uml](http://www.javatpoint.com/uml)