**UET INVENTORY MANAGEMENT SYSTEM**

**(SOFTWARE CONSTRUCTION AND DEVELOPMENT)**

****

**SUBMITTED TO:**

MAM ALINA MUNIR

**SUBMITTED BY:**

IMAN FATIMA(2021-SE-10)

KAUSAR FATIMA(2021-SE-25)

UM-E-HABIBA(2021-SE-27)

LAIBA AMBER EJAZ(2021-SE-37)

KHADEEJA RASTI(2021-SE-53)

**DEPARTMENT OF COMPUTER SCIENCE**

**UNIVERSITY OF ENGINEERING AND TECHNOLOGY LAHORE (KSK CAMPUS)**

**TABLE OF CONTENTS:**

1. VISION
2. PROJECT DESCRIPTION
3. PROJECT GOAL
4. PROJECT DELIVERABLES
5.  PROJECT MILESTONES
6.  GANTT CHART
7. FUNCTIONAL REQUIREMENTS
8.  NON-FUNCTIONAL REQUIREMENTS
9.  USE CASE DIAGRAM
10.  ACTIVITY DIAGRAM
11.  STATE DIAGRAM
12.  SEQUENCE DIAGRAM
13.  COLLABORATION DIAGRAM
14.  COMPONENT DIAGRAM
15.  DEPLOYMENT DIAGRAM
16.  CLASS DIAGRAM
17.  ER DIAGRAM
18.  WIREFRAME DIAGRAMS
19.  3NF DB DESIGN

**(-----:VISION OF PRODUCT:-----)**

**For** Campus Coordinator **who** wants to manage their hostel inventory. **The** **UETIMS** is a desktop app **that** aims to streamline the inventory processes, and provide a user-friendly interface . **Unlike** manual management, **our** focus will have to design comprehensive solution to hostel inventory management system.

**(-----:PROJECT DESCRIPTION:-----)**

The role of a Campus Coordinator in managing a university's hostel facilities has become increasingly complex. The need for an efficient, organized inventory management and hostel maintenance has never been greater. To address this challenge, we introduce the University Hostel Inventory and Management System (UETIMS), a desktop application designed to manage hostel inventory facilities. UETIMS will transform the way inventory is managed within hostels manually. It will categorize and track every item, from electrical and hardware supplies to sanitary and furniture items. Campus coordinator will have the ability to add, edit,view or remove inventory items. The system maintains a comprehensive list of hostels , including hostel types,no. of corridors, washrooms,no.of storeys, no. of rooms. The system maintains a comprehensive list of inventory used in public, staff and students rooms. The system records all incoming deliveries of new inventory items, capturing essential information such as delivery dates, supplier details, and items received. The system will maintain a log of incoming requests and issue and managing inventory in response to these concerns.The system offers search and filter options to easily find specific inventory items,maintenance records, or deliveries. The system will generate a monthly consumption report. An intuitive, user-friendly interface is designed for easy navigation and data entry. Accessibility features ensure usability for all users. The system provides user documentation and training materials. In its development, UETIMS places a strong emphasis on improving accuracy while prioritizing data security and maintenance.

**(-----:PROJECT GOAL:-----)**

Inventory management requires inventory visibility — knowing when to order, how much to order, where to store stock and how much to allocate to whom. The goal of UET Inventory Management is to have the right products in the right place at the right time.

**(-----:PROJECT DELIVERABLES:-----)**

The deliverables for an inventory management system (maintaining records of incoming, in stock, and outgoing inventory) includes a variety of artifacts across different stages of the software construction and development process.

**External Deliverables:**for benefiting product, customers or clients

* User Documentation
* Working Software
* Project Proposal
* Software Requirements Specification
* Software Design Document

**Internal Deliverables:**for benefiting team

* Database Design
* Software Design Document
* System Development
* Testing and Quality Assurance
* Deployment Plan
* Post-Implementation Support Plan
* Project Closeout Documentation

**Complete Deliverables Uptil Now:**

Following are deliverables we have completed at present.

1. **Project Proposal:**

The document contained:

* Vision of Product
* Project Description

1. **Software Requirements Specification:**

The document contained:

* Problem Domain, Significance And Business Rules.
* Functional Requirements (User Requirements and System Specification)
* Non-Functional Requirements
* Process of finding Potential classes
* Class Diagram

1. **Database Design:**

The document contained:

* Tables before Normalization
* Tables after 1st Normalization
* Tables after 2nd Normalization
* Tables after 3rd Normalization

1. **Software Design Document**

The document contained:

* Vision
* Project Description
* Project Goal
* Functional Requirements
* Non-Functional Requirements
* Use Case Diagram
* Activity Diagram
* State Diagram
* Sequence Diagram
* Collaboration Diagram
* Component Diagram
* Deployment Diagram
* Class Diagram
* ER Diagram
* Wireframe Diagrams
* 3Nf Db Design
* Project Deliverables
* Project Milestones
* Gantt Chart

**Deliverables Not Yet Completed:**

Following are deliverables that we aim to complete before final evaluation.

1. **System Development:**

* Source code for the Inventory Management System.
* Debugging documentation.

1. **User Documentation:**

* User manuals and guides for system installation and operation.
* Troubleshooting guides.

1. **Testing and Quality Assurance:**

* Test plans and test cases.
* Test results and bug reports.
* Verification and validation documentation.

1. **Deployment Plan:**

* Procedures for deploying the system to production.
* Rollback procedures in case of issues.

1. **Working Software:**

* Software Product that is to be deployed to client’s system

1. **Post-Implementation Support Plan:**

* Support and maintenance documentation.
* Contact information for support services.
* Plans for future updates and enhancements.

1. **Project Closeout Documentation:**

* Final project report.

These delieverables collectively aim to create an Inventory Management System for UET Hostels that enhances manages and keeps record of incoming, in-stock and outgoing inventory. The successful achievement of these deliverables contributes to the overall success and adoption of the system.

**(-----:PROJECT MILESTONES:-----)**

Following are the milestones that we have achieved uptil present:

1. Registered subject “Software Construction and Development”
2. Made the team of 5 members based on our mutual interests to work on semester project
3. Selected a project “Inventory Management System”
4. Got a chance to built Inventory Management System for UET Hostels
5. Gathered Requirements from Personal Assistant to Campus Coordinator of UET Lahore, New Campus
6. Documented Project Proposal
7. Documented System Requirement Specification Document
8. Designed Normalized Database Design Schema
9. Documented Software Design Document

Moreover, we will be achieving future milestones based on developing, debugging, testing, deploying to concerned authority and maintaining the management system.

**(-----:TIMELINE(GANTT CHART):-----)**

Following are the milestones respected to days in which we have achieved them uptil now:

Following are the milestones with respect to time we aim to achieve in future :

**(-----:FUNCTIONAL REQUIREMENTS:-----)**

**(-----:1. USER REQUIREMENTS:-----)**

1. The admin shall be able to login.
2. The admin shall be able to recover their account.
3. The admin shall be able to add items.
4. The admin shall be able to delete items.
5. The admin shall be able to update items.
6. The admin shall be able to search items.
7. The admin shall be able to view items.
8. The admin shall be able to add rooms to hostels.
9. The admin shall be able to delete rooms.
10. The admin shall be able to search rooms.
11. The admin shall be able to update rooms.
12. The admin shall be able to view rooms.
13. The admin shall be able to add hostels.
14. The admin shall be able to delete hostels.
15. The admin shall be able to search hostels.
16. The admin shall be able to update hostels.
17. The admin shall be able to view hostels.
18. The admin shall be able to add supplier details.
19. The admin shall be able to delete supplier details.
20. The admin shall be able to search supplier details.
21. The admin shall be able to update supplier details.
22. The admin shall be able to view supplier details.
23. The admin shall be able to view monthly report of items.
24. The admin shall be able to search requests against items.
25. The admin shall be able to view requests against items.
26. The admin shall be able to search requests against items.
27. The admin shall be able to update requests against items.
28. The admin shall be able to add requests against items.
29. The admin shall be able to log-out.

**(-----:2. SYSTEM REQUIREMENTS:-----)**

**REGISTRATION:**

1. The system must be able to allow admin to login.
2. The system must be able to allow admin to recover their account in case of forget password.

**INVENTORY MANAGEMENT:**

1. The system must be able to allow admin to add items.
2. The system must be able to allow admin to delete items.
3. The system must be able to allow admin to update items.
4. The system must be able to allow admin to search items.
5. The system must be able to allow admin to view items.
6. The system must be able to allow admin to add rooms to hostels.
7. The system must be able to allow admin to delete rooms.
8. The system must be able to allow admin to update rooms.
9. The system must be able to allow admin to search rooms.
10. The system must be able to allow admin to view rooms.
11. The system must be able to allow admin to add hostels.
12. The system must be able to allow admin to delete hostels.
13. The system must be able to allow admin to update hostels.
14. The system must be able to allow admin to search hostels.
15. The system must be able to allow admin to view hostels.
16. The system must be able to allow admin to add supplier details.
17. The system must be able to allow admin to delete supplier details.
18. The system must be able to allow admin to update supplier details.
19. The system must be able to allow admin to search supplier details.
20. The system must be able to allow admin to view supplier details.
21. The system must be able to allow admin to generate monthly report of items.
22. The system must be able to allow admin to search requests against items.
23. The system must be able to allow admin to delete requests against items.
24. The system must be able to allow admin to update requests against items.
25. The system must be able to allow admin to view requests against items.
26. The system must be able to allow admin to add requests against items.

**LOG-OUT:**

1. The system must be able to allow user to log-out.

**(-----:NON-FUNCTIONAL REQUIREMENTS:-----)**

Non-functional requirements for the University Hostel Inventory and Management System (UETIMS) based on the provided description:

**1**. **PERFORMANCE:**

UETIMS should be responsive and provide quick access to data and functionalities, even with a large amount of inventory and hostel information.

**2. SCALABILITY:**

The system should be capable of handling an increasing number of hostels, inventory items.

**3. AVAILABILITY:**

The system should be available 24/7 to accommodate the needs of campus coordinators .

**4. DATA SECURITY:**

UETIMS must implement robust data security measures to protect sensitive information, including inventory details, supplier data, and user accounts.

**5. DATA BACKUP AND RECOVERY:**

Regular data backups should be performed, and the system should have a robust data recovery mechanism in case of data loss or system failure.

**6. USER AUTHENTICATION AND AUTHORIZATION:**

The system should enforce strong user authentication and authorization mechanisms to ensure that only authorized personnel can access and modify data.

**7. USABILITY:**

UETIMS should provide an intuitive and user-friendly interface to enhance user adoption and minimize the need for extensive training.

**8. ACCURACY AND DATA INTEGRITY:**

The system should maintain data accuracy and integrity by validating and verifying data inputs and updates.

**9. AUDIT TRAILS:**

UETIMS should maintain audit trails to track user actions and changes made to inventory data for accountability and monitoring purposes.

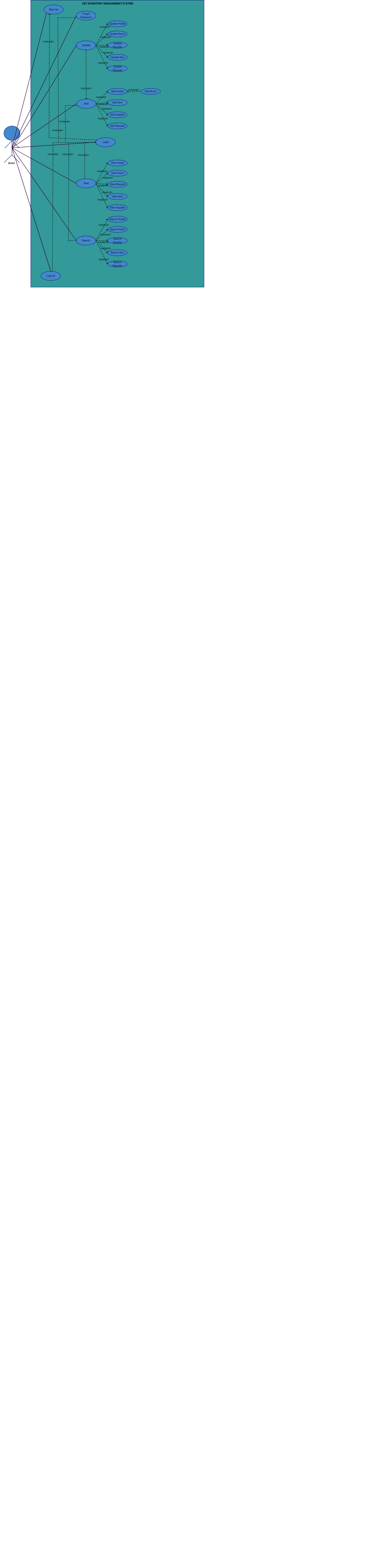
**10. COMPLIANCE:**

The system should adhere to relevant regulatory and compliance requirements related to data privacy and inventory management.

**11. RESPONSE TIME:**

The system should provide reasonable response times for generating reports and processing user requests to ensure efficient operations.

**(-----:USE CASE DIAGRAM:-----)**

****

**(-----:USE CASE ATTRIBUTES:-----)**

1. **SIGN-UP:**

**Name:** Admin Account Creation

**Actor:** Admin

**Precondition:** The admin must have access to the system and the system must be functional.

**Goal:** The goal of this use case is to allow the admin to create an account on the system.

**Trigger:** The admin adds details to create an account on the system.

**Post-condition:** The admin has an account on the system.

**Exception:** If the admin's credentials are incorrect or the system is not functional, the admin will not be able to create an account.

**Alternate flow:** If the admin already has an account, they can log in to the system instead of creating a new account.

**Main flow:**

The admin opens the system and clicks on the sign-up button.

The system presents a form to the admin to fill in their details such as name, email, password.

The admin fills in the required details and submits the form.

The system validates the details and creates an account for the admin.

1. **LOGIN:**

**Name:** Admin Login

**Actor:** Admin

**Precondition:** The admin must have an account on the system and the system must be functional.

**Goal:** Allow the admin to log in to the system, and provide a mechanism for password recovery if forgotten.

**Trigger:** The admin requests to log in to the system.

**Post-condition:** The admin has successfully logged in to the system.

**Exception:** If the admin's credentials are incorrect or the system is not functional, the admin will not be able to log in to the system.

**Main flow:**

The admin opens the system and clicks on the login button.

The system presents a form to the admin to fill in their credentials such as email and password.

The admin fills in their credentials and submits the form.

The system validates the credentials and grants the admin access to the system.

**Alternative Flow: (Incorrect Credentials)**

If the credentials are incorrect:

The system displays an error message.

Admin is prompted to re-enter the correct credentials.

The process returns to step 3.

If the credentials are valid:

The system grants access to the admin.

The admin is redirected to the system dashboard.

The use case ends.

**Alternate Flow: (Password Recovery)**

6. If the admin forgets their password:

Admin clicks on the "Forgot Password" link.

System prompts the admin to provide their registered email address.

Admin enters the email address and submits the request.

**Alternative Flow: (Invalid Email)**

If the provided email address is not registered in the system:

System displays an error message.

Admin is prompted to provide a valid registered email address.

The process returns to step 6.

If the email address is valid:

System sends a password reset link to the admin's email.

Admin receives an email with instructions and a link to reset the password.

**Alternative Flow: (Password Reset)**

8. Admin clicks on the provided link in the email.

System verifies the link's validity.

Admin is redirected to a page where they can reset their password.

Admin sets a new password and submits the form.

System updates the password and confirms the successful password reset.

Admin can now log in with the new password.

The use case ends.

1. **ADD-HOSTEL:**

**Name:** Add Hostel

**Actor:** Admin

**Precondition:** The admin must be logged in to the system and the system must be functional.

**Goal:** The goal of this use case is to allow the admin to add a hostel.

**Trigger:** The admin requests to add a hostel.

**Post-condition:** The hostel has been added to the system.

**Exception:** If the system is not functional, the admin will not be able to add a hostel.

**Main flow:**

The admin clicks on the hostel button.

The system presents a form to the admin to fill in the hostel details.

The admin fills in the required details and submits the form.

The system validates the details and adds the hostel to the system.

1. **UPDATE-HOSTEL:**

**Name:** Update Hostel

**Actor:** Admin

**Precondition:** The admin must be logged in to the system,hostel must already be present and the system must be functional.

**Goal:** The goal of this use case is to allow the admin to update(update and delete) a hostel.

**Trigger:** The admin requests to edit a hostel.

**Post-condition:** The hostel has been updated in the system.

**Exception:** If the system is not functional, the admin will not be able to update a hostel.

**Main flow:**

The admin clicks on the hostel button.

The system presents a list of hostels to the admin.

The admin selects the hostel they want to edit.

The system presents the details of the selected hostel to the admin.

The admin clicks on the update button and makes the necessary changes.

The system validates the changes and updates the hostel in the system.

1. **ADD-ROOM:**

**Name:** Add Room

**Actor:** Admin

**Precondition:** The admin must be logged in to the system, hostel must be there to which room can be added and the system must be functional.

**Goal:** The goal of this use case is to allow the admin to add a room.

**Trigger:** The admin requests to add a room.

**Post-condition:** The room has been added to the system.

**Exception:** If the system is not functional, the admin will not be able to add a room.

**Main flow:**

The admin clicks on the room button.

The system presents a form to the admin to fill in the room details.

The admin fills in the required details and submits the form.

The system validates the details,checks the hostel existence and adds the room to the system.

1. **UPDATE-ROOM:**

**Name:** Update Room

**Actor:** Admin

**Precondition:** The admin must be logged in to the system,room must already be present and the system must be functional.

**Goal:** The goal of this use case is to allow the admin to update(update and delete) a room.

**Trigger:** The admin requests to update a room.

**Post-condition:** The room has been updated in the system.

**Exception:** If the system is not functional, the admin will not be able to update a room.

**Main flow:**

The admin clicks on the room button.

The system presents a list of rooms to the admin.

The admin selects the room they want to edit.

The system presents the details of the selected room to the admin.

The admin clicks on the update button and makes the necessary changes.

The system validates the changes and updates the room in the system.

1. **ADD-ITEM:**

**Name:** Add Item

**Actor:** Admin

**Precondition:** The admin must be logged in to the system and the system must be functional.

**Goal:** The goal of this use case is to allow the admin to add an item.

**Trigger:** The admin requests to add an item.

**Post-condition:** The item has been added to the system.

**Exception:** If the system is not functional, the admin will not be able to add an item.

**Main flow:**

The admin clicks on the item button.

The system presents a form to the admin to fill in the item details.

The admin fills in the required details and submits the form.

The system validates the details and adds the item to the system.

1. **UPDATE-ITEM:**

**Name:** Update Item

**Actor:** Admin

**Precondition:** The admin must be logged in to the system,item must already be present and the system must be functional.

**Goal:** The goal of this use case is to allow the admin to update(update and delete) a item.

**Trigger:** The admin requests to update item.

**Post-condition:** The item has been updated in the system.

**Exception:** If the system is not functional, the admin will not be able to update a item.

**Main flow:**

The admin clicks on the item button.

The system presents a list of items to the admin.

The admin selects the item they want to edit.

The system presents the details of the selected item to the admin.

The admin clicks on the update button and makes the necessary changes.

The system validates the changes and updates the item in the system.

1. **ADD-SUPPLIER:**

**Name:** Add Supplier

**Actor:** Admin

**Precondition:** The admin must be logged in to the system and the system must be functional.

**Goal:** The goal of this use case is to allow the admin to add a supplier.

**Trigger:** The admin requests to add a supplier.

**Post-condition:** The supplier has been added to the system.

**Exception:** If the system is not functional, the admin will not be able to add a supplier.

**Main flow:**

The admin clicks on the supplier button.

The system presents a form to the admin to fill in the supplier details.

The admin fills in the required details and submits the form.

The system validates the details and adds the supplier to the system.

1. **UPDATE-SUPPLIER:**

**Name:** Update Supplier

**Actor:** Admin

**Precondition:** The admin must be logged in to the system,supplier must already be present and the system must be functional.

**Goal:** The goal of this use case is to allow the admin to update(update and delete) a supplier.

**Trigger:** The admin requests to update a supplier.

**Post-condition:** The supplier has been updated in the system.

**Exception:** If the system is not functional, the admin will not be able to update a supplier.

**Main flow:**

The admin clicks on the supplier button.

The system presents a list of suppliers to the admin.

The admin selects the supplier they want to edit.

The system presents the details of the selected supplier to the admin.

The admin clicks on the update button and makes the necessary changes.

The system validates the changes and updates the supplier in the system.

1. **ADD-REQUESTS:**

**Name:** Add Request

**Actor:** Admin

**Precondition:** The admin must be logged in to the system and the system must be functional.

**Goal:** The goal of this use case is to allow the admin to add a request.

**Trigger:** The admin requests to add a request.

**Post-condition:** The request has been added to the system.

**Exception:** If the system is not functional, the admin will not be able to add a request.

**Main flow:**

The admin clicks on the request button.

The system presents a form to the admin to fill in the request details.

The admin fills in the required details and submits the form.

The system validates the details and adds the request to the system.

1. **UPDATE-REQUEST:**

**Name:** Update Request

**Actor:** Admin

**Precondition:** The admin must be logged in to the system, request must already be present and the system must be functional.

**Goal:** The goal of this use case is to allow the admin to update(update and delete) a request.

**Trigger:** The admin requests to update a request.

**Post-condition:** The request has been updated in the system.

**Exception:** If the system is not functional, the admin will not be able to update a request.

**Main flow:**

The admin clicks on the request button.

The system presents a list of requests to the admin.

The admin selects the request they want to edit.

The system presents the details of the selected request to the admin.

The admin clicks on the update button and makes the necessary changes.

The system validates the changes and updates the request in the system.

1. **SEARCH-HOSTEL:**

**Name:** Search Hostel

**Actor:** Admin

**Precondition:**

The admin must be logged into the system.

The system must be functional.

**Goal:**

Allow the admin to search for requests based on specific criteria.

**Trigger:**

The admin initiates a search for hostels.

**Post-condition:**

The system displays a list of hostels matching the search criteria.

**Exception:**

If the system is not functional, the admin will be unable to search for hostels.

**Main Flow:**

Admin logs into the system.

Admin navigates to the "Search Hostels" section.

System presents search criteria options to the admin.

Admin enters the desired search criteria, such as hostel ID etc.

Admin initiates the search.

System retrieves and displays a list of hostels matching the specified criteria.

Admin reviews the search results.

1. **SEARCH-ROOM:**

**Name:** Search Room

**Actor:** Admin

**Precondition:**

The admin must be logged into the system.

The system must be functional.

**Goal:**

Allow the admin to search for rooms based on specific criteria.

**Trigger:**

The admin initiates a search for rooms.

**Post-condition:**

The system displays a list of rooms matching the search criteria.

**Exception:**

If the system is not functional, the admin will be unable to search for rooms.

**Main Flow:**

Admin logs into the system.

Admin navigates to the "Search Rooms" section.

System presents search criteria options to the admin.

Admin enters the desired search criteria, such as room ID etc.

Admin initiates the search.

System retrieves and displays a list of rooms matching the specified criteria.

Admin reviews the search results.

1. **SEARCH-ITEM:**

**Name:** Search Item

**Actor:** Admin

**Precondition:**

The admin must be logged into the system.

The system must be functional.

**Goal:**

Allow the admin to search for items based on specific criteria.

**Trigger:**

The admin initiates a search for items.

**Post-condition:**

The system displays a list of items matching the search criteria.

**Exception:**

If the system is not functional, the admin will be unable to search for items.

**Main Flow:**

Admin logs into the system.

Admin navigates to the "Search Items" section.

System presents search criteria options to the admin.

Admin enters the desired search criteria, such as item ID etc.

Admin initiates the search.

System retrieves and displays a list of items matching the specified criteria.

Admin reviews the search results.

1. **SEARCH-SUPPLIER:**

**Name:** Search Supplier

**Actor:** Admin

**Precondition:**

The admin must be logged into the system.

The system must be functional.

**Goal:**

Allow the admin to search for suppliers based on specific criteria.

**Trigger:**

The admin initiates a search for suppliers.

**Post-condition:**

The system displays a list of suppliers matching the search criteria.

**Exception:**

If the system is not functional, the admin will be unable to search for suppliers.

**Main Flow:**

Admin logs into the system.

Admin navigates to the "Search Suppliers" section.

System presents search criteria options to the admin.

Admin enters the desired search criteria, such as supplier ID etc.

Admin initiates the search.

System retrieves and displays a list of suppliers matching the specified criteria.

Admin reviews the search results.

1. **SEARCH-REQUEST:**

**Name:** Search Request

**Actor:** Admin

**Precondition:**

The admin must be logged into the system.

The system must be functional.

**Goal:**

Allow the admin to search for requests based on specific criteria.

**Trigger:**

The admin initiates a search for requests.

**Post-condition:**

The system displays a list of requests matching the search criteria.

**Exception:**

If the system is not functional, the admin will be unable to search for requests.

**Main Flow:**

Admin logs into the system.

Admin navigates to the "Search Requests" section.

System presents search criteria options to the admin.

Admin enters the desired search criteria, such as request ID etc.

Admin initiates the search.

System retrieves and displays a list of requests matching the specified criteria.

Admin reviews the search results.

1. **VIEW HOSTELS:**

**Name:** View Hostels

**Actor:** Admin

**Precondition:** The admin must be logged in to the system and the system must be functional.

**Goal:** The goal of this use case is to allow the admin to view hostels.

**Trigger:** The admin requests to view hostels.

**Post-condition:** The system displays a list of hostels to the admin.

**Exception:** If the system is not functional, the admin will not be able to view hostels.

**Main flow:**

The admin clicks on the hostels button.

The system presents a list of hostels to the admin.

1. **VIEW ROOMS:**

**Name:** View Rooms

**Actor:** Admin

**Precondition:** The admin must be logged in to the system and the system must be functional.

**Goal:** The goal of this use case is to allow the admin to view rooms.

**Trigger:** The admin requests to view rooms.

**Post-condition:** The system displays a list of rooms to the admin.

**Exception:** If the system is not functional, the admin will not be able to view rooms.

**Main flow:**

The admin clicks on the rooms button.

The system presents a list of rooms to the admin.

1. **VIEW ITEMS:**

**Name:** View Items

**Actor:** Admin

**Precondition:** The admin must be logged in to the system and the system must be functional.

**Goal:** The goal of this use case is to allow the admin to view items.

**Trigger:** The admin requests to view items.

**Post-condition:** The system displays a list of items to the admin.

**Exception:** If the system is not functional, the admin will not be able to view items.

**Main flow:**

The admin clicks on the items button.

The system presents a list of items to the admin.

1. **VIEW SUPPLIERS:**

**Name:** View Suppliers

**Actor:** Admin

**Precondition:** The admin must be logged in to the system and the system must be functional.

**Goal:** The goal of this use case is to allow the admin to view suppliers.

**Trigger:** The admin requests to view suppliers.

**Post-condition:** The system displays a list of suppliers to the admin.

**Exception:** If the system is not functional, the admin will not be able to view suppliers.

**Main flow:**

The admin clicks on the suppliers button.

The system presents a list of suppliers to the admin.

1. **VIEW REQUESTS:**

**Name:** View Requests

**Actor:** Admin

**Precondition:** The admin must be logged in to the system and the system must be functional.

**Goal:** The goal of this use case is to allow the admin to view requests.

**Trigger:** The admin requests to view requests.

**Post-condition:** The system displays a list of requests to the admin.

**Exception:** If the system is not functional, the admin will not be able to view requests.

**Main flow:**

The admin clicks on the requests button.

The system presents a list of requests to the admin.

1. **LOG-OUT:**

**Name:** Logout

**Actor:** Admin

**Precondition:** The admin must be logged in to the system.

**Goal:** The goal of this use case is to allow the admin to log out of the system.

**Trigger:** The admin requests to log out of the system.

**Post-condition:** The admin has been logged out of the system.

**Exception:** None.

**Main flow:**

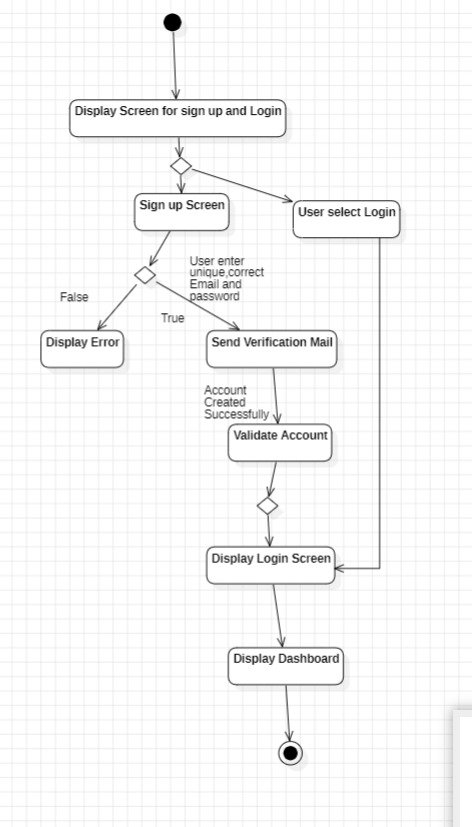
The admin clicks on the logout button.

The system logs the admin out of the system.

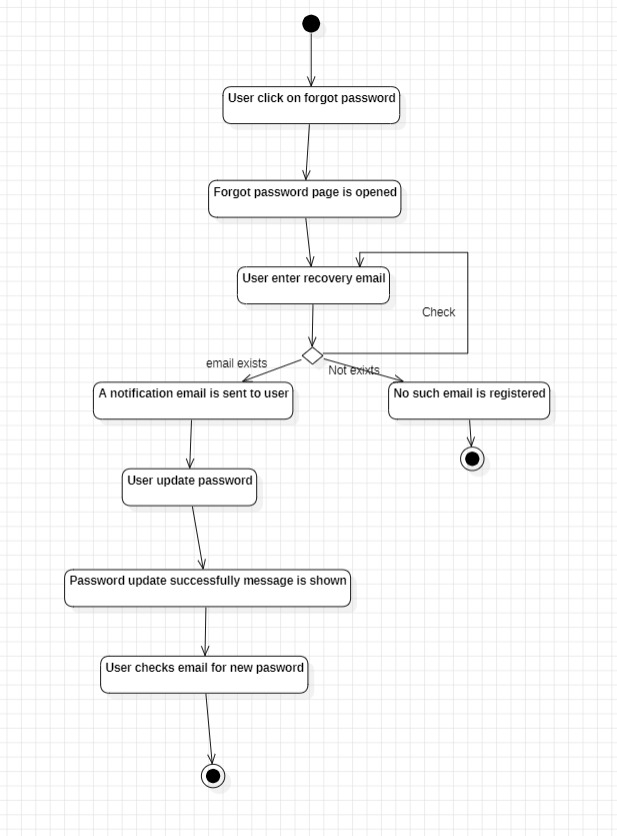
**(-----:ACTIVITY DIAGRAM:-----)**

This activity diagram shows the flow of events when the Admin sign-up,login,adds a hostel, updates a hostel, views a hostel,searches a hostel,adds a room, updates a room, views a room,searches a room,adds an item, updates an item, views an item,searches an item,adds a supplier, updates a supplier, views a supplier,searches a supplier,adds a request, updates a request, views a request,searches a request to the System and logout from the System, providing a visual representation of the interactions between the Admin and the System.

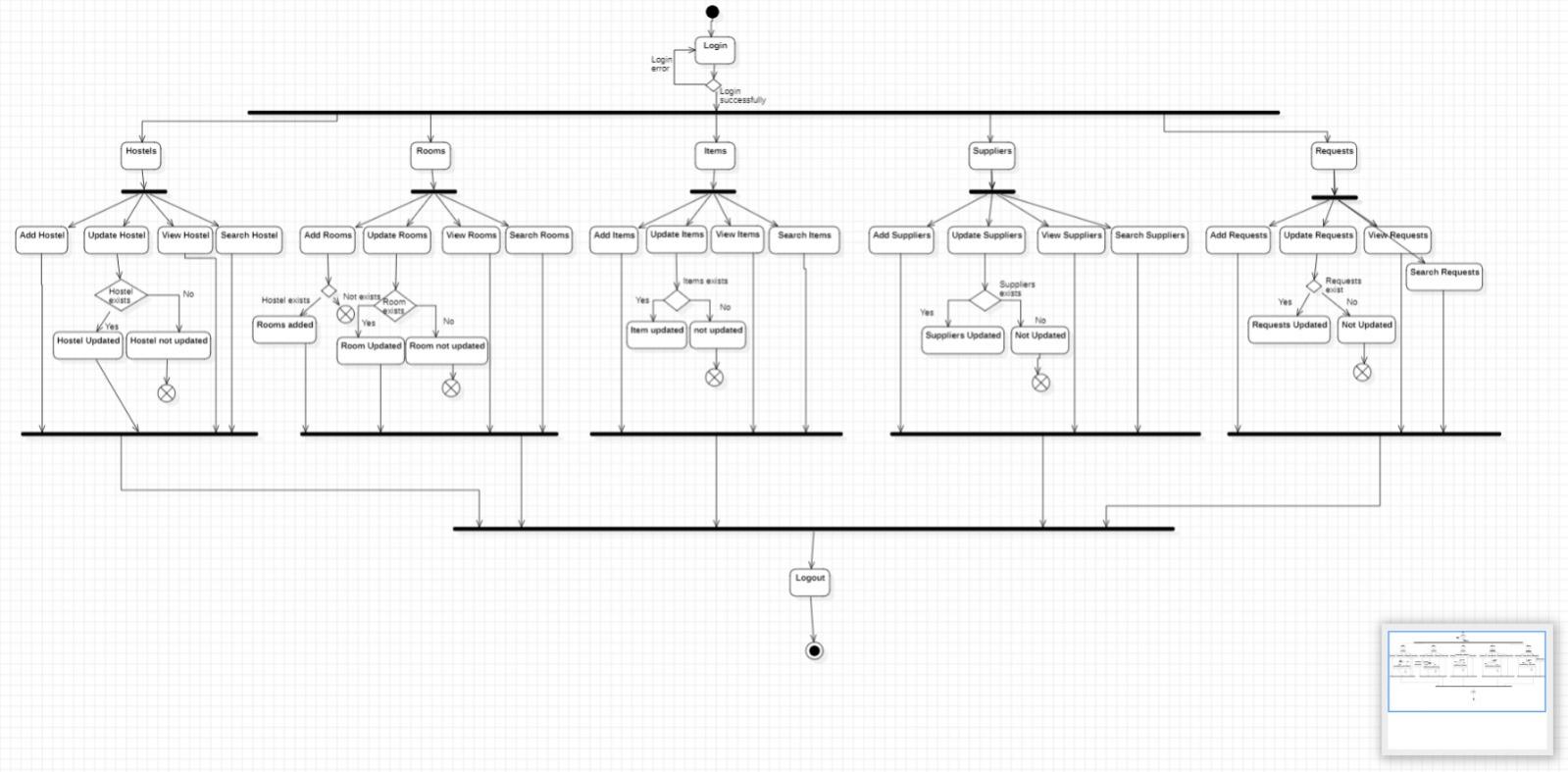
SIGN-UP:



FORGOT PASSWORD:

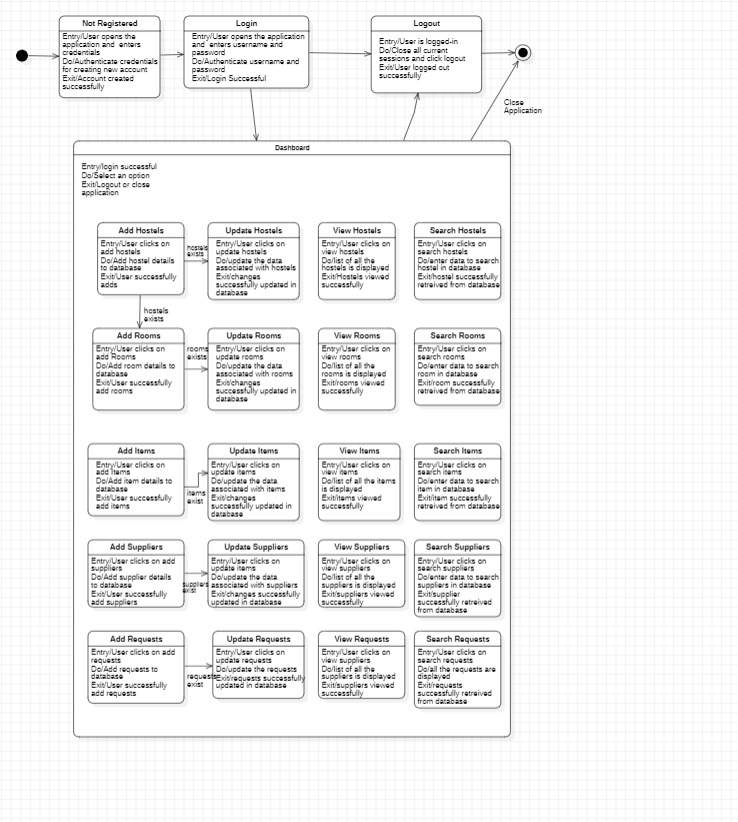


ACTIVITY AFTER LOGIN:



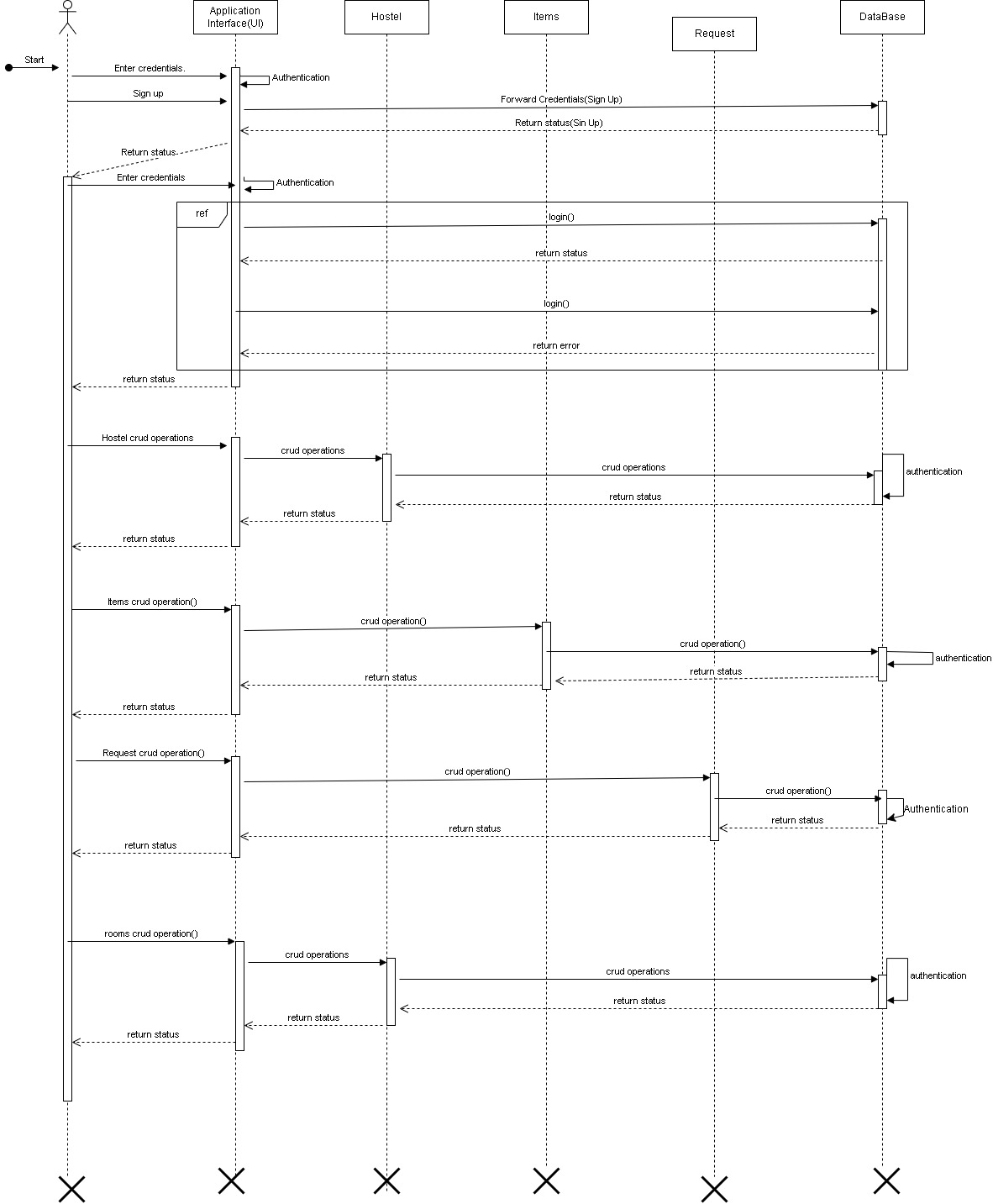
**(-----:STATE MACHINE DIAGRAM:-----)**

This state machine diagram shows the flow of events when the Admin sign-up,login,adds a hostel, updates a hostel, views a hostel,searches a hostel,adds a room, updates a room, views a room,searches a room,adds an item, updates an item, views an item,searches an item,adds a supplier, updates a supplier, views a supplier,searches a supplier,adds a request, updates a request, views a request,searches a request to the System and logout from the System, providing a visual representation of the interactions between the Admin and the System.



**(-----:SEQUENCE DIAGRAM:-----)**

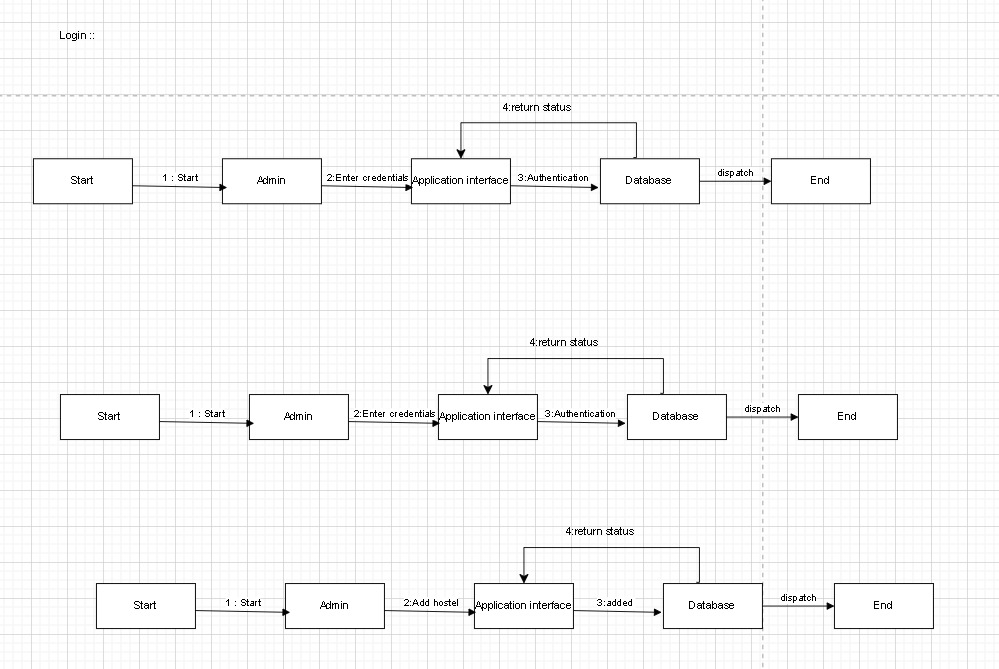
This sequence diagram visually outlines the chronological flow of events as the Admin interacts with the System. It depicts the sequential steps of sign-up, login, and various operations involving hostels, rooms, items, suppliers, and requests. The diagram illustrates how these interactions unfold over time, including updates, views, and searches. The roles of the Admin, System UI, and Database are clarified, showcasing their coordinated collaboration in processing tasks. This diagram serves as a comprehensive visual guide to understand the dynamic interactions and temporal order within the Admin's engagement with the System.

****

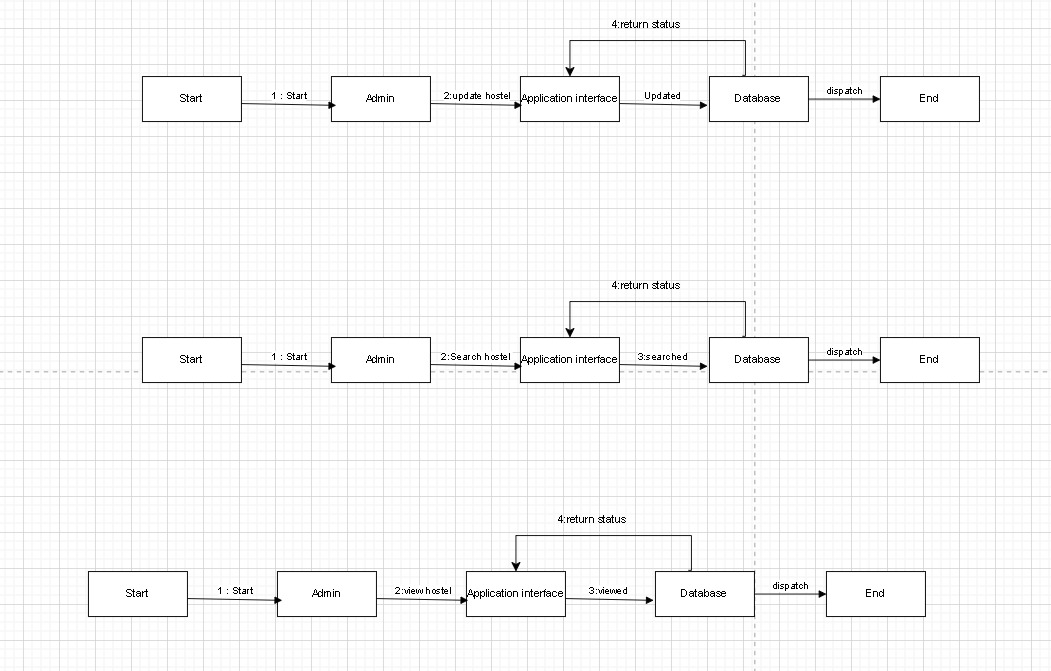
**(-----:COLLABORATION DIAGRAM:-----)**

This collaboration diagram visualizes Admin interactions with the System, depicting events such as sign-up, login, and operations involving hostels, rooms, items, suppliers, and requests. The flow illustrates the seamless communication as the Admin adds, updates, views, and searches for information within these entities. The diagram offers a clear representation of the dynamic collaboration between the Admin and the System during tasks related to hostels, rooms, items, suppliers, and requests, concluding with the logout action.

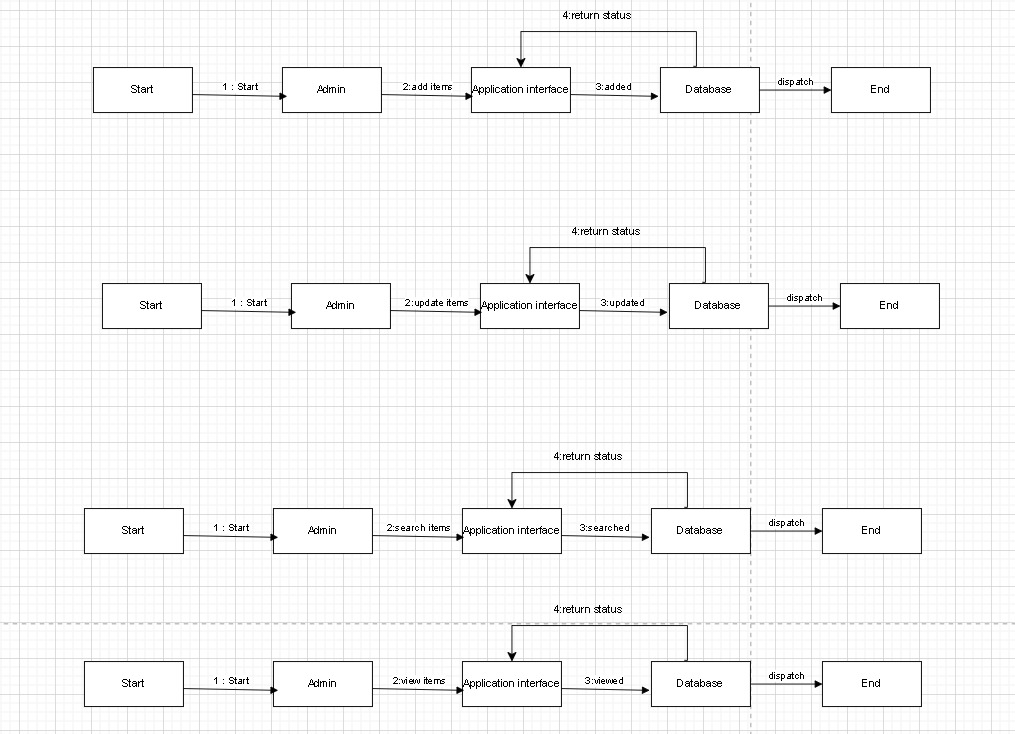
Login:



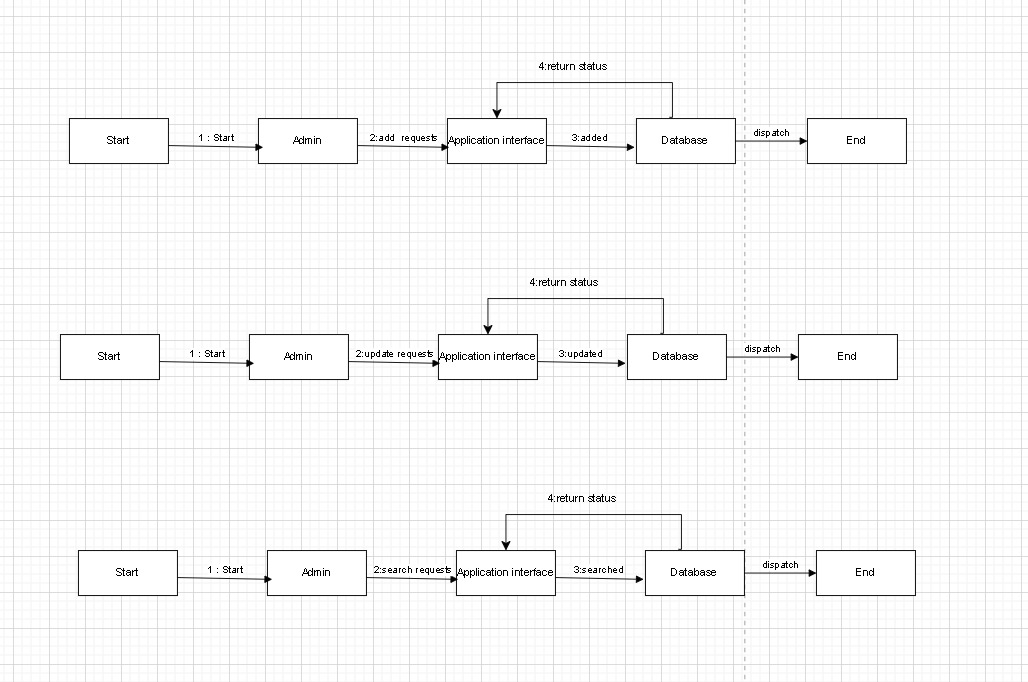
Hostel:

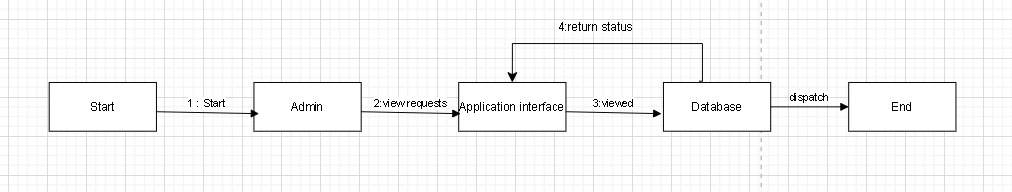


Item

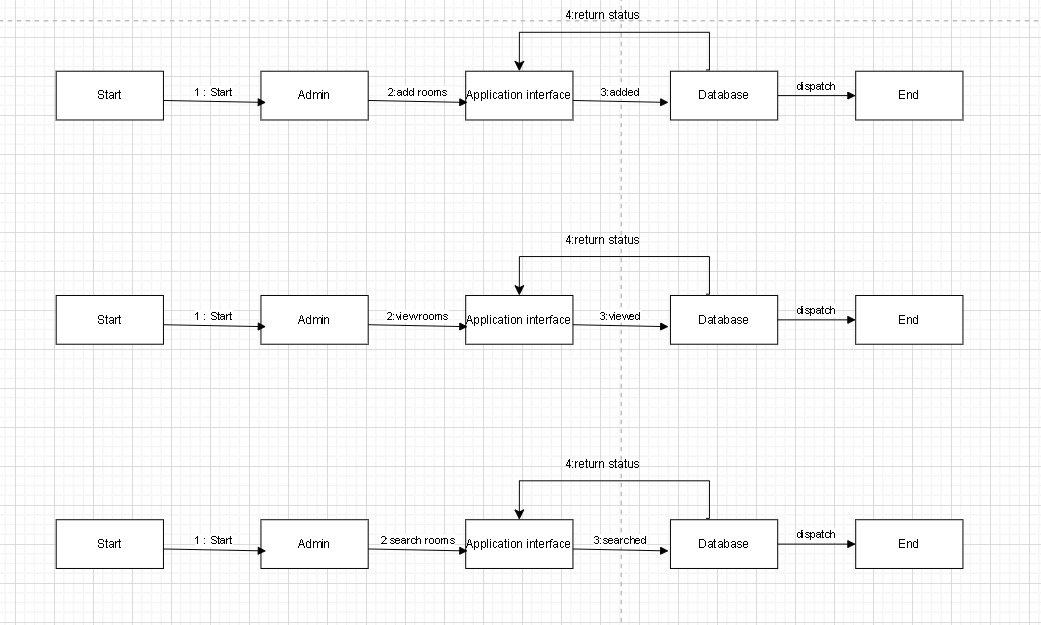


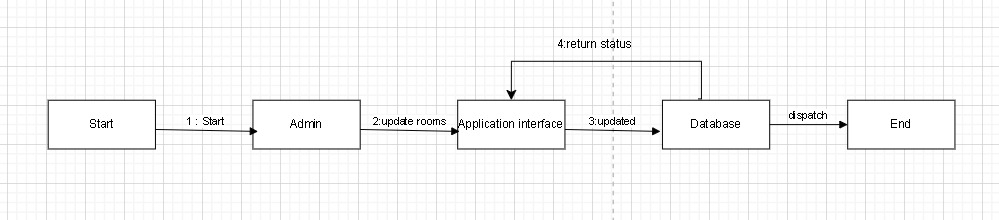
Request





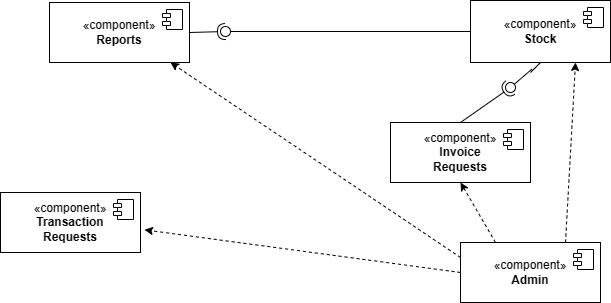
Rooms





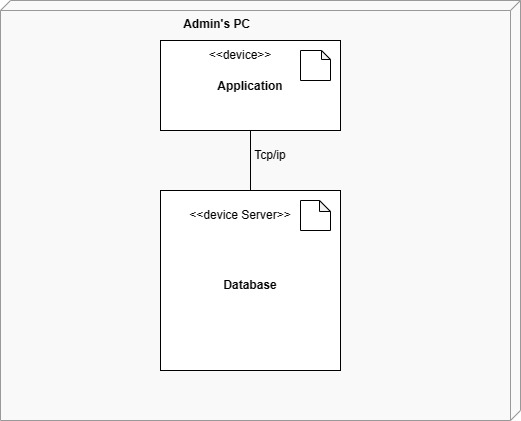
**(-----:COMPONENT DIAGRAM:-----)**

A component diagram represents the various components of the system, such as the Reports, Transaction Requests, Invoice Requests, Stock, Admin, along with their interfaces and dependencies.



**(-----:DEPLOYMENT DIAGRAM:-----)**

A deployment diagram for the shows the physical deployment of the software system and hardware components, such as Database Servers and User PC.

****

**(-----:PROCESS OF FINDING NOUNS AND VERBS:-----)**

**Step 1:Identify all relevant nouns:**

Following words are nouns identified in above description:

* Campus Coordinator
* University
* Hostel
* hostel types
* no. of corridors
* Washrooms
* no.of storeys
* no. of rooms
* Items
* Sanitory
* Furniture
* Hardware
* Electrical
* Room
* Category
* Inventory
* Supplier
* Requests
* Report

**Step 2:Identify classes:**

By looking at above nouns, we can identify that:

* Campus Coordinator is main performer of system
* University is just used as noun in description but is not part of the system.
* Hostel types is the type of Hostel
* Corridors also belong to hostel
* Washrooms can be classified as room category that belongs to hostel
* Storeys are of hostel
* No. of Rooms belongs compositely to hostel
* Sanitory is an item
* Furniture is an item
* Hardware is an item
* Electrical is an item
* Category is category of room, and items
* Inventory is referred as item
* Details are details of Supplier
* Delivery dates are dates of items supplied by Supplier
* Request belong to hostel, item and has other related info
* Records can be of item, complain that can be displayed using sql query so it is not required for now.

**Potential Classes:**

Following are potential classes identified:

1. Hostel
2. Rooms
3. Items
4. Admin
5. Supplier
6. Request

**Step 3:Identify Data Members:**

Some data members for above classes can be identified from above nouns(that are underlined) but some of them are explicitly mentioned:

1. **Hostel Class:**

Hostel\_id, Hostel\_name, Location, Total-Rooms, No. Of Floors, Warden

1. **Rooms Class:**

Room\_id, Floor no., Status, ItemCategory, Quantity-of-Items, Room-Category

1. **Items Class:**

Item\_id, Room\_id, Hostel\_id, Supplier\_id, Price, Category, Status, Quantity, Allocation\_Date

1. **Admin Class:**

Admin\_id, email, password, name

1. **Supplier Class:**

Supplier\_id, Name, Bill\_no, Price, Delivery-Category, Delivery-Date, Quantity

1. **Request Class:**

Request\_id, Room\_id, Hostel\_id, Item\_Category, Request\_Date, Quantity

**VERBS:**

Following underlined words are verbs identified in above description:

* Role
* Managing
* To Categorize item
* To track item
* Ability
* Add
* Edit
* View
* Remove
* To maintain hostel rooms
* Maintains inventory in rooms
* Records new delivery details
* Captures supplier details
* Maintain requests
* Response
* Offers search and filter options
* Generate reports

**Step 4:Identify operations for each of class:**

Giving convenient names to above verbs as add,update,view,search, and delete, following operations are identified:

Admin Class would be able to do all operations after having authenticated account.

Admin Class can Sign-Up

Admin Class can Log-In

Admin Class can log-out

Admin Class can recover password

Admin class can add Hostel Object

Admin class can view Hostel Object

Admin class can search Hostel Object

Admin class can update(delete as well) Hostel Object

Admin class can add Room Object

Admin class can view Room Object

Admin class can search Room Object

Admin class can update(delete as well) Room Object

Admin class can add Supplier Object

Admin class can view Supplier Object

Admin class can search Supplier Object

Admin class can update(delete as well) Supplier Object

Admin class can add Request Object

Admin class can view Request Object

Admin class can search Request Object

Admin class can update(delete as well) Request Object

Admin class can add Item Object

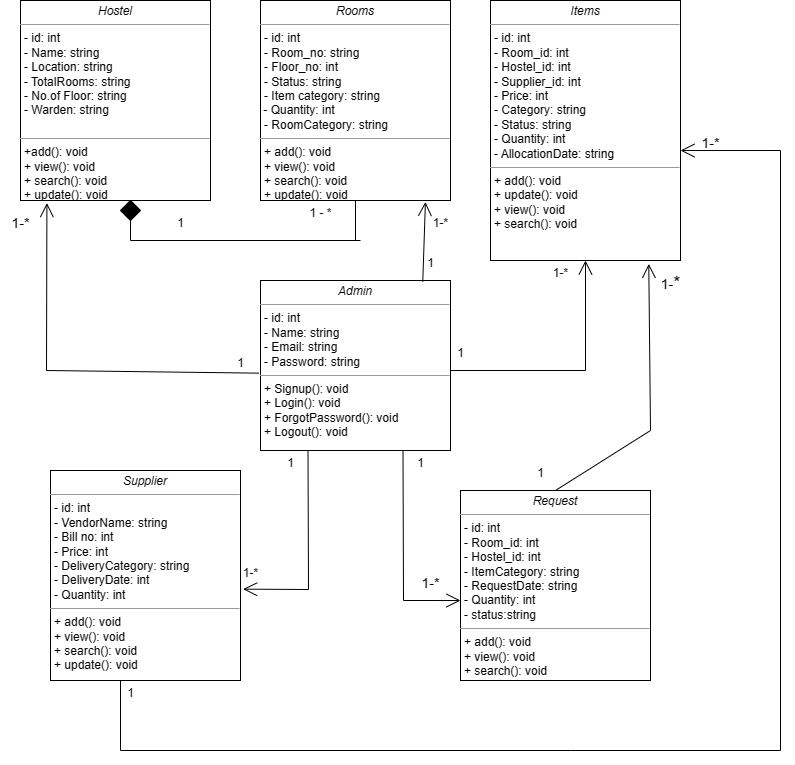
Admin class can view Item Object

Admin class can search Item Object

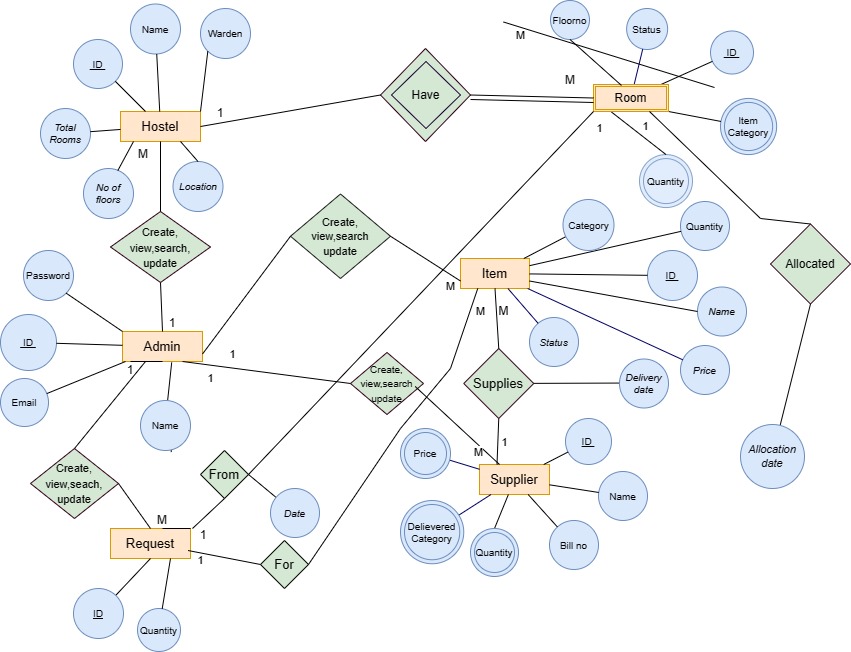
Admin class can update(delete as well) Item Object

**(-----:CLASS DIAGRAM:-----)**

The class diagram for the system includes classes such as Admin, Item, Supplier, Request, Hostel, and Rooms. These classes have attributes and methods specific to their functionality within the system. The relationships among classes are also specified.



**(-----:ER DIAGRAM:-----)**



**(-----:WIREFRAMES:-----)**

**Splash Screen**

****

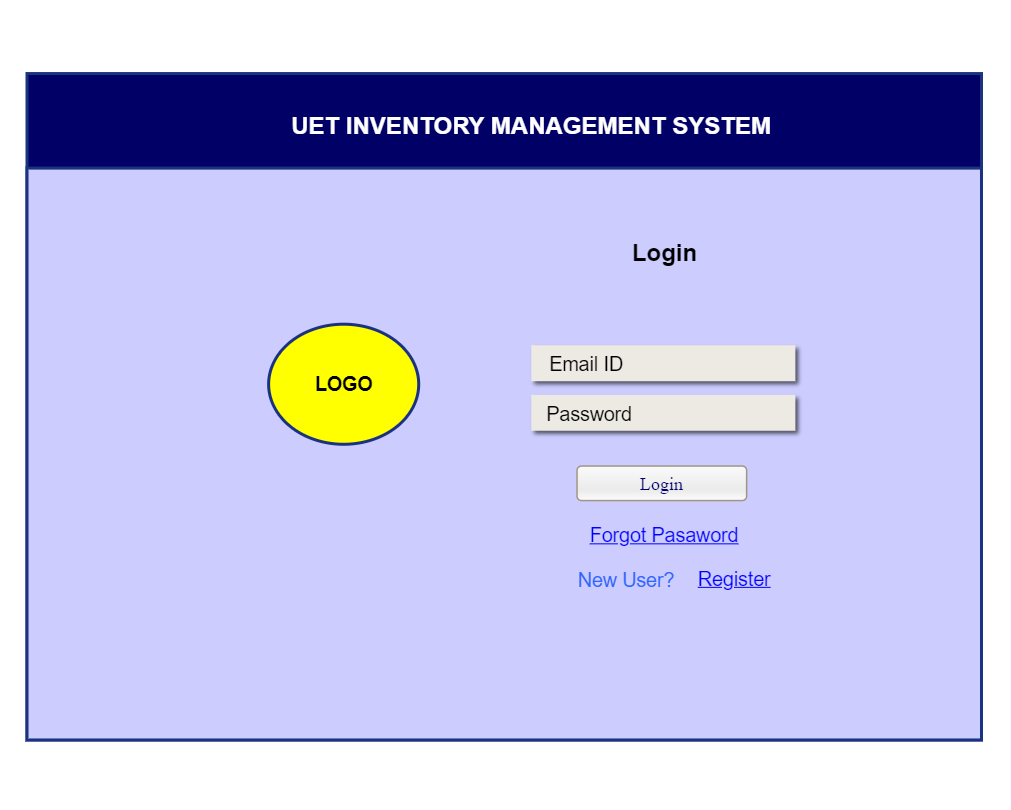
**Sign Up**



T1

|  |  |  |  |
| --- | --- | --- | --- |
| **As a User:** | | **As an Admin:** | |
| T1 | **As a user, I shall enter credential to create to create account.** | T1 | **As an admin, I shall allow user to create his/her account.** |

**Login**



T3

T2

T1

|  |  |  |  |
| --- | --- | --- | --- |
| **As a User:** | | **As an Admin:** | |
| T1 | **As a user, I shall be able to Register an account.** | T1 | **As an admin, I shall allow user to create his/her account.** |
| **T2** | **As a user, I shall be able to recover password if password forgotten.** | **T2** | **As an admin, I shall allow users to access forgot password feature.** |
| **T3** | **As a user, I shall login to account if already have Signed up.** | **T3** | **As an admin, I shall allow users to login to account.** |

**Recover Account**

****

T1

|  |  |  |  |
| --- | --- | --- | --- |
| **As a User:** | | **As an Admin:** | |
| T1 | **As a user, I shall enter email address to get recovery link.** | T1 | **As an admin, I shall allow user to get recovery link.** |

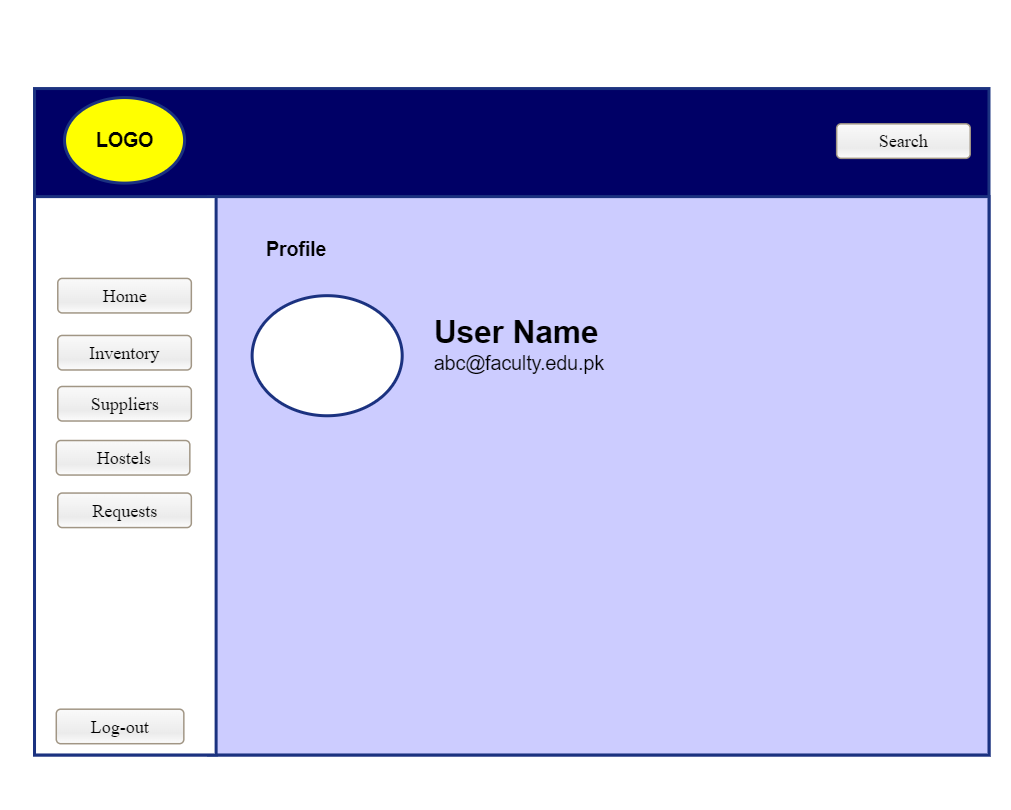
**Password Reset**

****

T1

|  |  |  |  |
| --- | --- | --- | --- |
| **As a User:** | | **As an Admin:** | |
| T1 | **As a user, I shall enter password and confirm password to update password.** | T1 | **As an admin, I shall allow user to reset password.** |

**User Profile**

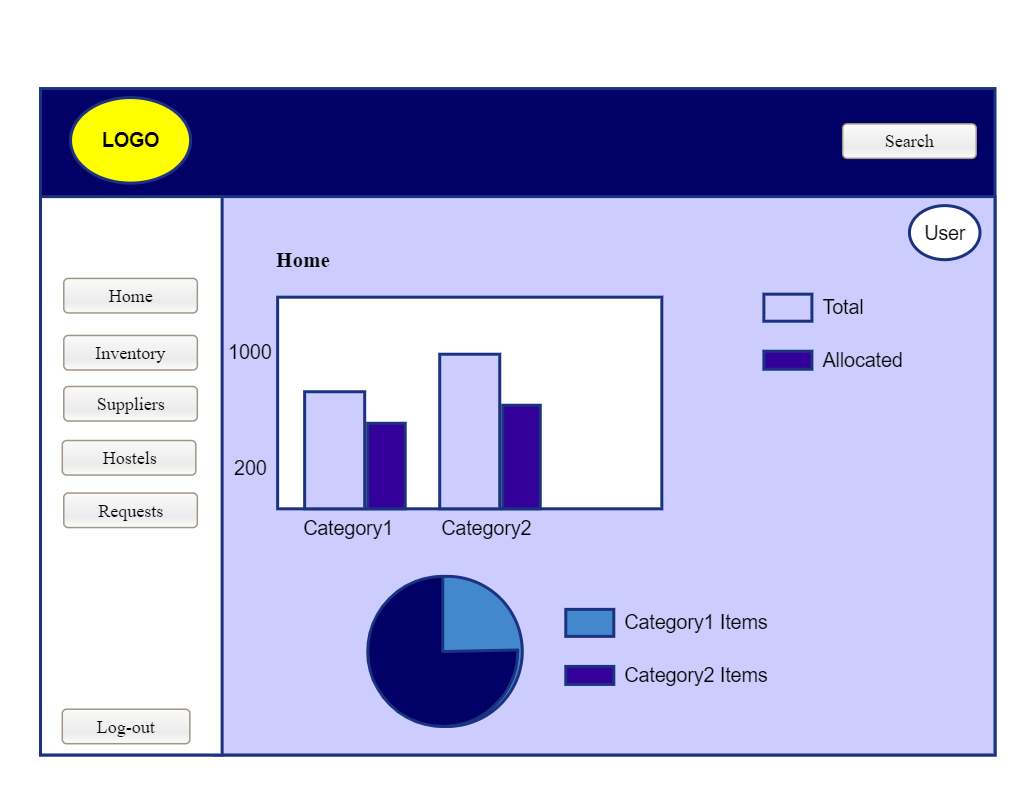
****

X

T1

|  |  |  |  |
| --- | --- | --- | --- |
| T1 | **As a user, I shall exit .** | T1 | **As an admin, I shall allow user to exit.** |

**Home**

****

T3

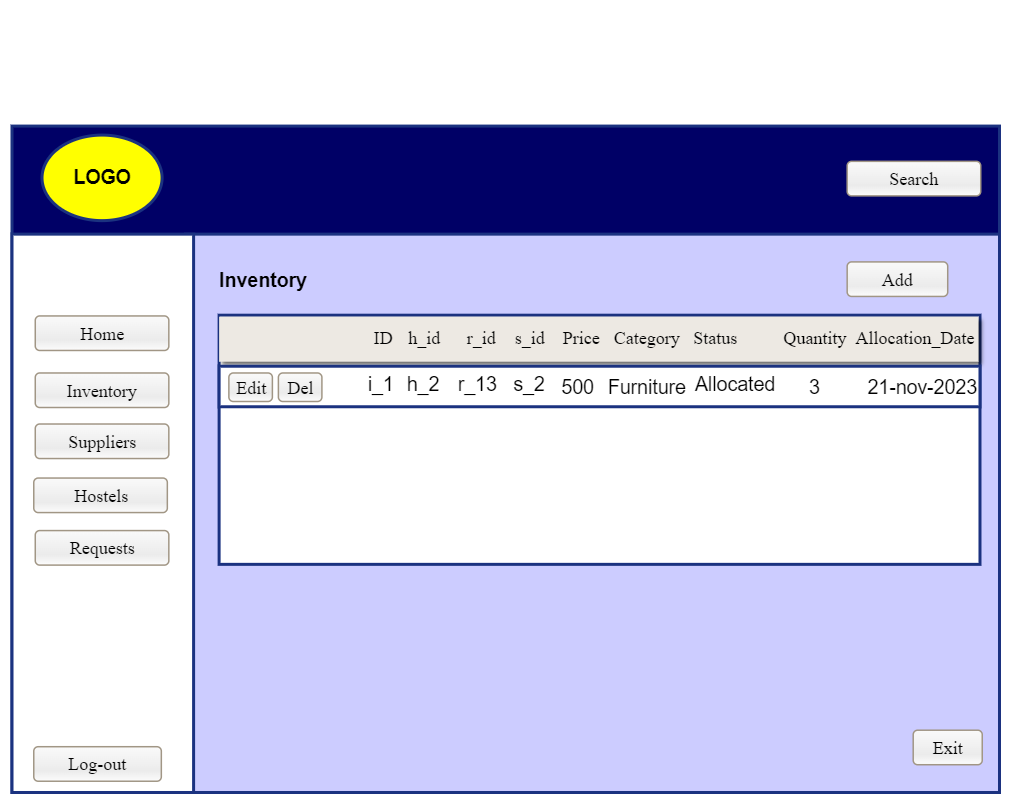
T4

T2

T1

|  |  |  |  |
| --- | --- | --- | --- |
| **As a User:** | | **As an Admin:** | |
| T1 | **As a user, I shall search desired one.** | T1 | **As an admin, I shall allow user to access desired one.** |
| T2 | **As a user, I shall to access and view desired page features.** | T2 | **As an admin, I shall allow user to access and view desired page features.** |
| T3 | **As a user, I shall view my profile.** | T3 | **As an admin, I shall allow user to view their profile.** |
| T4 | **As a user, I shall log-out from the system.** | T4 | **As an admin, I shall allow user to log-out.** |

**Inventory**

****

T1

T4

T3

T2

|  |  |  |  |
| --- | --- | --- | --- |
| T1 | **As a user, I shall add entry.** | T1 | **As an admin, I shall allow user to add entry.** |
| T2 | **As a user, I shall delete entry.** | T2 | **As an admin, I shall allow user to delete entry.** |
| T3 | **As a user, I shall edit entry.** | T3 | **As an admin, I shall allow user to edit entry.** |
| T4 | **As a user, I shall exit .** | T4 | **As an admin, I shall allow user to exit.** |

**Supplier**

****

T3

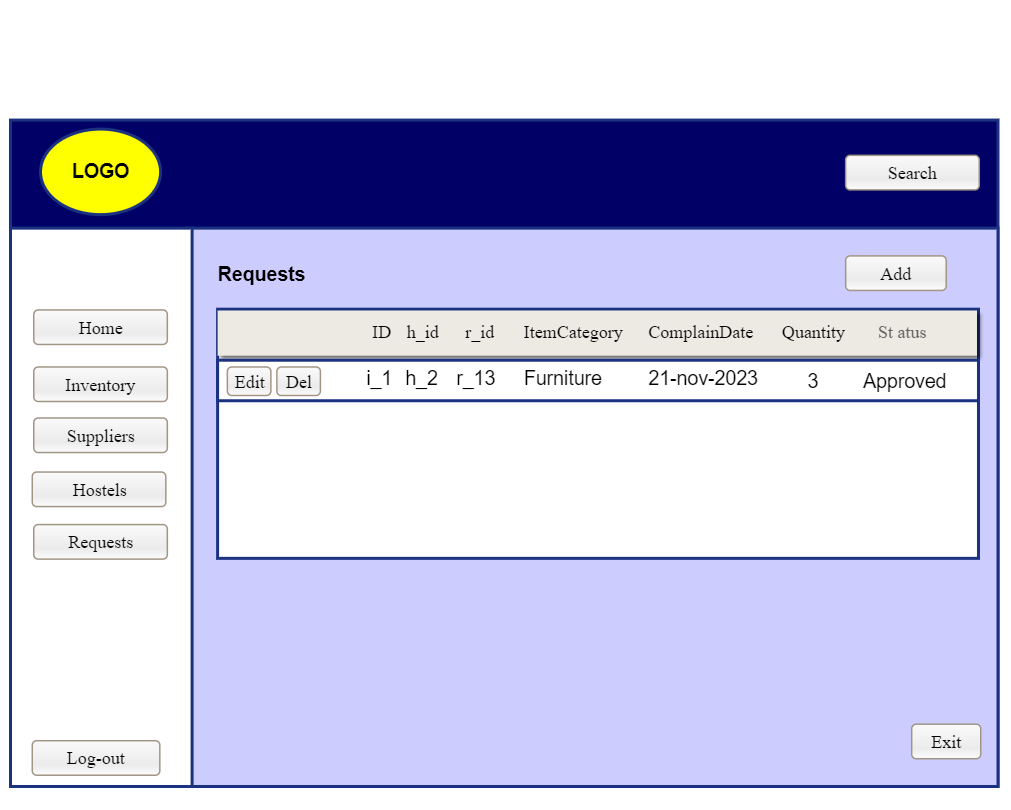
T2

T4

T1

|  |  |  |  |
| --- | --- | --- | --- |
| T1 | **As a user, I shall add entry.** | T1 | **As an admin, I shall allow user to add entry.** |
| T2 | **As a user, I shall delete entry.** | T2 | **As an admin, I shall allow user to delete entry.** |
| T3 | **As a user, I shall edit entry.** | T3 | **As an admin, I shall allow user to edit entry.** |
| T4 | **As a user, I shall exit .** | T4 | **As an admin, I shall allow user to exit.** |

**Requests**

****

T1

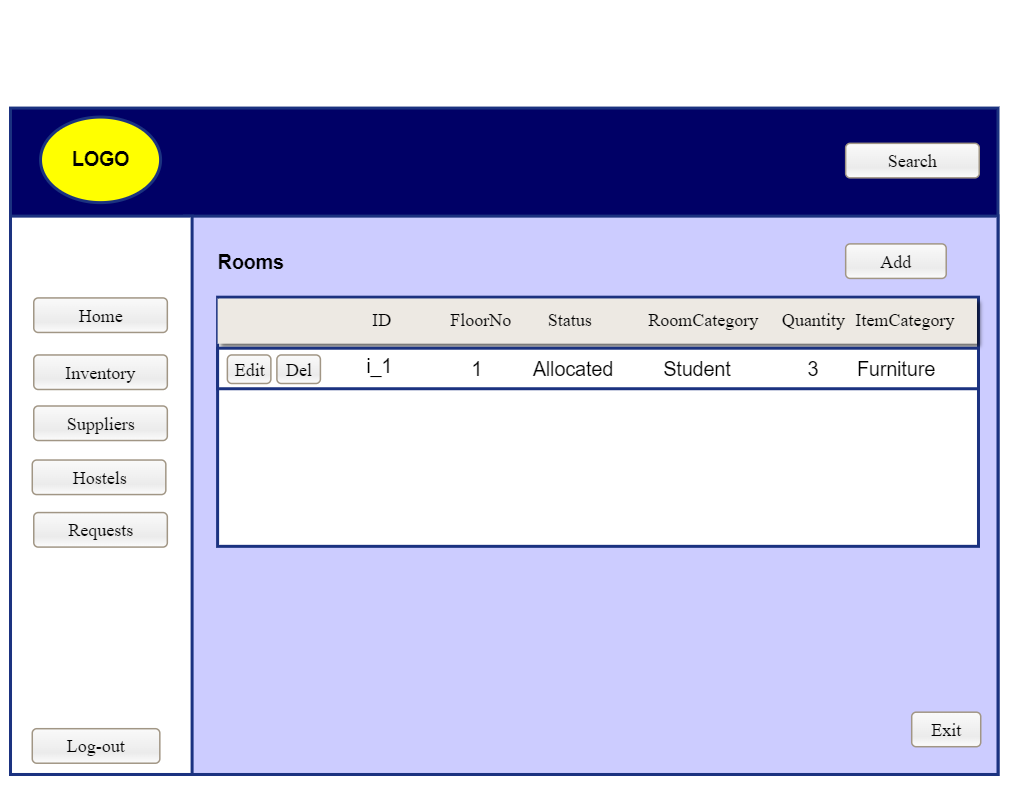
T4

T2

T3

|  |  |  |  |
| --- | --- | --- | --- |
| T1 | **As a user, I shall add entry.** | T1 | **As an admin, I shall allow user to add entry.** |
| T2 | **As a user, I shall delete entry.** | T2 | **As an admin, I shall allow user to delete entry.** |
| T3 | **As a user, I shall edit entry.** | T3 | **As an admin, I shall allow user to edit entry.** |
| T4 | **As a user, I shall exit .** | T4 | **As an admin, I shall allow user to exit.** |

**Rooms**

****

T1

T4

T2

T3

|  |  |  |  |
| --- | --- | --- | --- |
| T1 | **As a user, I shall add entry.** | T1 | **As an admin, I shall allow user to add entry.** |
| T2 | **As a user, I shall delete entry.** | T2 | **As an admin, I shall allow user to delete entry.** |
| T3 | **As a user, I shall edit entry.** | T3 | **As an admin, I shall allow user to edit entry.** |
| T4 | **As a user, I shall exit .** | T4 | **As an admin, I shall allow user to exit.** |

**Hostels**

T1

T4

T5

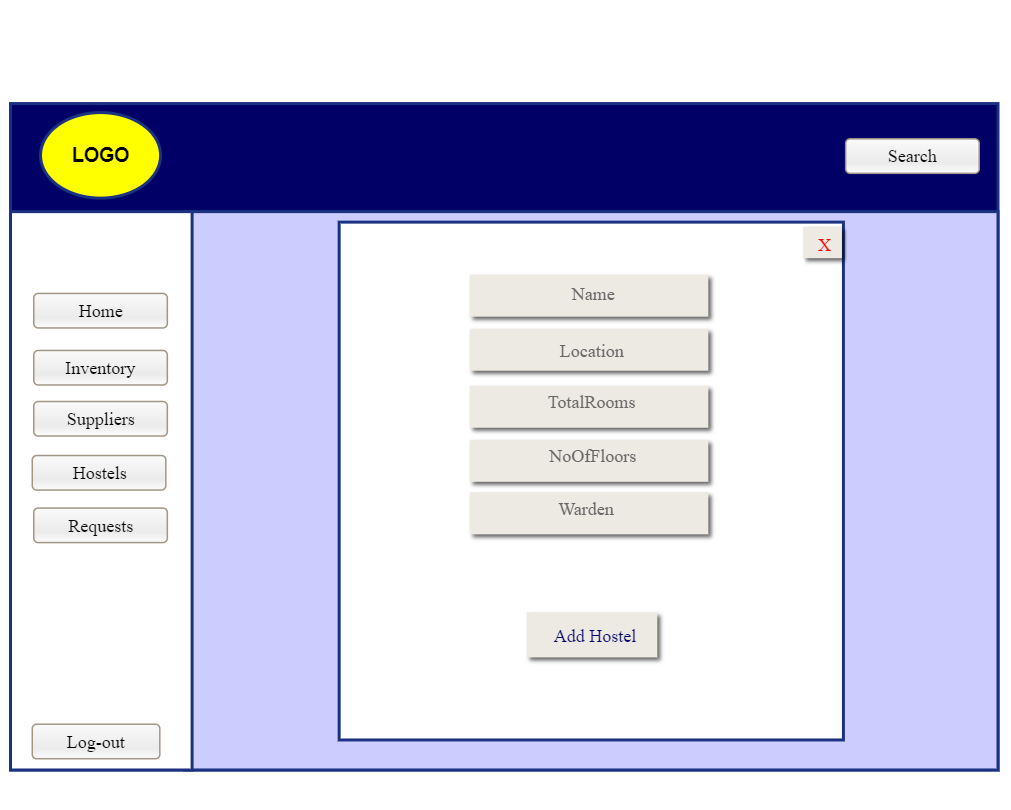
T3

T2

****

|  |  |  |  |
| --- | --- | --- | --- |
| T1 | **As a user, I shall add hostel entry.** | T1 | **As an admin, I shall allow user to add hostel entry.** |
| T2 | **As a user, I shall delete entry.** | T2 | **As an admin, I shall allow user to delete entry.** |
| T3 | **As a user, I shall edit entry.** | T3 | **As an admin, I shall allow user to edit entry.** |
| T4 | **As a user, I shall exit .** | T4 | **As an admin, I shall allow user to exit.** |
| T5 | **As a user, I shall add room entry.** | T5 | **As an admin, I shall allow user to add room entry.** |

**Add Hostels**

****

T2

T1

|  |  |  |  |
| --- | --- | --- | --- |
| T1 | **As a user, I shall add entry.** | T1 | **As an admin, I shall allow user to add entry.** |
| T2 | **As a user, I shall exit .** | T2 | **As an admin, I shall allow user to exit.** |

**Edit Hostels**

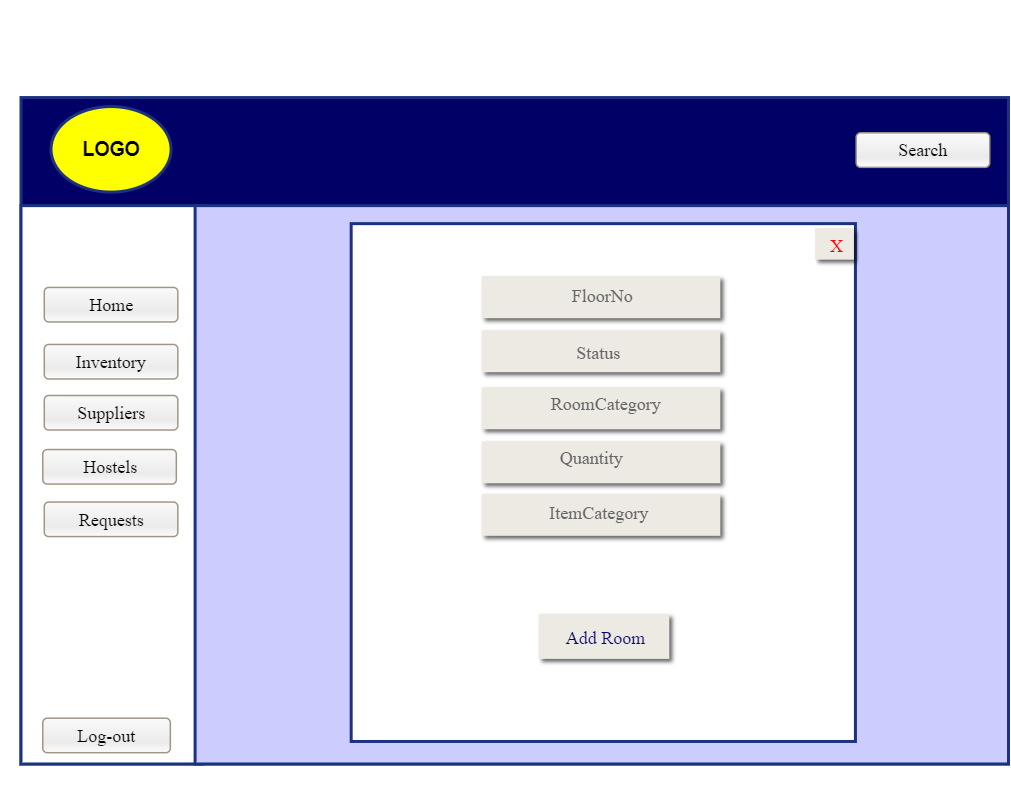
****

T2

T1

|  |  |  |  |
| --- | --- | --- | --- |
| T1 | **As a user, I shall edit entry.** | T1 | **As an admin, I shall allow user to edit entry.** |
| T2 | **As a user, I shall exit .** | T2 | **As an admin, I shall allow user to exit.** |

**Add Rooms**

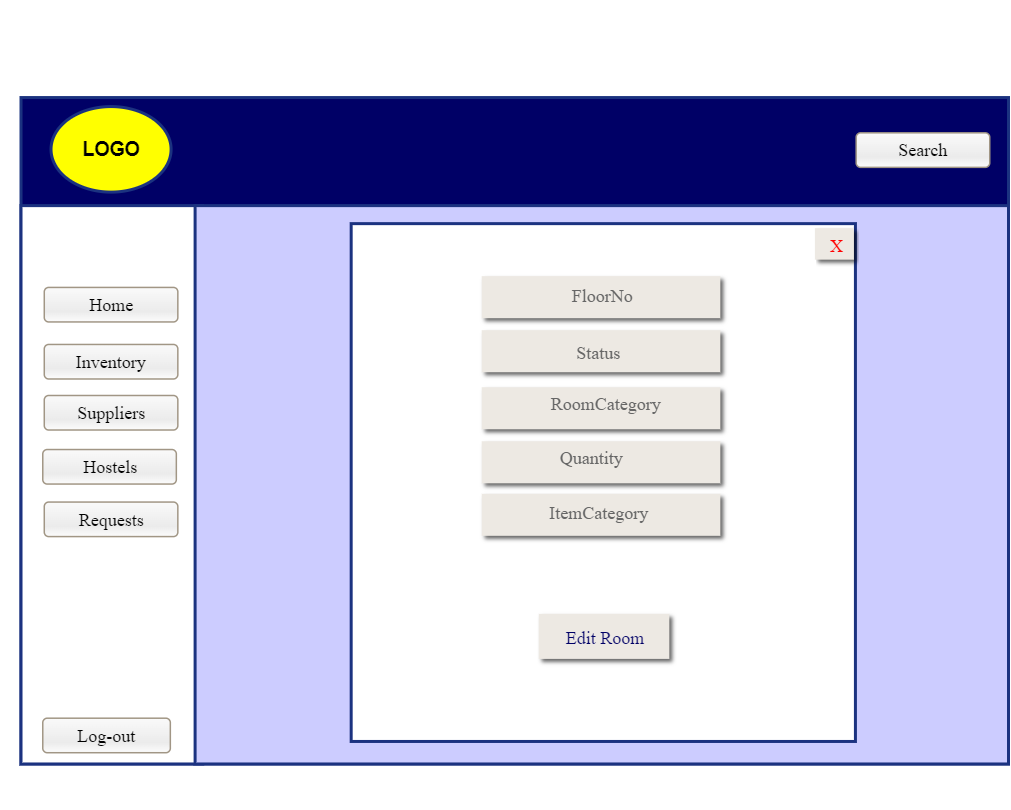
****

T2

T1

|  |  |  |  |
| --- | --- | --- | --- |
| T1 | **As a user, I shall add entry.** | T1 | **As an admin, I shall allow user to add entry.** |
| T2 | **As a user, I shall exit .** | T2 | **As an admin, I shall allow user to exit.** |

**Edit Rooms**

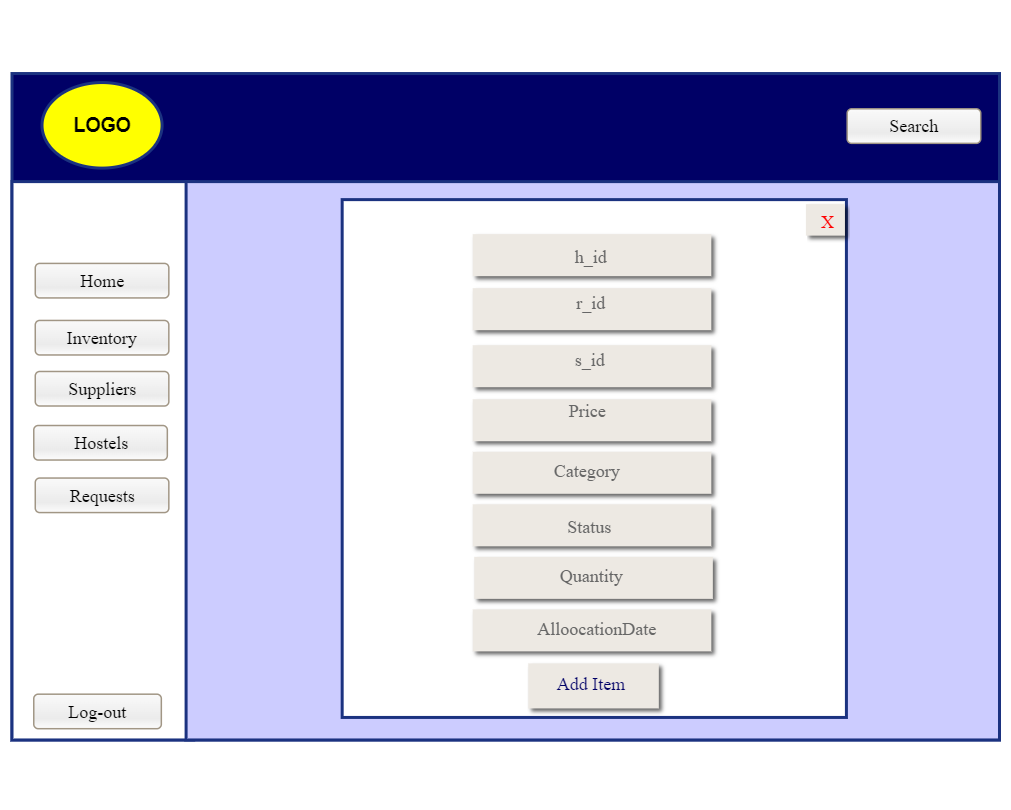
****

T2

T1

|  |  |  |  |
| --- | --- | --- | --- |
| T1 | **As a user, I shall edit entry.** | T1 | **As an admin, I shall allow user to edit entry.** |
| T2 | **As a user, I shall exit .** | T2 | **As an admin, I shall allow user to exit.** |

**Add Items**

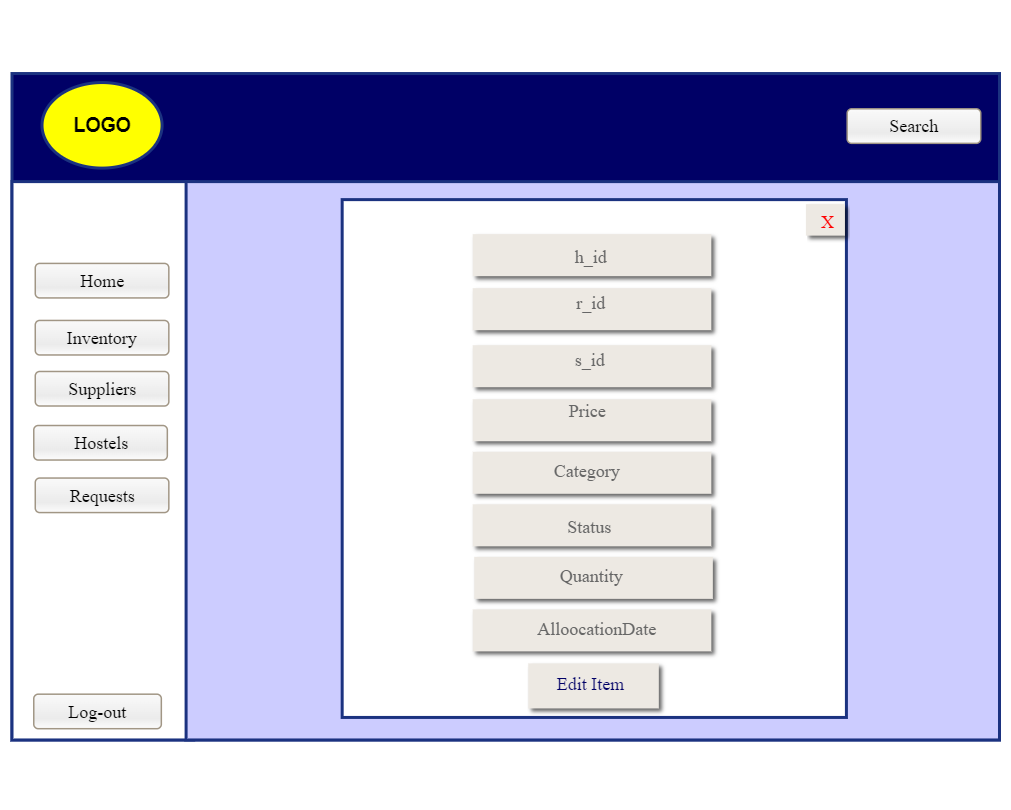
****

T2

T1

|  |  |  |  |
| --- | --- | --- | --- |
| T1 | **As a user, I shall add entry.** | T1 | **As an admin, I shall allow user to add entry.** |
| T2 | **As a user, I shall exit .** | T2 | **As an admin, I shall allow user to exit.** |

**Edit Items**

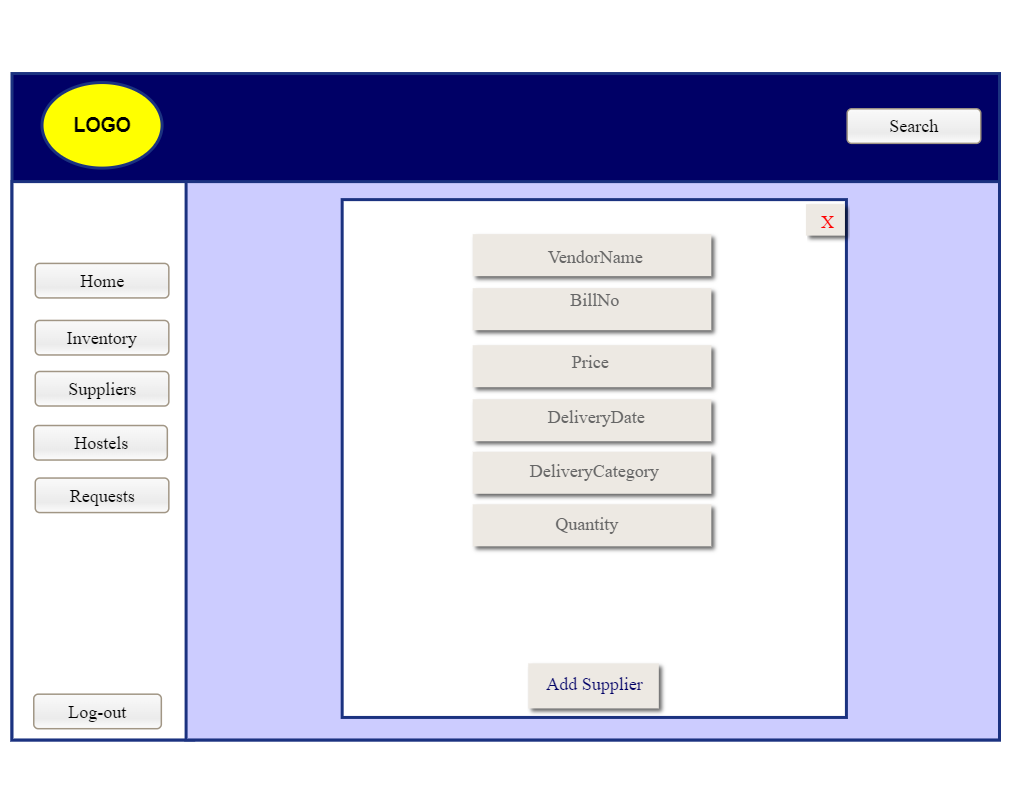
****

T2

T1

|  |  |  |  |
| --- | --- | --- | --- |
| T1 | **As a user, I shall edit entry.** | T1 | **As an admin, I shall allow user to edit entry.** |
| T2 | **As a user, I shall exit .** | T2 | **As an admin, I shall allow user to exit.** |

**Add Suppliers**

****

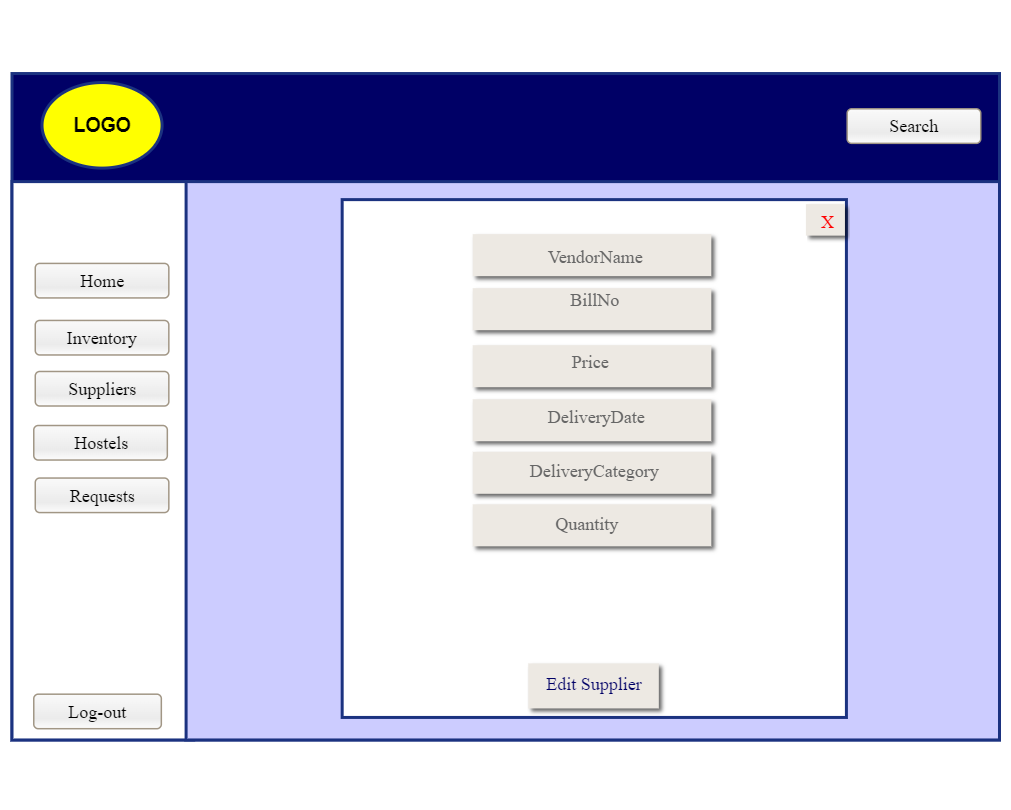
T2

T1

|  |  |  |  |
| --- | --- | --- | --- |
| T1 | **As a user, I shall add entry.** | T1 | **As an admin, I shall allow user to add entry.** |
| T2 | **As a user, I shall exit .** | T2 | **As an admin, I shall allow user to exit.** |

**Edit Suppliers**

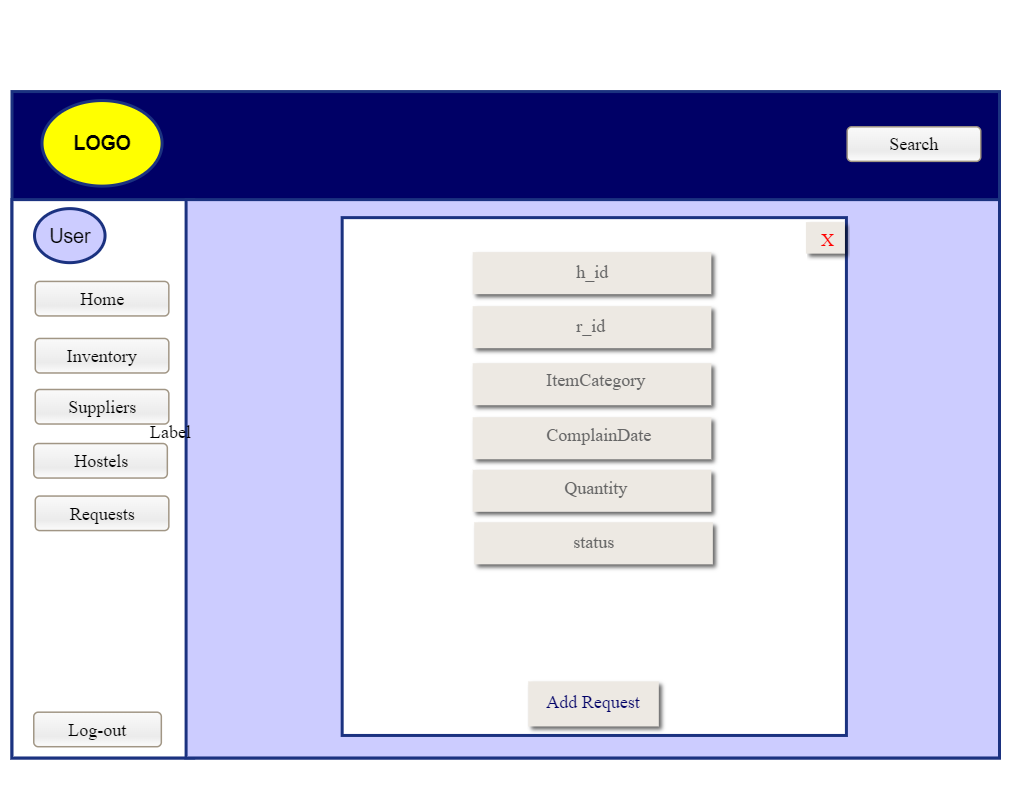
T2

****

T1

|  |  |  |  |
| --- | --- | --- | --- |
| T1 | **As a user, I shall edit entry.** | T1 | **As an admin, I shall allow user to edit entry.** |
| T2 | **As a user, I shall exit .** | T2 | **As an admin, I shall allow user to exit.** |

**Add Requests**

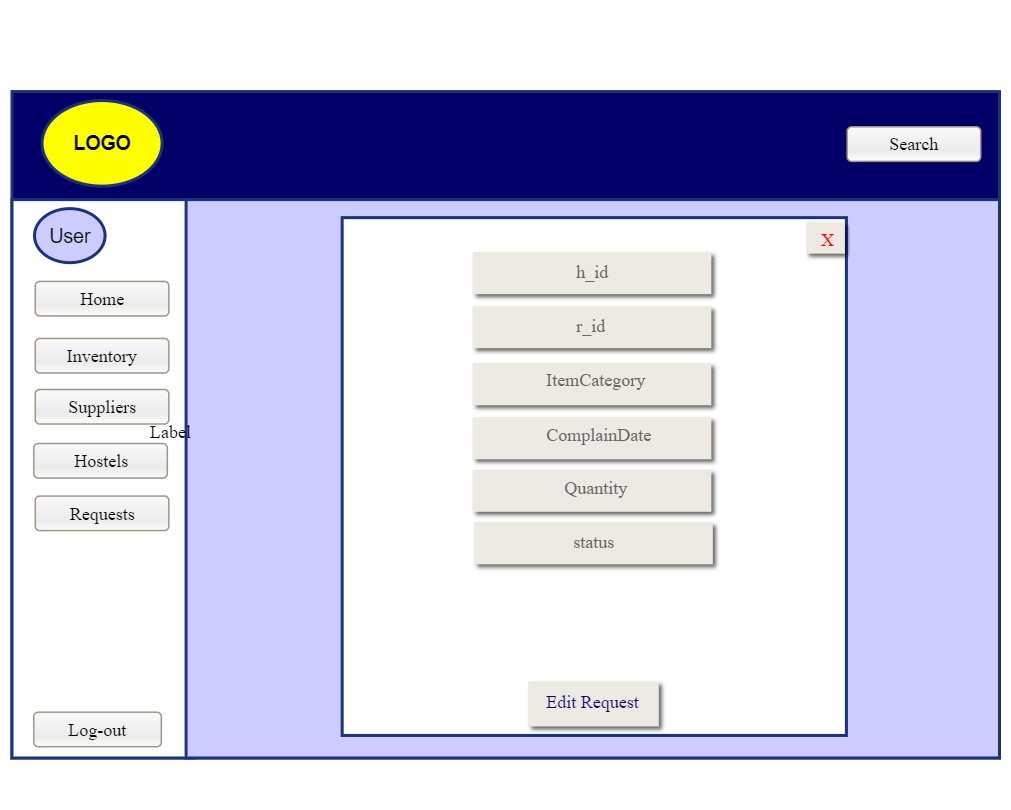
****

T2

T1

|  |  |  |  |
| --- | --- | --- | --- |
| T1 | **As a user, I shall add entry.** | T1 | **As an admin, I shall allow user to add entry.** |
| T2 | **As a user, I shall exit .** | T2 | **As an admin, I shall allow user to exit.** |

**Edit Requests**

****

T2

T1

|  |  |  |  |
| --- | --- | --- | --- |
| T1 | **As a user, I shall edit entry.** | T1 | **As an admin, I shall allow user to edit entry.** |
| T2 | **As a user, I shall exit .** | T2 | **As an admin, I shall allow user to exit.** |

**(-----:3NF DB DESIGN:-----)**

After doing normalization up-til 3NF, we have got following normalized data:

**-- Create Admin table**

CREATE TABLE Admin (

AdminId INT PRIMARY KEY,

Email VARCHAR(255),

Password VARCHAR(255),

Name VARCHAR(255)

);

**-- Insert data into the Admin table**

INSERT INTO Admin (AdminId, Email, Password, Name)

VALUES

(1, 'admin1@example.com', 'password123', 'Rasheed Admin'),

**-- Create Hostel table**

CREATE TABLE Hostel (

HostelId INT PRIMARY KEY,

Name VARCHAR(255),

Location VARCHAR(255),

NumberOfFloors INT,

Warden VARCHAR(255)

);

**-- Create HostelRooms table**

CREATE TABLE HostelRooms (

HostelId INT,

TotalRooms INT,

FOREIGN KEY (HostelId) REFERENCES Hostel(HostelId)

);

**-- Insert data into the Hostel table**

INSERT INTO Hostel (HostelId, Name, Location, NumberOfFloors, Warden)

VALUES

(1, 'Hostel A', 'City Center', 5, 'Ali'),

(2, 'Hostel B', 'Jannat chowk', 3, 'Ahmad'),

(3, 'Hostel C', 'Downtown', 7, 'Babar');

**-- Insert data into the HostelRooms table**

INSERT INTO HostelRooms (HostelId, TotalRooms)

VALUES

(1, 50),

(2, 30),

(3, 70);

**-- Create Rooms table**

CREATE TABLE Rooms (

RoomId INT PRIMARY KEY,

HostelId INT,

FloorNo INT,

Status VARCHAR(50),

RoomCategory VARCHAR(50),

FOREIGN KEY (HostelId) REFERENCES Hostel(HostelId)

);

**-- Create Items table**

CREATE TABLE Items (

ItemId INT PRIMARY KEY,

RoomId INT,

ItemName VARCHAR(255),

FOREIGN KEY (RoomId) REFERENCES Rooms(RoomId)

);

**-- Create ItemsCategory table**

CREATE TABLE ItemsCategory (

CategoryId INT PRIMARY KEY,

CategoryName VARCHAR(255)

);

**-- Insert data into the Rooms table**

INSERT INTO Rooms (RoomId, HostelId, FloorNo, Status, RoomCategory)

VALUES

(101, 1, 1, 'Occupied', 'Public'),

(102, 2, 2, 'Vacant', 'Student'),

(103, 3, 3, 'Occupied', 'Faculty');

**-- Insert data into the Items table**

INSERT INTO Items (ItemId, RoomId, ItemName)

VALUES

(1, 101, 'Desk'),

(2, 102, 'Chair'),

(3, 103, 'Lamp');

**-- Insert data into the ItemsCategory table**

INSERT INTO ItemsCategory (CategoryId, CategoryName)

VALUES

(1, 'Furniture'),

(2, 'Electronics'),

(3, 'Hardware');

**-- Create Suppliers table**

CREATE TABLE Suppliers (

SupplierId INT PRIMARY KEY,

Name VARCHAR(255),

BillNo VARCHAR(50),

DeliveredDate DATE

);

**-- Create SupplierItems table**

CREATE TABLE SupplierItems (

ItemId INT PRIMARY KEY,

SupplierId INT,

ItemName VARCHAR(255),

ItemPrice DECIMAL(10, 2), -- Assuming a decimal data type for price

FOREIGN KEY (SupplierId) REFERENCES Suppliers(SupplierId)

);

**-- Create SupplierItemsCategory table**

CREATE TABLE SupplierItemsCategory (

CategoryId INT PRIMARY KEY,

CategoryName VARCHAR(255)

);

**-- Insert data into the Suppliers table**

INSERT INTO Suppliers (SupplierId, Name, BillNo, DeliveredDate)

VALUES

(1, 'Supplier A', 'B12345', '2023-01-15'),

(2, 'Supplier B', 'C67890', '2023-02-28'),

(3, 'Supplier C', 'D54321', '2023-03-10');

**-- Insert data into the SupplierItems table**

INSERT INTO SupplierItems (ItemId, SupplierId, ItemName, ItemPrice)

VALUES

(1, 1, 'Desk', 150.00),

(2, 2, 'Chair', 50.00),

(3, 3, 'Lamp', 25.00);

**-- Insert data into the SupplierItemsCategory table**

INSERT INTO SupplierItemsCategory (CategoryId, CategoryName)

VALUES

(1, 'Furniture'),

(2, 'Electronics'),

(3, 'Hardware');

**-- Create Requests table**

CREATE TABLE Requests (

RequestId INT PRIMARY KEY,

RequestDate DATE

);

**-- Create RequestItem table**

CREATE TABLE RequestItem (

RoomId INT,

HostelId INT,

ItemName VARCHAR(255),

Quantity INT,

FOREIGN KEY (RoomId, HostelId) REFERENCES Rooms(RoomId, HostelId)

);

**-- Insert data into the Requests table**

INSERT INTO Requests (RequestId, RequestDate)

VALUES

(1, '2023-01-20'),

(2, '2023-02-15'),

(3, '2023-03-05');

**-- Insert data into the RequestItem table**

INSERT INTO RequestItem (RoomId, HostelId, ItemName, Quantity)

VALUES

(101, 1, 'Desk', 2),

(102, 2, 'Chair', 3),

(103, 3, 'Lamp', 1);