**CENTURION UNIVERSITY OF TECHNOLOGY AND**

**MANAGEMENT**



**MASTER IN COMPUTER APPLICATION**

**of the**

**DEPARTMENT OF APPLIED SCIENCE**

**A PROJECT REPORT ON**

**“****Remotel - Hostel Room Management System”**

**Submitted by**

**Manish Ram (240720100067)**

**Under the Guidance of**

**Prof. Rakesh Kumar Ray**

**In partial fulfilment for the award of the degree**

**Of**

**MASTER IN COMPUTER APPLICATION**

**CENTURION UNIVERSITY OF TECHNOLOGY AND MANAGEMENT**

**BHUBANESWAR, ODISHA**

**2024-2026**

**CENTURION UNIVERSITY OF TECHNOLOGY AND MANAGEMENT, ODISHA**

**SCHOOL OF APPLIED SCIENCES, BHUBANESWAR CAMPUS**

**DEPARTMENT OF MASTERS IN COMPUTER APPLICATION(MCA)**

***BONAFIDE CERTIFICATE***

This is to be certified that the project report “**Remotel - Hostel Room Management System**” has been submitted for the Masters in Computer Application (MCA), School of Applied Sciences, CUTM, Jatani during the academic year 2024-2025 is a persuasive piece of project work carried out by **“Manish Ram**”, who carried out the project work under the guidance of **Prof. Rakesh Kumar Ray** and also certified that the above-mentioned project has been duly carried out as per the norm of the school statutes of the university.

Signature of HOD of MCA Signature of Project Guide

**Prof. Rakesh Kumar Ray Prof. Rakesh Kumar Ray**

**CENTURION UNIVERSITY OF TECHNOLOGY AND MANAGEMENT (CUTM)**

**BHUBANESWAR CAMPUS, ODISHA**

**DECLARATION**

I am **Manish Ram (240720100067),** Bonafide student of CENTURION UNIVERSITY OF TECHNOLOGY AND MANAGEMENT, hereby declare that the dissertation entitled, **“Remotel - Hostel Room Management System”** is an original work and data provided in the study is authentic one, carried out by me under the guidance of **Prof. Rakesh Kumar Ray**, in partial fulfilment of the requirements for the MASTER OF COMPUTER APPLICATION, of the CENTURION UNIVERSITY OF TECHNOLOGY AND MANAGEMENT, BHUBANESWAR during the academic year 2024-2025.

Signature of HOD of MCA Signature of Project Guide

**Prof. Rakesh Kumar Ray Prof. Rakesh Kumar Ray**

**ACKNOWLEDGEMENT**

This project report consumed a huge amount of work, research, passion and dedication. In performing this project, we had to take help and guidance from some respected people, who deserve our greatest gratitude.

We owe a profound gratitude to our project in-charge, **Prof. Rakesh Kumar Ray** who not only introduced us to the methodology of work but also took a keen interest in our project and guided us all along.

In addition, we would also like to thank our respected Dean, **Dr Sujata Chakravarty** and the Head of The Department, Prof **Rakesh Kumar Ray**, for giving us the support and guidance to work on this report. We are also thankful to the college for providing us with the necessary resources and also many thanks to the staff of college for their valuable cooperation.

Lastly, we thank our classmates who made valuable suggestions which gave us inspiration to improve our project. We thank them all for their help to complete the report. We express our gratitude towards the countless individuals who shared their knowledge and helped us in this report.

**Manish Ram (240720100067)**

**ABSTRACT**

*Remotel* is a desktop-based application developed to streamline and automate hostel room management processes. Traditionally, room allotment and user request handling in hostels are done manually, leading to inefficiencies, data inconsistencies, and administrative overhead. Remotel addresses these challenges by providing a centralized and user-friendly system that allows users to register, request rooms, and manage their data seamlessly.

Built using **Java Swing** for the graphical user interface and **SQLite** for local database management, the system follows the **DAO and MVC design patterns** to ensure modularity, maintainability, and clear separation of concerns. Administrators are equipped with a dedicated dashboard to review requests, manage room availability, and update records efficiently.

The application not only automates tedious tasks but also minimizes errors associated with manual record-keeping. Future enhancements may include adding network support or developing a mobile version for broader accessibility.

**Table of Contents**

[ **INTRODUCTION** 7](#_Toc197928374)

[ **SOFTWARE AND HARDWARE REQUIREMENTS** 9](#_Toc197928375)

[ **THEORY OF OPERATION** 10](#_Toc197928376)

[ **IMPLEMENTATION** 11](#_Toc197928377)

[Overview: 11](#_Toc197928378)

[ **SOURCE CODE AND OUTPUT** 13](#_Toc197928379)

[ Source Code: 13](#_Toc197928380)

[ Output: 52](#_Toc197928381)

[ **CONCLUSION** 54](#_Toc197928382)

[ **REFERENCE** 55](#_Toc197928383)

[ **ASSESSMENT** 56](#_Toc197928384)

[ **COURSE OUTCOME (COs) ATTAINMENT** 57](#_Toc197928385)

# **INTRODUCTION**

* Remotelis a desktop-based application developed to simplify and automate the process of hostel room management. It provides a centralized platform for handling user registrations, room requests, and administrative operations efficiently. The application is built using **Java** for backend logic and **Java Swing** for the user interface, with **SQLite** serving as the local database for data persistence.
* Designed with both functionality and usability in mind, Remotel follows the **DAO (Data Access Object)** and **MVC (Model-View-Controller)** design patterns to maintain clear separation between data handling, logic, and presentation. The system enables students to register and request rooms seamlessly, while providing administrators with tools to review and manage requests, assign rooms, and maintain records—all from a single dashboard.
* By replacing manual processes with automation, Remotel reduces human error, improves operational efficiency, and enhances the overall user experience for both students and administrators.

## **Key Features**

* **User Registration and Login:**

**Students can create accounts and securely log in to access hostel services.**

* **Room Request and Booking System:**

**Users can submit room requests, which are processed by the admin through an approval system.**

* **Admin Dashboard:**

**A dedicated interface for administrators to view, approve/deny room requests, and manage room availability and user details.**

* **Database Integration with SQLite:**

**All data related to users, rooms, and requests is stored in a local SQLite database for quick and reliable access.**

* **Clear Separation of Concerns:**

**The system is built using the DAO (Data Access Object) and MVC (Model-View-Controller) patterns, ensuring clean code organization and easier maintenance.**

* **Status Tracking and Updates:**

**Real-time status updates for room requests and room assignments.**

* **Standalone Desktop Application:**

**Fully functional without requiring internet connectivity, ideal for local deployment in hostel offices.**

## AIM:

The aim of the *Remotel* project is to **develop a desktop-based application** that automates and simplifies the hostel room management process. It seeks to:

* Replace the traditional manual system with a **digital solution**.
* Ensure **efficient handling of room allotments** and user requests.
* Provide an **intuitive interface** for both students and administrators.
* Maintain **accurate and secure records** through database integration.
* Reduce administrative burden and **minimize human error**.

## SCOPE:

The *Remotel* application is designed to address the needs of hostel administrators and residents by providing an efficient digital platform for room management. The scope of the project includes:

* **User Management:** Registration, login, and profile handling for hostel residents.
* **Room Allotment:** Online submission and processing of room requests with automated status tracking.
* **Admin Operations:** Tools for administrators to manage room availability, view user requests, and assign rooms efficiently.
* **Local Deployment:** As a desktop-based application, Remotel is intended for use within the local hostel network without relying on continuous internet connectivity.
* **Scalability:** The application architecture allows for future expansion, such as adding networking features or migrating to a web or mobile-based platform.

Online Course Registration System is Web-based registration software that helps

you to register courses online. It is ideal for schools, educational camps, corporate

training programs, and online training programs. It also provides time to time

current status information related to courses. It can help for the student need to

register by giving necessary details, for the desired course.

In Online Course Registration we use PHP and MySQL database. It is web-based

registration software that helps you to register courses online. It is ideal schools,

educational camps, and corporate training program. It has two modules i.e.

Admin and Student.

Admin:

Admin will register the student and provide username; password and

pincode.Pincode will be used when a student enrolls for the course. Admin can

manage session, semester, departments, course, students, students log Admin

Panel. Admin can also add/ delete latest news and updates related to students.

Student:

Student can log in with valid reg no and password provided by admin. Student can

enroll for any course and can printout of the registered course.

Purpose:-

The purpose of project is to build an application program to reduce manual work

for managing the course through internet. This application has good appearance

and easy to operate. It is very simple and easy to access. This project provides lots

of features to manage in very well manner. This project contains advance

modules which make the backend system very powerfu

# **SOFTWARE AND HARDWARE REQUIREMENTS**

## SOFTWARE REQUIREMENTS

* Programming Language: Java (JDK 8 or higher)
* IDE: Eclipse / IntelliJ IDEA / NetBeans (any Java-supported IDE)
* UI Framework: Java Swing
* Database: SQLite
* Database Browser: SQLite DB Browser (optional for visualization)
* Operating System: Windows 7 or later / Linux / macOS
* Java Runtime: Java Runtime Environment (JRE 8 or higher)

## HARDWARE REQUIREMENTS

* Processor: Intel i3 or higher
* RAM: Minimum 4 GB (8 GB recommended)
* Hard Disk: Minimum 200 MB free space
* Display: Minimum 1024x768 resolution
* Input Devices: Keyboard and Mouse

# **THEORY OF OPERATION**



The project Remotel ran through following steps:

1. User Registration/Login

* New users register with basic details.
* Existing users log in using credentials.

2. Room Request Submission

* Logged-in users browse available rooms.
* Users submit room allotment requests.

3. Admin Review

* Admin accesses the dashboard.
* Admin reviews pending requests.

4. Approval or Rejection

* Admin approves or denies room requests.
* Status is updated in the system.

5. Database Update

* Room status (available/booked) is updated.
* User assignment details are saved.

6. User Notification

* User is notified of the room allotment decision

# **IMPLEMENTATION**

## Overview:

* The implementation phase of the *Remotel* project involved the development of a desktop-based hostel room management system using Java technologies and SQLite database. This chapter outlines the step-by-step process followed to build the system, from setting up the development environment to coding the application logic and testing each module thoroughly.

## Development Environment Setup

* Before initiating development, the following tools and setup were configured to ensure smooth project execution:
* **Install Java JDK 8+**: The core language used for developing the system.
* **Install SQLite**: Lightweight, file-based relational database for local data storage.
* **Set Up IDE**: Eclipse or IntelliJ IDEA configured with necessary plugins for Java Swing and database access.
* **Install SQLite JDBC Driver**: Added to the project libraries to enable Java-SQLite integration.

## Application Development

* The development process was divided into the following key components:
* **Design Database:** The SQLite database schema was designed to support the main functionalities of the system. Key tables include:
* **Users Table**: Stores user login data and personal information.
* **Rooms Table**: Stores room numbers, status (available/booked), and details.
* **Requests Table**: Manages room booking requests and their statuses (pending, approved, denied).

## Back-end Development

* The core application logic was implemented using Java, structured with DAO and MVC design patterns:
* **DAO Layer**: Interfaces for performing operations like addUser(), getRoomStatus(), and updateRequest().
* **Model Classes**: Represented entities like User, Room, and Request.
* **Service Classes**: Contained business logic for handling room assignments and admin decisions.

## Front-end Development

* The graphical user interface (GUI) was developed using **Java Swing**, offering a clean and interactive layout:
* **User Forms**: For registration, login, and room request submissions.
* **Admin Panel**: For reviewing requests, updating room details, and managing users.
* **Dialogs and Tables**: Used to display information dynamically based on database data.

## Testing

* Extensive testing was conducted to ensure the system’s reliability:
* **Unit Testing**: Core functions like database CRUD operations, login, and request processing were tested using JUnit.
* **Manual Testing**: Each form and feature was manually tested to check usability and correctness.
* **Edge Case Handling**: Scenarios like duplicate requests, invalid login attempts, and full occupancy were tested and handled gracefully.

## Deployment

* Since Remotel is a **desktop application**, deployment was done locally:
* **Packaging**: The project was compiled into .jar executable format using the IDE’s build tools.
* **Distribution**: The JAR file, along with the SQLite database, can be distributed to run on any compatible system with Java installed.
* **Database File Setup**: The SQLite .db file is placed in the working directory for persistent data access.

## Key Functionalities Implemented

* **User Registration & Login**: Secure access to the system based on user roles.
* **Room Request Submission**: Users can submit requests for room allotment.
* **Admin Dashboard**: View, approve, or deny room requests.
* **Database Integration**: Real-time updates to room availability and user assignments.
* **Status Notifications**: Users are notified of their request outcomes.

# **SOURCE CODE AND OUTPUT**

## Source Code:

**File name: RoomDAO.java**

package com.hostel.dao;

import com.hostel.model.Room;

import com.hostel.util.DatabaseUtil;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

public class RoomDAO {

public boolean addRoom(Room room) {

String query = "INSERT INTO rooms (room\_number, room\_type, is\_occupied, status) VALUES (?, ?, ?, ?)";

try (Connection conn = DatabaseUtil.getConnection();

PreparedStatement pstmt = conn.prepareStatement(query)) {

pstmt.setString(1, room.getRoomNumber());

pstmt.setString(2, room.getRoomType());

pstmt.setBoolean(3, room.isOccupied());

pstmt.setString(4, room.getStatus());

return pstmt.executeUpdate() > 0;

} catch (SQLException e) {

e.printStackTrace();

return false;

}

}

public List<Room> getAllRooms() {

List<Room> rooms = new ArrayList<>();

String query = "SELECT \* FROM rooms";

try (Connection conn = DatabaseUtil.getConnection();

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(query)) {

while (rs.next()) {

rooms.add(extractRoomFromResultSet(rs));

}

} catch (SQLException e) {

e.printStackTrace();

}

return rooms;

}

public List<Room> getAvailableRooms() {

List<Room> rooms = new ArrayList<>();

String query = "SELECT \* FROM rooms WHERE status = 'available' AND is\_occupied = false";

try (Connection conn = DatabaseUtil.getConnection();

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(query)) {

while (rs.next()) {

rooms.add(extractRoomFromResultSet(rs));

}

} catch (SQLException e) {

e.printStackTrace();

}

return rooms;

}

public boolean updateRoomStatus(int roomId, String status, boolean isOccupied, Integer occupantId) {

String query = "UPDATE rooms SET status = ?, is\_occupied = ?, occupant\_id = ? WHERE room\_id = ?";

try (Connection conn = DatabaseUtil.getConnection();

PreparedStatement pstmt = conn.prepareStatement(query)) {

pstmt.setString(1, status);

pstmt.setBoolean(2, isOccupied);

if (occupantId != null) {

pstmt.setInt(3, occupantId);

} else {

pstmt.setNull(3, Types.INTEGER);

}

pstmt.setInt(4, roomId);

return pstmt.executeUpdate() > 0;

} catch (SQLException e) {

e.printStackTrace();

return false;

}

}

public boolean allocateRoom(int roomId, int userId, Date expectedCheckoutDate) {

Connection conn = null;

try {

conn = DatabaseUtil.getConnection();

conn.setAutoCommit(false); // Start transaction

// Update room status

String updateRoomQuery = "UPDATE rooms SET is\_occupied = ?, occupant\_id = ?, status = ?, allocation\_date = CURRENT\_TIMESTAMP, expected\_checkout\_date = ? WHERE room\_id = ?";

try (PreparedStatement pstmt = conn.prepareStatement(updateRoomQuery)) {

pstmt.setBoolean(1, true);

pstmt.setInt(2, userId);

pstmt.setString(3, "occupied");

pstmt.setTimestamp(4, new Timestamp(expectedCheckoutDate.getTime()));

pstmt.setInt(5, roomId);

pstmt.executeUpdate();

}

// Add entry to allocation history

String insertHistoryQuery = "INSERT INTO room\_allocation\_history (room\_id, user\_id, allocation\_date, status) VALUES (?, ?, CURRENT\_TIMESTAMP, 'active')";

try (PreparedStatement pstmt = conn.prepareStatement(insertHistoryQuery)) {

pstmt.setInt(1, roomId);

pstmt.setInt(2, userId);

pstmt.executeUpdate();

}

conn.commit();

return true;

} catch (SQLException e) {

if (conn != null) {

try {

conn.rollback();

} catch (SQLException ex) {

ex.printStackTrace();

}

}

e.printStackTrace();

return false;

} finally {

if (conn != null) {

try {

conn.setAutoCommit(true);

conn.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

public boolean deallocateRoom(int roomId, String remarks) {

Connection conn = null;

try {

conn = DatabaseUtil.getConnection();

conn.setAutoCommit(false); // Start transaction

// Update room status

String updateRoomQuery = "UPDATE rooms SET is\_occupied = FALSE, occupant\_id = NULL, status = 'available', allocation\_date = NULL, expected\_checkout\_date = NULL WHERE room\_id = ?";

try (PreparedStatement pstmt = conn.prepareStatement(updateRoomQuery)) {

pstmt.setInt(1, roomId);

pstmt.executeUpdate();

}

// Update allocation history

String updateHistoryQuery = "UPDATE room\_allocation\_history SET checkout\_date = CURRENT\_TIMESTAMP, status = 'completed', remarks = ? WHERE room\_id = ? AND status = 'active'";

try (PreparedStatement pstmt = conn.prepareStatement(updateHistoryQuery)) {

pstmt.setString(1, remarks);

pstmt.setInt(2, roomId);

pstmt.executeUpdate();

}

conn.commit();

return true;

} catch (SQLException e) {

if (conn != null) {

try {

conn.rollback();

} catch (SQLException ex) {

ex.printStackTrace();

}

}

e.printStackTrace();

return false;

} finally {

if (conn != null) {

try {

conn.setAutoCommit(true);

conn.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

public List<Room> getRoomsWithAllocationInfo() {

List<Room> rooms = new ArrayList<>();

String query = """

SELECT r.\*, u.name as occupant\_name

FROM rooms r

LEFT JOIN users u ON r.occupant\_id = u.id

ORDER BY r.room\_number

""";

try (Connection conn = DatabaseUtil.getConnection();

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(query))

while (rs.next()) {

Room room = extractRoomFromResultSet(rs);

// Add occupant name if available

if (rs.getString("occupant\_name") != null) {

room.setOccupantName(rs.getString("occupant\_name"));

}

rooms.add(room);

}

} catch (SQLException e) {

e.printStackTrace();

}

return rooms;

}

public List<Map<String, Object>> getAllocationHistory(int roomId) {

List<Map<String, Object>> history = new ArrayList<>();

String query = """

SELECT rh.\*, r.room\_number, u.name as user\_name

FROM room\_allocation\_history rh

JOIN rooms r ON rh.room\_id = r.room\_id

JOIN users u ON rh.user\_id = u.id

WHERE rh.room\_id = ?

ORDER BY rh.allocation\_date DESC

""";

try (Connection conn = DatabaseUtil.getConnection();

PreparedStatement pstmt = conn.prepareStatement(query))

pstmt.setInt(1, roomId);

try (ResultSet rs = pstmt.executeQuery()) {

while (rs.next()) {

Map<String, Object> record = new HashMap<>();

record.put("allocation\_id", rs.getInt("allocation\_id"));

record.put("room\_number", rs.getString("room\_number"));

record.put("user\_name", rs.getString("user\_name"));

record.put("allocation\_date", rs.getTimestamp("allocation\_date"));

record.put("checkout\_date", rs.getTimestamp("checkout\_date"));

record.put("status", rs.getString("status"));

record.put("remarks", rs.getString("remarks"));

history.add(record);

}

}

} catch (SQLException e) {

e.printStackTrace();

}

return history;

}

private Room extractRoomFromResultSet(ResultSet rs) throws SQLException {

return new Room(

rs.getInt("room\_id"),

rs.getString("room\_number"),

rs.getString("room\_type"),

rs.getBoolean("is\_occupied"),

rs.getInt("occupant\_id"),

rs.getString("status")

);

}

}

**File name: RoomRequestDAO.java**

package com.hostel.dao;

import com.hostel.model.RoomRequest;

import com.hostel.util.DatabaseUtil;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class RoomRequestDAO {

    public boolean createRequest(RoomRequest request) {

        String query = "INSERT INTO room\_requests (user\_id, preferred\_room\_type, status, remarks, preferred\_checkout\_date) VALUES (?, ?, ?, ?, ?)";

        try (Connection conn = DatabaseUtil.getConnection();

             PreparedStatement pstmt = conn.prepareStatement(query)) {

            pstmt.setInt(1, request.getUserId());

            pstmt.setString(2, request.getPreferredRoomType());

            pstmt.setString(3, "pending");

            pstmt.setString(4, request.getRemarks());

            if (request.getPreferredCheckoutDate() != null) {

                pstmt.setTimestamp(5, new Timestamp(request.getPreferredCheckoutDate().getTime()));

            } else {

                pstmt.setNull(5, Types.TIMESTAMP);

            }

            return pstmt.executeUpdate() > 0;

        } catch (SQLException e) {

            e.printStackTrace();

            return false;

        }

    }

    public List<RoomRequest> getAllRequests() {

        List<RoomRequest> requests = new ArrayList<>();

        String query = "SELECT \* FROM room\_requests ORDER BY request\_date DESC";

        try (Connection conn = DatabaseUtil.getConnection();

             Statement stmt = conn.createStatement();

             ResultSet rs = stmt.executeQuery(query)) {

            while (rs.next()) {

                requests.add(extractRequestFromResultSet(rs));

            }

        } catch (SQLException e) {

            e.printStackTrace();

        }

        return requests;

    }

    public List<RoomRequest> getRequestsByUser(int userId) {

        List<RoomRequest> requests = new ArrayList<>();

        String query = "SELECT \* FROM room\_requests WHERE user\_id = ? ORDER BY request\_date DESC";

        try (Connection conn = DatabaseUtil.getConnection();

             PreparedStatement pstmt = conn.prepareStatement(query)) {

            pstmt.setInt(1, userId);

            try (ResultSet rs = pstmt.executeQuery()) {

                while (rs.next()) {

                    requests.add(extractRequestFromResultSet(rs));

                }

            }

        } catch (SQLException e) {

            e.printStackTrace();

        }

        return requests;

    }

    public boolean updateRequestStatus(int requestId, String status, String remarks) {

        String query = "UPDATE room\_requests SET status = ?, remarks = ? WHERE request\_id = ?";

        try (Connection conn = DatabaseUtil.getConnection();

             PreparedStatement pstmt = conn.prepareStatement(query)) {

            pstmt.setString(1, status);

            pstmt.setString(2, remarks);

            pstmt.setInt(3, requestId);

            return pstmt.executeUpdate() > 0;

        } catch (SQLException e) {

            e.printStackTrace();

            return false;

        }

    }

    private RoomRequest extractRequestFromResultSet(ResultSet rs) throws SQLException {

        RoomRequest request = new RoomRequest(

            rs.getInt("request\_id"),

            rs.getInt("user\_id"),

            rs.getString("preferred\_room\_type"),

            rs.getString("status"),

            rs.getTimestamp("request\_date"),

            rs.getString("remarks")

        );

        Timestamp checkoutDate = rs.getTimestamp("preferred\_checkout\_date");

        if (checkoutDate != null) {

            request.setPreferredCheckoutDate(checkoutDate);

        }

        return request;

    }

}

**File name: UserDAO.java**

package com.hostel.dao;

import com.hostel.model.User;

import com.hostel.util.DatabaseUtil;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class UserDAO {

    public User authenticate(String username, String password) {

        String query = "SELECT \* FROM users WHERE username = ? AND password = ?";

        try (Connection conn = DatabaseUtil.getConnection();

             PreparedStatement pstmt = conn.prepareStatement(query)) {

            pstmt.setString(1, username);

            pstmt.setString(2, password);

            try (ResultSet rs = pstmt.executeQuery()) {

                if (rs.next()) {

                    return extractUserFromResultSet(rs);

                }

            }

        } catch (SQLException e) {

            e.printStackTrace();

        }

        return null;

    }

    public boolean isUsernameExists(String username) {

        String query = "SELECT COUNT(\*) FROM users WHERE username = ?";

        try (Connection conn = DatabaseUtil.getConnection();

             PreparedStatement pstmt = conn.prepareStatement(query)) {

            pstmt.setString(1, username);

            try (ResultSet rs = pstmt.executeQuery()) {

                if (rs.next()) {

                    return rs.getInt(1) > 0;

                }

            }

        } catch (SQLException e) {

            e.printStackTrace();

        }

        return false;

    }

    public boolean createUser(User user) {

        // First check if username already exists

        if (isUsernameExists(user.getUsername())) {

            return false;

        }

        String query = "INSERT INTO users (username, password, role, name, email) VALUES (?, ?, ?, ?, ?)";

        try (Connection conn = DatabaseUtil.getConnection();

             PreparedStatement pstmt = conn.prepareStatement(query)) {

            pstmt.setString(1, user.getUsername());

            pstmt.setString(2, user.getPassword());

            pstmt.setString(3, user.getRole());

            pstmt.setString(4, user.getName());

            pstmt.setString(5, user.getEmail());

            return pstmt.executeUpdate() > 0;

        } catch (SQLException e) {

            e.printStackTrace();

            return false;

        }

    }

    public boolean createAdmin(String username, String password, String name, String email) {

        // Check if username already exists

        if (isUsernameExists(username)) {

            return false;

        }

        String query = "INSERT INTO users (username, password, role, name, email) VALUES (?, ?, 'admin', ?, ?)";

        try (Connection conn = DatabaseUtil.getConnection();

             PreparedStatement pstmt = conn.prepareStatement(query)) {

            pstmt.setString(1, username);

            pstmt.setString(2, password);

            pstmt.setString(3, name);

            pstmt.setString(4, email);

            return pstmt.executeUpdate() > 0;

        } catch (SQLException e) {

            e.printStackTrace();

            return false;

        }

    }

    public List<User> getAllUsers() {

        List<User> users = new ArrayList<>();

        String query = "SELECT \* FROM users";

        try (Connection conn = DatabaseUtil.getConnection();

             Statement stmt = conn.createStatement();

             ResultSet rs = stmt.executeQuery(query)) {

            while (rs.next()) {

                users.add(extractUserFromResultSet(rs));

            }

        } catch (SQLException e) {

            e.printStackTrace();

        }

        return users;

    }

    private User extractUserFromResultSet(ResultSet rs) throws SQLException {

        return new User(

            rs.getInt("id"),

            rs.getString("username"),

            rs.getString("password"),

            rs.getString("role"),

            rs.getString("name"),

            rs.getString("email")

        );

    }

}

**File name: Room.java**

package com.hostel.model;

import java.util.Date;

public class Room {

    private int roomId;

    private String roomNumber;

    private String roomType;

    private boolean isOccupied;

    private int occupantId;

    private String status; // available, allocated, maintenance

    private Date allocationDate;

    private Date expectedCheckoutDate;

    private String occupantName; // To display in UI

    public Room() {}

    public Room(int roomId, String roomNumber, String roomType, boolean isOccupied, int occupantId, String status) {

        this.roomId = roomId;

        this.roomNumber = roomNumber;

        this.roomType = roomType;

        this.isOccupied = isOccupied;

        this.occupantId = occupantId;

        this.status = status;

    }

    // Getters and Setters

    public int getRoomId() { return roomId; }

    public void setRoomId(int roomId) { this.roomId = roomId; }

    public String getRoomNumber() { return roomNumber; }

    public void setRoomNumber(String roomNumber) { this.roomNumber = roomNumber; }

    public String getRoomType() { return roomType; }

    public void setRoomType(String roomType) { this.roomType = roomType; }

    public boolean isOccupied() { return isOccupied; }

    public void setOccupied(boolean occupied) { isOccupied = occupied; }

    public int getOccupantId() { return occupantId; }

    public void setOccupantId(int occupantId) { this.occupantId = occupantId; }

    public String getStatus() { return status; }

    public void setStatus(String status) { this.status = status; }

    public Date getAllocationDate() { return allocationDate; }

    public void setAllocationDate(Date allocationDate) { this.allocationDate = allocationDate; }

    public Date getExpectedCheckoutDate() { return expectedCheckoutDate; }

    public void setExpectedCheckoutDate(Date expectedCheckoutDate) { this.expectedCheckoutDate = expectedCheckoutDate; }

    public String getOccupantName() { return occupantName; }

    public void setOccupantName(String occupantName) { this.occupantName = occupantName; }

    @Override

    public String toString() {

        StringBuilder sb = new StringBuilder();

        sb.append("Room ").append(roomNumber);

        sb.append(" (").append(roomType).append(")");

        if (isOccupied && occupantName != null) {

            sb.append(" - Occupied by: ").append(occupantName);

        } else {

            sb.append(" - ").append(status);

        }

        return sb.toString();

    }

}

**File name: RoomRequest.java**

package com.hostel.model;

import java.util.Date;

public class RoomRequest {

    private int requestId;

    private int userId;

    private String preferredRoomType;

    private String status; // pending, approved, rejected

    private Date requestDate;

    private String remarks;

    private Date preferredCheckoutDate;

    public RoomRequest() {}

    public RoomRequest(int requestId, int userId, String preferredRoomType, String status, Date requestDate, String remarks) {

        this.requestId = requestId;

        this.userId = userId;

        this.preferredRoomType = preferredRoomType;

        this.status = status;

        this.requestDate = requestDate;

        this.remarks = remarks;

    }

    public RoomRequest(int requestId, int userId, String preferredRoomType, String status, Date requestDate, String remarks, Date preferredCheckoutDate) {

        this(requestId, userId, preferredRoomType, status, requestDate, remarks);

        this.preferredCheckoutDate = preferredCheckoutDate;

    }

    // Getters and Setters

    public int getRequestId() { return requestId; }

    public void setRequestId(int requestId) { this.requestId = requestId; }

    public int getUserId() { return userId; }

    public void setUserId(int userId) { this.userId = userId; }

    public String getPreferredRoomType() { return preferredRoomType; }

    public void setPreferredRoomType(String preferredRoomType) { this.preferredRoomType = preferredRoomType; }

    public String getStatus() { return status; }

    public void setStatus(String status) { this.status = status; }

    public Date getRequestDate() { return requestDate; }

    public void setRequestDate(Date requestDate) { this.requestDate = requestDate; }

    public String getRemarks() { return remarks; }

    public void setRemarks(String remarks) { this.remarks = remarks; }

    public Date getPreferredCheckoutDate() { return preferredCheckoutDate; }

    public void setPreferredCheckoutDate(Date preferredCheckoutDate) { this.preferredCheckoutDate = preferredCheckoutDate; }

}

**File name: User.java**

package com.hostel.model;

public class User {

private int id;

private String username;

private String password;

private String role;

private String name;

private String email;

public User() {}

public User(int id, String username, String password, String role, String name, String email) {

this.id = id;

this.username = username;

this.password = password;

this.role = role;

this.name = name;

this.email = email;

}

// Getters and Setters

public int getId() { return id; }

public void setId(int id) { this.id = id; }

public String getUsername() { return username; }

public void setUsername(String username) { this.username = username; }

public String getPassword() { return password; }

public void setPassword(String password) { this.password = password; }

public String getRole() { return role; }

public void setRole(String role) { this.role = role; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public String getEmail() { return email; }

public void setEmail(String email) { this.email = email; }

}

**File name: AdminDashboard.java**

package com.hostel.ui;

import com.hostel.dao.RoomDAO;

import com.hostel.dao.RoomRequestDAO;

import com.hostel.model.Room;

import com.hostel.model.RoomRequest;

import com.hostel.model.User;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.\*;

import java.text.SimpleDateFormat;

import java.util.\*;

import java.util.List;

import java.sql.Timestamp;

public class AdminDashboard extends JFrame {

    private final User admin;

    private final RoomDAO roomDAO;

    private final RoomRequestDAO requestDAO;

    private JTable roomsTable;

    private JTable requestsTable;

    private DefaultTableModel roomsModel;

    private DefaultTableModel requestsModel;

    private JTextField searchField;

    private SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm");

    public AdminDashboard(User admin) {

        this.admin = admin;

        this.roomDAO = new RoomDAO();

        this.requestDAO = new RoomRequestDAO();

        setTitle("Admin Dashboard - Hostel Room Allotment");

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setSize(1000, 600);

        setLocationRelativeTo(null);

        // Main panel

        JPanel mainPanel = new JPanel(new BorderLayout(10, 10));

        mainPanel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

        mainPanel.setBackground(Color.WHITE);

        // Top panel with welcome message and search

        JPanel topPanel = createTopPanel();

        mainPanel.add(topPanel, BorderLayout.NORTH);

        // Center panel with tables

        JSplitPane splitPane = createSplitPane();

        mainPanel.add(splitPane, BorderLayout.CENTER);

        // Bottom panel with actions

        JPanel bottomPanel = createBottomPanel();

        mainPanel.add(bottomPanel, BorderLayout.SOUTH);

        add(mainPanel);

        // Load initial data

        refreshTables();

    }

    private JPanel createTopPanel() {

        JPanel panel = new JPanel(new BorderLayout(10, 10));

        panel.setBackground(Color.WHITE);

        // Welcome message

        JLabel welcomeLabel = new JLabel("Welcome, " + admin.getName());

        welcomeLabel.setFont(new Font("Arial", Font.BOLD, 16));

        panel.add(welcomeLabel, BorderLayout.WEST);

        // Search panel

        JPanel searchPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));

        searchPanel.setBackground(Color.WHITE);

        searchField = new JTextField(20);

        JButton searchButton = new JButton("Search");

        searchButton.setBackground(new Color(37, 99, 235));

        searchButton.setForeground(Color.BLACK);

        searchPanel.add(new JLabel("Search:"));

        searchPanel.add(searchField);

        searchPanel.add(searchButton);

        panel.add(searchPanel, BorderLayout.EAST);

        // Add search functionality

        searchButton.addActionListener(e -> filterRooms());

        searchField.addActionListener(e -> filterRooms());

        return panel;

    }

    private JSplitPane createSplitPane() {

        // Rooms table

        roomsModel = new DefaultTableModel(

            new Object[]{"Room ID", "Room Number", "Type", "Status", "Occupant", "Allocation Date", "Expected Checkout"}, 0

        ) {

            @Override

            public boolean isCellEditable(int row, int column) {

                return false;

            }

        };

        roomsTable = new JTable(roomsModel);

        JScrollPane roomsScrollPane = new JScrollPane(roomsTable);

        roomsScrollPane.setBorder(BorderFactory.createTitledBorder("Rooms"));

        // Requests table - Added Preferred Checkout column

        requestsModel = new DefaultTableModel(

            new Object[]{"Request ID", "User ID", "Room Type", "Status", "Request Date", "Preferred Checkout", "Remarks"}, 0

        ) {

            @Override

            public boolean isCellEditable(int row, int column) {

                return false;

            }

        };

        requestsTable = new JTable(requestsModel);

        JScrollPane requestsScrollPane = new JScrollPane(requestsTable);

        requestsScrollPane.setBorder(BorderFactory.createTitledBorder("Room Requests"));

        // Split pane

        JSplitPane splitPane = new JSplitPane(JSplitPane.VERTICAL\_SPLIT, roomsScrollPane, requestsScrollPane);

        splitPane.setResizeWeight(0.5);

        // Add double-click listeners

        roomsTable.addMouseListener(new MouseAdapter() {

            public void mouseClicked(MouseEvent e) {

                if (e.getClickCount() == 2) {

                    showRoomHistory();

                }

            }

        });

        requestsTable.addMouseListener(new MouseAdapter() {

            public void mouseClicked(MouseEvent e) {

                if (e.getClickCount() == 2) {

                    handleRequest();

                }

            }

        });

        return splitPane;

    }

    private JPanel createBottomPanel() {

        JPanel panel = new JPanel(new FlowLayout(FlowLayout.LEFT, 10, 10));

        panel.setBackground(Color.WHITE);

        JButton addRoomButton = new JButton("Add Room");

        addRoomButton.setBackground(new Color(22, 163, 74));

        addRoomButton.setForeground(Color.BLACK);

        JButton logoutButton = new JButton("Logout");

        logoutButton.setBackground(new Color(220, 38, 38));

        logoutButton.setForeground(Color.BLACK);

        panel.add(addRoomButton);

        panel.add(logoutButton);

        addRoomButton.addActionListener(e -> addNewRoom());

        logoutButton.addActionListener(e -> logout());

        return panel;

    }

    private void refreshTables() {

        // Clear existing data

        roomsModel.setRowCount(0);

        requestsModel.setRowCount(0);

        // Load rooms with allocation info

        List<Room> rooms = roomDAO.getRoomsWithAllocationInfo();

        for (Room room : rooms) {

            roomsModel.addRow(new Object[]{

                room.getRoomId(),

                room.getRoomNumber(),

                room.getRoomType(),

                room.getStatus(),

                room.getOccupantName() != null ? room.getOccupantName() : "-",

                room.getAllocationDate() != null ? dateFormat.format(room.getAllocationDate()) : "-",

                room.getExpectedCheckoutDate() != null ? dateFormat.format(room.getExpectedCheckoutDate()) : "-"

            });

        }

        // Load requests

        List<RoomRequest> requests = requestDAO.getAllRequests();

        for (RoomRequest request : requests) {

            String checkoutDate = request.getPreferredCheckoutDate() != null ?

                dateFormat.format(request.getPreferredCheckoutDate()) : "-";

            requestsModel.addRow(new Object[]{

                request.getRequestId(),

                request.getUserId(),

                request.getPreferredRoomType(),

                request.getStatus(),

                dateFormat.format(request.getRequestDate()),

                checkoutDate,

                request.getRemarks()

            });

        }

    }

    private void filterRooms() {

        String searchText = searchField.getText().toLowerCase();

        roomsModel.setRowCount(0);

        List<Room> rooms = roomDAO.getRoomsWithAllocationInfo();

        for (Room room : rooms) {

            if (room.getRoomNumber().toLowerCase().contains(searchText) ||

                room.getRoomType().toLowerCase().contains(searchText) ||

                room.getStatus().toLowerCase().contains(searchText)) {

                roomsModel.addRow(new Object[]{

                    room.getRoomId(),

                    room.getRoomNumber(),

                    room.getRoomType(),

                    room.getStatus(),

                    room.getOccupantName() != null ? room.getOccupantName() : "-",

                    room.getAllocationDate() != null ? dateFormat.format(room.getAllocationDate()) : "-",

                    room.getExpectedCheckoutDate() != null ? dateFormat.format(room.getExpectedCheckoutDate()) : "-"

                });

            }

        }

    }

    private void addNewRoom() {

        JTextField roomNumberField = new JTextField();

        JComboBox<String> roomTypeCombo = new JComboBox<>(new String[]{"Single", "Double", "Triple"});

        JPanel panel = new JPanel(new GridLayout(0, 1));

        panel.add(new JLabel("Room Number:"));

        panel.add(roomNumberField);

        panel.add(new JLabel("Room Type:"));

        panel.add(roomTypeCombo);

        int result = JOptionPane.showConfirmDialog(this, panel, "Add New Room",

                JOptionPane.OK\_CANCEL\_OPTION, JOptionPane.PLAIN\_MESSAGE);

        if (result == JOptionPane.OK\_OPTION) {

            String roomNumber = roomNumberField.getText();

            String roomType = (String) roomTypeCombo.getSelectedItem();

            if (!roomNumber.isEmpty()) {

                Room newRoom = new Room(0, roomNumber, roomType, false, 0, "available");

                if (roomDAO.addRoom(newRoom)) {

                    refreshTables();

                    JOptionPane.showMessageDialog(this, "Room added successfully!");

                } else {

                    JOptionPane.showMessageDialog(this, "Failed to add room", "Error", JOptionPane.ERROR\_MESSAGE);

                }

            }

        }

    }

    private void editRoom() {

        int selectedRow = roomsTable.getSelectedRow();

        if (selectedRow == -1) return;

        int roomId = (int) roomsTable.getValueAt(selectedRow, 0);

        String currentStatus = (String) roomsTable.getValueAt(selectedRow, 3);

        boolean isOccupied = !currentStatus.equals("available");

        JPanel panel = new JPanel(new GridLayout(0, 1));

        String[] statuses = isOccupied ?

            new String[]{"occupied", "maintenance"} :

            new String[]{"available", "maintenance"};

        JComboBox<String> statusCombo = new JComboBox<>(statuses);

        statusCombo.setSelectedItem(currentStatus);

        panel.add(new JLabel("Status:"));

        panel.add(statusCombo);

        // Add deallocate option if room is occupied

        if (isOccupied) {

            JButton deallocateButton = new JButton("Deallocate Room");

            deallocateButton.addActionListener(e -> {

                String remarks = JOptionPane.showInputDialog(this, "Enter remarks for deallocation:");

                if (remarks != null) {

                    if (roomDAO.deallocateRoom(roomId, remarks)) {

                        refreshTables();

                        JOptionPane.showMessageDialog(this, "Room deallocated successfully!");

                        Window win = SwingUtilities.getWindowAncestor(panel);

                        if (win != null) win.dispose();

                    } else {

                        JOptionPane.showMessageDialog(this, "Failed to deallocate room", "Error", JOptionPane.ERROR\_MESSAGE);

                    }

                }

            });

            panel.add(deallocateButton);

        }

        int result = JOptionPane.showConfirmDialog(this, panel, "Edit Room",

                JOptionPane.OK\_CANCEL\_OPTION, JOptionPane.PLAIN\_MESSAGE);

        if (result == JOptionPane.OK\_OPTION) {

            String newStatus = (String) statusCombo.getSelectedItem();

            if (roomDAO.updateRoomStatus(roomId, newStatus, newStatus.equals("occupied"), null)) {

                refreshTables();

                JOptionPane.showMessageDialog(this, "Room updated successfully!");

            } else {

                JOptionPane.showMessageDialog(this, "Failed to update room", "Error", JOptionPane.ERROR\_MESSAGE);

            }

        }

    }

    private void showRoomHistory() {

        int selectedRow = roomsTable.getSelectedRow();

        if (selectedRow == -1) return;

        int roomId = (int) roomsTable.getValueAt(selectedRow, 0);

        List<Map<String, Object>> history = roomDAO.getAllocationHistory(roomId);

        // Create history dialog

        JDialog historyDialog = new JDialog(this, "Room Allocation History", true);

        historyDialog.setSize(600, 400);

        historyDialog.setLocationRelativeTo(this);

        // Create history table

        String[] columns = {"Occupant", "Allocation Date", "Checkout Date", "Status", "Remarks"};

        DefaultTableModel historyModel = new DefaultTableModel(columns, 0) {

            @Override

            public boolean isCellEditable(int row, int column) {

                return false;

            }

        };

        JTable historyTable = new JTable(historyModel);

        JScrollPane scrollPane = new JScrollPane(historyTable);

        // Add history data

        for (Map<String, Object> record : history) {

            historyModel.addRow(new Object[]{

                record.get("user\_name"),

                formatDate((Timestamp) record.get("allocation\_date")),

                formatDate((Timestamp) record.get("checkout\_date")),

                record.get("status"),

                record.get("remarks")

            });

        }

        historyDialog.add(scrollPane);

        historyDialog.setVisible(true);

    }

    private void handleRequest() {

        int selectedRow = requestsTable.getSelectedRow();

        if (selectedRow == -1) return;

        int requestId = (int) requestsTable.getValueAt(selectedRow, 0);

        int userId = (int) requestsTable.getValueAt(selectedRow, 1);

        String currentStatus = (String) requestsTable.getValueAt(selectedRow, 3);

        String preferredCheckout = (String) requestsTable.getValueAt(selectedRow, 5);

        if (!currentStatus.equals("pending")) {

            JOptionPane.showMessageDialog(this, "This request has already been processed.");

            return;

        }

        // Show available rooms of requested type

        String preferredRoomType = (String) requestsTable.getValueAt(selectedRow, 2);

        List<Room> availableRooms = roomDAO.getAvailableRooms();

        List<Room> matchingRooms = new ArrayList<>();

        for (Room room : availableRooms) {

            if (room.getRoomType().equals(preferredRoomType)) {

                matchingRooms.add(room);

            }

        }

        if (matchingRooms.isEmpty()) {

            int choice = JOptionPane.showConfirmDialog(this,

                    "No matching rooms available. Would you like to reject the request?",

                    "No Rooms Available",

                    JOptionPane.YES\_NO\_OPTION);

            if (choice == JOptionPane.YES\_OPTION) {

                String remarks = JOptionPane.showInputDialog(this, "Enter remarks for rejection:");

                if (remarks != null) {

                    requestDAO.updateRequestStatus(requestId, "rejected", remarks);

                    refreshTables();

                }

            }

            return;

        }

        // Show room selection dialog

        Room selectedRoom = (Room) JOptionPane.showInputDialog(this,

                "Select a room to allocate:",

                "Allocate Room",

                JOptionPane.QUESTION\_MESSAGE,

                null,

                matchingRooms.toArray(),

                matchingRooms.get(0));

        if (selectedRoom != null) {

            // Use preferred checkout date from request

            try {

                Date checkoutDate;

                if (!preferredCheckout.equals("-")) {

                    checkoutDate = dateFormat.parse(preferredCheckout);

                } else {

                    // If no preferred date, ask for one

                    String checkoutDateStr = JOptionPane.showInputDialog(this,

                            "Enter expected checkout date (yyyy-MM-dd):",

                            new SimpleDateFormat("yyyy-MM-dd").format(new Date()));

                    checkoutDate = new SimpleDateFormat("yyyy-MM-dd").parse(checkoutDateStr);

                }

                if (roomDAO.allocateRoom(selectedRoom.getRoomId(), userId, checkoutDate)) {

                    requestDAO.updateRequestStatus(requestId, "approved", "Room " + selectedRoom.getRoomNumber() + " allocated");

                    refreshTables();

                    JOptionPane.showMessageDialog(this, "Room allocated successfully!");

                } else {

                    JOptionPane.showMessageDialog(this, "Failed to allocate room", "Error", JOptionPane.ERROR\_MESSAGE);

                }

            } catch (Exception e) {

                JOptionPane.showMessageDialog(this, "Invalid date format", "Error", JOptionPane.ERROR\_MESSAGE);

            }

        }

    }

    private String formatDate(Timestamp timestamp) {

        return timestamp != null ? dateFormat.format(timestamp) : "-";

    }

    private void logout() {

        new LoginWindow().setVisible(true);

        this.dispose();

    }

}

**File name: LoginWindow.java**

package com.hostel.ui;

import com.hostel.dao.UserDAO;

import com.hostel.model.User;

import javax.swing.\*;

import java.awt.\*;

import javax.swing.border.EmptyBorder;

// javac -cp "lib/\*;bin" -d bin src/main/java/com/hostel/ui/LoginWindow.java

// java -cp "lib/\*;bin" com.hostel.ui.LoginWindow

public class LoginWindow extends JFrame {

    private final UserDAO userDAO;

    private final JTextField usernameField;

    private final JPasswordField passwordField;

    public LoginWindow() {

        userDAO = new UserDAO();

        setTitle("Hostel Room Allotment - Login");

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setSize(400, 300);

        setLocationRelativeTo(null);

        // Main panel with padding

        JPanel mainPanel = new JPanel();

        mainPanel.setLayout(new BorderLayout());

        mainPanel.setBorder(new EmptyBorder(20, 20, 20, 20));

        mainPanel.setBackground(Color.WHITE);

        // Form panel

        JPanel formPanel = new JPanel();

        formPanel.setLayout(new GridBagLayout());

        formPanel.setBackground(Color.WHITE);

        GridBagConstraints gbc = new GridBagConstraints();

        gbc.fill = GridBagConstraints.HORIZONTAL;

        gbc.insets = new Insets(5, 5, 5, 5);

        // Title

        JLabel titleLabel = new JLabel("Login", SwingConstants.CENTER);

        titleLabel.setFont(new Font("Arial", Font.BOLD, 24));

        gbc.gridx = 0;

        gbc.gridy = 0;

        gbc.gridwidth = 2;

        formPanel.add(titleLabel, gbc);

        // Username

        JLabel usernameLabel = new JLabel("Useme:");

        gbc.gridx = 0;

        gbc.gridy = 1;

        gbc.gridwidth = 1;

        formPanel.add(usernameLabel, gbc);

        usernameField = new JTextField(20);

        gbc.gridx = 1;

        formPanel.add(usernameField, gbc);

        // Password

        JLabel passwordLabel = new JLabel("Password:");

        gbc.gridx = 0;

        gbc.gridy = 2;

        formPanel.add(passwordLabel, gbc);

        passwordField = new JPasswordField(20);

        gbc.gridx = 1;

        formPanel.add(passwordField, gbc);

        // Login button

        JButton loginButton = new JButton("Login");

        loginButton.setBackground(new Color(0, 0, 0));

        loginButton.setForeground(Color.BLACK);

        loginButton.setFocusPainted(true);

        gbc.gridx = 0;

        gbc.gridy = 3;

        gbc.gridwidth = 2;

        gbc.fill = GridBagConstraints.NONE;

        formPanel.add(loginButton, gbc);

        // Register button

        JButton registerButton = new JButton("Register");

        registerButton.setBackground(new Color(0,0,0));

        registerButton.setForeground(Color.BLACK);

        registerButton.setFocusPainted(true);

        gbc.gridy = 4;

        formPanel.add(registerButton, gbc);

        mainPanel.add(formPanel, BorderLayout.CENTER);

        add(mainPanel);

        // Add action listeners

        loginButton.addActionListener(e -> handleLogin());

        registerButton.addActionListener(e -> openRegistrationWindow());

        // Make enter key trigger login

        getRootPane().setDefaultButton(loginButton);

    }

    private void handleLogin() {

        String username = usernameField.getText();

        String password = new String(passwordField.getPassword());

        if (username.isEmpty() || password.isEmpty()) {

            JOptionPane.showMessageDialog(this,

                    "Please enter both username and password",

                    "Login Error",

                    JOptionPane.ERROR\_MESSAGE);

            return;

        }

        User user = userDAO.authenticate(username, password);

        if (user != null) {

            if (user.getRole().equals("admin")) {

                new AdminDashboard(user).setVisible(true);

            } else {

                new StudentDashboard(user).setVisible(true);

            }

            this.dispose();

        } else {

            JOptionPane.showMessageDialog(this,

                    "Invalid username or password",

                    "Login Error",

                    JOptionPane.ERROR\_MESSAGE);

        }

    }

    private void openRegistrationWindow() {

        new RegistrationWindow().setVisible(true);

        this.dispose();

    }

    public static void main(String[] args) {

        try {

            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

        } catch (Exception e) {

            e.printStackTrace();

        }

        SwingUtilities.invokeLater(() -> {

            new LoginWindow().setVisible(true);

        });

    }

}

**File name: RegistrationWindow.java**

package com.hostel.ui;

import com.hostel.dao.UserDAO;

import com.hostel.model.User;

// javac -cp "lib/\*;bin" -d bin src/main/java/com/hostel/ui/RegistrationWindow.java

import javax.swing.\*;

import java.awt.\*;

import javax.swing.border.EmptyBorder;

public class RegistrationWindow extends JFrame {

    private final UserDAO userDAO;

    private final JTextField usernameField;

    private final JPasswordField passwordField;

    private final JPasswordField confirmPasswordField;

    private final JTextField nameField;

    private final JTextField emailField;

    public RegistrationWindow() {

        userDAO = new UserDAO();

        setTitle("Register - Hostel Room Allotment");

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setSize(400, 500);

        setLocationRelativeTo(null);

        // Main panel with padding

        JPanel mainPanel = new JPanel();

        mainPanel.setLayout(new BorderLayout());

        mainPanel.setBorder(new EmptyBorder(20, 20, 20, 20));

        mainPanel.setBackground(Color.WHITE);

        // Form panel

        JPanel formPanel = new JPanel();

        formPanel.setLayout(new GridBagLayout());

        formPanel.setBackground(Color.WHITE);

        GridBagConstraints gbc = new GridBagConstraints();

        gbc.fill = GridBagConstraints.HORIZONTAL;

        gbc.insets = new Insets(5, 5, 5, 5);

        // Title

        JLabel titleLabel = new JLabel("Registration", SwingConstants.CENTER);

        titleLabel.setFont(new Font("Arial", Font.BOLD, 24));

        gbc.gridx = 0;

        gbc.gridy = 0;

        gbc.gridwidth = 2;

        formPanel.add(titleLabel, gbc);

        // Name

        JLabel nameLabel = new JLabel("Full Name:");

        gbc.gridx = 0;

        gbc.gridy = 1;

        gbc.gridwidth = 1;

        formPanel.add(nameLabel, gbc);

        nameField = new JTextField(20);

        gbc.gridx = 1;

        formPanel.add(nameField, gbc);

        // Username

        JLabel usernameLabel = new JLabel("Username:");

        gbc.gridx = 0;

        gbc.gridy = 2;

        formPanel.add(usernameLabel, gbc);

        usernameField = new JTextField(20);

        gbc.gridx = 1;

        formPanel.add(usernameField, gbc);

        // Email

        JLabel emailLabel = new JLabel("Email:");

        gbc.gridx = 0;

        gbc.gridy = 3;

        formPanel.add(emailLabel, gbc);

        emailField = new JTextField(20);

        gbc.gridx = 1;

        formPanel.add(emailField, gbc);

        // Password

        JLabel passwordLabel = new JLabel("Password:");

        gbc.gridx = 0;

        gbc.gridy = 4;

        formPanel.add(passwordLabel, gbc);

        passwordField = new JPasswordField(20);

        gbc.gridx = 1;

        formPanel.add(passwordField, gbc);

        // Confirm Password

        JLabel confirmPasswordLabel = new JLabel("Confirm Password:");

        gbc.gridx = 0;

        gbc.gridy = 5;

        formPanel.add(confirmPasswordLabel, gbc);

        confirmPasswordField = new JPasswordField(20);

        gbc.gridx = 1;

        formPanel.add(confirmPasswordField, gbc);

        // Register button

        JButton registerButton = new JButton("Register");

        registerButton.setBackground(new Color(0,0,0)); // bg-green-600

        registerButton.setForeground(Color.BLACK);

        registerButton.setFocusPainted(false);

        gbc.gridx = 0;

        gbc.gridy = 6;

        gbc.gridwidth = 2;

        gbc.fill = GridBagConstraints.NONE;

        formPanel.add(registerButton, gbc);

        // Back to Login button

        JButton backButton = new JButton("Back to Login");

        backButton.setBackground(new Color(0,0,0)); // bg-blue-600

        backButton.setForeground(Color.BLACK);

        backButton.setFocusPainted(false);

        gbc.gridy = 7;

        formPanel.add(backButton, gbc);

        mainPanel.add(formPanel, BorderLayout.CENTER);

        add(mainPanel);

        // Add action listeners

        registerButton.addActionListener(e -> handleRegistration());

        backButton.addActionListener(e -> backToLogin());

        // Make enter key trigger registration

        getRootPane().setDefaultButton(registerButton);

    }

    private void handleRegistration() {

        String name = nameField.getText();

        String username = usernameField.getText();

        String email = emailField.getText();

        String password = new String(passwordField.getPassword());

        String confirmPassword = new String(confirmPasswordField.getPassword());

        // Validation

        if (name.isEmpty() || username.isEmpty() || email.isEmpty() || password.isEmpty()) {

            JOptionPane.showMessageDialog(this,

                    "Please fill in all fields",

                    "Registration Error",

                    JOptionPane.ERROR\_MESSAGE);

            return;

        }

        if (!password.equals(confirmPassword)) {

            JOptionPane.showMessageDialog(this,

                    "Passwords do not match",

                    "Registration Error",

                    JOptionPane.ERROR\_MESSAGE);

            return;

        }

        if (!email.matches("^[A-Za-z0-9+\_.-]+@(.+)$")) {

            JOptionPane.showMessageDialog(this,

                    "Please enter a valid email address",

                    "Registration Error",

                    JOptionPane.ERROR\_MESSAGE);

            return;

        }

        User newUser = new User(0, username, password, "student", name, email);

        if (userDAO.createUser(newUser)) {

            JOptionPane.showMessageDialog(this,

                    "Registration successful! Please login.",

                    "Success",

                    JOptionPane.INFORMATION\_MESSAGE);

            backToLogin();

        } else {

            JOptionPane.showMessageDialog(this,

                    "Registration failed. Username may already exist.",

                    "Registration Error",

                    JOptionPane.ERROR\_MESSAGE);

        }

    }

    private void backToLogin() {

        new LoginWindow().setVisible(true);

        this.dispose();

    }

}

**File name: StudentDashboard.java**

package com.hostel.ui;

import com.hostel.dao.RoomDAO;

import com.hostel.dao.RoomRequestDAO;

import com.hostel.model.Room;

import com.hostel.model.RoomRequest;

import com.hostel.model.User;

//  javac -cp "lib/\*;bin" -d bin src/main/java/com/hostel/ui/StudentDashboard.java

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.util.List;

import java.text.SimpleDateFormat;

import java.text.ParseException;

import java.util.Calendar;

import java.util.Date;

public class StudentDashboard extends JFrame {

    private final User student;

    private final RoomDAO roomDAO;

    private final RoomRequestDAO requestDAO;

    private JTable roomsTable;

    private JTable requestsTable;

    private DefaultTableModel roomsModel;

    private DefaultTableModel requestsModel;

    public StudentDashboard(User student) {

        this.student = student;

        this.roomDAO = new RoomDAO();

        this.requestDAO = new RoomRequestDAO();

        setTitle("Student Dashboard - Hostel Room Allotment");

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setSize(900, 600);

        setLocationRelativeTo(null);

        // Main panel

        JPanel mainPanel = new JPanel(new BorderLayout(10, 10));

        mainPanel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

        mainPanel.setBackground(Color.WHITE);

        // Top panel with welcome message

        JPanel topPanel = createTopPanel();

        mainPanel.add(topPanel, BorderLayout.NORTH);

        // Center panel with tables

        JSplitPane splitPane = createSplitPane();

        mainPanel.add(splitPane, BorderLayout.CENTER);

        // Bottom panel with actions

        JPanel bottomPanel = createBottomPanel();

        mainPanel.add(bottomPanel, BorderLayout.SOUTH);

        add(mainPanel);

        // Load initial data

        refreshTables();

    }

    private JPanel createTopPanel() {

        JPanel panel = new JPanel(new BorderLayout(10, 10));

        panel.setBackground(Color.WHITE);

        // Welcome message

        JLabel welcomeLabel = new JLabel("Welcome, " + student.getName());

        welcomeLabel.setFont(new Font("Arial", Font.BOLD, 16));

        panel.add(welcomeLabel, BorderLayout.WEST);

        return panel;

    }

    private JSplitPane createSplitPane() {

        // Available Rooms table

        roomsModel = new DefaultTableModel(

            new Object[]{"Room Number", "Type", "Status"}, 0

        ) {

            @Override

            public boolean isCellEditable(int row, int column) {

                return false;

            }

        };

        roomsTable = new JTable(roomsModel);

        JScrollPane roomsScrollPane = new JScrollPane(roomsTable);

        roomsScrollPane.setBorder(BorderFactory.createTitledBorder("Available Rooms"));

        // My Requests table

        requestsModel = new DefaultTableModel(

            new Object[]{"Request ID", "Room Type", "Status", "Date", "Remarks"}, 0

        ) {

            @Override

            public boolean isCellEditable(int row, int column) {

                return false;

            }

        };

        requestsTable = new JTable(requestsModel);

        JScrollPane requestsScrollPane = new JScrollPane(requestsTable);

        requestsScrollPane.setBorder(BorderFactory.createTitledBorder("My Room Requests"));

        // Split pane

        JSplitPane splitPane = new JSplitPane(JSplitPane.VERTICAL\_SPLIT, roomsScrollPane, requestsScrollPane);

        splitPane.setResizeWeight(0.5);

        return splitPane;

    }

    private JPanel createBottomPanel() {

        JPanel panel = new JPanel(new FlowLayout(FlowLayout.LEFT, 10, 10));

        panel.setBackground(Color.WHITE);

        JButton requestRoomButton = new JButton("Request Room");

        requestRoomButton.setBackground(new Color(37, 99, 235)); // bg-blue-600

        requestRoomButton.setForeground(Color.BLACK);

        requestRoomButton.setFocusPainted(false);

        JButton logoutButton = new JButton("Logout");

        logoutButton.setBackground(new Color(220, 38, 38)); // bg-red-600

        logoutButton.setForeground(Color.BLACK);

        logoutButton.setFocusPainted(false);

        panel.add(requestRoomButton);

        panel.add(logoutButton);

        requestRoomButton.addActionListener(e -> showRequestDialog());

        logoutButton.addActionListener(e -> logout());

        return panel;

    }

    private void refreshTables() {

        // Clear existing data

        roomsModel.setRowCount(0);

        requestsModel.setRowCount(0);

        // Load available rooms

        List<Room> rooms = roomDAO.getAvailableRooms();

        for (Room room : rooms) {

            roomsModel.addRow(new Object[]{

                room.getRoomNumber(),

                room.getRoomType(),

                room.getStatus()

            });

        }

        // Load student's requests

        List<RoomRequest> requests = requestDAO.getRequestsByUser(student.getId());

        for (RoomRequest request : requests) {

            requestsModel.addRow(new Object[]{

                request.getRequestId(),

                request.getPreferredRoomType(),

                request.getStatus(),

                request.getRequestDate(),

                request.getRemarks()

            });

        }

    }

    private void showRequestDialog() {

        JComboBox<String> roomTypeCombo = new JComboBox<>(new String[]{"Single", "Double", "Triple"});

        JTextArea remarksArea = new JTextArea(3, 20);

        remarksArea.setLineWrap(true);

        remarksArea.setWrapStyleWord(true);

        JScrollPane remarksScroll = new JScrollPane(remarksArea);

        // Create date field with default value (3 months from now)

        Calendar cal = Calendar.getInstance();

        cal.add(Calendar.MONTH, 3);

        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");

        JTextField checkoutDateField = new JTextField(dateFormat.format(cal.getTime()));

        JPanel panel = new JPanel(new GridLayout(0, 1, 5, 5));

        panel.add(new JLabel("Preferred Room Type:"));

        panel.add(roomTypeCombo);

        panel.add(new JLabel("Preferred Checkout Date (yyyy-MM-dd):"));

        panel.add(checkoutDateField);

        panel.add(new JLabel("Additional Remarks:"));

        panel.add(remarksScroll);

        int result = JOptionPane.showConfirmDialog(this, panel, "Request Room",

                JOptionPane.OK\_CANCEL\_OPTION, JOptionPane.PLAIN\_MESSAGE);

        if (result == JOptionPane.OK\_OPTION) {

            String roomType = (String) roomTypeCombo.getSelectedItem();

            String remarks = remarksArea.getText();

            try {

                Date checkoutDate = dateFormat.parse(checkoutDateField.getText());

                // Validate checkout date is in the future

                if (checkoutDate.before(new Date())) {

                    JOptionPane.showMessageDialog(this,

                        "Checkout date must be in the future",

                        "Invalid Date",

                        JOptionPane.ERROR\_MESSAGE);

                    return;

                }

                RoomRequest request = new RoomRequest(0, student.getId(), roomType, "pending", null, remarks);

                request.setPreferredCheckoutDate(checkoutDate);

                if (requestDAO.createRequest(request)) {

                    refreshTables();

                    JOptionPane.showMessageDialog(this, "Room request submitted successfully!");

                } else {

                    JOptionPane.showMessageDialog(this, "Failed to submit request", "Error", JOptionPane.ERROR\_MESSAGE);

                }

            } catch (ParseException e) {

                JOptionPane.showMessageDialog(this,

                    "Invalid date format. Please use yyyy-MM-dd",

                    "Error",

                    JOptionPane.ERROR\_MESSAGE);

            }

        }

    }

    private void logout() {

        new LoginWindow().setVisible(true);

        this.dispose();

    }

}

**File name: DatabaseUtil.java**

package com.hostel.util;

import com.hostel.dao.UserDAO;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.sql.Statement;

public class DatabaseUtil {

    private static final String DB\_FILE = "hostel.db";

    private static final String DB\_URL = "jdbc:sqlite:" + DB\_FILE;

    static {

        // Initialize database on class load

        try {

            Class.forName("org.sqlite.JDBC");

            initializeDatabase();

        } catch (ClassNotFoundException e) {

            System.err.println("SQLite JDBC driver not found: " + e.getMessage());

            System.exit(1);

        }

    }

    public static Connection getConnection() throws SQLException {

        return DriverManager.getConnection(DB\_URL);

    }

    public static void initializeDatabase() {

        try (Connection conn = getConnection();

             Statement stmt = conn.createStatement()) {

            // Create Users table

            stmt.execute("""

                CREATE TABLE IF NOT EXISTS users (

                    id INTEGER PRIMARY KEY AUTOINCREMENT,

                    username TEXT UNIQUE NOT NULL,

                    password TEXT NOT NULL,

                    role TEXT NOT NULL,

                    name TEXT NOT NULL,

                    email TEXT

                )

            """);

            // Create Rooms table with additional fields

            stmt.execute("""

                CREATE TABLE IF NOT EXISTS rooms (

                    room\_id INTEGER PRIMARY KEY AUTOINCREMENT,

                    room\_number TEXT UNIQUE NOT NULL,

                    room\_type TEXT NOT NULL,

                    is\_occupied BOOLEAN DEFAULT FALSE,

                    occupant\_id INTEGER,

                    status TEXT DEFAULT 'available',

                    allocation\_date TIMESTAMP,

                    expected\_checkout\_date TIMESTAMP,

                    FOREIGN KEY (occupant\_id) REFERENCES users(id)

                )

            """);

            // Create RoomRequests table with preferred\_checkout\_date

            stmt.execute("""

                CREATE TABLE IF NOT EXISTS room\_requests (

                    request\_id INTEGER PRIMARY KEY AUTOINCREMENT,

                    user\_id INTEGER NOT NULL,

                    preferred\_room\_type TEXT NOT NULL,

                    status TEXT DEFAULT 'pending',

                    request\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

                    remarks TEXT,

                    preferred\_checkout\_date TIMESTAMP,

                    FOREIGN KEY (user\_id) REFERENCES users(id)

                )

            """);

            // Create RoomAllocationHistory table

            stmt.execute("""

                CREATE TABLE IF NOT EXISTS room\_allocation\_history (

                    allocation\_id INTEGER PRIMARY KEY AUTOINCREMENT,

                    room\_id INTEGER NOT NULL,

                    user\_id INTEGER NOT NULL,

                    allocation\_date TIMESTAMP NOT NULL,

                    checkout\_date TIMESTAMP,

                    status TEXT NOT NULL,

                    remarks TEXT,

                    FOREIGN KEY (room\_id) REFERENCES rooms(room\_id),

                    FOREIGN KEY (user\_id) REFERENCES users(id)

                )

            """);

            // Create default admin user if not exists

            UserDAO userDAO = new UserDAO();

            userDAO.createAdmin("admin", "admin123", "System Administrator", "admin@hostel.com");

        } catch (SQLException e) {

            e.printStackTrace();

            System.err.println("Error initializing database: " + e.getMessage());

        }

    }

    public static void closeConnection(Connection conn) {

        if (conn != null) {

            try {

                conn.close();

            } catch (SQLException e) {

                e.printStackTrace();

            }

        }

    }

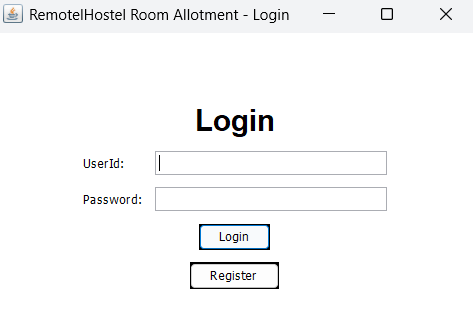
}

## Output:

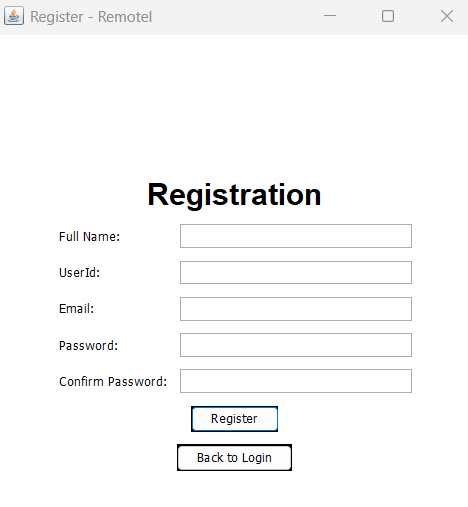
* Start the project by opening path and using command

“java -cp "lib/\*;bin" com.hostel.ui.LoginWindow”

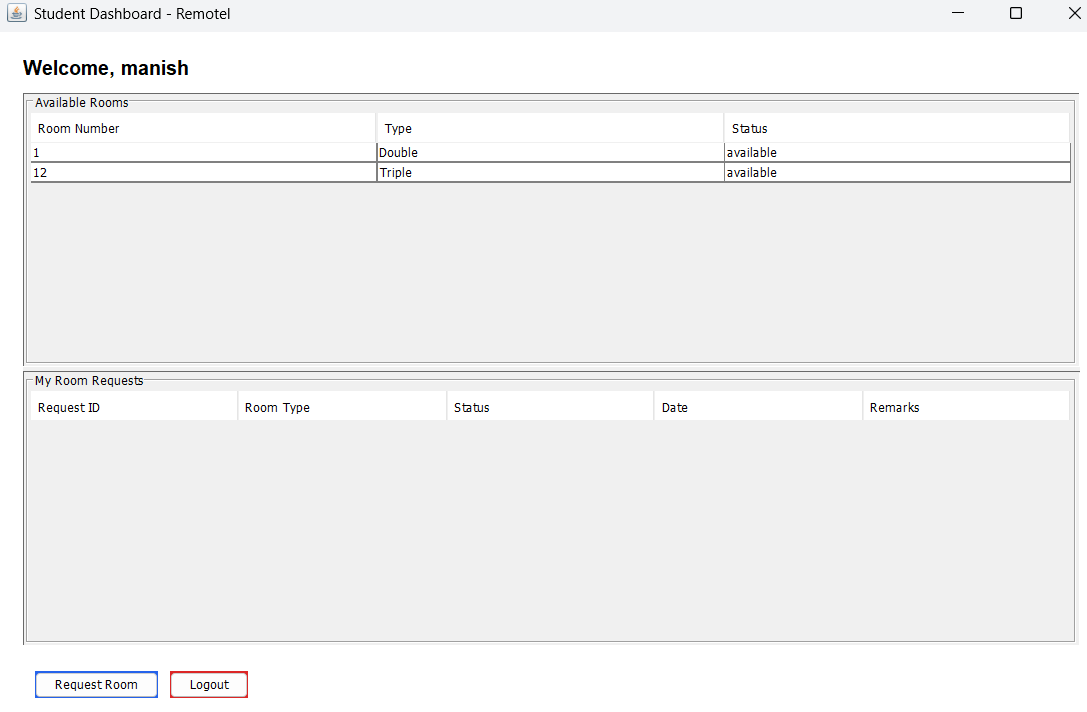
* LoginWindow



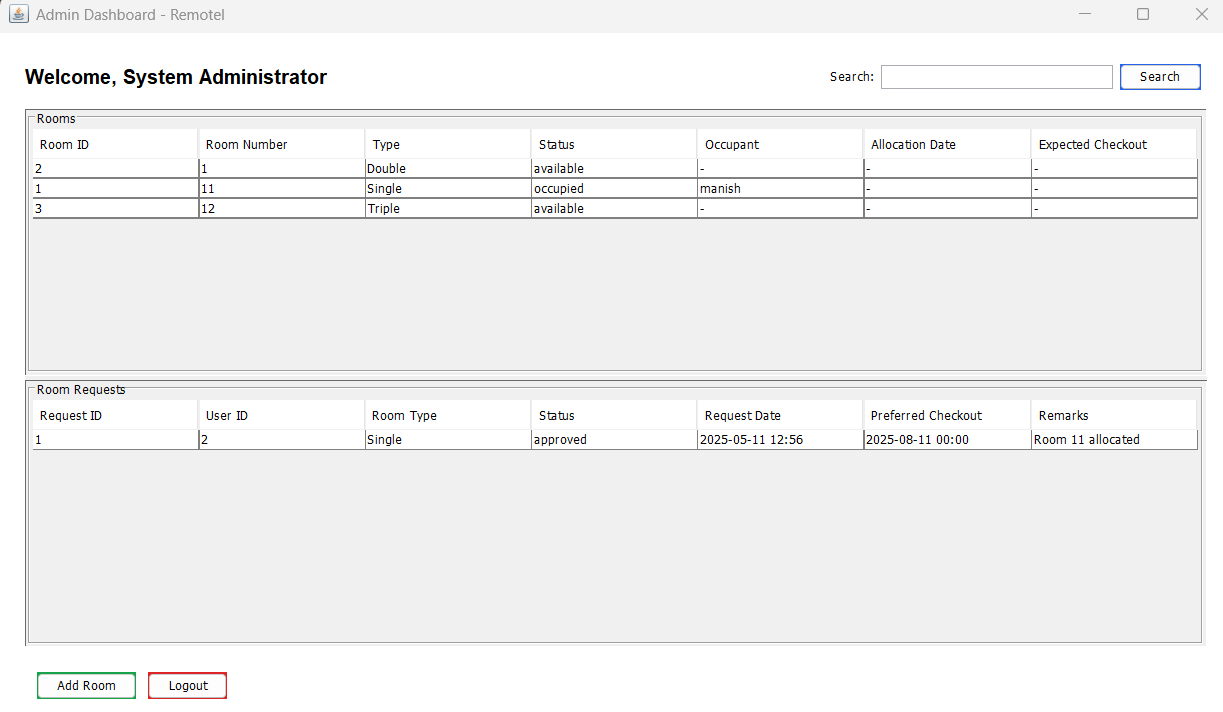
* If You are a new user then register yourself else input your credentials
* If you clicked register Button then RegistrationWindow opens and Now complete your registration and then login page occurs.



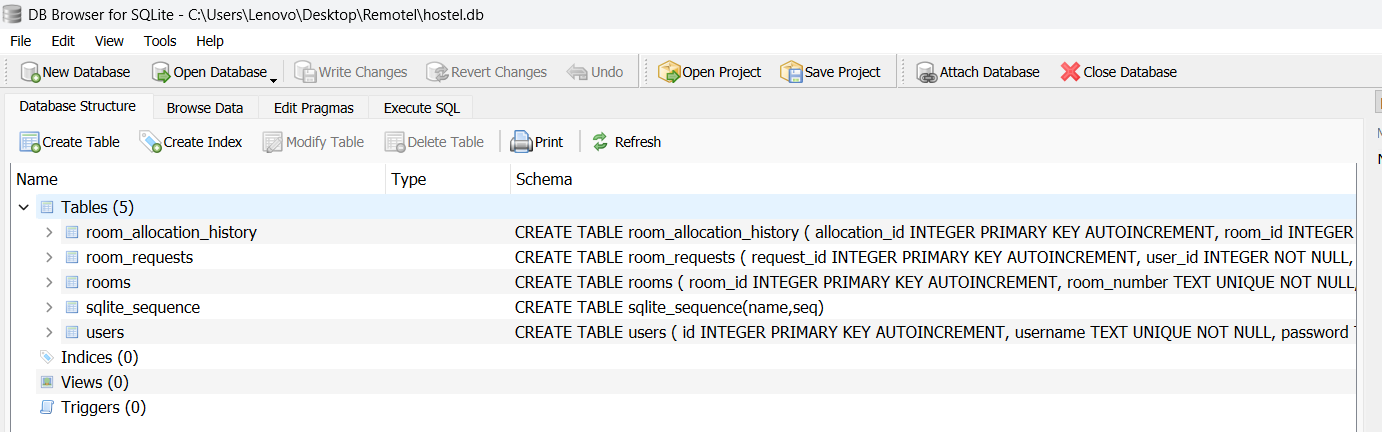
* Now as I have an account after putting my credentials StudentDashboard page pop outs and now as a student now we can request the available rooms.



* if an admin wants to manage room availability and requests of students by putting “admin” and “admin123” as userid and password now AdminDashboard page will be available for you.



* Database visuals



# **CONCLUSION**

The *Remotel* project successfully addresses the common challenges faced in manual hostel room management by automating the process through a user-friendly desktop application. By utilizing **Java Swing** for the GUI, **SQLite** for lightweight database storage, and the **DAO and MVC design patterns** for modularity and maintainability, the system ensures a robust and efficient solution for both users and administrators.

The application simplifies tasks such as **room request submission**, **user registration**, and **administrative approval workflows**, thereby minimizing human errors and administrative overhead. The layered architecture ensures ease of scalability and future enhancements, such as adding network capabilities or transitioning to a mobile or web-based version.

Overall, *Remotel* demonstrates a practical and well-structured implementation of software development principles, successfully achieving its objectives of improving hostel room allocation through automation.

# **REFERENCE**

1. Java Documentation – Oracle Official Java SE Documentation  
   <https://docs.oracle.com/javase/>
2. SQLite Documentation – SQLite Official Site  
   https://www.sqlite.org/docs.html
3. Java Swing Tutorial – Java Swing GUI Programming Tutorials  
   <https://docs.oracle.com/javase/tutorial/uiswing/>
4. DAO Pattern in Java – Data Access Object Design Pattern  
   https://www.geeksforgeeks.org/data-access-object-pattern/
5. MVC Architecture – Model View Controller Design Pattern Explained  
   https://www.javatpoint.com/mvc-in-java
6. SQLite JDBC Integration – JDBC for SQLite

# **ASSESSMENT**

**Internal:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SL NO** | **RUBRICS** | **FULL MARK** | **MARKS OBTAINED** | **REMARKS** |
| 1 | Understanding the relevance, scope and dimension of the project | 10 |  |  |
| 2 | Methodology | 10 |  |  |
| 3 | Quality of Analysis and Results | 10 |  |  |
| 4 | Interpretations and Conclusions | 10 |  |  |
| 5 | Report | 10 |  |  |
|  | **Total** | **50** |  |  |

**Date: Signature of the Faculty**

# **COURSE OUTCOME (COs) ATTAINMENT**

* **Expected Course Outcomes (COs):**

**(Refer to COs Statement in the Syllabus)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

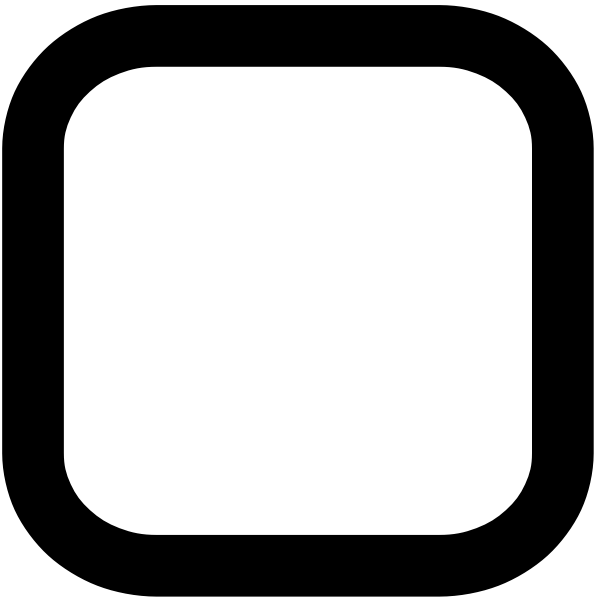
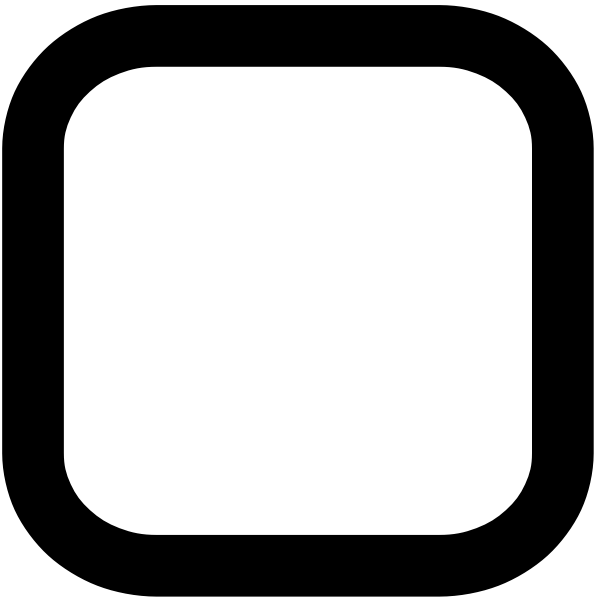
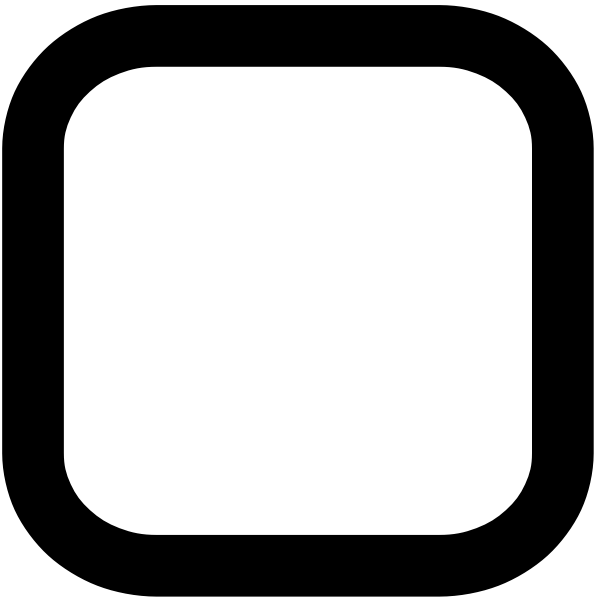
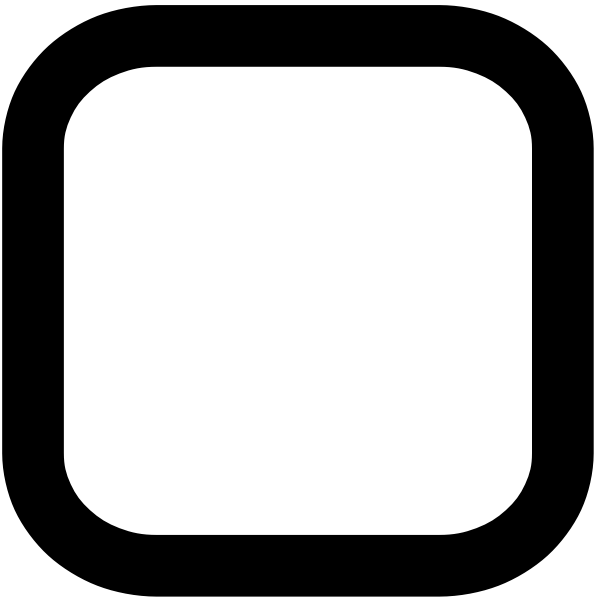
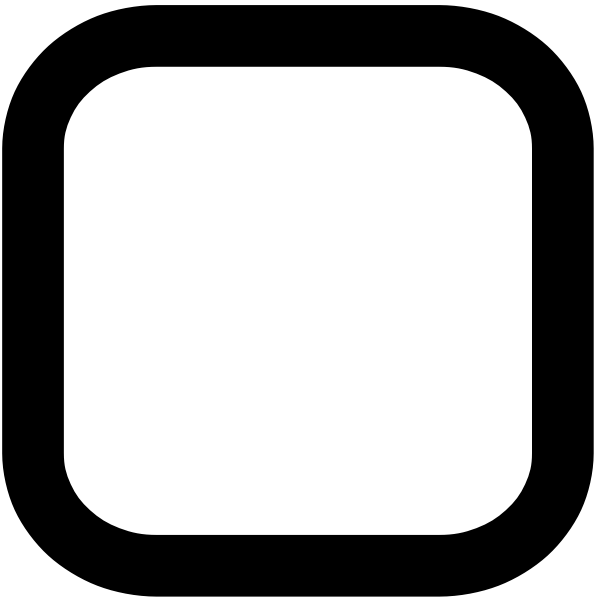
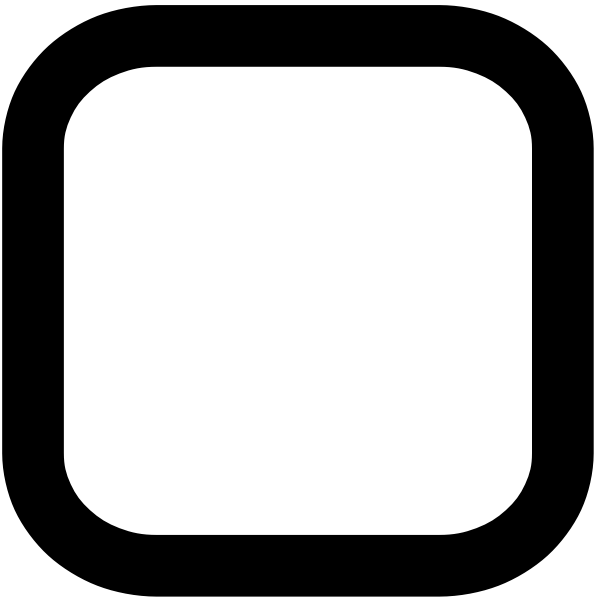
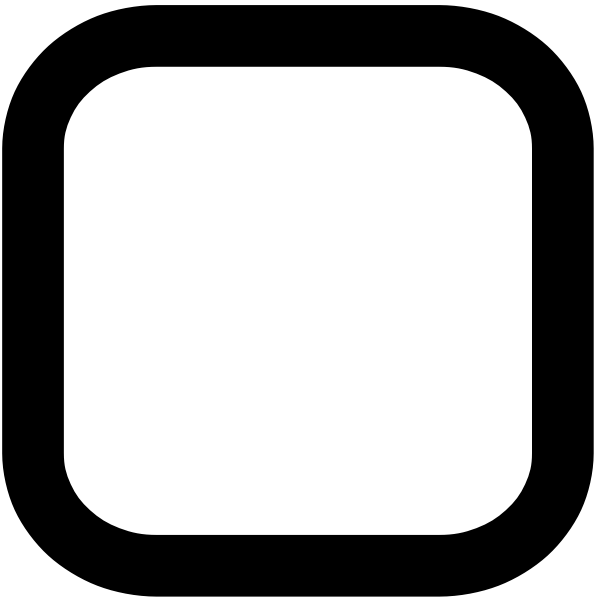
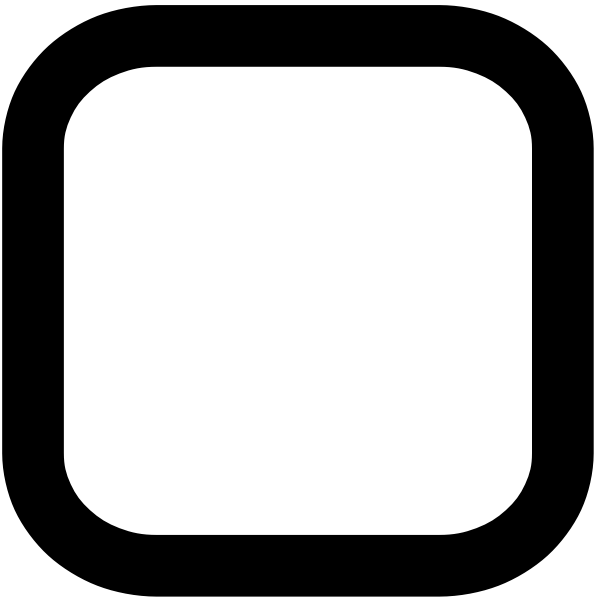
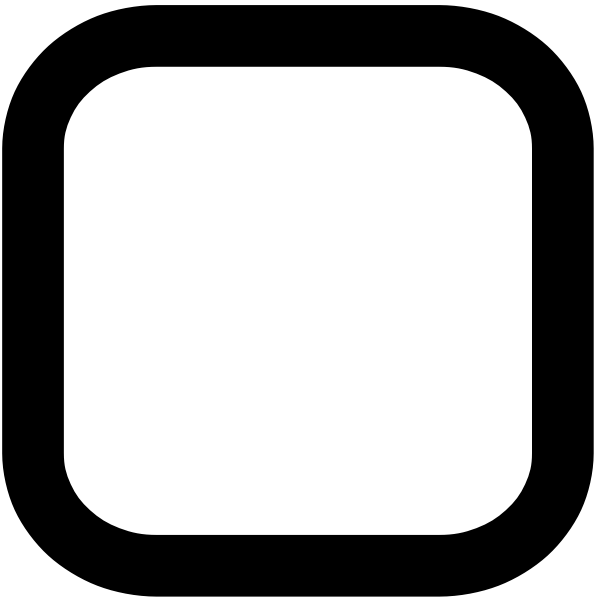
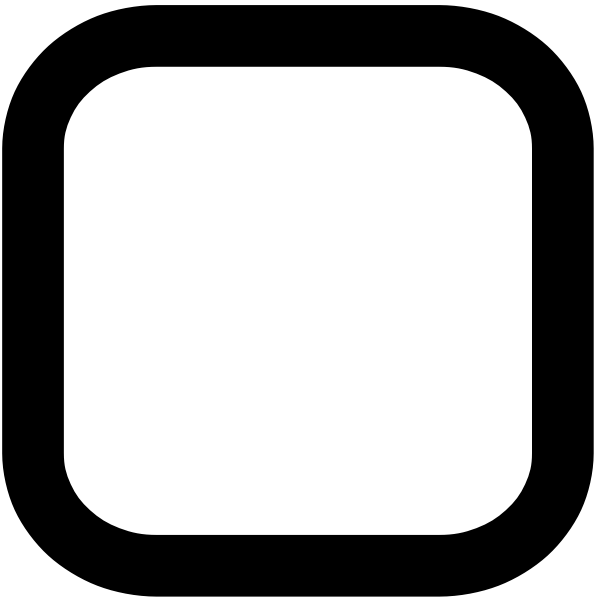
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

* **Course Outcome Attained:**

**How would you rate your learning of the subject based on the specified COs?**



**1 2 3 4 5 6 7 8 9 10**

**LOW HIGH**

* **Learning Gap (if any):**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* **Books / Manuals Referred:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date: Signature of the Student**

* **Suggestions / Recommendations:**

**(By the Course Faculty)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date: Signature of the Faculty**