**COMSATS University Islamabad (CUI)**

Logo, company name

Description automatically generated

**Event Management System(For University)**  
 ***By***

**Aiman Khan CIIT/SP22-BSE-003/ISB**

**Ayesha Siddiqa CIIT/SP22-BSE-009/ISB**

**Ume Rubab CIIT/SP22-BSE-050/ISB**

***Submitted to*SIR MUZAFFAR IQBAL**

Contents

[Project Description 3](#_Toc137036515)

[Team Member Work Division 3](#_Toc137036516)

[Class Diagram: 3](#_Toc137036517)

[1. EMS Class: 3](#_Toc137036518)

[2. AdminUserLogin Class: 3](#_Toc137036519)

[3. Admin Class: 3](#_Toc137036520)

[4. Room Class: 3](#_Toc137036521)

[5. User Class: 3](#_Toc137036522)

[6. Student Class: 4](#_Toc137036523)

[7. Faculty Class: 4](#_Toc137036524)

[8. Booking Class: 4](#_Toc137036525)

# Project Description

The Event Management System is an innovative web application designed to address the challenges faced by Comsats University in allocating classrooms and lecture halls efficiently. The current manual process is time-consuming, error-prone, and lacks transparency, resulting in conflicts and suboptimal utilization of rooms. This system aims to streamline the room scheduling process, reduce administrative workloads, and enhance the academic experience for students and faculty.

The system includes features such as user registration and authentication, allowing different user roles for faculty, staff, and students.

For administrators, the system provides comprehensive room management capabilities. They can manage room information such as room types, capacity, and availability. This includes tasks like adding and deleting rooms, updating room information, and ensuring accuracy and timeliness of data.

Room reservations are a key aspect of the system. Student and Faculty can create, update, and cancel reservations while ensuring they are made in accordance with room availability, thus avoiding scheduling conflicts. The system provides an intuitive and user-friendly interface to facilitate efficient time and resource management.

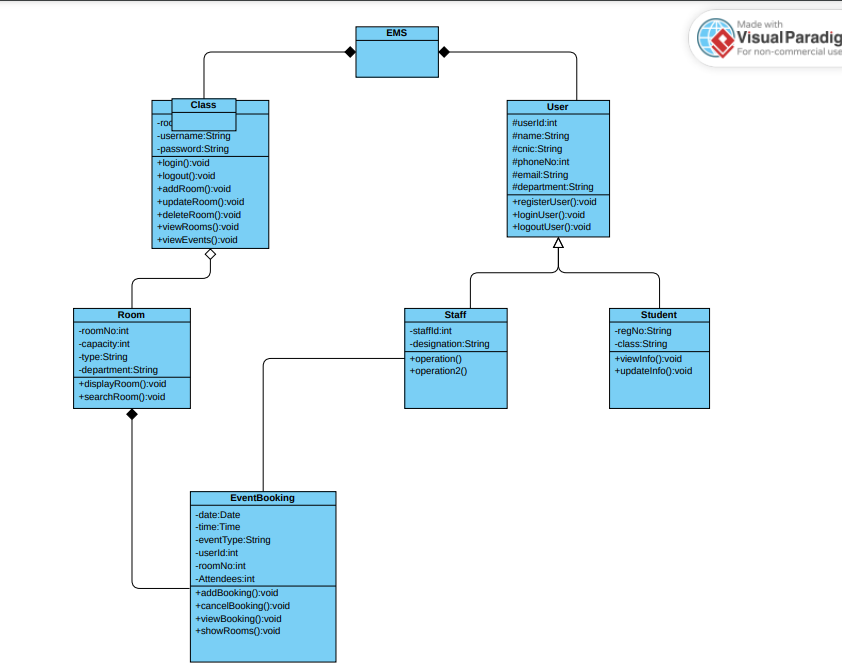
Overall, the Event Management System offers a user-friendly, efficient, and flexible solution for scheduling and managing rooms and events within the university. It aims to enhance productivity, reduce conflicts, and optimize resource utilization for a better academic experience.

# Team Member Work Division

Table. Team Member Work Division for Event Management System

|  |  |  |
| --- | --- | --- |
| **Student Name** | **Student Registration Number** | **Work** |
| Aiman Khan | SP22-BSE-003 | Admin class , EventBooking class , AdminUserLogin class |
| Ayesha Siddiqa | SP22-BSE-009 | User class, Student class, Faculty class, AdminUserLogin class |
| Ume Rubab | SP22-BSE-050 | Room class, EMS class, Admin class, AdminUserLogin class |

# Class Diagram:



# EMS Class:

package com.mycompany.ems;

import javax.swing.\*;

public class EMS extends JFrame {

public static void main(String[] args)

{

System.out.println("-------------------- EVENT MANAGEMENT SYSTEM -----------------");

System.out.println("-----------------------------Admin:-------------------------");

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

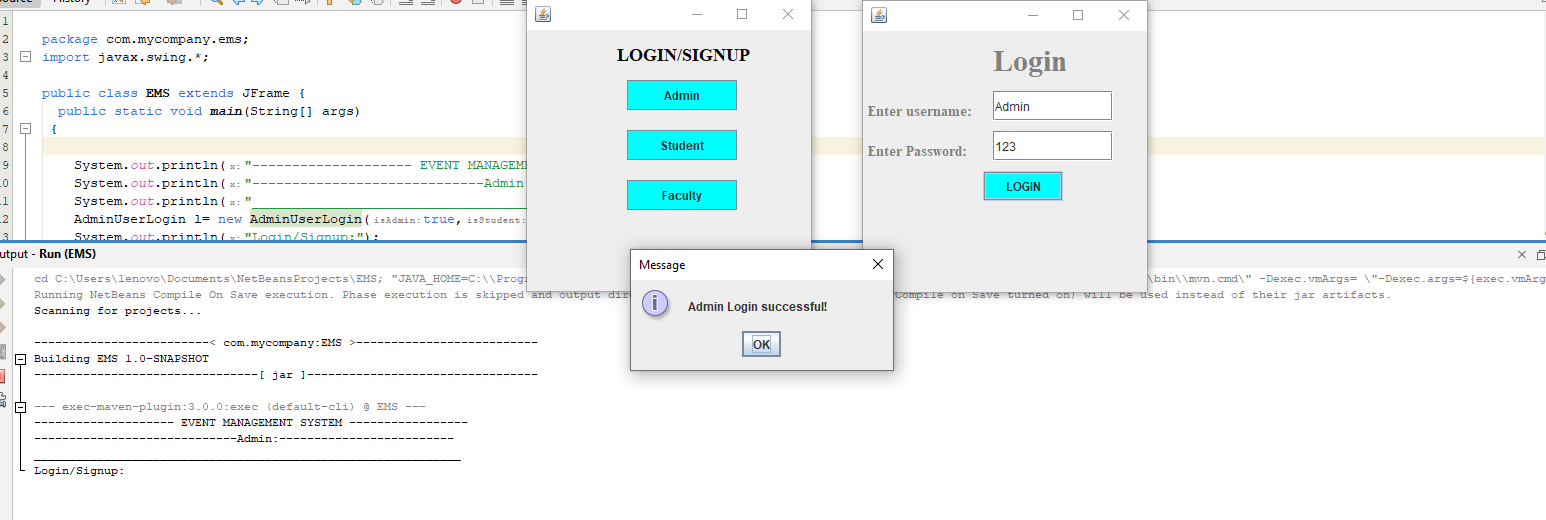
AdminUserLogin l= new AdminUserLogin(true,true,true);

System.out.println("Login/Signup:");

l.callGui();

}

}



# AdminUserLogin Class:

import javax.swing.\*;

import java.awt.\*;

import java.util.ArrayList;

import java.awt.event.\*;

import java.io.\*;

public class AdminUserLogin extends JFrame implements Serializable {

static ArrayList<Student> students=new ArrayList<>();

JTextField t1, t2;

JLabel l1, name,pass;

JButton loginButton, signUpButton;

boolean isStudent;

boolean isFaculty;

AdminUserLogin(boolean isAdmin,boolean isStudent, boolean isFaculty) {

setLayout(null);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

l1 = new JLabel("Login");

name= new JLabel("Enter username:");

pass = new JLabel("Enter Password:");

l1.setFont(new Font("Times New Roman", Font.BOLD, 30));

l1.setForeground(Color.GRAY);

l1.setBounds(130, 10, 300, 40);

name.setFont(new Font("Times New Roman", Font.BOLD, 14));

name.setForeground(Color.GRAY);

name.setBounds(5, 60, 300, 40);

pass.setFont(new Font("Times New Roman", Font.BOLD, 14));

pass.setForeground(Color.GRAY);

pass.setBounds(5, 100, 300, 40);

add(l1);

add(name);

add(pass);

t1 = new JTextField(60);

t2 = new JTextField(60);

t1.setBounds(130, 60, 120, 30);

t2.setBounds(130, 100, 120, 30);

loginButton = new JButton("LOGIN");

loginButton.setBounds(120, 140, 80, 30);

loginButton.setBackground(Color.CYAN);

add(t1);

add(t2);

add(loginButton);

if (isStudent) {

signUpButton = new JButton("SIGN UP");

signUpButton.setBounds(120, 180, 80, 30);

signUpButton.setBackground(Color.CYAN);

add(signUpButton);

signUpButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent ae) {

StudentRegistrationWindow s = new StudentRegistrationWindow();

s.setSize(300, 250);

s.setVisible(true);

}

});

}

if (isFaculty) {

signUpButton = new JButton("SIGN UP");

signUpButton.setBounds(120, 180, 80, 30);

signUpButton.setBackground(Color.CYAN);

add(signUpButton);

signUpButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent ae) {

FacultyRegistrationWindow f = new FacultyRegistrationWindow();

f.setSize(300, 250);

f.setVisible(true);

}

});

}

loginButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent ae) {

boolean loginSuccessful = false;

if (isAdmin) {

if (t1.getText().equals("Admin") && t2.getText().equals("123")) {

JOptionPane.showMessageDialog(null, "Admin Login successful!");

loginSuccessful = true;

} else {

JOptionPane.showMessageDialog(null, "Admin Login not successful!");

}

} else {

String username = t1.getText();

String password = t2.getText();

if(isStudent){

if (studentCheckUserLogin(username, password,true)) {

JOptionPane.showMessageDialog(null, "User Login successful!");

loginSuccessful = true;

}

else if(isFaculty){

if (facultycheckUserLogin(username, password,isFaculty)) {

JOptionPane.showMessageDialog(null, "User Login successful!");

loginSuccessful = true;

}

}

else {

JOptionPane.showMessageDialog(null, "Invalid username or password. Please try again.");

}

}}

if(isAdmin){

if (loginSuccessful) {

Admin a=new Admin();

a.AdminMenu();

setVisible(false);

dispose();

}}

if(isStudent){

if (loginSuccessful) {

Student s=new Student();

s.StudentMenu();

setVisible(false);//to close login window

dispose();

}

}

if(isFaculty){

if (loginSuccessful) {

Faculty f=new Faculty();

f.FacultyMenu();

setVisible(false);//to close login window

dispose();

}

}

}

});

}

public boolean studentCheckUserLogin(String username, String password, boolean isStudent) {

try {

ObjectInputStream ois = new ObjectInputStream(new FileInputStream("Studentrecord.txt"));

while (true) {

try {

Student student = (Student) ois.readObject();

students.add(student);

} catch (EOFException e) {

break;

}

}

for (Student student : students) {

if( student.username!=null && student.username.equalsIgnoreCase(username) && student.password!=null&& student.password.equals(password)) {

return true;

}

}

} catch (IOException | ClassNotFoundException e) {

System.out.println(e);

}

return false;

}

public boolean facultycheckUserLogin(String username, String password, boolean isFaculty) {

try {

ObjectInputStream ois = new ObjectInputStream(new FileInputStream("Facultyrecord.txt"));

ArrayList<Faculty> faculties = new ArrayList<>();

while (true) {

try {

Faculty faculty = ( Faculty) ois.readObject();

faculties.add(faculty);

} catch (EOFException e) {

break;

}

}

for (Faculty faculty : faculties) {

if (faculty.username!=null&&faculty.username.equalsIgnoreCase(username) &&faculty.password!=null&& faculty.password.equals(password)) {

return true;

}

}

} catch (IOException | ClassNotFoundException e) {

System.out.println(e);

}

return false;

}

public static class FirstWindow extends JFrame {

JButton adminButton, studentButton,facultyButton;

JLabel loginLabel;

FirstWindow() {

setLayout(null);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

adminButton = new JButton("Admin");

studentButton = new JButton("Student");

facultyButton = new JButton("Faculty");

loginLabel = new JLabel("LOGIN/SIGNUP");

loginLabel.setFont(new Font("Times New Roman", Font.BOLD, 18));

loginLabel.setForeground(Color.BLACK);

studentButton.setBackground(Color.CYAN);

facultyButton.setBackground(Color.CYAN);

adminButton.setBackground(Color.CYAN);

loginLabel.setBounds(90, 10, 200, 30);

adminButton.setBounds(100, 50, 110, 30);

studentButton.setBounds(100, 100, 110, 30);

facultyButton.setBounds(100, 150, 110, 30);

add(adminButton);

add(studentButton);

add(facultyButton);

add(loginLabel);

adminButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent ae) {

AdminUserLogin adminLoginWindow = new AdminUserLogin(true,false,false);

adminLoginWindow.setSize(300, 300);

adminLoginWindow.setVisible(true);

}

});

facultyButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent ae) {

//public static boolean isFaculty=true;

AdminUserLogin facultyLoginWindow = new AdminUserLogin(false,false,true);

facultyLoginWindow.setSize(300, 300);

facultyLoginWindow.setVisible(true);

}

});

studentButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent ae) {

// public static boolean isStudent=true;

AdminUserLogin studentLoginWindow = new AdminUserLogin(false,true,false);

studentLoginWindow.setSize(300, 300);

studentLoginWindow.setVisible(true);

}

});

}

}

public static class StudentRegistrationWindow extends JFrame implements Serializable{

JLabel registerLabel, usernameLabel, passwordLabel,confirmPassLabel;

JTextField usernameField,passwordField,confirmPassField;

JButton submitButton;

StudentRegistrationWindow() {

setLayout(null);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

registerLabel = new JLabel("STUDENT SIGNUP");

registerLabel.setFont(new Font("Times New Roman", Font.BOLD, 16));

registerLabel.setForeground(Color.BLACK);

registerLabel.setBounds(90, 10, 200, 30);

add(registerLabel);

usernameLabel = new JLabel("Username:");

usernameLabel.setBounds(20, 50, 80, 30);

add(usernameLabel);

passwordLabel = new JLabel("Password:");

passwordLabel .setBounds(20, 90, 80, 30);

add(passwordLabel );

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

confirmPassLabel .setBounds(20, 130, 80, 30);

add(confirmPassLabel );

usernameField = new JTextField(60);

usernameField.setBounds(100, 50, 120, 30);

add(usernameField);

passwordField=new JTextField();

passwordField.setBounds(100, 90, 120, 30);

add(passwordField);

confirmPassField= new JTextField(60);

confirmPassField.setBounds(100, 130, 120, 30);

add(confirmPassField);

submitButton = new JButton("SUBMIT");

submitButton.setBounds(120, 170, 80, 30);

submitButton.setBackground(Color.CYAN);

add(submitButton);

submitButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent ae) {

Student student = new Student();

student.username = usernameField.getText();

student.password = passwordField.getText();

students.add(student);

try (ObjectOutputStream objectOutputStream = new ObjectOutputStream(new FileOutputStream("Studentrecord.txt"))) {

for (Student s: students) {

objectOutputStream.writeObject(s);

}

}

catch (IOException e) {

JOptionPane.showMessageDialog(null, "Student Signup not successful!");

}

JOptionPane.showMessageDialog(null, "Student Signup successful!");

Student s=new Student();

s.StudentMenu();

usernameField.setText("");

passwordField.setText("");

confirmPassField.setText("");

}

});

}}

public static class FacultyRegistrationWindow extends JFrame implements Serializable {

JLabel registerLabel, usernameLabel, DesignationLabel,passwordLabel, confirmPassLabel;

JTextField usernameField,passwordField,confirmPassField;

JButton submitButton;

FacultyRegistrationWindow() {

setLayout(null);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

registerLabel = new JLabel("FACULTY REGISTRATION");

registerLabel.setFont(new Font("Times New Roman", Font.BOLD, 16));

registerLabel.setForeground(Color.BLACK);

registerLabel.setBounds(90, 10, 200, 30);

add(registerLabel);

usernameLabel = new JLabel("Username:");

usernameLabel.setBounds(20, 50, 80, 30);

add(usernameLabel);

passwordLabel = new JLabel("Password:");

passwordLabel.setBounds(20, 90, 80, 30);

add(passwordLabel);

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

confirmPassLabel .setBounds(20, 130, 80, 30);

add(confirmPassLabel );

usernameField = new JTextField(60);

usernameField.setBounds(100, 50, 120, 30);

add(usernameField);

passwordField=new JTextField(60);

passwordField.setBounds(100, 90, 120, 30);

add(passwordField);

confirmPassField= new JTextField(60);

confirmPassField.setBounds(100, 130, 120, 30);

add(confirmPassField);

submitButton = new JButton("SUBMIT");

submitButton.setBounds(120, 170, 80, 30);

submitButton.setBackground(Color.CYAN);

add(submitButton);

submitButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent ae) {

String Fusername = usernameField.getText();

String Fpassword = passwordField.getText();

Faculty faculty = new Faculty();

faculty.username = Fusername;

faculty.password = Fpassword;

try {

ObjectOutputStream objectOutputStream = new ObjectOutputStream(new FileOutputStream("Studentrecord.txt",true));

objectOutputStream.writeObject(faculty);

objectOutputStream.close();

}

catch (IOException e)

{

System.out.println("Error writing to file: " + e.getMessage());

}

JOptionPane.showMessageDialog(null, "Faculty Signup successful!");

Faculty f=new Faculty();

f.FacultyMenu();

usernameField.setText("");

passwordField.setText("");

confirmPassField.setText("");

}

});

}

}

public void callGui() {

FirstWindow firstWindow = new FirstWindow();

firstWindow.setSize(300, 300);

firstWindow.setVisible(true);

}}

# Admin Class:

import java.io.\*;

import java.util.Scanner;

import java.util.ArrayList;

import java.util.logging.Level;

import java.util.logging.Logger;

public class Admin

{

private ArrayList<Room> rooms;

private String username;

private String password;

private Scanner input;

public Admin() {

username = "Admin";

password = "123";

rooms = new ArrayList<Room>();

input = new Scanner(System.in);

}

public void setRooms(ArrayList<Room> rooms) {

this.rooms = rooms;

}

public ArrayList<Room> getRooms() {

return rooms;

}

public String getUsername() {

return username;

}

public String getPassword() {

return password;

}

public void AdminMenu(){

while (true) {

System.out.println("\n\t \* ADMIN MENU \*");

System.out.println("1. Add Room");

System.out.println("2. View Rooms");

System.out.println("3. Update Room");

System.out.println("4. Delete Room");

System.out.println("5. Exit");

System.out.print("Enter your choice: ");

int choice = input.nextInt();

input.nextLine(); // Consume the newline character

switch (choice) {

case 1:

addRoom();

break;

case 2:

viewRoom();

break;

case 3:

updateRoom();

break;

case 4:

deleteRoom();

break;

case 5:

System.out.println("Exiting program...");

System.exit(0);

default:

System.out.println("Invalid choice. Please try again.");

break;

}

}

}

public void addRoom()

{

System.out.println("\t!!..Enter Room Details..!!");

System.out.print("Enter Room No: ");

String rollNo = checkRoomNO();

int rNo = Integer.parseInt(rollNo);

input.nextLine(); // Consume the newline character

boolean matchFound = true;

while (matchFound) {

matchFound = false;

for (Room room : rooms) {

if (room.getRoomNo() == rNo) {

System.out.println("This room number is already taken...");

rollNo = checkRoomNO();

rNo = Integer.parseInt(rollNo);

input.nextLine(); // Consume the newline character

matchFound = true;

break;

}

}

}

System.out.print("Enter Room Capacity: ");

int capacity = input.nextInt();

input.nextLine(); // Consume the newline character

System.out.print("Enter Room Type (Classroom, lab, seminarHall): ");

String type = checkRoomType();

System.out.print("Enter department of new Room: ");

String department = checkDepartment();

Room newRoom = new Room();

newRoom.setRoomNo(rNo);

newRoom.setCapacity(capacity);

newRoom.setType(type);

newRoom.setDepartment(department);

rooms.add(newRoom);

try {

ObjectOutputStream objectOutputStream = new ObjectOutputStream(new FileOutputStream("Roomrecord.txt"));

for (Room room : rooms) {

objectOutputStream.writeObject(room);

}

objectOutputStream.close();

} catch (IOException e) {

System.out.println("Error writing to file: " + e.getMessage());

}

System.out.println("Room added Successfully!");

}

public void deleteRoom() {

System.out.print("Enter Room No: ");

String rNo = checkRoomNO();

int roomNo = Integer.parseInt(rNo);

input.nextLine(); // Consume the newline character

try {

ObjectInputStream objectInputStream = new ObjectInputStream(new FileInputStream("Roomrecord.txt"));

try {

while (true) {

Room room = (Room) objectInputStream.readObject();

rooms.add(room);

}

} catch (EOFException e) {

// Reached end of file

}

objectInputStream.close();

boolean roomFound = false;

for (Room room : rooms) {

if (room.getRoomNo() == roomNo) {

rooms.remove(room);

roomFound = true;

break;

}

}

if (roomFound) {

try {

ObjectOutputStream objectOutputStream = new ObjectOutputStream(new FileOutputStream("Roomrecord.txt"));

for (Room room : rooms) {

objectOutputStream.writeObject(room);

}

objectOutputStream.close();

} catch (IOException e) {

System.out.println("Error writing to file: " + e.getMessage());

}

System.out.println("Room deleted successfully!!");

} else {

System.out.println("Sorry...Room not Found!");

}

} catch (IOException | ClassNotFoundException e) {

System.out.println("Error accessing file: " + e.getMessage());

}

}

public void updateRoom() {

System.out.println("Enter Room Number to Update:");

int roomNo = input.nextInt();

input.nextLine(); // Consume the newline character

boolean roomFound = false;

try {

ObjectInputStream objectInputStream = new ObjectInputStream(new FileInputStream("Roomrecord.txt"));

try {

while (true) {

Room room = (Room) objectInputStream.readObject();

rooms.add(room);

}

} catch (EOFException e) {

// Reached end of file

}

objectInputStream.close();

for (Room room : rooms) {

if (room.getRoomNo() == roomNo) {

roomFound = true;

break;

}

}

if (roomFound) {

for (Room room : rooms) {

if (room.getRoomNo() == roomNo) {

System.out.print("Enter new Room Capacity: ");

int capacity = input.nextInt();

input.nextLine(); // Consume the newline character

System.out.print("Enter new Room Type (Classroom, lab, seminarHall): ");

String type = checkRoomType() ;

System.out.print("Enter new department of the Room: ");

String department = checkDepartment();

room.setCapacity(capacity);

room.setType(type);

room.setDepartment(department);

break;

}

}

try {

ObjectOutputStream objectOutputStream = new ObjectOutputStream(new FileOutputStream("Roomrecord.txt"));

for (Room room : rooms) {

objectOutputStream.writeObject(room);

}

objectOutputStream.close();

} catch (IOException e) {

System.out.println("Error writing to file: " + e.getMessage());

}

System.out.println("Room information updated successfully!");

} else {

System.out.println("Sorry...Room not Found!");

}

} catch (IOException | ClassNotFoundException e) {

System.out.println("Error accessing file: " + e.getMessage());

}

}

public void viewRoom() {

System.out.print("Enter Room No to View: ");

String rNo = checkRoomNO();

int roomNo = Integer.parseInt(rNo);

input.nextLine(); // Consume the newline character

boolean roomFound = false;

try {

ObjectInputStream objectInputStream = new ObjectInputStream(new FileInputStream("Roomrecord.txt"));

try {

while (true) {

Room room = (Room) objectInputStream.readObject();

rooms.add(room);

}

} catch (EOFException e) {

// Reached end of file

}

objectInputStream.close();

for (Room room : rooms) {

if (room.getRoomNo() == roomNo) {

roomFound = true;

System.out.println("Room Details:");

System.out.println("-----------------------");

System.out.println("Room No: " + room.getRoomNo());

System.out.println("Capacity: " + room.getCapacity());

System.out.println("Type: " + room.getType());

System.out.println("Department: " + room.getDepartment());

System.out.println("-----------------------");

break;

}

}

if (!roomFound) {

System.out.println("Room not Found!");

}

} catch (IOException | ClassNotFoundException e) {

System.out.println("Error accessing file: " + e.getMessage());

}

}

public void searchRoom(int attendees) {

boolean roomFound = false;

try {

ObjectInputStream objectInputStream = new ObjectInputStream(new FileInputStream("Roomrecord.txt"));

try {

while (true) {

Room room = (Room) objectInputStream.readObject();

rooms.add(room);

}

} catch (EOFException e) {

// Reached end of file

}

objectInputStream.close();

for (Room room : rooms) {

if (room.getCapacity() >= attendees) {

System.out.println("Available Rooms:");

System.out.println("-----------------------");

System.out.println("Room No: " + room.getRoomNo());

System.out.println("Capacity: " + room.getCapacity());

System.out.println("Type: " + room.getType());

System.out.println("Department: " + room.getDepartment());

System.out.println("-----------------------");

roomFound = true;

}

}

} catch (IOException | ClassNotFoundException e) {

System.out.println("Error searching room: " + e.getMessage());

}

if (!roomFound) {

System.out.println("No available rooms found for the specified capacity.");

}

}

public Room findRoomByNumber(int roomNumber) {

try {

ObjectInputStream objectInputStream = new ObjectInputStream(new FileInputStream("Roomrecord.txt"));

try {

while (true) {

Room room = (Room) objectInputStream.readObject();

rooms.add(room);

}

} catch (EOFException e) {

// Reached end of file

}

objectInputStream.close();

for (Room room : rooms) {

if (room.getRoomNo() == roomNumber) {

return room;

}

}

} catch (IOException | ClassNotFoundException e) {

System.out.println("Error accessing file: " + e.getMessage());

}

return null;

}

public String checkRoomType()

{

String s;

boolean valid;

do {

s = input.nextLine();

valid = s.matches("[a-zA-Z]+");

if (!valid) {

System.out.println("Enter Correct Room type (e.g class,seminar,Lab): ");

}

}

while (!valid);

return s;

}

public String checkDepartment()

{

String s;

boolean valid;

do {

s = input.nextLine();

valid = s.matches("[a-zA-Z]+");

if (!valid) {

System.out.println("Enter Correct Department (e.g Computer,English,BBA): ");

}

}

while (!valid);

return s;

}

public String checkRoomNO()

{

String s;

boolean valid;

do {

s = input.nextLine();

valid = s.matches("^\\d+$");

if (!valid) {

System.out.println("Enter Correct Room NO (e.g 121): ");

}

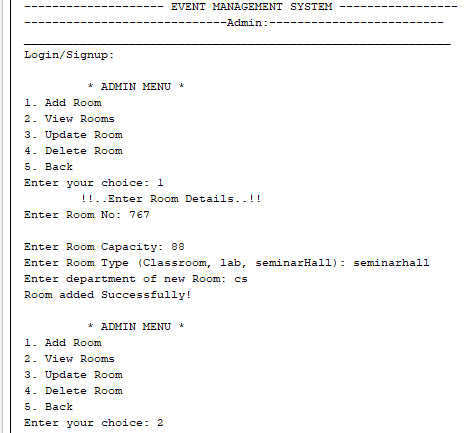
}

while (!valid);

return s;

}

}



A screenshot of a computer

Description automatically generated with medium confidence



# Room Class:

package com.mycompany.ems;

import java.io.Serializable;

public class Room implements Serializable {

private int roomNo;

private int capacity;

private String type;

private String department;

public Room()

{

roomNo = 000;

capacity = 00;

type = null;

department = null;

}

public void setRoomNo(int roomNo)

{

this.roomNo = roomNo;

}

public void setCapacity(int capacity)

{

this.capacity = capacity;

}

public void setType(String type)

{

this.type = type;

}

public void setDepartment(String department)

{

this.department = department;

}

public int getRoomNo()

{

return roomNo;

}

public int getCapacity()

{

return capacity;

}

public String getType()

{

return type;

}

public String getDepartment()

{

return department;

}

public void displayRoom()

{

System.out.println("Room No: " + roomNo + "\nRoom Capacity: " + capacity + "\nRoom Type: " + type + "\nRoom location: " + department);

}

}

# User Class:

package com.mycompany.ems;

import java.util.Scanner;

public class User {

protected Scanner input;

public String username;

public String password;

protected int userId;

protected String firstName;

protected String lastName;

protected String userCNIC;

protected String userPhoneNo;

protected String userEMail;

protected String userDepartment;

public User() {

input = new Scanner(System.in);

userId = (int) (Math.random() \* 999);

password = null;

username = null;

firstName = null;

lastName = null;

userCNIC = null;

userPhoneNo = null;

userEMail = null;

userDepartment = null;

}

public void UserMenu()

{

Scanner input = new Scanner(System.in);

while (true) {

System.out.println("\n\t\t\t\*\* Menu \*\*");

System.out.println("1. Student Account");

System.out.println("2. Faculty Account");

System.out.println("3. Exit");

System.out.print("Enter your choice: ");

System.out.print("Enter your choice: ");

String userChoice = input.next();

try

{

int choice = Integer.parseInt(userChoice);

switch (choice) {

case 1:

Student student = new Student();

student.StudentMenu();

break;

case 2:

Faculty faculty = new Faculty();

faculty.FacultyMenu();

break;

case 3:

System.out.println("Exiting program...");

System.exit(0);

default:

System.out.println("Invalid choice. Please try again.");

break;

}

} catch (NumberFormatException e) {

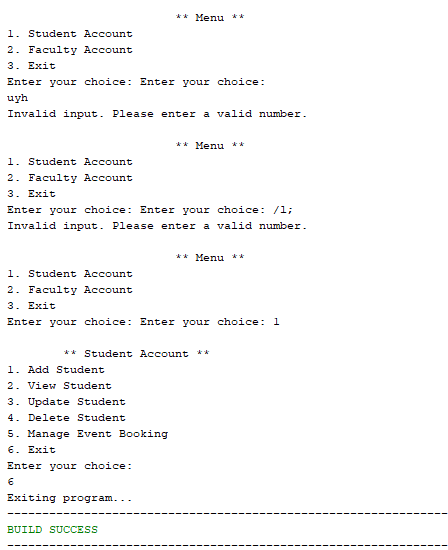
System.out.println("Invalid input. Please enter a valid number.");

}

}

}

}



# Student Class:

package com.mycompany.ems;

import java.io.\*;

import java.util.\*;

public class Student extends User implements Serializable {

private ArrayList<Student> students;

private String regNo;

private String sClass;

Student() {

super();

students = new ArrayList<>();

regNo = null;

sClass = null;

}

public void setRegNo(String regNo) {

this.regNo = regNo;

}

public void setStudentClass(String sClass) {

this.sClass = sClass;

}

public void StudentMenu()

{

BookingEvent newEvent = new BookingEvent();

Scanner input = new Scanner(System.in);

while (true) {

System.out.println("\n\t\*\* Student Account \*\*");

System.out.println("1. Add Student");

System.out.println("2. View Student");

System.out.println("3. Update Student");

System.out.println("4. Delete Student");

System.out.println("5. Manage Event Booking");

System.out.println("6. Exit");

System.out.print("Enter your choice: ");

String studentChoice = input.next();

try

{

int choice = Integer.parseInt(studentChoice);

switch (choice) {

case 1:

addStudent();

break;

case 2:

viewStudent();

break;

case 3:

updateStudent();

break;

case 4:

deleteStudent();

break;

case 5:

newEvent.EventBookingMenu();

break;

case 6:

System.out.println("Exiting program...");

System.exit(0);

default:

System.out.println("Invalid choice. Please try again.");

break;

}

} catch (NumberFormatException e) {

System.out.println("Invalid input. Please enter a valid number.");

}

}

}

public void addStudent() {

System.out.println("\t!!..Enter Student Details..!!");

int studentId = (int) (Math.random() \* 999);

System.out.println("Your student Id is: " + studentId);

System.out.print("Enter your Registration Number (without hyphen): ");

String regno;

regno = input.next();

while (regno.isEmpty() || !regno.matches("[A-Z]{2}\\d{2}[A-Z]{3}\\d{3}")) {

System.out.println("Invalid input... Enter registration number: ");

regno = input.next();

input.nextLine();

}

input.nextLine();

System.out.print("Enter your Class/Section e.g BSE3A: ");

String classSection = input.next();

while(classSection.isEmpty() || !classSection.matches("[A-Z]{3}\\d{1}[A-Z]{1}"))

{

System.out.println("This data is invalid...enter class/section:");

classSection = input.next();

}

input.nextLine();

System.out.print("Enter Your First Name: ");

String fName = input.nextLine();

while (fName.isEmpty() || !fName.matches("[a-zA-Z]+")) {

System.out.println("Invalid input. First name should contain only alphabetic characters. Enter your first name:");

fName = input.next();

}

input.nextLine();

System.out.print("Enter Your Last Name: ");

String lName = input.next().trim();

while (lName.isEmpty() || !lName.matches("[a-zA-Z]+")) {

System.out.println("Invalid input. First name should contain only alphabetic characters. Enter your last name:");

lName = input.next();

}

input.nextLine();

System.out.print("Enter your CNIC in (00000-0000000-0) format: ");

String cnic = input.next();

while (cnic.isEmpty() || !cnic.matches("\\d{5}-\\d{7}-\\d")) {

System.out.println("Invalid input. CNIC should be in the format 00000-0000000-0. Enter your CNIC:");

cnic = input.next();

}

input.nextLine();

System.out.print("Enter your phone no (without dashes or spaces): ");

String phoneNo = input.next();

while (phoneNo.isEmpty() || !phoneNo.matches("\\d{11}")) {

System.out.println("Invalid input. phone no should be in the format 00000000000. Enter your Phone no:");

phoneNo = input.next();

}

System.out.print("Enter your E-mail Address: ");

String email = input.next();

while (email.isEmpty() || !email.contains("@")) {

System.out.println("Invalid input. Email should be in the format abc@gmail.com. Enter your email:");

email = input.next();

}

System.out.print("Enter your Department: ");

String department = input.next();

while(department.isEmpty() || !department.matches("[a-zA-Z]+"))

{

System.out.println("Invalid input. deaprtment should contain only alphabetic characters. Enter your department name:");

department = input.next();

}

input.nextLine();

Student newStudent = new Student();

newStudent.userId = studentId;

newStudent.setRegNo(regno);

newStudent.setStudentClass(classSection);

newStudent.firstName = fName;

newStudent.lastName = lName;

newStudent.userCNIC = cnic;

newStudent.userPhoneNo = phoneNo;

newStudent.userEMail = email;

newStudent.userDepartment = department;

students.add(newStudent);

try {

ObjectOutputStream objectOutputStream = new ObjectOutputStream(new FileOutputStream("Studentrecord.txt", true));

for (Student student : students) {

objectOutputStream.writeObject(student);

}

} catch (IOException e) {

System.out.println("Error writing to file: " + e.getMessage());

}

System.out.println("\n\t\t!!..Your Information Saved Successfully..!!");

}

public void deleteStudent() {

System.out.print("Enter Student ID Number to Delete: ");

String studentId = input.next();

while(studentId.isBlank() || !studentId.matches("[0-9]+"))

{

System.out.println("Invalid input. Student id should contain only digits. Enter your Id again:");

studentId = input.next();

input.nextLine();

}

boolean found = false;

for (int i = 0; i < students.size(); i++) {

if (students.get(i).userId == Integer.parseInt(studentId)) {

students.remove(i);

found = true;

break;

}

}

if (found) {

// Remove the booking from the file

try (ObjectOutputStream objectOutputStream = new ObjectOutputStream(new FileOutputStream("Bookingrecord.txt"))) {

for (Student student : students) {

objectOutputStream.writeObject(student);

}

System.out.println("Student Data Deleted successfully.");

} catch (IOException e) {

System.out.println("Error writing to file: " + e.getMessage());

}

} else {

System.out.println("Student Data not found. Cancellation failed.");

}

}

public void updateStudent() {

System.out.print("\n\t!!..Update Student Details..!!");

System.out.print("\nEnter Student ID Number: ");

String studentId = input.next();

while(studentId.isBlank() || !studentId.matches("[0-9]+"))

{

System.out.println("Invalid input. Student id should contain only digits. Enter your Id again:");

studentId = input.next();

input.nextLine();

}

try {

ObjectInputStream objectInputStream = new ObjectInputStream(new FileInputStream("Studentrecord.txt"));

try {

while (true) {

Student student = (Student) objectInputStream.readObject();

students.add(student);

}

} catch (EOFException e) {

// Reached end of file

}

objectInputStream.close();

boolean found = false;

for (int i = 0; i < students.size(); i++) {

if (students.get(i).userId == Integer.parseInt(studentId)) {

Student student = students.get(i);

found = true;

System.out.print("Enter your Registration Number: ");

String regno;

regno = input.next();

while(regno.isBlank())

{

System.out.println("This field cannot be blank..enter reg no:");

regno = input.next();

input.nextLine();

}

System.out.print("Enter your Class/Section: ");

String classSection = input.next();

while(classSection.isBlank())

{

System.out.println("This field cannot be blank..enter class/section:");

classSection = input.next();

input.nextLine();

}

System.out.print("Enter Your First Name: ");

String fName = input.nextLine();

while (fName.isBlank() || !fName.matches("[a-zA-Z]+")) {

System.out.println("Invalid input. First name should contain only alphabetic characters. Enter your first name:");

fName = input.next();

}

System.out.print("Enter Your Last Name: ");

String lName = input.next();

while (lName.isBlank() || !lName.matches("[a-zA-Z]+")) {

System.out.println("Invalid input. First name should contain only alphabetic characters. Enter your last name:");

lName = input.next();

}

System.out.print("Enter your CNIC in (00000-0000000-0) format: ");

String cnic = input.next();

while (cnic.isBlank() || !cnic.matches("\\d{5}-\\d{7}-\\d")) {

System.out.println("Invalid input. CNIC should be in the format 00000-0000000-0. Enter your CNIC:");

cnic = input.next();

}

System.out.print("Enter your phone no (without dashes or spaces): ");

String phoneNo = input.next();

while (phoneNo.isBlank() || !phoneNo.matches("\\d{11}")) {

System.out.println("Invalid input. phone no should be in the format 00000000000. Enter your Phone no:");

phoneNo = input.next();

}

System.out.print("Enter your E-mail Address: ");

String email = input.nextLine();

while (email.isBlank() || !email.contains("@")) {

System.out.println("Invalid input. Email should be in the format abc@gmail.com. Enter your email:");

email = input.nextLine();

}

System.out.print("Enter your Department: ");

String department = input.next();

while(department.isBlank() || !department.matches("[a-zA-Z]+"))

{

System.out.println("Invalid input. deaprtment should contain only alphabetic characters. Enter your department name:");

department = input.next();

}

student.setRegNo(regNo);

student.setStudentClass(classSection);

student.firstName = fName;

student.lastName = lName;

student.userCNIC = cnic;

student.userPhoneNo = phoneNo;

student.userEMail = email;

student.userDepartment = department;

ObjectOutputStream objectOutputStream = new ObjectOutputStream(new FileOutputStream("Studentrecord.txt"));

for (Student s : students) {

objectOutputStream.writeObject(s);

}

objectOutputStream.close();

System.out.println("\n\t\t!!..Student Information Updated Successfully..!!");

found = true;

break;

}

}

if (!found) {

System.out.println("No student record found with ID " + studentId + ".");

}

} catch (IOException | ClassNotFoundException e) {

System.out.println("Error accessing file: " + e.getMessage());

}

}

public void viewStudent() {

System.out.println("\n\t!!..View Student Details..!!");

System.out.print("Enter Student ID Number: ");

String studentId = input.next();

while(studentId.isBlank() || !studentId.matches("[0-9]+"))

{

System.out.println("Invalid input. Student id should contain only digits. Enter your Id again:");

studentId = input.next();

input.nextLine();

}

try {

ObjectInputStream objectInputStream = new ObjectInputStream(new FileInputStream("Studentrecord.txt"));

try {

while (true) {

Student student = (Student) objectInputStream.readObject();

students.add(student);

}

} catch (EOFException e) {

// Reached end of file

}

objectInputStream.close();

boolean found = false;

for (Student student : students) {

if (student.userId == Integer.parseInt(studentId)) {

System.out.println("Student Details:");

System.out.println("-----------------------");

System.out.println("\nStudent ID: " + student.userId);

System.out.println("Registration Number: " + student.regNo);

System.out.println("Class/Section: " + student.sClass);

System.out.println("First Name: " + student.firstName);

System.out.println("Last Name: " + student.lastName);

System.out.println("CNIC: " + student.userCNIC);

System.out.println("Phone Number: " + student.userPhoneNo);

System.out.println("Email Address: " + student.userEMail);

System.out.println("Department: " + student.userDepartment);

System.out.println("-----------------------");

found = true;

break;

}

}

if (!found) {

System.out.println("No student record found with ID " + studentId + ".");

}

} catch (IOException | ClassNotFoundException e) {

System.out.println("Error accessing file: " + e.getMessage());

}

}

}

# Faculty Class:

package com.mycompany.ems;

import java.io.\*;

import java.util.ArrayList;

import java.util.\*;

public class Faculty extends User implements Serializable {

private ArrayList<Faculty> facultys;

private String designation;

Faculty() {

super();

facultys = new ArrayList<>();

designation = null;

}

public void setDesignation(String designation) {

this.designation = designation;

}

public String getDesignation() {

return designation;

}

public void FacultyMenu()

{

Scanner input = new Scanner(System.in);

while (true) {

System.out.print("Enter your choice: ");

System.out.println("\n\t\*\* Faculty Account \*\*");

System.out.println("1. Add Faculty");

System.out.println("2. View Faculty");

System.out.println("3. Update Faculty");

System.out.println("4. Delete Faculty");

System.out.println("5. Manage Event Booking");

System.out.println("6. Exit");

System.out.print("Enter your choice: ");

String facultyChoice = input.next();

try {

int choice = Integer.parseInt(facultyChoice);

switch (choice) {

case 1:

addFaculty();

break;

case 2:

viewFaculty();

break;

case 3:

updateFaculty();

break;

case 4:

deleteFaculty();

break;

case 5:

BookingEvent newEvent=new BookingEvent();

newEvent.EventBookingMenu();

break;

case 6:

System.out.println("Exiting program...");

System.exit(0);

default:

System.out.println("Invalid choice. Please try again.");

break;

}

} catch (NumberFormatException e) {

System.out.println("Invalid input. Please enter a valid number.");

}

}

}

public void addFaculty() {

System.out.println("\t!!..Enter Faculty Details..!!");

int facultyId = (int) (Math.random() \* 999);

System.out.println("Your faculty Id is: " + facultyId);

System.out.print("Enter Your First Name: ");

String fName = input.next();

while (fName.isEmpty() || !fName.matches("[a-zA-Z]+")) {

System.out.println("Invalid input. First name should contain only alphabetic characters. Enter your first name:");

fName = input.next();

}

input.nextLine();

System.out.print("Enter Your Last Name: ");

String lName = input.next().trim();

while (lName.isEmpty() || !lName.matches("[a-zA-Z]+")) {

System.out.println("Invalid input. First name should contain only alphabetic characters. Enter your last name:");

lName = input.next();

}

input.nextLine();

System.out.print("Enter your CNIC in (00000-0000000-0) format: ");

String cnic = input.next();

while (cnic.isEmpty() || !cnic.matches("\\d{5}-\\d{7}-\\d")) {

System.out.println("Invalid input. CNIC should be in the format 00000-0000000-0. Enter your CNIC:");

cnic = input.next();

}

input.nextLine();

System.out.print("Enter your phone no (without dashes or spaces): ");

String phoneNo = input.next();

while (phoneNo.isEmpty() || !phoneNo.matches("\\d{11}")) {

System.out.println("Invalid input. phone no should be in the format 00000000000. Enter your Phone no:");

phoneNo = input.next();

}

System.out.print("Enter your E-mail Address: ");

String email = input.next();

while (email.isEmpty() || !email.contains("@")) {

System.out.println("Invalid input. Email should be in the format abc@gmail.com. Enter your email:");

email = input.next();

}

input.nextLine();

System.out.print("Enter your Department: ");

String department = input.next();

while(department.isEmpty() || !department.matches("[a-zA-Z]+"))

{

System.out.println("Invalid input. deaprtment should contain only alphabetic characters. Enter your department name:");

department = input.next();

}

input.nextLine();

System.out.print("Enter your Designation: ");

String desig = input.next();

while(desig.isEmpty() || !desig.matches("[a-zA-Z]+"))

{

System.out.println("Invalid input. designation should contain only alphabetic characters. Enter your designation name:");

desig = input.next();

}

Faculty newfaculty = new Faculty();

newfaculty.userId = facultyId;

newfaculty.firstName = fName;

newfaculty.lastName = lName;

newfaculty.userCNIC = cnic;

newfaculty.userPhoneNo = phoneNo;

newfaculty.userEMail = email;

newfaculty.userDepartment = department;

newfaculty.setDesignation(desig);

facultys.add(newfaculty);

try {

ObjectOutputStream objectOutputStream = new ObjectOutputStream(new FileOutputStream("Facultyrecord.txt", true));

for (Faculty faculty : facultys) {

objectOutputStream.writeObject(faculty);

}

} catch (IOException e) {

System.out.println("Error writing to file: " + e.getMessage());

}

System.out.println("\n\t\t!!..Your Information Saved Successfully..!!");

}

public void updateFaculty() {

System.out.print("\n\t!!..Update Faculty Details..!!");

String id;

do {

System.out.print("Enter Faculty ID Number to update: ");

id = input.nextLine().trim();

if (!id.matches("[0-9]+")) {

System.out.println("Invalid ID. Please enter only numeric digits.");

} else if (id.isEmpty()) {

System.out.println("ID cannot be empty. Please enter a value.");

}

else { }

} while (!id.matches("[0-9]+") || id.isEmpty());

int facultyId = Integer.parseInt(id);

input.nextLine(); // Consume the newline character

try {

ObjectInputStream objectInputStream = new ObjectInputStream(new FileInputStream("Facultyrecord.txt"));

try {

while (true) {

Faculty faculty = (Faculty) objectInputStream.readObject();

facultys.add(faculty);

}

} catch (EOFException e) {

// Reached end of file

}

objectInputStream.close();

boolean found = false;

for (int i = 0; i < facultys.size(); i++) {

if (facultys.get(i).userId == facultyId) {

Faculty faculty = facultys.get(i);

found = true;

System.out.print("Enter Your First Name: ");

String fName = input.next();

while (fName.isEmpty() || !fName.matches("[a-zA-Z]+")) {

System.out.println("Invalid input. First name should contain only alphabetic characters. Enter your first name:");

fName = input.next();

}

input.nextLine();

System.out.print("Enter Your Last Name: ");

String lName = input.next().trim();

while (lName.isEmpty() || !lName.matches("[a-zA-Z]+")) {

System.out.println("Invalid input. First name should contain only alphabetic characters. Enter your last name:");

lName = input.next();

}

input.nextLine();

System.out.print("Enter your CNIC in (00000-0000000-0) format: ");

String cnic = input.next();

while (cnic.isEmpty() || !cnic.matches("\\d{5}-\\d{7}-\\d")) {

System.out.println("Invalid input. CNIC should be in the format 00000-0000000-0. Enter your CNIC:");

cnic = input.next();

}

input.nextLine();

System.out.print("Enter your phone no (without dashes or spaces): ");

String phoneNo = input.next();

while (phoneNo.isEmpty() || !phoneNo.matches("\\d{11}")) {

System.out.println("Invalid input. phone no should be in the format 00000000000. Enter your Phone no:");

phoneNo = input.next();

}

System.out.print("Enter your E-mail Address: ");

String email = input.next();

while (email.isEmpty() || !email.contains("@")) {

System.out.println("Invalid input. Email should be in the format abc@gmail.com. Enter your email:");

email = input.next();

}

input.nextLine();

System.out.print("Enter your Department: ");

String department = input.next();

while(department.isEmpty() || !department.matches("[a-zA-Z]+"))

{

System.out.println("Invalid input. deaprtment should contain only alphabetic characters. Enter your department name:");

department = input.next();

}

input.nextLine();

System.out.print("Enter your Designation: ");

String desig = input.next();

while(desig.isEmpty() || !desig.matches("[a-zA-Z]+"))

{

System.out.println("Invalid input. designation should contain only alphabetic characters. Enter your designation name:");

desig = input.next();

}

faculty.firstName = firstName;

faculty.lastName = lastName;

faculty.setDesignation(desig);

faculty.userCNIC = cnic;

faculty.userPhoneNo = phoneNo;

faculty.userEMail = email;

faculty.userDepartment = department;

ObjectOutputStream objectOutputStream = new ObjectOutputStream(

new FileOutputStream("Facultyrecord.txt"));

for (Faculty f : facultys) {

objectOutputStream.writeObject(f);

}

objectOutputStream.close();

System.out.println("\n\t\t!!..Faculty Information Updated Successfully..!!");

break;

}

}

if (!found) {

System.out.println("No faculty record found with ID " + facultyId + ".");

}

} catch (IOException | ClassNotFoundException e) {

System.out.println("Error accessing file: " + e.getMessage());

}

}

public void deleteFaculty() {

String id;

do {

System.out.print("Enter Faculty ID Number to Delete: ");

id = input.nextLine().trim();

if (!id.matches("[0-9]+")) {

System.out.println("Invalid ID. Please enter only numeric digits.");

} else if (id.isEmpty()) {

System.out.println("ID cannot be empty. Please enter a value.");

}

else { }

} while (!id.matches("[0-9]+") || id.isEmpty());

int facultyId = Integer.parseInt(id);

input.nextLine(); // Consume the newline character

try {

ObjectInputStream objectInputStream = new ObjectInputStream(new FileInputStream("Facultyrecord.txt"));

ArrayList<Faculty> facultyRecords = new ArrayList<>();

try {

while (true) {

Faculty faculty = (Faculty) objectInputStream.readObject();

facultyRecords.add(faculty);

}

} catch (EOFException e) {

// Reached end of file

}

objectInputStream.close();

boolean found = false;

for (int i = 0; i < facultyRecords.size(); i++) {

if (facultyRecords.get(i).userId == facultyId) {

facultyRecords.remove(i);

found = true;

break;

}

}

if (found) {

File file = new File("Facultyrecord.txt");

if (file.exists()) {

file.delete();

}

ObjectOutputStream objectOutputStream = new ObjectOutputStream(

new FileOutputStream("Facultyrecord.txt"));

for (Faculty faculty : facultyRecords) {

objectOutputStream.writeObject(faculty);

}

objectOutputStream.close();

System.out.println("Faculty record with ID " + facultyId + " has been deleted.");

} else {

System.out.println("No faculty record found with ID " + facultyId + ".");

}

} catch (IOException | ClassNotFoundException e) {

System.out.println("Error accessing file: " + e.getMessage());

}

}

public void viewFaculty() {

System.out.println("\n\t!!..View Faculty Details..!!");

String id;

do {

System.out.print("Enter Faculty ID Number: ");

id = input.nextLine().trim();

if (!id.matches("[0-9]+")) {

System.out.println("Invalid ID. Please enter only numeric digits.");

} else if (id.isEmpty()) {

System.out.println("ID cannot be empty. Please enter a value.");

}

else { }

} while (!id.matches("[0-9]+") || id.isEmpty());

int facultyId = Integer.parseInt(id);

input.nextLine(); // Consume the newline character

try {

ObjectInputStream objectInputStream = new ObjectInputStream(new FileInputStream("Facultyrecord.txt"));

ArrayList<Faculty> facultyRecords = new ArrayList<>();

try {

while (true) {

Faculty faculty = (Faculty) objectInputStream.readObject();

facultyRecords.add(faculty);

}

} catch (EOFException e) {

// Reached end of file

}

objectInputStream.close();

boolean found = false;

for (Faculty faculty : facultyRecords) {

if (faculty.userId == facultyId) {

found = true;

System.out.println("\nFaculty ID: " + faculty.userId);

System.out.println("First Name: " + faculty.firstName);

System.out.println("Last Name: " + faculty.lastName);

System.out.println("Designation: " + faculty.getDesignation());

System.out.println("CNIC: " + faculty.userCNIC);

System.out.println("Phone Number: " + faculty.userPhoneNo);

System.out.println("Email Address: " + faculty.userEMail);

System.out.println("Department: " + faculty.userDepartment);

break;

}

}

if (!found) {

System.out.println("No faculty record found with ID " + facultyId + ".");

}

} catch (IOException | ClassNotFoundException e) {

System.out.println("Error accessing file: " + e.getMessage());

}

}

}

# Booking Class:

package com.mycompany.ems;

import java.io.\*;

import java.util.ArrayList;

import java.util.Scanner;

public class BookingEvent implements Serializable

{

private ArrayList<BookingEvent> bookings;

private transient Scanner input; // Use transient keyword to exclude Scanner from serialization1

private String date;

private String time;

private String eventType;

private int bookingId;

private int userId;

private int roomNo;

private int attendees;

public BookingEvent() {

bookings = new ArrayList<>();

input = new Scanner(System.in);

date = null;

time = null;

eventType = null;

bookingId = 0;

userId = 0;

roomNo = 0;

attendees = 0;

}

public void setDate(String date) {

this.date = date;

}

public String getDate() {

return date;

}

public void setTime(String time) {

this.time = time;

}

public String getTime() {

return time;

}

public void setEventType(String eventType) {

this.eventType = eventType;

}

public String getEventType() {

return eventType;

}

public void setBookingId(int bookingId)

{

this.bookingId = bookingId;

}

public int getBookingId() {

return bookingId;

}

public void setUserId(int userId) {

this.userId = userId;

}

public int getUserId() {

return userId;

}

public void setRoomNo(int roomNo) {

this.roomNo = roomNo;

}

public int getRoomNo() {

return roomNo;

}

public void setAttendees(int attendees) {

this.attendees = attendees;

}

public int getAttendees() {

return attendees;

}

public void EventBookingMenu() {

Scanner input = new Scanner(System.in);

System.out.println("Welcome to the Smart Room Booking Event System!");

// Create a menu for user interaction

String choice;

while(true){

System.out.println("\n----- Event Booking Menu -----");

System.out.println("1. Create Booking");

System.out.println("2. Cancel Booking");

System.out.println("3. update Booking");

System.out.println("4. View Booking");

System.out.println("0. Exit");

System.out.print("Enter your choice: ");

choice = input.next();

while(choice.isBlank())

{

System.out.println("Field cannot be empty..enter an option:");

choice = input.next();

input.nextLine(); // Consume the remaining newline character

}

int ch;

try {

ch = Integer.parseInt(choice);

switch (ch) {

case 1:

createBooking();

break;

case 2:

cancelBooking();

break;

case 3:

updateBooking();

break;

case 4:

viewBooking();

break;

case 0:

System.out.println("Exiting the program. Goodbye!");

System.exit(ch);

break;

default:

System.out.println("Invalid choice. Please try again.");

break;

}

} catch(NumberFormatException e){

System.out.println("Invalid input. Please enter a valid number.");

}

//while (ch != 2);

}

}

public void createBooking() {

System.out.println("\t!!..Enter Booking Details..!!");

int bookingId = (int) (Math.random() \* 999);

System.out.println("Your Booking Id Number: " + bookingId);

System.out.print("Enter Your Id Number: ");

int id = input.nextInt();

input.nextLine(); // Consume the remaining newline character

System.out.print("Enter Date in (dd/mm/yyyy) format: ");

String date = checkDate();

System.out.print("Enter time in the format HH:MM (24-hour format): ");

String time = checkTime();

System.out.print("Enter The Event Type of Booking: ");

String eventType = checkEventType();

System.out.print("Enter The Number of Attendees: ");

int attendees = input.nextInt();

// Check for available rooms that match the specified date, time, and capacity

Admin admin = new Admin();

admin.searchRoom(attendees);

System.out.print("Enter the Room Number for your booking: ");

int roomNumber = input.nextInt();

// Find the selected room from the available rooms

Room selectedRoom = admin.findRoomByNumber(roomNumber);

if (selectedRoom == null) {

System.out.println("Invalid room number. Booking creation failed.");

return;

}

// Check if the room is available at the specified date and time

if (!checkRoom(selectedRoom, date, time)) {

System.out.println("The room is already booked at the specified date and time. Booking creation failed.");

return;

}

// Create the booking

BookingEvent booking = new BookingEvent();

booking.setDate(date);

booking.setTime(time);

booking.setEventType(eventType);

booking.setBookingId(bookingId);

booking.setUserId(id);

booking.setRoomNo(selectedRoom.getRoomNo());

booking.setAttendees(attendees);

// Store the booking in the system (e.g., add it to the ArrayList)

bookings.add(booking);

try (ObjectOutputStream objectOutputStream = new ObjectOutputStream(new FileOutputStream("Bookingrecord.txt"))) {

for (BookingEvent b : bookings) {

objectOutputStream.writeObject(b);

}

} catch (IOException e) {

System.out.println("Error writing to file: " + e.getMessage());

}

System.out.println("Booking created and added successfully!");

}

public void cancelBooking() {

System.out.println("\t!!..Cancel Booking..!!");

System.out.print("Enter Your Booking Id Number: ");

int bookingId = input.nextInt();

input.nextLine(); // Consume the remaining newline character

// Check if the booking exists in the record

boolean bookingFound = false;

for (BookingEvent booking : bookings) {

if (booking.getBookingId() == bookingId) {

bookings.remove(booking);

bookingFound = true;

break;

}

}

if (bookingFound) {

// Remove the booking from the file

try (ObjectOutputStream objectOutputStream = new ObjectOutputStream(new FileOutputStream("Bookingrecord.txt"))) {

for (BookingEvent booking : bookings) {

objectOutputStream.writeObject(booking);

}

System.out.println("Booking canceled successfully.");

} catch (IOException e) {

System.out.println("Error writing to file: " + e.getMessage());

}

} else {

System.out.println("Booking not found. Cancellation failed.");

}

}

public void updateBooking() {

System.out.println("\t!!..Update Booking..!!");

System.out.print("Enter Your Booking Id Number: ");

int bookingId = input.nextInt();

input.nextLine(); // Consume the remaining newline character

boolean bookingFound = false;

try (ObjectInputStream objectInputStream = new ObjectInputStream(new FileInputStream("Bookingrecord.txt"))) {

try {

while (true) {

BookingEvent booking = (BookingEvent) objectInputStream.readObject();

bookings.add(booking);

}

} catch (EOFException e) {

// Reached end of file

}

for (BookingEvent booking : bookings) {

if (booking.getBookingId() == bookingId) {

System.out.print("Enter The New Date in (dd/mm/yyyy) format: ");

String newDate = checkDate();

booking.setDate(newDate);

System.out.print("Enter The New Time in the format HH:MM (24-hour format): ");

String newTime = checkTime();

booking.setTime(newTime);

System.out.print("Enter The New Event Type of Booking: ");

String newEventType = checkEventType();

booking.setEventType(newEventType);

System.out.print("Enter The New Number of Attendees: ");

int newAttendees = input.nextInt();

input.nextLine(); // Consume the remaining newline character

Admin admin = new Admin();

admin.searchRoom(newAttendees);

booking.setAttendees(newAttendees);

System.out.print("Enter The New Room Number for the Booking: ");

int newRoomNumber = input.nextInt();

input.nextLine(); // Consume the remaining newline character

// Check if the new room number exists in the room record file

Room newRoom = admin.findRoomByNumber(newRoomNumber);

if (newRoom == null) {

System.out.println("Invalid room number. Booking update failed.");

return;

}

// Check if the room is available at the specified date and time

if (!checkRoom(newRoom, newDate ,newTime)) {

System.out.println("The room is already booked at the specified date and time. Booking creation failed.");

return;

}

// Update the booking information

booking.setRoomNo(newRoom.getRoomNo());

// Update the booking in the file

ObjectOutputStream objectOutputStream = new ObjectOutputStream(new FileOutputStream("Bookingrecord.txt"));

for (BookingEvent updatebooking : bookings) {

objectOutputStream.writeObject(updatebooking);

}

objectOutputStream.close();

System.out.println("Booking updated successfully!");

bookingFound = true;

break;

}

}

} catch (IOException | ClassNotFoundException e) {

System.out.println("Error updating Booking information: " + e.getMessage());

}

if (!bookingFound) {

System.out.println("Sorry...Booking not Found!");

}

}

public void viewBooking() {

System.out.print("Enter Your Booking Id Number: ");

int bookingId = input.nextInt();

input.nextLine(); // Consume the newline character

boolean bookingFound = false;

try {

ObjectInputStream objectInputStream = new ObjectInputStream(new FileInputStream("Bookingrecord.txt"));

try {

while (true) {

BookingEvent booking = (BookingEvent) objectInputStream.readObject();

bookings.add(booking);

}

} catch (EOFException e) {

// Reached end of file

}

objectInputStream.close();

for (BookingEvent booking : bookings) {

if (booking.getBookingId() == bookingId) {

System.out.println("Booking Details:");

System.out.println("-----------------------");

System.out.println("User Id: " + booking.getUserId());

System.out.println("Date: " + booking.getDate());

System.out.println("Time: " + booking.getTime());

System.out.println("Room Number: " + booking.getRoomNo());

System.out.println("Attendees: " + booking.getAttendees());

System.out.println("Event Type: " + booking.getEventType());

System.out.println("-----------------------");

bookingFound = true;

break;

}

}

if (!bookingFound) {

System.out.println("Booking not Found!");

}

} catch (IOException | ClassNotFoundException e) {

System.out.println("Error accessing file: " + e.getMessage());

}

}

public String checkDate() {

String s;

boolean valid;

do {

s = input.nextLine();

valid = s.matches("^(3[01]|[12][0-9]|0[1-9])/(1[0-2]|0[1-9])/[0-9]{4}$");

if (!valid) {

System.out.println("Enter a correct date in the format YYYY-MM-DD: ");

}

} while (!valid);

return s;

}

public String checkTime() {

String s;

boolean valid;

do {

s = input.nextLine();

valid = s.matches("^([01]?[0-9]|2[0-3]):[0-5][0-9]$");

if (!valid) {

System.out.println("Enter a correct time in the format HH:MM (24-hour format): ");

}

} while (!valid);

return s;

}

public boolean checkRoom(Room room, String date, String time) {

for (BookingEvent booking : bookings) {

if (booking.getRoomNo() == room.getRoomNo() && booking.getDate().equals(date) && booking.getTime().equals(time)) {

return false; // Room is already booked at the specified date and time

}

}

return true; // Room is available at the specified date and time

}

public String checkEventType()

{

String s;

boolean valid;

do {

s = input.nextLine();

valid = s.matches("[a-zA-Z]+");

if (!valid) {

System.out.println("Enter Correct Event type (e.g Lecture): ");

}

}

while (!valid);

return s;

}

}

