JAVA MINI PROJECT

Sanchi Dalvi 23302D028

Janhavi Taware 23302D0049

Rushabh Bagad 2332D0034

**Java Venu Booking Sytem**

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import com.toedter.calendar.JDateChooser;

import java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

import java.text.SimpleDateFormat;

import java.util.\*;

public class VenueBookingSystem extends JFrame {

private DefaultTableModel tableModel;

private JTable bookingTable;

private JComboBox<String> venueComboBox, startTimeComboBox, startAmPmComboBox, endTimeComboBox, endAmPmComboBox;

private JDateChooser dateChooser;

private JTextField eventField, facultyField;

private JButton logoutButton, bookButton, editButton, deleteButton;

private String loggedInUser, role;

private static final String FILE\_NAME = "bookings.txt";

public VenueBookingSystem(String username, String role) {

this.loggedInUser = username;

this.role = role;

setTitle("Venue Booking System - VSIT");

setSize(900, 600);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(new BorderLayout());

ImageIcon logoIcon = new ImageIcon(getClass().getResource("/vsit.png"));

JLabel logoLabel = new JLabel(logoIcon, JLabel.CENTER);

add(logoLabel, BorderLayout.NORTH);

JTabbedPane tabbedPane = new JTabbedPane();

if (!role.equals("admin")) {

tabbedPane.addTab("📚 Faculty Booking", createFacultyPanel());

}

tabbedPane.addTab("📋 View & Edit Bookings", createBookingTablePanel());

add(tabbedPane, BorderLayout.CENTER);

logoutButton = new JButton("🚪 Logout");

logoutButton.setBackground(Color.RED);

logoutButton.setForeground(Color.WHITE);

logoutButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new LoginPage();

dispose();

}

});

add(logoutButton, BorderLayout.SOUTH);

loadBookings();

setVisible(true);

}

VenueBookingSystem(String loggedInUser) {

throw new UnsupportedOperationException("Not supported yet."); }

private JPanel createFacultyPanel() {

JPanel panel = new JPanel(new GridLayout(9, 2, 10, 10));

panel.add(new JLabel("📍 Venue:"));

String[] venues = {"Seminar Hall", "Auditorium", "Classroom 101"};

venueComboBox = new JComboBox<>(venues);

panel.add(venueComboBox);

panel.add(new JLabel("📅 Date:"));

dateChooser = new JDateChooser();

dateChooser.setDateFormatString("dd MMM yyyy");

panel.add(dateChooser);

panel.add(new JLabel("⏰ Start Time:"));

startTimeComboBox = new JComboBox<>(new String[]{"09:00", "10:00", "11:00", "12:00", "01:00", "02:00", "03:00", "04:00", "05:00", "06:00", "07:00", "08:00"});

startAmPmComboBox = new JComboBox<>(new String[]{"AM", "PM"});

JPanel startTimePanel = new JPanel(new FlowLayout());

startTimePanel.add(startTimeComboBox);

startTimePanel.add(startAmPmComboBox);

panel.add(startTimePanel);

panel.add(new JLabel("⏳ End Time:"));

endTimeComboBox = new JComboBox<>(new String[]{"09:00", "10:00", "11:00", "12:00", "01:00", "02:00", "03:00", "04:00", "05:00", "06:00", "07:00", "08:00"});

endAmPmComboBox = new JComboBox<>(new String[]{"AM", "PM"});

JPanel endTimePanel = new JPanel(new FlowLayout());

endTimePanel.add(endTimeComboBox);

endTimePanel.add(endAmPmComboBox);

panel.add(endTimePanel);

panel.add(new JLabel("🎤 Event:"));

eventField = new JTextField();

panel.add(eventField);

panel.add(new JLabel("👨‍🏫 Faculty:"));

facultyField = new JTextField(loggedInUser);

panel.add(facultyField);

bookButton = new JButton("✅ Book Venue");

bookButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

bookVenue();

}

});

panel.add(bookButton);

return panel;

}

private JPanel createBookingTablePanel() {

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

String[] columns = {"Venue", "Date", "Start Time", "End Time", "Event", "Faculty"};

tableModel = new DefaultTableModel(columns, 0);

bookingTable = new JTable(tableModel);

JScrollPane scrollPane = new JScrollPane(bookingTable);

panel.add(scrollPane, BorderLayout.CENTER);

JPanel buttonPanel = new JPanel(new FlowLayout());

editButton = new JButton("✏️ Edit");

editButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

editBooking();

}

});

deleteButton = new JButton("❌ Delete");

deleteButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

deleteBooking();

}

});

buttonPanel.add(editButton);

buttonPanel.add(deleteButton);

panel.add(buttonPanel, BorderLayout.SOUTH);

return panel;

}

private void bookVenue() {

String venue = (String) venueComboBox.getSelectedItem();

String event = eventField.getText();

String faculty = facultyField.getText();

Date date = dateChooser.getDate();

String startTime = startTimeComboBox.getSelectedItem() + " " + startAmPmComboBox.getSelectedItem();

String endTime = endTimeComboBox.getSelectedItem() + " " + endAmPmComboBox.getSelectedItem();

if (date == null || event.isEmpty() || faculty.isEmpty()) {

JOptionPane.showMessageDialog(null, "⚠️ Please fill in all fields!");

return;

}

SimpleDateFormat sdf = new SimpleDateFormat("dd MMM yyyy");

String formattedDate = sdf.format(date);

tableModel.addRow(new Object[]{venue, formattedDate, startTime, endTime, event, faculty});

saveBookings();

JOptionPane.showMessageDialog(null, "✅ Booking Successful!");

}

private void editBooking() {

int selectedRow = bookingTable.getSelectedRow();

if (selectedRow == -1) {

JOptionPane.showMessageDialog(null, "⚠️ Please select a booking to edit.");

return;

}

String faculty = (String) tableModel.getValueAt(selectedRow, 5);

if (!role.equals("admin") && !faculty.equals(loggedInUser)) {

JOptionPane.showMessageDialog(null, "❌ You can only edit your own bookings.");

return;

}

String newFaculty = JOptionPane.showInputDialog("Enter new faculty name:", faculty);

if (newFaculty != null && !newFaculty.isEmpty()) {

tableModel.setValueAt(newFaculty, selectedRow, 5);

saveBookings();

}

}

private void deleteBooking() {

int selectedRow = bookingTable.getSelectedRow();

if (selectedRow == -1) {

JOptionPane.showMessageDialog(null, "⚠️ Please select a booking to delete.");

return;

}

tableModel.removeRow(selectedRow);

saveBookings();

}

private void saveBookings() {

try (BufferedWriter writer = new BufferedWriter(new FileWriter(FILE\_NAME))) {

for (int i = 0; i < tableModel.getRowCount(); i++) {

StringBuilder line = new StringBuilder();

line.append(tableModel.getValueAt(i, 0).toString()).append(",");

line.append(tableModel.getValueAt(i, 1).toString()).append(",");

line.append(tableModel.getValueAt(i, 2).toString()).append(",");

line.append(tableModel.getValueAt(i, 3).toString()).append(",");

line.append(tableModel.getValueAt(i, 4).toString()).append(",");

line.append(tableModel.getValueAt(i, 5).toString());

writer.write(line.toString());

writer.newLine();

}

} catch (IOException e) {

System.out.println("⚠️ Error saving bookings.");

}

}

private void loadBookings() {

File file = new File(FILE\_NAME);

if (!file.exists()) return;

try (BufferedReader reader = new BufferedReader(new FileReader(file))) {

String line;

tableModel.setRowCount(0);

while ((line = reader.readLine()) != null) {

String[] parts = line.split(",");

if (parts.length == 6) {

tableModel.addRow(parts);

}

}

} catch (IOException e) {

System.out.println("⚠️ Error loading bookings: " + e.getMessage());

}

}

}

**Booking.java**

import java.util.Objects;

public class Booking {

private String venue;

private String date;

private String timeSlot;

private String bookedBy;

public Booking(String venue, String date, String timeSlot, String bookedBy) {

if (venue == null || venue.isEmpty() ||

date == null || date.isEmpty() ||

timeSlot == null || timeSlot.isEmpty() ||

bookedBy == null || bookedBy.isEmpty()) {

throw new IllegalArgumentException("⚠️ Booking details cannot be empty!");

}

this.venue = venue;

this.date = date;

this.timeSlot = timeSlot;

this.bookedBy = bookedBy;

}

public String getVenue() {

return venue;

}

public String getDate() {

return date;

}

public String getTimeSlot() {

return timeSlot;

}

public String getBookedBy() {

return bookedBy;

}

public String toFileString() {

return venue + "," + date + "," + timeSlot + "," + bookedBy;

}

public static Booking fromFileString(String line) {

String[] parts = line.split(",");

if (parts.length == 4) {

return new Booking(parts[0].trim(), parts[1].trim(), parts[2].trim(), parts[3].trim());

}

return null;

}

public void displayBooking() {

System.out.println("\n📌 Booking Details:");

System.out.println("📍 Venue : " + venue);

System.out.println("📅 Date : " + date);

System.out.println("⏰ Time Slot : " + timeSlot);

System.out.println("👤 Booked By : " + bookedBy);

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

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj == null || getClass() != obj.getClass()) return false;

Booking booking = (Booking) obj;

return venue.equals(booking.venue) &&

date.equals(booking.date) &&

timeSlot.equals(booking.timeSlot) &&

bookedBy.equals(booking.bookedBy);

}

@Override

public int hashCode() {

return Objects.hash(venue, date, timeSlot, bookedBy);

}

}

**FileHandlerjava**

import java.io.\*;

import java.util.\*;

public class FileHandler {

private final String filePath;

public FileHandler(String filePath) {

this.filePath = filePath;

}

public List<String> readFile() {

List<String> lines = new ArrayList<>();

File file = new File(filePath);

if (!file.exists()) {

System.out.println("📂 File not found: " + filePath);

return lines;

}

try (BufferedReader reader = new BufferedReader(new FileReader(file))) {

String line;

while ((line = reader.readLine()) != null) {

lines.add(line);

}

System.out.println("✅ File read successfully!");

} catch (IOException e) {

System.err.println("⚠️ Error reading file: " + filePath);

e.printStackTrace();

}

return lines;

}

public void writeFile(List<String> lines) {

try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath))) {

for (String line : lines) {

writer.write(line);

writer.newLine();

}

System.out.println("✅ File written successfully!");

} catch (IOException e) {

System.err.println("⚠️ Error writing to file: " + filePath);

e.printStackTrace();

}

}

public void appendToFile(String line) {

try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath, true))) {

writer.write(line);

writer.newLine();

System.out.println("✅ Data appended successfully!");

} catch (IOException e) {

System.err.println("⚠️ Error appending to file: " + filePath);

e.printStackTrace();

}

}

}

**MainApp.java**

import java.util.Scanner;

public class MainApp {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("👤 Enter Username: ");

String loggedInUser = scanner.nextLine().trim();

while (loggedInUser.isEmpty()) {

System.out.print("⚠️ Username cannot be empty. Please enter again: ");

loggedInUser = scanner.nextLine().trim();

}

VenueBookingSystem app = new VenueBookingSystem(loggedInUser);

app.setVisible(true);

scanner.close();

}

}