```java
package util;

import model.*;

import java.util.*;

public class TimetableGenerator {
  private List<Course> courses;
  private List<Classroom> classrooms;
  private List<User> instructors;
  private Map<TimeSlot, ScheduledClass> timetable;

  public TimetableGenerator(List<Course> courses, List<Classroom> classrooms, List<User> instructors) {
    this.courses = courses;
    this.classrooms = classrooms;
    this.instructors = instructors;
    this.timetable = new HashMap<>();
  }

  public Map<TimeSlot, ScheduledClass> generate() {
    String[] days = { "Mon", "Tue", "Wed", "Thu", "Fri" };
    int periodsPerDay = 8;

    Random rand = new Random();

    for (Course course : courses) {
      for (int c = 0; c < course.getCredits(); c++) {
        boolean scheduled = false;

        for (int attempt = 0; attempt < 100 && !scheduled; attempt++) {
          String day = days[rand.nextInt(days.length)];
          int period = rand.nextInt(periodsPerDay) + 1;

          TimeSlot slot = new TimeSlot(day, period);

          boolean instructorFree = true;
          boolean classroomFree = true;

          // Find an available classroom
          Classroom availableRoom = null;
          for (Classroom cl : classrooms) {
            boolean isTaken = false;
            for (ScheduledClass sc : timetable.values()) {
              if (sc.getClassroom().getName().equals(cl.getName()) &&
                  sc.getTimeSlot().equals(slot)) {
                isTaken = true;
                break;
              }
            }
            if (!isTaken) {
              availableRoom = cl;
              break;
            }
          }

          // Check instructor conflict
          for (ScheduledClass sc : timetable.values()) {
            if (sc.getInstructor().getUsername().equals(course.getIC().getUsername()) &&
                sc.getTimeSlot().equals(slot)) {
              instructorFree = false;
              break;
            }
          }

          if (availableRoom != null && instructorFree) {
            ScheduledClass newClass = new ScheduledClass(course, course.getIC(), availableRoom, slot);
            timetable.put(slot, newClass);
            scheduled = true;
          }
        }
      }
    }

    return timetable;
  }

  public void printTimetable() {
    for (Map.Entry<TimeSlot, ScheduledClass> entry : timetable.entrySet()) {
      TimeSlot slot = entry.getKey();
      ScheduledClass sc = entry.getValue();
      System.out.println(slot.getDay() + " Period " + slot.getPeriod() + ": " +
          sc.getCourse().getCode() + " by " +
          sc.getInstructor().getUsername() + " in " +
          sc.getClassroom().getName());
    }
  }
}
```

```
package db;

public class TestConnection {
  public static void main(String[] args) {
    try {
      Class.forName("com.mysql.cj.jdbc.Driver");
      java.sql.Connection conn = DBConnection.getConnection();
      System.out.println("✅ Connected successfully to MySQL!");
      conn.close();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}

package util;

import model.ScheduledClass;

import java.util.*;

public class ConflictChecker {

  public static boolean hasConflicts(List<ScheduledClass> classes) {
    Set<String> instructorSlots = new HashSet<>();
    Set<String> roomSlots = new HashSet<>();
    Set<String> courseConflicts = new HashSet<>();

    for (ScheduledClass cls : classes) {
      String instKey = cls.getInstructor().getUsername() + "_" + cls.getTimeSlot().getDay() + "_" + cls.getTimeSlot().getPeriod();
      String roomKey = cls.getClassroom().getName() + "_" + cls.getTimeSlot().getDay() + "_" + cls.getTimeSlot().getPeriod();
      String courseKey = cls.getCourse().getCode() + "_" + cls.getTimeSlot().getDay() + "_" + cls.getTimeSlot().getPeriod();

      if (!instructorSlots.add(instKey) || !roomSlots.add(roomKey) || !courseConflicts.add(courseKey)) {
        return true;
      }
    }
    return false;
  }
}


package ui;

import javax.swing.*;
import java.awt.*;

public class TeacherPanel extends JPanel {
  public TeacherPanel(model.User user) {
    setLayout(new BorderLayout());
    JLabel label = new JLabel("Teacher Dashboard", SwingConstants.CENTER);
    label.setFont(new Font("Arial", Font.BOLD, 20));
    add(label, BorderLayout.NORTH);

    JTabbedPane tabbedPane = new JTabbedPane();

    tabbedPane.addTab("My Schedule", new ui.panels.TeacherTimetablePanel(user.getUsername()));
    tabbedPane.addTab("My Profile", new ui.panels.ProfilePanel(user));

    add(tabbedPane, BorderLayout.CENTER);
  }
}


package ui;

import javax.swing.*;
import java.awt.*;

public class StudentPanel extends JPanel {
  public StudentPanel(model.User user) {
    setLayout(new BorderLayout());
    JLabel label = new JLabel("Student Dashboard", SwingConstants.CENTER);
    label.setFont(new Font("Arial", Font.BOLD, 20));
    add(label, BorderLayout.NORTH);

    JTabbedPane tabbedPane = new JTabbedPane();

    tabbedPane.addTab("My Timetable", new ui.panels.StudentTimetablePanel(user.getUsername()));
    tabbedPane.addTab("Available Courses", new ui.panels.AvailableCoursesPanel());
    tabbedPane.addTab("My Profile", new ui.panels.ProfilePanel(user));

    add(tabbedPane, BorderLayout.CENTER);
  }
}
```

```java
package ui;

import javax.swing.*;

import db.DBConnection;

import java.awt.*;
import java.sql.Connection;
import java.sql.PreparedStatement;

public class RegisterFrame extends JFrame {
  public RegisterFrame() {
    setTitle("Register User");
    setSize(800, 500);
    setLocationRelativeTo(null);
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    setLayout(new BorderLayout());

    JLabel titleLabel = new JLabel("Register New User", SwingConstants.CENTER);
    titleLabel.setFont(new Font("Arial", Font.BOLD, 18));
    add(titleLabel, BorderLayout.NORTH);

    // Form components
    JPanel formPanel = new JPanel(new GridLayout(10, 1, 5, 5));

    JTextField nameField = new JTextField();
    JTextField usernameField = new JTextField();
    JPasswordField passwordField = new JPasswordField();
    JTextField emailField = new JTextField();
    JTextField departmentField = new JTextField();

    // Role Selection
    JRadioButton adminBtn = new JRadioButton("Admin");
    JRadioButton teacherBtn = new JRadioButton("Teacher");
    JRadioButton studentBtn = new JRadioButton("Student");
    ButtonGroup group = new ButtonGroup();
    group.add(adminBtn);
    group.add(teacherBtn);
    group.add(studentBtn);
    studentBtn.setSelected(true); // default

    JPanel rolePanel = new JPanel(new FlowLayout());
    rolePanel.add(adminBtn);
    rolePanel.add(teacherBtn);
    rolePanel.add(studentBtn);

    // Add to form
    formPanel.add(new JLabel("Full Name:"));
    formPanel.add(nameField);
    formPanel.add(new JLabel("Username:"));
    formPanel.add(usernameField);
    formPanel.add(new JLabel("Password:"));
    formPanel.add(passwordField);
    formPanel.add(new JLabel("Email:"));
    formPanel.add(emailField);
    formPanel.add(new JLabel("Department:"));
    formPanel.add(departmentField);
    formPanel.add(new JLabel("Select Role:"));
    formPanel.add(rolePanel);

    JPanel centerWrapper = new JPanel(new FlowLayout(FlowLayout.CENTER, 0, 20));
    centerWrapper.add(formPanel);
    add(centerWrapper, BorderLayout.CENTER);

    // Register button logic
    JButton registerBtn = new JButton("Register");
    registerBtn.addActionListener(e -> {
      String name = nameField.getText();
      String username = usernameField.getText();
      String password = new String(passwordField.getPassword());
      String email = emailField.getText();
      String dept = departmentField.getText();
      String role = adminBtn.isSelected() ? "Admin" : teacherBtn.isSelected() ? "Teacher" : "Student";

      if (name.isEmpty() || username.isEmpty() || password.isEmpty() || email.isEmpty() || dept.isEmpty()) {
        JOptionPane.showMessageDialog(this, "Please fill in all fields.", "Error", JOptionPane.ERROR_MESSAGE);
        return;
      }

      // Insert to DB
      try (Connection conn = DBConnection.getConnection()) {
        String sql = "INSERT INTO users (name, username, password, email, department, role) VALUES (?, ?, ?, ?, ?,?)";
        PreparedStatement stmt = conn.prepareStatement(sql);
        stmt.setString(1, name);
        stmt.setString(2, username);
```

```
        stmt.setString(3, password);
        stmt.setString(4, email);
        stmt.setString(5, dept);
        stmt.setString(6, role);
        stmt.executeUpdate();

        JOptionPane.showMessageDialog(this, "Registered successfully!");
        new LoginFrame();
        dispose();

      } catch (Exception ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage());
      }

    });

    add(registerBtn, BorderLayout.SOUTH);
    setVisible(true);
  }
}


package ui;

import javax.swing.*;
import java.awt.*;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class LoginFrame extends JFrame {
  private model.User authenticateUser(String username, String password, String role) {
    try (Connection conn = db.DBConnection.getConnection()) {
      String sql = "SELECT * FROM users WHERE username = ? AND password = ? AND role = ?";
      PreparedStatement stmt = conn.prepareStatement(sql);
      stmt.setString(1, username);
      stmt.setString(2, password);
      stmt.setString(3, role);

      ResultSet rs = stmt.executeQuery();
      if (rs.next()) {
        String name = rs.getString("username");
        String uname = rs.getString("username");
        String pass = rs.getString("password");
        String mail = rs.getString("email");
        String dep = rs.getString("department");
        String rol = rs.getString("role");
        return new model.User(name, uname, pass, mail, dep, rol);
      }
    } catch (Exception e) {
      e.printStackTrace();
      JOptionPane.showMessageDialog(this, "Database error: " + e.getMessage());
    }
    return null;
  }

  public LoginFrame() {
    setTitle("Time Table Builder - Login");
    setSize(800, 450);
    setLocationRelativeTo(null);
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    setLayout(new BorderLayout());

    JLabel titleLabel = new JLabel("Login", SwingConstants.CENTER);
    titleLabel.setFont(new Font("Arial", Font.BOLD, 18));
    add(titleLabel, BorderLayout.NORTH);

    // --- Center Panel: Login Form ---
    JPanel formPanel = new JPanel(new GridLayout(5, 1, 5, 5));
    JTextField usernameField = new JTextField();
    JPasswordField passwordField = new JPasswordField();

    // Role Selection
    JRadioButton adminBtn = new JRadioButton("Admin");
    JRadioButton teacherBtn = new JRadioButton("Teacher");
    JRadioButton studentBtn = new JRadioButton("Student");

    ButtonGroup group = new ButtonGroup();
    group.add(adminBtn);
    group.add(teacherBtn);
    group.add(studentBtn);
    adminBtn.setSelected(true);

    JPanel radioPanel = new JPanel(new FlowLayout());
    radioPanel.add(adminBtn);
    radioPanel.add(teacherBtn);
```

```
    radioPanel.add(studentBtn);

    formPanel.add(new JLabel("Username:"));
    formPanel.add(usernameField);
    formPanel.add(new JLabel("Password:"));
    formPanel.add(passwordField);
    formPanel.add(radioPanel);

    // adding formPanel
    formPanel.setPreferredSize(new Dimension(300, 200)); // or whatever size you want
    JPanel centerWrapper = new JPanel(new FlowLayout(FlowLayout.CENTER, 0, 30));
    centerWrapper.add(formPanel);
    add(centerWrapper, BorderLayout.CENTER);

    // --- Bottom Panel: Login + Register ---
    JPanel bottomPanel = new JPanel(new GridLayout(2, 1));

    JButton loginBtn = new JButton("Login");
    loginBtn.addActionListener(e -> {
      String selectedRole = adminBtn.isSelected() ? "Admin" : teacherBtn.isSelected() ? "Teacher" : "Student";
      String username = usernameField.getText();
      String password = new String(passwordField.getPassword());

      // Auth Logic
      model.User user = authenticateUser(username, password, selectedRole);
      if (user != null) {
        JOptionPane.showMessageDialog(this, "Login successful! Welcome " + user.getUsername());
        new DashboardFrame(user); // Or pass user object if DashboardFrame supports it
        dispose();
      } else {
        JOptionPane.showMessageDialog(this, "Invalid username/password");
        usernameField.setText("");
        passwordField.setText("");
      }

    });

    JPanel registerPanel = new JPanel(new FlowLayout());
    JLabel regLabel = new JLabel("New user?");
    JButton regBtn = new JButton("Register Here");
    regBtn.setBorderPainted(false);
    regBtn.setForeground(Color.BLUE);
    regBtn.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));

    regBtn.addActionListener(e -> {
      new RegisterFrame();
      dispose();
    });

    registerPanel.add(regLabel);
    registerPanel.add(regBtn);

    bottomPanel.add(loginBtn);
    bottomPanel.add(registerPanel);

    add(bottomPanel, BorderLayout.SOUTH);

    setVisible(true);
  }
}


package ui;

import javax.swing.*;
import java.awt.*;

public class DashboardFrame extends JFrame {
  public DashboardFrame(model.User user) {
    String role = user.getRole();
    setTitle(role + " Dashboard");
    setSize(800, 500);
    setLocationRelativeTo(null);
    setDefaultCloseOperation(EXIT_ON_CLOSE);

    // Create a container panel to hold dashboard + logout
    JPanel wrapperPanel = new JPanel(new BorderLayout());

    // Top bar with Logout Button
    JButton logoutButton = new JButton("Log Out");
    logoutButton.addActionListener(e -> {
      dispose(); // close this frame
      new LoginFrame(); // show login again
    });

    JPanel topBar = new JPanel(new FlowLayout(FlowLayout.RIGHT));
    topBar.add(logoutButton);
```

```
    // Load role-based content
    JPanel dashboardPanel = null;
    switch (role) {
      case "Admin":
        dashboardPanel = new AdminPanel(user);
        break;
      case "Teacher":
        dashboardPanel = new TeacherPanel(user);
        break;
      case "Student":
        dashboardPanel = new StudentPanel(user);
        break;
    }

    wrapperPanel.add(topBar, BorderLayout.NORTH);
    wrapperPanel.add(dashboardPanel, BorderLayout.CENTER);

    setContentPane(wrapperPanel);
    setVisible(true);
  }
}
```

```
package ui;

import javax.swing.*;

public class App {
  public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> new LoginFrame());
  }
}
```

```
package ui;

import javax.swing.*;
import java.awt.*;

public class AdminPanel extends JPanel {
  public AdminPanel(model.User user) {
    setLayout(new BorderLayout());
    JLabel label = new JLabel("Admin Dashboard", SwingConstants.CENTER);
    label.setFont(new Font("Arial", Font.BOLD, 20));
    add(label, BorderLayout.NORTH);

    JTabbedPane tabbedPane = new JTabbedPane();

    tabbedPane.addTab("Add Classroom", new ui.panels.ClassroomEntryPanel());
    tabbedPane.addTab("Add Course", new ui.panels.CourseEntryPanel());
    tabbedPane.addTab("Manual Timetable", new ui.panels.ManualTimetablePanel());
    tabbedPane.addTab("Courses", new ui.panels.CourseListPanel());
    tabbedPane.addTab("Students", new ui.panels.StudentListPanel());
    tabbedPane.addTab("Teachers", new ui.panels.TeacherListPanel());

    add(tabbedPane, BorderLayout.CENTER);
  }
}
```

```
package model;

import java.util.*;

public class User {
  String name;
  String username;
  String password;
  String email;
  String department;
  String role;
  List<Course> enrolledCourses;

  public User(String name, String username, String password, String email, String department, String role) {
    this.name = name;
    this.username = username;
    this.password = password;
    this.email = email;
    this.department = department;
    this.role = role;
  }

  public String getName() {
    return name;
  }
```

```
  public String getUsername() {
    return username;
  }

  public String getPassword() {
    return password;
  }

  public String getEmail() {
    return email;
  }

  public String getDepartment() {
    return department;
  }

  public String getRole() {
    return role;
  }

  public List<Course> getEnrolledCourses() {
    return enrolledCourses;
  }

  public void setEnrolledCourses(List<Course> enrolledCourses) {
    this.enrolledCourses = enrolledCourses;
  }
}


package model;

public class TimeSlot {
  private String day;
  private int period;

  public TimeSlot(String day, int period) {
    this.day = day;
    this.period = period;
  }

  public String getDay() {
    return day;
  }

  public int getPeriod() {
    return period;
  }

  @Override
  public boolean equals(Object o) {
    if (this == o)
      return true;
    if (o == null || getClass() != o.getClass())
      return false;
    TimeSlot that = (TimeSlot) o;
    return period == that.period && day.equals(that.day);
  }

  @Override
  public int hashCode() {
    return day.hashCode() * 31 + period;
  }
}

package model;

public class ScheduledClass {
  private Course course;
  private User instructor;
  private Classroom classroom;
  private TimeSlot timeSlot;

  public ScheduledClass(Course course, User instructor, Classroom classroom, TimeSlot timeSlot) {
    this.course = course;
    this.instructor = instructor;
    this.classroom = classroom;
    this.timeSlot = timeSlot;
  }

  public Course getCourse() {
    return course;
  }

  public User getInstructor() {
    return instructor;
  }
```

```
  public Classroom getClassroom() {
    return classroom;
  }

  public TimeSlot getTimeSlot() {
    return timeSlot;
  }
}
```

```
package model;

import java.util.*;

public class Course {
  String code;
  String name;
  User IC;
  int credits;
  List<TimeSlot> timeSlots;

  public Course(String code, String name, User IC, List<User> instructors, int credits) {
    this.code = code;
    this.name = name;
    this.IC = IC;
    this.credits = credits;
  }

  public String getCode() {
    return code;
  }

  public String getName() {
    return name;
  }

  public List<TimeSlot> getTimeSlots() {
    return timeSlots;
  }

  public int getCredits() {
    return credits;
  }

  public User getIC() {
    return IC;
  }
}
```

```
package model;

public class Classroom {
  private int id;
  private String name;
  private boolean isLab;

  public Classroom(int id, String name, boolean isLab) {
    this.id = id;
    this.name = name;
    this.isLab = isLab;
  }

  public int getId() {
    return id;
  }

  public String getName() {
    return name;
  }

  public boolean isLab() {
    return isLab;
  }

}
```

```
package db;

import java.sql.*;

public class DBConnection {
  private static final String URL = "jdbc:mysql://localhost:3306/timetable_db?useSSL=false&serverTimezone=UTC";
  private static final String USER = "root"; // your DB user
  private static final String PASS = "Ojus@132"; // your DB password
```

```
  public static Connection getConnection() throws SQLException {
    return DriverManager.getConnection(URL, USER, PASS);
  }
}
```