**SOURCE CODES OF PHASE 2 END PROJECT**

**RAILWAY CROSSING STATUS**

**JAVA cods:**

**DatabaseConnection**

package com.simpli;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DatabaseConnection {

private static final String URL =

"jdbc:mysql://localhost:3306/railway\_crossing\_status";

private static final String USERNAME = "root";

private static final String PASSWORD = "Mani@$7120";

public static Connection getConnection() {

Connection connection = null;

try {

Class.forName("com.mysql.jdbc.Driver");

connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);

} catch (ClassNotFoundException | SQLException e) {

e.printStackTrace();

}

return connection;

}

}

**RailwayCrossing**

package com.simpli;

public class RailwayCrossing {

private int id;

private String name;

private String address;

private String landmark;

private String trainSchedule;

private String personInCharge;

private String status;

public RailwayCrossing() {

super();

// TODO Auto-generated constructor stub

}

public RailwayCrossing(int id, String name, String address, String

landmark, String trainSchedule,

String personInCharge, String status) {

super();

this.id = id;

this.name = name;

this.address = address;

this.landmark = landmark;

this.trainSchedule = trainSchedule;

this.personInCharge = personInCharge;

this.status = status;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getLandmark() {

return landmark;

}

public void setLandmark(String landmark) {

this.landmark = landmark;

}

public String getTrainSchedule() {

return trainSchedule;

}

public void setTrainSchedule(String trainSchedule) {

this.trainSchedule = trainSchedule;

}

public String getPersonInCharge() {

return personInCharge;

}

public void setPersonInCharge(String personInCharge) {

this.personInCharge = personInCharge;

}

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

public void setId(String string) {

// TODO Auto-generated method stub

}

}

================================================================================**RailwayCrossingDAO**

package com.simpli;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

public class RailwayCrossingDAO {

private Connection connection;

public RailwayCrossingDAO() {

// Initialize the database connection

connection = DatabaseConnection.getConnection();

}

// Fetch all values from the database

public List<RailwayCrossing> getAllCrossings() {

List<RailwayCrossing> crossings = new ArrayList<>();

try {

String query = "SELECT \* FROM railway\_crossing";

PreparedStatement statement = connection.prepareStatement(query);

ResultSet resultSet = statement.executeQuery();

while (resultSet.next()) {

RailwayCrossing crossing = new RailwayCrossing();

crossing.setId(resultSet.getInt("id"));

crossing.setName(resultSet.getString("name"));

crossing.setAddress(resultSet.getString("address"));

crossing.setLandmark(resultSet.getString("landmark"));

crossing.setTrainSchedule(resultSet.getString("train\_schedule"));

crossing.setPersonInCharge(resultSet.getString("person\_in\_charge"));

crossing.setStatus(resultSet.getString("status"));

crossings.add(crossing);

}

} catch (SQLException e) {

e.printStackTrace();

}

return crossings;

}

// Search values by id and display it from the database

public RailwayCrossing getCrossingById(int crossingId) {

RailwayCrossing crossing = null;

try {

String query = "SELECT \* FROM railway\_crossing WHERE id = ?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setInt(1, crossingId);

ResultSet resultSet = statement.executeQuery();

if (resultSet.next()) {

crossing = new RailwayCrossing();

crossing.setId(resultSet.getInt("id"));

crossing.setName(resultSet.getString("name"));

crossing.setAddress(resultSet.getString("address"));

crossing.setLandmark(resultSet.getString("landmark"));

crossing.setTrainSchedule(resultSet.getString("train\_schedule"));

crossing.setPersonInCharge(resultSet.getString("person\_in\_charge"));

crossing.setStatus(resultSet.getString("status"));

}

} catch (SQLException e) {

e.printStackTrace();

}

return crossing;

}

// update the values in the database

public void updateCrossing(RailwayCrossing crossing) {

try {

String query = "UPDATE railway\_crossing SET name=?, address=?,landmark=?, train\_schedule=?, person\_in\_charge=?, status=? WHERE id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, crossing.getName());

statement.setString(2, crossing.getAddress());

statement.setString(3, crossing.getLandmark());

statement.setString(4, crossing.getTrainSchedule());

statement.setString(5, crossing.getPersonInCharge());

statement.setString(6, crossing.getStatus());

statement.setInt(7, crossing.getId());

statement.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

// Delete Railway Crossing

public void deleteCrossing(int crossingId) {

try {

connection.setAutoCommit(false); // Start transaction

// Delete associated favorite crossings

String deleteFavoriteCrossingsQuery = "DELETE FROM favorite\_crossing WHERE railway\_crossing\_id = ?";

try (PreparedStatement deleteFavoriteCrossingsStatement =

connection

.prepareStatement(deleteFavoriteCrossingsQuery)) {

deleteFavoriteCrossingsStatement.setInt(1, crossingId);

deleteFavoriteCrossingsStatement.executeUpdate();

}

// Delete the railway crossing

String deleteRailwayCrossingQuery = "DELETE FROM railway\_crossing WHERE id = ?";

try (PreparedStatement deleteRailwayCrossingStatement = connection

.prepareStatement(deleteRailwayCrossingQuery)) {

deleteRailwayCrossingStatement.setInt(1, crossingId);

deleteRailwayCrossingStatement.executeUpdate();

}

connection.commit(); // Commit the transaction

connection.setAutoCommit(true); // Reset auto-commit to true

} catch (SQLException e) {

try {

connection.rollback(); // Rollback the transaction if an error occurs

} catch (SQLException rollbackException) {

rollbackException.printStackTrace();

}

e.printStackTrace();

}

}

// Add values to the database

public void addCrossing(RailwayCrossing crossing) {

try {

String query = "INSERT INTO railway\_crossing (name, address, landmark, train\_schedule, person\_in\_charge, status) VALUES (?, ?, ?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, crossing.getName());

statement.setString(2, crossing.getAddress());

statement.setString(3, crossing.getLandmark());

statement.setString(4, crossing.getTrainSchedule());

statement.setString(5, crossing.getPersonInCharge());

statement.setString(6, crossing.getStatus());

statement.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

public List<RailwayCrossing> getFavoriteCrossings() {

List<RailwayCrossing> favoriteCrossings = new ArrayList<>();

try (Connection connection = DatabaseConnection.getConnection()) {

String query = "SELECT rc.\* FROM railway\_crossing rc "+ "JOIN favorite\_crossing fc ON rc.id = fc.railway\_crossing\_id";

PreparedStatement statement = connection.prepareStatement(query);

ResultSet resultSet = statement.executeQuery();

while (resultSet.next()) {

RailwayCrossing crossing = new RailwayCrossing();

crossing.setId(resultSet.getInt("id"));

crossing.setName(resultSet.getString("name"));

crossing.setAddress(resultSet.getString("address"));

crossing.setLandmark(resultSet.getString("landmark"));

crossing.setTrainSchedule(resultSet.getString("train\_schedule"));

crossing.setPersonInCharge(resultSet.getString("person\_in\_charge"));

crossing.setStatus(resultSet.getString("status"));

favoriteCrossings.add(crossing);

}

} catch (SQLException e) {

e.printStackTrace();

}

return favoriteCrossings;

}

public void addToFavorites(int crossingId) {

try (Connection connection = DatabaseConnection.getConnection()) {

String sql = "INSERT INTO favorite\_crossing (railway\_crossing\_id)VALUES (?)";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setInt(1, crossingId);

statement.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

// Handle the exception as needed

}

}

public void removeFromFavorites(int crossingId) {

try (Connection connection = DatabaseConnection.getConnection()) {

String sql = "DELETE FROM favorite\_crossing WHERE railway\_crossing\_id = ?";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setInt(1, crossingId);

statement.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

// Handle the exception as needed

}

}

}

User

package com.simpli;

public class User {

private String name;

private String email;

private String password;

public User() {

super();

}

public User(String name, String email, String password) {

this.name = name;

this.email = email;

this.password = password;

}

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;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

UserDAO

package com.simpli;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

public class UserDAO {

private Connection connection;

public UserDAO() {

// Initialize the database connection

connection = DatabaseConnection.getConnection();

}

public void registerUser(User user) {

try {

String query = "INSERT INTO user\_signup (name, email, password) VALUES (?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, user.getName());

statement.setString(2, user.getEmail());

statement.setString(3, user.getPassword());

statement.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

public boolean loginUser(String email, String password) {

try {

String query = "SELECT \* FROM user\_signup WHERE email=? AND password=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, email);

statement.setString(2, password);

ResultSet resultSet = statement.executeQuery();

return resultSet.next();

} catch (SQLException e) {

e.printStackTrace();

}

return false;

}

public User getUserByEmail(String email) {

try {

String query = "SELECT \* FROM user\_signup WHERE email=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, email);

ResultSet resultSet = statement.executeQuery();

if (resultSet.next()) {

User user = new User();

user.setName(resultSet.getString("name"));

user.setEmail(resultSet.getString("email"));

user.setPassword(resultSet.getString("password"));

return user;

}

} catch (SQLException e) {

e.printStackTrace();

}

return null;

}

}