import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.\*;

public class QuizTechie extends JFrame implements ActionListener {

private JComboBox<String> tableComboBox;

private JButton insertButton, updateButton, deleteButton;

public QuizTechie() {

super("Quiz Techie");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Create the menu bar

// Create the menu items for each table

// Set layout for the main content pane

setLayout(new FlowLayout());

// Create the combo box, buttons, and add them to the content pane

tableComboBox = new JComboBox<>();

tableComboBox.addItem("quiz");

tableComboBox.addItem("users");

tableComboBox.addItem("questions");

tableComboBox.addItem("score");

add(tableComboBox);

insertButton = new JButton("Submit");

updateButton = new JButton("Modify");

deleteButton = new JButton("Delete");

insertButton.addActionListener(this);

updateButton.addActionListener(this);

deleteButton.addActionListener(this);

add(insertButton);

add(updateButton);

add(deleteButton);

setSize(600, 400);

setLocationRelativeTo(null);

setResizable(true);

}

public void actionPerformed(ActionEvent e) {

String command = e.getActionCommand();

if (command.equals("quiz") || command.equals("users") || command.equals("questions") || command.equals("score")) {

tableComboBox.setSelectedItem(command);

} else if (e.getSource() == insertButton) {

performInsert();

} else if (e.getSource() == updateButton) {

performUpdate();

} else if (e.getSource() == deleteButton) {

performDelete();

}

}

private void performInsert() {

String tableName = (String) tableComboBox.getSelectedItem();

switch (tableName) {

case "quiz":

performCreateQuiz();

break;

case "users":

performCreateUser();

break;

case "questions":

performCreateQuestion();

break;

case "score":

performCreateScore();

break;

}

}

private void performUpdate() {

String tableName = (String) tableComboBox.getSelectedItem();

switch (tableName) {

case "quiz":

performUpdateQuiz();

break;

case "users":

performUpdateUser();

break;

case "questions":

performUpdateQuestion();

break;

case "score":

performUpdateScore();

break;

}

}

private void performDelete() {

String tableName = (String) tableComboBox.getSelectedItem();

switch (tableName) {

case "quiz":

performDeleteQuiz();

break;

case "users":

performDeleteUser();

break;

case "questions":

performDeleteQuestion();

break;

case "score":

performDeleteScore();

break;

}

}

// Rest of the code...

private void performCreateQuiz() {

String quizId=JOptionPane.showInputDialog(null, "Enter quiz id:");

String quizName = JOptionPane.showInputDialog(null, "Enter quiz name:");

createQuiz(quizId,quizName);

}

private void createQuiz(String quizId,String quizName) {

try {

Connection connection = getConnection();

int quizid=Integer.parseInt(quizId);

String sql = "INSERT INTO quiz (quiz\_id,quiz\_name) VALUES (?,?)";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setInt(1, quizid);

statement.setString(2, quizName);

statement.executeUpdate();

statement.close();

connection.close();

JOptionPane.showMessageDialog(null, "Quiz created successfully!");

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null, "Error: " + ex.getMessage());

}

}

private void performCreateUser() {

String userID = JOptionPane.showInputDialog(null, "Enter user id:");

String userName = JOptionPane.showInputDialog(null, "Enter user name:");

String password = JOptionPane.showInputDialog(null, "Enter password:");

createUser(userID,userName, password);

}

private void createUser(String userID,String userName, String password) {

try {

Connection connection = getConnection();

int userid=Integer.parseInt(userID);

String sql = "INSERT INTO users (user\_id,user\_name, password) VALUES (?,?, ?)";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setInt(1, userid);

statement.setString(2, userName);

statement.setString(3, password);

statement.executeUpdate();

statement.close();

connection.close();

JOptionPane.showMessageDialog(null, "User created successfully!");

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null, "Error: " + ex.getMessage());

}

}

private void performCreateQuestion() {

int questionId = Integer.parseInt(JOptionPane.showInputDialog(null, "Enter the question ID:"));

String correctAnswers = JOptionPane.showInputDialog(null, "Enter the correct answer:");

int quizId= Integer.parseInt(JOptionPane.showInputDialog(null, "Enter the quiz id:"));

createQuestion(questionId,correctAnswers,quizId);

}

private void createQuestion(int questionId, String correctAnswers, int quizId) {

try {

Connection connection = getConnection();

String sql = "INSERT INTO questions (question\_id, correct\_answers,quiz\_id) VALUES (?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setInt(1, questionId);

statement.setString(2, correctAnswers);

statement.setInt(3, quizId);

statement.executeUpdate();

statement.close();

connection.close();

JOptionPane.showMessageDialog(null, "Question created successfully!");

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null, "Error: " + ex.getMessage());

}

}

private void performCreateScore() {

int quizId = Integer.parseInt(JOptionPane.showInputDialog(null, "Enter the quiz ID:"));

int userId = Integer.parseInt(JOptionPane.showInputDialog(null, "Enter the user ID:"));

int scoreId = Integer.parseInt(JOptionPane.showInputDialog(null, "Enter the score ID:"));

int totalScore = Integer.parseInt(JOptionPane.showInputDialog(null, "Enter the total score:"));

createScore(quizId, userId, scoreId, totalScore);

}

private void createScore(int quizId, int userId, int scoreId, int totalScore) {

try {

Connection connection = getConnection();

String sql = "INSERT INTO score (quiz\_id, user\_id,score\_id, total\_score) VALUES (?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setInt(1, quizId);

statement.setInt(2, userId);

statement.setInt(3, scoreId);

statement.setInt(4, totalScore);

statement.executeUpdate();

statement.close();

connection.close();

JOptionPane.showMessageDialog(null, "Score created successfully!");

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null, "Error: " + ex.getMessage());

}

}

private void performUpdateQuiz() {

int quizId = Integer.parseInt(JOptionPane.showInputDialog(null, "Enter the quiz ID:"));

String quizName = JOptionPane.showInputDialog(null, "Enter the new quiz name:");

updateQuiz(quizId, quizName);

}

private void updateQuiz(int quizId, String quizName) {

try {

Connection connection = getConnection();

String sql = "UPDATE quiz SET quiz\_name = ? WHERE quiz\_id = ?";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, quizName);

statement.setInt(2, quizId);

statement.executeUpdate();

statement.close();

connection.close();

JOptionPane.showMessageDialog(null, "Quiz updated successfully!");

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null, "Error: " + ex.getMessage());

}

}

private void performUpdateUser() {

int userId = Integer.parseInt(JOptionPane.showInputDialog(null, "Enter the user ID:"));

String userName = JOptionPane.showInputDialog(null, "Enter the new user name:");

String password = JOptionPane.showInputDialog(null, "Enter the new password:");

updateUser(userId, userName, password);

}

private void updateUser(int userId, String userName, String password) {

try {

Connection connection = getConnection();

String sql = "UPDATE users SET user\_name = ?, password = ? WHERE user\_id = ?";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, userName);

statement.setString(2, password);

statement.setInt(3, userId);

statement.executeUpdate();

statement.close();

connection.close();

JOptionPane.showMessageDialog(null, "User updated successfully!");

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null, "Error: " + ex.getMessage());

}

}

private void performUpdateQuestion() {

int questionId = Integer.parseInt(JOptionPane.showInputDialog(null, "Enter the question ID:"));

String correctAnswer = JOptionPane.showInputDialog(null, "Enter the new correct answer:");

int quizId = Integer.parseInt(JOptionPane.showInputDialog(null, "Enter the quiz ID:"));

updateQuestion(questionId, correctAnswer,quizId);

}

private void updateQuestion(int questionId, String correctAnswer,int quizId) {

try {

Connection connection = getConnection();

String sql = "UPDATE questions SET correct\_answers = ?,quiz\_id=? WHERE question\_id = ?";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, correctAnswer);

statement.setInt(2,quizId);

statement.setInt(3, questionId);

statement.executeUpdate();

statement.close();

connection.close();

JOptionPane.showMessageDialog(null, "Question updated successfully!");

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null, "Error: " + ex.getMessage());

}

}

private void performUpdateScore() {

int scoreId = Integer.parseInt(JOptionPane.showInputDialog(null, "Enter the score ID:"));

int totalScore = Integer.parseInt(JOptionPane.showInputDialog(null, "Enter the new total score:"));

updateScore(scoreId, totalScore);

}

private void updateScore(int scoreId, int totalScore) {

try {

Connection connection = getConnection();

String sql = "UPDATE score SET total\_score = ? WHERE score\_id = ?";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setInt(1, totalScore);

statement.setInt(2, scoreId);

statement.executeUpdate();

statement.close();

connection.close();

JOptionPane.showMessageDialog(null, "Score updated successfully!");

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null, "Error: " + ex.getMessage());

}

}

private void performDeleteQuiz() {

int quizId = Integer.parseInt(JOptionPane.showInputDialog(null, "Enter the quiz ID:"));

deleteQuiz(quizId);

}

private void deleteQuiz(int quizId) {

try {

Connection connection = getConnection();

String sql = "DELETE FROM quiz WHERE quiz\_id = ?";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setInt(1, quizId);

statement.executeUpdate();

statement.close();

connection.close();

JOptionPane.showMessageDialog(null, "Quiz deleted successfully!");

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null, "Error: " + ex.getMessage());

}

}

private void performDeleteUser() {

int userId = Integer.parseInt(JOptionPane.showInputDialog(null, "Enter the user ID:"));

deleteUser(userId);

}

private void deleteUser(int userId) {

try {

Connection connection = getConnection();

String sql = "DELETE FROM users WHERE user\_id = ?";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setInt(1, userId);

statement.executeUpdate();

statement.close();

connection.close();

JOptionPane.showMessageDialog(null, "User deleted successfully!");

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null, "Error: " + ex.getMessage());

}

}

private void performDeleteQuestion() {

int questionId = Integer.parseInt(JOptionPane.showInputDialog(null, "Enter the question ID:"));

deleteQuestion(questionId);

}

private void deleteQuestion(int questionId) {

try {

Connection connection = getConnection();

String sql = "DELETE FROM questions WHERE question\_id = ?";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setInt(1, questionId);

statement.executeUpdate();

statement.close();

connection.close();

JOptionPane.showMessageDialog(null, "Question deleted successfully!");

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null, "Error: " + ex.getMessage());

}

}

private void performDeleteScore() {

int scoreId = Integer.parseInt(JOptionPane.showInputDialog(null, "Enter the score ID:"));

deleteScore(scoreId);

}

private void deleteScore(int scoreId) {

try {

Connection connection = getConnection();

String sql = "DELETE FROM score WHERE score\_id = ?";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setInt(1, scoreId);

statement.executeUpdate();

statement.close();

connection.close();

JOptionPane.showMessageDialog(null, "Score deleted successfully!");

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null, "Error: " + ex.getMessage());

}

}

private Connection getConnection() throws SQLException {

String url = "jdbc:oracle:thin:@localhost:1521:xe";

String username = "arjun";

String password = "komal";

return DriverManager.getConnection(url, username, password);

}

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

public void run() {

QuizTechie quizTechie = new QuizTechie();

quizTechie.setVisible(true);

}

});

}

}