Package dao;

Import java.sql.Connection;

Import java.sql.DriverManager;

Import java.sql.PreparedStatement;

Import java.sql.ResultSet;

Import java.sql.SQLException;

Import java.util.ArrayList;

Import java.util.List;

Import model.Question;

Import model.Quiz;

Public class QuizDAO {

Private String jdbcURL = “jdbc:mysql://localhost:3306/quizdb”;

Private String jdbcUserName = “root”;

Private String jdbcPassword = “root”;

// SQL Queries

Private static final String INSERT\_QUIZ\_SQL = “INSERT INTO quizzes (title, description) VALUES (?, ?);”;

Private static final String INSERT\_QUESTION\_SQL = “INSERT INTO questions (quiz\_id, question\_text, option\_a, option\_b, option\_c, option\_d, correct\_answer) VALUES (?, ?, ?, ?, ?, ?, ?);”;

Private static final String SELECT\_QUIZ\_BY\_ID = “SELECT \* FROM quizzes WHERE id = ?;”;

Private static final String SELECT\_QUESTIONS\_BY\_QUIZ\_ID = “SELECT \* FROM questions WHERE quiz\_id = ?;”;

Private static final String SELECT\_ALL\_QUIZZES = “SELECT \* FROM quizzes;”;

Private static final String DELETE\_QUIZ\_SQL = “DELETE FROM quizzes WHERE id = ?;”;

Private static final String DELETE\_QUESTIONS\_BY\_QUIZ\_ID = “DELETE FROM questions WHERE quiz\_id = ?;”;

Private static final String UPDATE\_QUIZ\_SQL = “UPDATE quizzes SET title = ?, description = ? WHERE id = ?;”;

Public QuizDAO() { }

// Database connection

Public Connection getConnection() {

Connection connection = null;

Try {

Class.forName(“com.mysql.cj.jdbc.Driver”);

Connection = DriverManager.getConnection(jdbcURL, jdbcUserName, jdbcPassword);

} catch (SQLException | ClassNotFoundException e) {

e.printStackTrace();

}

Return connection;

}

// Insert a new quiz

Public void insertQuiz(Quiz quiz) {

Try (Connection connection = getConnection();

PreparedStatement preparedStatement = connection.prepareStatement(INSERT\_QUIZ\_SQL)) {

preparedStatement.setString(1, quiz.getTitle());

preparedStatement.setString(2, quiz.getDescription());

preparedStatement.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

// Insert a new question for a quiz

Public void insertQuestion(Question question, int quizId) {

Try (Connection connection = getConnection();

PreparedStatement preparedStatement = connection.prepareStatement(INSERT\_QUESTION\_SQL)) {

preparedStatement.setInt(1, quizId);

preparedStatement.setString(2, question.getQuestionText());

preparedStatement.setString(3, question.getOptionA());

preparedStatement.setString(4, question.getOptionB());

preparedStatement.setString(5, question.getOptionC());

preparedStatement.setString(6, question.getOptionD());

preparedStatement.setString(7, question.getCorrectAnswer());

preparedStatement.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

// Retrieve a quiz by ID

Public Quiz getQuizById(int id) {

Quiz quiz = null;

Try (Connection connection = getConnection();

PreparedStatement preparedStatement = connection.prepareStatement(SELECT\_QUIZ\_BY\_ID)) {

preparedStatement.setInt(1, id);

ResultSet resultSet = preparedStatement.executeQuery();

If (resultSet.next()) {

Quiz = new Quiz(resultSet.getInt(“id”), resultSet.getString(“title”), resultSet.getString(“description”));

Quiz.setQuestions(getQuestionsByQuizId(id));

}

} catch (SQLException e) {

e.printStackTrace();

}

Return quiz;

}

// Retrieve all quizzes

Public List<Quiz> getAllQuizzes() {

List<Quiz> quizzes = new ArrayList<>();

Try (Connection connection = getConnection();

PreparedStatement preparedStatement = connection.prepareStatement(SELECT\_ALL\_QUIZZES);

ResultSet resultSet = preparedStatement.executeQuery()) {

While (resultSet.next()) {

Int id = resultSet.getInt(“id”);

String title = resultSet.getString(“title”);

String description = resultSet.getString(“description”);

Quizzes.add(new Quiz(id, title, description));

}

} catch (SQLException e) {

e.printStackTrace();

}

Return quizzes;

}

// Retrieve all questions for a specific quiz

Public List<Question> getQuestionsByQuizId(int quizId) {

List<Question> questions = new ArrayList<>();

Try (Connection connection = getConnection();

PreparedStatement preparedStatement = connection.prepareStatement(SELECT\_QUESTIONS\_BY\_QUIZ\_ID)) {

preparedStatement.setInt(1, quizId);

ResultSet resultSet = preparedStatement.executeQuery();

While (resultSet.next()) {

Int id = resultSet.getInt(“id”);

String questionText = resultSet.getString(“question\_text”);

String optionA = resultSet.getString(“option\_a”);

String optionB = resultSet.getString(“option\_b”);

String optionC = resultSet.getString(“option\_c”);

String optionD = resultSet.getString(“option\_d”);

String correctAnswer = resultSet.getString(“correct\_answer”);

Questions.add(new Question(id, questionText, optionA, optionB, optionC, optionD, correctAnswer));

}

} catch (SQLException e) {

e.printStackTrace();

}

Return questions;

}

// Update a quiz

Public boolean updateQuiz(Quiz quiz) {

Boolean updated = false;

Try (Connection connection = getConnection();

PreparedStatement preparedStatement = connection.prepareStatement(UPDATE\_QUIZ\_SQL)) {

preparedStatement.setString(1, quiz.getTitle());

preparedStatement.setString(2, quiz.getDescription());

preparedStatement.setInt(3, quiz.getId());

updated = preparedStatement.executeUpdate() > 0;

} catch (SQLException e) {

e.printStackTrace();

}

Return updated;

}

// Delete a quiz and its questions

Public boolean deleteQuiz(int id) {

Boolean deleted = false;

Try (Connection connection = getConnection()) {

// Delete questions for the quiz

PreparedStatement deleteQuestionsStmt = connection.prepareStatement(DELETE\_QUESTIONS\_BY\_QUIZ\_ID);

deleteQuestionsStmt.setInt(1, id);

deleteQuestionsStmt.executeUpdate();

// Delete the quiz

PreparedStatement deleteQuizStmt = connection.prepareStatement(DELETE\_QUIZ\_SQL);

deleteQuizStmt.setInt(1, id);

deleted = deleteQuizStmt.executeUpdate() > 0;

} catch (SQLException e) {

e.printStackTrace();

}

Return deleted;

}

}