PROJECT TITLE:
MATH QUIZ APPLICATION

VIRTUNEXA

PRESENTED BY:

BHAVANI ATKAPURAM

INTERN ID:

VN-JD-4W213

Introduction

The **Math Quiz Application** is an interactive tool designed to help users practice basic arithmetic operations, such as addition, subtraction, and multiplication, through a quiz-style format. The application generates random math problems based on the selected operation and provides immediate feedback on user responses. At the end of the quiz, users receive a score summary, offering a clear understanding of their performance.

This project aims to create an engaging and educational experience where users can improve their mathematical skills while having fun. The application is structured to track user scores and store them in a local SQLite database, allowing for easy retrieval and future analysis of quiz results.

Objective and Scope

The main objective of this project is to:

- Develop an interactive, user-friendly quiz application that generates random math problems.
- Provide a system for tracking and saving user scores for future reference.
- Offer immediate feedback on user answers to promote learning and improvement.
- Ensure robust error handling and validation for user inputs.

The scope of the project includes:

- Basic arithmetic operations such as addition, subtraction, and multiplication.
- A simple user interface (either GUI or command-line) to interact with the application.
- A backend database (SQLite) for storing user scores.

Technologies Used

The project leverages the following technologies:

- **Java**: Core programming language used for developing the application.

- **Swing**: Java framework used for creating the Graphical User Interface (GUI).
- **JDBC**: Java Database Connectivity to interact with the SQLite database.
- **SQLite**: Lightweight database used to store and manage user scores.
- **Random Class**: Used for generating random math questions.
- **Event Handling**: For capturing and responding to user actions (button clicks,text input)

System Requirements

1. Hardware Requirements

- **Processor**: A minimum of 1.5 GHz CPU or equivalent.
- **RAM**: At least 2 GB of RAM (4 GB recommended for optimal performance).
- **Storage**: Minimum 100 MB of free disk space for the application and database files.
- **Display**: A display capable of 1024x768 resolution for optimal graphical user interface (GUI) display.
- **Input Devices**: Keyboard and mouse (for interaction with the application).

2. Software Requirements

- **Operating System**:
  - Windows 10 or higher, macOS, or any Linux distribution.
- **Java Development Kit (JDK)**:
  - JDK 8 or higher is required to compile and run the Java-based application.
  - JDK should be installed and the JAVA_HOME environment variable should be set properly.
- **Database**:
  - **SQLite**: Required for database management. The application uses an SQLite database to store quiz scores. SQLite is a self-contained, serverless database, so no additional installation is required other than the database file.
- **IDE (Integrated Development Environment)** (Optional but recommended):
  - **Eclipse** or **IntelliJ IDEA** for Java development, which offers syntax highlighting, debugging tools, and project management features.

- **Additional Libraries/Tools**:
  - **JDBC**: For connecting the Java application with SQLite.
  - **Swing**: Java's built-in framework for GUI development, used to create the application's user interface.

3. Installation Prerequisites

- **Java Setup**:
  1. Download and install JDK (Java Development Kit) version 8 or higher.
  2. Set up the **JAVA_HOME** environment variable to point to the JDK directory.
  3. Add the JDK **bin** folder to the system's **PATH**.
- **SQLite Setup**:
  1. The application automatically uses the **SQLite JDBC** driver, but ensure the **sqlite-jdbc** library is included in your project's dependencies.
  2. The database file (quiz_scores.db) will be created automatically when the application is run for the first time.

4. Software Requirements for Deployment

- **Java Runtime Environment (JRE)**:
  - JRE 8 or higher installed on the machine to run the compiled Java program if the user is not developing.
- **Executable JAR File** (Optional for distribution):
  - If distributing the application as an executable JAR file, ensure that all dependencies (including JDBC and SQLite libraries) are bundled inside the JAR or included in the classpath.

5. Network Requirements (Optional)

- No specific network requirements are necessary for this local application. It functions offline with SQLite stored locally on the user's machine.
- If the application were to be expanded with features like online score sharing, an internet connection would be required.

Design and Architecture

1. **Architecture Pattern: MVC (Model-View-Controller)**:
   - **Model**: Handles quiz logic (score tracking, question generation) and database interactions.
   - **View**: GUI (Swing) displays math questions, receives user input, and provides feedback.
   - **Controller**: Manages user input, triggers quiz flow, validates answers, and updates the score.
2. **Core Components**:
   - **MathQuizApp (Main Class)**: Initializes and controls the app flow.
   - **User Interface**: Built using Swing components (JFrame, JTextArea, JTextField, JButton).
   - **Database Interaction**: Uses SQLite (via JDBC) to store and retrieve quiz scores.
3. **Quiz Flow**:
   - **Question Generation**: Random math problems (addition, subtraction, multiplication).
   - **User Input**: User enters the answer and receives feedback (correct/incorrect).
   - **Score Tracking**: Increments on correct answers, stored in a database after the quiz ends.
4. **Data Management**:
   - **SQLite Database**: Stores scores persistently in a scores table.
5. **Error Handling**:
   - **Input Validation**: Ensures valid answers and handles invalid input with error messages.
   - **Database Error Handling**: Catches and logs any errors during database interactions.

6. **Extendability**: The system is easily extendable to support more operations or features like a leaderboard.

Important Steps for Project Setup

1. **Install Java Development Kit (JDK)**:
   - Download and install JDK 8 or above from Oracle.
2. **Install Visual Studio Code (VSCode)**:
   - Download VSCode from here.
3. **Install Java Extension Pack in VSCode**:
   - Open VSCode.
   - Go to Extensions (Ctrl+Shift+X).
   - Search and install **Java Extension Pack** (includes essential Java extensions).
4. **Project Structure**: Create the following folder structure:

```
MathQuizApp/
├── src/
│   └── MathQuizApp.java
├── lib/
│   └── sqlite-jdbc-x.x.x.jar (SQLite JDBC Driver)
└── quiz_scores.db (SQLite Database)
```

5. **Create MathQuizApp.java**:
   - Inside the src/ folder, add the Java code for the quiz app.
6. **Add SQLite JDBC Driver**:
   - Download and add the SQLite JDBC jar file to the lib/ folder.
7. **Configure VSCode for Java**:
   - Ensure Java is properly set up by running java -version in the terminal.
8. **Run the Application**:
   - Compile and run the Java code

Functionalities and Features of the Math Quiz App

The **Math Quiz App** offers an interactive and educational experience for users to test and improve their math skills through a variety of arithmetic operations such as addition, subtraction, and multiplication. The app provides a dynamic user interface where the user can select their preferred operation and solve math problems.

Key functionalities include:

- **Dynamic Question Generation**: The app generates random math questions based on the selected operation (addition, subtraction, or multiplication) and displays them to the user.
- **Answer Validation**: Users input their answers, and the app checks if they are correct, providing immediate feedback on whether the answer is correct or incorrect.
- **Score Tracking**: The app keeps track of the user's score, updating it after every question, and displays the final score at the end of the quiz.
- **User Feedback**: After each question, the app gives feedback to the user with a message indicating if their answer was correct or wrong.
- **Database Integration**: The app stores the user's scores in an SQLite database, allowing for the retrieval and tracking of performance over time. This helps to maintain a history of quiz results.
- **Multiple Operations**: The app supports different types of math operations, and the user can choose between addition, subtraction, or multiplication to test their skills.
- **Interactive GUI**: A simple yet effective graphical user interface (GUI) built with Swing enables users to interact with the application seamlessly, providing an intuitive experience for selecting operations and submitting answers.

These functionalities combine to offer a comprehensive, interactive, and fun way for users to enhance their arithmetic skills while tracking their progress over time.

Source code

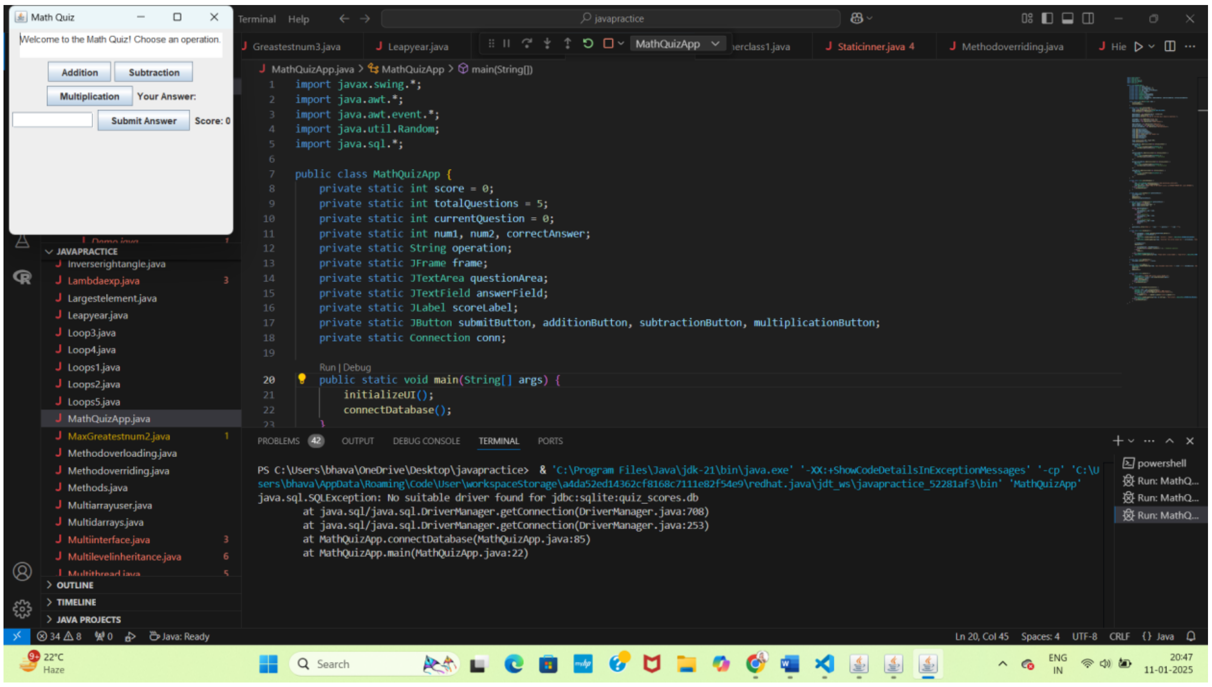
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.Random;
import java.sql.*;

public class MathQuizApp {
    private static int score = 0;
    private static int totalQuestions = 5;
    private static int currentQuestion = 0;
    private static int num1, num2, correctAnswer;
    private static String operation;
    private static JFrame frame;
    private static JTextArea questionArea;
    private static JTextField answerField;
    private static JLabel scoreLabel;
    private static JButton submitButton, additionButton, subtractionButton,
multiplicationButton;
    private static Connection conn;

    public static void main(String[] args) {
        initializeUI();
        connectDatabase();
    }

    private static void initializeUI() {
```

```
frame = new JFrame("Math Quiz");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setLayout(new FlowLayout());

questionArea = new JTextArea(2, 20);
questionArea.setEditable(false);
questionArea.setText("Welcome to the Math Quiz! Choose an operation.");

scoreLabel = new JLabel("Score: 0");
answerField = new JTextField(10);
submitButton = new JButton("Submit Answer");

additionButton = new JButton("Addition");
subtractionButton = new JButton("Subtraction");
multiplicationButton = new JButton("Multiplication");

frame.add(questionArea);
frame.add(additionButton);
frame.add(subtractionButton);
frame.add(multiplicationButton);
frame.add(new JLabel("Your Answer:"));
frame.add(answerField);
frame.add(submitButton);
frame.add(scoreLabel);

frame.setSize(300, 300);
frame.setVisible(true);

additionButton.addActionListener(new ActionListener() {
   @Override
```

```
        public void actionPerformed(ActionEvent e) {
            startQuiz(1); // Addition
        }
    });

    subtractionButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            startQuiz(2); // Subtraction
        }
    });

    multiplicationButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            startQuiz(3); // Multiplication
        }
    });

    submitButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            checkAnswer();
        }
    });
}

private static void connectDatabase() {
    try {
        conn = DriverManager.getConnection("jdbc:sqlite:quiz_scores.db");
```

```
            Statement stmt = conn.createStatement();
            stmt.execute("CREATE TABLE IF NOT EXISTS scores (id INTEGER PRIMARY
KEY, score INTEGER)");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private static void startQuiz(int operationChoice) {
        currentQuestion = 0;
        score = 0;
        updateScore();
        nextQuestion(operationChoice);
    }

    private static void nextQuestion(int operationChoice) {
        Random random = new Random();
        num1 = random.nextInt(100) + 1;
        num2 = random.nextInt(100) + 1;

        switch (operationChoice) {
            case 1: // Addition
                correctAnswer = num1 + num2;
                operation = "+";
                break;
            case 2: // Subtraction
                correctAnswer = num1 - num2;
                operation = "-";
                break;
            case 3: // Multiplication
                correctAnswer = num1 * num2;
                operation = "*";
                break;
        }

        questionArea.setText("What is " + num1 + " " + operation + " " + num2 + "?");
    }

    private static void checkAnswer() {
        try {
            int userAnswer = Integer.parseInt(answerField.getText());
            if (userAnswer == correctAnswer) {
                score++;
                JOptionPane.showMessageDialog(frame, "Correct!", "Answer",
JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(frame, "Incorrect! The correct answer was " +
correctAnswer, "Answer", JOptionPane.ERROR_MESSAGE);
            }

            currentQuestion++;
            updateScore();

            if (currentQuestion < totalQuestions) {
                nextQuestion(new Random().nextInt(3) + 1); // Randomize operation
            } else {
                endQuiz();
            }
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(frame, "Please enter a valid number.", "Input
Error", JOptionPane.ERROR_MESSAGE);
        }
    }

    private static void updateScore() {
        scoreLabel.setText("Score: " + score);
    }

    private static void endQuiz() {
        JOptionPane.showMessageDialog(frame, "Quiz Finished! Final Score: " + score + "/" +
totalQuestions, "Quiz Over", JOptionPane.INFORMATION_MESSAGE);
        saveScore();
        score = 0;
        updateScore();
    }

    private static void saveScore() {
        try {
            String query = "INSERT INTO scores (score) VALUES (?)";
            PreparedStatement pstmt = conn.prepareStatement(query);
            pstmt.setInt(1, score);
            pstmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public static void displayHistoricalScores() {
        try {
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM scores");
            StringBuilder sb = new StringBuilder("Historical Scores:\n");
            while (rs.next()) {
                sb.append("Score: ").append(rs.getInt("score")).append("\n");
            }
            JOptionPane.showMessageDialog(frame, sb.toString(), "Past Scores",
JOptionPane.INFORMATION_MESSAGE);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

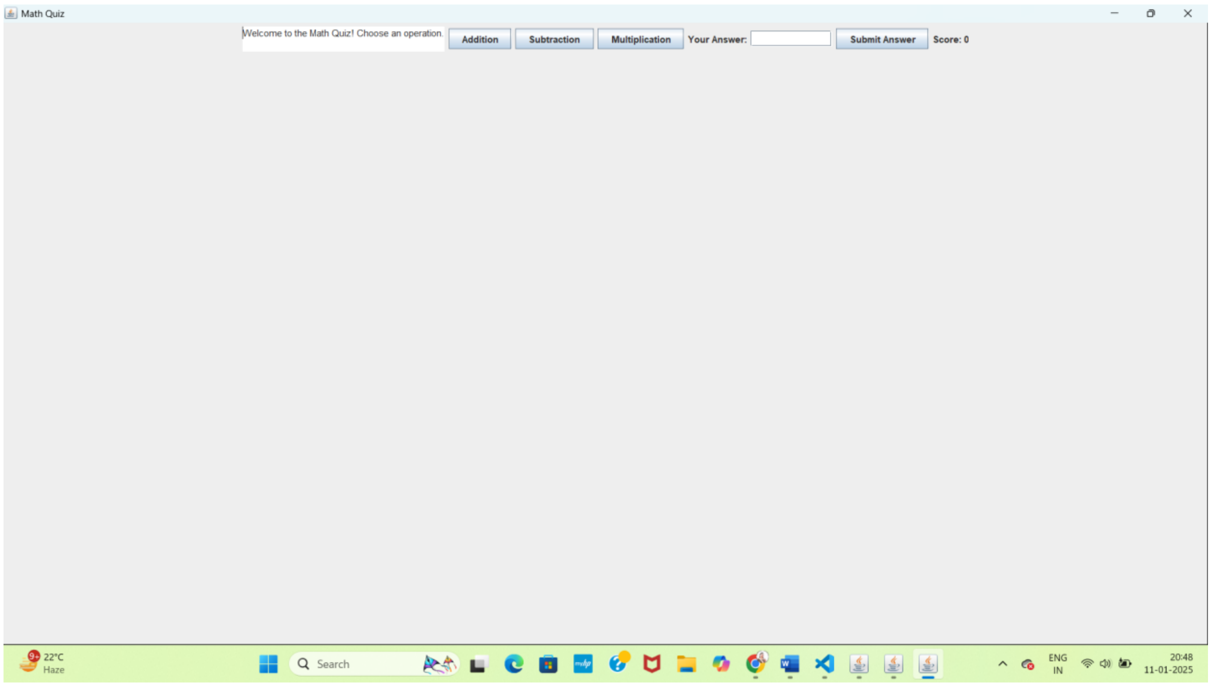
Result:



Welcome to the Math Quiz! Choose an operation.

```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.Random;
import java.sql.*;

public class MathQuizApp {
    private static int score = 0;
    private static int totalQuestions = 5;
    private static int currentQuestion = 0;
    private static int num1, num2, correctAnswer;
    private static String operation;
    private static JFrame frame;
    private static JTextArea questionArea;
    private static JTextField answerField;
    private static JLabel scoreLabel;
    private static JButton submitButton, additionButton, subtractionButton, multiplicationButton;
    private static Connection conn;

    public static void main(String[] args) {
        initializeUI();
        connectDatabase();
    }
```

```
PS C:\Users\bhava\OneDrive\Desktop\javapractice>  & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\U
sers\bhava\AppData\Roaming\Code\User\workspaceStorage\a4da52ed14362cf8168c7111e82f54e9\redhat.java\jdt_ws\javapractice_52281af3\bin' 'MathQuizApp'
java.sql.SQLException: No suitable driver found for jdbc:sqlite:quiz_scores.db
        at java.sql/java.sql.DriverManager.getConnection(DriverManager.java:708)
        at java.sql/java.sql.DriverManager.getConnection(DriverManager.java:253)
        at MathQuizApp.connectDatabase(MathQuizApp.java:85)
        at MathQuizApp.main(MathQuizApp.java:22)
```



Welcome to the Math Quiz! Choose an operation.

Future Enhancements

1. **User Profile and Authentication**:
    - Implement a user authentication system where users can create accounts, log in, and track their individual progress over time. This would allow users to view their quiz history, previous scores, and overall improvements.
2. **Advanced Operations**:
    - Expand the range of mathematical operations to include division, exponentiation, square roots, and more complex operations such as algebraic equations, trigonometry, and calculus problems for advanced users.
3. **Difficulty Levels**:
    - Introduce difficulty levels (easy, medium, hard) that adjust the range of numbers and the complexity of the questions. This will allow users to select a level based on their expertise and progressively improve their skills.
4. **Timed Mode**:
    - Add a timed quiz mode where users have a limited time to answer each question. This feature would challenge users to think quickly, adding an element of urgency to the quiz.
5. **Leaderboard**:
    - Create a global or local leaderboard where users can compare their scores with others. This would encourage friendly competition and motivate users to improve their performance.
6. **Question Customization**:
    - Allow users to customize the types of questions they want to answer, such as selecting specific operations or setting the number range. This customization would provide a more tailored experience.
7. **Multiplayer Mode**:

  - Implement a multiplayer feature where two or more users can play against each other in real-time, answering questions and competing to get the highest score.

Conclusion

The Math Quiz App is a versatile and interactive application designed to help users enhance their mathematical skills through engaging and challenging quizzes. By offering a variety of operations, customizable questions, and tracking features, the app provides a comprehensive platform for users to test and improve their abilities in arithmetic and other mathematical concepts.

Throughout the development process, key features such as a user-friendly graphical interface, real-time score tracking, and a database for storing user scores have been implemented. These features ensure that the app is not only functional but also intuitive and motivating for users. Additionally, by storing scores in a local SQLite database, users' progress can be tracked over time, providing valuable feedback on their performance.

The project lays the foundation for future improvements, including adding more advanced mathematical operations, difficulty levels, multiplayer modes, and even mobile versions. These enhancements would further enrich the user experience and make the app more suitable for a broader range of users, from beginners to advanced learners.

In conclusion, the Math Quiz App serves as an excellent educational tool, blending learning and fun while offering a seamless and interactive experience for users to sharpen their mathematical skills. The ongoing potential for enhancement ensures the app can continue to evolve and adapt to meet the growing needs of its users.

References:

1. https://docs.oracle.com/en/java/
2. https://docs.oracle.com/javase/8/docs/api/javax/swing/package-summary.html
3. https://www.geeksforgeeks.org/java/