

Department of Computer Science Engineering

**SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE**

**Project Report**

**for**

**JAVA MINI PROJECT**

**QUIZ APP**

**Submitted by:**

Aditi Goyal (003)

Sri Venkata Neelima Chinta (070)

Arundarasi Rajendran (081)

## **Table of Contents**

1. Introduction
   1. Context and purpose
   2. Audience
   3. Scope
2. Novelty of work
3. Requirements
   1. Software requirements
   2. Hardware requirements
4. Imports and Libraries used
   1. Java AWT
   2. Java Swing
   3. Java SQL
      1. Database and connections
5. Class diagram and flowchart
6. Code Snippets with logic and output
7. Timeline
8. Team Member contribution

**1. INTRODUCTION :**

**1.1 Context & purpose**

Through the ages our knowledge has been a very important aspect of growth and so has everyone been tested based on the natural equivalent of comparison i.e. information and so it becomes a compulsion to gather more information and this process soon becomes a serious thing. The software requirement specification of the following project is a detailed information about the first step toward the creation of a program that makes quizzing people a fun thing.

Topics that people relate to and have a good understanding can be chosen and people can have a nice time interacting with a program that questions their understanding about their favourite topic. As time passes the questions get more challenging and the player forgets the boredom he/she is surrounded with.

**1.2 Audience**

The quiz is for all the age groups, starting from school-going kids to retired males/females. The variety of topics provided makes it fun for everyone. The program is for users to test their knowledge in a specific field and so can be played by everyone.

**1.3 Scope**

Knowledge has no bounds, and so there’s no end to the number of topics or questions which can be added to the quiz. Also, the quiz can be translated in different languages depending on the country and the audience. The purpose of this game is simply for entertainment purposes where ones that enjoy taking general knowledge quizzes in their leisure times can get their friends and play this game.

**2. NOVELTY OF WORK:**

The idea has remained similar to what was decided initially. The project has quizzes and a leader board that shows the scores for each player. Another feature that we decided to add was the bookmark feature. So basically, if the player is unable to answer a question at a specific, he/she can bookmark it and view them after they are done with the rest of the questions.

There are many quiz apps online. However, we rarely came across the bookmark feature in any of it. Also, the leaderboard feature, though we can’t show it to the user as we didn’t make it online, most of the quizzes online don’t have that feature too.

**3. REQUIREMENTS:**

**3.1 Software requirements:**

* Java development kit
* Java virtual machine
* Java runtime environment
* SQL (for database management)
* JDBC Connector
* Terminal and Notepad

**3.2 Hardware requirements:**

* Computer or Laptop or PC

**4. IMPORTS AND LIBRARIES USED:**

Our project is purely based on the usage of the API (Application Programming Interface). AWT (Abstract Window Toolkit) and Swing have been used to create the user interface of the quiz and this has been connected to the database using the JDBC concept. The packages imported are:

1. java.awt.\*;
2. java.awt.event.\*;
3. javax.swing.\*;
4. java.sql.\*;

Awt and swing libraries provide the access to the basic components like the buttons, text fields, labels,etc. We do not have to import the specific libraries for each of the components. However, the awt event is a package that provides us with all the classes and interfaces for handling the events fired by the components of awt and swing. This package has three categories: First, the events - these classes represent the events generated by the components. Second, the event listeners - these are the most important as they help us specify the function for all the components. So basically, these are all interfaces defining methods, one for each event class, which must be implemented by any object that wants to be notified when a particular event takes place. The event listener used in this project is the ActionListener. The Java ActionListener is notified whenever you click on the button or menu item. It is notified against ActionEvent. It has only one method: actionPerformed(). Mainly the actionlistener is used with JButton. The buttons’ method is added in the abstract actionPerformed method.

**4.1 Java AWT**

Java AWT is used in Java to develop graphic user interface or window-based applications. Java AWT components are platform independent because the components are displayed according to the operating system. The components are heavyweight because it uses the resources of the operating system. The Java AWT package consists of classes such as, TextField, Label, TextArea, RadioButton, CheckBox, Choice, List and more. In the API, the Java AWT has components such as, a container, a window, a panel, and a frame. A container can keep other components like buttons, text fields, labels and more. A window has no border and menu bars. A panel does not contain a title bar and menu bars but has buttons or text fields. A frame contains a title bar and can have menu bars plus, has buttons or text fields.

**4.2 Java Swing**

Java Swing is used along with Java AWT to create window based applications on API. They are platform independent because their displays do not change in any operating system. The Java Swing components are lightweight. This package has classes such as, JButton, JTextField, JTextArea, JRadioButton, JCheckBox, JMenu, JColorChooser, and more. It also provides powerful components such as tables, lists, colorchooser and more.

**4.3 Java SQL**

Java SQL package provides the API for accessing and processing data stored in a data source usually by a relational database using the Java programming language. This package contains the entire JDBC API that sends SQL statements to relational databases and retrieves the results of executing those SQL statements. Using the Java SQL package, it can make connections with a database, it can send SQL statements to a database, retrieve and update the results of a query, standard mappings, have exceptions, and have metadata.

**4.3.1 Database and Connections**

In our project we have used a database to store the users’ name and their score. For this, first we had to install mysql on our laptop. The mysql setup already has the JDBC connector with it. We need to make sure our computer has a server, or else it won’t install.

After installing the appropriate drivers, we need to establish the JDBC connection. There are four steps for this:

1. **Import JDBC packages:**

The Import statements tell the Java compiler where to find the classes you reference in your code and are placed at the very beginning of your source code.

import java.sql.\*

1. **Register JDBC driver:**

You must register the driver in your program before you use it. Registering the driver is the process by which the Oracle driver's class file is loaded into the memory, so it can be utilized as an implementation of the JDBC interfaces.

There are two approaches to this:

1. Using class.forname()
2. Using DriverManager.registerDriver()

We used the first approach. It’s the most common method used to register a driver. It

Dynamically loads the driver class file into the memory, which automatically registers it.

This method is preferable because it allows us to make the driver registration

configurable and portable.

Here is a snippet from our code:

try {

**Class.forName("com.mysql.jdbc.driver");**

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/quiz", "root", "harsha");

PreparedStatement p= con.prepareStatement("insert into username(Useramecol)values(?)");

p.setString(1,t.getText());

int x=p.executeUpdate();

}

catch(Exception e1)

{

System.out.println(e1);

}

1. **Database URL formulation:**

After the driver is loaded, we can establish the connection using DriverManager.getConnection() method. There are three overloaded methods:

1. getConnection(String url)
2. getConnection(String url, Properties prop)
3. getConnection(String url, String user, String password)

We used the third method. Here each form requires a database URL. A database URL is a

URL that points to the database.

Here is a snippet from our code:

try {

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

**Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/quiz", "root", "harsha");**

PreparedStatement p= con.prepareStatement("insert into username(Useramecol)values(?)");

p.setInt(1,count);

}

catch(Exception e1)

{

System.out.println(e1);

}

1. **Create connection object:**

We have used the PreparedStatement method in our connection. The PreparedStatement interface is a subinterface of Statement. It is used to execute parameterized queries. The reason why we used the PreparedStatement object is that it enhances the performance of the application and makes it fast because the query is compiled only once.

Here is a snippet from our code:

try {

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

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/quiz", "root", "harsha");

**PreparedStatement p= con.prepareStatement("insert into username(Useramecol)values(?)");**

**p.setInt(1,count);**

}

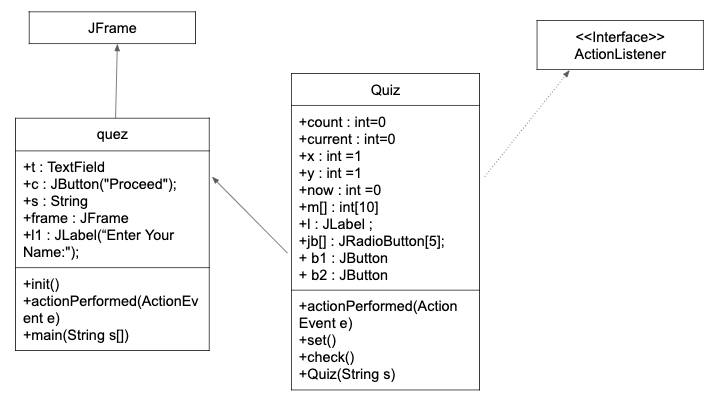
catch(Exception e1)

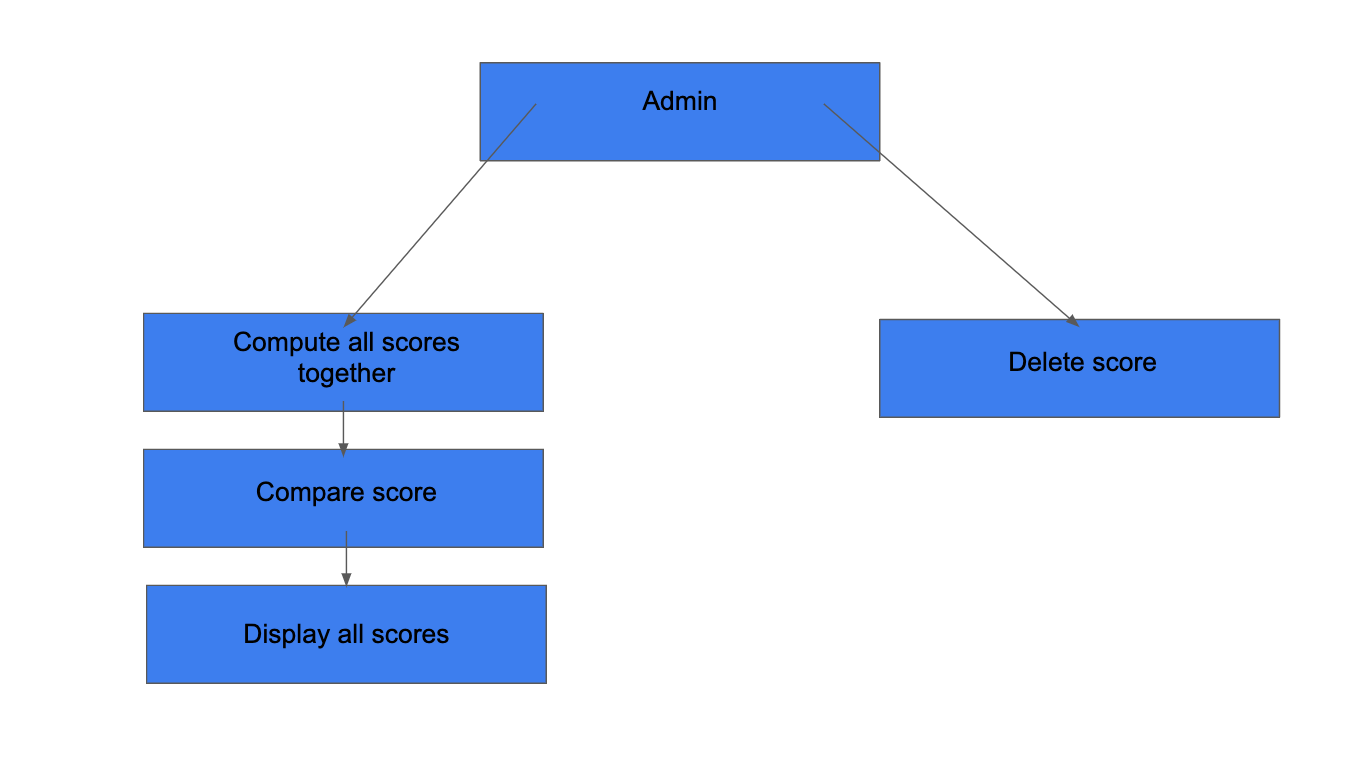
{

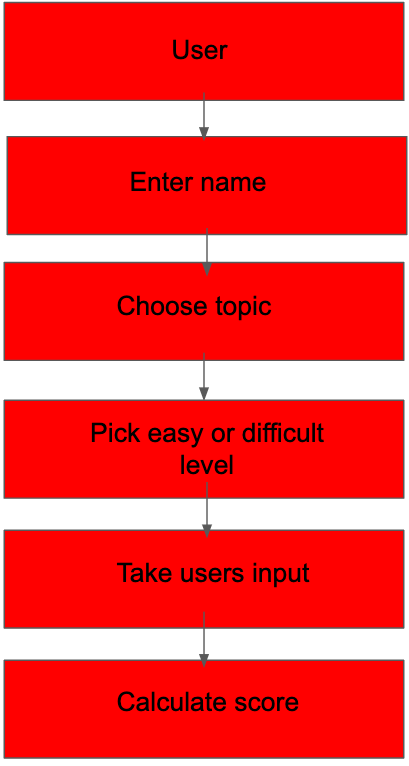
System.out.println(e1);

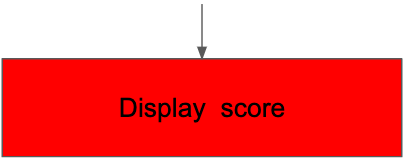
}

**5. CLASS DIAGRAM AND FLOWCHART:**

****

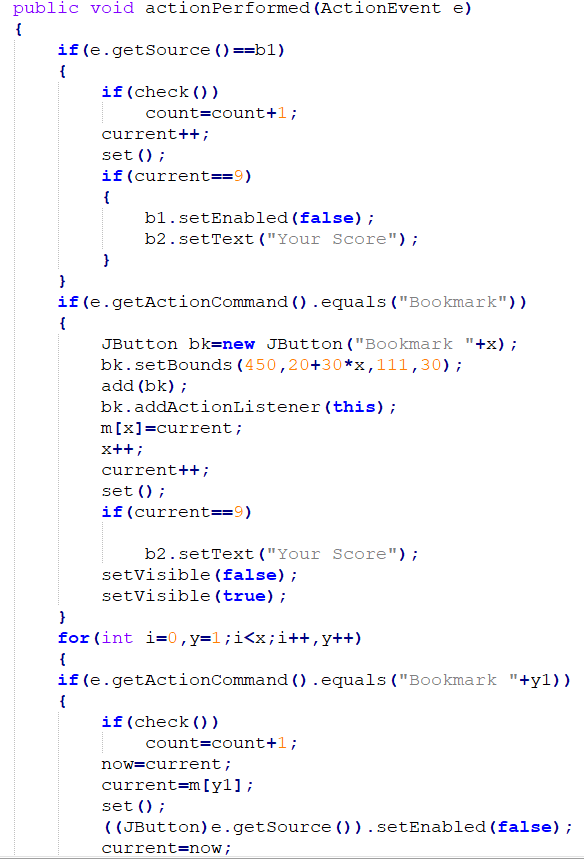






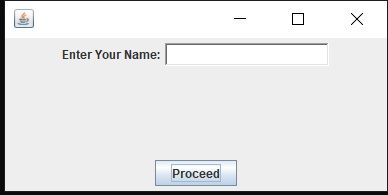
**6. CODE SNIPPETS WITH LOGIC AND OUTPUT:**

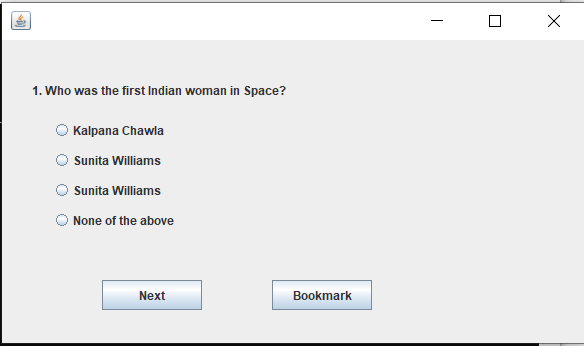
1. Code Snippets

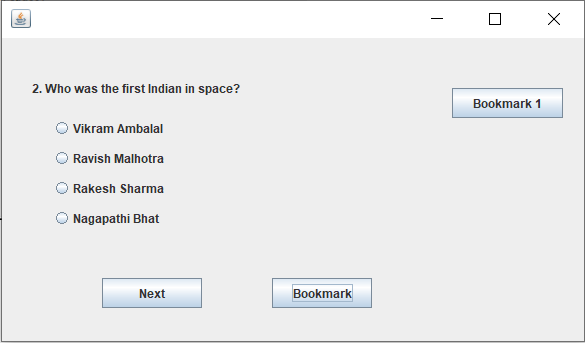


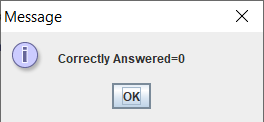
//Method defining actions for next and bookmark buttons

1. Output screenshots for quiz

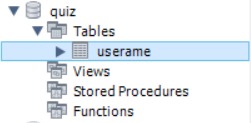
 FRAME 1

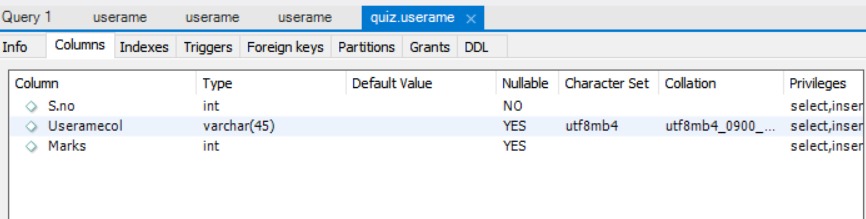
 FRAME 2

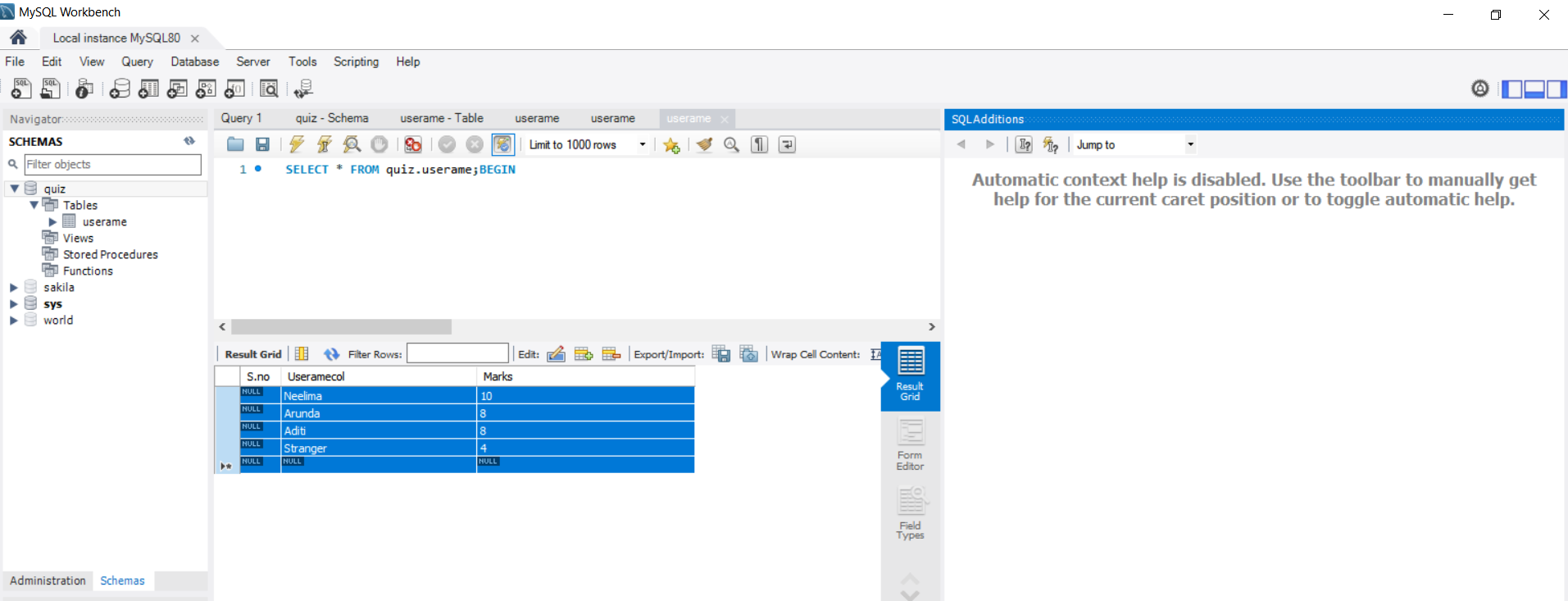
when ‘Bookmark’ is clicked

 END FRAME

1. Database (not visible to user)







**7. TIMELINE**

| Date | Work |
| --- | --- |
| March 25 | Planning of project |
| April 1 | Started work on creating the quiz in GUI |
| April 8 | Started work on database (leaderboard) in GUI |
| April 15 | Fixed minor issues in code |
| April 20 | Submission of report |

**8. TEAM MEMBER CONTRIBUTION:**

* Aditi Goyal (003) - UI (all actionPerformed() and username frame)
* Sri Venkata Neelima Chinta (070) - Database
* Arundarasi Rajendran (081) - UI( Quiz() and set())