**Design Document**

**Section 1 - Project Description**

**1.1 Project**

**Project Name:** Hobby Trivia Quiz

**1.2 Description**

The Hobby Trivia Quiz Application is a Java-based program. It features a graphical user interface (GUI) that allows users to participate in a quiz, view the rules, and check their scores. It also includes the ability to connect to the MySQL database to upload the user’s score after completing the quiz.

The application is divided into four main components: a game launcher (QuizGamebyIan.java), the quiz logic (Quiz.java), a scoring system (Score.java), and a rules display (Rules.java).

**1.3 Revision History**

| **Date** | **Comment** | **Author** |
| --- | --- | --- |
| 2024-08-10 | Code created | Ian Bajwa |
| 2024-08-11 | Demo has been recorded; database finished | Ian Bajwa |
| 2024-08-17 | Unit tests added, code refined | Ian Bajwa |

**Section 2 - Overview**

**2.1 Purpose**

The purpose of this module is to provide a comprehensive and fun quiz game experience. The intended audience includes users who enjoy participating in quizzes and wish to test their knowledge across various questions while also exploring the mind of the creator known as “Ian Bajwa” and his hobbies.

**2.2 Scope**

This module includes functionalities for starting the quiz, displaying rules, showing scores, handling user interactions and storing their data. It covers both the GUI and the backend/frontend logic necessary to run a quiz game.

**2.3 Requirements**

*This program will deliver a neatly organized list of functional requirements.*

**2.3.1 Functional Requirements**

* **Requirement 1 (R1):** The system shall provide a GUI for users to start the quiz, view rules, and exit the application.
* **Requirement 2 (R2):** The system shall allow users to input their name and start the quiz.
* **Requirement 3 (R3):** The system shall display quiz questions one at a time, allow users to select answers, and navigate between questions.
* **Requirement 4 (R4):** The system shall calculate and display the final score based on correct answers.
* **Requirement 5 (R5):** The system shall show the rules of the quiz when requested.
* **Requirement 6 (R6):** The system shall connect to the assigned SQL database using the correct login info, to upload the entered user data.

**2.3.2 Non-Functional Requirements**

* **Performance:** The application should handle typical user inputs and interactions with minimal delay.
* **Reliability:** The application should not crash or lose data during use. It should handle incorrect inputs gracefully.

**2.3.3 Technical Requirements**

* **Hardware:** The application should run on any machine with a Java runtime environment.
* **Software:** The application is developed using Java Swing for the GUI, MySQL for data storage, and is intended to run on a Java Virtual Machine (JVM).

**2.3.4 Security Requirements**

* **Authentication:** The application does require light user authentication for the quiz game. Any entered username will suffice and helps easily locate the user in the database.
* **Data Encryption:** The application does handle data that requires decryption to access.

**2.3.5 Estimates**

| **#** | **Description** | **Hrs. Est.** |
| --- | --- | --- |
| 1 | Implement GUI components (buttons, labels) | 10 |
| 2 | Develop quiz logic and scoring mechanism | 15 |
| 3 | Create rules display functionality | 5 |
| 4 | Create MySQL database, create connector | 5 |
| 5 | Integrate components and test functionality | 10 |
| **TOTAL:** |  | **45** |

**2.3.6 Traceability Matrix**

| **SRS Requirement** | **SDD Module** |
| --- | --- |
| R1 | 3.1, 6.1 |
| R2 | 3.1, 6.2 |
| R3 | 3.1, 6.3 |
| R4 | 3.1, 6.3 |
| R5 | 3.1, 6.1 |
| R6 | 3.1, 6.1 |

**Section 3 - System Architecture**

**3.1 Overview**

The system architecture comprises a Java Swing GUI for user interaction and separate classes to handle quiz logic, scoring, and rules display. The QuizGamebyIan class serves as the main entry point, allowing users to start the quiz, view rules, or exit the application. The Quiz class manages the quiz questions and user answers. The Score class calculates and displays the final score, while the Rules class shows the rules of the quiz. The MySQL table carries the user data through a java created connector for storage purposes.

**3.2 Architectural Diagrams**

* **Component Diagram:** Illustrates the relationships between QuizGamebyIan, Quiz, Score, and Rules.
* **Sequence Diagram:** Shows the interaction and connectivity between GUI components, external database and the underlying logic during quiz operations.
* **Class Diagram:** Displays the attributes and methods of each class (QuizGamebyIan, Quiz, Score, Rules).

**Section 4 - Data Dictionary**

**Table:** User Scores

| **Field** | **Notes** | **Type** |
| --- | --- | --- |
| UserID | Unique identifier | INTEGER |
| UserName | The user's name | VARCHAR(100) |
| Score | The final score | INTEGER |
| Date | Date uploaded | TIMESTAMP |

**Section 5 - Data Design**

**5.1 Persistent/Static Data**

**5.1.1 Dataset:**

* **Entities:**
  + **Score:** Stores the final score for each user.
    - **Attributes:** UserID (PK), UserName, Score, Date

**Section 6 - User Interface Design**

**6.1 User Interface Design Overview**

* **Main Menu:** Contains buttons to start the quiz, view rules, and quit.
* **Quiz Screen:** Displays questions and options for answers.
* **Score Screen:** Shows the uploaded score after completing the quiz.
* **Rules Screen:** Displays the rules of the quiz game.

**6.2 User Interface Navigation Flow**

1. **Main Menu → Quiz Screen:** Click "Start Quiz".
2. **Quiz Screen → Score Screen:** Complete the quiz and submit.
3. **Main Menu → Rules Screen:** Click "Rules" to view rules.

**6.3 Use Cases / User Function Description**

* **Start Quiz:** User clicks "Start Quiz", inputs name, and begins the quiz.
* **Quit:** User clicks "Quit" to exit the application.
* **View Rules:** User clicks "Rules" to view the quiz rules.

**Section 7 - Testing**

**7.1 Test Plan Creation**

**Objective:** Validate functionality of GUI components, quiz logic, scoring and data transfer. **Scope:** Includes unit tests for button actions and quiz functionality. **Resources:** Java Swing, JUnit **Schedule:**

* **Component Testing:** 1 week
* **Integration Testing:** 1 week

**Test Cases:**

| **Test Case** | **Input** | **Expected Output** | **Actual Output** |
| --- | --- | --- | --- |
| Test Start Button | Click "Start Quiz" | Quiz window opens | Pass |
| Test Quit Button | Click "Quit" | Application exits | Pass |
| Test Rules Display | Click "Rules" | Rules window opens | Pass |
| Test Quiz Functionality | Answer questions | Correct score displayed | Pass |
| Test Database Insertion | Click “Submit” | Automatic data upload | Pass |

**Section 8 - Monitoring**

**Performance Metrics:**

* Response time of GUI interactions.
* Throughput of quiz question handling.
* Data transfer accuracy

**Error Metrics:**

* Rate of errors encountered during quiz execution.
* Types of errors and affected users.

**Availability Metrics:**

* Uptime of the application during testing.

**User Metrics:**

* Number of active users during testing.
* User session durations.

**Section 9 - Other Interfaces**

**9.1 Interface X**

* **Description:** MySQL Workbench interface is used to view the data table.

**Section 10 - Extra Design Features / Outstanding Issues**

* **Outstanding Issues:** The Unit Test for QuizGamebyIan.java can’t finish the testQuitButtonAction() function without quitting the program.

**Section 11 – References**

* **Java Swing Documentation:** [Java Swing API](https://docs.oracle.com/javase/8/docs/api/javax/swing/package-summary.html)
* **JUnit Documentation:** JUnit API

**Section 12 – Glossary**

* **GUI:** Graphical User Interface
* **JVM:** Java Virtual Machine
* **PK:** Primary Key
* **SQL:** Structured Query Language