**Title**: *System Design Document for QuiZone*

**Team Name**: *QuiZone*

**Project Description**: *QuiZone is a quiz platform that allows users to create, take, and review quizzes with customizable features like question randomization, timers, and instant feedback.*

**Team Members:** *Karine Davtyan, Tejleen Kaur , Shaqayeq Salimy*

**Date**: *November 3rd, 2024*

*Table of Contents*


1. Class Responsibility Collaborator (CRC) Cards
    - Description of Classes
    - Class Responsibilities and Collaborators
2. System Interaction with Environment
    - Operating Environment
    - Dependencies
3. Software Architecture Diagram
    - MVC Architecture Diagram
    - Explanation of Components
4. Error Handling Strategies
5. Summary

**1. CRC Cards:**

| Class Name: Question | |
|---|---|
| Parent Class: none<br>Subclasses: none | |
| Responsibilities:<br>• Represents an individual question in a quiz.<br>• Stores question text, possible options, and the correct answer. | Collaborators:<br>• Quiz (as part of a collection of questions within a quiz) |

| Class Name: Quiz | |
|---|---|
| Parent Class: none<br>Subclasses: none | |
| Responsibilities:<br>• Represents a quiz with a title, description, and a list of questions.<br>• Manages the collection of Question objects.<br>• Holds metadata about the quiz (e.g., title, description). | Collaborators:<br>• Question (contains a list of questions)<br>• QuizController (for creating and retrieving quizzes) |

| Class Name: QuizController | |
|---|---|
| Parent Class: none<br>Subclasses: none | |
| Responsibilities:<br>• Handles HTTP requests related to quiz operations.<br>• Contains endpoints for creating and retrieving quizzes.<br>• Saves and loads quiz data from quizzes.json. | Collaborators:<br>• Quiz (creates and manages quiz instances)<br>• Question (manages questions within quizzes) |

| Class Name: QuizProjectApplication | |
|---|---|
| Parent Class: none<br>Subclasses: none | |
| Responsibilities:<br>• Initializes and runs the Spring Boot application.<br>• Configures the application context and manages dependencies. | Collaborators:<br>• QuizController (initializes the controller for handling requests) |

| Class Name: form-creation.html | |
|---|---|
| Parent Class: none<br>Subclasses: none | |
| Responsibilities:<br>• Provide input fields for quiz titles, descriptions, and questions.<br>• Use JavaScript to validate form inputs on the client side.<br>• Send data to the backend API to save quiz details. | Collaborators:<br>• QuizController (for saving quiz data to the backend).<br>• styles.css (for form styling and layout).<br>• JavaScript functions for form validation and handling API request |

| Class Name: quiz-history.html | |
|---|---|
| Parent Class: none<br>Subclasses: none | |
| Responsibilities:<br>• Display a list of past quizzes, including flagged questions.<br>• Fetch quiz history from the backend API.<br>• Provide options to delete specific quiz history records | Collaborators:<br>• QuizController (for retrieving and deleting quiz history).<br>• JavaScript functions to handle API requests and render quiz history.<br>• styles.css (for visual styling of the quiz history layout). |

| Class Name: feedback.html | |
|---|---|
| Parent Class: none<br>Subclasses: none | |
| Responsibilities:<br>• Display quiz questions for users to answer.<br>• Fetch quiz data from the backend.<br>• Display immediate feedback based on user answers. | Collaborators:<br>• QuizController (for fetching quiz data and saving user answers).<br>• JavaScript functions to handle user input, API requests, and feedback display.<br>• styles.css (for styling feedback messages and quiz layout). |

| Class Name: styles.css | |
|---|---|
| Parent Class: none<br>Subclasses: none | |
| Responsibilities:<br>• Define visual styles for layout, colors, and typography across all frontend pages.<br>• Apply styling for feedback messages to distinguish correct and incorrect answers. | Collaborators:<br>• form-creation.html, quiz-history.html, feedback.html (HTML files using these styles).<br>• JavaScript feedback functions in feedback.html (to apply feedback-specific styles). |

## 2. System Interaction with Environment

**Overview**: The system is a web-based quiz application built with Spring Boot. It interacts with various components like the operating system, file storage, and the user's browser to function smoothly. The application expects a compatible environment with specific dependencies.

**Dependencies**:

- **Operating System**: The system can run on any OS that supports Java and Spring Boot (Linux, macOS, Windows).
- **Java Virtual Machine (JVM)**: Since the project uses Spring Boot, a compatible JVM (Java 11 or above) is required.
- **Web Browser**: The application relies on a modern browser (e.g., Chrome, Firefox, Safari) for rendering HTML pages and supporting JavaScript.
- **Storage**: Data is stored in a local JSON file (`quizzes.json`), which acts as a simple database to save quizzes and retrieve them when needed.

**Assumptions**:

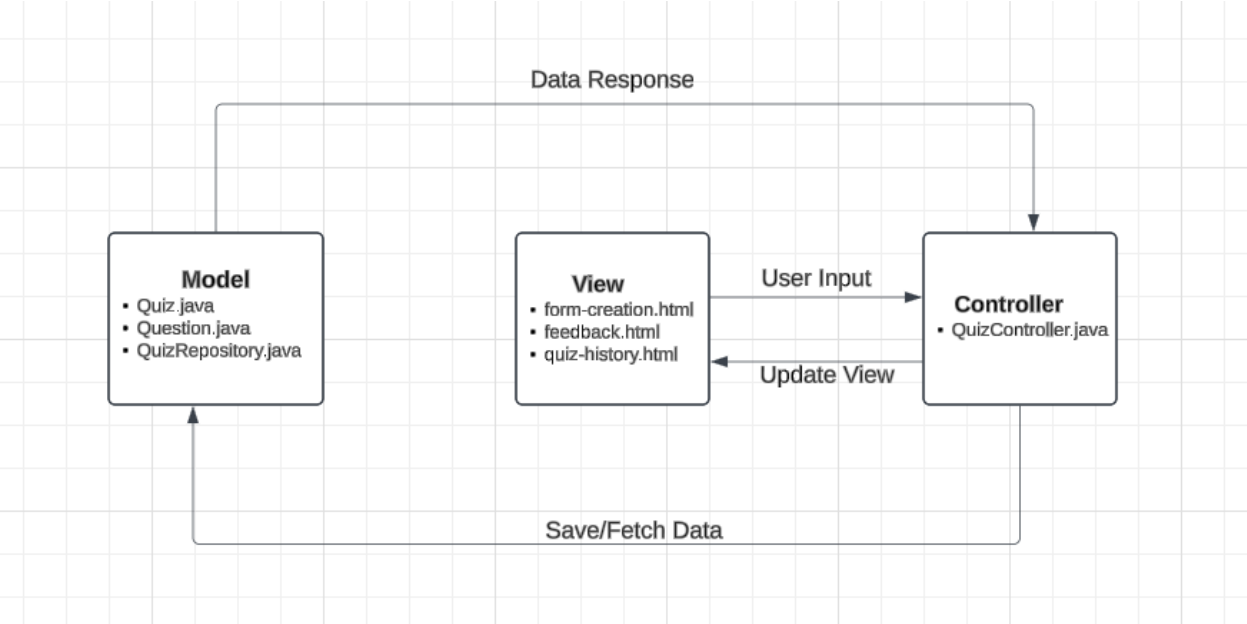
- The system assumes that the JSON file (`quizzes.json`) is located in the root directory and is accessible with read and write permissions.
- The server is expected to be running on port 8080 unless configured otherwise.
- The user's browser must support JavaScript to interact with the frontend dynamically.

## 3. System Architecture

The architecture of the system is based on the **Model-View-Controller (MVC)** pattern. The components are:

1. **Model**:
    - `Quiz` and `Question` classes serve as the data model for storing quiz details and individual question data.
    - Data is stored in `quizzes.json`, a JSON file acting as a lightweight database.
2. **View**:
    - The frontend HTML files (`form-creation.html`, `feedback.html`, and `history.html`) allow users to create, view feedback, and check history.
    - JavaScript code in the HTML files fetches data from the backend and displays it to the user.

3. **Controller**:
    - `QuizController` serves as the backend controller, managing API endpoints for saving quizzes, retrieving feedback, and accessing history.
    - The controller processes HTTP requests, interacts with the Model, and returns data to the View.




## 4. Error Handling Strategy

**Description**: The application anticipates and manages errors to ensure a stable user experience. Here's a summary of how various error types are handled:

1. **Invalid User Input**:
    - Input validation is applied in the HTML forms. JavaScript checks that required fields are filled out before submission.
    - The backend (`QuizController`) verifies the JSON structure of incoming data, ensuring that mandatory fields (like title, questions) are present.
2. **File I/O Errors**:
    - When saving data to `quizzes.json`, the system catches `IOException` errors to handle cases where the file may not be accessible.
    - If the file is not writable, the system logs an error and displays a user-friendly message.
3. **Server Errors**:
    - If the Spring Boot server fails to start or encounters issues, error logs are generated. Users are shown a fallback error page, while specific exceptions are logged for developers.
4. **Network/External System Failures**:

- Although this is a local application, if the system is deployed online, it would rely on stable network connections. Failed fetch requests in JavaScript display user-friendly messages, like "Error loading feedback data."

**Error Messages**: For each type of error, the system provides feedback to users, such as:

- “Failed to save the quiz. Please try again.”
- “Error loading feedback data.”

## 5. Summary

The system design is based on a simple, maintainable structure using the MVC pattern, with quizzes stored in a JSON file. The design anticipates user errors, file access issues, and general server exceptions. When an error occurs, the system provides clear feedback to the user and logs the issue for developers.

The architecture is scalable enough to support additional features, such as storing data in an actual database, enhancing input validation, and expanding the API for more interaction with external systems if deployed.