Project Documentation

1. Overview

The project implements a quiz creation mechanism in accordance with the CSC8404 coursework specification. The system facilitates the creation of student quizzes, the management of quiz results, and provides a final evaluation decision based on the performance of the student. Using object-oriented programming concepts like interface-based design and immutability, the system ensures flexibility and maintainability.

Key System Features:

- **Question Types**: Supports multiple-choice and free-response questions.
- **Quiz Generation**: Generates both standard and revision quizzes based on a student's prior performance.
- **Student Performance Tracking**: Records quiz attempts and issues final grades (PASS, FAIL, or TBD).
- **Statistics**: Produces detailed reports on student performance.

---

2. System Overview

The system is composed of several classes and services, ensuring separation of concerns to promote modularity and extensibility.

2.1 Core Components:

- **Question Types**:
    - **Free Response Questions**: Students provide an open-ended response, which is compared against the correct answer.
    - **Multiple Choice Questions**: Students select one or more correct answers from a list of predefined options.
- These types of questions are implemented as subclasses of an abstract class `AbstractQuestion`, which defines common behaviors.
- **QuizService**:

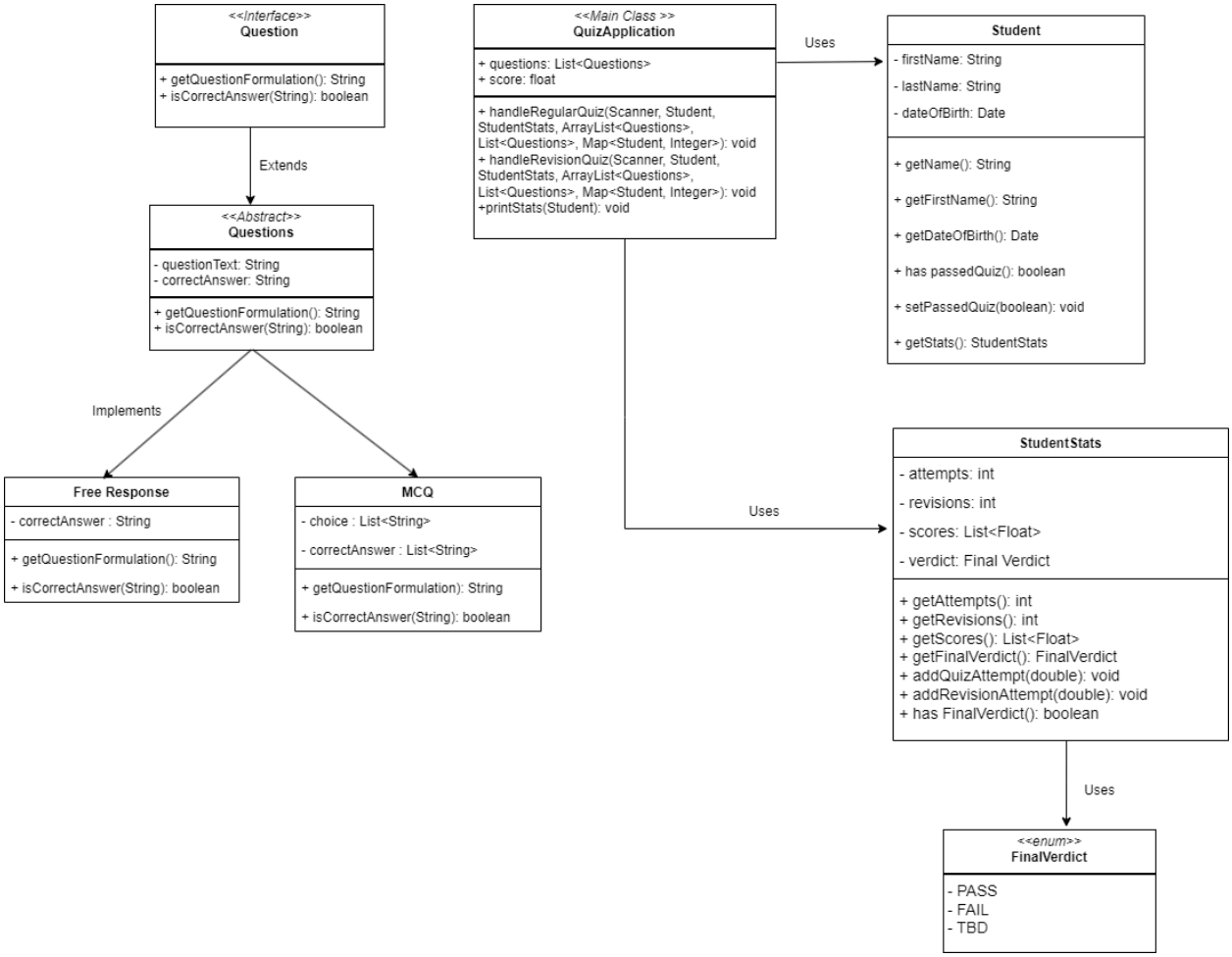
    - Generates both regular and revision quizzes based on student performance.
    - Regular quizzes are generated from a pool of questions, while revision quizzes focus on questions the student previously answered incorrectly.
- **Student Performance Tracking**:
    - The `Student` class records student information and quiz attempts to support the generation of revision quizzes and performance statistics.
- **StatisticsService**:
    - Tracks performance data and generates the student's final evaluation (PASS, FAIL, or TBD) based on quiz results.

2.2 Design Principles:

- **Abstraction and Inheritance**: The system generalizes behaviors using abstract classes and allows specialized implementations through subclasses (e.g., `FreeResponseQuestion`, `MultipleChoiceQuestion`).
- **Interface-Based Design**: Interfaces define contracts for component interaction, making future extensions simple and seamless.
- **Modularity**: Each class has a clearly defined role, enabling easy addition of new features (e.g., additional question types) without altering existing code.

---

3. UML Class Diagram



---

4. Unit Tests and Testing

Unit tests are implemented to ensure the system functions correctly in various scenarios, including normal operations, edge cases, and exceptional conditions.

(Insert a UML diagram illustrating the relationships between classes and interfaces in your system.)

4.1 Core Components Tested:

- **QuizService Tests**: Ensures that both regular and revision quizzes are generated accurately.
- **Question Type Tests**: Verifies that free-response and multiple-choice answers are evaluated correctly.

- **Student and Statistics Tests**: Tests the system's ability to track quiz attempts and issue correct verdicts (PASS, FAIL, TBD).

4.2 Testing Results:

(Describe the test coverage, success rates, and any edge cases or exceptional scenarios that were handled.)

---

5. Conclusion

The system meets the project specifications, offering flexible quiz generation and comprehensive student performance tracking. Following sound software engineering principles, the system is designed for extensibility, allowing for future enhancements such as additional question types or integration with a database. The inclusion of thorough unit testing ensures that the system is reliable and robust.