Online Quiz System (OQS)

1. Functional Requirements

User Authentication & Authorization

• User Registration: The system should allow instructors and students to create an account using their email and password.

• User Login: After ensuring the data the user entered is correct, the system allows the user to log in.

• Role Based Access: Different roles with different permissions. For example, students can solve quizzes but cannot create them.

• Tokens Generation: After the user logs in, the system generates an access token (short time expiration) and a refresh token (long time expiration).

Instructor: Quiz & Question Management

• The system should allow instructors to create a new quiz with metadata: title, description, duration, start/end time, total marks, and visibility.

• The system supports multiple-choice questions (MCQs) with one correct answer.

Student: Attempting Quizzes

• The system should allow a student to start a quiz only during its availability time.

• Students can see all quizzes they entered and their answers, graded quizzes, and passed or non-passed quizzes.

Grading Mechanism

• After a student submits, the grade is calculated and answers are stored in the database.

• Students cannot edit any answer after submitting and cannot start the same quiz again.

UI & Accessibility

• Separated dashboard for each role compliant with the functions they can perform in the system.

• Responsive UI due to using asynchronous programming, meaning no freezing of threads.

2. Non-Functional Requirements

Security

• All passwords are saved encrypted in the database.

• Using a strong secret key with JWT tokens.

Performance

• Using caching for quizzes currently available to reduce database access because most students will request the same quiz at the same time.

Compatibility

• The system should support modern browsers (Chrome, Firefox, Edge, Safari) and degrade gracefully on older browsers.

Maintainability

• The codebase should be modular (separation of concerns).

Reliability & Availability

• Applying rate limiting to prevent users from sending an infinite stream of requests.