**TESTING**

**Unit Testing**

Unit testing plays a crucial role in ensuring that individual components of a website function correctly. In the context of our "Quiz Hub" project, unit testing was applied to every significant feature and module to guarantee that they worked as expected in isolation. We utilized **Jest**, a JavaScript testing framework, because of its versatility and integration with React (which was used in the front-end of the website).

Each part of the quiz website, from the timer functionality to the question-answer logic, was subjected to unit tests. For instance, we wrote tests for the functions responsible for handling user inputs, calculating scores, and tracking the user's progress throughout the quiz. This approach enabled us to identify issues early in the development process, allowing us to fix bugs in the logic before they became more complex.

A major challenge with unit testing was ensuring that each function was tested independently. This meant mocking external dependencies, such as database calls or API requests, to simulate a controlled environment for each test. By doing so, we ensured that our tests were stable and not reliant on external factors. Overall, unit testing helped us ensure that our basic components were functioning perfectly, contributing to a smoother overall website experience.

**Integration Testingj**

Once individual components were unit-tested, we needed to ensure that the different parts of the system worked together. Integration testing focused on testing the interaction between multiple components of the website. The goal was to identify issues that might arise when multiple modules interacted in a real-world scenario.

For example, we tested how the quiz's front-end interface interacted with the back-end API, ensuring that when a user submitted answers, the system correctly recorded responses and calculated the final score. We also examined how data like user progress was passed through various modules — from the front-end to the back-end and back to the user interface.

One of the primary tools we used for integration testing was **Cypress**. This tool allowed us to simulate user interactions and check if the components worked together in a real browser environment. For instance, we automated tests for quiz flow, verifying that the user could seamlessly navigate between questions and see results correctly.

Integration testing was critical because it highlighted issues that unit testing couldn’t, such as issues related to the communication between components or dependencies between the front end and back end. Through this process, we discovered and fixed bugs such as inconsistent state management or incorrect handling of data between different sections of the site.

**Browser Testing**

Ensuring that the quiz website functioned well across different browsers was a key part of our testing process. We focused on testing three main browsers: **Google Chrome**, **Mozilla Firefox**, and **Safari**. Since different browsers handle rendering and JavaScript execution in unique ways, it was important to ensure cross-browser compatibility.