SOEN390 - Testing Plan for Sprint 3

Goal & Overview

For Sprint #3, our development focus is on achieving a target code coverage of 40%. This entails a comprehensive testing strategy, with a primary emphasis on the implementation of Unit Testing. By systematically validating individual units of code, we aim to enhance the reliability and functionality of our software. Additionally, to streamline the integration process and ensure code quality, we will introduce the Continuous Integration (CI) pipeline. This CI pipeline will be configured to trigger automatically each time a merge is attempted from our feature branch to the main branch. This proactive approach to testing and integration reflects our commitment to delivering robust and high-quality code, ultimately contributing to the overall success of our project. The tool set that we will use for these tests are Cypress for End-to-End testing, Jest for Unit Testing and integration testing.

Unit Testing

We are putting a lot of focus on comprehensive unit testing in Sprint #3 to make sure our software is reliable. Every controller will have thorough unit tests written for it, with stubbed or mocked external calls that exit the controller file. This method ensures that our classes are tested in a fully isolated environment, improving the functionality and dependability of each part. Concurrently, we're attempting to optimize the cooperation between Cypress and Jest in order to effectively test almost every new component separately.

Integration Testing

The main goal of our integration testing activities will be to prepare the foundation for our Next.js TypeScript and PostgREST-based project. By confirming important API endpoints, highlighting appropriate communication, and managing different HTTP methods, our main goal will be to provide the groundwork for a fundamental grasp of system interactions. In order to verify fundamental functions including data retrieval, insertion, updating, and deletion, database interaction tests will be started concurrently. By guaranteeing data quality and consistency, the goal is to strengthen the accuracy of these exchanges. We will also use Jest's robust mocking features to mimic external services and dependencies, giving us the ability to methodically manage and modify database queries and API answers. This method ensures that the first integration tests are carried out methodically, laying the groundwork for further testing, by providing a concentrated and regulated testing environment.

System Testing

We will include Cypress into our system testing methodology to do thorough End-to-End (E2E) validation. confirming the correctness of data exchanges and guaranteeing the smooth integration of components. These E2E tests are intended to achieve a minimum coverage of 40% and an aspirational objective of 60%. They will be crucial in the examination of the system's overall operation. We want to detect and resolve such problems early on by methodically testing important user scenarios, which will increase the project's resilience and dependability.