**Milestone 4: Testing update report**

Team Pentakill

Our project's success depends on delivering a user-friendly and robust web application. To achieve this, we've developed a comprehensive Front-End Testing Strategy focusing on thorough examination and continuous improvement. Based on our milestone 3, we focus on three testing procedures, the unit test, system integration test, and the user acceptance test. This strategy contains several key components, each targeting different aspects of the application to ensure its overall quality and usability.

**1. Testing Implementation Breakdown**

**Front End Testing Strategy and Updates**

Unit Testing with Selenium:

Our primary tool for unit testing is the Selenium suite. The test-driven approach allows us to address potential issues as early as possible. These tests are written and defined prior to completing the coding. Below are all the front-end pages we have run unit tests on so far. Navigation Bar, Home Page, Event Category Page, Event Registration Page, Event Information Page, User Login Page, Ticket Purchasing Page, and User Registration Page. We rigorously test the user interface, focusing on data format validation, button functionalities, form submissions, and ensuring a logical flow throughout the user journey.

Front-end Integration Testing:

As the project progresses, we maintain a cycle of comparison, linkage, and logic testing among the front-end components. This ongoing evaluation helps in identifying any discrepancies in the user experience, ensuring that transitions between different parts of the application are seamless and intuitive. We achieve this by defining a set of ground rules via continuous communication, as well as constant code reviews before and after we merge all components together.

System Integration Testing:

While full-scale system integration testing will commence once we integrate the front-end with the back-end, preparatory steps are already underway. This is achieved by frequent communication between the back-end and front-end team members; so the chance of discrepancies will be minimized.

Future User Acceptance Testing (UAT):

User feedback is essential for refining the application, which is why User Acceptance Testing (UAT) will be conducted post-launch. This phase will engage real users in testing the application, providing insights into its functionality, usability, and alignment with user expectations.

**2. Future Plans and Improvements**

A key area of focus for future enhancements is the unification of the user interface across different platforms. This ensures that the application provides a consistent and responsive experience whether accessed via PC or mobile devices. Such cross-platform compatibility is vital for meeting the user's expectations.

We have successfully passed numerous unit tests, particularly in areas concerning data validation and server communication. These accomplishments underscore the application's capability to handle user inputs accurately and maintain connections to the correct components.

However we do occasionally encounter challenges when the actual logic flow derives from our initial test plans, especially when new features are introduced. An example is the introduction of system alert notifications, which could disrupt processes like form submissions if not previously accounted for in testing scenarios. Moving forward, we aim to dynamically update our test plans to incorporate new functionalities, ensuring that our testing framework remains as agile and comprehensive as the development process itself.

**Back End Testing Strategy and Updates**

Unit Test:

We currently only unit test the basic functions of the Business class. Below are all the back-end classes we have run unit tests on so far: CustomerFactory, Customer, EventHolder, EventRegistry, Event, Order, OrderEvent, PaymentMethod, Rating, ShoppingCart.

Future Plan:

For now, every unit test works just great and is ready to begin connecting the front and back end. After connecting the front and back end, we will test if the data is connected correctly to the front end and if the database works fine.

**Automation Implemented:**

Automated Unit and Integration Testing:

We utilize Selenium and Jest for automated unit and integration testing of our React frontend components. Selenium is used to simulate user interactions, while Jest is employed for writing and running test cases.

Workflow Integration:

Automated testing is integrated into our development workflow. Upon each code submission, Jest automatically executes our test suite to ensure the expected behavior of frontend components.

Benefits and Advantages:

* Automated testing enhances code quality and system stability by quickly identifying and rectifying potential issues.
* Integration of Selenium and Jest provides developers with continuous feedback, aiding in faster development and reduction of human errors.

Challenges and Limitations:

* There is a learning curve associated with implementing automated testing using Selenium and Jest, especially for complex interactive components.
* Maintaining automated test suites requires continuous attention and updates to keep pace with changes in the application.

Feedback and Improvement:

The implementation of automated testing has received positive feedback from the development team, acknowledging its contribution to improved code quality and accelerated development processes. We plan to expand the coverage of automated testing and optimize the execution speed and stability of our test suites.