April 30, 2023

Rasool Abbas, Onel Jimenez, Michael Mcfarlin, Ethan Wharton

Project ROME

Test design and planning

Test specification

# Introduction

This document outlines how the team has designed and planned testing for Project ROME. This document should provide an outline of all tests made by the team, and all items/features that were not tested. It also highlights our intended approach for testing and criteria to determine test success/failure.

# Items Tested

* Fundraiser editing (manual and automated)
* Fundraiser creation (automated)
  + Exclusive API testing
  + End-to-end (front-end to database)
* Query fundraisers through API (automated)
* User login API (automated)
* User registration (end-to-end automated)
* Dashboard page layout (automated)
* Profile settings (manual)

# Items Not Tested

* Querying database
* Donations (relatively simple form submission, no notable features)
* Pagination

# Approach

Our main goal is to automate testing where possible. At the current iteration, we are at approximately 50% testing coverage of the application, with most of the coverage in critical methods of the application. Automated testing is implemented using nUnit. Testing methods are structured as “declare-setup-act;” declare all necessary pre-requisites, set up the pre-requisites and testing scenario, act by initiating the test and checking if it has passed or failed. This implementation is to minimize inconsistency between tests. It also ensures a standard for future tests, should further development occur.

Manual testing occurs usually when something significant occurs during development, or when automation efforts cost more than manual testing. If implementing testing methods is efficient, and if there are no other critical tasks in backlog, automated testing is preferred and encouraged.

We have also designed an end-to-end testing implementation using Playwright. It tests UI elements on the front-end and validates responses from actions (almost like simulating real-world scenarios, excepted through automation). It also ensures that the UI remains consistent.

It is critical to ensure tests are maintained with new feature implementation. If testing fails, further development should not proceed until the failing tests are addressed.

# Item Pass/Fail Criteria

An item passes if the following occur;

* Automated tests pass.
* Item meets customer requirements.

It is recommended that implementation meets the standards of the customer. So, to meet constraints designated by the customer while also ensuring that development can continue, a “strong pass” occurs if;

* Automated tests pass.
* Item meets customer requirements.
* Customer approves of item’s usability and functionality.

(For the sake of the project, strong pass won’t occur but is defined in case of further development)

A failure is if any one condition is not meet in the pass criteria. However, not every failure will be documented, and it is discouraged to document every failure. Significant failures will be documented in the Incident Report.

# Testing Deliverables

In addition to this document, the Incident Report and the Test Specification document include applications of this procedure.

The Test Specification highlights detailed testing scenarios.

The Incident Report highlights incidents that occur where test scenarios fail. It also may include remedy implementations for those incidents.