ROMAVERSIO

TEST PLAN

Authors

Sukh Virdi, Kacper Kiermasz, Simone Ram, Harman Kalair

Table of Contents

Test Plan

Project "Roman Numeral Converter"

Document Revision History

Date	Version	Description	Author	Reviewer & Approver
23.11	0.1	Document Created	Sukh Virdi	Kacper Kiermasz
23.11	0.2	Section 1, 2, 3	Harman Kalair	Kacper Kiermasz
24.11	0.3	Section 4, 5, 6	Sukh Virdi	Kacper Kiermasz
24.11	0.4	Section 7, 8, 9	Simone Ram	Kacper Kiermasz
24.11	0.5	Quality Check	Sukh Virdi	Kacper Kiermasz

1 INTRODUCTION

This project involves delivering a functional Roman Numerals Encoder/Decoder website, which has passed all stages of our own testing life cycle. The front-end side of the application is our main focus; however, we have also implemented a back-end which tests the logic of the application. It is essential that we maintain the same quality and performance in both parts of the application in order to deliver all of the requirements.

The Test Plan has been created to facilitate communication within the team members. This document describes approaches and methodologies that will apply to the unit, integration and system testing of the 'Romaversio' Application. It includes the objectives, test responsibilities, entry and exit criteria, scope, schedule major milestones, entry and exit criteria and approach. This document has clearly identified what the test deliverables will be, and what is deemed in and out of scope.

2 SCOPE

The test plan is focused towards producing sufficient documentation that describes how the tester will verify that 'Romaversio' works as intended. The development team will create tests using Jest in order to adhere to our test driven development strategy, following development we will create front-end snapshot testing (also using Jest). Upon completion of development, manual QA tests scripts will be created based on the user's requirements, and manual testing will be performed on the front-end. Once all the tests have passed, and development has concluded, the project team will consult an external/third party to conduct manual UAT testing.

3 QUALITY OBJECTIVES

3.1 Primary Objectives

A primary objective of testing is to: assure that the 'Romaversio' product meets the full requirements, including quality requirements (functional and non-functional requirements) and fit metrics for each quality requirement and satisfies the use case scenarios and maintain the quality of the product. At the end of the project development cycle, the user should find that the roman numeral converter/decoder has met or exceeded all of their expectations as detailed in the product requirements.

Any changes, additions, or deletions to the requirements document, Functional Specification, or Design Specification will be documented and tested at the highest level of quality allowed within the remaining time of the 'Romaversio' project and within the ability of the test team which is conducted by the QA tester.

3.2 **Secondary Objectives**

The secondary objectives of testing will be to: identify and expose all issues and associated risks, communicate all known issues to the project team, and ensure that all issues are addressed in an appropriate matter before release. As an objective, this requires careful and methodical testing of the application to first ensure all areas of the system are scrutinized and, consequently, all issues (bugs) found are dealt with appropriately.

4 TEST APPROACH

The approach that we are adopting as a team is acting in accordance to our requirements-based strategy, where an analysis of the requirements specification forms the basis for planning, estimating and designing tests. Test cases will be created during third sprint of our software development life cycle. All test types are determined in discovery phase of the project. We will conduct experience-based testing and error guessing utilize testers' skills and intuition, along with their experience with similar applications or technologies.

The project is using an agile approach, with weekly iterations (3 in total). At the end of each week the requirements identified for that iteration will be delivered to the team and will be tested.

4.1 **Test Automation**

The automated unit test scripts are present within both the front-end of the application and the back-end. In the back-end tests scripts test the logic of the application such as converting roman numerals to normal numerals (vice versa); in the frontend the tests are used to ensure that each component is rendered and rendered correctly. Smoke tests are included within the documentation directory on GitHub.

5 ROLES AND RESPONSIBILITIES

Role	Team Member	Responsibilities
Scrum Master	Sukh Virdi	Managing the Kanban board, sprints and created tickets. In addition to being. Responsible for the project documentation (e.g. test plan).
QA	Kacper Kiermasz	Ensures that all code is of high quality and adheres to our chosen SQA standard. Includes conducting code reviews which follows our code review strategy.
UX	Simone Ram	Responsible for designing the user interfaces and ensuring a good user experience. Includes collaboration with DevOps to conduct pre-project UI testing.
DevOps	Harman Kalair	In charge of the CI Pipeline and GitHub workflows.

6 TEST STRATEGY

6.1 QA role in test process

Understanding Requirements:

- Requirement specifications will be underpinned by the project team.
- Understanding of requirements will be conducted by all team members.

Preparing Test Cases:

- QA will be preparing test cases based on the exploratory testing. This will cover all scenarios for requirements. This will cover all scenarios for requirements.

Reviewing test cases:

- Peer review will be conducted for test cases by QA.
- Any comments or suggestions on test cases and test coverage will be provided by reviewer respective Author of Test Case.

- Suggestions or improvements will be re-worked by author and will be send for approval.
- Re-worked improvements will be reviewed and approved by reviewer.

Executing Test Cases:

- Test cases will be executed by the QA project member based on designed scenarios, test cases and test data.
- Test result (Actual Result, Pass/Fail) will updated in test case document.

Defect Logging and Reporting

QA will be logging the defect/bugs in Word document, found during execution of test cases. After this, QA will inform respective developer about the defect/bugs.

Retesting and Regression Testing:

Retesting for fixed bugs will be done by respective QA once it is resolved by respective developer and bug/defect status will be updated accordingly. In certain cases, regression testing will be done if required.

Deployment/Delivery:

• The website will not be deployed to a live environment as we will only be using one environment which is the development environment.

6.2 **Types of testing conducted**

- UI Testing

  The UI Testing will be completed using Jest and Automated snapshot tests will be able to be found within the frontend src directory. In addition, the frontend will also be tested via manual QA test scripts.

- Automated Tests

  The automated unit test scripts are present within both the front-end of the application and the back-end. In the back-end tests scripts test the logic of the application such as converting roman numerals to normal numerals (vice versa); in the frontend the tests are used to ensure that each component is rendered and rendered correctly. Smoke tests are also included within the documentation directory.

- CI/CD Pipeline

  Two GitHub Actions pipelines have been created, these run all of the automated tests within the project after a pull request has been created. This ensures that all of our tests pass prior to merging to the main branch will notifies us whether the entire application is still functional even after many changes.

- Accessibility and Performance Audit

  Microsoft Accessibility Insights for Web will conduct tests on the front end of the application, results can be found within the README.

6.3 **Bug Severity and Priority Definition**

Bug Severity and Priority fields are both very important for categorizing bugs and prioritizing if and when the bugs will be fixed. The bug Severity and Priority levels will be defined as outlined in the following tables below. Testing will assign a severity level to all bugs. QA will be responsible to see that a correct severity level is assigned to each bug. However, as we are adopting the Test-Driven-Development approach the possibility of having any existing bugs are minimal.

6.4 **Severity List**

The tester entering a bug into is also responsible for entering the bug Severity.

Severity ID	Severity	Severity Description
1	Critical	The module/product crashes or the bug causes no recoverable conditions. System crashes, GP Faults, or database or file corruption, or potential data loss, program hangs requiring reboot are all examples of a Sev. 1 bug.
2	High	Major system component unusable due to failure or incorrect functionality. Sev. 2 bugs cause serious problems such as a lack of functionality, or insufficient or unclear error messages that can have a major impact to the user, prevents other areas of the app from being tested, etc. Sev. 2 bugs can have a work around, but the work around is inconvenient or difficult.
3	Medium	Incorrect functionality of component or process. There is a simple work around for the bug if it is Sev. 3.
4	Minor	Documentation errors or signed off severity 3 bugs.

6.5 **Priority List**

Priority	Priority Level	Priority Description
1	Must Fix	This bug must be fixed immediately; the product cannot ship with this bug.
2	Should Fix	These are important problems that should be fixed as soon as possible. It would be an embarrassment to the company if this bug shipped.
3	Fix When Have Time	The problem should be fixed within the time available. If the bug does not delay shipping date, then fix it.
4	Low Priority	It is not important (at this time) that these bugs be addressed. Fix these bugs after all other bugs have been fixed. Enhancements/ Good to have features incorporated just are out of the current scope.

7 RESOURCE AND ENVIRONMENT NEEDS

7.1 **Testing Tools**

Process	Tool
Test case creation	Microsoft Excel
Test case tracking	Microsoft Excel
Test case execution	Manual
Test case management	Microsoft Excel
Defect management	Microsoft Word

7.2 **Product Device Accessibility**

Accessible to all screen resolutions, and mobile devising. Note that we will be using a single environment to conduct development and testing on.

8 TEST SCHEDULE

Task Name	Start	Finish	Effort	Comments
Test Planning	23.11	24.11	Complete	
Review Requirements documents	24.11	24.11	Complete	
UI Testing	1.12	8.12	Complete	
Automated Testing	8.12	N/A	Complete	Always occurring when running the code
Accessibility and Performance Audit	10.12	10.12	Complete	
Manual Unit Testing	13.12	13.12	Complete	
UAT Testing	14.12	14.12	Incomplete	

9 APPROVALS

	Scrum Master	**QA**
Name	**Sukh Virdi**	**Kacper Kiermasz**
Signature