SE 3XA3: Test Plan
Euneva

Team 9, Euneva
Mehta, Jash - mehtaj8
Sharma, Aditya - shara24
Ren, Zackary - renx11

March 5, 2021

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
04/03/2021	1.0	Initial version of Test Plan (all members)

1 General Information

1.1 Purpose

The purpose of this document is to provide a plan for testing, validation, and verification procedures for the Avenue linked To-Do list software being implemented. The program will need to interpret user input and present accurate output. These components will be tested to build confidence that the software project will function as intended.

1.2 Scope

This document outlines the tests that will be done for testing the functional and non-functional requirements and proof of concept. The scope of testing will primarily cover the accuracy of the output and logic.

1.3 Acronyms, Abbreviations, and Symbols

Table 2: Table of Abbreviations

Abbreviation	Definition
UI	User Interface
JSON	JavaScript Object Notation, a readable file format to transmit data objects
JS	JavaScript
CSS	Cascading Style Sheets
TS	TypeScript

Table 3: Table of Definitions

Term	Definition
Structural Testing	Type of testing carried out to test the structure of the code
Functional Testing	Type of software testing that validates the software system based on the functional requirements
Dynamic Testing	Type of testing that validates the behaviour and performance of the software system
Static Testing	Type of testing where the actual program or application is not used
Manual Testing	Type of software testing that involves the Testers to annually execute the tests
Automated Testing	Type of software testing that involves the use of special testing software that controls the execution of tests and compares the results to the expected
Unit Testing	Type of software testing where the individual units/components of the software are tested
System Testing	Type of testing where a complete and integrated software is tested from the internal structure
Integration Testing	Type of software testing where individual components are tested in groups
React	An Open-Source application framework
Group-9	Project Team

1.4 Overview of Document

This document will walk through the test plan for Euneva, this includes a plan for how the testing will be conducted, a system test description which goes over the tests for the functional and non-functional requirements, a plan for testing proof of concept, and a unit testing plan.

2 Plan

2.1 Software Description

This software will allow students from McMaster University to access Avenue to Learn assignments and quizzes for each class in one convenient location. The Front-End To-Do list application is made using the React web application framework, whilst the Back-End is implemented in Node. Python is used for the web-scraping done on Avenue to Learn.

2.2 Test Team

The test team will consist of Group-9 members involved in the development of the project. The test team members are, Jash Mehta, Aditya Sharma, Zackary Ren.

2.3 Automated Testing Approach

The overwhelming majority of our project's functionality consists of user interaction. Therefore we determined that it would be costly to create a suit of unit tests of all base logic. Instead, our testing technique will be a manual test suite that ensures the correct responses to the user's input. If each test passes it is implied that the underlying logical functionality of our program corresponds properly.

2.4 Testing Tools

The testing for this project will be done with PyTest, which is a testing suite framework for Python projects, which deals with unit testing and code coverage. As mentioned earlier, the Python component of our project is used for web-scraping; however, there are some functions that carry out tasks like data cleaning and data processing. These functions will be tested via PyTest.

2.5 Testing Schedule

See Gantt Chart for Testing Schedule.

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 UI

1. test-UI1

   Type: Functional, Automated
   Initial State: The app has been deployed
   Input: N/A
   Output: A boolean response
   How test will be performed: An automated test will be written that checks if a button to log into Avenue is rendered
   Mapping: FR1

2. test-UI2

   Type: Functional
   Initial State: The app has been deployed
   Input: N/A
   Output: A boolean response
   How test will be performed: An automated test will be written that checks if a button exists that adds an item to the list
   Mapping: FR2

3. test-UI3

   Type: Functional
   Initial State: The app has been deployed
   Input: N/A
   Output: A boolean response
   How test will be performed: An automated test will be written that checks if each item has a date assigned and displayed to it
   Mapping: FR3

4. test-UI4

Type: Functional
Initial State: The app has been deployed
Input: N/A
Output: A boolean response
How test will be performed: An automated test will be written that if the item name, due date, time and update button occurs when editing the item
Mapping: FR6

5. test-UI5

   Type: Functional
   Initial State: An item has been added
   Input: User marks item as completed via click
   Output: Item disappears from list
   How test will be performed: A visual test will be performed to confirm that the item that has been clicked is no longer being rendered
   Mapping: FR8, 10

6. test-UI6

   Type: Functional
   Initial State: An item has been added
   Input: Random user input to an item's name
   Output: An update in the items information
   How test will be performed: A visual test will be performed to confirm that when an item is edited, its changes are being reflected and rendered
   Mapping: FR9

3.2 Tests for Non-Functional Requirements

3.2.1 Look and Feel Requirements

1. test-LFA1

   Type: Functional
   Initial State: The web app has been loaded
   Input: N/A

Output: N/A
How test will be performed: A visual test will be performed to confirm that the product looks similar to Apple's Reminders Application
Mapping: LFA1, LFS1

3.2.2 Usability and Humanity Requirements

1. test-UHE1

   Type: Manual
   Initial State: The web app has been loaded
   Input: N/A
   Output: N/A
   How test will be performed: A interaction based assessment will be conducted by having an adult not involved in development use the product and provide their feedback in terms of learning to use the product and navigating the product
   Mapping: UHE1, UHP1, UHA1

3.2.3 Performance Requirements

1. test-PSL1

   Type: Automated, Functional
   Initial State: The backend of the application will be loaded
   Input: Requests will be sent to each route
   Output: The routes will retrieve and send back the data requested with the response time
   How test will be performed: The backend server will be initialized and requests to all routes will be sent to it. The routes will return the data along with the response time. The response time should be less than the RESPONSE_TIME
   Mapping: PSL1

2. test-PPA1

   Type: Automated
   Initial State: The app has been deployed

Input: A task is added to the list
Output: The correct information about the task is saved
How test will be performed: An automated test will add different tasks to the list and check if the correct information is saved
Mapping: PPA1

3. test-PRA1

   Type: Manual
   Initial State: The web app has been loaded
   Input: User interacts with the application
   Output: Application renders appropriate data on device
   How test will be performed: A tester will confirm the application can be accessed on a device and upon interacting, the expected behaviour is observed
   Mapping: PRA1, PRA2, PRF1

4. test-PCR1

   Type: Manual
   Initial State: The web app has been loaded
   Input: User interacts with the application
   Output: N/A
   How test will be performed: The application will be used by NUMBER_OF_USERS at the same time and the performance will be evaluated
   Mapping: PCR1

3.2.4 Operational and Environmental Requirements

1. test-OER1

   Type: Manual
   Initial State: The web app has been loaded
   Input: N/A

Output: N/A
How test will be performed: A tester will access the application on multiple different devices that can connect to the Internet
Mapping: EPE1, RIAS1, RR1, RR2

3.2.5 Security Requirements

1. test-SR1

   Type: Manual
   Initial State: The app has been loaded
   Input: User enters their log-in credentials
   Output: To-do list syncs with Avenue
   How test will be performed: A tester will enter the log-in information of verified McMaster Students and random log-in information to confirm only McMaster students have access to the application
   Mapping: AR1, AR2, IR1, PR1, PR2

3.2.6 Cultural Requirements

1. test-CR1

   Type: Manual
   Initial State: The app has been loaded
   Input: N/A
   Output: N/A
   How test will be performed: A tester will examine the application and ensure it is written in English and there are no inappropriate graphics or text
   Mapping: CPC1, CPC2

3.3 Traceability Between Test Cases and Requirements

The following table maps test cases to their respective requirements.

Table 4: Requirements Traceability Matrix

Requirement ID	Description of Fit Criterion	Test ID(s)
FR1	A button is rendered allowing the user to log-in to Avenue to Learn	UI1
FR2	A button allows the user to add a task to the to-do list	UI2
FR3	The applications displays relevant information pertaining to a certain date	UI3
FR6	Display confirmation message when a task is added	UI4
FR8, FR10	Task is removed from to-do list upon completion or deletion	UI5
FR9	The user can edit a task	UI6
LFA1, LFS1	The application will have an intuitive user flow and simplistic UI	LFA1
UHE1, UHP1, UHA1	The user can navigate throughout the application with ease	UHE1
PSL1	The application responds to user actions after RESPONSE_TIME	PSL1
PPA1	The information of a task accurately reflects information on Avenue	PPA1
PRA1, PRA2, PRF1	The application can be accessed on a device connected to the Internet	PRA1
PCR1	The application can support NUMBER_OF_USERS at a time	PCR1
EPE1, RIAS1, RR1, RR2	The application can be accessed on different types of devices connected to the Internet	OER1
AR1, AR2, IR1	Only McMaster students can access the application	SR1
CPC1, CPC2	The application displays text in English, without offensive graphics or text.	CR1

4 Tests for Proof of Concept

4.1 Application Testing

1. test-POC1

   Type: Functional, Dynamic, Manual
   Initial state: The web app has been loaded
   Input: The user interacts with the application
   Output: The application displays appropriate response when the respective action is performed
   How the test will be performed: The user will interact with different aspects of the application and the expected response will be visually confirmed

5 Unit Testing Plan

5.1 Unit Testing of Internal Functions

Internal functions of the program will be methods that will have return values or execute a specific task. Unit tests for internal functions will involve having various inputs for the methods, and comparing the methods output to the expected output. Unit tests will include various inputs for methods consisting of normal inputs, boundary inputs, and inputs that will generate exceptions and errors for the program. The unit tests will display test cases that have passed, and test cases that have failed with appropriate reasoning behind the failure. Coverage methods, such as code coverage and branch coverage, will be used to determine how much of the program has been covered by the unit testing.

5.2 Unit Testing of Output Files

The application does not produce an output file. Therefore, there is no requirement to perform testing on an output file.

6 Appendix

6.1 Symbolic Parameters

SP1. **RESPONSE_TIME**: 2 seconds
SP2. **NUMBER_OF_USERS**: 50

6.2 Usability Survey Questions

We want to question users on their experience with our application to gauge whether or not it accomplishes our goal of alleviating some stress from academics.

1. Did you have any difficulties navigating throughout the application?
2. Did the user onboarding help with navigation around the application?
3. Did the use of the application reduce academic related stress?
4. Did it take long to get familiar with the application? If so, what felt unnatural?
5. Were there any features you wish were added?