SE 3XA3: Test Report
Euneva

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

April 3, 2021

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
31/03/2021	1.0	Initial version of Test Report (all members)

This document describes the test report generated after executing the Test Plan for Euneva.

1 Functional Requirements Evaluation

1.1 User Interface

1. test-UI1

   Type: Functional, Manual
   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
   Result: PASS. The test confirmed a button enabling Avenue connection is rendered on the to-do list.

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
   Result: PASS. The test confirmed a button allowing the user to add an item to the list is rendered.

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

Result: PASS. Each item added to the to-do has an associated date and is displayed properly.

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
   Result: PASS. When adding a new item to the to-do list, the user has the option to enter the item name, due date, time and confirm the changes.

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
   Result: PASS. When an item is checked off as completed, it no longer appears on the to-do list.

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
   Result: PASS. After performing an edit to an item, the changes are appropriately reflected on the to-do list.

1.2 Scraping Functionality (SF)

1. test-SF1

   Type: Functional, Manual
   Initial State: The app has been deployed
   Input: User credentials for Avenue to Learn
   Output: Console log, 'logged on' displayed
   How test will be performed: A visual test will be performed to see if the correct value was displayed in the console.
   Result: PASS. After entering the correct user credentials for Avenue to Learn, a successful log-in message was logged in the console.

2. test-SF2

   Type: Functional, Manual
   Initial State: The app has been deployed
   Input: N/A
   Output: Console log, 'Filtered Courses' displayed
   How test will be performed: A visual test will be performed to see if the correct value was displayed in the console.
   Result: PASS. Upon logging in to Avenue and filtering courses, only the courses that are currently being taken were logged to the console.

3. test-SF3

   Type: Functional, Manual
   Initial State: The app has been deployed
   Input: N/A
   Output: Console log, 'Assignment info collected' displayed
   How test will be performed: A visual test will be performed to see if the correct value was displayed in the console.
   Result: PASS. The assignments of a particular course were requested. The output logged to the console was checked against the assignments seen on Avenue.

4. test-SF4

Type: Functional, Manual
Initial State: The app has been deployed
Input: N/A
Output: Console log, 'Quiz info collected' displayed
How test will be performed: A visual test will be performed to see if the correct value was displayed in the console.
Result: PASS. The quizzes of a particular course were requested. The output logged to the console was checked against the quizzes seen on Avenue.

5. test-SF5

Type: Functional, Manual
Initial State: The app has been deployed
Input: N/A
Output: Console log, 'Filtered Information' displayed
How test will be performed: A visual test will be performed to see if the correct value was displayed in the console.
Result: PASS. After filtering the current tasks of an individual, only tasks with future due dates and an incomplete status remained.

6. test-SF6

Type: Functional, Manual
Initial State: The app has been deployed
Input: N/A
Output: The correct information is placed in the database and displayed on the UI
How test will be performed: A visual test will be performed to see if the correct information is displayed.
Result: PASS. After executing the Avenue scraping script, the database was correctly populated with the filtered information.

2 Nonfunctional Requirements Evaluation

2.1 Look and Feel

1. test-LFA1

Type: Manual
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
Result: PASS. The to-do application was loaded and compared to Apple's Reminders application and the visual appearance was deemed similar.

2.2 Usability and Humanity

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
Result: PASS. A tester who had no prior experience with the to-do app was presented with the app. They managed to figure out the features intuitively and could use the app as intended.

2.3 Performance

1. test-PSL1

Type: Manual
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.
Result: PASS. The contents of various courses was requested, the response time was logged to the console and confirmed that it remained lower than RESPONSE_TIME.

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
   Result: PASS. A variety of different tasks were added using the to-do list. The database was checked to confirm the item was successfully saved.

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.
   Result: PASS. The test team accessed the app on different devices and after interacting with the app, confirmed that it is behaving as expected.

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.
Result: PASS.

2.4 Operational and Environmental

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
   Result: PASS. Each tester could successfully access the application on different devices connected to the Internet.

2.5 Security

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.
   Result: PASS. Entering the log-in credentials of a verified McMaster Student allowed access to the application. An incorrect log-in did not allow the individual to use the application.

2.6 Cultural

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.
Result: PASS. The application was inspected and all text in the application was in English. Additionally, no inappropriate or offensive text/graphics were found.

3 Comparison to Existing Implementation

The existing implementation of the to-do list was built fine with modules exhibiting low coupling and high cohesion. The existing modules each had their own individual function and were not strongly related to one another. Each of the components of the user interface was clearly modularized, as such we did not have to modify any of the user interface components. We decided to introduce the ability to connect to Avenue to Learn and developed our own modules for this feature. To connect this feature to the application itself, we added two text fields and a button to the user interface allowing the user to enter their login credentials. Additionally, a task in the existing implementation did not have a date associated with it. We decided to add a date field as we felt a to-do list should keep track of the due date of a task.

4 Unit Testing

Our team made the decision to forego unit testing due to the increased resources required for little to no gain. Since our app primarily focuses on the user interface and user experience, we opted to create a suite of manual tests to validate the various aspects of our app.

5 Changes Due to Testing

5.1 UI

No changes were made due to testing

5.2 Scraping Functionality

No changes were made due to testing

5.3 Look and Feel

No changes were made due to testing

5.4 Usability and Humanity

No changes were made due to testing

5.5 Performance

No changes were made due to testing

5.6 Operational and Environmental

No changes were made due to testing

5.7 Security

No changes were made due to testing

5.8 Cultural

No changes were made due to testing

6 Automated Testing

A couple of the tests defined in the test plan were to be automated; however, during testing it was deemed infeasible. Given the time constraints, the team decided that the additional time investment required to set up the automated tests was not worth it. Instead we decided to perform manual tests and visually confirm our app was displaying the correct output.

7 Trace to Requirements

Table 2: Requirements Traceability Matrix

Requirement ID	Test ID(s)
FR1	UI1
FR2	UI2
FR3	UI3
FR4	UI4
FR5, FR7	UI5
FR6	UI6
FR8	SF1
FR9	SF2
FR10	SF3
FR11	SF4
FR12	SF5
FR13	SF6
LFA1, LFS1	LFA1
UHE1, UHP1, UHA1	UHE1
PSL1	PSL1
PPA1	PPA1
PRA1, PRA2, PRF1	PRA1
PCR1	PCR1
EPE1, RIAS1, RR1, RR2	OER1
AR1, AR2, IR1	SR1
CPC1, CPC2	CR1

A description of the requirements can be found in the SRS.

8 Trace to Modules

The latest Module Guide can be found here.

Table 3: Module Traceability Matrix

Test ID	Module(s)
UI1	M1
UI2	M1
UI3	M1
UI4	M1
UI5	M1
UI6	M1
SF1	M2
SF2	M3
SF3	M4
SF4	M5
SF5	M5
SF6	M6
LFA1	M1
UHE1	M1
PSL1	M3, M6
PPA1	M3, M6
PRA1	M1, M2, M3, M6
PCR1	M2
OER1	M1, M2, M3, M4, M5, M6
SR1	M1, M2, M3, M4, M5, M6
CR1	M1

9 Code Coverage Metrics

The team decided against using code coverage due to the fact that all the tests being performed are all manual tests. Due to not having any automated test

cases another way we confirm that the code coverage was met was through the examination of the manual tests and the traceability matrix between modules. This makes it clear that the important functionality has been tested and the team feels confident the software is reliable and consistent.

10 Appendix

10.1 Symbolic Parameters

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