Test Plan for Study Buddy (Version 1.0)

- **PROJECT NAME:** Study Buddy
- **DATE:** November 27, 2022
- **PREPARED BY:** DNM

Table of Contents

1. DESCRIPTION OF TESTING METHODOLOGIES

1.1 CLOSED BOX TESTING WITH SELENIUM:

We will be using Selenium to interact with the application in a way that mimics the behaviour of a user upon the elements of the website. Selenium acts on the web elements within a website, eg. Ids, to script testing functionality and allow testing of edge cases upon a product. Through a comprehensive set of tests using selenium scripts, a test environment that closely replicates that of an end-user is created. Closed box testing involves comparing the functionality of a product to its requirement documents on a high level; it compares the product to the requirements as specified in the requirements document. Some features of Selenium that make it an ideal tool for closed-box testing include:

- Multi-Browser support
- Multi-language compatibility
- Methods to test dynamic web elements
- Able to work with different operating systems
- WebDrivers; a list of classes and methods that aid the debugging in the testing automation and aid testing complex web elements
- Automation of testing scripts

2. TESTING INFRASTRUCTURE

2.1 CLASSES AND METHODS

Selenium provides a variety of classes and methods that can be integrated with the use of drivers to link the code base of the product to the testing scripts. Some of these include;

- Libraries for testing purposes
- Selenium drivers can be connected to test cases for the automation to take place
- Specify the name of the driver and the location of the driver
- Create references variable to interact with the driver

- Import the drivers into our tests cases
- The find element method is used to find the element by its ID
- ID name, implemented in the web elements, is used to connect the reference variable of the selenium driver to access and test that specific web element
- The driver.sendKeys{“any key you want to send”} method can be used to send specific strings to the web element
- Elements can also be found by their class name
- The click{} method can be used to simulate the effect of clicking on that web element
- There are other methods like driver.getTitle{} to fetch information from specific web elements
- The web page can be closed directly from selenium using the driver.close{} method
- The console provided by the web browser can be utilised to check test results or they can be outputted to a specific file
- The thread.sleep{“time unit to sleep”} method can be used to slow down the testing to see how each test case executes

3. TEST CASES

3.1 FRONT END TESTING

Testing for the GUI will consist of replicating the way a user interacts with a product. The Test cases sequentially deal with typical and atypical entries for each instance of user input to the app.

Phase A: User is not logged in.

Phase A1: User attempts to log in.

- Test case 1: User presses Login Button and enters correct login info
- Test case 2: User presses Login Button and enters incorrect login info

Phase B: User is logged in.

Phase B1: User attempts to add a Group

- Test case 1: Name field is blank
- Test case 2: Name is whitespace
- Test case 3: Name field contains 1 character
- Test case 4: Name field contains 10 characters
- Test case 5: Name field contains 50 characters
- Test case 6: Name field contains 100 characters
- Test case 7: Name field contains invalid characters (injection issues)
- Test case 8: Name is the same as an existing Group

- Test case 9: Colour not selected

Phase B2: User attempts to add a Task

- Test case 1: Name field is blank
- Test case 2: Name is whitespace
- Test case 3: Name field contains 1 character
- Test case 4: Name field contains 10 characters
- Test case 5: Name field contains 50 characters
- Test case 6: Name field contains 100 characters
- Test case 7: Name field contains invalid characters (injection issues)
- Test case 8: No valid Groups to select from
- Test case 9: No valid Type to select from
- Test case 10: No Group selected
- Test case 11: No Type selected
- Test case 12: No Date selected
- Test case 13: Invalid date is selected (before current date)

Phase B3: User attempts to delete a Task

- Test case 1: Delete First Task
- Test case 2: Delete Middle Task
- Test case 3: Delete Last Task
- Test case 4: Delete All Tasks
- Test case 5: Attempt to Delete a task when the list is empty
- Test case 6: Attempt to Delete task with duplicate names

Phase B4: User attempts to delete a Group

- Test case 1: Delete First Group
- Test case 2: Delete Middle Group
- Test case 3: Delete Last Group
- Test case 4: Delete All Groups
- Test case 5: Attempt to Delete a Group when only default Group exists
- Test case 6: Attempt to Delete Group with duplicate names
- Test case 7: Delete a Group which has tasks assigned to it

Phase B5: User attempts to mark a Task as completed

- Test case 1: Mark First Task
- Test case 2: Mark Middle Task
- Test case 3: Mark Last Task
- Test case 4: Mark All Tasks
- Test case 5: Attempt to Mark a task when list is empty
- Test case 6: Attempt to Mark task with duplicate names

Phase B6: User Logs out

- Test case 1: User selects the logout button

4. INTERNAL TEST ENVIRONMENT

4.1 TESTING WITH NPM

Testing for the back end will consist of testing the internal process of the product that utilises Firebase. Without knowledge of working on a JavaScript test environment and everything required to ensure the functions work properly, we feel we would require an environment based on NPM.

NPM forms the basis for most of the test tools we would be using. It is required for this test setup to have NPM running the test environment. NPM is short for Node Package Manager. Alongside this, Node.JS is also required within the test environment. After installing these two pieces of software, a Firebase NPM library is also required to be able to locally run test environments. Lastly, NPM testing is required to be installed.

To test through the JavaScript code, with the use of the NPM testing library, unit tests can be created that can deal with multiple test cases. This library allows for a large degree of automation of the test environment. In addition, API calls can be automated within this environment. While they are not used within our program, another program using a different database structure, or moving to a cheaper database solution would require this part of the testing library. To create a local test environment for small-scale testing without hosting anything, the ideal tools to use would be NPM to create and host a Firebase instance with the use of Firebase tools. This test environment allows every single function made within the application to be tested thoroughly.

Without further reading into the documentation of these libraries that are required to test this application, the above offers a sample of what a test environment might look like. With more time, we hope to explore some elements of testing automation for this product after the implementation of this class. But to be able to match the time requirement (the true hard requirement for the project) we must keep this mostly conceptual.

5. Sign-Offs

Signature:	*David Botero*	Date:	10/21/2022
Print Name:	David Botero		
Title:	Student		
Role:	Test Lead, UI Co-Lead,		

Signature:	*Joshua Lade*	Date:	10/21/2022
Print Name:	Joshua Lade		
Title:	Student		
Role:	UI Lead, Test Co-lead		

Signature:	Hayden Lister	Date:	10/21/2022
Print Name:	Hayden Lister		
Title:	Student		
Role:	Contact Person, Project Management		

Signature:	Abhay Parashar	Date:	10/21/2022
Print Name:	Abhay Parashar		
Title:	Student		
Role:	Version Control Lead, Dev Lead, Project Management Co-lead		

Signature:	Robert Stewart	Date:	10/21/2022
Print Name:	Robert Stewart		
Title:	Student		
Role:	Documentation, Dev Co-Lead		