Software Testing Plan

Presented By: Team HealthLit

Devin O'Neal

Leslie Gurrola

Brendan Tjeerdsma

Ahmir Hughes

Project Sponsor: Dr. Olivia Lindly

Team Mentor: Han Peng

Project Title: Parent Health Literacy Mobile App

April 3rd, 2022

Version 2.0

Contents

1 | Introduction

Our team built the HealthLit application to help parents of young children learn skills that are crucial to the proper care of their children. HealthLit uses health literacy testing and interactive modules to teach parents these skills and then reinforce them. This, combined with easy access to information that is easy to understand, will ensure that these parents retain the information given to them. In turn, this will prevent many of the common accidents that occur with young children.

Software testing is done to ensure that the software that has been created meets the standards outlined in the requirements document, as well as the standards agreed to by the client and development team. This testing aims to ensure that each part of the software works as intended, both on its own and in conjunction with the other parts of the software. Software testing must be rigorous to ensure that the highest quality software is presented to the client.

Our team's software testing will consist of three main parts, with those being unit testing, integration testing, and usability testing. Unit testing will consist of tests for the web and mobile applications, ensuring that all of their components perform as expected. Next, integration testing will consist of tests to ensure that the separate components of the app can work together as expected. Finally, usability testing will consist of tests designed to gauge how well naive users can navigate the app. All of these tests will ensure that our application functions as specified. Next, we will discuss our unit testing strategies.

2 | Unit Testing

Unit testing is testing individual parts of our code by isolating sections of our project to maximize efficiency of use. For unit testing we will test both the web and mobile application individually, using controlled input to test out the performance of each part from the average user's perspective. We will be testing the performance of the page setups and correct movement throughout our applications. To further our testing we will be using the Flutter's unit testing package that is readily available to download from the internet. The unit testing package allows users to create test files for the classes made for our applications, with these testing files working independently from the actual performance of the application. We will be focusing our testing on receiving and retrieving data to make sure that there is no lack of performance with communication between our two components – the web and mobile applications – along with our database.

For our mobile application we will be testing:

- Module Setup
- Module Completion
- Sign-In/Up Response
- Game Response

For our web application we will be testing:

- Module Upload
- Database Retrieval & Filtering

2.1 | Mobile Application

This section will outline the main functionalities of the mobile application that we will be testing. Most of the topics we will cover are the responses from the database to the mobile application, however we will also test the components within our application, such as our games.

Module Setup:

Modules for our mobile application act as the functional component of our app, bringing in the information of a certain topic such as Food Nutrition or Medication Dosing to the app's user for them to read about. The modules will contain a text document full of information that has been provided by our client, Dr. Lindly. This information will contain the title, pictures, and module text needed for the addition of any new module. The user should be able to click the 'open' button to get into the modules, then the modules should contain the corresponding information as mentioned previously. There should be a correlating game that is customized to each module, with any pictures needed. The modules should also be formatted and displayed according to the design that our team has agreed upon.

- Module 1 Unit Testing:
  - Select Module from list of Modules
  - Test scrolling feature to make sure that it contains all of the information
  - Click on game button
  - Flip through game see the response to mimic the use of app use
  - Exit out of game in the middle of use to test the response of trying to go back home
- Module 2 Unit Testing:
  - Select Module from list of Modules
  - Test scrolling feature to make sure that it contains all of the information
  - Click on game button
  - Click one wrong answers to see the response of game
  - Fail test to see if there is a lag of getting information

- Module 3 Unit Testing:
    - Select Module from list of Modules
    - Test scrolling feature to make sure that it contains all of the information
    - Click on game button
    - Test game by clicking on different components to see response

Module Completion:

Unlike *Module Setup*, module completion will have its own testing properties due to the statistical need of transferring each mobile user's data across the database and viewing it from the web application. The module completion will be tested by using a corresponding user variable that will store a boolean that will trigger if the application user has completed the module. The boolean value will be true if completed and false if the module is not completed. The variable will be able to track all the current modules and be able to correspond to the uniqueID of the user.

Sign-In/Up Response:

For this stage of testing, we will first create an account with an email address that has not been used. After filling out all the sign-up information, we will then need to check with the database or web application if all the information of the 'new' mobile application user has been successfully uploaded. Then we will login using the newly created account and once logged in the account will have access to the module selection page. On the other hand, if they are not successful, then they will be notified that there is no account information based on their inputs and they will not be signed in and they will remain on the login page.

Game Response:

In terms of testing the gamified sections and NVS tests in the application, we will need to check certain variables that are utilized inside the files in order to make sure that they are what they're supposed to be. When the user does certain actions in the game section, certain variables will be modified accordingly in order to simulate the functionality of the game. Essentially, we will make unit tests that will simulate certain actions by the user and the variables (typically integers and arrays) hold values that would have contents that would only happen if that action takes place. The NVS will contain a couple of unit tests that will test if the user got a certain amount of questions correct based on the responses they submit. This will be done by having a unit test simulate inputs that the user would do when actually taking the test and checking the status of the game's variables. There will also be an edge case unit test for this to ensure that certain questions are skipped if they answer some questions in a certain way.

The first module game is a simple flashcard game where the user can click through different cards to review terms and definitions. Due to the simplicity of this game, the only unit test that will be done here is one that will test whether the contents of the array contain the correct answer. The next module involves a quiz the user has to take where they cannot move on to the next question until they choose a "correct" answer. A unit test will be done to test that if the user gets a question wrong, the question number variable will not increase. The third module game involves the user utilizing a slider in order to get the proper medicine dosage for each problem. Unit tests will be implemented to ensure that the function that calculates the correct medicine dosage is accurate and to solve any edge cases as needed.

2.2 | Web Application

This section will outline the main functionalities of the web application that we will be testing. Because the web application works in tandem with the database, all of the topics involve writing or reading from the database.

Module Upload:

The web application is designed to upload text into the database. For the purposes of testing, we need to make sure that not only does the Module Text transfer correctly into the database, but also the module title correctly gives a unique title in numerical counting order in relation to the pre-existing modules. In order to test this, the unit test will create a module with sample text and then read the database after the creation in order to test that the sample text transferred correctly and that the module title is correct.

Database Retrieval & Filtering:

The web application is also designed to obtain and display user data, while being able to filter it based on specific filter types. For the purposes of testing, we need to make sure the information read is correct, and that each filter works as intended, along with multiple filters being able to work. In order to test this, the unit test will read the database information and verify that the information read by the application is the same information read by the unit test. To test each filter, the unit test will set each filter and verify that the filters change the results each time as intended by checking if the results match what the filter is intended to do. There are a total of six filters to be tested, so not only does each filter need to be tested separately, but each combination of filters needs to be tested too, creating 63 unique tests.

3 | Integration Testing

3.1 | Mobile Application and Database Integration

The mobile application is responsible for communication with the database and the web portal. The database acts as a middle man where it receives and gives data to the mobile application. While the mobile application does not immediately interact with the web application, the database acts as its proxy.

Our database is a Google Firestore database, this allows the two end points to be able to interact with each other using the same database. The web application is able to view and parse through application users, and upload modules to be transferred to the database and therefore to the mobile application. To test these components in integration testing, we will first verify that a new user's information is readily available to view from the web application. If viewing the new mobile application user is successful then that means that the transferring of user's data across the components is correctly done.

The second component that falls under the mobile application and database is the successful integration of creating a module through the web application. As mentioned above, another feature of the web application is the ability to create new modules. Once all the information of the module components are successfully uploaded to the web application then that should be able to be viewed from our mobile application.

Another component that is under the mobile application is data collection for the database. There are certain aspects of the application that need to grab certain statistics from the user when they use the application. An example of this is figuring out how long a user takes on a module. We will need to be able to test that the data can be successfully recorded in the application and sent into the database for the web application to utilize.

4 | Usability Testing

The last kind of testing that we will implement is usability testing, which involves potential users of our software. The goal of usability testing is to ensure that a naive set of users can use the application as intended and achieve expected results. Users are selected and guided through a certain number of use cases, and their responses to the tasks are recorded along with their feedback. This type of testing helps to ensure that the application works as intended and successfully reaches our target audience.

To execute usability testing for our application, we will find a sample size of naive users to present our application to. Ideally, these users will be a part of our targeted demographic of parents. After this population is found, we will present them with the application and guide them through several use cases, recording their results and comments. These use cases will consist of activities such as logging into the app, creating an account, and completing a number of modules. After this, we will refine our application and continue these tests until we are satisfied with the application's performance. We hope to complete these tests by April 15th, 2022, so that we can effectively implement any changes to the application.

5 | Conclusion

To summarize, extensive unit, integration and usability testing will ensure that our application meets all of our requirements and satisfies our target userbase. We have outlined our plans for these three types of testing in this document, and we believe that these tests will accurately gauge how well we have implemented the required features into our application, and in the case of usability testing, give us good feedback to implement. The goal of the HealthLit project is to educate parents so that they can care for their young children. Part of reaching this goal is ensuring that the work we have done guides us there, and these tests will validate all of our hard work. Currently, our team is preparing to begin the testing process on our application and preparing our final version for presentation to our peers. Our team is excited to complete development of this application and share it with the world.