LAMI Full-Stack Application

Project Testing and Acceptance Plan

Logo

Description automatically generated

**LAMI Team**

Jacob Smith and Jerrel Chapman

10/24/2022

**TABLE OF CONTENTS**

[**I.**](#_gjdgxs) **Introduction 4**

[I.1.](#_30j0zll) Project Overview 4

[I.2.](#_1fob9te) Test Objectives and Schedule 4

[I.3.](#_3znysh7) Scope 4

[**II.**](#_2et92p0) **Testing Strategy 4**

[**III.**](#_tyjcwt) **Test Plans 4**

[III.1.](#_3dy6vkm) Unit Testing 4

[III.2.](#_1t3h5sf) Integration Testing 4

[III.3.](#_4d34og8) System Testing 4

[III.3.1.](#_2s8eyo1) Functional testing: 4

[III.3.2.](#_3rdcrjn) Performance testing: 4

[III.3.3.](#_26in1rg) User Acceptance Testing 5

[**IV.**](#_lnxbz9) **Environment Requirements 5**

[**V.**](#_1ksv4uv) **Glossary 5**

[**VI.**](#_44sinio) **References 5**

# Introduction

## Project Overview

The software that will be tested is the LAMI full-stack medication reminder application. Some components of this application will be tested like the apps alert component where we will test the ability to alert the user of medication times and see if they can be reminded of multiple medications on the same time. Also, to test the sig in feature to make sure the database is registering people correctly. So that the residents and caretakers can access their accounts.

## Test Objectives and Schedule

We plan to test the soon, however at present we would only be testing things that were tested by last years team that worked on this project. There will be tests, but it will be mostly edge case tests that the previous tests didn’t tackle. The test will most likely be created by some time next month when we get farther into our implementations of the application. When the tests are finished, we will output a code file that will be for testing features and most likely a video displaying some full application tests.

## Scope

The purpose of this document is to give insight into what we are going to test and how we will test those things. Through this document we will discuss testing strategies we will be implementing to test functions and implementation of our code. However, as previously started we will most likely nit be making test since much of the logical code is implemented from last years LAMI team. So, we will be only adding minimal tests since the majority of what we will work on is UI based which is hard to test.

# Testing Strategy

Describe the overall approach to testing and provide the overall flow of the testing process. An example is provided in Appendix A.

Will you be using Continuous Integration (CI) and/or Continuous Delivery (CD) in your testing? If you’re not using CI or CD, make a \*very\* strong case for your decision.

# Test Plans

Describe the plan for testing your project in the context of the following testing activities. You may include additional test activities, if necessary.

For each of the following activity, describe how the testing will be conducted. What would be the sequence of events, and how will the testing activity take place? Please refer to the CptS422 class notes for details on testing strategies.

## Unit Testing

The methods for the following functional requirements for each client will be unit tested using the Flutter test package:

House Residents

* + Log-in to an existing account
  + Sign-in up a new account
  + Add an alarm
  + Turn an alarm on/off
  + Edit an alarm
  + Delete an alarm

Staff

* + View a user’s alarms
  + View all existing user accounts

## Integration Testing

A Flutter package for integration testing exists that allows us to test each widget in the application. With this testing framework, rightfully named integration\_testing, we will test each button by triggering it and checking for the correct response.

## System Testing

## Functional testing:

As stated previously, the major functional testing requirements for each client are as

follows:

House Residents

* + Sign-up for a new account
  + Log-in to an existing account
  + Add an alarm
  + Turn an alarm on/off
  + Edit an alarm
  + Delete an alarm

Staff

* + View all existing user accounts
  + View a user’s alarms

## Performance testing:

To test the performance of the application, we will register enough accounts to ensure the free Firebase Firestore database plan can handle the quantity of accounts needed for all residents at the house.

## User Acceptance Testing:

We will test the app by using it on our personal phones and computers for an extended period, as well as messing with it to try and cause errors. In addition, we are also able to post the web app early, to let the staff look at and use both the resident and staff sides of the application. We are also able to do this for free, as every Firebase project is provided with a free subdomain on the web.app and firebaseapp.com domains.

# Environment Requirements

Specify both the necessary and desired properties of the test environment. The specification should contain the physical characteristics of the facilities, including the hardware, communications and system software, the mode of usage (for example, stand-alone), and any other software or supplies needed to support the test. Identify special test tools needed.

To do our tests we will only need the phones and the website connected to the application so that we can test our implementation. By having both the android and the iPhone we can test both phones application to make sure there isn’t any problems with one or the other. Then the website will need a computer component to be able to be tested so a laptop or computer will be needed to run these tests for the website application.

# Glossary

# References

**Appendix-A**

**Example Testing Strategy:**

1. Identify the requirements to be tested. All test cases shall be derived using the current Software Requirements Specification.
2. Identify which particular test(s) will be used to test each module.
3. Review the test data and test cases to ensure that the unit has been thoroughly verified and that the test data and test cases are adequate to verify proper operation of the unit.
4. Identify the expected results for each test.
5. Document the test case configuration, test data, and expected results.
6. Perform the test(s).
7. Document the test data, test cases, and test configuration used during the testing process. This information shall be submitted via the revised Test Plan document.
8. Successful unit testing is required before the unit is eligible for component integration/system testing.
9. Unsuccessful testing requires a bug form to be generated. This document shall describe the test case, the problem encountered, its possible cause, and the sequence of events that led to the problem. It shall be used as a basis for later technical analysis.
10. Test documents and reports shall be submitted. Any specifications to be reviewed, revised, or updated shall be handled immediately.