Test Plan: Time Estimation App

Test Plan Identifier

This test plan shall be referred to as TEA-Test-Plan-v1.0.
Audrey Eley (contact: aeley2020@my.fit.edu) prepared this version on Sept. 26, 2024.
This is the first version of the document, and it has sustained no revisions.

References

[1] Eley, Audrey. "Software Requirements Specification for Time Estimation App." Florida Institute of Technology, Senior Design 2024-2025, September 2024.
[2] V. Novak, “Testing · React Native,” *reactnative.dev*, Aug. 15, 2024. Available: https://reactnative.dev/docs/testing-overview. [Accessed: Sep. 27, 2024]
[3] “Overview,” *Postman Learning Center*, Oct. 19, 2023. Available: https://learning.postman.com/docs/introduction/overview/. [Accessed: Sep. 27, 2024]
[4] “Introduction to Firebase Local Emulator Suite,” *Firebase*, Sep. 23, 2024. Available: https://firebase.google.com/docs/emulator-suite. [Accessed: Sep. 27, 2024]
[5] “Firebase Test Lab | Firebase,” *Firebase*, 2019. Available: https://firebase.google.com/docs/test-lab. [Accessed: Sep. 27, 2024]
[6] “Appium Documentation - Appium Documentation,” *appium.io*, 2012. Available: https://appium.io/docs/en/2.1/. [Accessed: Sep. 27, 2024]
[7] “The Selenium Browser Automation Project,” *Selenium*, 2024. Available: https://www.selenium.dev/documentation/. [Accessed: Sep. 27, 2024]
[8] “Firebase Realtime Database,” *Firebase*, 2019. Available: https://firebase.google.com/docs/database. [Accessed: Sep. 25, 2024]
[9] “Jira Documentation | Atlassian Support | Atlassian Documentation,” *confluence.atlassian.com*, 2024. Available: https://confluence.atlassian.com/jira/jira-documentation-1556.html. [Accessed: Sep. 27, 2024]
[10] “Introduction · React Native,” *reactnative.dev*, Aug. 15, 2024. Available: https://reactnative.dev/docs/getting-started. [Accessed: Sep. 25, 2024]
[11] “Documentation | Firebase,” *Firebase*, 2019. Available: https://firebase.google.com/docs. [Accessed: Sep. 25, 2024]

Introduction

This document aims to establish the features to be tested, the testing approach, the procedures, and the expected deliverables associated with the testing of the Time Estimation App. The execution of the instructions provided herein shall ensure the fulfillment of requirements specified in the Time Estimation App’s SRS document [1].

The scope of this document shall be limited to the testing of the interface, feature, and non-functional requirements of the application on iOS, Android, and Windows platforms.

Test Items

The following are the items to be tested. These are specific requirements organized by the categories from the SRS document [1]. The remaining functional requirements from the SRS document will be implicitly tested through the testing of these items.

1. User Authentication
    a. ID: UAR1
       Title: User Registration
    b. ID: UAR2
       Title: Login Requirement
    c. ID: UAR3
       Title: Logout
2. Record Time Estimates
    a. ID: RTER1
       Title: Input Task Information
    b. ID: RTER2
       Title: Store Task Information
3. Track Time to Task Completion
    a. ID: TTTCR1
       Title: Timer Display
    b. ID: TTTCR2
       Title: Timer Data Storage
    c. ID: TTTCR3
       Title: Timer Record Comparison to Time Estimate
4. Customizable Interface
    a. ID: CuIR1
       Title: Task Organization
5. Display Analytics
    a. ID: DAR1
       Title: Time Comparison Visualization
    b. ID: DAR2
       Title: Progress Report Visualization
6. Software Interfaces
    a. ID: SWIR1
       Title: Firebase Realtime Database Storage
    b. ID: SWIR3
       Title: Chart.js Analytical Graphics

Features To Be Tested

From a high-level perspective, these are the features that will be tested according to this plan. In each case, the role of the user is that of a student.

1. Create an account

2. Login to account
3. Logout of account
4. Enter tasks with a time estimate
5. Time tasks
6. View predicted versus actual time to complete tasks
7. View previous tasks
8. Change the organizational layout of the screen
9. View graphics of task data analytics

Features Not To Be Tested

Non-functional requirements relating to software performance and quality will not be tested according to this plan. These are considered to be outside the scope of this testing cycle but may be tested in the future. From a high level, this means that quick loading speeds and a guaranteed amount of uptime for the app will not necessarily be fulfilled.

Approach

1. Testing Levels
    a. Unit Testing
        - Objective: Verify the functionality of individual program units.
        - Methodology:
            - Test individual functions in isolation.
        - Coverage: Each functional component (e.g. registration, timer, view tasks) of the app shall be subject to unit testing.
        - Tools: The built-in testing framework for React Native (Jest) shall be used to write and execute cases for unit testing [2].
    b. Integration Testing
        - Objective: Verify that program components interact correctly.
        - Methodology:
            - Test interaction between the frontend and backend for:
                - Data storage
                - Authentication
            - Test the interactions that the visualization library (Chart.js) has with the UI and the backend.
        - Coverage: Each instance of components interfacing with one another shall be tested.
        - Tools: The built-in React Native Debugger will be used to detect defects in the frontend [2]. Postman may be used to test the interactions between the frontend and backend [3]. Firebase Emulator will be used to test the

functionality of the backend in an emulated environment, and Firebase Test Labs will be used for testing the app on actual devices [4, 5].

c. System Testing
    - Objective: Verify the fulfillment of requirements.
    - Methodology:
        - Use dynamic testing of use cases to verify requirement fulfillment.
        - Establish the functionality of each feature across all supported platforms (iOS, Android, and Windows).
    - Coverage: Each requirement specified in the Test Items list shall be verified using system testing.
    - Tools: Appium shall be used to perform system testing across iOS and Android, and Selenium will be used for Windows testing [6, 7].

d. Regression Testing
    - Objective: Ensure that changes to the program do not damage existing, functional code.
    - Methodology:
        - Re-run tests of functionalities after the addition of each new functionality.
        - Re-run tests of functionalities after each bug fix.
    - Coverage: All functionalities that are detailed in the “features to be tested” section.
    - Tools: Jest and Appium shall be used for automated testing.

2. Testing Methods
    a. Black Box Testing
        - Objective: Test program requirements from a user perspective by testing complete use cases without viewing the internal construction of the program.
        - Methodology:
            - Write test cases with the intention of testing requirements.
        - Tools: Testing will primarily be performed manually. Appium may be used for automated functional and UI testing.

    b. White Box Testing
        - Objective: Test the program by examining its internal structure and logic.
        - Methodology:
            - Statement coverage: Each line of code is reviewed and tested at least once.
            - Condition coverage: Coverage of every condition is guaranteed by testing each boolean expression in the program for all possible true and false outcomes.

            - Loop testing: The functionality of each loop in the program is analyzed and tested.
        - Tools: Jest and the React Native Debugger will be used to facilitate white box testing.
3. Test Data
    a. Manual Data Creation
        - Objective: Create data for testing purposes.
        - Methodology: Manually make test accounts, and populate them with task data using the UI, or by directly populating the backend.
        - Tools: Can use Firebase Realtime Database tools for importing data to the backend [8].
    b. Automated Data Creation
        - Objective: Quickly create data in large quantities, for testing scenarios where many accounts are needed.
        - Methodology: Write scripts to generate a large volume of data, then import this data to the backend.
        - Tools: Python will be used to write these scripts, and Firebase will be used as the backend.
4. Metrics
    a. Test Coverage
        - Objective: Accurate representation of the level to which requirements are covered by test cases.
        - Methodology: Track whether or not each requirement specified in the SRS has been addressed by a test case.
        - Tools: Use Jira to track this information [9].
    b. Defect Density
        - Objective: Assess the frequency of defects, with respect to the size of each module.
        - Methodology: Determine the number of defects present per one thousand lines of code (KLOC) to calculate defect density.
        - Tools: Use Jira to help evaluate this value.

Item Pass/Fail Criteria

- A test has been passed if its outcome satisfies expectations without defect.
- A test has failed if its outcome differs at all from the expected outcome.

Suspension Criteria and Resumption Requirements

Testing shall be suspended if the system crashes during testing, or if the tester deems that the system is experiencing critical failures of functionalities. In the case of a system crash, testing may resume once the system has successfully rebooted. In the case of critical failure of

functions, testing may resume once the defects causing failure have been addressed, and the program has undergone regression testing.

Test Deliverables

The artifacts produced during testing will be:

1. Test Plan: This document. Establishes the testing procedures for the system.
2. Test Scenarios: Outlines a set of circumstances under which the program should be tested.
3. Test Cases: Provides a detailed set of instructions for executing a test of a functionality of the program. Includes preconditions, expected inputs and outputs, and postconditions.
4. Test Data: A record of all of the inputs required to execute a given test.
5. Test Logs: A record of executed tests.
6. Defect Reports: Documents flaws in the program found during testing.
7. Summary Report: Provides an overview of testing progress.

Test Tasks

The tasks described here must be completed to generate the test deliverables.

Environmental Needs

The completion of this testing plan is reliant on the availability of either actual, or emulated iOS, Android, and Windows platforms. Furthermore, access to a reliable internet connection will be required to complete testing related to cloud-based data storage via Firebase.

Responsibilities

The development team of the Time Estimation App shall be responsible for the completion of this testing plan.

Staffing and Training

The development team of this project will need to be trained to use testing tools associated with the libraries and frameworks used in this project. The development team will be trained to use the testing tools available from React Native, and Firebase [9, 10].

Schedule

Testing will begin at the beginning of work on Milestone 2 (October 1st). Unit testing should be completed in 1 week. Integration testing will be completed in 1 week (following the completion of unit testing). System testing will be completed in 2 weeks (following the completion of unit and integration testing). Regression testing will occur in parallel to each other type of testing. A battery of regression testing will be executed after the implementation of every new feature, to ensure functionality is maintained.

Risks and Contingencies

Given the inexperience of the development team concerning the development and testing of an application, there is a risk that the testing outlined here will not be completed by October 28th (Milestone 2 evaluation).

In an effort to mitigate the risk of delay, the most time-consuming form of testing (system testing) will be allocated twice the completion window of the other 2 phases.

Approvals

The client and advisor for this project, Dr. David Luginbuhl, shall be responsible for approving the testing phases described in this document. Dr. Luginbuhl's approval shall be required for a testing phase to be deemed complete and the subsequent phase to begin.