**System Design Document**

**For**

**COVID-19 Tracking App**

Team Members:

● Sultan Abdalla Salim Saif Alteneiji

● Justin Andrews

● Bryce Cole

● Jonah Corbin

|  |  |  |
| --- | --- | --- |
| Version | Date | Authors |
| V1 | 11/10/2020 | Justin |
| Final | 11/30/2020 | Justin |

# Table of Contents

[**Table of Contents**](#_w23p6q9qily4) **2**

[**1 Introduction**](#_oefunvmdjzst) **3**

[1.1 Purpose](#_mujlbix2au55) 3

[1.2 Objectives](#_nlrhhj4a784o) 3

[**2 Functional Scope**](#_ie4zhy7jnqpl) **3**

[**3 Overall Strategy and Approach**](#_5cocbmt6yrs6) **3**

[3.1 Testing Strategy](#_hkxsjccxk1g6) 3

[3.2 System Testing Entrance Criteria](#_97pd05mmj1xg) 3

[3.3 Testing Types](#_pdpfzb605mwf) 4

[3.3.1 Usability Testing](#_ndbjrwgitcsh) 4

[3.3.2 Functional Testing](#_cad4cs7mx1tb) 4

[3.4 Suspension Criteria and Resumption Requirements](#_wmiit9fvkykc) 4

[3.4.1 Suspension Criteria](#_d5byzbu7yydy) 4

[3.4.2 Resumption Requirements](#_d5byzbu7yydy) 4

[**4 Execution Plan**](#_d4ybtpohz153) **4**

[4.1 Execution Plan](#_u0629pef2onn) 4

[**5 Traceability Matrix & Defect Tracking**](#_45i4x59y2gyy) **7**

[5.1 Traceability Matrix](#_pmzx93gvle9q) 7

[5.2 Defect Severity Definitions](#_pmzx93gvle9q) 7

[**6 Environment**](#_vizmcpruj3xc) **8**

[6.1 Environment](#_es130irsnkzr) 8

[**7 Assumptions**](#_g8dsogg3ibjs) **8**

[**8 Risks and Contingencies**](#_g8dsogg3ibjs) **8**

[**9 Appendices**](#_g8dsogg3ibjs) **9**

# 

# 

# 1 Introduction

## 1.1 Purpose

The purpose of this document is to be a test plan for the COVID-19 Tracking project. This document will clearly describe the approach and methods the team will use to test the application and ensure that it meets the requirements set by the team and product owner.

## 1.2 Objectives

The objectives of this document are:

* To guarantee that the application created meets the requirements that were set by the team and product owner.
* To guarantee that the application works without defects.
* To guarantee that all functionalities of the application are working as intended.

# 2 Functional Scope

This document will outline testing off all functionality and all files that are in the covid\_app/lib folder of the application. At a high level this will include the ability for the user to upload their data to the app, have that data be parsed inside the app, alter the weights of the ego network algorithm if the user chooses so, have the ego network algorithm analyze that data, and output an interactive result to the user. We will also be testing the intuitiveness and usability of the app UI. Examples of this include how easy it is to understand what each tab of the app does, can a new user of the app easily understand how to upload their data and then analyze it, can the user see the results from their most previous ego network calculation.

# 3 Overall Strategy and Approach

## 3.1 Testing Strategy

The testing of the COVID-19 Tracking project will include testing what is listed in the Functional Scope section above. The testing of this project will also include testing of functionalities that are added or modified, data validation, and workflows.

## 3.2 System Testing Entrance Criteria

To begin testing there must be criteria to determine if testing is ready to begin, these criteria are:

* Functionality Completion: The functionality/functionalities being tested must be either completed or operational. To find if a functionality is completed or operational see the SCRUM board in this project’s GitHub repository. If the issue related to the functionality has been moved to the “Done” section, then the functionality can be assumed to be completed or operational.
* Functionality Integration: The functionality/functionalities being tested must be fully integrated into the app. To find if a functionality has been integrated see the SCRUM board in this project’s GitHub repository. If the issue related to the functionality has been moved to the “Done” section, then the functionality can be assumed to be integrated.

## 3.3 Testing Types

### 3.3.1 Usability Testing

This testing includes the user interface intuitiveness, usability, presentation, and completion. Usability testing ensures that the user interface is easy for a user to operate and allows them access to all functionality and content within the application.

### 3.3.2 Functional Testing

This testing will ensure that all the major and minor functionalities set by the team and product owner in other documents relating to this project are operational and working correctly.

## 3.4 Suspension Criteria and Resumption Requirements

This section contains the criteria to suspend and resume testing of the application.

### 3.4.1 Suspension Criteria

Testing of the application must be suspended if issues in the application prevent testing of other portions of the application. Once testing is halted the issues in the application must be fixed so that testing may resume. Based on what the issue was portions of the application may need to be re-tested.

### 3.4.2 Resumption Requirements

Testing may be resumed once the issue regarding the functionality that was being tested has been fixed and has been re-tested to ensure the issue and application were fixed. To determine if the functionality has been fixed check this project’s SCRUM Board in GitHub. If the issue that was related to the bug has been moved to the “Done” section, then testing can resume.

# 4 Execution Plan

## 4.1 Execution Plan

The test cases to be executed for this project are:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case Number | Test Scenario | Test Steps | Expected Result | Actual Result | Pass/Fail |
| T-1 | Upload Data | 1. Go to Upload Data tab 2. Click Upload Instagram or Upload Snapchat button 3. Navigate to the folder that contains the data 4. Click the USE THIS FOLDER button at the bottom 5. Click allow to allow the app to access your files | The user will be redirected to the Upload Data tab and below the Data Sources box it will say “Data Source: \*path to folder\*”. | The user gets redirected back to the Upload Data tab and the path to the folder is displayed below the Data Sources box. | Pass |
| T-2 | Delete Data | 1. Go to the Upload Data tab 2. Inside the Data Sources box click the trash can icon next to Instagram or Snapchat depending which you want to delete | Below the Data Sources box where it says Data Source, the path to the file should disappear. | The file path does not disappear once the button is clicked. | Fail |
| T-3 | Calculate Ego Network | 1. Data has been linked successfully 2. Go to Ego Network tab 3. Click Calculate | The user will be shown a categorized, interactive list of their friends based on the data that was analyzed. | The user is presented with their friends sorted into groups that they can click on to get more detailed information about along with a graph of their ego network. | Pass |
| T-4 | Re-Calculate Ego Network | 1. Data has been linked successfully 2. An ego network has previously been generated 3. There has been a change that would result in a different ego network result (either newer data has been linked or the algorithm weights have been altered) 4. Go to the Ego Network tab 5. Click Re-Calculate | The ego network results should be displayed and should be different than the previous ego network. | Different ego network results are displayed. | Pass |
| T-5 | View Friendship Levels | 1. Calculate an Ego Network 2. Click on one of the levels of the ego network (Serious, Good, Friends, Distant) | The user should see a list of all friends in that level with the name of the friend being on the left and the closeness score being on the right. | The app displays a list of the friend’s username on the left and their closeness score is on the right. | Pass |
| T-6 | Adjust Algorithm Weights | 1. Go to Weights tab 2. Click on the weight you wish to adjust 3. Adjust the slider to the desired value 4. Go to the Ego Network tab 5. Calculate or Re-Calculate an ego network | The user can successfully input a value between 0.00 and 1.00 and then calculate an ego network using those altered weights. | The user can alter the weights then calculate an ego network with different results than the previous calculation. | Pass |
| T-7 | App Intuitiveness | 1. A user is given the app with no instructions 2. Using the Help tab inside the app they can get instructions on how to download and upload their data as well as how to use all functionalities within the app | The user can navigate to the Help tab and use the help pages to learn how to successfully use all functionalities of the app. | The help pages thoroughly describe how to download and upload the data as well as use the app and all functionalities within it. | Pass |
| T-8 | Reset Algorithm Weights | 1. Go to the Weights tab 2. Alter multiple weights 3. Calculate an ego network 4. Observe the results 5. Go to the Weights tab 6. Select Reset Defaults 7. Go to the Ego Network tab 8. Select Re-Calculate | The weights should have returned to their default values and the recalculated ego network should have different results from the ego network with altered weights. | The ego network results have different closeness scores for all friends in the network. | Pass |

# 5 Traceability Matrix & Defect Tracking

## 5.1 Traceability Matrix

|  |  |
| --- | --- |
| Test Case | Corresponding Requirements |
| T-1 | F-7, D-1,, S-1, S-3, QA-1 |
| T-2 | F-6, S-1, S-3, QA-1 |
| T-3 | F-1, F-2, F-3, F-4, F-6, F-8, F-10, D-2, S-2, S-3, QA-1 |
| T-4 | F-1, F-2, F-3, F-4, F-6, F-8, F-10, D-2, S-2, S-3, QA-1 |
| T-5 | F-6, F-10, S-2, S-3, QA-1 |
| T-6 | F-2, F-3, F-4, F-9 |
| T-7 | UH-1, UH-2 |
| T-8 | F-2, F-3, F-4, F-9 |

## 5.2 Defect Severity Definitions

|  |  |
| --- | --- |
| Critical | The defect causes a loss of major, core functionality of the application. The functionality cannot be used or does not operate correctly. A defect of this magnitude will take significant effort to fix.  Examples of critical defects are:   * Data cannot be linked * Ego network results are not displayed * Ego network results are incorrect |
| Medium | This defect causes disruption to the user, but the functionality of the application is still intact. A defect of this magnitude will take medium effort to fix.  Examples of medium defects are:   * Data cannot be linked on the first attempt * Algorithm weights cannot be adjusted |
| Low | This defect is typically cosmetic and causes little disruption to the user. A defect of this magnitude takes minimal effort to fix.  Examples of low defects are:   * Text fonts are incorrect * Fields do not align correctly |

# 6 Environment

## 6.1 Environment

To test this application the tester will need to have an environment capable of simulating this app on an emulator. They will need to have the newest versions of Flutter and Dart downloaded and configured as well as an IDE or code editor that supports mobile development (such as XCode, Android Studio, or VS Code). They will then need to have a mobile device emulator installed. If these conditions are met, they should be able to compile our code to the emulator. Once compiled to the emulator they can test our app by interacting with it using their mouse.

# 7 Assumptions

* The user has Instagram and Snapchat accounts.
* They have used both accounts to communicate with multiple people using multiple different communication mediums.
* The user has downloaded their data from both accounts.
* The user has unzipped the folders containing their data.
* The user has uploaded the unzipped folders to the device the app is installed on.

# 8 Risks and Contingencies

* Future versions of Instagram and Snapchat could cause issues when analyzing the linked data as the format of each company's downloaded data may change and cause errors when interfacing with our code.
* If data is downloaded in other languages it would also cause issues if key value names were changed to a different language in the JSON files. Our code was made for the English version using key value names in English.

# 9 Appendices

Not applicable currently.