Comprehensive Monkeypox Dashboard

Functional Testing Document

version 1.0

Prepared by
Saketh Dendi, Felix Huang, and Sakin Kirti (Dendi Huang Kirti Group)
CSDS 393
27 October 2022

Table of Contents

Introduction

1.1 Purpose

This document serves as an outline of the comprehensive testing that we plan to include in order to make sure that our project and its various programs work as intended. To do so, we have split our test descriptions into two categories - test specifications and a table of functional tests to go along with the specifications. The test specifications outline any background information on the features being tested and what each test will focus on, which includes database updating, database retrieval, data cleaning, data predictions, etc. The functional tests outline the implementation of the test specifications and specify how each test is executed. In addition to specifying the various tests that will be accomplished, we will also describe the method's intended purpose before describing what the various inputs and correct outputs will be. In order to do this, we will be calculated in our approach, ensuring that we are testing thoroughly and efficiently.

In order to accomplish our tests, we plan to use the 'pytest' framework for testing any python methods for the backend and the 'Jest' framework for testing JavaScript and JSX methods for the frontend.

1.2 Document Conventions

The formatting of this document is done the following way to stay consistent and allow any viewer to quickly orient themselves.

- Heading: Google Docs' *Heading 1* Format (20pt Calibri, black)
- Section Headers: Google Docs *Heading 2* Format (16pt Calibri, black)
- Subsection Headers: Google Docs' *Heading 3* Format (14pt Calibri, black)
- Body: 12pt Calibri, black

Test Design Specifications

In this document, we have outlined specific tests for each of our different components and classes. Each test will be simulated with various input values that could be given to the specific method. Since our system is automated, we do not need to worry about different user inputs and can therefore specify our tests for the potential input formats we may have. In the various languages that we are using, we will use assert statements, or the equivalent in the different programming frameworks. Tests will be written and executed for each specific component or upon their completion and will also be executed upon the completion of other classes or

components within the same testing environment. Tests that pass will be considered done and those that do not pass will be considered as failed.

Test Specifications

3.1 Frontend Test Specifications

3.1.1 View Overall Interactive Map

The following steps specify a use case for the rendering of content in the map view of the website that would be tested for in the functional tests table below.

1. The user navigates to the base URL of the website or clicks the “Map” link on the navigation bar.
2. The website renders an interactive, color-coded map as well as tables for statewide confirmed cases data as well as nationwide predictive analytics data for the following week.

Test	Description	Test Setup Steps	Expected Result
OIMT-1	Color-coding of map according to legend	Enter different manual confirmed cases data (0, 5, 15, 75, 250, 1000) for a given state to be fetched from the MapView	The map should render a different color for the state that corresponds to the legend of confirmed cases
OIMT-2	Sort by descending order for table of statewide confirmed cases data	Enter different manual confirmed cases data (0, 5, 15, 75, 250, 1000) for a given state to be fetched from the MapView	The map view should render a table with the state name and cases number in the correct position in the table

3.1.2 Movement Actions on Interactive Map

The following steps specify possible use cases for the user interactions with the overall interactive map that would be tested in the functional tests table below.

1. The user drags left, right, up, or down on the map using a trackpad or mouse.
2. The map moves in the corresponding direction.
3. The user zooms in or out on the map using a trackpad or mouse or using the zoom controls of the map.
4. The map zooms in or out in the corresponding direction.

5. If the zoom level or drag amount exceeds that of the specified view of the U.S., then the map will bounce back to the previous position of the map to prevent the user from viewing other parts of the world.

Test	Description	Test Setup Steps	Expected Result
AIMT-1	Movement around map	Drag left, right, up, and down	Map moves in corresponding direction
AIMT-2	Zoom on map	Zoom drag up and down, and press zoom in (+) button and zoom out (-) button of zoom controls	Map zoom in corresponding direction
AIMT-3	Attempt to drag out of U.S. bounds	Drag towards west toward Asian countries and east toward European countries	Map bounces back to previous position
AIMT-4	Attempt to zoom out of U.S. bounds	Zoom out as much as possible	No action taken for zoom level beyond level 10, an amount specified according to React Leaflet documentation

3.1.3 View Pop-up Chart of Confirmed Cases for a State

The following steps specify possible use cases for the U.S. map specifically that would be tested in the functional tests table below.

1. The user clicks on a U.S. state on the map or in the table of statewide confirmed cases.
2. The website renders a pop-up window of the chart of confirmed cases data for that state since the first confirmed case.

Test	Description	Test Setup Steps	Expected Result
PUCT-1	Map click on specific state	Click on any U.S. state in the interactive U.S. map	The map view shows a pop-up window displaying the chart of confirmed cases since the first confirmed cases
PUCT-2	Table click on specific state	Click on any U.S. state in the table of statewide confirmed cases	The map view shows a pop-up window displaying the chart of confirmed cases since the first confirmed cases
PUCT-3	Attempt to	Click on Canada and	No action taken

	click on other countries in the map view	Mexico, and any countries present in the map view	
PUCT-4	Close pop-up chart	Click on “x” button of pop-up chart	View of pop-up window chart exits

3.1.4 View Chart of Confirmed Cases for a State

The following steps specify a possible use case for viewing the chart of time series data for confirmed cases for a specific state.

1. The user clicks on the “Chart link” on the navigation bar.
2. The website navigates to the chart view of confirmed cases with the default data set as the U.S.
3. The user clicks on a specific state in the table of statewide confirmed cases.
4. The chart view renders a display of time series data since the first confirmed case for that state.

Test	Description	Test Setup Steps	Expected Result
CCCST-1	Table click on specific state	Click on any U.S. state in the table of statewide confirmed cases	Time series chart of confirmed cases data since the first confirmed case is displayed along with prediction data as an extension of the graph

3.1.5 View Day-specific Cumulative and Predictive Data for a State

The following steps specify possible use cases for interactions with the chart view that would be tested in the functional tests table below.

1. The user hovers over the chart view of confirmed cases at a specific point.
2. The website displays a small window that contains information about the date and cumulative confirmed cases up to that date, or, the cumulative number of cases in the future if hovering over the predicted progression of cases.

Test	Description	Test Setup Steps	Expected Result
DSCPDT-1	Chart view hover over current data	Hover over a specific point along the currently available data on the time series chart	Chart view renders a small window displaying information about confirmed cases data for a specified date

DSCPDT-2	Chart view hover over predictive data	Hover over a specific point along the projected progression of confirmed cases	Chart view renders a small window displaying information about predicted confirmed cases data for a specified date

3.1.6 Navigate to Different Sections (Map, Chart, and Help) of Website

The following steps specify possible use cases for the navigation bar of the website that would be tested in the functional tests table below.

1. The user clicks on the “Map” link on the navigation bar.
2. The website navigates to the interactive map view of the U.S.
3. The user clicks on the “Chart” link on the navigation bar.
4. The website navigates to the chart view of confirmed cases with the default data set as the U.S.
5. The user clicks on the “Help” link on the navigation bar.
6. The website navigates to a list of FAQs and answers.

Test	Description	Test Setup Steps	Expected Result
NT-1	Map navigation bar link	Click on “Map” section of navigation bar	Interactive map view, which is the home page, renders
NT-2	Chart navigation bar link	Click on “Chart” section of navigation bar	Interactive chart view renders
NT-3	Help navigation bar link	Click on “Help” section of navigation bar	Help section of FAQs and answers renders

3.2 Backend Database Updating Test Specifications

3.2.1 Database connection functionality

The functionality here is performed by 2 methods but there’s very limited checks to complete before testing

1. Makes sure the database exists to connect to

Test	Description	Test Setup Stats	Expected Result
DB-1	Check if backend can connect to the database	Run the db_connect method using the assert method	A message saying we are connected to the database
DB-2	Check if the backend can disconnect to the database	Run the db_disconnect using the assert method	A message saying we have been disconnected to the database

3.2.2 Database Updating functionality

The functionality is propped up by a couple of features but we need to have a few checks before moving to the next steps

1. Cleaning method has to work properly
2. The online excel sheet has to be up and updated from the last time we accessed it
3. Make sure we are connected to the Database

Test	Description	Test Setup Stats	Expected Result
UP-1	The database has been changed from last time	Use assert statement to see if the data in the pd.Dataframe.Columns has changed from the last time	The columns are changed to match the new database
UP-2	Sending data to the cleaner	Use assert statement to see if the data goes to the cleaner after it is pulled	Print a verification that the cleaner has received the data

3.2.3 Database Synching functionality

The functionality for the database synching is extremely straightforward and can be tested with almost zero preconditions

1. The time is the same as the time it should be in order to run the Syncher

Test	Description	Test Setup Stats	Expected Result
SY-1	The time that the database needs to be synched with the online function has arrived	Use assert statement to see if there is a statement generated at that specific time	Will print out a statement and a time that statement was printed

3.3 Data Cleaning and Prediction Test Specifications

3.3.1 Data Cleaning Functionality

This entire functionality is performed by a single method. Therefore, there are a few checks to complete before moving on to the next steps:

1. The attributes present in the dataframe are only those that are needed
2. The attributes contain data with the correct data types
3. There are no missing values in the data
4. The data is filtered to only those cases in the U.S.
5. All cases have a state (location)
6. Any duplicated data is removed

Test	Description	Test Setup Steps	Expected Result
DCF-1	Only necessary attributes are present	Use assert statement to check if the pd.DataFrame.columns attributes returns the right attribute names	The returned columns are exactly what is expected
DCF-2	Attributes are of the correct data type	Use assert statements to check that each attribute is of the correct type	All tests pass where each attribute is of the expected data type
DCF-3	No missing values in the data	Use assert statement to check if pd.DataFrame contains missing values	Test where pd.DataFrame.isna() is checked should be False
DCF-4	Data should contain only U.S. cases	Use an assert statement to check if all values for the country attribute U.S.	Test for unique values in the country attribute - where the only value should be U.S.

DCF-5	All cases should have a state	Use assert statements to check that each location is a valid state name	Test if each state name is in the list of valid states
DCF-6	No duplicate data is present	Use assert statement to check if there is any duplicated data	Use pd.DataFrame.duplicated() to check for duplicated rows - expected False

3.3.2 Data Summarization

All of the visualizations that we produce are only reliant on summary data. Specifically, the number of new cases identified on each day (in each state).

1. Attributes present in the data are date, number of cases, state
2. No missing data

Test	Description	Test Setup Steps	Expected Results
DS-1	Summarized data should contain only specific attributes	Use assert statement to check that the columns present are those expected	Use pd.DataFrame.columns to check that the attributes are those expected
DS-2	No missing data	Use assert statement to check for no NAs or missing values	Use pd.DataFrame.isna() where the expected result is False

3.3.3 Public Health Statistics Calculation

1. Values calculated here are ratios with population size as the denominator, so they should always be <1
2. Incidence rate should be lower than prevalence rate if cases are going down, vice versa if case numbers are increasing

Test	Description	Test Setup Steps	Expected Results
PHSC-1	Ensure that the calculated value is reasonable	Use assert statement to ensure that the calculated value is within a certain range	0 < incidence rate < 1 0 < prevalence rate < 1 1 < case-fatality ratio
PHSC-2	Incidence rate and prevalence	Use assert statement to ensure that the values	Incidence rate < prevalence rate if case counts are decreasing,

	rate relationship	are in the right range	Incidence rate > prevalence rate if case counts are increasing

3.3.4 Data Predictions

This data output will be in the same format as that of data summarization and public health statistics calculations section. Refer to those sections (3.3.2 and 3.3.3) for relevant test cases and expected outcomes.

3.3.5 Relational Data Table Setup

Finally, data should be distributed across 2 tables in the proper format so that updating our database is easy

1. Case counts data which includes both given and predicated case counts (with an additional attribute to differentiate predicted from given cases)
2. Public health statistics data which includes both given and predicted statistics (with an additional attribute to differentiate predicted from given cases)

Test	Description	Test Setup Steps	Expected Results
RDTS-1	Case counts includes given and predicted data	Use assert statement to make sure 14 days of predicted data is in the final table to insert into database	True - that there are 14 days worth of predicted data
RDTS-2	Public health statistics includes given and predicted data	Use assert statement to make sure predicted and given data is in the final table	True - that both predicted and given public health statistics have been calculated

Defect Responsibility and Resolution

4.1 Issue Tracking

We plan to use Github's Issue Tracking software to track any relevant issues that come up. This tool is powerful enough for explaining issues and how to solve them and is easily accessible because all of our members have access to it.

4.2 Individual Responsibilities

Because of our team's well-distributed workload, any defects/issues that are found in an individual's portion of code will be fixed by that person. These are outlined as follows:

System	Developer	Responsibilities
Frontend and relevant testing	Felix Huang	Creating an aesthetically pleasing display with easy-to-use functions. Additionally, creating the tests to go along with this.
Backend database updating and relevant testing	Saketh Dendi	Updating the backend database as well as writing tests to go along with this.
Backend data cleaning and prediction generation and relevant testing	Sakin Kirti	Data cleaning, prediction generation, and database setup as well as the tests to go with this.

Appendix

5.1 Inspection Report

This section of the appendix contains a progress log detailing the parts that each individual worked on along with a short description of the overall breakdown of how much of the document that each team member had a hand in producing.

5.1.1 Progress Log

FH = Felix Huang, SD = Saketh Dendi, SK = Sakin Kirti

Date	**Person**	**Item**
10/25/2022	SK	Outlined document format and progress log
10/25/2022	SK	Wrote introduction including purpose and document conventions
10/26/2022	FH	Wrote test specifications and test cases for frontend components
10/26/2022	FH, SD, SK	Wrote Defect Responsibility and Resolution section
10/26/2022	SK	Wrote tests for data cleaning and prediction test specifications
10/27/2022	SD	Wrote tests and test specifications for data updating and

		synching
10/27/2022	FH, SD, SK	Checked report for content, formatting, and grammar correctness
10/27/2022	SK	Wrote section 5.1.2 - work breakdown as a summary of work distribution
10/27/2022	FH, SD, SK	Checked inspection report for correctness

5.1.2 Work Breakdown

This document was worked on very evenly. Saketh performed 30% of the work, writing all test cases for the database updater and database syncher. Sakin performed 35% of the work, including the inspection report work breakdown, introduction, and data cleaner sections. Felix worked on 35% of the document, writing functional test cases for the front-end environment. Additionally, everyone took part in grammar editing and content editing. Overall, the project was split evenly between group members.