Mission: The UMBC Covid-19 Tracker - Provide Covid-19 related case and vaccine information for public safety and knowledge

DID: Software Test Plan (STP)

PREPARED by: Quinque Equos

(

[Alex Wilson](mailto:alexanw1@umbc.edu)

Safia Shah

[Israel Morocho](mailto:cx59298@umbc.edu)

[Duncan Taylor](mailto:ak58697@umbc.edu)

[Tom Tennant](mailto:tent1@umbc.edu)

[David Kravets](mailto:kravets1@umbc.edu)

)

DATE: APRIL-2021

REVISION LOG

| Revision | By | Description | Date |
| --- | --- | --- | --- |
| A | Team | Initial Version | 04/06/2021 |
| B | Team | Ready to be turned in | 4/20/2021 |

Table of Contents

[**Scope**](#_twyevmyex27v) **5**

[1.1 Identification](#_hihizpuydlz4) 5

[1.2 System Overview](#_kfrq87of8wog) 5

[1.3 Document Overview](#_igt24z9vbx0h) 6

[1.4 Relationship to other Plans](#_wg2tni6a55w2) 6

[**Referenced Documents**](#_7o8g4tkoo3je) **6**

[**Software Test Environment**](#_94jhp2709z6d) **7**

[3.1 Desktop test site](#_1hfbuop26384) 7

[3.1.1 Software items](#_xhc0v8oppxq0) 7

[3.1.2 Hardware and firmware items](#_dgotvp71e7k3) 8

[3.1.3 Other materials](#_u4tu5ptwe6ct) 8

[3.1.4 Proprietary nature, acquirer’s rights, and licensing](#_104bifi1ua9u) 8

[3.1.5 Installation, testing, and control](#_5a0z3e4hnyfk) 8

[3.1.6 Participating organizations](#_ffxuj3msd5zw) 8

[3.1.7 Personnel](#_946voehnrb4y) 9

[3.1.8 Orientation Plan](#_9qcfvosxu9vd) 9

[3.1.9 Tests to be Performed](#_1fdnttq0rp5t) 9

[**Test Identification**](#_wgsn58hso99m) **10**

[4.1 General Information](#_ms7lsepj7ydc) 10

[4.1.1 Test levels.](#_9gt9pvedtfva) 10

[4.1.2 Test classes.](#_591zt1j7ltke) 10

[4.1.3 General test conditions.](#_u0togktvb1oc) 10

[4.1.4 Test progression.](#_m9ofu63ggdb4) 10

[4.1.5 Data recording, reduction, and analysis.](#_8hehquovsiut) 11

[4.2 Planned Tests](#_lnz5thz4atv8) 11

[4.2.1 Inputs testing.](#_w8e8brnktw0w) 11

[4.2.2 Input conversion testing.](#_3s0vmsbrirrc) 11

[4.2.3 Outputs testing.](#_audumjh0akfc) 11

[**Test Schedules**](#_7fewve7edkc1) **12**

[**Requirements traceability**](#_94jab8dibni4) **12**

STP References:

Figure 1 5

Figure 2 6

Table 1 9

Table 2 12

Table 3 12

# Scope

## 1.1 Identification

This Software Test Plan (STP) applies to the testing of the web application called the UMBC Covid-19 Data tracker (CDT). The CDT is in its initial version, version 1.0 to which this STP applies to. The intended audience includes the engineers tasked with developing the application and the test conductors responsible for the verification and acceptance of the application.

## 1.2 System Overview

The Covid-19 Data Tracker web applications main purpose is to be informative about the covid-19 virus and its currently available vaccines. The CDT web application will be designed to display a heat map of a selected state and, if applicable, the counties within and their covid data. This web application does not expand outside of U.S data. The data will be queried from the Broadstreet covid data sets in order to generate an organized display for the user selected areas.

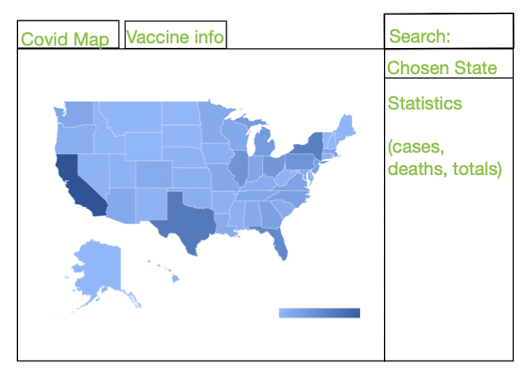


Figure 1 from SDD: Initial visualization of the CDT web application



Figure 2 from SDD: displays the flow of execution of the CDT

## 1.3 Document Overview

Section 1 describes the scope of the project. Section 2 provides the referenced documents in the STP. The software test environment and the test identifications are presented in sections 3 and 4, respectively. Test schedules are included in Sections 5. Finally, requirements traceability can be found in Section 6.

## 1.4 Relationship to other Plans

This STP has been developed after the initial versions of the CDT Software Development Plan (SDP), the Software Requirements Specification (SRS), and the Software Design Description (SDD). Requirements and feedback presented in those documents will be reflected in the STP. The results of the following test plans will be presented in the CDT Software Test Report (STR), which does not exist at this time.

# Referenced Documents

* 1. STP template: <https://alt-5954eb74c7df4.blackboard.com/bbcswebdav/pid-4725369-dt-content-rid-47159286_1/courses/CMSC447_11379_SP2021/STP.htm?one_hash=4EF8F81534A240151D24877E3D39B062&f_hash=71D1895263DF75FDBE206B0D1558D496>
  2. The Broadstreet Covid-19 Data project

<https://covid19dataproject.org/>

# Software Test Environment

Our testing environment will only consist of a virtual environment which will be used for input checking, display testing, integration testing, automation testing, and functional testing.

## 3.1 Desktop test site

All of our testing will be done on a desktop machine. Due to this we have no physically specified testing area as such we can perform tests in any location our members deem appropriate. The testing environment will be in a virtual setting and is accessible to all members allowing everything to be transferable onto different desktop machines.

### 3.1.1 Software items

Github will be used as our operating system as it allows us to have multiple people work on different sections of the project concurrently. Github also provides a branching feature which helps in viewing and updating different versions of our project. This will allow us to easily make and view changes made by other members.

React will be used by our front end group to build the user interfaces and UI components for our project. This will serve as our API for users to make queries to our database. Node.js will be used by our backend group and is where our database will be built and tested. NPM will be used as our package manager which is included with Node.js.

### 3.1.2 Hardware and firmware items

The application will have no other interaction outside of the virtual environment. As such no additional 3rd party firmware will be used for testing. The only hardware requirements are for users to have 8GB of RAM for a smooth experience.

### 3.1.3 Other materials

A copy of the database for covid case along with a map could be used as reference when testing the program. This would be used to assure the accuracy of our data when performing functional and display testing. All members will have access to a copy of the database and maps can be provided from online sources.

### 3.1.4 Proprietary nature, acquirer’s rights, and licensing

Our testing plans are not tied with any proprietors nor require licensing or assumed rights. All our environments are free and open-sourced. Our testing plans will be performed under those environments.

### 3.1.5 Installation, testing, and control

Software developers and test conductors will be required to download Github and npm for creating and testing our product. Github will be the host environment in which we will be testing the modules while npm will be used to help manage the packages which comes with npx which builds the packages.

### 3.1.6 Participating organizations

Broadstreet will be providing us the covid cases data needed as a baseline for our group to run our scripts and have something to compare when performing functionality testing.

### 3.1.7 Personnel

Table 1

| Personnel | Role | Skills |
| --- | --- | --- |
| Database Manager | Manages the Database, will ensure all data is in it’s correct location and will append data when necessary | Knowledge of database manipulation through SQL or other database languages. |
| API Manager | Ensures the quality of the API system and that queries are being read and followed. | Knowledge of API integration |
| Test conductor | Conducts test that the Database and API are able to communicate with one another | Embedded Software development |

### 3.1.8 Orientation Plan

The test conductors must be familiar or familiarize themselves with what the heatmap will be displaying and how it should be displayed. They will also need to have access to the database in order to ensure the correct data is displayed.

### 3.1.9 Tests to be Performed

All tests that we will be conducting will be able to be performed by one person, in one environment. The tests should also be able to be replicated by each person with the same results returned.

# Test Identification

## 4.1 General Information

### 4.1.1 Test levels.

The plan is to test the Covid Data Tracker web application at the CSC and CSCI level. Each aspect of the application broken down into components will be Database, API, and Front-end, each with their own tests. There will be system-level testing involving the application in its completed form. This will involve transactions between each of the three components of the application.

### 4.1.2 Test classes.

Function-based tests are planned for the Covid-Data tracker that cover the following functionalities:

* Database - Tracker’s ability to query the database and generate appropriate results.
* API - Be able to be interacted with from the frontend, and pass that onto the backend database in an efficient manner
* Display - Ensures response data from the API is translated and visualized properly on the heatmap component of the frontend.
* Integration - Tracker’s ability to communicate with the backend API and properly query data based on given input.
* Input Checking - Validate input from user to guarantee no invalid characters are passed onto subsequent sections of the application

### 4.1.3 General test conditions.

All tests could be performed on any operating system by any person as long as they have database access. Tests will be performed on a private instance of the web app but will be performed on the public, app wide, database. Writing to the database will not be done and therefore should not be able to compromise the integrity of the database.

### 4.1.4 Test progression.

There are no planned tests that require chaining CSCIs.

### 4.1.5 Data recording, reduction, and analysis.

Recording for the tests will not be needed if the test is a success. If the test fails, the recording will be screenshots and a step-by-step on how to recreate the issue. The outputs will simply be compared with the values in the database. If the values match, the test was a success. If the values do not match, there is an issue that needs to be resolved and then screenshots will be taken and passed to the appropriate person who would be in charge of fixing said problem. Display of vaccination pin points on the heat map will be tested for accuracy based on its location, screenshotted if needed to adjust.

## 4.2 Planned Tests

### 4.2.1 Inputs testing.

The inputs of the system will need to be tested. We need to ensure that on valid input, valid results are returned. This will be by entering many existing places that we have data for. This should pass the data towards the database to and will attempt to return the information asked for. We will also try entering places that don’t exist and places that we do not have data for. This should result in errors displayed to the users telling them what the problem was and how they can fix this. Inputs testing also includes the many buttons throughout the program and making sure pressing them, changes the web app accordingly.

### 4.2.2 Input conversion testing.

It will not be unlikely that someone will enter an address into the search bar. This could be their home address, their destination, or any other address. Our data is sorted by county. We will need to use that address to figure out what county to display, if a county is not provided. We will be entering many addresses and making sure the program translates it to the correct county. Fake addresses should not get this far in the program’s logic, and therefore should not be considered as something to test for here.

### 4.2.3 Outputs testing.

It is very important that the correct data is displayed. We will be entering good and bad inputs as described above. Upon good entry, the heat map will be populated with the correct data from the database and the statistics module will be populated with the correct data, again from the database. We will, by hand, compare what is displayed to what is in the database to ensure that the correct data is displayed.

# Test Schedules

The test schedule is presented in Table 2 (below):

| Requirement | Site | April 3rd Week | April 4th Week | May 1st Week | May 2nd Week | Responsible |
| --- | --- | --- | --- | --- | --- | --- |
| *Orientation* | Desktop |  | x |  |  | Test Conductor |
| *Conducting Tests* | Desktop |  |  | x | x | Test Conductor |
| *Preparation, review, and approval of the Software Test Report (STR)* | N/A |  |  |  | x | Backend Developers, Frontend Developers, Test Conductor |

# Requirements traceability

Table 3

| Requirement ID | Inputs testing | Inputs conversion testing | Outputs testing |
| --- | --- | --- | --- |
| CT\_F\_1 | x |  |  |
| CT\_F\_2 | x |  |  |
| CT\_F\_3 | x |  |  |
| CT\_F\_4 | x | x |  |
| CT\_F\_5 |  |  | x |
| CT\_F\_6 |  |  | x |
| CT\_F\_7 |  |  | x |
| CT\_F\_8 |  |  | x |
| CT\_F\_9 | x |  |  |
| CT\_F\_10 | x |  |  |
| CT\_F\_11 | x |  |  |
| CT\_F\_12 | x | x |  |
| CT\_F\_13 |  |  | x |
| CT\_F\_14 | x |  |  |
| CT\_F\_15 |  |  | x |