Software Design Specification

for

Customizable Analysis and Visualization Tool for COVID Cases

Version 1.0

Prepared by Calvin Burns, Sam Hartle, Stian Olsen, Nicole Wright

Florida Institute of Technology Senior Design

September 8, 2020

Table of Contents

Table of Contents 2

Revision History 3

1.0 Introduction 4

1.1 Goals and objectives 4

1.2 Statement of scope 4

1.3 Intended Audience 4

2.0 Definitions 4

3.0 System Structure **5**

3.1 Architecture diagram 5

3.2 Description for Website 5

3.3 Description for Server 5

3.4 Description for Frontend 6

3.5 Description for Backend 7

3.6 Description for Admins 7

4.0 Data Design **8**

4.1 HTTP request on the server 8

4.2 Database model 9

5.0 User interface design **10**

5.0.1 Screen images 10

5.0.2 Objects and actions 11

5.1 Interface design rules 12

Revision History

Name	Date	Reason For Changes	Version

1.0 Introduction

This section provides an overview of the entire requirement document. This document describes all data, functional and behavioral requirements for software.

1.1 Goals and objectives

The goal of this project is to create a web application that shows COVID-19 case data, allow users to perform customizable analyses/visualization of results, and allow users to add additional pieces of data. The user has the ability to customize and get the results they are looking for rather than being stuck with certain results.

1.2 Statement of scope

The Customizable Analysis and Visualization Tool for COVID is a web application that shows COVID case data, allows users to perform customizable analyses/visualization of results, and allows users to add additional pieces of data. There are various COVID dashboards out there, but the analyses are pre-determined. We are developing a COVID dashboard where the analyses are not predetermined. In this software, the user will have the ability to customize and get the results they are looking for rather than being stuck with definite results.

1.3 Intended Audience

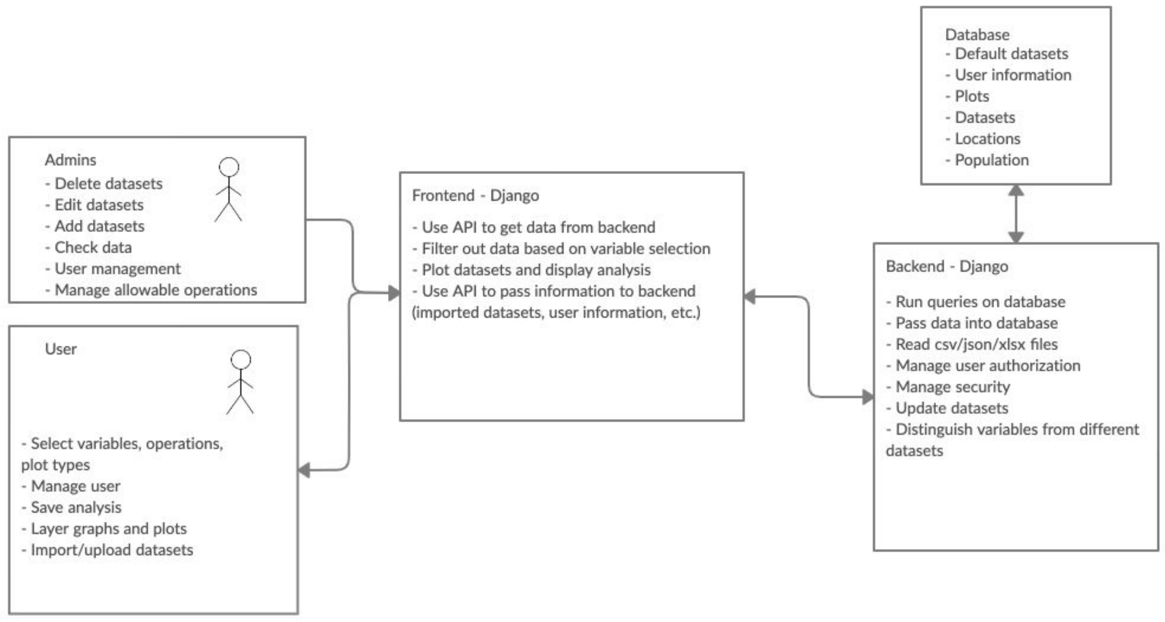
The intended audience for this project is anyone with an interest in COVID. The purpose of the project is to allow users to perform customizable analyses/visualization of results. Therefore, the goal is to attract people with an interest in performing analyses other than the predetermined ones that are available today.

2.0 Definitions

DEFINITION	EXPLANATION
HTTP	Hypertext Transfer Protocol. Client opening a connection to make a request.
API	Application Programming Interface. Defines calls/requests a client can make, how to format data before requesting, conventions to follow, etc.
URI	Uniform Resource Identifier. Uniquely identifies a resource.
Metadata	Set of data describing and giving information about other datasets.

3.0 System Structure

3.1 Architecture diagram


Database
- Default datasets
- User information
- Plots
- Datasets
- Locations
- Population
Admins
- Delete datasets
- Edit datasets
- Add datasets
- Check data
- User management
- Manage allowable operations
Frontend - Django
- Use API to get data from backend
- Filter out data based on variable selection
- Plot datasets and display analysis
- Use API to pass information to backend (imported datasets, user information, etc.)
Backend - Django
- Run queries on database
- Pass data into database
- Read csv/json/xlsx files
- Manage user authorization
- Manage security
- Update datasets
- Distinguish variables from different datasets
User
- Select variables, operations, plot types
- Manage user
- Save analysis
- Layer graphs and plots
- Import/upload datasets

3.2 Description for User

Users will be able to view the application's user interface on a website on their hardware devices. The users will be able to access all the functionalities on the website. Once the application is opened, users will be able to perform their preferred analysis. The code deployed to the web server will be accessible to the user.

3.2.1 Methods

Some of the methods the user will interact with are the 'store_variables' and 'create_user' methods

	METHOD	DESCRIPTION
1	store_variables	Store all variable selections in an array and store it in an object containing variable selections, operations, plot types, datasource, etc.
2	create_user	Takes user information inputted by the user as input and generates a unique user id and profile.

3.3 Description for Admins

The admins will be able to work directly on a separate admin user interface. The admins will be able to delete, edit, and add datasets.

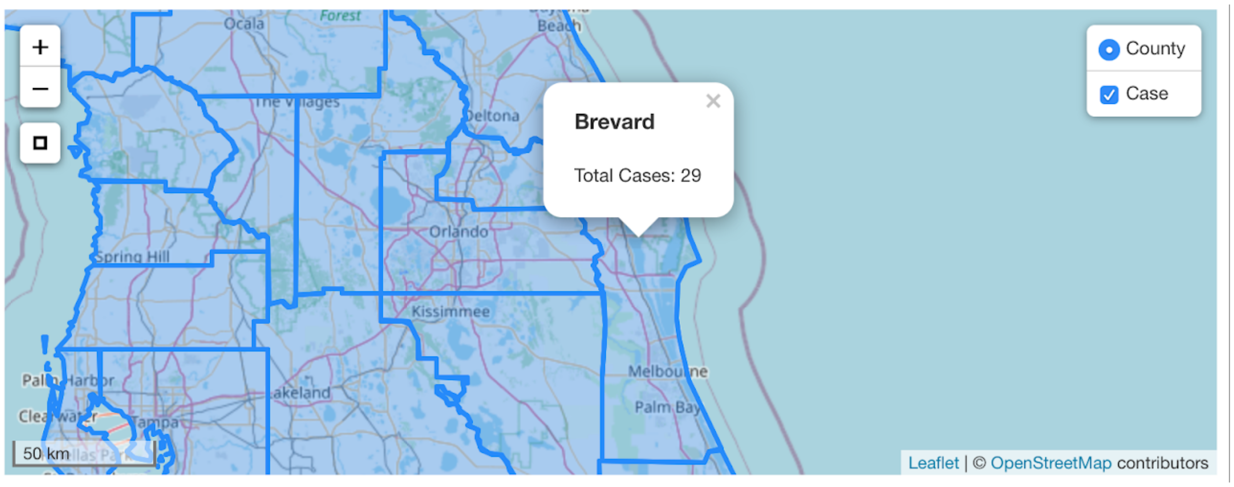
3.3.1 Methods

	METHOD	DESCRIPTION
1	add_datasource	Reads a url, converts the data into json format and distinguishes the data variables for correct display.
2	edit_datasource	Reads a dataset id and prompts the admin with an input for the admin to specify which data will be edited.
3	delete_datasource	Reads dataset id and deletes the dataset from the database

3.4 Description for Frontend

The frontend component will handle the user interface and take care of any input operations done by the user. The frontend component will use the API to get data from the backend. The data from the backend will be returned in a format the frontend can handle. Based on operations done by the user, the data can be filtered, stored, displayed and plotted. The technology used for frontend development is Django which is a web-framework in Python.

The picture below demonstrates a possible user interface. The user would be able to select 'County' and 'Case' on the top right, and click on an area to get COVID case data.



Brevard

Total Cases: 29

3.4.1 Methods

Methods we will be using are map_display and get_data (see description in the table below).

	METHOD	DESCRIPTION
1	map_display	Render a request taken as input and make map data accessible for the html for display on the user interface
2	get_data	Takes a dataset and input from the user as parameters. Filter out irrelevant data from the dataset the user is operating on and display a relevant analysis.

3.5 Description for Backend

The backend component adds utility to the frontend. The backend will create, manage and integrate a database. The backend will also run queries on the database based on data requested by the frontend. The backend will also handle web-server technologies, such as Heroku, which will be used in this project. This component also handles security for the system, maintenance, deployment, API integration, and abstracting data from input files.

3.5.1 Methods

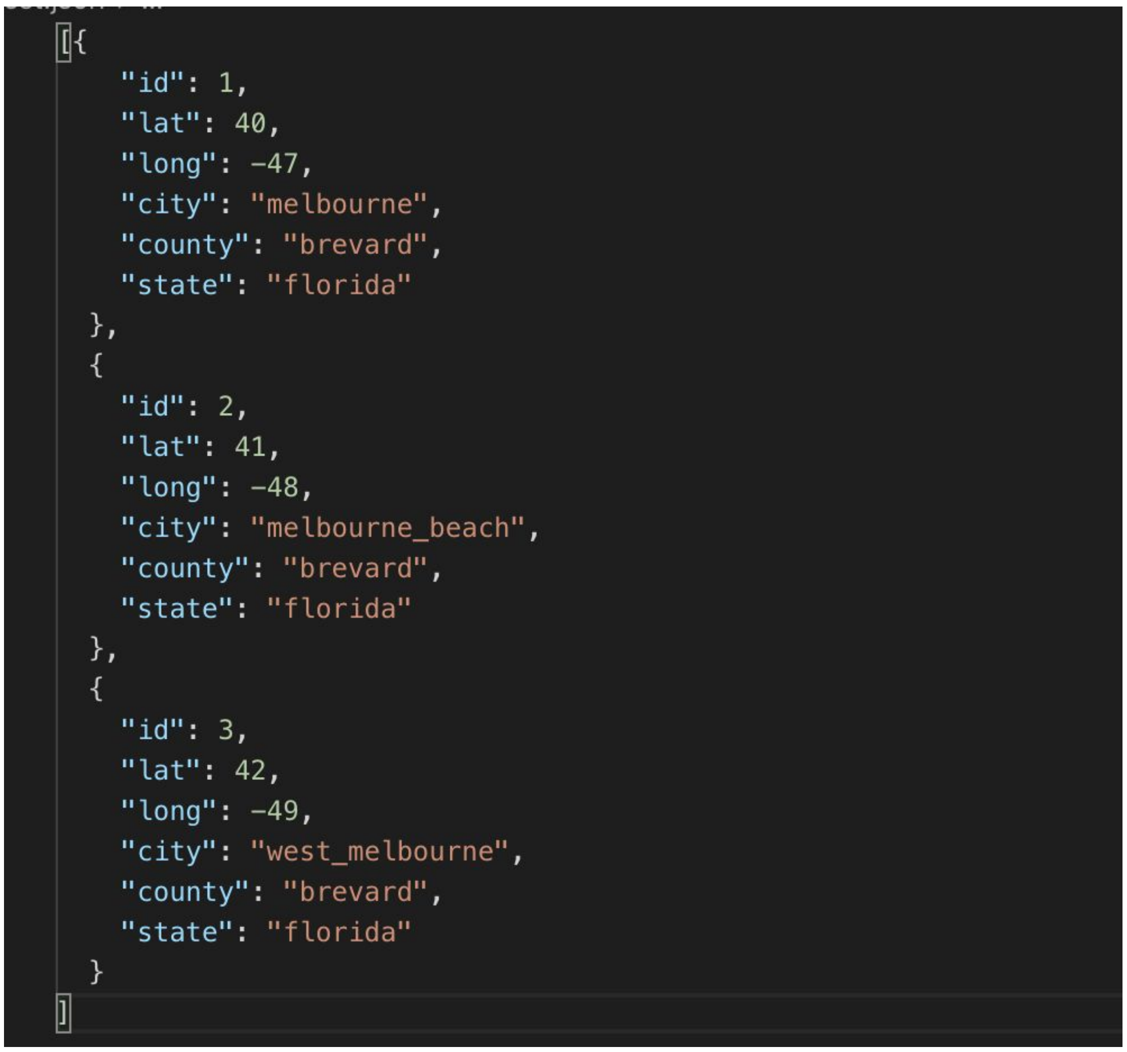
	METHOD	DESCRIPTION
1	handle_datasource	Reads an input file, converts the data into json format and distinguishes the data variables for correct display.

4.0 Data Design

4.1 HTTP request on the server

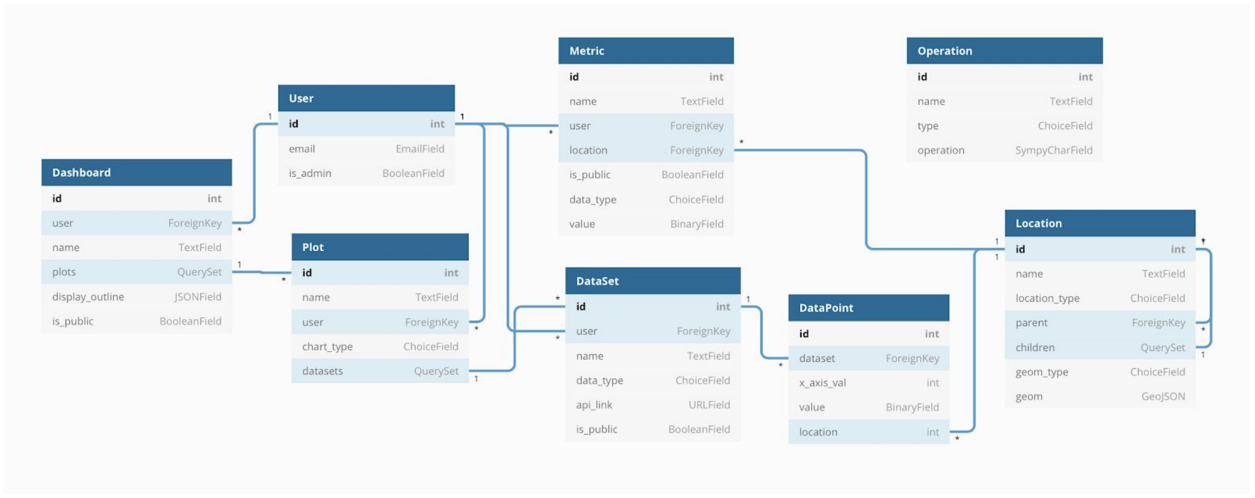
Example command: GET <url>/locations/?state=florida&county=brevard

Example output without HTTP response code:



```
[{
    "id": 1,
    "lat": 40,
    "long": -47,
    "city": "melbourne",
    "county": "brevard",
    "state": "florida"
  },
  {
    "id": 2,
    "lat": 41,
    "long": -48,
    "city": "melbourne_beach",
    "county": "brevard",
    "state": "florida"
  },
  {
    "id": 3,
    "lat": 42,
    "long": -49,
    "city": "west_melbourne",
    "county": "brevard",
    "state": "florida"
  }
]
```

4.2 Database model



4.2.1 Method Descriptions

4.2.1.1 User Methods

- **get_plots:** return Plot.objects.filter(user=self)
- **get_datasets:** return DataSet.objects.filter(user=self)
- **get_dashboards:** return Dashboard.objects.filter(user=self)

4.2.1.2 DataSet Methods

- **render_as_JSON:** return “Dataset in JSON format”
- **update_data:** “Using API link, download and update the dataset.”

4.2.1.3 DataPoint Methods

- **get_value:** return self.value.decode(self.dataset.datatype)

4.2.1.4 Plot Methods

- **render_plot:** return “Datasets’ datapoints exported in self.type JSON format for Chart.JS”

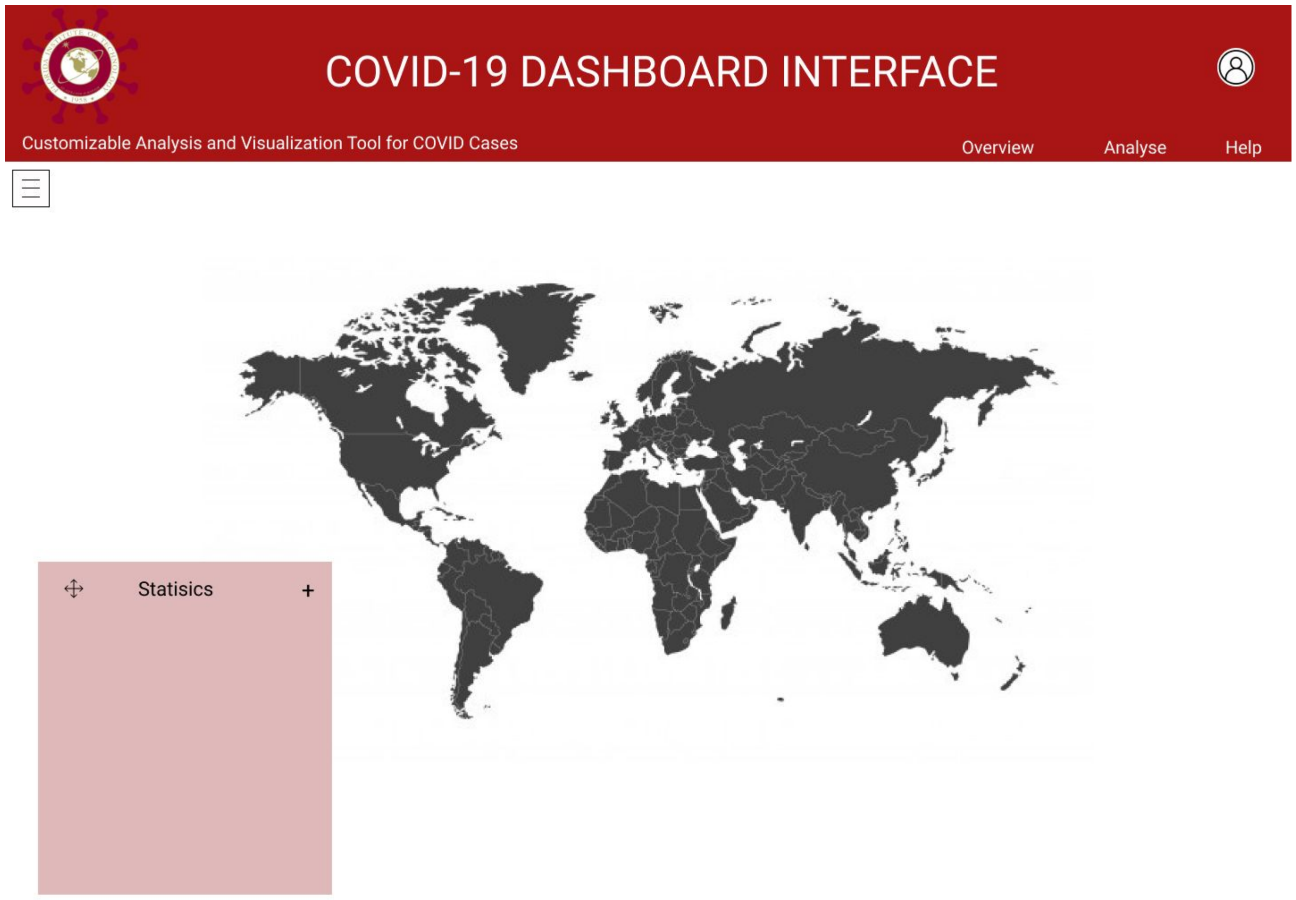
4.2.1.5 Operation Methods

- **aggregate:** input: DataSet, self.operation; output: double/int
- **relative:** input: DataSet, Metric, self.operation; output: DataSet/Metric
- **scaling:** input: DataSet, self.operation; output: DataSet

5.0 User interface design

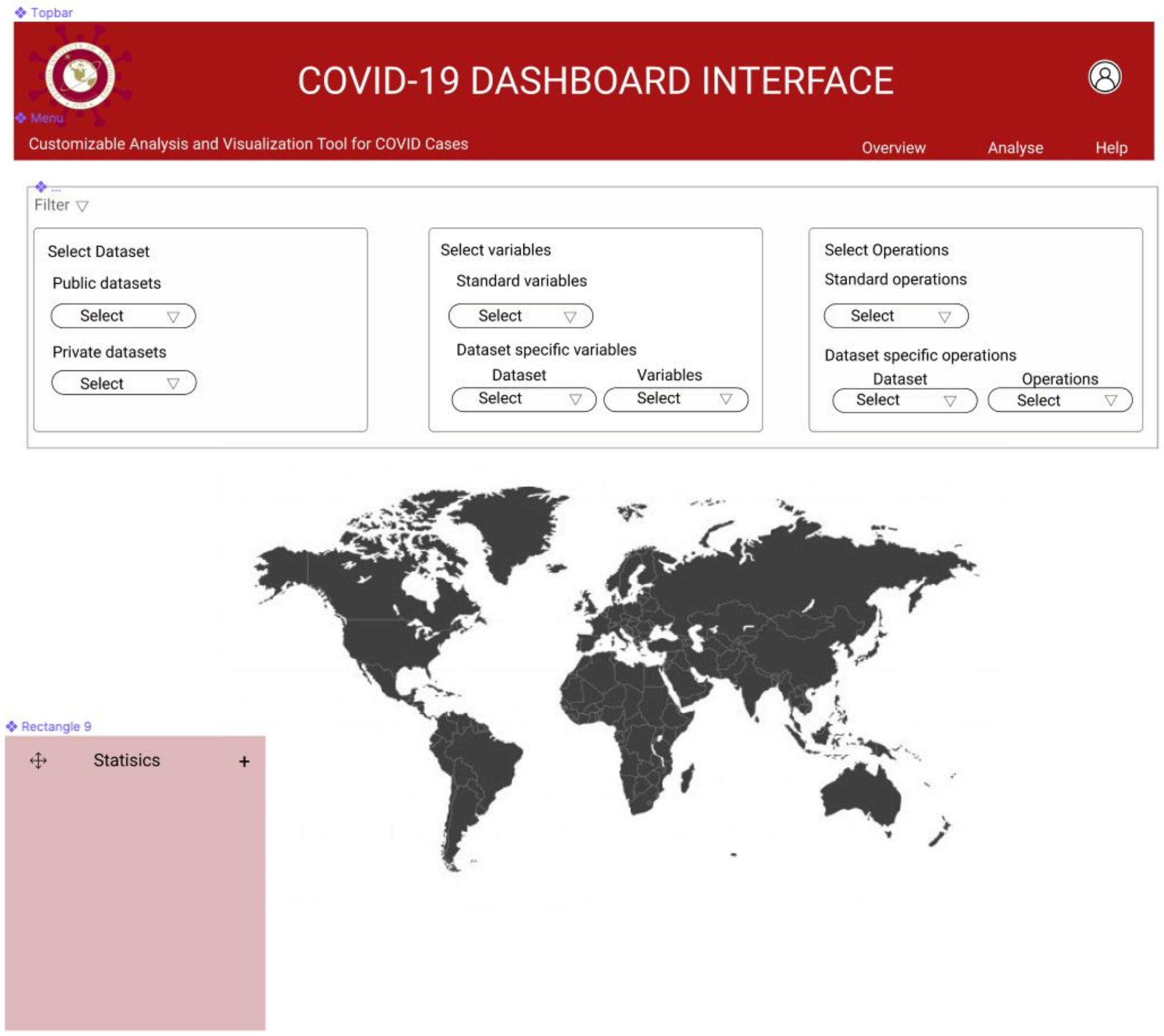
5.0.1 Screen images

5.0.1.1 Default dashboard

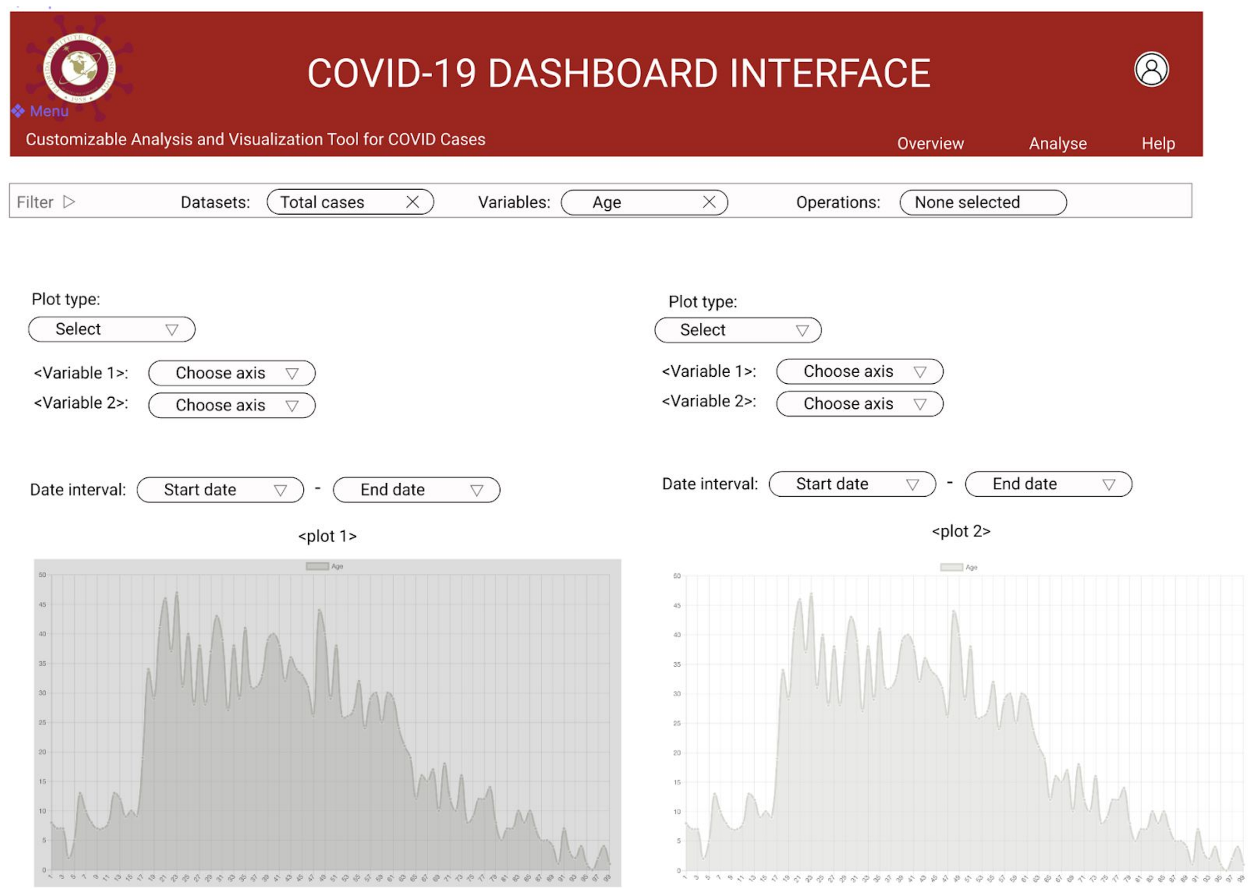

COVID-19 DASHBOARD INTERFACE
Customizable Analysis and Visualization Tool for COVID Cases
Overview
Analyse
Help
Statisics
+

5.0.1.1 Customizable Operations on Variables

This feature includes an expandable filter. The dropdowns for specific selections will only be available if a dataset is chosen. Several datasets, variables and operations can be chosen.



5.0.1.2 Customizable Plotting of Results



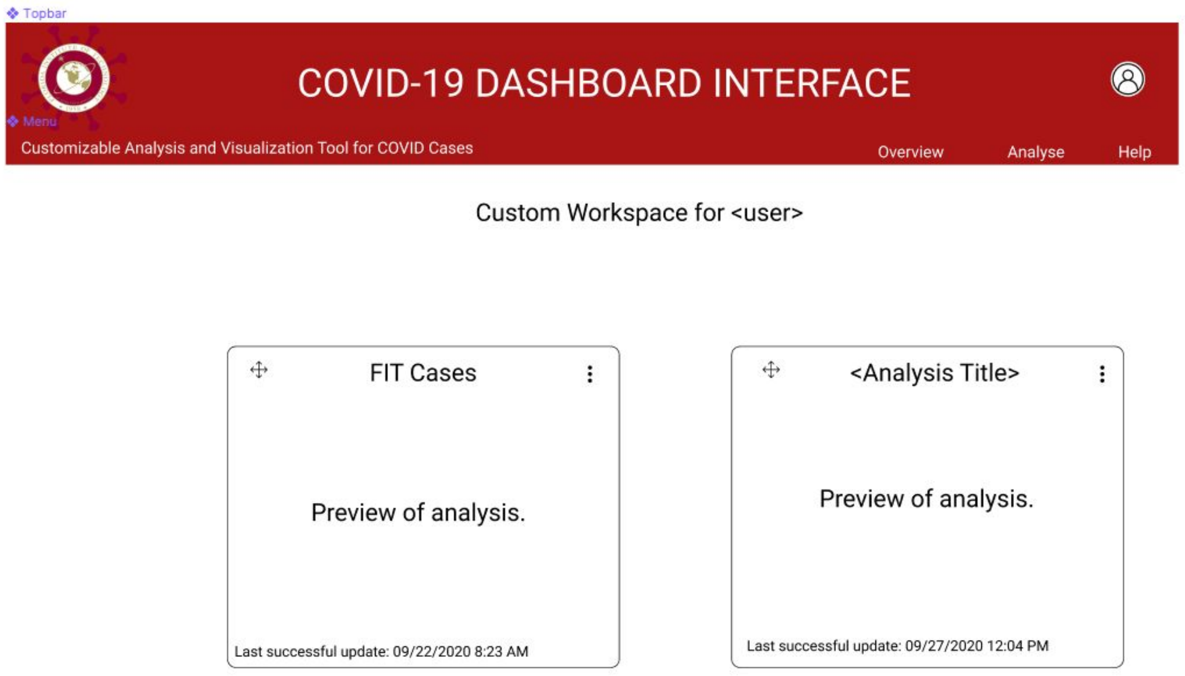
Datasets: Total cases

Variables: Age

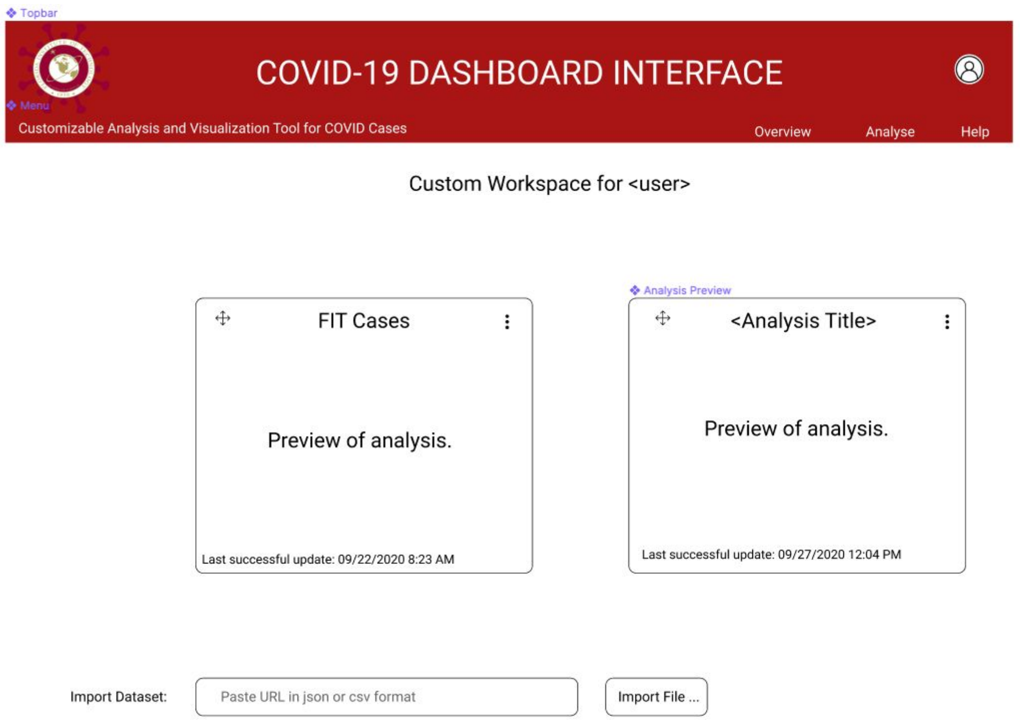
Operations: None selected

5.0.1.3 Save Customizable Plots to Custom Workspace

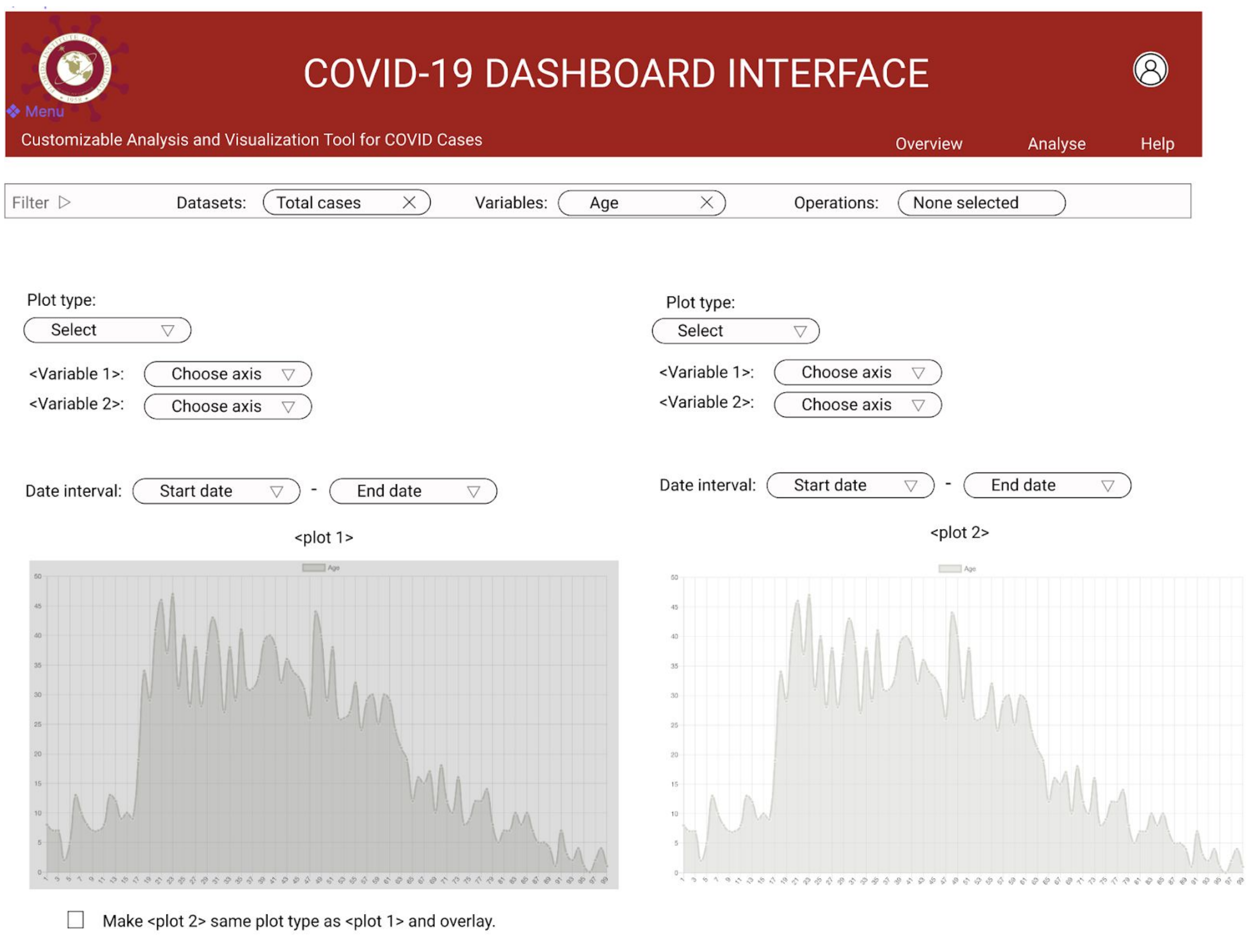
The custom workspace will contain the saved analysis. The analysis will be displayed with a preview. The user will have the possibility to click on a preferred analysis to expand it and get more information about it.


Topbar
COVID-19 DASHBOARD INTERFACE
Menu
Customizable Analysis and Visualization Tool for COVID Cases
Overview
Analyse
Help
Custom Workspace for <user>
FIT Cases
Preview of analysis.
Last successful update: 09/22/2020 8:23 AM
<Analysis Title>
Preview of analysis.
Last successful update: 09/27/2020 12:04 PM

5.0.1.4 Import Additional Datasets chosen by User


Topbar
COVID-19 DASHBOARD INTERFACE
Menu
Customizable Analysis and Visualization Tool for COVID Cases
Overview
Analyse
Help
Custom Workspace for <user>
FIT Cases
Preview of analysis.
Last successful update: 09/22/2020 8:23 AM
Analysis Preview
<Analysis Title>
Preview of analysis.
Last successful update: 09/27/2020 12:04 PM
Import Dataset:
Paste URL in json or csv format
Import File ...

5.0.1.5 Layer Datasets/Operations in a Single Plot



COVID-19 DASHBOARD INTERFACE

Menu

Customizable Analysis and Visualization Tool for COVID Cases

Overview Analyse Help

Filter ▷ Datasets: Total cases ✕ Variables: Age ✕ Operations: None selected

Plot type: Select ▽

<Variable 1>: Choose axis ▽

<Variable 2>: Choose axis ▽

Date interval: Start date ▽ - End date ▽

<plot 1>

Plot type: Select ▽

<Variable 1>: Choose axis ▽

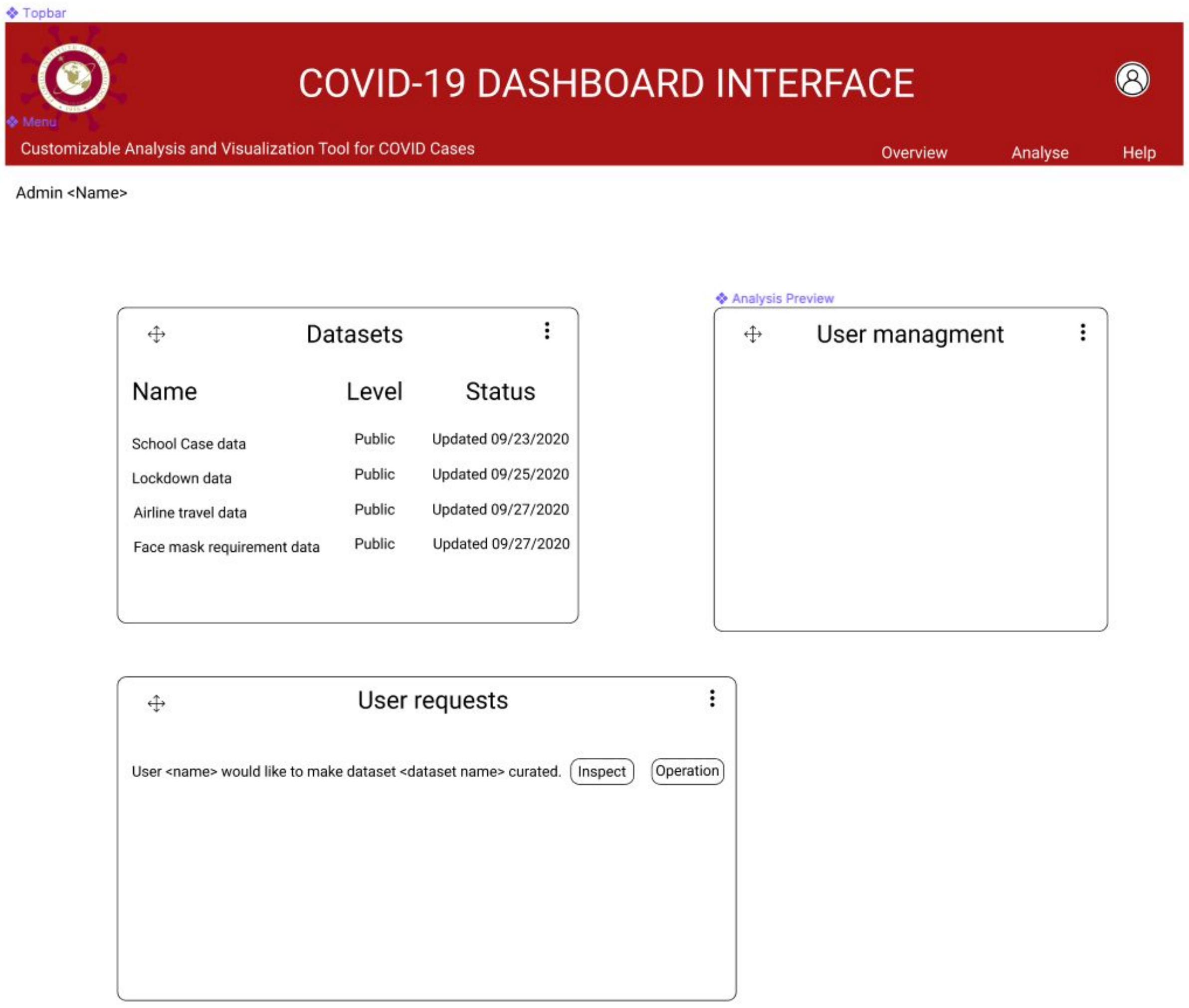
<Variable 2>: Choose axis ▽

Date interval: Start date ▽ - End date ▽

<plot 2>

☐ Make <plot 2> same plot type as <plot 1> and overlay.

5.0.1.6 Automatic Updating of Datasets and Shared, Curated, and Private Datasets



Datasets

Name	Level	Status
School Case data	Public	Updated 09/23/2020
Lockdown data	Public	Updated 09/25/2020
Airline travel data	Public	Updated 09/27/2020
Face mask requirement data	Public	Updated 09/27/2020

User <name> would like to make dataset <dataset name> curated.

5.0.2 Objects and actions

In the mock-ups, there is a toolbar and a menu which will be accessible for the user on every template in the application.

Toolbar: The toolbar contains the logo, the name of the application and an user-icon. If the user clicks on the icon, the user will be able to log in and access the customized dashboard.

Menu: The menu contains x number of tabs. Clicking on the ‘Overview’ will navigate the user to the default or customized dashboard based on if the user is logged in or not. The ‘Analyse’ tab will navigate the user to another template where the user can perform new analysis. The ‘Analyse’ template will allow the user to perform complex and customizable analysis. The

‘Help’ tab will navigate the user to a template with detailed description on how to use the application.

Default dashboard: Once a user accesses the dashboard, a set of default analysis will be displayed. From here, the user will be able to log in and access the customized dashboard. The customized dashboard will allow the user to remove the default analysis and organize the dashboard as the user likes. As the user starts doing customized analysis, the user will be able to save the analysis to the customized dashboard so the user can easily access it whenever needed.

5.1 Interface design rules

1. Consistency: Correct use of colors is important. CSS/SCSS variables will be used to make sure there is a correct use of colors, fonts, component placements.
2. Shortcuts: Abbreviations, function keys, hidden commands, macro facilities.
3. Informative feedback: Appropriate error messages will be displayed when needed. “Loading” will be displayed when retrieving data.
4. Dialogs after completed/failed tasks
5. Error handling: invalid input, required inputs, etc.
6. Easy reversal of actions.
7. Locus of control: Users should feel they’re in charge of the system.
8. Reduce short-term memory load: Remove saved analysis after a certain amount of time if it’s not been used.