

Customizable Analysis and Visualization Tool for COVID Cases

Milestone 1

Team Members

- ▶ Calvin Burns, cburns2017@my.fit.edu (Team Lead)
- ▶ Sam Hartle, shartle2017@my.fit.edu
- ▶ Nicole Wright, nwright2017@my.fit.edu
- ▶ Stian Olsen, shagboeolsen2017@my.fit.edu

Faculty Advisor/Client

- ▶ Dr. Philip Chan, pkc@cs.fit.edu

Progress Matrix

Task	Completion %	Stian	Sam	Nicole	CJ	To do
1. Investigate tools	100%	25%	25%	10%	40%	none
2. "Hello World" demos	100%	15%	35%	5%	45%	none
3. Requirements Document	100%	10%	10%	70%	10%	none
4. Design Document	100%	70%	10%	10%	10%	none
5. Test Plan	100%	10%	10%	70%	10%	none

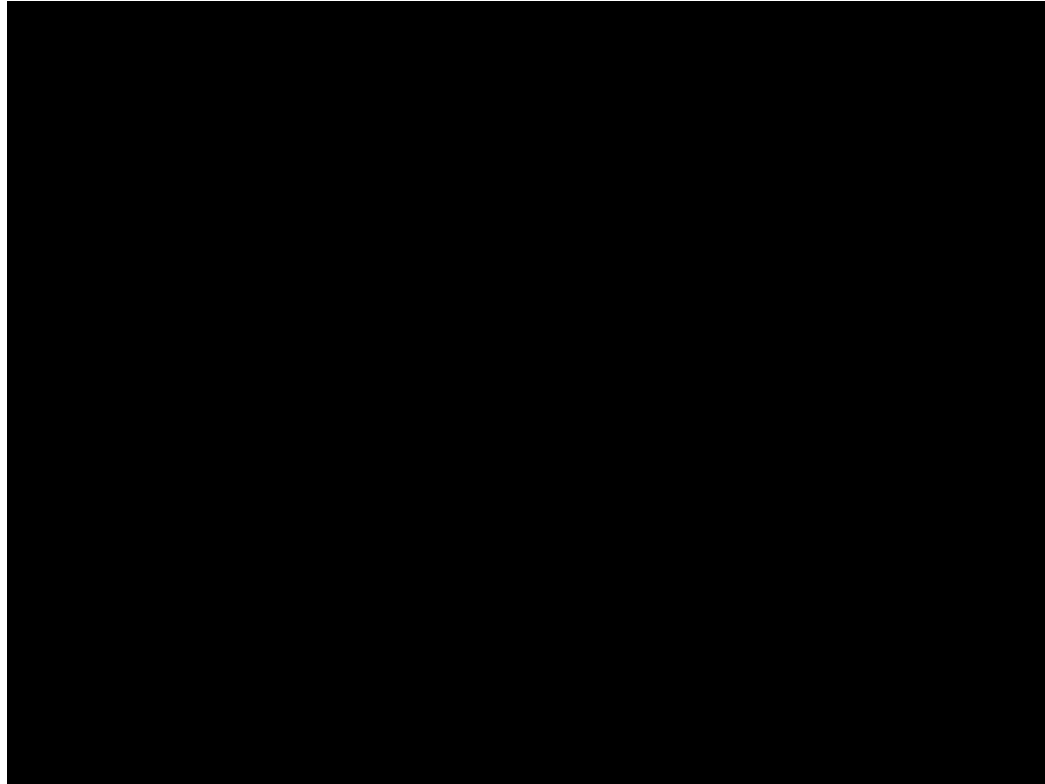
Task 1 Summary - Investigate Tools

- ▶ Frontend
 - ▶ HTML/JavaScript (Using Django Templates)
- ▶ Plotting
 - ▶ Charts.js
 - ▶ Leaflet.js
- ▶ Backend
 - ▶ Python/Django
- ▶ Databases (with GIS support)
 - ▶ SQLite with SpatiaLite extension
 - ▶ PostgreSQL with PostGIS extension
 - ▶ Provided easy-to-use admin portal that may make our lives easier
 - ▶ Leaning towards that option

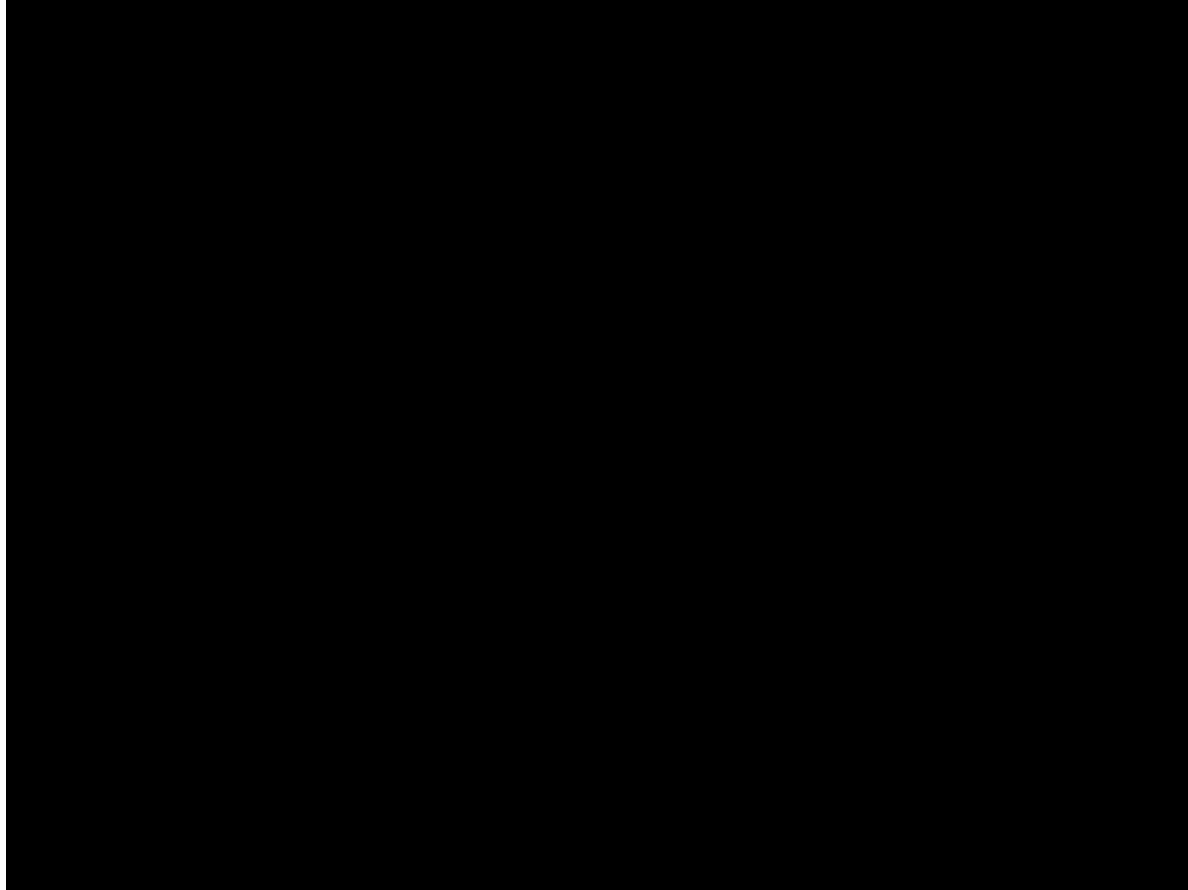
Task 2 Summary - Demos

- ▶ 4 “Hello World” demos
 - ▶ Plotting graphs using Chart.JS
 - ▶ Ability to layer GIS data on a web browser
 - ▶ Taking user input from the frontend and immediately displaying it
 - ▶ PostgreSQL with GIS support (PostGIS)

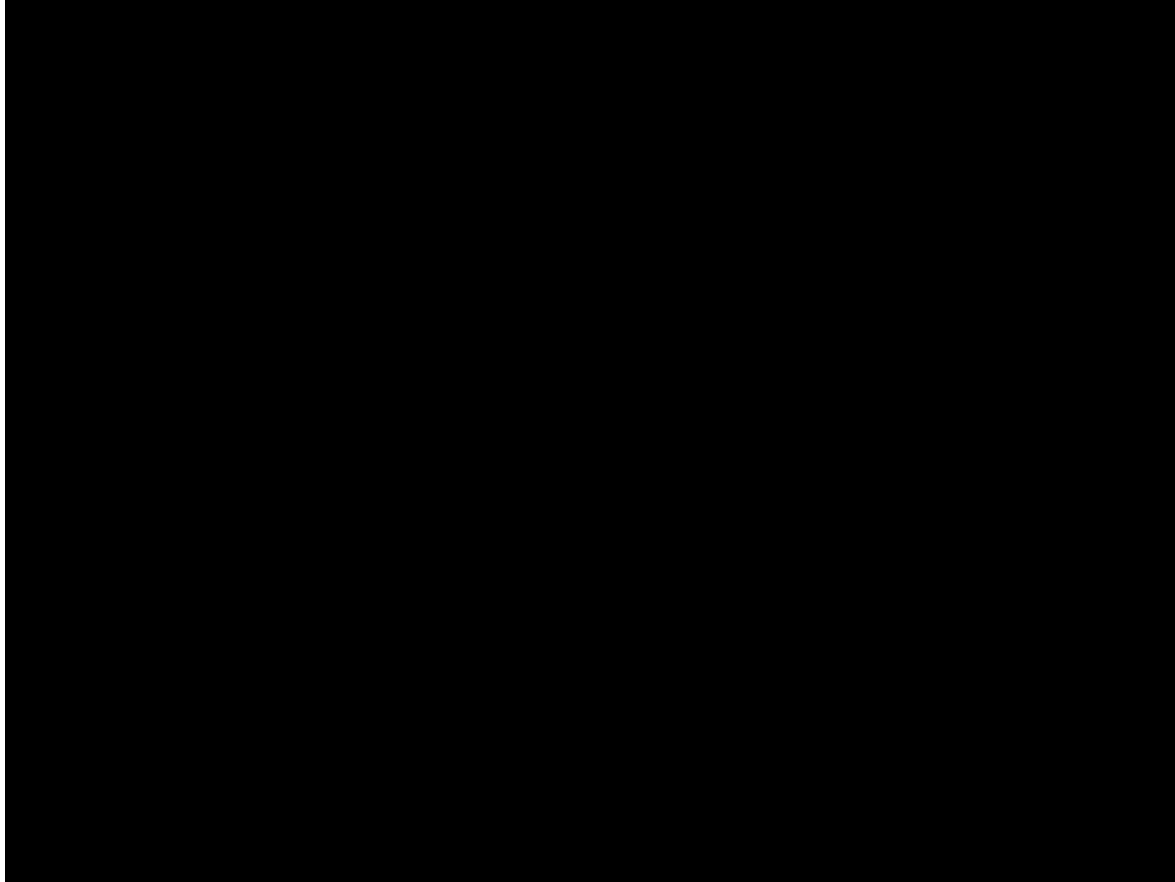
Demo - Plotting graphs with Chart.JS



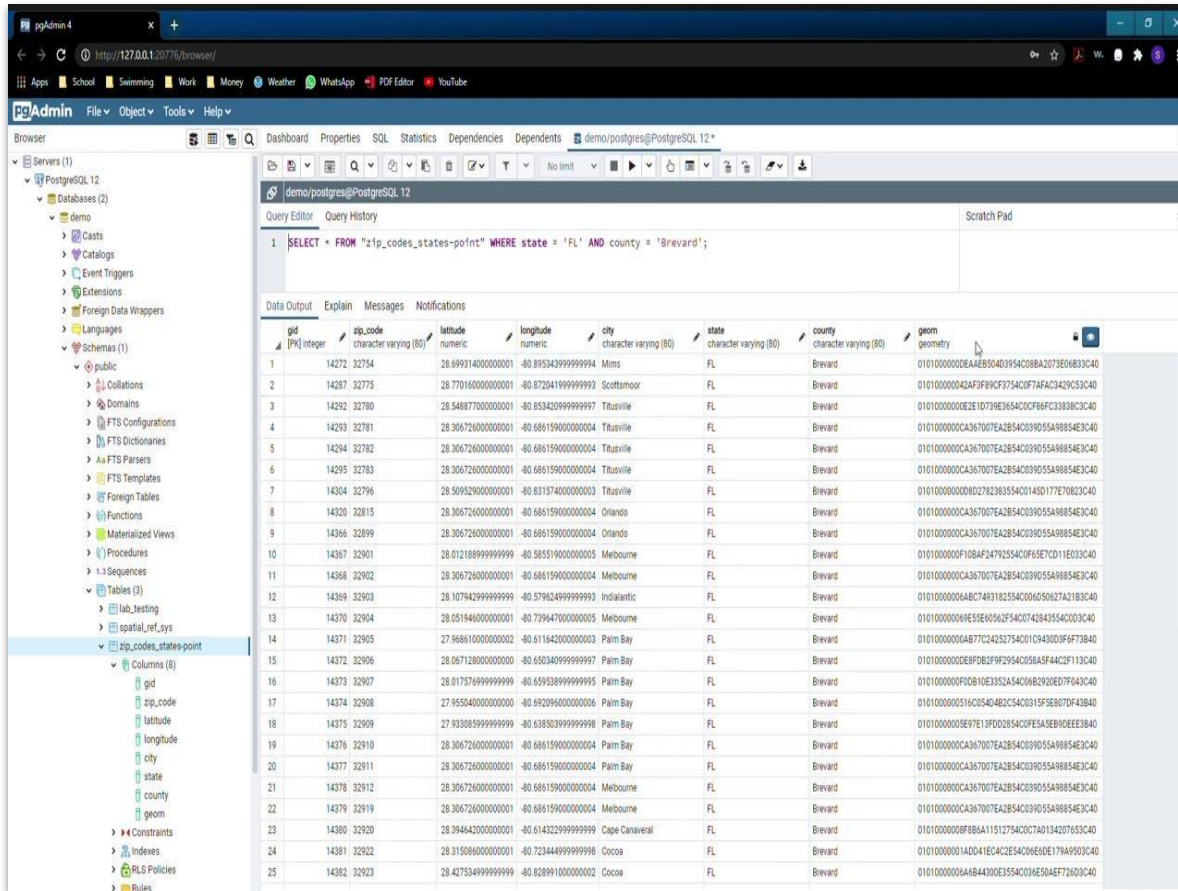
Demo - GIS Layering on a Browser



Demo - User Input



Demo - PostGIS



The screenshot displays the pgAdmin 4 web interface in a browser window. The left sidebar shows the database structure for 'demo/postgres@PostgreSQL 12', with the 'zip_codes_states-point' table selected under the 'public' schema. The main pane shows a SQL query in the 'Query Editor' and its results in the 'Data Output' table.

Query Editor:

```
1 SELECT * FROM "zip_codes_states-point" WHERE state = 'FL' AND county = 'Brevard';
```

Data Output Table:

gid	zip_code	latitude	longitude	city	state	county	geom
[PK] integer	character varying (80)	numeric	numeric	character varying (80)	character varying (80)	character varying (80)	geometry
1	14272	32754	-80.6903140000000001	Mims	FL	Brevard	01010000000E4E850403954C08BA2073E0B833C40
2	14287	32775	-80.8720419999999993	Scottsmeer	FL	Brevard	0101000000042AF3B9CF375407AF4C3429C53C40
3	14292	32780	-80.8534209999999997	Titusville	FL	Brevard	01010000000E2E10739E365AC0CF86FC33838C3C40
4	14293	32781	-80.8615900000000004	Titusville	FL	Brevard	01010000000CA367007EA2B54C039055A98854E3C40
5	14294	32782	-80.8615900000000004	Titusville	FL	Brevard	01010000000CA367007EA2B54C039055A98854E3C40
6	14295	32783	-80.8615900000000004	Titusville	FL	Brevard	01010000000CA367007EA2B54C039055A98854E3C40
7	14304	32796	-80.8315740000000003	Titusville	FL	Brevard	01010000000D802782383554C01430177E70823C40
8	14320	32815	-80.8615900000000004	Orlando	FL	Brevard	01010000000CA367007EA2B54C039055A98854E3C40
9	14366	32899	-80.8615900000000004	Orlando	FL	Brevard	01010000000CA367007EA2B54C039055A98854E3C40
10	14367	32901	-80.8551900000000005	Melbourne	FL	Brevard	01010000000F108AF34792554C0F9A5E7CD11E033C40
11	14368	32902	-80.8615900000000004	Melbourne	FL	Brevard	01010000000CA367007EA2B54C039055A98854E3C40
12	14369	32903	-80.8615900000000004	Melbourne	FL	Brevard	01010000000CA367007EA2B54C039055A98854E3C40
13	14370	32904	-80.8519460000000001	Melbourne	FL	Brevard	010100000006E35E60563F54C07428435454C033C40
14	14371	32905	-80.8611642000000003	Palm Bay	FL	Brevard	010100000000AB7C24252754C01C943003F9F73B40
15	14372	32906	-80.8503409999999997	Palm Bay	FL	Brevard	01010000000E8F20F29F2954C058A5F44C2F113C40
16	14373	32907	-80.8593899999999995	Palm Bay	FL	Brevard	01010000000F0B10E33525A4C062920E027F043C40
17	14374	32908	-80.8920960000000006	Palm Bay	FL	Brevard	01010000000516C054D482C54C0315F3E8070F43B40
18	14375	32909	-80.8385039999999998	Palm Bay	FL	Brevard	010100000005E97E13FD02854C0F3E8A5E80EE3B40
19	14376	32910	-80.8615900000000004	Palm Bay	FL	Brevard	01010000000CA367007EA2B54C039055A98854E3C40
20	14377	32911	-80.8615900000000004	Palm Bay	FL	Brevard	01010000000CA367007EA2B54C039055A98854E3C40
21	14378	32912	-80.8615900000000004	Melbourne	FL	Brevard	01010000000CA367007EA2B54C039055A98854E3C40
22	14379	32919	-80.8615900000000004	Melbourne	FL	Brevard	01010000000CA367007EA2B54C039055A98854E3C40
23	14380	32920	-80.8944820000000001	Cape Canaveral	FL	Brevard	010100000008B86A11512754C0C7A0134207653C40
24	14381	32922	-80.7234449999999998	Cocoa	FL	Brevard	010100000001A0D41ECAC254C06A0E41794A9503C40
25	14382	32923	-80.8289910000000002	Cocoa	FL	Brevard	010100000006A4B484300E354C03E5A0AEF72603C40

Task 3 Summary - Requirements

- ▶ Defined the system's individual features in detail
 - ▶ Explained functional requirements for each of them
 - ▶ Priority levels were also defined for each feature
- ▶ Obstacles included an initial lack of specificity on certain features
 - ▶ Most prevalent with higher priorities
 - ▶ Feedback from advisor/client helped us understand choices we must make to fulfill the requirements of higher priority features

Task 4 Summary - Design

- ▶ Defined the system architecture
- ▶ Layed out roles and methods each type of user will have available
- ▶ Defined various methods for handling data on both frontend/backend
- ▶ Database design, multiple “mock-ups”, and their objects/actions were included near the end of the document for a proposed UI design

Task 5 Summary - Test Plan

- ▶ Layed out features that needed to be verified for correct behavior
- ▶ Defined expected functionality
 - ▶ Provided at at least 2 test cases to verify sub-features
- ▶ Feature usage/inputs were formulated from the *user's* point of view
- ▶ Obstacles included an initial lack of test cases for testing sub-features

Challenges

- ▶ Lack of usage among group members for certain tool categories
 - ▶ Databases with GIS support
 - ▶ Frontend plotting frameworks
- ▶ Configuration process for database demos
 - ▶ CSV to Shapefile conversion for storing geometry (GIS) data posed a challenge
- ▶ Initial lack of specificity for Requirements, Design, and Test Documents

Task Matrix for Milestone 2

Task	Stian	Sam	Nicole	CJ
1. Set up Django environment	Configuring local machine (20%)	Configuring local machine (20%)	Configuring local machine (20%)	Will write majority of bash script (40%)
2. Create database model (#s are order of completion)	2: Location (easy), 8: Dashboard (JSON: Will need research)	3: Metric (easy), 6: Operation (Need to learn about SymPy)	1: User (easy), 7: Plot (Chart.js)	4: DataSet, 5: DataPoint (database relations and queries)
3. Import data to newly created database via API or CSV	Import Lockdown Data	Import Population Data	Import Mask Mandates	Import FDOH case line data via Python Script
4. Implement Feature 4.1 (Customizable Operations on Variables)	Create basic UI and display choices for variables and operations. Allow users to <i>select</i> variables and <i>perform</i> operations on them. Pass selected variables and operations to backend	Implement 40% of available operations on appropriate variables. Assist Stian with frontend development.	Implement 20% of available operations on appropriate variables. Assist Stian with frontend development.	Implement 40% of available operations on appropriate variables. Assist Stian with frontend development and oversee frontend, backend, and database interactions

Questions?