Progress Evaluation - Milestone 1

Title: Customizable Analysis and Visualization Tool for COVID Cases

Team Members:

- Sam Hartle, shartle2017@my.fit.edu
- Calvin Burns, cburns2017@my.fit.edu
- Nicole Wright, nwright2017@my.fit.edu
- Stian Olsen, shagboeolsen2017@my.fit.edu

Advisor: Dr. Philip Chan, pkc@cs.fit.edu

Client: Dr. Philip Chan, pkc@cs.fit.edu

Progress Matrix for Milestone 1:

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. Requirement 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

Discussion of each accomplished task and obstacles for Milestone 1:

- Investigating Tools:

  For the frontend, we looked at React, Angular, and ultimately decided on Django Templates to interact with each other and display data to the user. For the backend, we have decided on using Python/Django. Two database candidates with GIS support were explored (SQLite with SpatiaLite extension and PostgreSQL with PostGIS extension). Both worked well, but PostgreSQL with PostGIS did provide an easy-to-use admin portal that may make our lives easier. We are likely leaning towards that option. For testing our application, we plan on using a Heroku web server until a nicer domain is obtained near the release date for the product. We were looking into two options for software collaboration: GitHub and Microsoft Azure DevOps. Both of them have very similar features, but the GitHub Student Developer Pack gave us extra tools that will be useful in this project. Therefore, we chose to go with GitHub.

- Demos:

  We had 3 categories of “Hello World” demos for this milestone. The first was a demo demonstrating our ability to layer GIS data on a web browser. The second one was showing that we could take user input from the frontend and immediately display it on the frontend. The third and final demo was a comparison between two different databases with GIS support. The two databases and their GIS extensions were PostgreSQL with PostGIS and SQLite with a SpatiaLite extension. PostgreSQL proved more user-friendly and likely better-suited to our project due to their GUI admin tool.

- Requirements Document:

  We defined the system’s individual features in detail and what actions are required for each of them. Priority levels were also defined for each feature. Obstacles included an initial lack of specificity on certain features, specifically those with higher priorities. Dr. Chan’s notes helped us to understand the choices we must make to fulfill the requirements of these higher priority features and how they will be implemented (see Design).

- Design Document:

  We defined the system architecture, roles and methods each kind of user will have available to them in their specific role, and various methods for handling data on both the frontend and backend. Database design, multiple “mock-ups”, and their objects/actions were included near the end of the document for a proposed UI design.

- Test Plan:

  We defined the features that needed to be verified for correct behavior, their expected functionality, and at least two test cases to verify the correctness of sub-features within main features. Feature usage and test inputs were formulated from the point of view of the *user*. Obstacles included an initial lack of test cases for testing sub-features.

Discussion of contribution of each team member to Milestone 1:

- Stian: Investigated AngularJS and Typescript as an alternative for frontend development. Investigated Microsoft Azure DevOps as an alternative to GitHub for software collaboration tools. Made a demo in Angular to demonstrate how we could read csv/json files and display the data properly. Created the system architecture diagram and the user interface mock-ups which are available in the design document.

- CJ: Investigated technologies including Django, Python, GeoJSON, Chart.JS, and Leaflet.JS. Setup GitHub project boards and repositories. Created demos for plotting charts, web GIS maps, and user input. Created final database model (4.2 Design Document) and associated methods.

- Sam: Investigated PostgreSQL with PostGIS extension and SQLite with SpatiaLite extension for database candidates. Created demos showing storing and querying of shapefiles in GIS-enabled database options. Created initial database model and associated methods. Wrote majority of progress evaluation and presentation.

- Nicole: Helped investigate GIS tools and did a small demo on QGIS with layering counties over the state of Florida. Wrote the majority of the requirements document. Wrote the majority of the test plan. Made updates to the requirements and testing docs after feedback from the advisor.

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 *select* variables and *perform* 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

Discussion of each planned task for Milestone 2:

- Task 1:

  This task will ensure that each team member has the same development environment. The primary objective of this task is to create a script that will set environment variables depending on the specific OS and then start your local server.

This development environment is a Python virtual environment. A list of packages will be installed in the virtual environment, including an installation of Django and the necessary libraries(i.e. GeoJSON).

Django itself provides a set of Python scripts for creating and working with Django projects, along with a simple development web server that we will use to test local (i.e. on our machines, not on an external web server) Django web applications on our machine's web browser.

- Task 2:

  This task will ensure that we can effectively run queries on a database. The database will need a logical structure which determines how data can be stored, manipulated and organized. This database model will follow the model in section 4.2 in the design document.

  The database models will be defined in the Django models and imported/created in PostgreSQL with PostGIS extension. PostgreSQL supports both SQL and JSON querying, which makes it useful as we will be using GeoJSON.

- Task 3:

  This task involves writing a series of Python scripts to import data from the API sources we found starting with the Florida Department of Health ArcGIS server. The scripts will create instances of our database models, link, and then save. Other data sources include government action data and population data. Some data will need custom scripts to be imported.

- Task 4:

  This is the first feature we will be implementing. For frontend, this task involves making a user interface which allows the user to select variables and operations. The frontend will also filter out any irrelevant data from the datasets and display it in an organized way so the user can easily compare filtered data.

  For backend and database, this task involves processing the custom operations on selected variables. An extensive list of operations is listed in section 4.1.1 of the requirements document. We anticipate this processing will involve using the SymPy tool mentioned above in the task matrix for Milestone 2. SymPy is a Python library for symbolic mathematics. It will allow us to easily perform these operations in our backend code and display them on the frontend. A more detailed description can be found in section 4.1 of the requirements document.

Date(s) of meeting(s) with Client/Advisor (same) during Milestone 1:

- September 11th
- September 25th

Client feedback on the current milestone: See Faculty Advisor Feedback below

Faculty Advisor feedback on each task for Milestone 1:

- Task 1:

  Select a tool for each category (frontend, backend, database, plotting, etc)

- Task 2:

  Integrated demo of multiple tools. Goal is to run a server with a Django backend that uses a PostgreSQL (PostGIS) database. We want to render the frontend using Django Templates and utilize Leaflet.JS for mapping and Chart.JS for graphing. Data will be transformed into the appropriate JSON type (GeoJSON for maps, Chart.JS JSON for graphs) before being passed to the frontend using Python.

- Task 3:

  “On a variable that has the same name, but from different datasets, a better example is that cases from the Florida health dept dataset have an age variable, and cases from the Texas health dept dataset also have an age variable. We can assume each dataset name is unique because when a user tries to share a dataset to the public, the system can check and refuse duplicate dataset names. That is, variable names, including dataset names, are unique.

  On each team member identifying 10 operations from a different dashboard/website, I suggest you to think about:

  1. given the case dataset from the FL health dept (or a similar dataset from another state), where each case has ~20? variables/attributes,

  2. What are the operations needed to get the values on the various plots on a dashboard?

  As I mentioned during our first meeting, I think the dataset from the FL health dept is the original source of data for FL; that is, other websites/dashboards applied some operations to get the values in their Plots.” - Dr. Chan’s email notes for Requirements Document on 9/26

- Task 4:

  “Please update/add the different "mock up" screens for the different features from the Requirement Document. For example, on selecting variables and/or operations (your first feature), I don't see a "mock up" screen to show how one may be to do that.” - Dr. Chan’s email notes from 9/25 for Design Document

  Dr. Chan’s In-Meeting Notes from 9/25 for Design Document:
  - Decide format for resolving naming conflicts (Feature 4.4)
  - Specify our choice of information that needs to be saved (Feature 4.3)
  - Specify our choice of saving format (Feature 4.3)

  “Looks like more thoughtful designs are needed for the basic design of GUI.

  For example, in simple plots, usually at least 2 variables are involved (to see possible relationship/trend/...), one for the x-axis and one for the y-axis. The current GUI doesn't seem to allow specifying two variables, …

  Using the dataset from the FL health dept, the most basic plot I think would be the (new) case count per day:

  a. how does one specify dates as x and case count as y?

  b. how does one specify the count operation on the cases for each day?

  c. [slightly more involved, how does one specify a range of dates? e.g. focusing on the second spike in the June/July/August...]”

  - Dr. Chan’s email notes from 9/28 for Design Document

- Task 5:

  “I suggest you to have at least 2 test cases for each feature, testing different "sub"features.” - Dr. Chan’s email notes from 9/25 for Test Plan