Documentation (CovidApp)

Overall Project Structure:

- Data Source – JOHNS HOPKINS GitHub Repository
  - https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data/csse_covid_19_time_series
- Database – Support for MySQL/SQLite
- Client - Angular 10 (TypeScript)
- Server - Flask (Python)

Database:

- Tables
  - *Countries – List of all countries available*
  - *States/Provinces/Territories – Lists all for a country if available*
  - *Counties (*US Only)* – lists all counties in the US:
    - FIELDS: id, fips_code, county, state, latitude, longitude
  - *Users* – stores all registered user information:
    - *FIELDS*: id, firstName, lastName, username, email, password, county, state
  - *Individual (Countries, states,provinces, territories)* – 3000+ tables representing each location, *each table has daily records dating back to January 2020*
    - *FIELDS: id, cases, deaths, time*
    - separated individually for performance reasons
    - store daily COVID records for the location
    - naming convention:
      - country
      - country_state
      - country_province
      - country_territory
      - country_state_county

Client (Docker Image):

- Components
  - Home
    - Threejs visualization of earth
      - Show data for all locations available
    - Allow user to see 30 days of county records (no predictions, no twitter sentiment analysis)
  - Login
  - Registration
  - Dashboard
    - Displays 4 different quick county information charts
  - Reports

By Jesualdo Cristovao

        - Records – displays table of county records
        - Predictions – displays table of county predictions for next 20 days
    - Twitter Feed
        - Displays tweets and shows sentiments in the tweet
    - Settings
        - Allow to update user profile information
    - Sidebar
        - Collapsible navigation bar
    - Header
        - Contains basic information as well as some navigation buttons
    - Footer

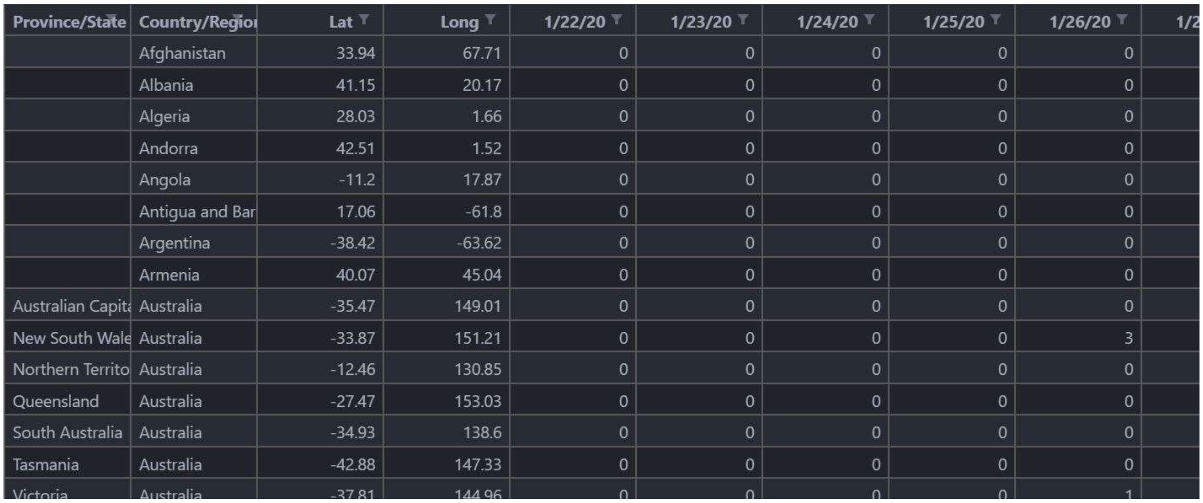
- Services
    - Login
        - Handles all the login process including retrieving and making Json Web Token (JWT) available for the service to use when making requests to server protected routes
    - Data
        - Performs most of the data requests to the server (*uses JWT retrieved by login to access protected routes in the server)
        - All data used client are retrieved from own server
    - Route Guards
        - Manages access to routes

Note: Components are by most part lazy loaded with exception of layout components and the login component

Server (Docker Image):

- *Restful API*
    - Server API endpoints were exposed using Flask
    - JWT authentication was implemented
    - Rate limiter for routes implemented
    - Multiples access header were also defined though not strictly filtered
    - The API exposes 11 different HTTP endpoints (JSON requests and reponses)
        - Globe (GET)
            - *Returns all the data used for visualization by the Threejs animation*
        - Login (POST) <username, password>
        - Registration (POST) < firstName, lastName, username, email, password, country, state, county>
            - Add new user for access to predictions and tweet swntiment analysis
        - Usernames (GET)
            - Returns list of usernames already registered, used by the client to make use usernames are unique
        - Data (POST) <country, state, county, days>

            - Returns all the COVID records for the county requested for some number of days
        - Countries (GET)
            - Returns list of all countries available
        - States (POST) <country>
            - Returns list of states, provinces, territories of a country
        - Counties (POST) <country, state>
            - Returns list of counties of a state (US Only), used by the client for registration to prevent user from incorrectly linking county and states
        - Predictions (POST) <country, state, county>
            - Performs and returns predictions for a particular county
        - Twitter (POST) <country, state, county>
            - Performs sentiment analysis, returns tweets as well as sentiments attached to the tweets
        - Profile Information (PUT) < firstName, lastName, username, email, password, country, state, county>
            - Used to update user information
- *Covid Data*
    - Covid data was retrieved from JOHNS HOPKINS GitHub Repository
    - Data is retrived once a day at midnight to update the application's database
    - Daily records are organized by columns



Province/State	Country/Regior	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/2
	Afghanistan	33.94	67.71	0	0	0	0	0	
	Albania	41.15	20.17	0	0	0	0	0	
	Algeria	28.03	1.66	0	0	0	0	0	
	Andorra	42.51	1.52	0	0	0	0	0	
	Angola	-11.2	17.87	0	0	0	0	0	
	Antigua and Bar	17.06	-61.8	0	0	0	0	0	
	Argentina	-38.42	-63.62	0	0	0	0	0	
	Armenia	40.07	45.04	0	0	0	0	0	
Australian Capit	Australia	-35.47	149.01	0	0	0	0	0	
New South Wale	Australia	-33.87	151.21	0	0	0	0	3	
Northern Territo	Australia	-12.46	130.85	0	0	0	0	0	
Queensland	Australia	-27.47	153.03	0	0	0	0	0	
South Australia	Australia	-34.93	138.6	0	0	0	0	0	
Tasmania	Australia	-42.88	147.33	0	0	0	0	0	
Victoria	Australia	-37.81	144.96	0	0	0	0	1	

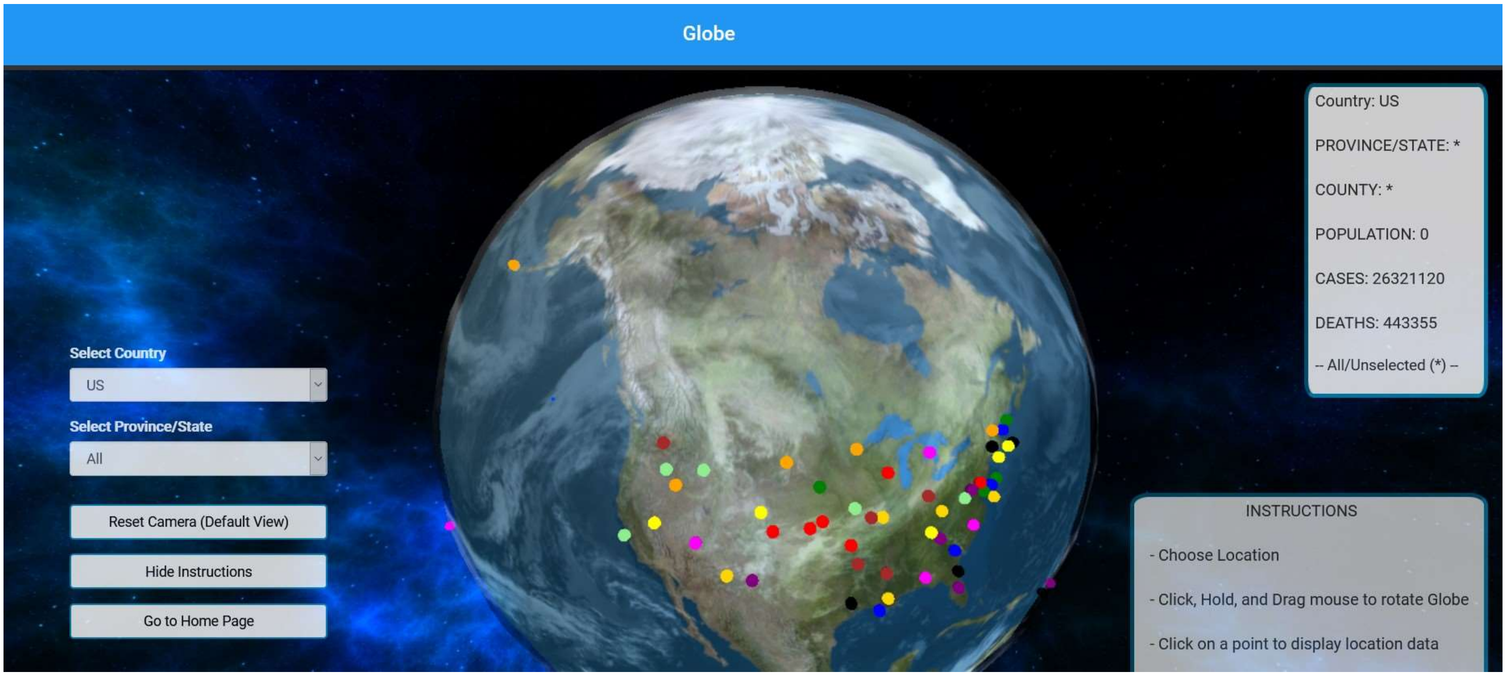
- *DB Connection*
    - MySQL and SQLite support
    - Used standard SQL queries to interact with MySQL and SQLite databases
    - Constant check to reconnect to make use it is always to attend database timeouts settings set
- *Prediction*
    - Use *Sklearn* to perform polynomial regression on COVID data and predict number of cases and deaths in each county
    - Model is scheduled to be retrained every day after midnight upon addition of day's numbers
    - Predictions are performed on demand using saved model for each county

- *Twitter Sentiment Analysis*
    - Used *Tweepy* package to easily interact and retrieve tweets from Twitter API (*Required a Twitter developer account first)
    - Used *Textblob* package to perform sentiment analysis (primary)
    - Used *Sklearn* and some training data to perform sentiment analysis as well, with various results

---

The entire application was dockerized, using three different Images for each (Client, Server, DB), and then deployed to the Cloud.

SOME SCREENSCHOTS



Select Country: US

Select Province/State: All

Country: US

PROVINCE/STATE: *

COUNTY: *

POPULATION: 0

CASES: 26321120

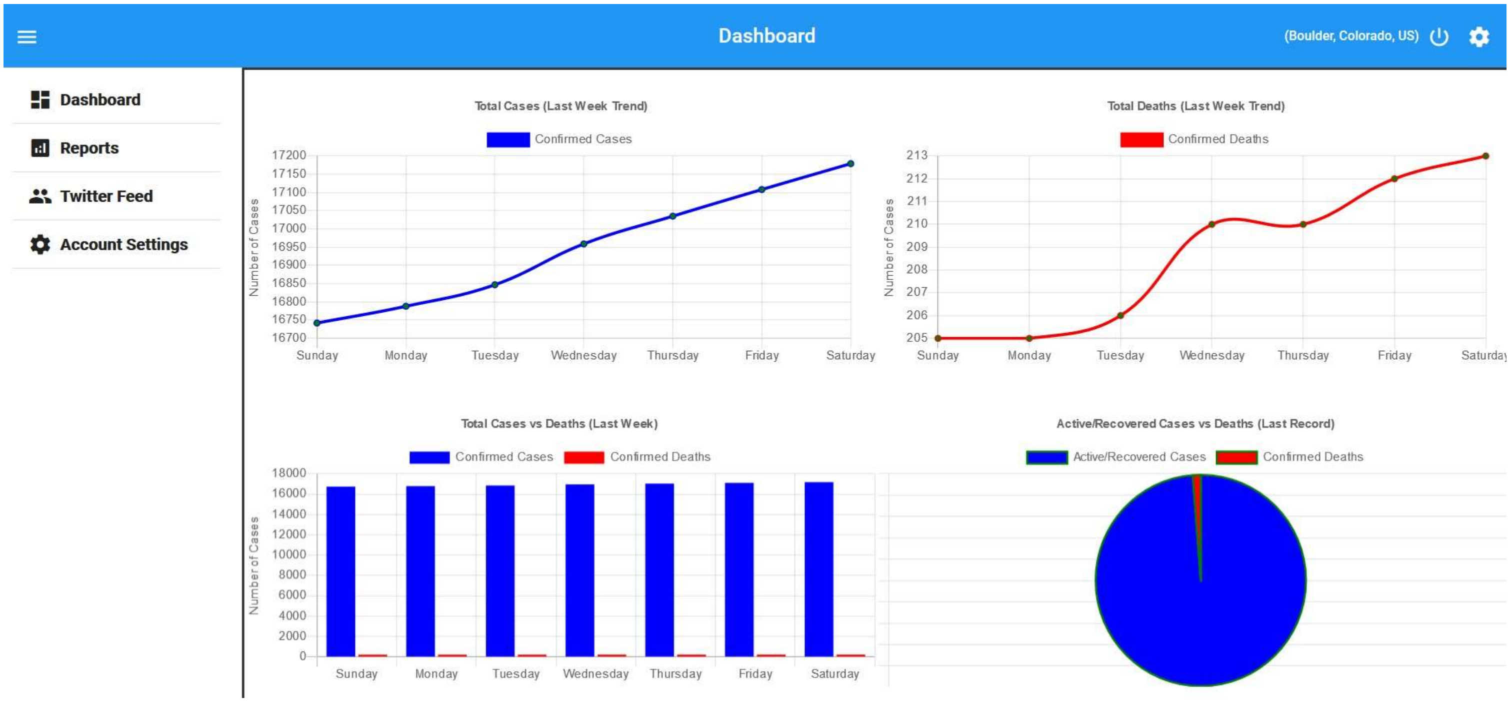
DEATHS: 443355

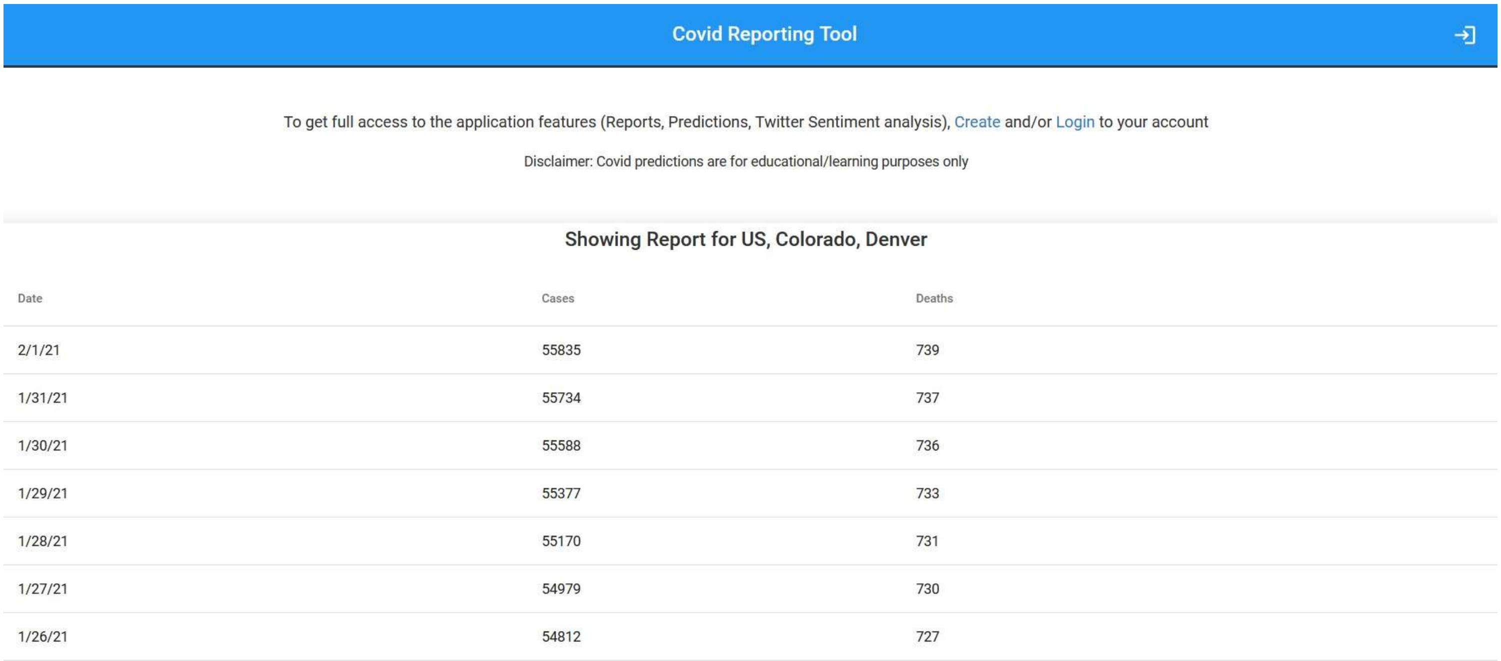
-- All/Unselected (*) --

INSTRUCTIONS

- Choose Location
- Click, Hold, and Drag mouse to rotate Globe
- Click on a point to display location data

By Jesualdo Cristovao





Covid Reporting Tool

To get full access to the application features (Reports, Predictions, Twitter Sentiment analysis), Create and/or Login to your account

Disclaimer: Covid predictions are for educational/learning purposes only

Showing Report for US, Colorado, Denver

Date	Cases	Deaths
2/1/21	55835	739
1/31/21	55734	737
1/30/21	55588	736
1/29/21	55377	733
1/28/21	55170	731
1/27/21	54979	730
1/26/21	54812	727

By Jesualdo Cristovao

HOW TO RUN APP

1) Go Backend/app
    a. Run “pip install -r requirements.txt” to install all dependencies
2) Go Backend/app/db
    a. Open sql_conector.py
    b. Instantiate a DbManagement instance (*choose between the default sqlite, and MySQL)
    c. Run function initial_set_up() to setup all tables and data in the db
    d. Delete instance
3) Go Backend/app
    a. Run app.py to start server

4) Go to Frontend/covidapp/src/app
    a. Run “npm install” to install all dependencies
    b. Run “ng serve” to start client
    c. Go to browser and navigate to URL

By Jesualdo Cristovao