Documentation (CovidApp)

Overall Project Structure:

- Data Source – NY Times Github repo
    - https://github.com/nytimes/covid-19-data/blob/master/live/us-counties.csv
- Database - MySql
- Client - Angular 10 (TypeScript)
- Server - Flask (Python)

Database:

- Tables
    - *Counties* – lists all counties in the US: fips code, county, state, latitude, longitude
    - *Users* – stores all registered user information: id, firstName, lastName, username, email, password, county, state
    - *Individual counties* – 3000+ tables representing each county (separated for performance reasons) named using unique fips code identifier that stores covid records for the county wich contains the following: cases, deaths, confirmed_deaths, confirmed_cases, probable_cases, probable_deaths

Note: Fips – unique indentifier for each county in the US

Client (Docker Image):

- Components
    - Login
    - Registration
    - Dashboard
        - Displays 4 different quick county information charts
    - Reports
        - Records – displays table of county records
        - Predictions – displays table of county predictions for next 10, 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

By Jesualdo Cristovao

        - 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

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 8 different HTTP endpoints (JSON requests and reponses)
        - Login (POST) <username, password>
    - Registration (POST) < firstName, lastName, username, email, password, county, state>
        - Usernames (GET)
            - Returns list of usernames already registered, used by the client to make use usernames are unique
        - Data (POST) <county, state, days>
            - Returns all the Covid records for the county requested for some number of days
        - Counties (POST) <state>
            - Returns list of counties in a specified state, used by the client for registration to prevent user from incorrectly linking county and states
        - Predictions (POST) <county, state>
            - Performs and returns predictions for a particular county
        - Twitter (POST) <county, state>
            - Performs sentiment analysis, returns tweets as well as sentiments attached to the tweets
    - Profile Information (PUT) < firstName, lastName, username, email, password, county, state>
            - Used to update user information
- *Covid Data*
    - Covid data was retrieved from NY times github repository which updates county data daily
    - Access to commit history was performed due to the way the repository set up in which only the latest data is display (*only last days numbers are show). Since this app required to show number for past days as well, that was accomplished by searching through the git commits
    - Some filtering had to be done due the fact that the repository is updated multiple times a days which resulted in having multiple commits for a day
    - Data is retrived once a day at midnight to update the application's database
- *DB Connection*

    - Used standard python mysql connector to interact with database using standard SQL queries
    - Constant check to reconnect to make use it is always to attend database timeouts settings set on Azure
    - Requires SSL certificate (provided in the repo) to interact with MySQL database in the cloud
- *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 with the exception of the database. In total two docker images were used (one for client and one for the server) and then deployed to the Cloud. The database was deployed using direct MySQL cloud services

By Jesualdo Cristovao