COVID Dashboard

ECM1400 Coursework Project

The aim of this coursework was to create a personalized data dashboard which would provide the user with up-to-data COVID statistics, along with relavant news stories.

This was implemented using various modules, including:

- flask >> Used to run the web application, implementing the interface using the *index.html* file provided
- sched >> Used to schedule updates to the interface (statistics and news) at times specified by the user
- uk_covid19 and requests >> Used to fetch data from the relevant APIs

Along with other backend modules:

- json >> Used to load API responses into a readable dictionary format
- logging >> Used to track the program during runtime
- pytest >> Used to run tests

Prerequisites / Installation

Python Version

This project is intended to be run using Python 3.9 (64-bit)

Installing Required Modules

A list of modules to be installed can be found in *requirements.txt*. Installing these modules can be done by running *requirements.bat*, or manually by executing `pip install -r requirements.txt` in the project directory.

API key

In order for news stories to be displayed, a key for the NewsAPI is required. Visit https://newsapi.org/ and create a free account. Open *config_template.json*, and replace `[api-key]` with your key. Once this is done, make sure to rename this file to *config.json*.

Personalising

To display Covid stats for a different location, change the "location" in *config.json* to the desired location. To change the search criteria for the displayed news stories, modify "news_search_terms" in *config.json* (if using multiple search terms, separate each term with a space).

Running the Dashboard

Now, to start the dashboard, run *dashboard.bat*. This will automatically run the test suite, host the flask application, and open the correct url in the default browser.

Runtime Errors

Any runtime errors are logged in the .log file (*/__log__/dashboard_log.log* by default, but this can be modified from within the config file).

Testing

To manually run the test suite, execute `python -m pytest` in the project directory.
The test suite contains tests for both modules, including tests for accessing the api.
There is also a set of tests for the validity of the *config.json* file.

Developer Documentation

External Module Documentation

Below is a list of links to the documentation for any modules used in this project.

Standard Library:

- json
- logging
- os
- sched
- time

Third-Party:

- flask
- pytest
- requests
- uk_covid19

Function Summary

A summary of the functions from both supplementary modules is detailed below.

covid_data_handler.

- parse_csv_data() >> see process_csv_data
- process_csv_data() >> used in conjunction to extract data from a static file
- covid_API_request() >> utilises the uk_covid19 module to request data
- get_stats_from_json() >> extracts a specific metric from json returned from the above function
- get_covid_stats() >> utilises the previous function to get a set of metrics for the interface
- update_covid_data() >> updates a global data structure with the output of the previous function
- sched_covid_update_repeat() >> recursively schedules update_covid_data every 24 hours
- schedule_covid_updates() >> schedules update_covid_data after an interval

covid_news_handling.

- news_API_request() >> utilises the requests module to request news stories

- format_news_article() >> injects article information into a format compatible with the interface
- remove_title() >> marks an article as "seen"
- purge_articles() >> calls remove_title on all currently displayed articles
- update_news() >> updates a global data structure with (formatted) news articles
- sched_news_update_repeat() >> recursively schedules update_news every 24 hours
- schedule_news_updates() >> schedules update_news after an interval

Docstrings

Below are the docstrings for each module, and the contained functions

main

This module handles: the main flask application; incoming client requests (leading to scheduling/cancelling updates) and updates to the interface (by passing values into the template).

```
index():
    Handles incoming client requests, and injects values into the interface
```

covid_data_handler

This module handles: uk-covid-19 api requests; fetching up to date stats and scheduling stats updates.

```
parse_csv_data(csv_filename: str) -> list:
    Returns list of strings for rows in the file.

    Args:
        csv_filename: Filename of static csv file

    Returns:
        Lines from file


process_covid_csv_data(covid_csv_data: list) -> tuple[int, int, int]:
    Extracts covid stats from csv format.

    Args:
        covid_csv_data : List of lines from covid data csv - as returned from
parse_csv_data

    Returns:
        Three metrics, detailed below, from the csv lines list.

        last7days_cases_total: Summative latest week case count
        current_hospital_cases: Current hospital cases
        total_deaths: Latest death toll
```

```
covid_API_request(location: str=None, location_type: str=None) -> dict:
    Returns a json containing the set of metrics.

    Args:
        location: Area code for api request
        location_type: Area type for api request

    Returns:
        A dictionary containing the metrics specified in the metrics[dict]
variable.
        The format of the returned dictionary, along with types, is detailed
below.

        {
            data[list][dict] : {
                date[str]: %format YYYY-MM-DD
                areaName[str]: as specifed in area[list][str]
                areaType[str]: see above
                newCasesByPublishDate[int]: case count
                cumDeaths28ByPublishDate[int]: death toll
                hospitalCases[int]: hospital cases
            }
            lastUpdate[str]: time of latest entry, %format YYYY-MM-DDtime
            length[int]: number of entries fetched from api
            totalPages[int]: number of pages fetched from api
        }


get_stats_from_json(covid_stats_json: dict, metric: str, count: int=1, skip:
bool=False) -> tuple[str, int]:
    Extracts the specified metric from a json.

    Args:
        covid_stats_json: raw json - as returned from covid_API_request
        metric: stat to extract
        count: number of entries to cache (used for 7 day sums)
        skip: flag for skiping first entry (more accurate for certain metrics)

    Returns:
        The area code associated with the api request, along with the value
requested.


get_covid_stats() -> tuple[str, int, str, int, int, int]:
    Fetches relevant statistics to be displayed in the interface.

    Returns:
        Area codes (local and national), along with 4 statistics.
        Detailed below.
```

```
        area: local area code - stored in config.json
        last7days_cases_local: summative previous week case count (local)
        nation: nation area code
        last7days_cases_nation: summative previous week case count (nation)
        hospital_cases: current hospital cases
        total_deaths: cumulative death toll


update_covid_data() -> type(None):
    Updates the covid_data data structure (global) with the latest stats.


sched_covid_update_repeat(sch: sched.scheduler) -> type(None):
    Uses recursion to implement 24 hour repeating updates.

    Args:
        sch: associated scheduler


schedule_covid_updates(update_interval: float, update_name: str, repeating:
bool=False) -> type(None):
    Creates scheduler (stored in covid_data_sch) and schedules news updates.

    Args:
        update_interval: delay of (initial) scheduled update
        update_name: label of update in interface - index of scheduler in
covid_data_sch
        repeating: flag for repeating updates (every 24 hours)
```

covid_news_handling

This module handles: news-api requests; fetching (and formatting) new news stories; scheduling these news updates.

```
news_API_request(covid_terms: str=None,page_size: int=20) -> dict:
    Fetches covid-related news stories from the news api.

    Args:
        covid_terms: search terms for api request
        page_size: number of articles fetched from api

    Returns:
        A dictionary containing news articles returned from the api.
        The format of the returned dictionary, along with types, it detailed
below.

        Expected (i.e. no error):
        {
            status[str]: flag for request successful, "ok"
```

```
            totalResults[int]: article count
            articles[list][dict]: {
                source[dict]: {
                    id[str]: identifier
                    name[str]: display name
                }
                author[str]: author name
                title[str]: article title
                description[str]: short description of article
                url[str]: link to article
                urlToImage[str]: link to related image for article
                publishedAt[str]: %format YYYY-MM-DDtime
                content[str]: article content, max 200 characters
            }
        }

        Exception (i.e. error):
        {
            staus[str]: as before, "error"
            code[str]: see https://newsapi.org/docs/errors, (200, 400, 401, 429,
500)
            message[str]: error description
        }


format_news_article(article_json: dict) -> dict:
    Formats news article into a dictionary which can be input into the flask
template.

    Args:
        article_json: raw json representing a single article

    Returns:
        A dictionary representing a news article.
        The format of the returned dictionary, along with types, it detailed
below.

        {
            title[str]: title of article
            content[str]: short description and ling (formatted with flask.Markup)
        }


remove_title(title: str) -> type(None):
    Adds article to (global) list of removed articles, so it is not redisplayed
when the interface is updated.

    Args:
        title: title of article to be removed
```

```
purge_articles() -> type(None):
    Removes all currently displayed articles (i.e. marked as seen, not
redisplayed).


update_news(covid_terms: str=None, article_count: int=10, sch: bool=True) ->
type(None):
    Updates the covid_news data structure (global) with the latest articles.

    Args:
        covid_terms: search terms for news_api request - stored in config.json
        article_count: number of articles to be displayed in the interface
        sch: flag for scheduled (over intermittent) updates


sched_news_update_repeat(sch: sched.scheduler) -> type(None):
    Uses recursion to implement 24 hour repeating updates.

    Args:
        sch: associated scheduler


schedule_news_updates(update_interval: float, update_name: str, repeating:
bool=False) -> type(None):
    Creates scheduler (stored in covid_news_sch) and schedules news updates.

    Args:
        update_interval: delay of (initial) scheduled update
        update_name: label of update in interface, index of scheduler in
covid_news_sch
        repeating: flag for repeating updates (every 24 hours)
```

Other Details

Author

Thomas Newbold
tn337@exeter.ac.uk

Specification

https://vle.exeter.ac.uk/pluginfile.php/2954508/mod_label/intro/CA-specification.pdf

ECM1400 Programming Continuous Assessment

This CA comprises 100% of the module assessment.
Set: 1st November 2021
Due: 10th December 2021 @ 11:59am