**Code with Annotations**

DIGITAL SOLUTIONS IA2 ⏐ CAITLIN PETT ⏐ 2024 ⏐ YEAR 12

**Introduction**

This Document contains the annotated code to create the QPC website. Python, HTML, JavaScript, CSS, and SQL languages were used to create the page, and it was written on the code editor Visual Studio Code. White space is minimised to fit code in 4 pages.

1. **Python Flask project structure (viewed in VSCode)**

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a black screen

Description automatically generated

/templates – the html web pages inc. JavaScript and jinja code

/\*.py the service side python code (app.py is the primary file)

/requirements.txt the additional python libraries

/QPC.db the sqlite database

/DATA the downloaded csv gov data

/static – images and CSS

**A screenshot of a computer screen

Description automatically generated**

Database tables (see create statement py scripts for details)

* bus\_stops gov bus stop data
* properties test real estate prop
* schools gov schools data
* users user data

1. **Python flask application (app.py)**

App.py contains the server-side python code that does not create any physical elements to the website, but instead listens to the http requests from the web pages. The code uses the Python Flask framework to setup routes for each request the website can do (coded elements). The routes process the requests such as a GET and POST by performing the business logic such as validation and storing the data in the database.

App.py - continued

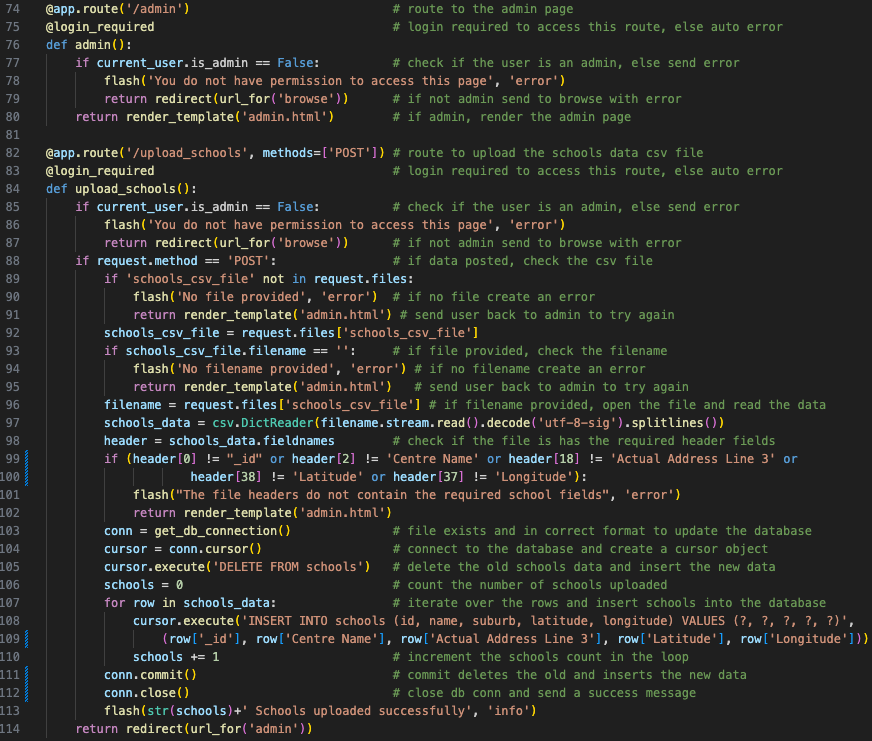
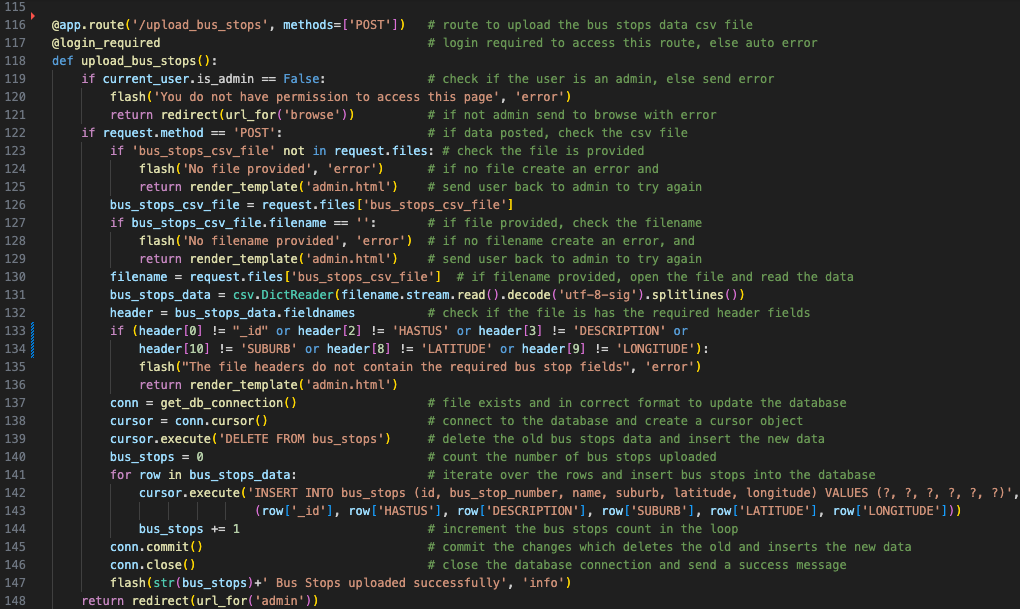
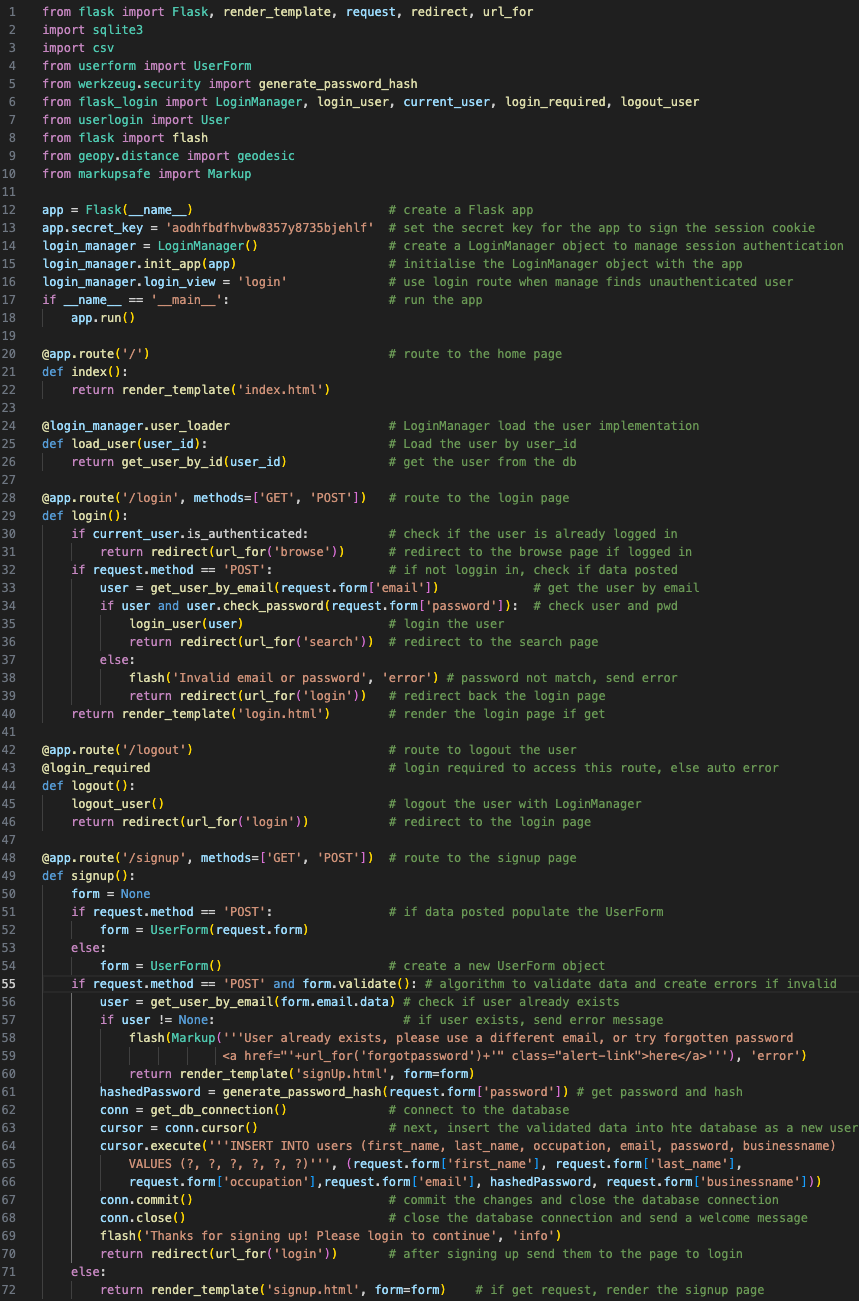
- create route for admin page to upload csv files, and routes to support upload of each type of data with validation (helper functions are in next page)

- algorithms: ‘record records from a csv file and store in a database table’, ‘checking the csv data before loading it into the database’, ‘admins must be able to upload transport and school info to the website from .csv file’

App.py

- create the login manager and start the app, create routes to support welcome page, login, logout and signup, & validation (helper functions are on the next page)

- algorithms: ‘use session variable to manage users login’, ‘record and validate user registration details’, ‘users must be able to register personal details’, ‘incorrect user registration will not be stored in the database’ ‘ensure site is being operated by authenticated user’, ‘register the following details (first name, last name, occ, name of real estate agency/business)’

****

**A screen shot of a computer program

Description automatically generatedA computer screen with text

Description automatically generated**

App.py continued

- helper functions to load data from the database, compare distances, create database connection

- code is self explanatory so have not provided line by line annotations

- geodesic is a python function ‘calculates the shortest path between 2 points on any surface’ in this case using lat long points on earth

App.py continued

- create remainder of the routes, browse, search, profile, datapolicy and forgot password routes (helper functions are to the right on this page)

- algorithms / req ‘search the database on specific criteria and display the results’, ‘user must be able to enter a keyword to find information about bus stops and schools relevant to the are they are interested in’, ‘users must be able to register personal details’, ‘incorrect user registration will not be stored in the database’, ‘appropriate validation checks on new data before it can be uploaded to the database (service side version here, for client side see templates and UserForm), ‘maintain the following details (first name, last name, occ, agency)’

1. **Web page templates and custom css**

base.html

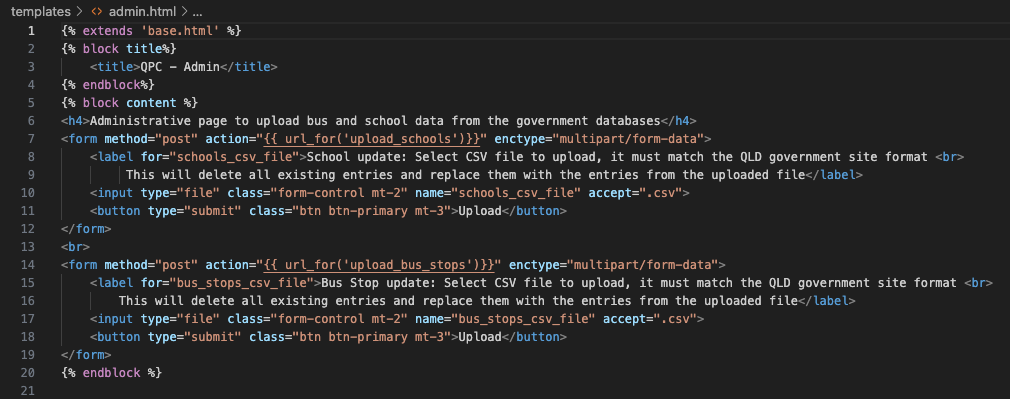
- all templates inherit from this base, includes link to bootstrap style, and free icons from fontawesome, creates navigation, common error message handling, footer and provides template block for title and content in all subpages.

- algorithms / req: client side check for ‘authenticated user’ and admin use in nav, bootstrap provides base for ‘a responsive and dynamic web interface’, and ‘Zooming of pages’. Template provides base for ‘page titles must be in browser for all tabs’. See each page for implementation. ‘Non-Mouse navigation is logical’ has been tested every page

login.html – extends base, collects email and pwd for login

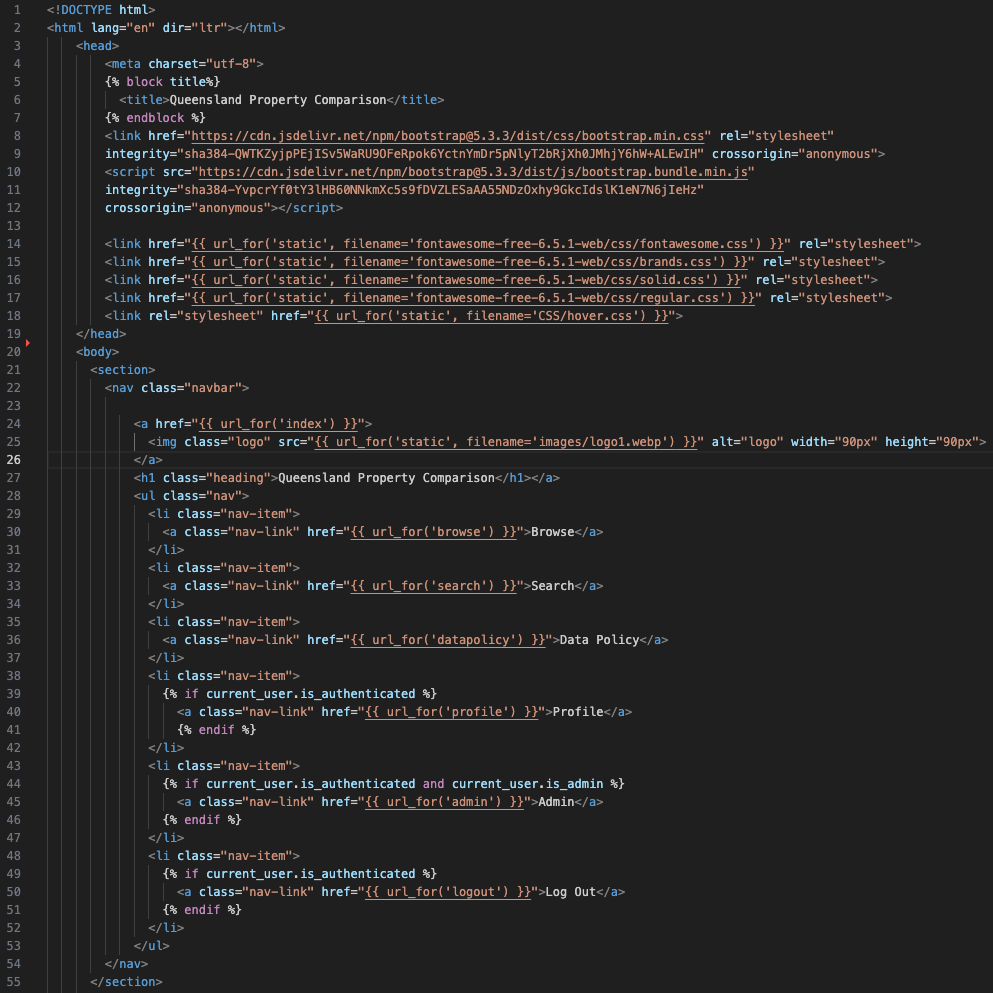
- accessibility; inc. title and appropriate heading, fields indicate required and use html types, have labels, bootstrap style for responsiveness and zooming

A screen shot of a computer program

Description automatically generated**A screen shot of a computer program

Description automatically generatedA screen shot of a computer program

Description automatically generatedA screen shot of a computer program

Description automatically generated**

admin.html – extends base, ‘admin tool that allows editing and updating the database

- inc. title, heading. Field has lable, format type csv. BS style for responsive/zooming

forgotpassword.html – extends base, collects email for pwd reset

- accessibility; inc. title and appropriate heading, fields indicate required and use html types, have labels, bootstrap style for responsiveness and zooming

browse.html – extends base, ‘allows exploration of properties’

- accessibility; inc. title and appropriate heading, bootstrap style for responsiveness and zooming, appropriate alt text for house

index.html – extend base, landing/welcome page

- inc. title and heading, uses bootstrap style for responsiveness and zooming

1. **Data base creation and test data python scripts including admin script**

1. **Importing Data Files**

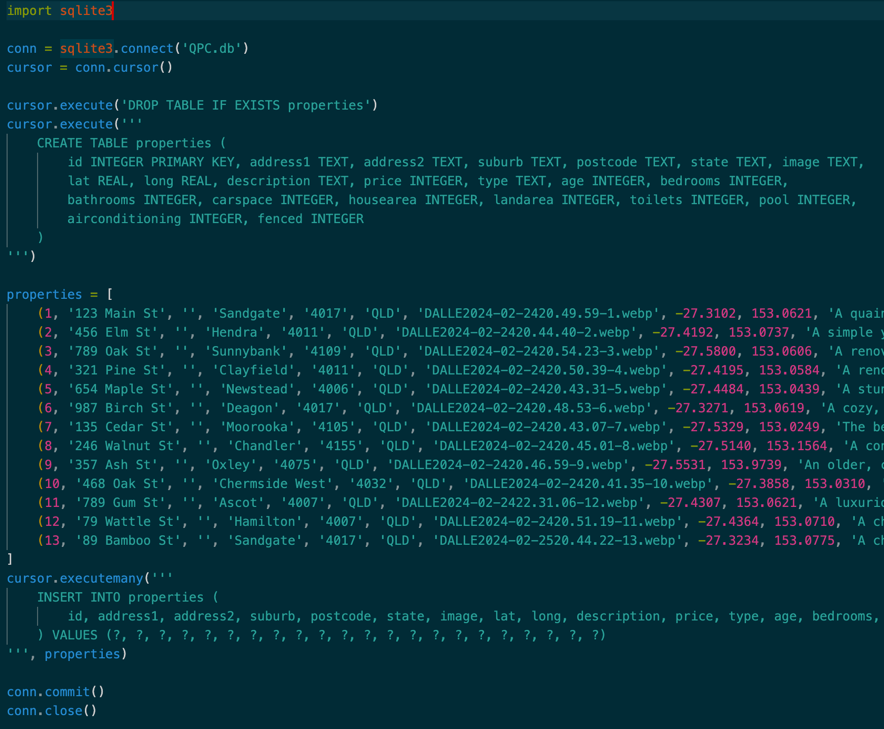
These files use python to create the tables in the QPC database, and add data when relevant, each file represents the creation of a new table. Only two files have been shown. Firstly importbus\_stops.py, which takes the data given in the technical proposal, creates a bus\_stop table, and inserts the data into there. The same logic is used with the importschools.py file, except it creates the schools table and inserts the schools data.

1. bus\_stops table: importbusstops.py. Code annotations are in grey within the screenshot for this file.

A blue text on a dark background

Description automatically generated

1. properties table: importproperties.py. The same logic is used to create the properties.

A small differences is that this file inserts data written within the file into the table, shown here. All the data did not fit within the screenshot, but it simply extends further to the left, inserting the required data.

All the data is fictional created by \_\_\_ and is not relevant to this document.

REST OF DATA

