**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, userform.py, userlogin.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

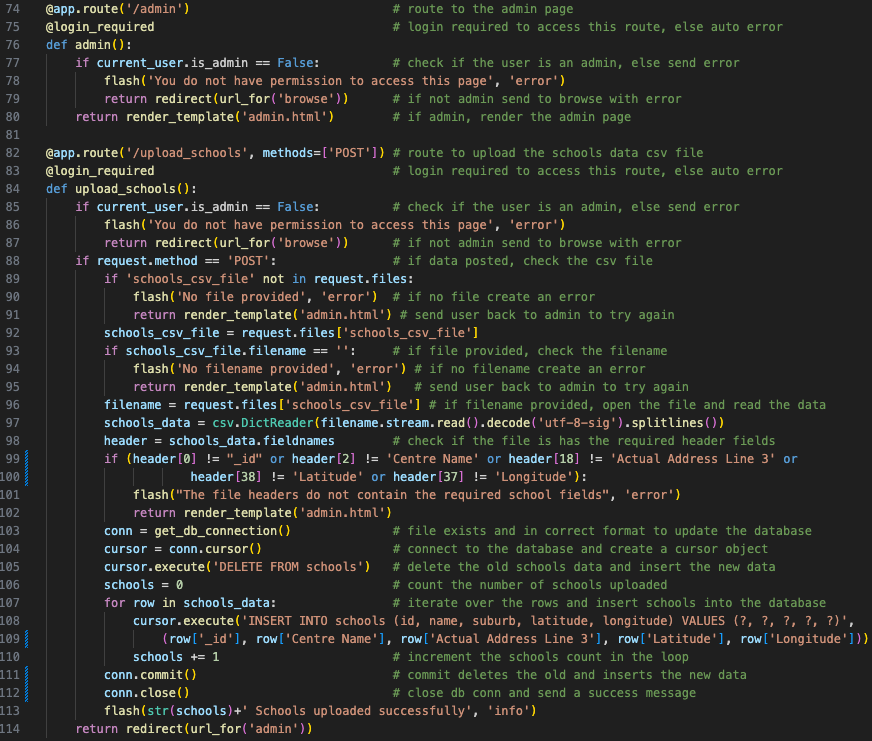
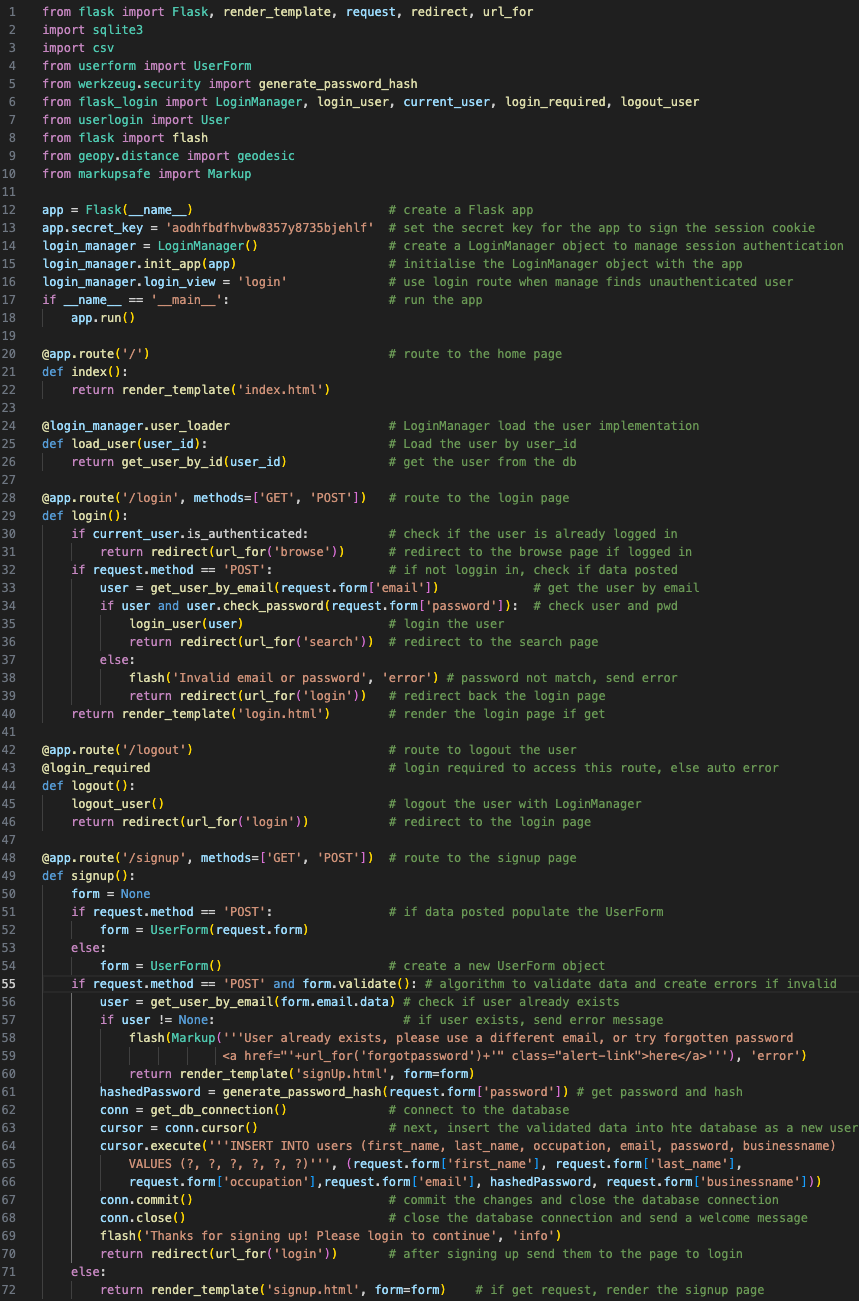
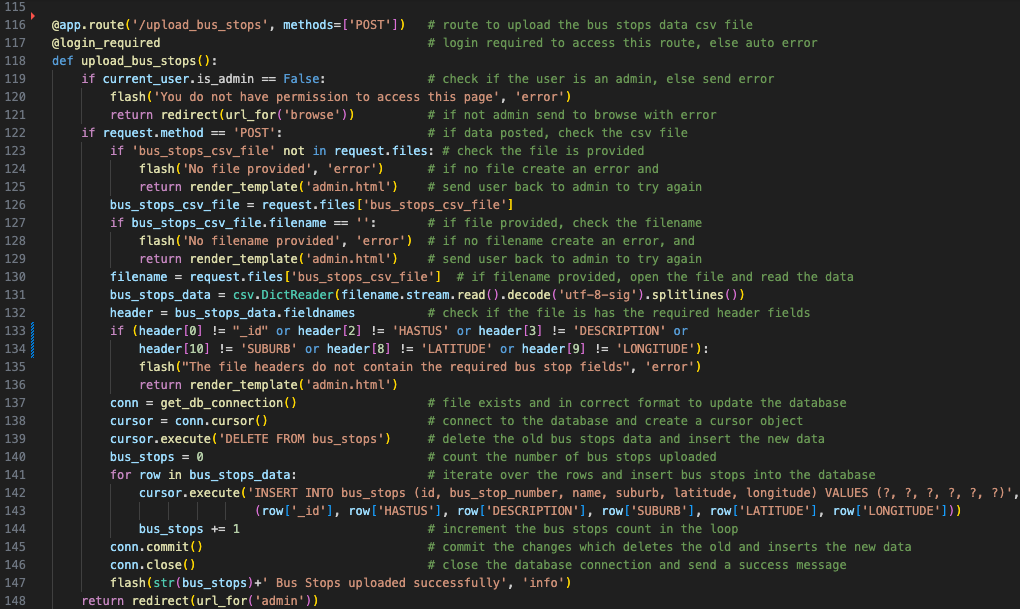
- create route for admin page to upload csv files, and routes to support upload of each type of data with validation (helper functions are in next page)

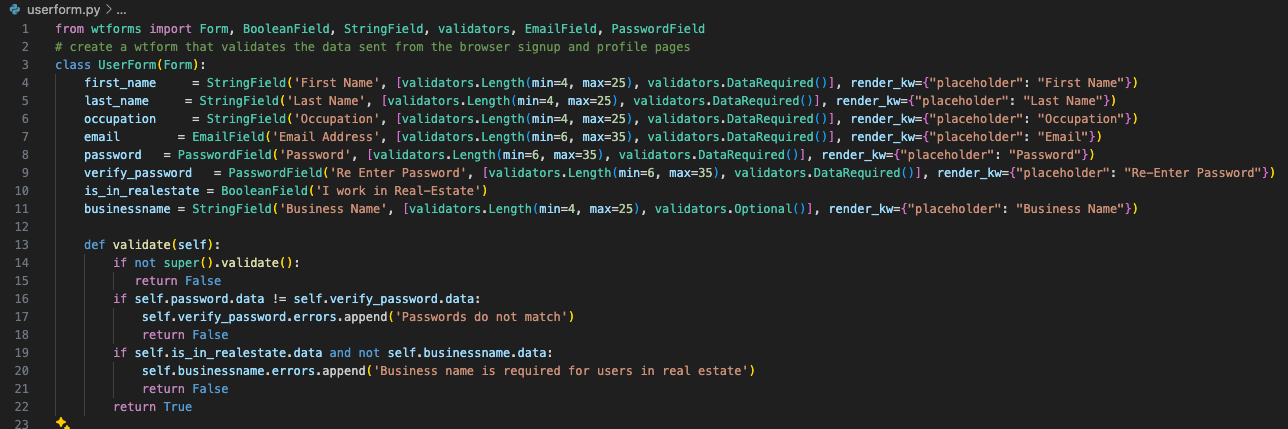
- algorithms: ‘record records from a csv file and store in a database table’, ‘checking the csv data before loading it into the database’, ‘admins must be able to upload transport and school info to the website from .csv file’

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**

userform.py – a wtform implementation that manages field level validation for the signup and profile html forms. Each field has a set of validators applied along with placeholder text that is used in the input field. The Class also manages custom validation for the pwd and if the real estate flag is triggered makes sure a business name is provided. If any field fail this class generates specific errors msgs

userlogin.py – login manager uses this class to do session based authentication

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

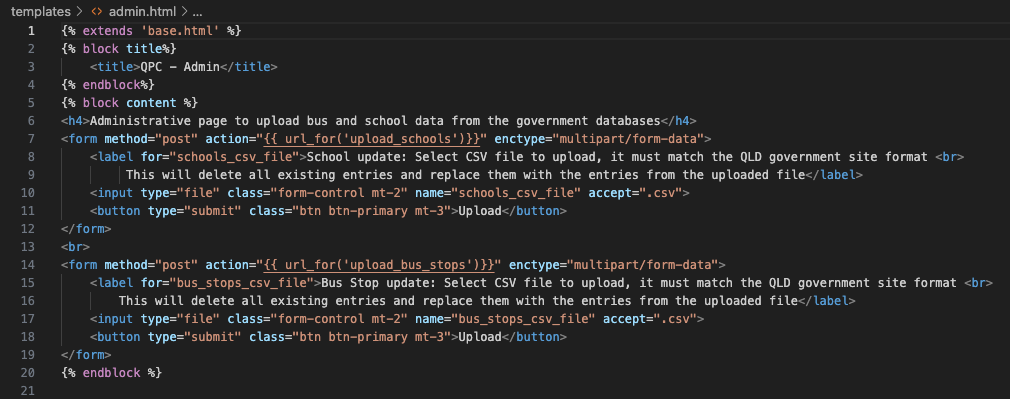
- all templates inherit from this base, includes link to bootstrap style, and free icons from fontawesome, creates navigation, common error message handling, footer and provides template block for title and content in all subpages.

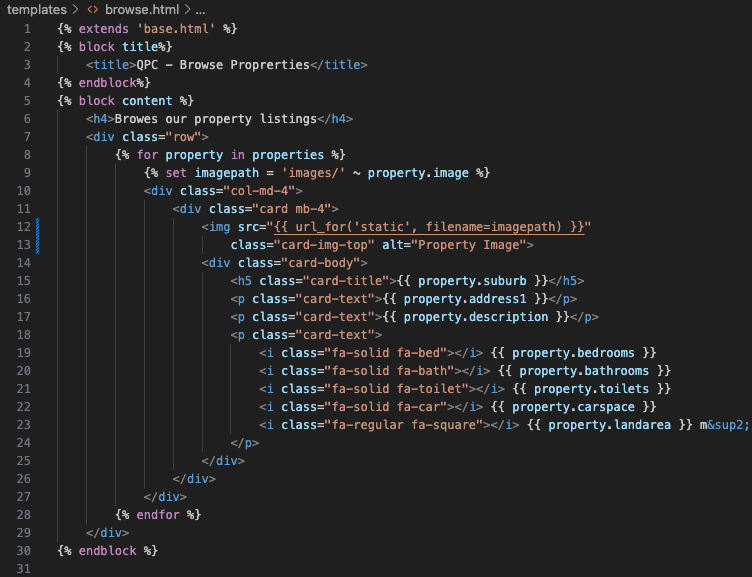
- algorithms / req: client side check for ‘authenticated user’ and admin use in nav, bootstrap provides base for ‘a responsive and dynamic web interface’, and ‘Zooming of pages’. Template provides base for ‘page titles must be in browser for all tabs’. See each page for implementation. ‘Non-Mouse navigation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

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

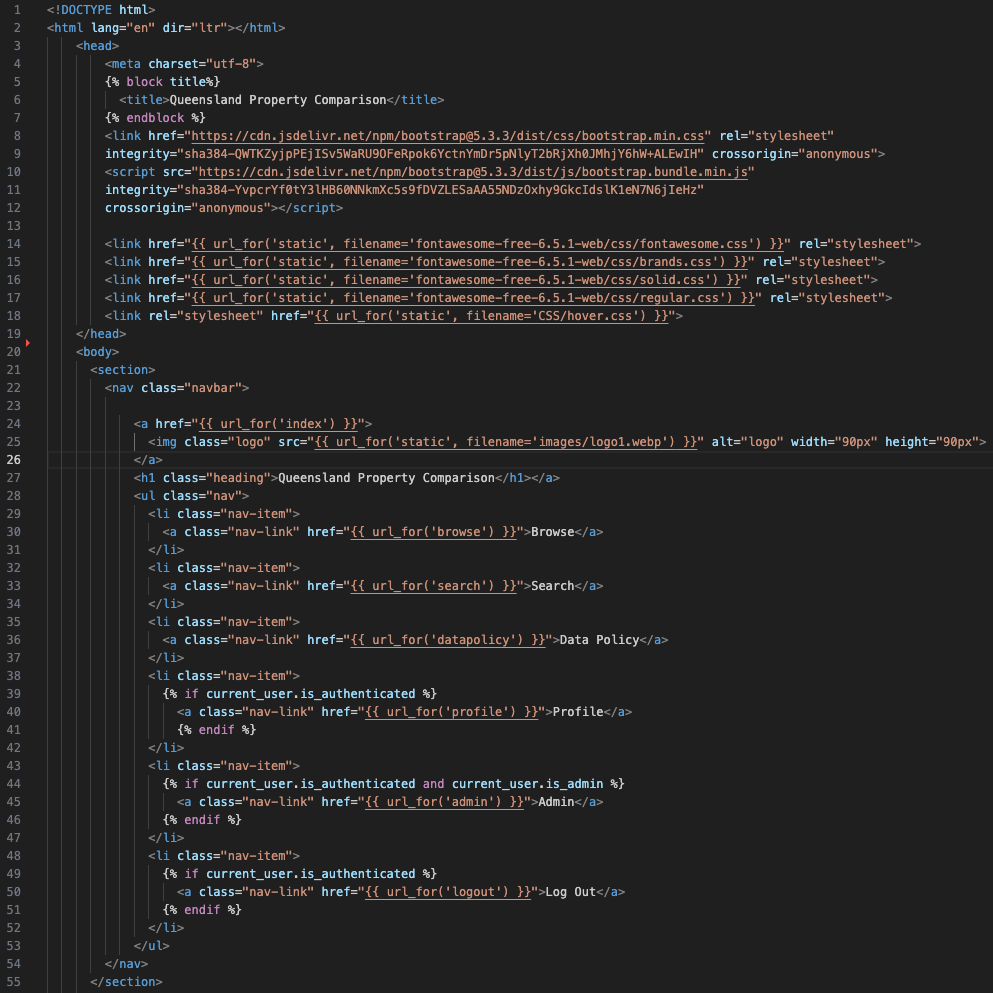
- accessibility; inc. title and appropriate heading, fields indicate required and use html types, have labels, bootstrap style for responsiveness and zooming

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

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

A screenshot of a computer program

Description automatically generated****

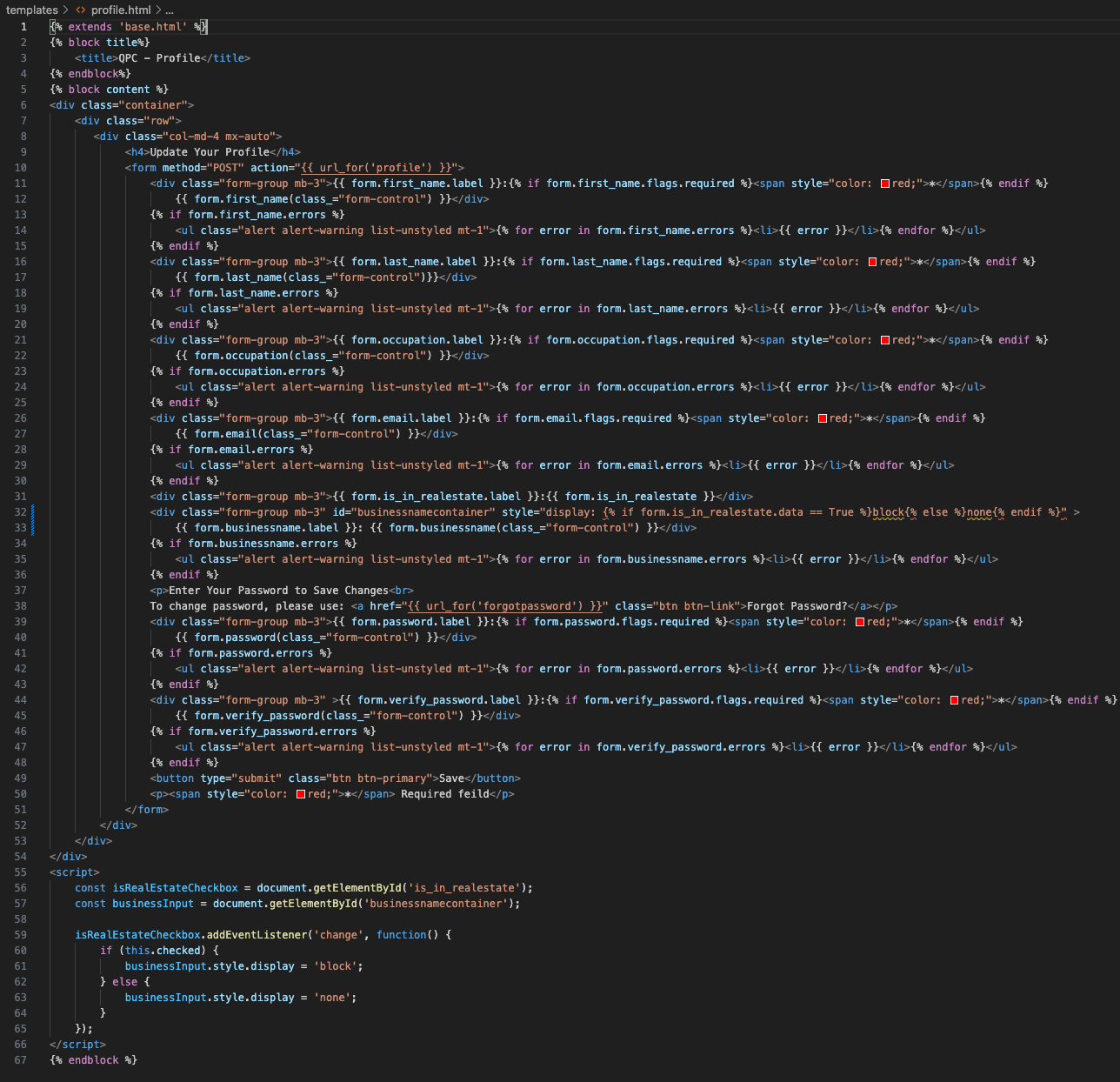
datapolicy.html – extends base, ‘contains an information page that provides appropriate explanations of data, the source of data and how the data is used’, ‘information page with, definition of terms, source of information and how it is used, explanations that help users understand the data’

- accessibility, inc. title, heading, appropriate instructions.

REPLACE.

search.html – extend base, ‘allow exploration and comparison of the property market using school location and transportation services criteria data’, ‘allow users to search for properties based on their criteria (schools and transportation)’

- accessibility: inc. title, heading, uses BS style for responsiveness and zooming. Fields have labels, formats indicated and tabbing order logical. Uses alt text for images appropriately, appropriate instructions at top of the form, data is not lost on submission, appropriate contrast ratio.

****

signup.html – extend base.html

- algorithms / req: ‘Users must be able to register personal details with the site’, ‘following details, first name, last name, occupation, real estate business’. ‘appropriate validation checks on new data before it can be uploaded to the database’ via client side validation mapped to the UserForm.

- accessibility. Inc. title, heading, use Bootstrap style for responsiveness and zooming. Non mouse tabbing is logical order, and access to all page elements. Fields have labels, mandatory fields have red \* and html 5 required indicators. Errors are clear and specific to each field, and do not reset the form.

- annotation: the page uses the jinja template language to load a wtfrom UserForm. (see the userform.py code in section 1 on app code.) If any errors are found they are displayed next to the field. Also the data submitted is returned and pre-populated so that the form is not reset.

The bottom javascript will show/hide the business name input field when the checkbox is triggered

profile.html – extend base.html

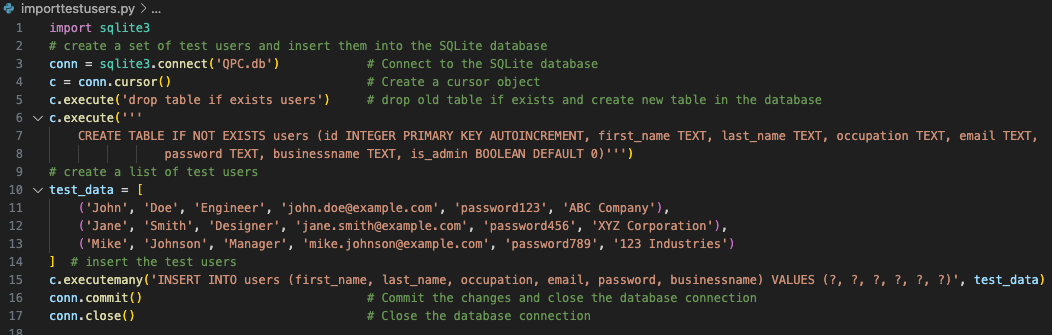
- algorithms / req: ‘must be able to register users and maintain the following details, first name, last name, occupation, real estate business’. ‘appropriate validation checks on new data before it can be uploaded to the database’ via client side validation mapped to the UserForm.

- accessibility. Inc. title, heading, use Bootstrap style for responsiveness and zooming. Non mouse tabbing is logical order, and access to all page elements. Fields have labels, mandatory fields have red \* and html 5 required indicators. Errors are clear and specific to each field, and do not reset the form.

- annotation: this page is similar to the signup, except the layout of the pwd field is separate. Following the common security pattern that ensure second factor authentication for pwd change (email), the user may update all their data except pwd. To update pwd they must use reset pwd.

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

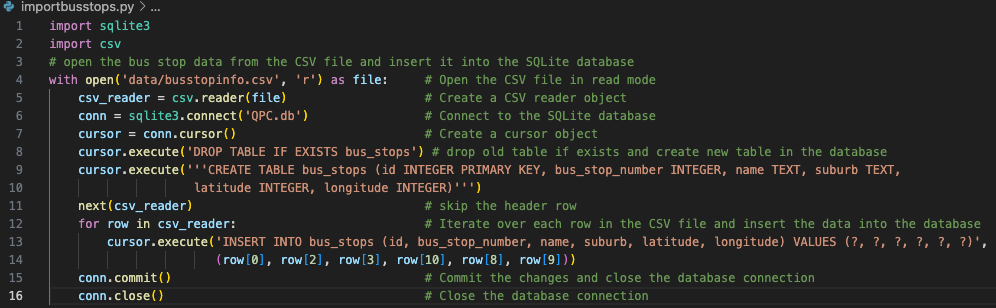
Below are the python scripts to create the data base schema. Each script also includes a set of test data, users, properties and in the case of the bus stops and schools, the initial load of the csv before the admin function was created. Finally there is a script to change an existing user to be an admin so they can access the admin page to upload new bus stop and school data.



importtestusers.py

- req: ‘users must be able to register personal details within the site’

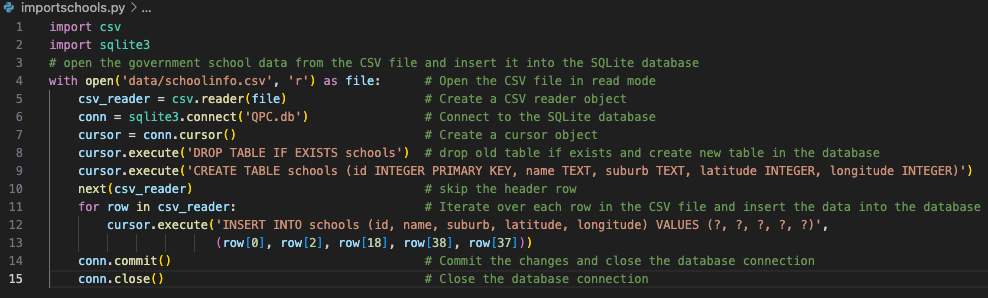
- annotation: create a simple user table that matches the requirements. Extend to add is\_admin and set to false by default. See makeadmin.py below on how to update admin. Create a number of test users, these are not be used in production. New users created by the app have pwd hashed, see signup route, userlogin.



Importbusstops.py

- req: ‘read records from csv file and store them in db table’, ‘search properties based on school and transportation’

- annotation: create table for bus stops but store only data required for site not all bus stop data. Also includes first upload of data for fast testing. However the admin routes now provide web functionality to do this.



Importschools.py

- req: ‘read records from csv file and store them in db table’, ‘search properties based on school and transportation’

- annotation: create table for schools but store only data required for site not all school data. Also includes first upload of data for fast testing. However the admin routes now provide web functionality to do this.

madeadmin.py

- req: ‘provide admin tool that allows edition and updating of the database set’

- annotation: simple script to make a user admin so they can then see admin routes to do the admin updates of data sets

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

Importproperties.py

- req: ‘ exploration and comparison of property market…’

- annotation: create a set of test properties. Include real lat long based on the suburb centre so that the school and bus stop distance checks work. Include a number of real estate property data to make tile view effective