Technical Documentation

Programming Pioneers: Easy Plots

Vincent Mays
Tai Britt
Mark Bird
Jason Cook

Team Website: https://sites.google.com/mail.fhsu.edu/programmingpioneers/home

CSCI441
Submitted: 05/05/2024

General:

- Things to install:
    - Nothing! We have launched the application live on PythonAnywhere.com:
        - https://easyplots.pythonanywhere.com/login
    - Or, to run locally:
        - pip (if you haven't already)
        - flask: pip install flask
            - framework to build python website, simpler than Django
        - flask-login: pip install flask-login
            - package to help make handling user login easier
        - flask-sqlalchemy: pip install flask-sqlalchemy
            - wrapper for SQL to make it easier to work with databases
        - You will be prompted by error messages to install any other packages needed to run the program locally. This will depend on your local python environment.
- Folder hierarchy:
    - EasyPlots (folder):
        - static (folder): #used to hold things that don't change for our web app like javaScript files and css files
            - index.js (file)
            - pyPlot.png (file)
        - templates (folder): # this folder needs to be named 'templates', for and holds html pages
            - base.html (file)
            - home.html (fiile)
            - login.html (file)
            - plotTemplate.html (file)
            - sign_up.html (file)
            - upload.html (file)
        - __init__.py (file)
        - auth.py (file)
        - models.py (file)
        - pyPlot.py (file)
        - views.py (file)
    - instance (folder):
        - database.db (file)
    - main.py (file)
- You can use the application directly here: https://easyplots.pythonanywhere.com/login
    - Or, ff running locally, to check how the app is working, you have to go to the link the server provides. This link will show up in the terminal when you run the mian.py file. For me it was http://127.0.0.1:5000.

Overview of app:

- This project implements a web application using python and flask web app framework.
- The webpages are built using a template system. The base.html files serve as the template for each webpage and this template has certain ‘blocks’ that are overwritten by code in the other .html files (home.html, login.html, sign_up.html).
- These webpages (html files) are called on (rendered) using the views.py and auth.py files. Within these files is the code that routes to the respective .html files or redirects to another route. This allows us to run functions in python that receive variables from user inputs (POST methods) and update the page before rendering it to the user. So the view.py and auth.py files allow us to both set routes to coordinate navigation of our website and get user entered data to run in python functions, then update the webpage based on this data before displaying the page to the user.
- Information for the user is stored in a database using sqlalchemy in python.

Breakdown of folders and files:

__init__.py:
- makes the EasyPlots folder a “python package”
- whatever is in the __init__.py file will run automatically when we import the EasyPlots folder.
- this is where we make the function that we will run in main.py to initialize the application

main.py:
- This is where the app will actually run from
- in order to run this file though, you will need to make sure you’re running a python interpreter through VS code (or however you handle this in your IDE

models.py:
- where database tables are built

pyPlot.py:
- python file where plots are built

views.py:
- stores endpoint urls for frontend of website
    - These will be the webpages the user actually goes to and views (login is the only exception because it’s part of authentication, so it’s url will go in the auth.py file)
    - This views file works with the .html files in the templates folder to display the webpages to the user.
        - Think of the views.py files as the navigation hub or “Blueprint” for navigating the website and the .html files in the templates folder as the code that will generate the web page to display (the .html code).

auth.py:

- “Blueprint” file like views.py to allow us to navigate the web app.
- The difference is simply that auth.py is only used for web pages that involve authenticating the user (login. logout, sign_up)

base.html:

- When implementing web pages with the ‘templates strategy’, we typically have a ‘base’ html file that then is ‘filled in’ with our code for each web page found within their own respective html files.
    - Think of the base.html as the template which is filled in with code from the other html files.
    - This makes the website easily scalable
    - Think of base.html as the theme of your website. Some parts are overwritten by the specific code in your other html files.

home.html:

- contains html needed to overwrite corresponding block in base.html and implement the home page

login.html:

- contains html needed to overwrite corresponding block in base.html and implement the login page

sign_up.html:

- contains html needed to overwrite corresponding block in base.html and implement the sign_up page

plotTemplate.html

- webpage for user to select dataset and delete it if desired. This page is navigated to automatically after uploading a csv.

upload.html

- webpage for user to upload a .csv file to be stored on the database and used to build plots.

index.js:

- Used to store any needed JavaScript for the html.

pyPlot.png:

- Image file that is created by pyPlot.py when created plots.