**Flask + Peewee + Matplotlib = Forum1040**

**Contents:**

1. The objectives of the project
2. Modules used for the project
3. The structure of the project
4. Instructions on using the project
5. Future development and reflection
6. **Project overview**

Originally, since all of the data processing and visualization we did were based on given data, we thought it would be cool and useful if we could work with real time data, i.e. databases, to generate graphs based on what is inside the databases as a more intuitive way to monitor the performance of our applications and systems associated with the databases. Connecting to databases and getting data using Python for future processing turned out to be much easier than we thought. So we took a step further, building our own application, a forum, in Python.

While drifting on the ocean of code full of high waves and despair without a direction, Stack Overflow has been the lighthouse time and time again to guide us safe and sound to the shore of happiness. That’s where we realize the importance of peer communication and community learning. Inspired by that, we decided to build a forum system that, while enabling the communication between peer students, gives teachers and tutors more intuitive ways to monitor the posts made by students. In other words, we built a forum that integrated the data visualization and text processing as a part of its system.

Our forum application provides the following functionalities to its two groups of users:

For students:

* Login / change password
* View notices that are posted by the admin users of the forum
* Make posts
* View posts and comments
* Reply other posts

For admin users (teachers, tutors):

* Login
* Create notices
* Delete posts/comments/notices by ID
* View statistics

1. **Modules used for the project**

There are many frameworks that make the job much easier for web development in Python. Django and Flask are perhaps the most famous two.

We chose Flask over Django for it is, as a minor yet powerful framework, easy to start with. Flask allows more freedom for designing the project layout. Traditionally, we have MVC pattern as the structure for a dynamic website, where MVC stands for model, view, and controller respectively. For Flask, however, the pattern used is MTV, where T stands for template, a Flask way to represent data, and V stands for view functions, which is the interface between the models and templates.

For connecting with databases, we used Peewee, a small but robust module based on the idea of ORM, object-relational mapping. Peewee treats a relation as a class, and creating tables, updating, transactions, and querying can all be done by calling methods to the relation class in quite an elegant way.

1. **The structure of the project**

All the files related to Forum1040 is stored inside the Forum1040 folder, which contains three subfolders, data, static, and templates. Files like icons, CSS files, JavaScript files, fonts, and images are stored in static. Templates folder has html files that are used by the render\_template function in flask module together with other parameters passed in to generate views dynamically. Finally, data directory contains files that work with database.

1. app.py: the controller of the project, handles input generated by processingDB.py and forms submitted by user from various sites to dynamically alter the content of each html file.
2. add\_admin.py: by running this file, user with access to this file will be prompted to input the name and password for creating a new admin user one at a time.
3. register.py: opens a text file named id.txt and assigns each line (ID) a default password
4. data directory:
   1. configuration.py: this file is used to get the directory path of 1040.db file so that files from different locations can connect to the same database file
   2. models.py: using Peewee to define relations, and handles the connection with database
   3. processingDB.py: queries real time data mainly using Peewee from database, and uses Matplotlib to handle the visualization of data
   4. register\_demo.py: creates a text file which has usernames for demonstration purpose from u1000 to u1099
   5. post\_engine.py: processes two files, fake\_comment.txt and fake\_post.txt and insert them into the comment and post relation respectively as the demo comments and posts.
5. templates directory:
   1. base.html: one of the interesting points of using flask; since all of the websites share the same overall style and only vary in terms of content, base.html takes care of the overall style. Other html pages “extends” base.html to deal with content based on the style specified in base.html
   2. index.html: the home page of the forum which displays all the notices as well
   3. forum.html: aka the hardest part; deals with displaying all the posts and their associative comments
   4. post.html: where students make posts
   5. login.html: takes a user’s username and password and deals with the validation by passing the value to login function inside app.py
   6. change\_password.html: where a user changes his or her password
   7. admin.html: the home page of admin, displaying all the statistics related to the contents in the forum database
   8. query.html: aka the interesting part; admin users can visit this page to delete inappropriate posts, comments, or outdated notices via IDs. Moreover, admin users can write SQL queries and get the corresponding result.
   9. notice.html: admin users can add notices that are going to appear on the home page of the students’ side.
6. **Instructions on using the project**

Make sure the environment in which the project is ran has Python version 3.5, matplotlib, flask, peewee, sqlite3, and radar installed.

To register users:

* Register students: input the students’ student IDs in the id.txt and run register.py afterwards. This will register students in the forum database by assigning each id a default password that is the same as the ID. Students can login using the initial password and then change the default password later.
* Register admin users: run the add\_admin.py file and operate accordingly to add admin user one at a time.

After completing the previous steps, run the app.py file and visit the <http://localhost:5000> to view the page. To visit the admin pages, visit <http://localhost:5000/admin>.

1. **Future development and reflection**

Web developing in Flask and Peewee is really fun! Even though the forum that we built is still pretty far from a real product, but it laid a solid foundation for understanding the techniques in web development.

Password hashing and authentication are next steps to take. After done with that, we can show the product to teachers and students for opinion. Based on the feedback, adjust the project accordingly. Deploying the web app to a server is the next but not final thing to do.

One of the biggest takeaway from this project is to learn new things by doing. We started off by building a blog so simple that it was written in less than 100 lines of code. The trade-off here was that we ended up with tons of revising work to do. For our next web app, there will definitely be more planning, using Data Flow Diagram for example, beforehand.