Team 18
Varun Ganesh vganesh7@vt.edu

ECE4564, Assignment 3

Objective:

The objective of this assignment was to learn about the RESTful API. The RESTful API, or REST, is the Representative State Transfer. Another focus of this assignment was the focus on using the different frameworks and Python libraries to implement the API. These frameworks were Flask, the Python requests library, and HTTP Basic Authentication.

This assignment required us to start the program requiring a port number. This port number established the port where my API would be listening for any HTTP requests being sent. Once started, HTTP requests could be sent through the web browser, or cURL in the terminal. These requests had the same general format of "http://x.x.x.x:port/" This is the general page. When any of the routes are accessed, a username and password is required to enter. Upon entering a bad username and/or password, the prompt clears and prompts again until a valid username and password is provided. There are two other routes established in this API, one accesses Marvel and retrieves stories, the other accesses Canvas and downloads the requested file.

Design:

In this project, I had to use Flask to create a web page for my API. Flask allowed my program to establish a server listening for HTTP requests. I also had to use Flask-HTTPAuth to authenticate the user attempting to send any requests. Finally, the Python requests library was used to send the HTTP requests to external APIs to gather information. Design decisions lay around how each route was implemented and how authentication was done. Authentication was completed by taking the provided username and checking if it was present in the users list that was preinitialized with two valid users. If the username provided exists in the list, the passwords are then compared. If the passwords matched the user is granted access, but if the passwords do not match the user is prompted again to enter a username and password.

One logged in, the user can access the main route, which will just display the message "Hello x!", with the "x" being the username. The other routes accessible are the Marvel route and Canvas route. The Marvel route takes the form of "http://x.x.x.x:port/Marvel?story=yy". This route accesses the Marvel endpoint and provides a string query of value "yy" to the route. The function called upon access to this route will take that value, generate a hash key and timestamp to access the Marvel API and issue a GET request to collect the data regarding that story. That data is stored in a JSON format in a text file in the same directory as the program script. The Canvas route is very similar to Marvel. It will take the form "http://x.x.x.x:port/Canvas?file=yy". The "yy" is the name of the file. The function called upon access to this route will issue a GET request to the Canvas API to collect the data about the file. It will then deserialize the JSON Response and parse it for the download URL. Once obtained, it will issue a second GET request using the download URL and copy the contents into a newly created file using the file name in the same directory as the program script.

Outcome:

The outcome of this project was that the code would continuously run and allow for multiple GET requests to be issued through various methods. Given an appropriate port value, the Flask web page would appear when accessed through the web browser or cURL. Authentication will be required upon first access of any of the API routes. This project gave me a better understanding of building and using a RESTful API through Flask, the Python requests library, and basic HTTP Authentication.