Omkar Savkur

* 1. Term Project Proposal

**Project Description**:

My project is called CMU Foodie. This application will recommend where the user should eat on CMU’s campus and on Craig St based on a search query and the specials and menus of the restaurants. It will also keep track of the user’s past searches and rating/reviews of the locations they went to in order to suggest where to eat next. The recommendations should consider how far the restaurants are from the user in terms of absolute distance and walking distance.

**Competitive Analysis**:

The main idea of this project is to help the users find where to eat on and near CMU’s campus. With use, a user will get more and more personalized suggestions for where to eat on campus. Similar products include Yelp, Zagat, and LocalEats. Yelp allows the user to see reviews and ratings from other customers. However, it is not very good at rating where CMU students eat, as most college students do not write Yelp reviews often. With Zagat, there are only a few cities for which there are the reviews are given by professional food critics, who may not be local to the city. LocalEats provide the reviews of the best restaurants in a city and know which local restaurants are the best. They want to find the best unique restaurants that are specific to a city.

My project will be customizable to the user’s likes and dislikes. The opinion of professional and local critics can be valuable, but the most important information that dictates where you go to eat is your own personal preferences. Every person has their own favorite restaurants, so to be competitive in a market like this, my project should be tailored to each individual user to provide them with the best possible place to eat.

**Structural Plan**:

\_\_init\_.py file: Runs the user interface (Modal App object), rating form will be another mode, and handles the login, searching and recommendation functions

restaurant.py file: Has the Restaurant class that goes and scrapes the website html for the name, location, hours, description, menu, and specials; Also draw function to draw the restaurants on the canvas

userData.py file: Reads in user information from users.xml and returns a User object with all the data from the xml file

users.xml file: where all the data from the users will be stored as xml – will be parsed using Beautiful Soup with a similar parser that is used to parse the html

**Algorithmic Plan**:

Searching:

* Go through each restaurant’s description, menu, and specials
* Create a dictionary mapping each word in the text to its frequency
  + If the query is multiple words, create frequencies of word combinations up to the number of words in the query
* Webscrape thesaurus.com to find synonyms for each word in the query
* Convert the count of occurrence of synonyms/query into percentage
* Find the distance from the user’s current position and each of the restaurants
* Compute the weighted average of percent match and distance for each restaurant
* Sort the restaurants according to the weighted average

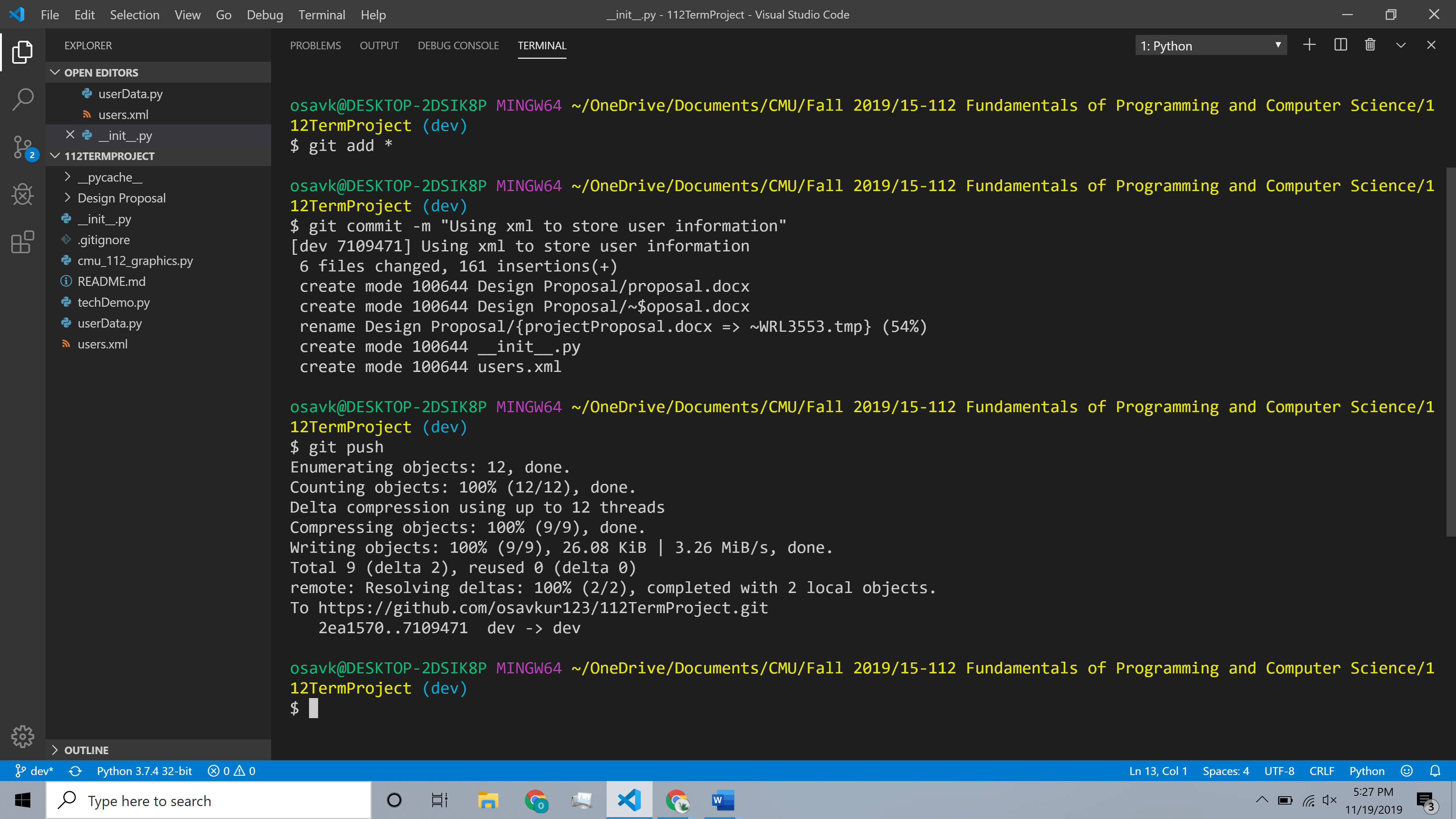
Recommendation:

* Create a bag of words for each restaurant
* For each review in the user profile
  + A positive review:
    - For each keyword (word that occurs more than once or twice), run the searching algorithm on all the rest of the restaurants
  + A negative review:
    - For each keyword (word that occurs more than once or twice), run the searching algorithm on all the rest of the restaurants
* Rank the restaurants that are most similar to the positive review and the least similar to the negative reviews

**Timeline Plan**:

|  |  |
| --- | --- |
| Task | Deadline |
| Webscrape CMU Restaurant Info | 11/16 |
| Basic UI | 11/18 |
| Read and Update User Info | 11/20 |
| Implement Rating Form | 11/20 |
| New user button and password hashing | 11/21 |
| Add more Restaurants from Craig St | 11/22 |
| Searching Algorithm | 11/24 |
| Recommendation Algorithm | 11/26 |
| Have an interactive search bar | 11/28 |
| More robust Recommendation Algorithm | 11/30 |
| Improved UI | 12/2 |
| Expand Restaurants to Pittsburgh | 12/4 |

**Version Control Plan**: I am planning to use GitHub for version control and backing up my code. Repository: <https://github.com/osavkur123/112TermProject> The master branch is for code that’s working (right now, it contains the tech demo). The dev branch is where I am working on code.



**Module List**: cmu\_112\_graphics library, tkinter, PIL, requests, bs4, textblob, geopy

**TP2 Update:**

I have switched my recommendation algorithm to use K-Nearest Neighbors to find users that like similar restaurants. Based on the restaurants that the similar users like, I recommend the highest rated ones to the user. I am also caching the CMU restaurant website and the Yelp restaurant website in case the response gets blocked. To calculate the walking distance between the user and the restaurants, I am using Bing Maps REST Services API. The distance is displayed for the user once they sort by walking distance. I used a hash algorithm based on Daniel Bernstein’s hash function (<https://en.wikipedia.org/wiki/Universal_hashing#Hashing_strings>) to hash passwords to store them for later.

**TP3 Update:**

I implemented Sentiment Analysis to the recommendation system, so the distance between users is now based on how positive/negative they rated as well as their comments on a restaurant. When you get the recommendations, I added a percent match for each restaurant, which is calculated by finding the average percent difference between the neighbors and the max distance and weighting that with the z-score of the rating the neighbors gave the restaurant. I added a sorting by distance feature to the search results as well. Based of initial user testing, I added reviews from other users when you click on the restaurant cards. When you click on a restaurant card, you see the image that is from CMU’s dining website or Yelp’s website as the background. From user testing, I was told to make the homescreen easier to read (I had an image of different foods), so I used a blue-green gradient as the background instead.