Project Proposal

Project Description:

My project is called Restaurant Match. It is a tinder style app where people match with restaurants in their area that they have not tried yet. The app only recommends places that the user has not matched with/ eaten at yet. Users will have the ability to save their favorite restaurants and view the favorite restaurants of any users they are friends with on the app.

Competitive Analysis:

While researching features to implement into my app I found a few other apps online that function very similarly to how mine does. Most of them had the same design but unlike my app they were all mobile apps. Each of them had a swipe feature, filters for price, range, and type of food, and a 'more details' button for each restaurant. These are all designs that are already or will be implemented into my apps. A design decision that I made that I did not see on other apps was that I plan to incorporate graphs into my app that will display a restaurant's rating vs. its price per person. The purpose of this is to give the user more information for them to determine if they can get a better 'bang for their buck'. If I have the time, I also hope to recommend restaurants based on the data in the graph.

Structural Plan:

Currently my app is using the modal app structure shown in the animation optional lecture. Initially there is a class called Login Page that contains a list of functions related to creating or logging in to the users account. Once the user logs in they are directed to the Main Page class. This class holds all the functions that display the restaurant information obtained from the Yelp API such as restaurant profiles and user friends. There will be a Favorites class that stores the functions needed to display a user's favorite restaurants. Lastly there is the match class. This class simply displays all the information for the restaurant the user 'matched' with. In addition to these modes there are helper classes that store information about one feature. There is the button class that stores all the functions related to making a button and calling an instance of the button class in one of the modal classes can create a button. Another helper class is the Yelp class. The yelp class contains all the functions that parse the data from the Yelp API into data that can be displayed in the Main Page class. Lastly there is the text class. The text class stores

all the functions related to creating text boxes for user input. Calling an instance of the text class creates a text box that displays and stores user input.

Algorithmic Plan:

Currently the most challenging aspects of my project have been getting and parsing the data from my API, creating and updating user data within my csv files, and making text boxes for user input. My first challenge was the Yelp API. For getting the data, I first do a Yelp business search and extract all the restaurant IDs and store them in a list in the Main Page class. I then loop through the list and take in the ID as a parameter within the Yelp API class and do a Yelp ID search on each restaurant. Once the data for the restaurant has been parsed using static helper functions, I store each restaurant as an object in the Yelp class. When I need to display the restaurant data, I make an instance of the yelp class within the main page and loop through the list of restaurant objects stored in the Yelp Class. My next challenge was updating csv files. Unfortunately, within the python csv module when writing in a file you either have an 'a' permission or a 'w' permission. The former can only append rows to a csv file while the latter rewrites the entire file. In order to update a csv file at a specific row I had to rewrite the whole file. To do this, I wrote the Profile class. This class reads the csv file and separates the data into current user data and other user data. Since the current user's data is the only one that changes, each time a new friend or new favorite restaurant is added the profile class updates the user's data and rewrites the csv file. My final challenge has been creating my own user input in the form of text boxes. To accomplish this, I created the text class. This class has 3 key features that allow it to work. These are drawWord, self.word, and self.letters. Each time a letter is pressed that key is added into the list of self.letters and displayed to the user by drawWord that joins the letters in the list and displays them. Pressing the enter key saves the input to self.word in string format. Calling an instance of the text class creates a text box anywhere on the canvas. Lastly, I plan on incorporating a graph made within tkinter, but I have not considered how this will be done.

Timeline Plan:

Saturday April 18 – Display features for favorites and friends will be finished

Sunday April 19 – User input for range, price, and term will be finished

Wednesday April 22 – Tkinter graph will be completed

Thursday April 23 – Monday April 27 – UI graphics will be polished

Monday April 27 – Wednesday April 29 -TP3 ReadMe and Project Demo will be completed

Module List:

For my project I will be using the Yelp API and the python CSV module.

Version Control:

Each time a new feature is being implemented or integrated into my main code, I create a new python file in order to have a point to go back to in case things go wrong when implementing a new feature. The folder containing all of my files for the term project is regularly uploaded to my google drive.

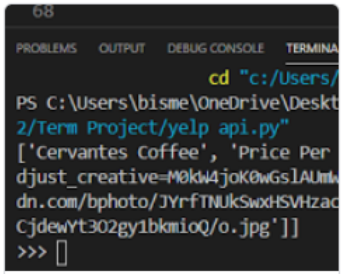
My Drive > Term Project

Folders

Name

__pycache__

Files



api demo 1.png

api Key.py

cmu_112_graphics.py

logic tests.py

Profile class(friends,favs a...

Project Proposal.docx

read and write csv.py

requests-2.23.0-py2.py3-no...

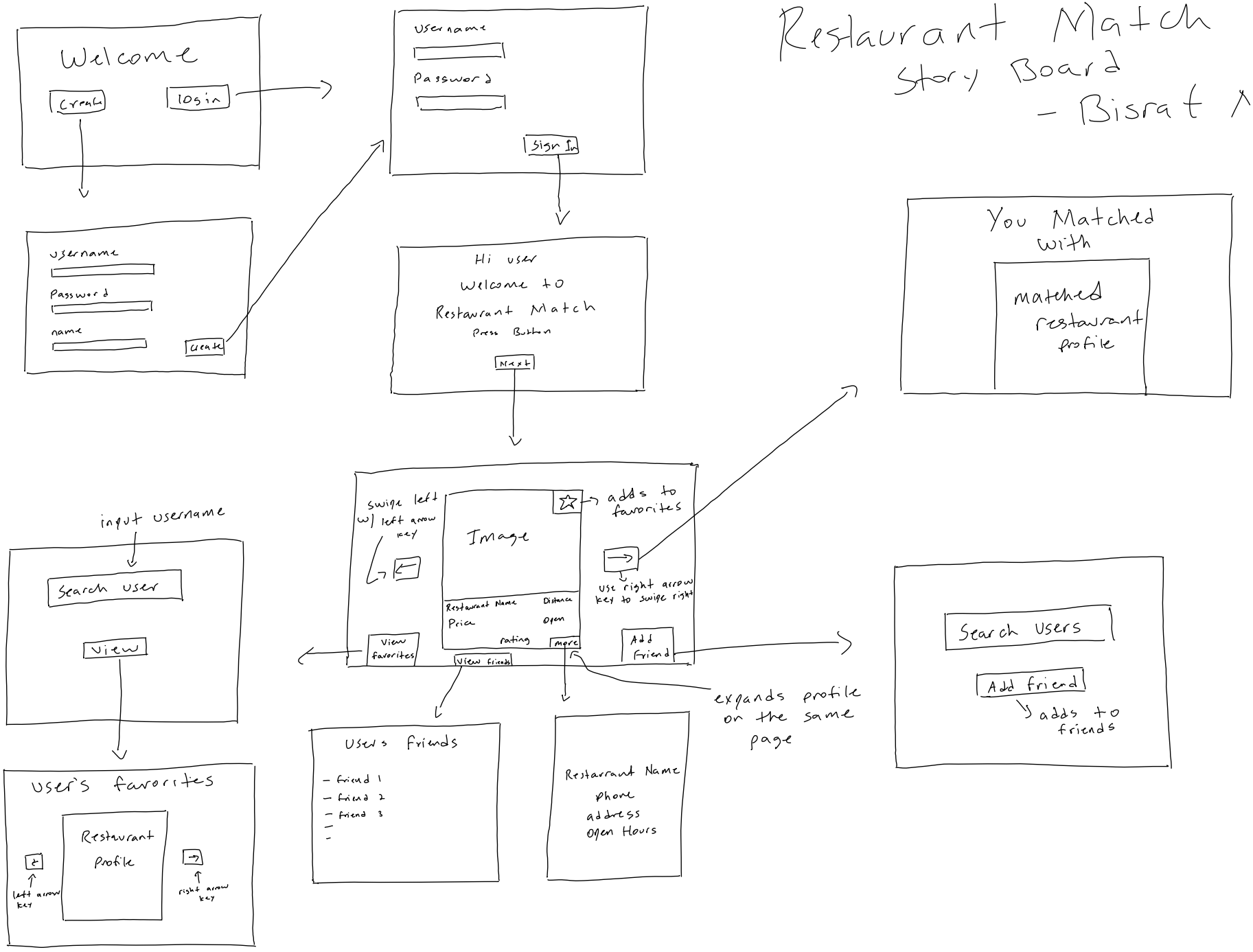
test1.csv

textbox.py

TP V1.py

TP V2.py

Restaurant Match
Story Board
- Bisrat Mekonnen



TP2 Update

Structural Plan:

I have added an additional Graph class that contains all of the functions for the graph that is drawn fully within tkinter.

Algorithmic Plan:

An additional challenge to my project has been making a graph fully within tkinter. To accomplish this I created a list of tuples containing all of the points that needed to be plotted to make the graph and sorted them by their x values. I then used the x and y values of where the graph is drawn to model where the points are supposed to lie on the screen. After that, I looped through the list of points and called the create line to connect the dots.

TP3 Update

Algorithmic Plan:

I have added an interactive element to the tkinter graph, where clicking on any point redirects the user to the restaurant profile that corresponds to that point. By making each point a button and having the button redirect to the appropriate restaurant index in the list of restaurant objects. Additionally I also created a visited list row within the csv file so that a restaurant that the user has matched with isn't recommended again.