CS 5200 Database Management Systems – Final Project

Title: New England Ice Cream Network
Group Name: ArmandZ
Members: Zachary Armand

README

Required python packages:

- pymysql
- sys

Included files in zip package:

- ArmandZ.py
- ArmandZ.sql
- edit_functions.py
- home_menu.py
- search_functions.py
- view_functions.py

Instructions:

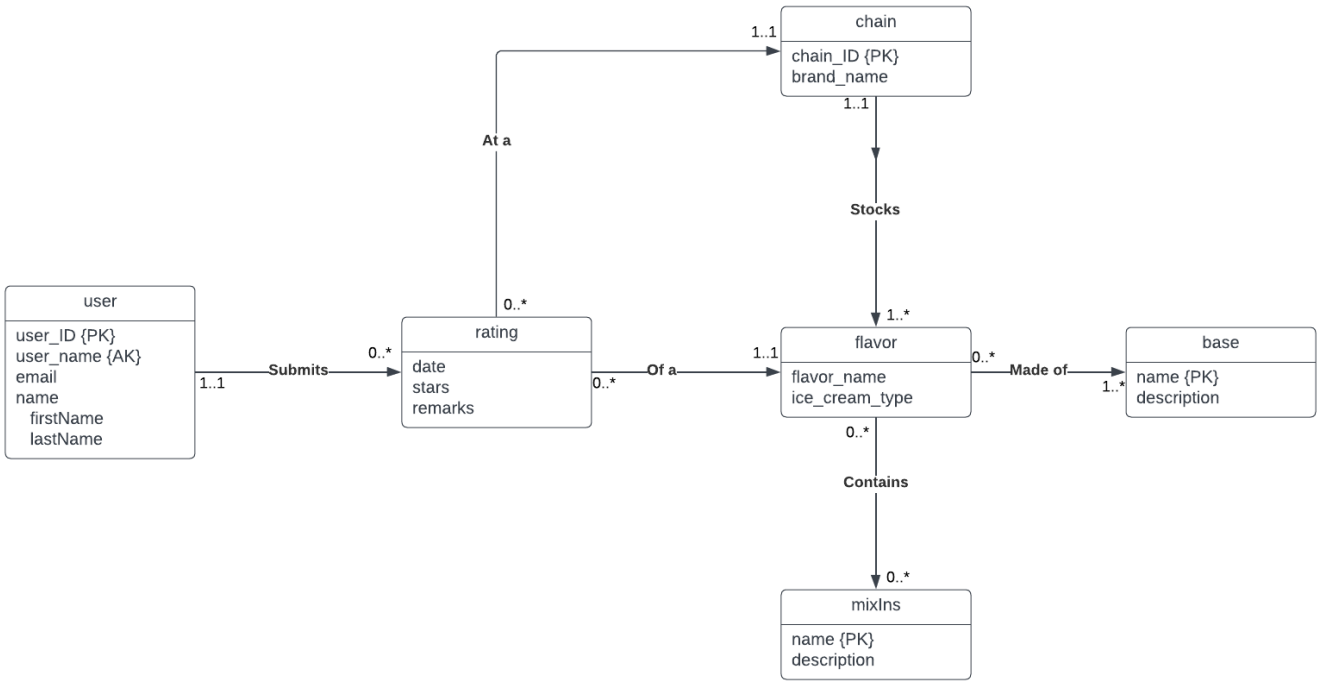
1) Unpack zip file and download files to chosen directory. Make sure all python files listed above are included. All python files except "ArmandZ.py" are support files. Only "ArmandZ.py" runs the application
2) Open ArmandZ.sql in database client. Run file to import database dump for `new_england_ice_cream` database.
3) Determine local database host, username, and passwordfor use in python script. You can find the host name by running the following command in your SQL client:
   SHOW VARIABLES WHERE Variable_name = 'hostname';
4) Make sure that python3 instance is installed on your machine with packages listed above installed. Instructions can be found online.
5) Run python file 'ArmandZ.py'. All examples below are using MacOS Terminal commands, but steps should be the same regardless of OS.
   a. Make sure that MySQL database connection containing imported `new_england_ice_cream ` is running.
   b. Make sure that required packages are installed in python.
   c. Open command line/terminal
   d. Change directory to directory containing python file:
      'cd <folder name>'
   e. Run python file. Individual commands may vary based on your installation, such as "python" or "python3", etc.:
      'python ArmandZ.py'
   f. When prompted, enter the correct host, username, and password for your MySQL database connection.

6.) Follow prompts. Application will terminate upon user prompting script to exit.

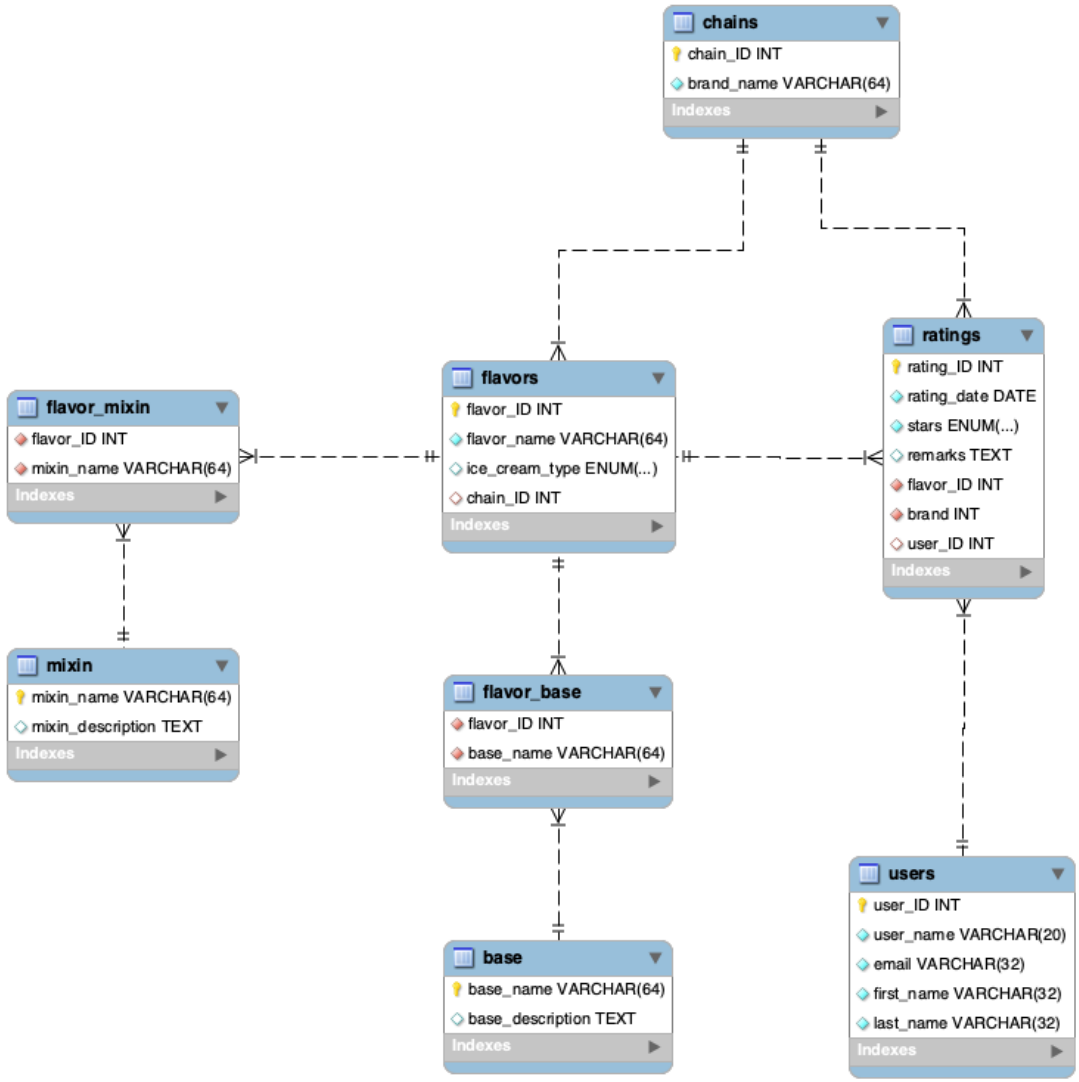
Technical Specifications

This project uses a relational database (SQL), using MySQL. Additionally, the frontend uses python, utilizing a basic terminal/command line user interface. All operations for the user occur in the command line. There are no known machine restrictions.

UML Diagram



Logical Design (ERD)



User Flow

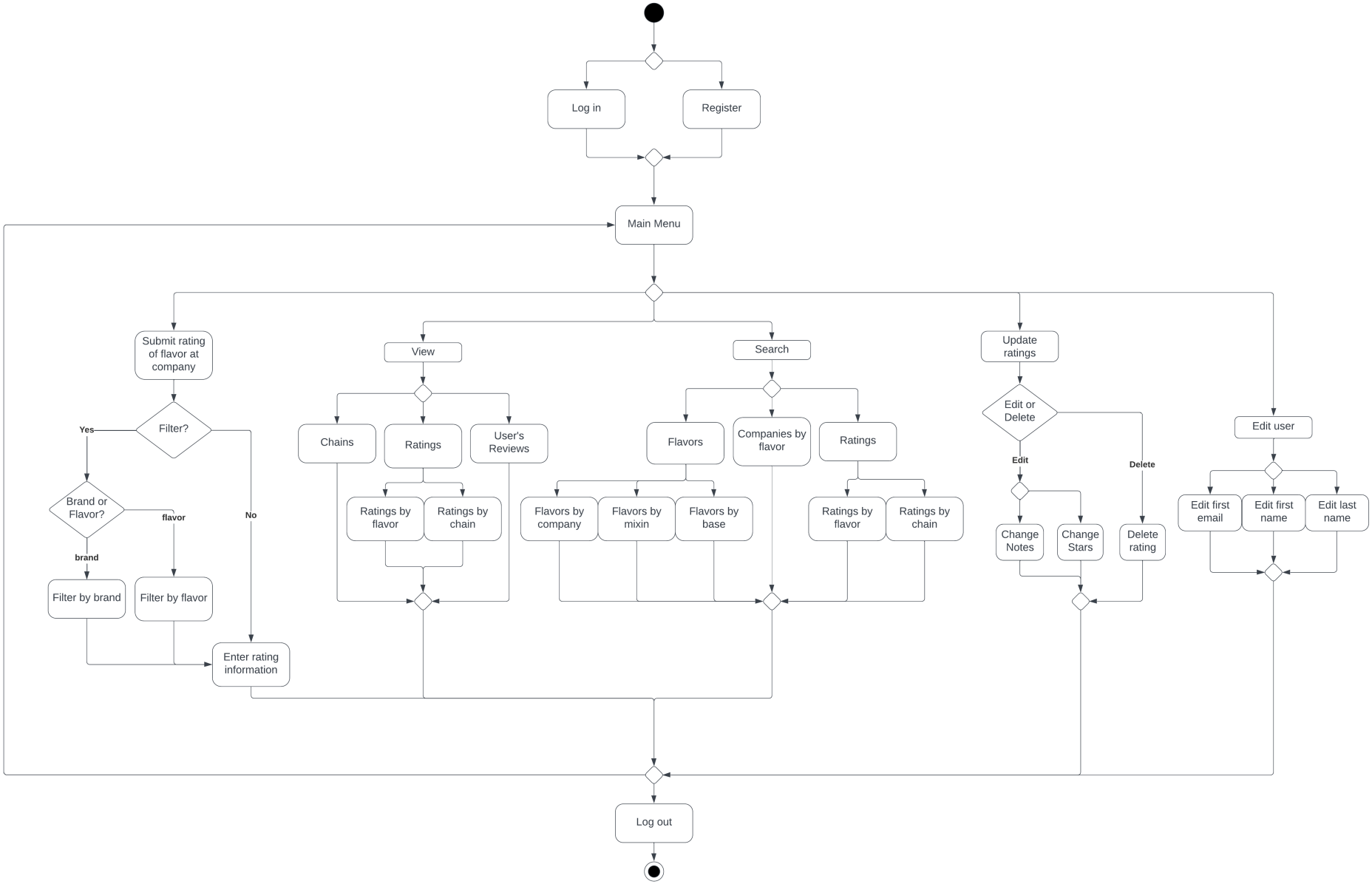
First, the user decides whether to login, register an account, or quit the system
Once successfully logged in, they can:

1. Explore network
    a. Avg. reviews by chains
    b. Avg. reviews by flavor
    c. All chains
    d. All your reviews
2. Submit review
    a. Filter flavors by flavor or chain
3. Search flavors, companies, reviews
    a. flavors by company
    b. companies by flavor

    c. reviews by company
    d. reviews by flavor
    e. flavors by base
    f. flavors by mix in
4. Edit review
    a. Edit stars
    b. Edit remarks
    c. Delete review
    d. Return to menu
5. Edit user info
    a. Edit email
    b. Edit first name
    c. Edit last name
6. Quit

Throughout each of these sub-menus, users have the choice to return to the main menu. In the main menu, they can quit the program.

Activity Diagram



Lessons Learned

Technical Expertise Gained

This project taught me a multitude of lessons about SQL, python connections to SQL, and user interfaces. I had never worked with connecting python to MySQL (besides during the homework), so I was able to really get a good grip on how to connect, query, and commit changes to a database from a program. I also had not done much work on user interfaces, technically or conceptually, (and arguably still have much to learn), so it was a learning experience trying to user-proof input and make sure that nothing a user inputted would cause the system to crash or commit a bad query. I also gained experience writing SQL procedures and functions, which I had also worked on in this class, but gained more practice experience tying them into an external database connection.

Insights

I could continue to add more functionality to this project, and there's an almost infinite amount that I could also learn. I spent a lot of time writing python code, and there's some aspects of the data domain that I definitely felt myself wanting to improve or update, if time allowed (like some of the changes listed later). It was a challenge to not rabbit hole too deeply into one aspect of the project as I worked on it. I found myself reminding myself to return to the bigger picture instead of perfecting one small aspect of the whole project.

Alternative approaches

I considered having users be able to add flavors. However, after thinking about the data and user domains, I realized that this fell best into the user domain of another, not yet developed user class for chain representatives. I also considered having a way to users to see which flavors are in stock, but realized that the other user class for chain representatives would probably be needed in order to make that functionality useful (updating in stock/out of stock). Having a way to suggest missing flavors would also be interesting.

Documentation

As far as I am aware, all code is working, and there should be no instances where errors cause the program to crash. After rigorous testing, I am confident that all code works as intended.

Future Work

Planned Usage

Planned uses might include me using this in my own life! I thought of the idea as a way to track flavors that I've tried at different chains, and keep track of fun and unique flavors that I've enjoyed at different chains. So, I could try to manually update the database with my own personal thoughts as if I were a user. Also, perhaps I could work on a project to scrape menus online in order to fully populate that database. No concrete plans are in motion yet.

Potential Added Functionality

One major added functionality would be to add more user classes, such as chain representatives, so that they could keep their menus up to date. This way, it could be ensured that all flavors are represented for a location, and chain representatives could even see which flavors are popular to make sure that they keep stock.

I could also add a table for locations in the future, so that chains could be decomposed into specific locations. That way, if a user wanted to try a flavor, they could see if there are any branches near them. I didn't implement this functionality partly because of time constraints, and as a start, chains generally have the same menu across locations, so there wasn't a major need in order to have a complete data domain. If I knew more, I could also make a more elegant user interface.

Of course, with more knowledge and expertise, I could add more user interface functionalities. Right now, my user interface is nearly as basic as possible, using a command line interface. If I knew of more ways to make web interfaces or database application systems, I could create a future versions that migrates to one of these more graceful solutions.