EECS 341 Project Report

Project Title: Woohoo Pizza

Name of Team Members: Sid Thakur, Isaac Ng, Mark Wang

Description of Proposed Project: We built a basic pizza ordering website for a business named Woohoo Pizza. They are first brought to the login page where they can either sign up or sign in using an existing user account stored in the database. If they sign up, they will be given an alert that they have been able to sign up. If they sign up, they will then be taken the menu/ordering page. They can add various items and then click submit order where they will be taken to the order history page, which also has various log. They can then go back to the menu or logout.

Scope: The frontend is basic html/css/js with the Flask framework being used in order to connect and send queries to our MySQL database. The MySQL database used our setup sql file in order to create the tables and insert various data into it. The orders and sign ups update the database and the order history page should call analytics queries in order to get past order history, calorie count, and total price from the MySQL database.

Technology requirements:

- Application server must be able to run on the class Linux server
  - o We used Flask which calls our config.ini file in order to have the permissions needed in order to connect to and write to the database on the Linux server.
- Application database must be MySQL.
  - o We are using the MySQL database available on the Linux server.
- User interface must run in a common web browser, e.g., Chrome or Firefox.
  - o Website/frontend UI works on Firefox and Edge (Not Chrome due to them denying XMLHttpRequests.
- No back-end frameworks (for example, ORMs) are allowed unless approved by the instructor.
  - o Not using this.

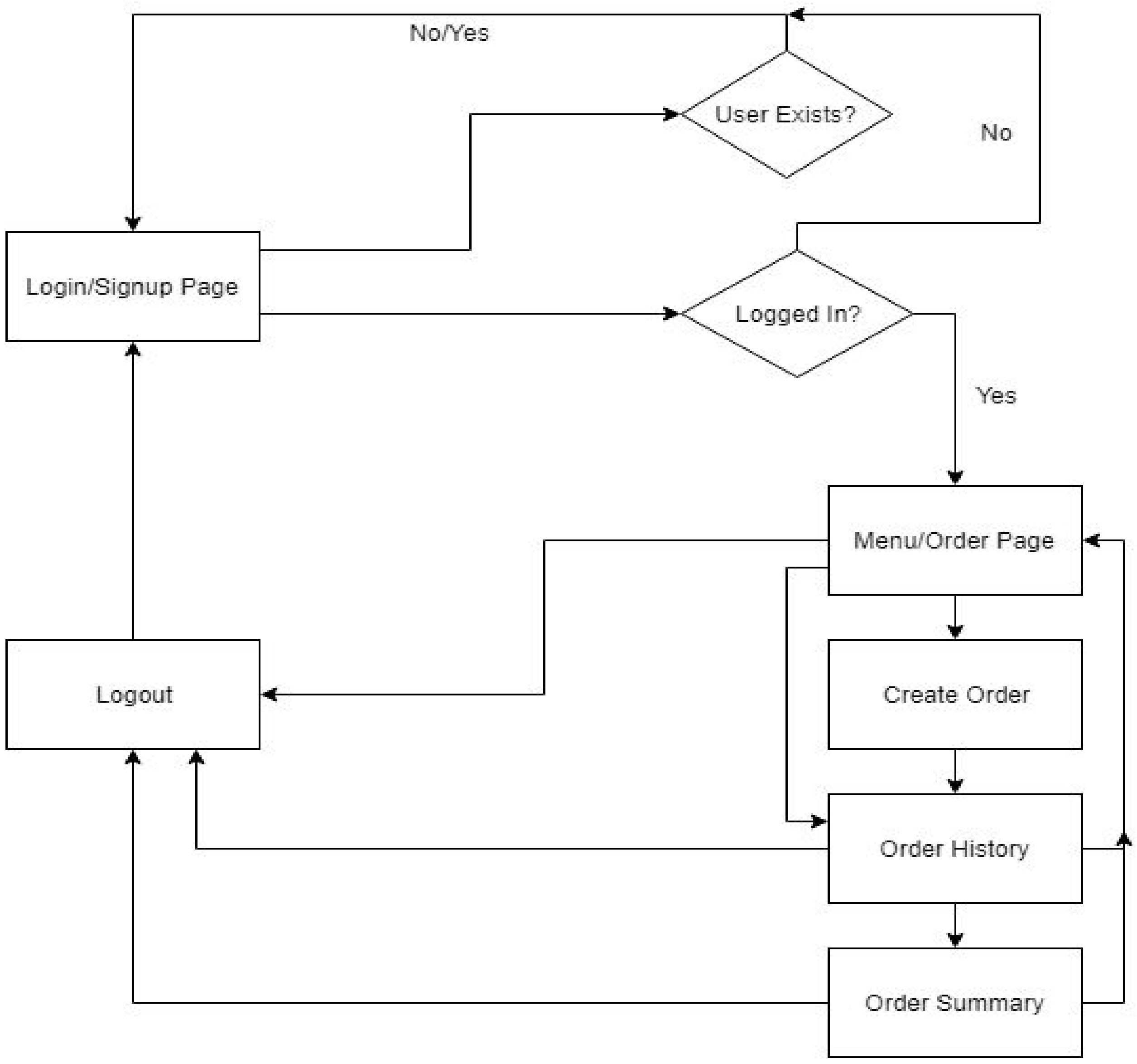
Database interaction requirements:

- Application must create, read, update, and delete data via SQL.
  - o Our setup.sql file creates the tables that we write to. Our run.py file contains the necessary commands in order to create, read, update, and delete data depending on the buttons pressed and features utilized (sign up, submit order, order history page, sign in, delete order history).
- Application must include some queries that utilize aggregate functions in SQL.
  - o When you visit the order history page, we execute analytical queries in order to calculate the total price and total calories of each order.

What we did beyond the requirements:

- Ensured password security of users by storing/comparing hashes

Basic UI Framework:



Contributions:

- **Isaac:** I basically functioned as the database manager and frontend helper/full stack. I helped create the config.ini file, connected to the database, created tables, deleted tables, and various data. I did a bunch of updating the databases as we discarded columns and updated the columns values in the tables. It was a nightmare. I also edited the frontend files in order to implement basic functionality and debugged the various issues occurring in the frontend. I also implemented good coding practices and improved quality of the code while debugging both frontend and backend.
- **Mark:** I functioned as the frontend developer. I built out most of the frontend UI using basic HTML, JS, and CSS. I created the menu.html and order history.html. I coordinated with the backend developer in order to properly design features and buttons in order to

send data properly to the backend, as well as the structure of the class to send to the front end. I also styled all of the pages, and set up the Flask code in those pages to display database information.

- **Sid:** I was the backend developer. I setup the Flask aspect of the project which enabled the connection of the frontend and backend and enabled the basic use cases for our website. I worked on functionality that dealt with writing, reading, creating, and deleting data from the database. I also created the GitHub repository and organized various meeting times for us to work.

Things we learned:

- **Isaac:** Javascript is a very hard language to learn after learning the fundamentals of other languages. The developer console in Google is a good tool for figuring out what is going wrong with your code. However, you have to keep in mind that often times the line error doesn't point to exactly the line the error is at. Often times, syntactical errors much earlier cause the line that the error points to breaks and you have to go debug and search for various JS language paradigms that you didn't obey. I also learned the various challenges of setting up a three tier web app via Flask, HTML/CSS/JS and a MySQL database. Learning about the MySQL database and SQL more in depth by interacting with it so much through the ubuntu vm. I also had to do a lot of database debugging and figuring out which MySQL statements worked and how to alter the tables appropriately. It was honestly a nightmare due to our poor relational model.
- **Mark:** I learned XML requests, how to work with Flask on the front end, by generating the for loops that would display all the necessary code, as well as working with someone else to create the data structures that would best represent our relational model in the frontend to pass to the backend. The Javascript code that I wrote for the food_menu.html took a long time because Issac and I had to comb through for simple syntactical errors, as often times text editors wouldn't register them. In the future, more familiarity with Sublime may be required. In addition, integrating the data that I got from the backend was another challenge, and properly displaying the information.
- **Sid:** Flask redirects can be tricky, I spent time figuring out why certain requests would redirect the user and others wouldn't. I learned how to use cookies to store information on the user's browser. I learned jinja2 templating for putting python variables directly into html and creating for loops to generate rows. I learned about the cursor.lastrowid for python's mysql connector which allows us to retrieve the primary key/id of the last row inserted into db. I also learned about how to automate generating prepared statements.