**Proposal for Second Project**

**About my idea:**

My idea is to create a McDonalds system for both employes and customers.

Restaurant staff can manage orders, track inventory and handle customer data. This system can be beneficial for employees because it helps them to be more efficient, to provide quick service, to manage resources effectively and to create various statistics that help them improve their services.

Customers might want to place orders online. For those who have this opportunity, it enhances their experience and satisfaction.

**Main features that will be implemented are:**

* *Menu management* (Each product should have an ID, name, price, category and availability status. This can be accessed by both employees and customers).
* *Order placement and tracking* (Customers can place orders and track them; for example, the order status will be updated to help the person know when to expect the order and whether there has been a delay).
* *Employee management* (Handle employee's data: ID, name, etc. Add, update, and remove employees)
* *Customer management* (Handle customer data: ID, name, age, gender, address, and order history. For such a big company it is important to create statistics based on the average age, gender or the most ordered products. Add, update, and remove customers)

For sure, I will implement a stat but more likely a simple one based on gender or age.

**Networking features and what my database includes:**

In my mind I have this scenario in which the server takes clients (customers) orders. It may offer them first recommendations based on their order history, and then present what is available on the menu, and ask them what they want to serve. Also to have a simple way of tracking the order by simply printing some informational messages: the order is being prepared, the order is being delivered, etc. And maybe to end up with clients' reviews.

In addition, an employee can modify the menu by interacting with the server.

Therefore:

* *Server Role*: Process requests from the clients, manage the menu, take orders, provide updates on order status and handle recommendations.
* *Client Role*:
* Customers: They interact with the server, they can view the menu, place orders, receive updates and maybe they can also provide feedback and reviews.
* Employees: They can modify the menu.
* Use HTTPS for such interactions, simulated over the TCP connections, such as:
* GET: a customer request to see the current menu
* POST: a customer places an order
* POST: an employee updates the menu
* DELETE: an employee removes something from the menu
* Recommendations are made by storing customers' past orders in a database.
* Both server and client will use a TCP socket (to listen for incoming connections and to establish a connection with the server, to send requests and receive responses)

My database includes information about the menu, clients and employees separately. I should store information about menu items (ID, name, price, category and availability status) which can be modified as an employee and visualized as a customer, customers (ID, name, age, gender, address and order history) and employees (ID, name, etc.). I may modify this structure in the future. For example, including a table for orders or making a different table for recommendations. This really depends on how I will develop my system.

**External libraries:**

* *Flask* – used for developing web applications famous for simplicity and independence. I will use it to create and manage the web server and endpoints.
* *Flask-RESTX* – an extension for Flask that adds support for quickly building REST APIs. This simplifies the API development process with additional features.
* *socket* - a quick connection which allows the transmission of data between two processes on the same machine or different machines over a network. This is how I will manage the relation between server and clients, crucial for managing the menu and processing orders.
* *SQLAlchemy* - allows developers to access and manage SQL databases using Python classes and objects. I will use it to manage the interactions with the database efficiently.
* pytest - a Python testing framework that originated from the PyPy project. I will use it to ensure that the application functions correctly.

Above, I only mentioned a few HTTPS, but there will be many. I will cover these and additional details in a README file. Mainly there will be a similar structure to the last project, but now I am focusing on how I will use the TCP socket and how I will manage the databases. I will attach a simple architecture of my project made in a Whiteboard.