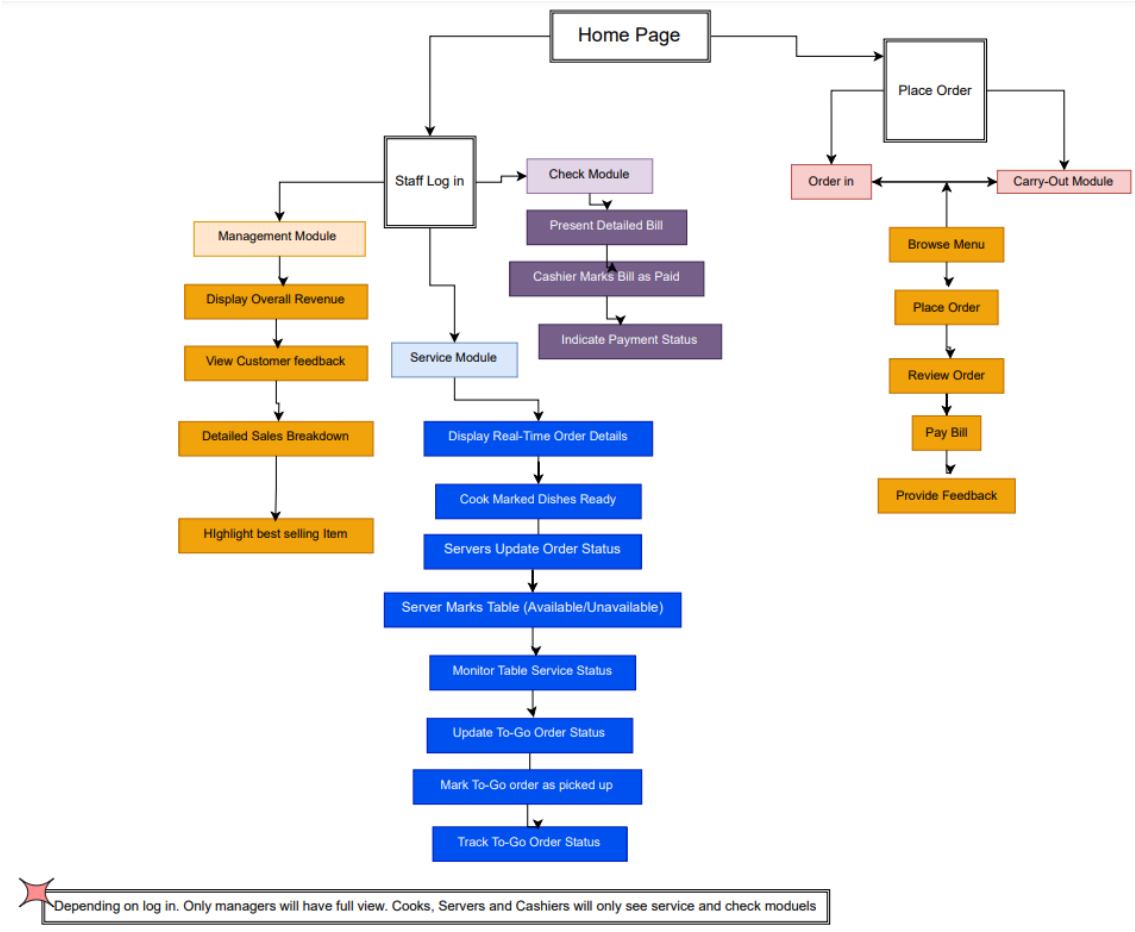
Problem Statement

Explain the purpose of your project briefly.

- As a business you need to know what your best and worst selling product is, and you need to make sure your restaurant is running smoothly and efficiently. That's what this apps go is. This project aims to develop a streamlined restaurant management platform to facilitate the interaction between customers, servers, cashiers, and manager. The system can provide real-time information on table availability, service status, and checkouts, enhancing operational efficiency. Additionally, the system will feature managerial oversight, offering the restaurant manager insights into total income within specified period and detailed analytics on dish popularity and top sales.

Flow Diagram:



Module Breakdown:

Explain the purpose and functions of modules that your project will use. You will not need to show the implementations here, that is for the final project, but in your final report this section will hold code and discussion on the features or importance of each module you add.

1. **Login Module:** To authenticate the identity of restaurant staff, ensuring only authorized personnel can access the system
    a. Use pygame package to create an interface that enables the restaurant staff including servers, cashiers, cooks, mangers, to log into the system (as shown in the image below).

Home Screen

2. **Order Module:** Enable customers to view the menu, place orders, pay the bills, and provide feedbacks through the app.
    a. Enable customers to browse the menu.
    b. Provides an interface for customers to place their orders.
    c. Offers a feature for customers to review their current and past orders.
    d. Enable customers to pay their bills through the app.
    e. Allows the customers to provide feedback on dishes and service quality.

3. **Carry-out Module:** To offer customers a convenient platform to view the restaurant's offerings and place orders for takeaway.
    f. Enable customers to browse the menu through the app.
    g. Provides a user-friendly interface for customers to place to-go orders.
    h. Enable customers to pay their bills through the app.
    i. Allows the customers to view their order status.

4. **Service Module:** To optimize the coordination between the kitchen and floor staff, ensuring timely preparation and delivery of orders.
    j. Displays real-time order details to both cooks and servers.
    k. Gives cooks the ability to mark dishes as ready, notifying servers promptly.

l. Allows servers to update the status once a dish is served to customers.
m. Provides servers with a feature to mark tables as available or occupied.
n. Enables servers and managers to monitor the real-time service status of each table.
o. Displays table availability dynamically to servers and managers, facilitating quick seating decisions.
p. Allows servers to update if a to-go order is ready for picking up, notifying customers their order status.
q. Provides servers with the feature to mark a to-go order as picked up, ensuring accurate order tracking.
r. Allows servers and managers to track the live status of to-go orders.

5. **Check Module:** To handle the financial transactions ensuring customers are billed correctly and payments are tracked.
    s. Presents the detailed bill for each table/ to-go order to cashiers, servers, and managers.
    t. Enables the cashiers to mark a table's bill/ to-go order as paid when payment is received over the counter.
    u. Indicates to servers, cashiers, and managers whether a table/ a to-go has been paid.

6. **Management Module:** To provide an overview of the restaurant's operations, assisting managers in making informed decisions.
    v. Displays overall revenue (including dine-in and to-go) during a specified timeframe.
    w. Allows managers to view feedback from customers.
    x. Provides a detailed breakdown of sales for each dish during a chosen period, allowing for menu optimizations.
    y. Highlights the best-selling items, giving insights into customer preferences.

Codebase Organization:

In this section, explain how your code will be organized. Even if your project is not so large that it needs a comprehensively planned organizational pattern, it must be scalable to add new features or modules in a way that will not necessitate reorganizing the project.

We will include the following files to realize the module functions in this project:

1. **home_pygame.py**: use pygame to design a home screen that have two options: place order and staff login.

2. **dish.py:** define a Dish class that stores the attributes include dish name, dish price.

3. **table.py:** to define a Table class that has the following attributes:
   - table ID
   - ordered_dishes: a list of all the dishes that a table has ordered (to be updated by customers)
   - uncooked_dishes: a list of dishes that have not been cooked (to be updated by cooks)
   - ready_dishes: a list of dishes that have been cooked and ready to be served (to be updated by cooks)
   - pending_dishes: a list of dishes that has not been served for the table (to be updated by server)
   - feedback: feedback from the customers of a table (to be updated by customers)
   - bill: total bill amount
   - is_occupied: whether the table is occupied (False) or not (True) (to be updated by server)
   - all_served: "True" if all the dishes for this table have been served or "False" if not
   - payment_status: if a table has paid their bill (True) or has not paid (False) (to be updated by customers or cashier)

4. **togo.py:** define a Togo class that inherit most of attributes from Table class except the following attributes:
   - order_number: instead of Table_ID, each to-go order will be assigned a unique order number.
   - ready_pickup: instead of “all_served” attribute in “Table” class, the “Togo” class will have an attribute to describe whether a to-go order is ready for picked up. "True" if it is ready and “False” if not (to be updated by server)
   - is_pickedup: “True” if the order has been picked up or “Fasle” if not (to be updated by server)

5. **restaurant.py**: define a “Restaurant” class that has the following attributes:
   - tables: a list of tables that have placed orders in the restaurant
   - revenue: total earnings from all tables
   - dish_sales: a dictionary with dish names as keys and the number of times each dish has been sold as values

   The Restaurant class also contains the following methods:

   - best_seller(): return the best-selling dishes during a specified timeframe
   - display_serice_status(): use dictionary to display each table's service status

- display_table_availability(): use dictionary to display table availiblity in restaurant
- display_togo_status(): use dictionary to display each togo order's status

6. **server.py:** define a "Server" class that has attributes including username and password. "Server" class will also have the following methods:
   - update_table_order_status():to update table order status, and has two parameters: a Dish instance and a Table instance
   - mark_table_occupied(): to update whether a table is occupied or not, and receive 2 parameters: a Table instance and occupied (True/False)
   - update_togo_order_status(): to update a to-go order status, and receives 2 parameters: a Dish instance and a Togo instance
   - update_pickup_status(): to update whether a to-go order has been picked up, and receive 2 parameters: a togo instance and is_pickedup (True/False)

7. **cashier.py:** define a "Cashier" class that inherit attributes from the "Server" class
   The "Cashier" class has the following methods:
   - update_table_payment_status(): to update the payment status of dine-in orders and receive 2 parameters: table instance and is_paid(True or False)
   - update_togo_payment_status(): to update payment status for to-go orders and receive 2 parameters: togo instance and is_paid(True or False)

8. **cook.py:** to define a "Cook" class that inherit attributes from the "Server" class
   The "Cook" class has the following methods:
   - update_tableDish_status(): to update whether a dish has been cooked and ready to serve to table, and it has two parameters: a Dish instance and a Table instance
   - update_togoDish_status(): to update whether a to-go dish has been cooked and ready to be packed, and it has two parameters: a Dish instance and a togo instance

9. **manager.py:** to define a "Manager" class that inherit attributes from the "Server" class

10. **table_customer.py:** to define a "TableCustomer" class that has the attributes including:

    - time: when a Customer instance was created
    - table: a Table instance
    - restaurant: a Restaurant instance

    "TableCustomer" can have the following methods:

    - add_order(): to add dish to its table order
    - add_feedback(): to add feedback
    - make_payment(): to make payments (update payment status)
    - display_orders(): to display the orders of the table
    - display_bills(): to display bills of the table

11. **togo_cutomer.py:** to define a “TogoCustomer” class that inherit the attributes from “TableCustomer” except the following:
    - Instead of a “Table: instance, it receives a “Togo” instance

    “TogoCustomer” can have the following methods:

    - add_order(): to add dish to a to-go order
    - add_feedback(): to add feedback to a to-go order
    - make_payment(): to make payments (update payment status)
    - display_orders(): to display the orders of the to-go order
    - display_bills(): to display bills of the to-go order

12. **main.py**: Create and initialize class objects, provides an entry point to the application, and integrate the above functionalities to work together.

Task Assignment

Here you will outline the assigned tasks for each of your team members. Have them explain their contributions along with estimated time that it will take to complete their tasks

Member Luis Lozano:
Define attributes and methods for Dish class, and clarify relationships between the Dish and other classes. The attributes should include the dish name and dish price (1 hour). Will also be working on any other class that involves the dish class, for example, file.(3 hours)

Member Tina Zhou:

Define attributes and methods for Table class, and clarify relationships between the Table and other classes. This class should have roughly 10 attributes(5 hours). Will also be working on any other class that involves the table class, for example, tableCustomer.py file. (3 hours)

Member Nathaniel Salazar:
Define attributes and methods for Restaurant class, and clarify relationships between the Restaurant and other classes (member C). This class should have roughly 3 attributes with 4 methods(2 hours). Will also be working on any other class that involves the restaurant class, for example, server.py file.(4 hours)

SIDE NOTE: if need to split the workload more we will, and we will help the next person out if we finish our share of the work.