Phase II: Design Report

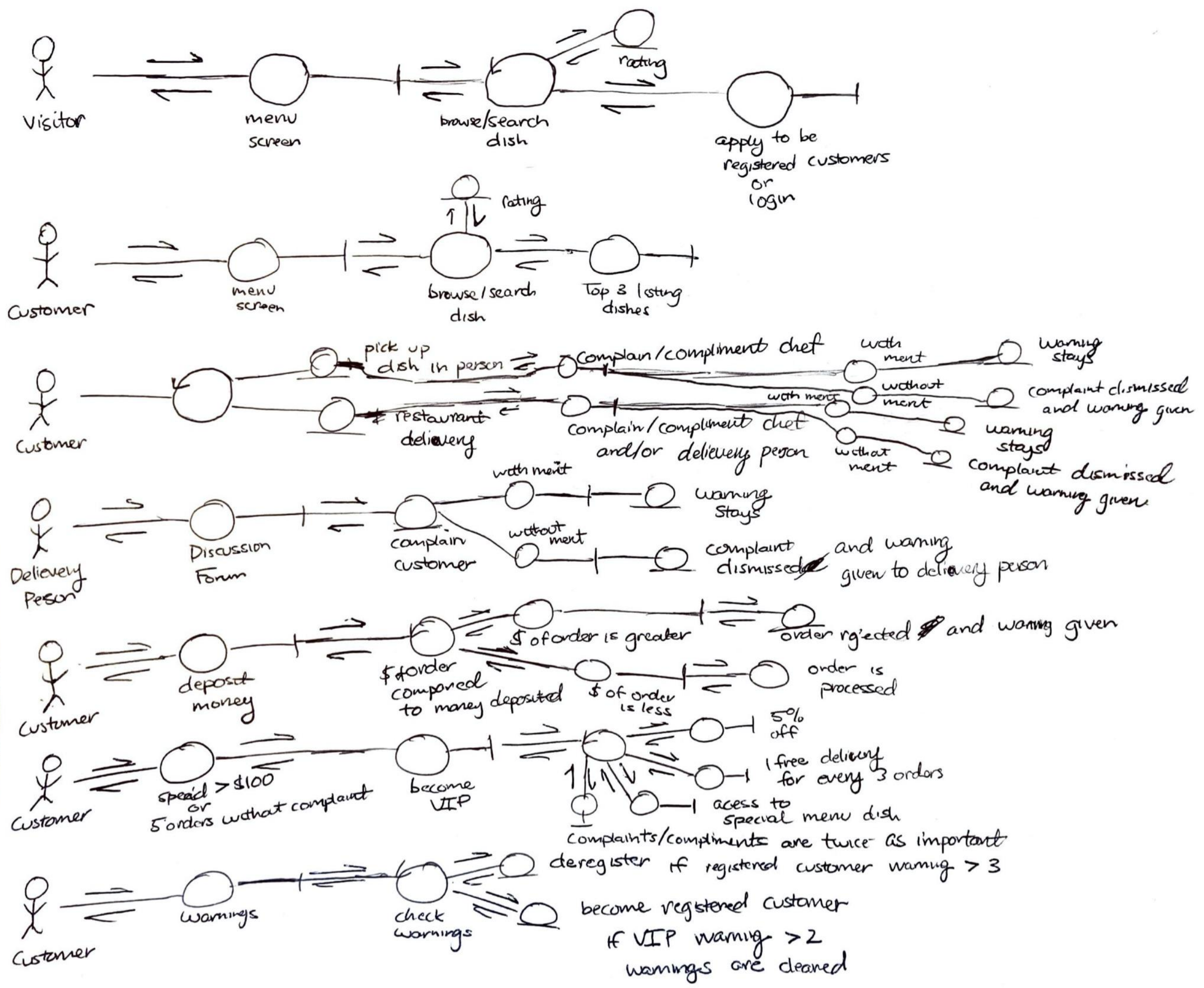
Team U: Ahmed Imran, Abdul Raheem, Stefan Tan

This report is meant to provide the data structure and logic to carry out the functionalities dictated by the specification. Keep in mind the implementation will be entirely based on this document, adequate details should thus be given at length.

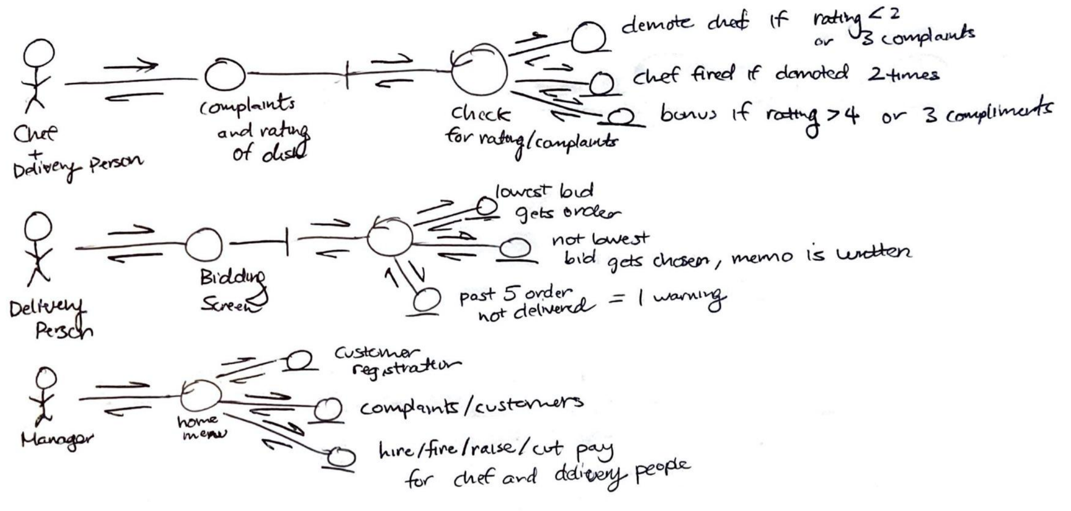
Deadline: Apr. 28, Thu.

1. **Introduction**

   an overall picture of the system using collaboration class diagram. Please see the next page.



Visitor
menu screen
browse/search dish
rating
apply to be registered customers or login
Customer
menu screen
browse/search dish
rating
Top 3 listing dishes
Customer
pick up dish in person
complain/compliment chef
with merit
warning stays
without merit
complaint dismissed and warning given
restaurant delievery
complain/compliment chef and/or delievery person
with merit
warning stays
without merit
complaint dismissed and warning given
Delievery Person
Discussion Forum
complain customer
with merit
warning stays
without merit
complaint dismissed and warning given to delievery person
Customer
deposit money
$ of order compared to money deposited
$ of order is greater
order rejected and warning given
$ of order is less
order is processed
Customer
spend > $100 or 5 orders without complaint
become VIP
5% off
1 free delivery for every 3 orders
acess to special menu dish
complaints/compliments are twice as important
Customer
warnings
check warnings
deregister if registered customer warning > 3
become registered customer if VIP warning > 2
warnings are cleared



Chef + Delivery Person
complaints and rating of dish
check for rating/complaints
demote chef if rating < 2 or 3 complaints
chef fired if demoted 2 times
bonus if rating > 4 or 3 compliments
Delivery Person
Bidding Screen
lowest bid gets order
not lowest bid gets chosen, memo is written
past 5 order not delivered = 1 warning
Manager
home menu
customer registration
complaints/customers
hire/fire/raise/cut pay for chef and delivery people

2. **All use cases**

o Scenarios for each use case: normal AND exceptional scenarios

Open Account: Visitors can open an account to become customers. Delivery people and chefs can open accounts to apply for their jobs. The manager will be a system admin by default and won't need to open an account. Customers can't open accounts since they are considered to already have one. Users will have to wait for their accounts to be registered before they have full access to the application.

Register User: The manager can verify customers and give them access to the application. Users who were previously banned will not be able to register again.

Log In: Registered users can log into their accounts. They will have three attempts to log in before the account is locked.

Hire/Fire Employee: The manager can register chefs and delivery people by hiring them after they open an account. He/she will also be prompted to fire an employee when they become flagged due to multiple complaints. The manager must submit a reason for firing any employee that isn't flagged by the system.

Raise/Cut Pay: The manager can adjust the pay of the employees. Salary of employees with multiple complaints can not be raised until the complaints are marked as resolved. The manager will be prompted to give a raise to employees with multiple good reviews. The salary for any employee can not be less than declared minimum wage for the region.

Edit Menu: The chefs will be able to make changes to the menus and can add or remove any items they want. The manager will be prompted to give final approval to any changes made.

Browse Menu and Ratings: Both visitors and registered customers can browse the restaurant menu and ratings for all items.

Deposit Money: Customers can deposit money to their accounts. The amount deposited has to be greater than 0.

Order Food: Customers can add items to their cart and order food. They will choose either pickup or delivery. Upon checkout, if their account balance is lower than the total cost, the order will be rejected and a warning will be sent to their accounts.

Update Dish Status: Chefs will be able update the status of the order. Statuses include "order received", "in the works", and "ready". If the order is for pickup, the customer will be notified. If it is delivery, the manager and delivery people will be notified.

Bid for Order: Delivery people will be able to bid for any orders that are marked ready.

Select Bid for Delivery: The manager will choose the winning bid for each order. Each order will be assigned to only one delivery person.

Review Food/Delivery: The customer will be able to leave a review on the food. If the food was delivered, the user will also be able to leave a review on the delivery. The order must be marked as "picked up" or "delivered" before the customer can leave a review.

Review Customer: The delivery people will be able to leave a review on the customer after the food is delivered. The order must be marked as "delivered" before the delivery people can leave a review.

Review Complaints: The manager can look over all pending complaints. He/she can either accept or invalidate complaints. Accepted complaints will send a warning to the accountable party. Any complaint deemed invalid will send a warning to the reviewer's account.

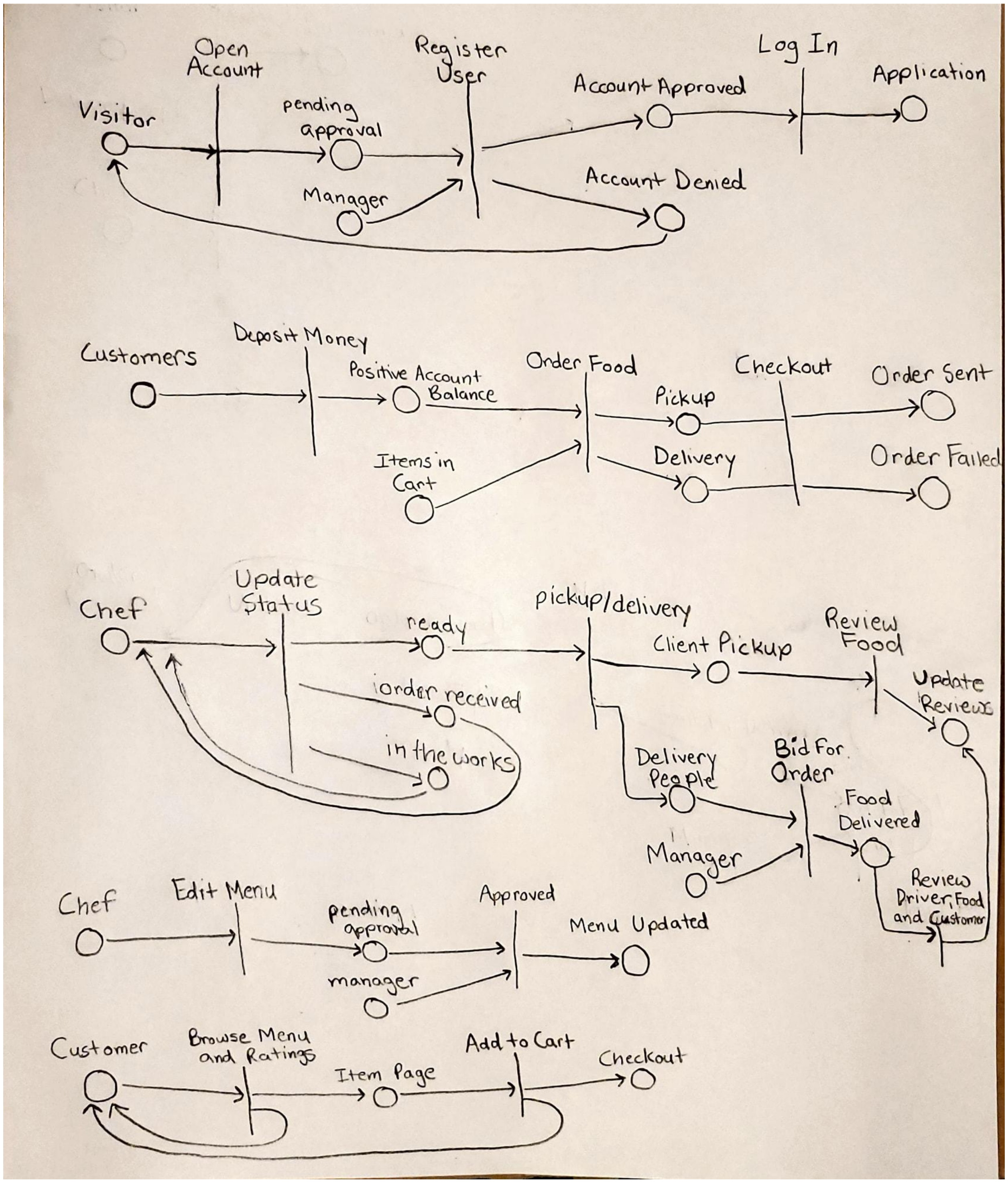
Dispute Complaints: Customers and employees have the option to dispute any complaints that they receive.

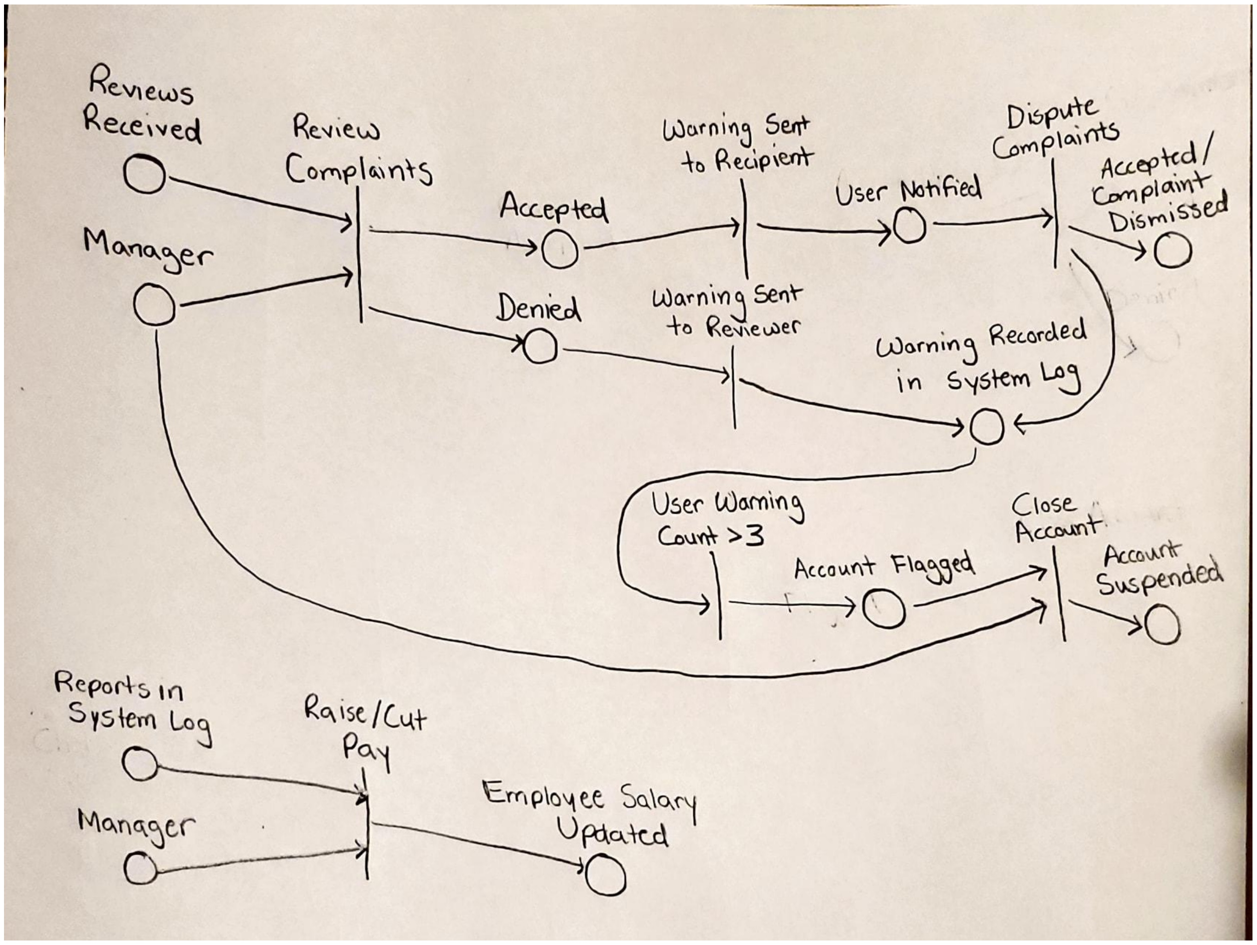
Discussion Posts: Users can view and write to the discussion forum about anything they want. Users must be registered, i.e. they are verified customers or employees. This ensures accountability.

Close Accounts: The manager is responsible for closing accounts. Accounts can be closed because of flags or an extended period of inactivity. For the latter case, the user can appeal to reactivate their account at a later time if he/she wants to.

o Collaboration or sequence class diagram for each use case, choose 3 or more use cases:

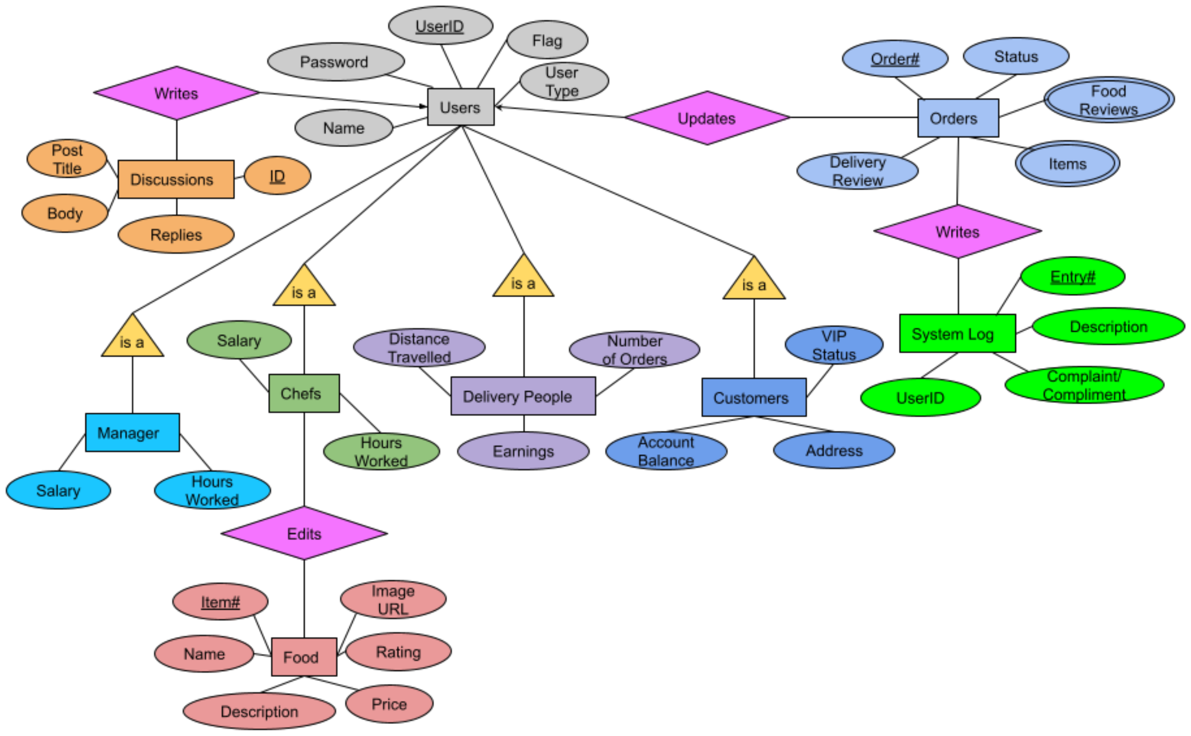
draw the Petri-nets instead of class diagrams




Reviews Received
Manager
Review Complaints
Accepted
Denied
Warning Sent to Recipient
Warning Sent to Reviewer
User Notified
Dispute Complaints
Accepted/ Complaint Dismissed
Warning Recorded in System Log
User Warning Count >3
Account Flagged
Close Account
Account Suspended
Reports in System Log
Manager
Raise/Cut Pay
Employee Salary Updated

3. **E-R diagram for the entire system**

attributes and key for each class should be provided



4. **Detailed design:**

   for EVERY method use pseudo-code to delineate the input/output and main functionalities

- -For our geocode API we will be using Google Maps

- -For our interface API we will be using Tkinter (inputbars, buttons, searchbars, etc.)

- -For SQL, we will be using mySQL

Import requests

Import json

Import urllib

CONNECTION method: Takes log-in credentials to fetch the proper user object in the database with the corresponding password/username.

Returns type of SQL connection

Static Connection con;

struct ORDERS: - - Queue structure that holds all pending orders that have been bidded for until they have been successfully delivered

Class Order:

```
    int ordernum;
    vector <int> shoppingCart;
    vector <int> foodReview;
    int deliveryReview;
    int status;
```

init_Order: Initializes the order if it is successfully placed and bidded for, pushing it into the queue

void setReviewFood(vector<int> rev): sets the review for all items on the shopping cart to what was input into this method once the order status is 1 or delivered.

void setDeliveryReview(int rev): sets the review for the driver once the order status is 1 or delivered

void isDelivered(): triggers when the order is popped out of the Orders queue, sets status as 1 to signify that it has been cooked and 2 to signify it has been delivered.

struct MENU: - - map structure that holds all menu items in one directory – we utilize the item# as keys for retrieval and the FOOD object as the value of the key

Class FOOD:

int foodnumber;

int price;

int rating;

int chefID;

String name;

String desc;

IMAGE img;

init_FOOD: constructor class to be used to create new food items

void updateRating(): Uses the rating database in our SQL server to update the rating of an item after every purchase (by fetching the average through a query)

void changePrice(int newPrice): Changes the item price to something else depending on the input

void changeDesc(String newDesc): Changes the description of the item.

Superclass USER: - - abstract class to act as a template

int userID;

String userType;

String name;

String password;

Subclass CUSTOMER implements USER

int userID;

int balance;

int warnings;

int merits;

vector <int> recommendations; -- has the integer IDs of all the most commonly purchased dishes for VIP users OR the recommendations for new users.

bool VIP;

dataset Address;

init_CUSTOMER: Loads data from the database using the CONNECTION method and the log-in credentials.

If the user does not exist, create it in the database and initialize all values to 0 except address which the user will be prompted for in registration

Void setBalance: Changer method that changes the balance of the user after a purchase or a deposit

Int getDistance(address 1, address 2): Method takes two points on a map – the customer's location and a potential location of a store purchase.

These two points are then fed into a Geocode API through a query to get back the distance between them.

The distance is then returned using this method to be used for tracking.

void setVIP(): checks if the merit is in range to make a user VIP

Order placeOrder(bool pickup, vector <int> shoppingCart): Places an order from this user into the query for bidding– performs the following checks:

Ensures that the account has a valid balance

Ensures that the restaurant is within deliverable range

Once the two checks have occurred, we move the query to bids if it is delivery or put an ETA on the dashboard if it is pickup.

Increment the merit count of the user at the end of each purchase.

Run setVIP on the new merit to see if the User becomes a VIP

void incrementWarning(): Used to increment the current warnings on a user for the various system flags that might trigger it

void giveReview(vector <int> shoppingCart, int driverID): After a successful order has been received, method is invoked to individually give a compliment or a complaint to every item on the shopping cart.

This system of compliments and complaints are then queued up for employee review.

Subclass COOK Implements USER

int userID;

int pay;

int warnings;

int hoursWorked;

vector <int> dishes;

Queue complaints/compliments;

init_COOK: Initializes cook user to the application and loads data from USERS database

void incrementHours(int h): increments the number of hours worked after each workday.

void setPay(int s): used to change the pay variable within the chef object.

void removeDish(int ID): removes a dish from the chef's dishes vector– which lists all dish IDs that this chef is responsible for

void addDish(int ID): adds a dish to the chef's dishes vector– which lists all dish IDs that this chef is responsible for.

void incrementWarnings(): Increments the warning counter by one when a system flag is triggered

Subclass DELIVERYMAN implements USER

int payment;

int currentBid;

Order currentOrder;

int warnings;

Queue complaints/compliments;

INIT_Deliveryman: Initializer method to create delivery person for use in the application

Order bid(int i): Function to send a bid into the queue for an order, if the bid is accepted the function will return the Order object– otherwise return NULL

Calls setOrder to change the currentOrder

void setOrder(order o): sets the currentOrder object to the order in the parameters.

void incrementPayment(int i): takes the currentBid and the distance traveled to deliver the order to increment the payment that the driver receives.

**void incrementWarnings(**): Increments the warnings of the delivery man if triggered by a system flag.

int login(String username, String password)**: - -** connects to the database to retrieve the proper USER

Writes SQL query to return userID that corresponds to the proper username/password combination

Return 0 if unsuccessful

bool register(String fname, String lname, String address, String password, String typeU)

Writes SQL query to create a user with the fname, lname, address, password, and typeU prompted by the register.

This query can fail for the following reasons:

The username is already being used.

The type of the user is not authorized.

The address is not valid.

In this case, return false and end program. Otherwise, return true.

void registerInterface(): - - switches the main interface to the register interface with the following components:

Inputbar fname;

Inputbar lname;

Inputbar address;

Inputbar username;

Inputbar password;

Dropdown typeUser;

Button register;

If (register.callback(register(fname.input, lname.input, address.input, username.input, password.input, typeUser.input))) - - We use the a callback to assign a function to the button, in this case we assign register which was defined above

login(username.input, password.input); - - If we successfully register, we invoke the login method to login the user.

void loginInterface(): - - switches the main interface to the log-in interface with the following components

Inputbar username;

Inputbar password;

Button login;

Button newuser;

```
If ( login.callback(login(username.input, password.input)) != 0)
        loggedInInterface(login(username.input, password.input));
newuser.callback(registerInterface());
```

Void loggedInInterface(int userID):

--If the usertype is 1, it is a customer. 2 is chef. 3 is driver. 4 is manager.

```
int userType  = userID.type (we would get this from the userID superclass)
if (userType == 1){  - - if the usertype is a customer
    Customer x = new Customer(userID);
    Image dish1 = get image of the first recommendation in the Customers recommendation class variable
    Image dish2 = get image of the second recommendation in the Customers recommendation class variable
    Image dish3 =  get image of the third recommendation in the Customers recommendation class variable
    Button see-more.callback (
    for (int i = 0; i<menu.size; i++){
        new Image(get Image  from the ith member of the menu);
        new textBox(get description from the ith member of the menu);
        new textBox(get price from the ith member of menu);
        new Button(adds a button to the image so that a user can add the item to their checkout just by clicking);
    } )  - - This sets the see-more button to list every item on the menu along with their image, description, and price
    Button checkout;
    checkout.callback(placeOrder(userID, vector<int> shoppingCart);  - - links the checkout button to the placeOrder function of the Customer class
if (userType == 2){ - - the User is a chef
    Chef x = new Chef(userID); - - load chef object from the users database
```

```
        for(int i = 0; i<dishes.size(); i++){
                - - For loop lists out all dishes that this chef is responsible for on their
                dashboard
                - -Also prints out the compliments/complaints of these dishes that are
                currently pending
        }
        for(int i = 0; i<Orders.size(); i++){
                if ( chef is responsible for the order at i){
                        – print out the order on the chef's dashboard
                        – list a button that the chef can press if the order has been cooked
                        which changes the order status to 1 to show it is ready to be
                        delivered
        }
        **Button viewComments.callback(**
        for(int i = 0; i<complaints.size(); i++){
                --lists out the complaint queue for each individual chef onto their
                dashboard
                --option to agree or disagree with each complaint on the queue
        }
        **Textbox pay;**  - - retrieve pay from Chef object to list to the chef
        **Textbox warnings; - -** List any warnings as well
        }
if (usertype == 3) { - - if user is a driver
```

```
        Deliveryman x = new Deliveryman(userID);
        for(int i = 0; i<bids.size(); i++){
                – list out all bids for current orders for the deliverymen
                – create input bar and button for giving a bid to a current pending order
        }
        – if an order is successfully bidded, the dashboard for the deliveryman turns into
        an address route from the restaurant to the customer using the GeoCode API
        –button for successful delivery will set orderStatus to 2 and pop it from the Orders
        queue
        for(int i = 0; i<complaints.size(); i++){
                –Lists out all complaints for the delivery person and a button for whether
                or not they agree or disagree with the claims
        }
        – also lists the deliveryman's pay and hours worked for the day
}
if (usertype == 4){ - - user is a manger
        User x = new User(userID);
        for(int i = 0; i<complaints.size(); i++){
                – Manager queue for all complaints, where they can be approved or
                disapproved
                –can also invoke incrementWarnings method from other classes
        }
        Button changewage;
```

Dropdown employeeID;

– We can use a dropdown box with employeeID to manually change wages for chefs

Main function(will act as the interface):

Button login;

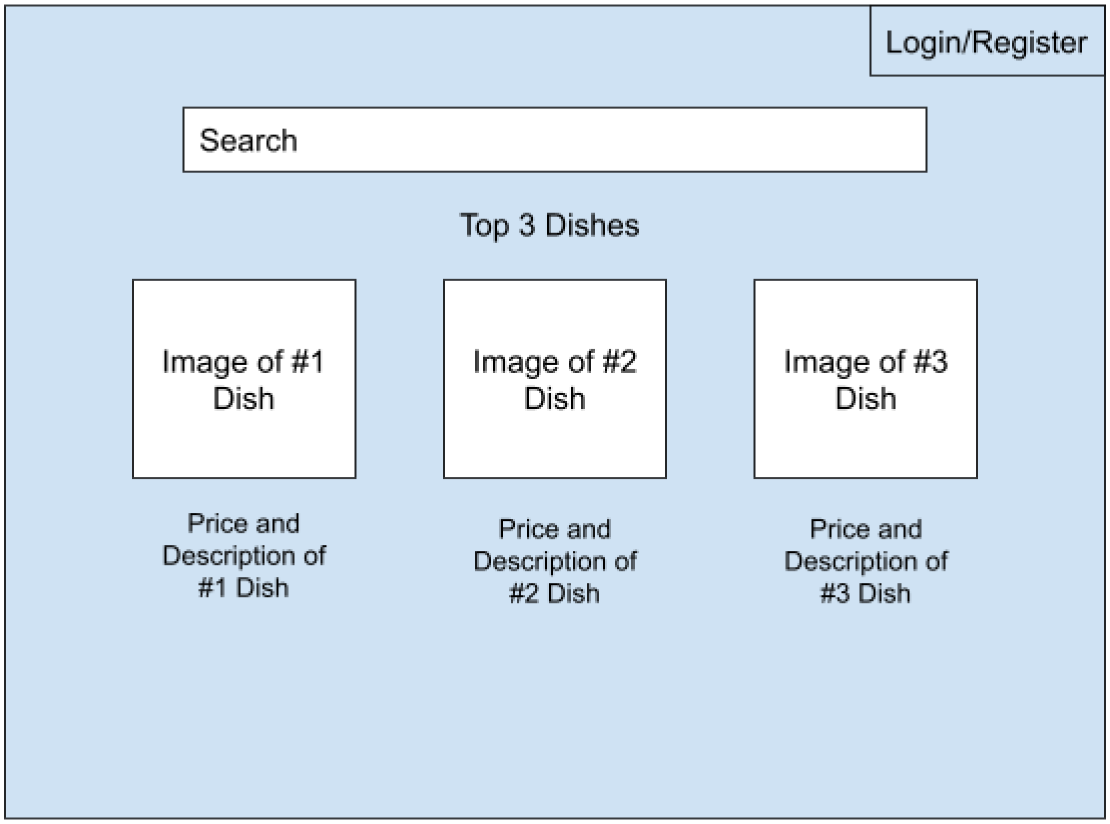
Inputbar s;

Vector <int> recommendations;

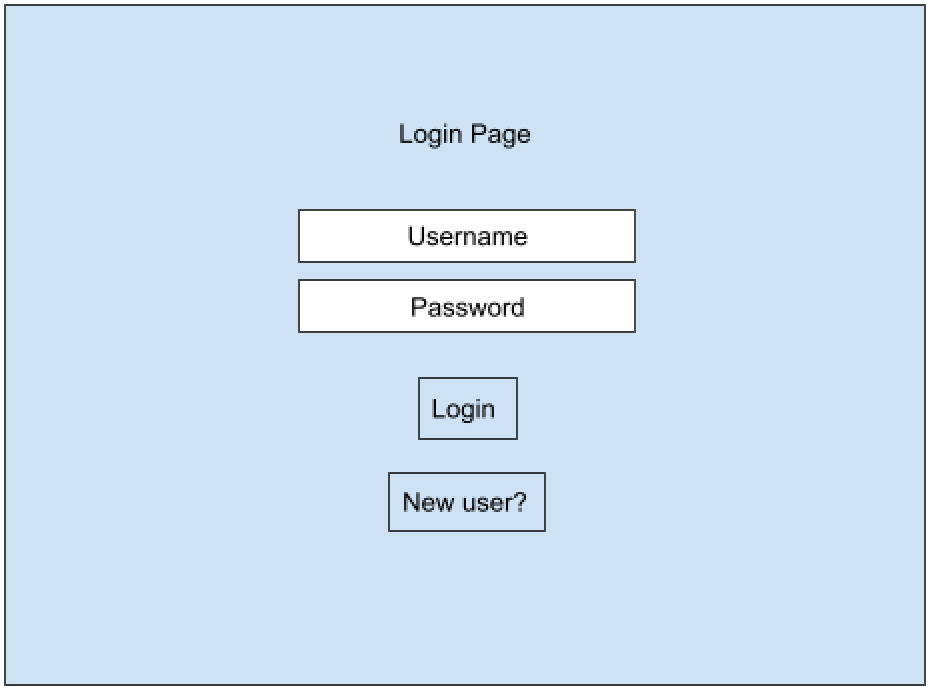
login.callback(loginInterface()) – leads to other interfaces depending on the userID returned from the login function

5. **System screens:**

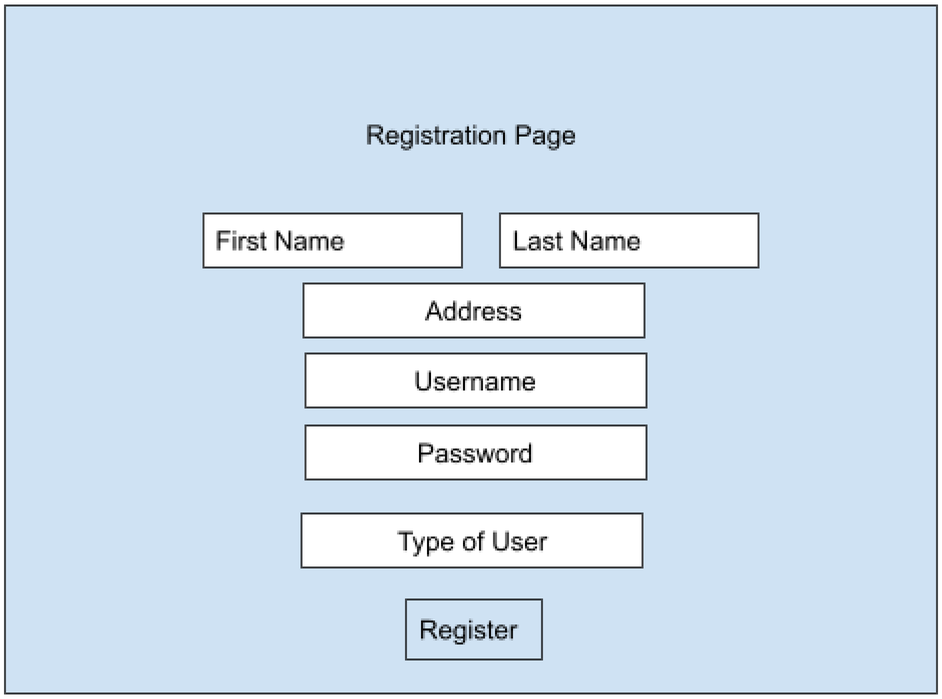
demonstrate major GUI screens of the system and a sample prototype of one functionality of your own choice.


Login/Register
Search
Top 3 Dishes
Image of #1 Dish
Image of #2 Dish
Image of #3 Dish
Price and Description of #1 Dish
Price and Description of #2 Dish
Price and Description of #3 Dish

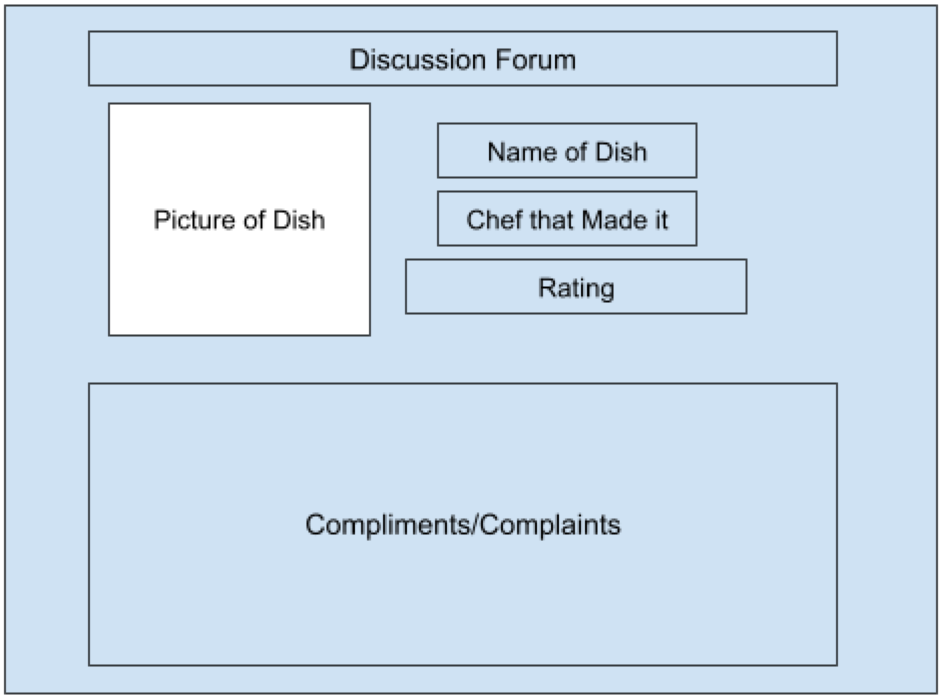
This is the first page that is seen by the user. For visitors and new customers, they will see the top 3 rated dishes and top 3 popular dishes. They will have the option to navigate to the login/register page from this page as well. For users other than visitors and new customers, they will instead have a logout option and their name and role shown. The top 3 dishes shown to them will be unique to them specifically. There will also be an option that allows the user to check their account information. If they are a customer, there will also be an option to deposit money.



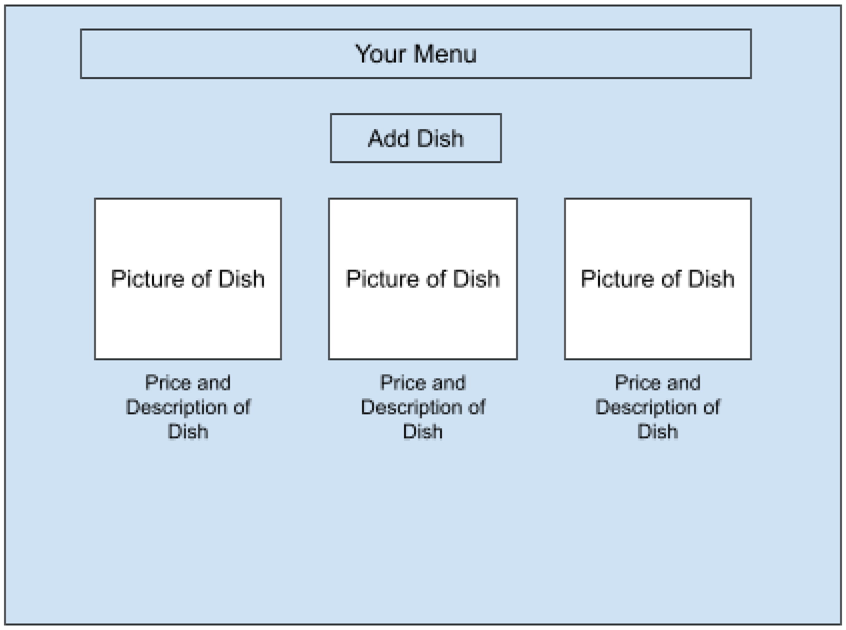
The login page allows a registered user to login using the username and password that they registered with. If the user does not have an account then they will have to register.



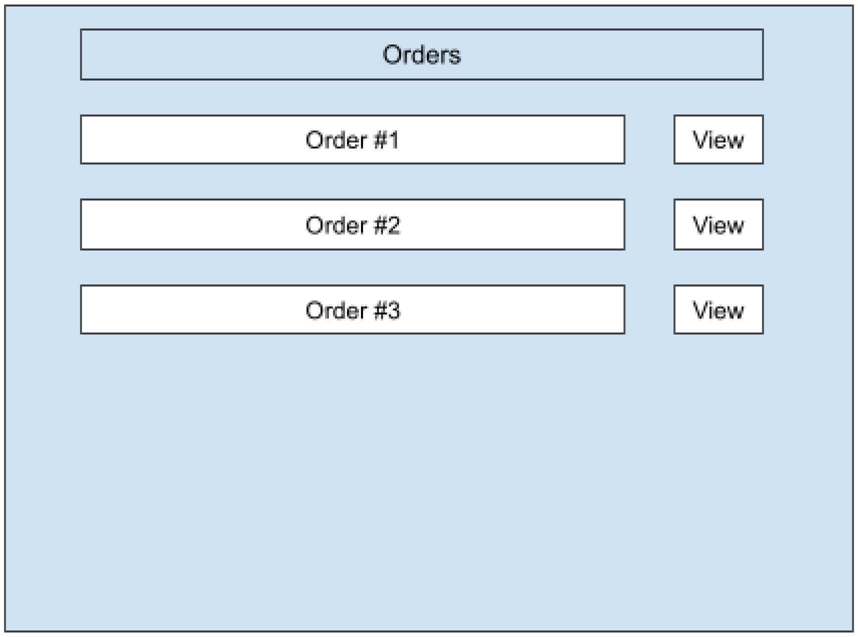
If the user is new, then they are allowed to register. They are allowed to create their own unique username and password. They are asked for their information and also about what role they have between customer, chef, and delivery person. Their actions and responsibilities are dependent on the role they select. They must be approved by the manager before the user can actually login. If the information matches someone on the blacklist, then they are not allowed to register.



Customers may write compliments or complaints about a certain dish in the discussion forum. The discussion forum will be similar for the customers and delivery people which the delivery people and the customers respectively can complain or compliment about as well. It will have information about the customer instead of the dish. For the managers, they will have the power to manage the compliments and complaints on the forum.


Your Menu
Add Dish
Picture of Dish
Picture of Dish
Picture of Dish
Price and Description of Dish
Price and Description of Dish
Price and Description of Dish

For chefs, instead of seeing the popular dishes, they will see the dishes they have made. Here, they will be given the option to add more dishes. There will also be the option to remove a specific dish when they select a dish from their menu.



For delivery people, they will get to see the current orders. They can select the order to see more details on it and also bid on it to deliver the order. Once the delivery person wins the bid for the order, they won't be able to bid for another order until they complete the delivery for that order.

6. **Memos of group meetings and possible concerns of team work**

We've had group meetings and we discussed what we were working on and if we have any updates to share. If there were any questions then we would help each other out and find a solution by brainstorming together. We also keep each other updated on new findings on what we are working on and on what we are planning to work on as well. There are no possible concerns of team work as of now.

7. **Address of the git repo (github, gitlab, bitbucket, etc) of your team's work so far - put all materials including this report there**

   https://github.com/StefanTan18/No-Pork-Kitchen