Streamline Dining

https://github.com/karneetarora/SE-Group-4

Submitted on 12/14/2020

Group 4:

Karneet Arora [ksa66]

Pablo Hernandez [pmh101]

Justice Jubilee [jlj173]

Harman Kailey [hsk63]

Srinu Koritela [sk2093]

Talya Kornbluth [tk453]

Max Lightman [ms2789]

Joel Usita [jbu3]

Eugene Langmer [ekl49]

Table of Contents

Contents

Individual Contributions Breakdown

Topic	Harman	Eugene	Max	Joel	Justice	Karneet	Pablo	Srinu	Talya
Individual Contributions	12.5%	-.01%	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%
Software Development	13.33%	-	6.67%	13.33%	13.33%	13.33%	16.67%	16.67%	13.33%
Debugging	12.5%	-	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%
Summary of Changes	12.5%	-	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%
Customer Problem Statement	30%	-	8.33%	8.33%	8.33%	20%	8.33%	8.33%	8.33%
Glossary of Terms	16%	-	20%	-	16%	16%	16%	16%	-
Functional Requirements	30%	-	10%	-	10%	10%	20%	10%	10%
Nonfunctional Requirements	10%	-	-	-	-	30%	30%	-	30%
User Interface Requirements	-	-	10%	10%	20%	20%	20%	-	20%
Effort Estimation	10%	-	-	-	-	30%	30%	-	30%
System Sequence Diagrams	50%	-	6.25%	6.25%	6.25%	12.5%	6.25%	6.25%	6.25%
Use Case Diagrams	8.75%	-	8.75%	38.75%	8.75%	8.75%	8.75%	8.75%	8.75%
Fully Dressed Description	-	-	-	-	-	40%	-	30%	30%
Traceability Matrix	6.25%		6.25%	50%	6.25%	6.25%	6.25%	6.25%	12.5%
Domain Analysis	-	-	-	-	-	-	50%	30%	20%
System Operation Contracts	6.25%	-	6.25%	6.25%	6.25%	6.25%	6.25%	28.125%	28.125%
Interaction Diagrams	5%	-	5%	5%	5%	5%	40%	35%	5%
Class Diagram and Interface Specification		-	20%			30%	2.5%	2.5%	50%

4

System Architecture and System Design	16%	-	16%	16%	16%	-	16%	16%	4%
Hardware Requirements	50%	-	50%	-	-	-	-	-	-
Algorithms and Data Structures	40%	-	30%			30%	-	-	-
User Interface Design and Implementation	15%	-	15%	10%	-	10%	20%	20%	10%
Design of Tests	5%	-	5%	15%	15%	15%	15%	15%	5%
History of Work		-						80%	20%
References	11.11%	-	11.11%	11.11%	11.11%	11.11%	22.23%	11.11%	11.11%

Summary of Changes

- Created new format for proposal
- Completed system requirements
- Merged both parts, added pagination, organized content, expanded individual contributions, made a table of contents, corrected system sequence designs
- Explain the process of taking orders, explain what situations when customers can all food items at once, or a different time.
- Updated Interaction Diagrams to match with class diagrams.
- Added a section describing reward system
- Stated application is browser based, number of applications to be done is one with different interfaces.
- Updated the history of work
- Removed mentions of feedback system
- Updated the problem statement
- Updated Use Cases
- Updated Glossary of Terms
- Updated Traceability Matrices
- Updated User Interface
- Updated Use Case Diagram
- Updated System Requirements
- Spacing, pagination.
- Updated Data Structures
- History of Work
- Addition of UC for Hostess

I. Customer Statement of Requirements

1. Problem Statement

A restaurant is a complex business that has many different aspects that have to all work in conjunction for operation. Restaurant owners must ensure that each of these aspects and roles are able to work together efficiently and communication of information is vital. Currently many restaurants are still operating in an environment that is not too different that it was pre-digital age, coordinating by word of mouth or paper; this can often lead to communication errors amongst the staff members, which can be pinpointed to improper hearing or messy writing.

Beyond coordination issues, one can observe similar inefficiencies in the payment process which often results in customers unnecessarily waiting additional time just for their server to process their transaction. This allotted time could reduce the waiting time for another potential customer to be seated and served.

Our project aims to fix these problems as well as ones that are faced in more nuanced situations such as when there is interaction between different roles and actors in a restaurant environment. We are offering a web abased application that is compatible with not only on computers but most mobile devices. Our application will be loaded onto big screens, like tablets, and be placed in the front of the restaurant for hosts as well as in the kitchen for chefs to use, which will be specifically only for those types of employees. Other employees, such as servers, will be able to use their mobile devices to use the application. It would be up to the manager's discretion as to what device they wish to use to access the application.

a.) Managers

The manager has the most responsibilities in a restaurant and must be up to date information about all operations.

Employee Management

- There are many different employees that work under a manager, so management must be able to keep track of employee schedules, supervise payroll, log hours worked by employees, log their attendance, and their performance. In the event someone is unable to make their shift, a manager should be able to view all the schedules of the employees, so they may be able to find a replacement. In addition, a manager should be able to add new employees into the system to account for recent hires. The manager should be able to see which employee is working on which order and which servers are attending to which table.

Inventory Management

- Keeping track of items can be a cumbersome task, especially if everything is not already itemized and accounted for. There are so many different units that are needed in a restaurant that it is very easy to lose track of inventory as well as which items need to be replenished. Therefore, managers as well as chefs should have access to real time inventory levels as well as an interactive list of which items need to be ordered.

Order Management

- Oftentimes if a customer has an issue with their order, they ask for the manager. So, a manager must have access to all orders, completed orders and those being processed, so they can seamlessly address any issues or complaints a customer may have. The manager also needs an easy way to track which employees are tied to which order/table. Food orders will link directly to customers. Whether order is toggled for table or takeout will decision the destination of the meal, which will be accessed by customer class.

b.) Chefs

Chefs play the most vital role, as they are responsible for producing good quality products and need the expertise to add or remove ingredients depending on the customers' needs. It is crucial for the chef to be aware of any allergies or omissions a certain table may have, so they can accommodate it while prepping the meal. They must also accommodate for certain requests, I.e., when a customer orders a steak, they may specify how well done they want the meat. If a server's handwriting is messy or they didn't hear the customer correctly, it could cause potentially fatal issues with the customer. The chef should be able to see the customer's order online to minimize mistakes. When an order is ready to be taken to the host for takeout or to the customer table, the chef should be able to communicate the order status to the server. To prepare meals for customers, chefs need ingredients; so, they should have access to a real time list which displays the inventory level of the ingredients as well as be able to update the list. Chefs will have a tablet in the kitchen, as tablets have bigger screen than most mobile devices making it easier to view, with the application pre-loaded for them to use.

c.) Servers

Servers are responsible for more than one table at a time, so they need to be quick in order to be able to tend to guests. Every server should have access to the menu and should be able to select the items to place the order directly. A server should have real time information about the tables they are serving, the orders that were placed, and the progress of the orders to ensure efficiency. To expedite the transaction process, servers should be able to take payment on the go. For takeout orders, they should be able to pull up a specific order, check the status of the order, and complete it.

d.) Customers

A customer desires to enter a restaurant, be seated, have their order taken, receive their food, and pay for their meal as quickly and seamlessly as possible. Not all customers are dine-in orders, so they should be able to order takeout or delivery without having to call. There are many times when staff is occupied with a busy day and phone calls require long waiting times and lead to poor a poor customer experience. An application can help to mend some of these issues.

Sometimes a customer is forced to wait for a long time before being given a table. There should be a way for customers to check their device to see if the store has any open seating available as well as a reservation system. They should be able to get estimated wait times in case there is a queue for tables. A customer should also be able to order items as soon as they sit down rather than waiting for a server to come by and take down the desired items. There should also be a way of viewing and editing menu items since some customers are allergic to certain ingredients.

After customers are done with their meal, they are forced to stay seated until the server brings the check, and the payment is processed. This excessive time can be saved if customers were able to pay by themselves virtually. That’s why with our application, a customer can checkout using the application besides having to wait for their server to settle the bill. If the customer does not want to perform self-checkout using the application, they would wait for a server who would use the application to settle the bill.

e.) Hosts

The hosts should be able to track the occupancy status of all the tables. They are the first point of contact for the customers, so the host must decide where to seat customers whether they have made previous reservations or not. They need to be informed of the estimated wait times and are the point of contact for takeout/delivery orders since they are the closest to the entrance. Hosts should also encourage customers to sign up for the restaurant’s application for a more seamless experience and rewards points. The host should update the order interface accordingly if a customer has the application or not. Customers with the application should be able to check-in with the host at the entrance.

II. Glossary of Terms

Actors – Human beings that interact with our system.

Customers – A type of user role in the system.

Customer Portal – This is a log in page that will be used by customers to log into their specific configuration of our application.

Employee – A type of user role in the system.

Employee Portal – This is a log in page that will be used by Employees to log into their specific configuration of our application.

Employee Management – A database of active and discharged employees and all the accompanying properties.

Inventory Management – A live database of thew companies' inventory. This concept also contains automation that will intelligently order new supplies when the current supply level is below a certain threshold.

Employee - A user in the system.

Order – An item containing information about food that is to be prepared to a customer.

Order Management- A database of live and completed orders that is visible to servers and managers.

Chefs – A user role in the system.

Host – is a person who greets and organizes customers, and assigns tables based on the restaurants policy.

Schedule – A chart of containing information regarding which employees are working on what days.

Servers – A user role in the system. Takes orders and deliver food to customers.

An order– An item containing information regarding table number, contents of the order, the creator of the order, and special notes such as customizations and allergens.

Inventory - Will have a running record of raw ingredients as well as keep track of utensils and other resources. Also, notifies manager and supplier when resources are running low.

Floor Map – A list of tables, capacity, and availability.

Profile – A status that a user has that will tell the system what information to give said user access to.

Payment – record of a financial transaction.

Wage – The hourly pay of each employee

11

III. System Requirements

Priority 5 highest priority

Priority 1 lowest priority

1. Enumerated Functional Requirements

Identifier	Priority	Requirement
REQ-1	5	The application will allow employees to log in to their own portal.
REQ-2	3	The application will allow employees to check their schedule, wage, and other logistical information.
REQ-3	5	The application will allow employees to look at ongoing order information.
REQ-4	4	The application will allow managers to add new employees.
REQ-5	4	The application will allow managers to add new menu items.
REQ-6	5	The application will allow managers to access inventory information and make orders.
REQ-7	4	The application will allow chefs to see order information.
REQ-8	5	The application will allow customers to log in to their own portal.
REQ-9	4	The application will allow customers to reserve a table when available.
REQ-11	5	The application will allow customers to decide between takeout, dine-in, and delivery along with setting a time for takeout and delivery.
REQ-12	3	The application will let servers know if customers need assistance.
REQ-13	3	The application will let servers place orders and write any additional comments.
REQ-14	3	The application will let servers know to check on costumers periodically.
REQ-15	5	Employees should be able to process transactions via application.
REQ-16	5	The application will allow the customer to order food through the application for either take-out or dine in
REQ-17	4	Application will forward a copy of the receipt to email address provided
REQ-18	3	Manager should be able to survey when employees are available
REQ-19	2	The application will use the data collected from the user rewards reviews and graphically represent numerical values as well as make the reviews easy to access for the manager and specified employees.
REQ-20	4	The application will allow hosts to see which tables are available, let hosts reserve tables, and let hosts see table reservations.

2. Enumerated Nonfunctional Requirements

Identifier	Priority	Requirement
NREQ-1	5	Customers are not able to update the menu
NREQ-2	5	Employees are not able to alter salaries
NREQ-3	2	The website should be able to handle 200 users at once
NREQ-4	3	Application must run IOS and Android

NREQ-5	4	Application must have a backup in database for important stagnant data and restaurant documents in case a failure occurs
NREQ-6	5	Employees are not able to change the schedule
NREQ -7	5	Employees are not able to update the menu
NREQ -8	4	Employees are not able to place orders for inventory

3. User Interface Requirements

Identifier	Priority	Requirement
IREQ-1	5	Must be easy to navigate through app
IREQ-2	5	Should be easy to backtrack to home page
IREQ-3	3	Display ingredients
IREQ-4	5	Display total with option to add tip
IREQ-5	3	Have menu visible with tabs above to jump to other categories
IREQ-6	2	Customers have option to search menu for specific item/ingredient
IERQ-7	2	Should display an average time of food preparation.
IREQ-8	4	Display different payment methods
IREQ-9	2	Allow menu sorting for customer preferences

13

IV. Functional Requirements Specifications

1. Stakeholders:

a. Restaurant Owners:

- Will help optimize their business's traffic and efficiency of employees

b. Restaurant Managers:

- Will be easier to manage the employees and other information required to manage a restaurant

d. Restaurant Customers

- Dining experience will be more efficient with less snags; simplified service means talking to less middlemen and getting the experience they desire

2. Actors and Goals:

Actor	Role	Goal
Manager	An employee that is responsible for daily restaurant management operations. This includes talking to customers to deal with complaints, checking worker's schedules to make sure there are enough employees and no over booking, and to deal with suppliers.	Log into the system. Log out of the system. Schedule employees. Access employee clock in. Access employee clock out. Access to stock level. Approve stock orders. Order maintenance works. Pay employees. Pay suppliers.
Chefs	An employee that reads the orders from the servers and creates the dishes ordered, including all of the omissions/allergies.	Log into the system. Log out of the system. Clock in for shift. Clock out of shift. See what menu item was ordered. Notify server when food is ready. Order stock. Notify manager about needed maintenance.
Servers	An employee that provides direct customer service to the customers. This includes taking orders, answering questions about the menu, and taking payments from customers.	Log into the system. Log out of the system. Clock in for shift. Clock out of shift. Input orders from customers.

		Know when an order is ready. Accept payment from customers.
Customers	Patrons of the restaurant. Will have the same application as the restaurant staff but will have access to limited functionalities and have many features hidden.	Log into the system. Log out of the system. Make reservations. Access menu items. Put in an order. Process payment through system. Know when a delivery driver arrives.
Hosts	An employee that is responsible for seating incoming customers to a table based on availability. When online reservations are made, the host is responsible for ensuring that the table is available for the reserved time.	Log into the system. Log out of the system. Clock in for shift. Clock out of shift. Access to online reservations. See which tables are empty. Assign customers to empty tables. Notify servers about customers arrival. Notify Busboy when customers are done.

3. Use Cases:

a. Casual Description:

UC – 1: Ordering: Allows an employee or a customer to place an order.

Derived From: REQ#5. REQ#13, REQ#16

UC – 2: Reservations: Allows a customer and host to reserve a table ahead of time for a certain time frame using the application for their party.

Derived From: REQ# 9, REQ #20

UC – 3: Payment: Allows the customer to pay through the application or though their server.

Derived From: REQ#15

UC – 4: Take-out: The customer can use the application to place an order for take-out for a scheduled time or for as soon as possible.

Derived From: REQ# 11

UC – 5: Food omissions: Allows customers to modify recipes based on their dietary preference or restrictions.

Derived From: IREQ# 6, IREQ#9

UC – 6: Food filtering: Allows customers to sort the menu based on ingredients/type of food they are looking for

Derived From: REQ#, IREQ# 3, IREQ#6

UC – 7: Clocking in/out: Allows employees to punch in and punch out of their shift using the application on the premises.

Derived From: REQ# 1

UC – 8: Login: Allow all users to log in from the standard portal

Derived From: REQ# 5, REQ# 8

UC – 9: Schedule Employees: Allows the manager to schedule employees through the app and helps minimize over and under scheduling

Derived From: REQ# 2

UC – 10: Inventory Management: Allows the manager or supplier to add items into inventory such as ingredients, restaurant ware, etc.

Derived From: REQ# 6, NREQ# 9

UC – 11: Payroll: Allow the manager to manage payrolls for employees.

Derived From: REQ# 1

UC – 12: Menu modification: Allow managers and chiefs to modify the menu based on what ingredients are available.

Derived From: REQ# 13

UC – 13: Ordering ingredients: Allow the chef to put in an order for more ingredients

Derived From: REQ# 6, REQ# 7,

UC – 14: Purchasing ingredients: Allow the manager to approve the order created in UC 12.

Derived From: REQ# 6, NREQ# 9

UC – 15: Seating Chart: Allows the host to view the real time seating chart and assign customers to a table.

Derived From: REQ# 20

b. Use Case Diagram

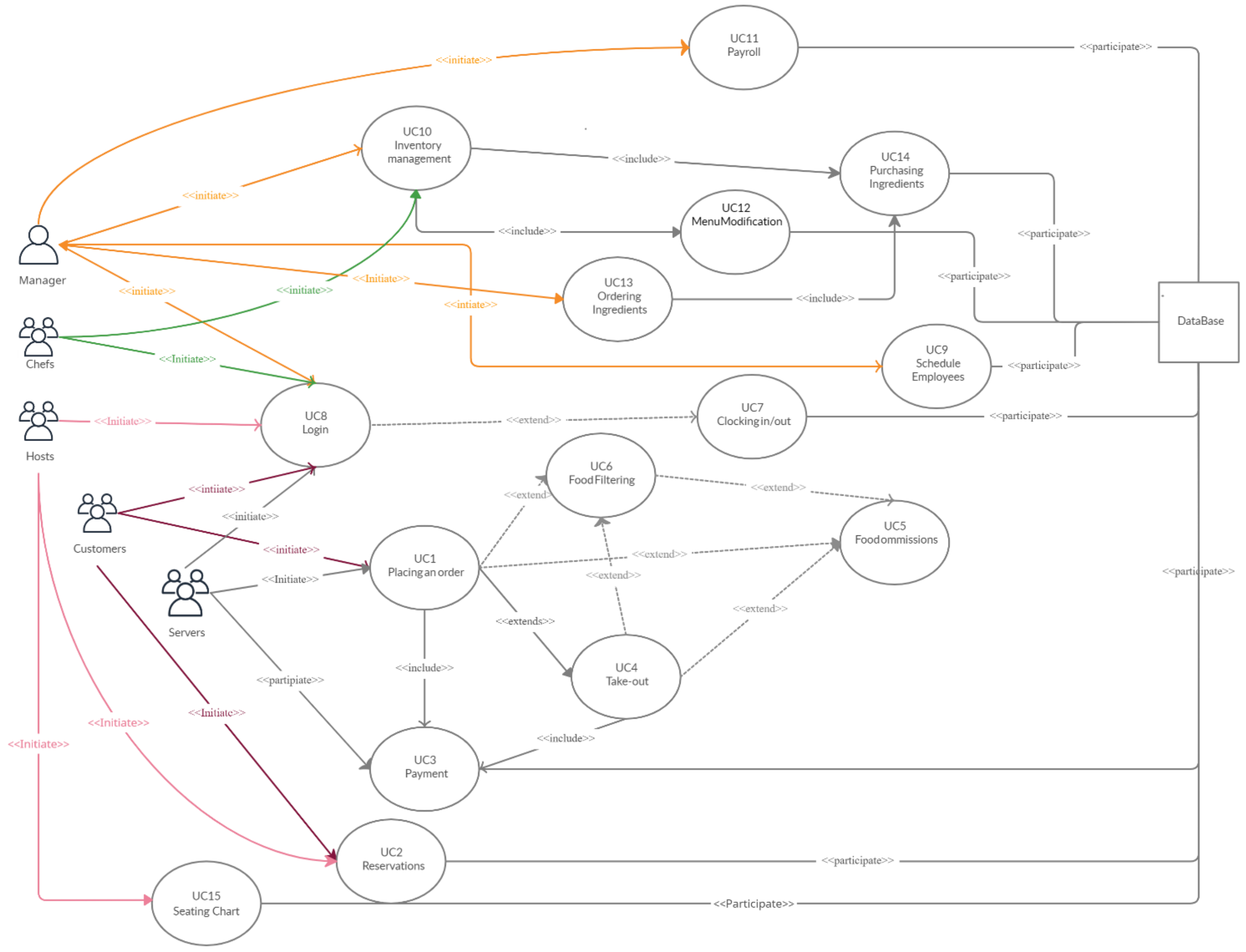


Figure 1

17

c. Traceability Matrix

Identifiers	P	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8	UC-9	UC-10	UC-11	UC-12	UC-13	UC-14	UC-15
REQ-1	5							X				X				
REQ -2	3									X						
REQ -3	5															
REQ-4	4															
REQ-5	4	X							X							
REQ-6	5										X			X	X	
REQ-7	4													X		
REQ-8	5								X							
REQ-9	4		X								X				X	
REQ-10	2															
REQ-11	5				X											
REQ-12	3															
REQ-13	3	X		X									X			
REQ-14	3															
REQ-15	5															
REQ-16	5	X														
REQ-17	4															
REQ-18	3											X				
REQ - 19	2															
REQ - 20	4		X													X
NREQ-1	5															
NREQ-2	5															
NREQ-3	2															
NREQ-4	3															
NREQ-5	4															
NREQ-6	5															
NREQ-7	5															
NREQ-8	4															
IREQ-1	5															
IREQ-2	5															
IREQ-3	3						X									
IREQ-4	5															
IREQ-5	3															
IREQ-6	2					X	X									
IREQ-7	2															
IREQ-8	4															
IREQ-9	2					X										

18

4. Fully Dressed Description:

FDD: Ordering Dine-In

Related Requirements: REQ#5, REQ#8 REQ#13, REQ#16

Actor:

- Customer

Participating Actor:

- Server
- Database

Preconditions:

- The application is loaded and ready to be used

Postconditions:

Flow of Events for Main Success Scenario:

1. → Customer will open the application
2. ← Application prompts the customer to select "Login" or "Guest"
3. → They will specify if dine-in or take-out and if dine-in, they will specify their table number
4. ← System will display menu.
5. → Customer will select the food for their party including any omissions
6. → They will detail any serious allergies that their party may have
7. ← They will submit their order and it will go to the kitchen
8. → Food items may be order at once or as the customer continues through their meal
9. → Once finished dining, server will ask for payment information
10. ← Server will submit payment information and get a receipt based on order.

OR

1. The server will complete these steps for the customer when they get to their table

FDD: Reservations

Related Requirements: REQ#1, REQ#8, REQ# 9

Actor:

- Customer
- Host

Participating Actor:

- Server
- Database

Preconditions:

- A customer is logged in

Postconditions:

- A reservation is created

Flow of Events for Main Success Scenario:

1. → Customer will select reservation option.
2. ← System will display available reservation times.
3. → Customer will select time and party size for reservation.

4. ← Customer will be asked to submit any additional information.
5. → Customer will submit any additional information.
6. ← Customer will be asked to confirm reservation.
7. → Customer will be asked to confirm reservation.
8. ← Reservation will be created with the given information and stored in the database.
9. ← Available reservation times will be updated.

OR

1. Customer will call the location and the host will do the following steps.

FDD: Payment

Related Requirements: REQ#1, REQ#8, REQ# 15

Actor:

- Customer
- Servers

Participating Actor:

- Server
- Database

Preconditions:

- A user is logged in.
- The order and respective bill are displayed.
- The user will input payment details.

Postconditions:

- Payment is confirmed.
- A receipt is provided.

Flow of Events:

1. → User will select the “Check/pay” option.
2. → User will insert credit card information.
3. ← User will receive receipt.
4. ← Order and payment details are archived.

OR

1. Server will do this for the user.

FDD: Clocking in/out

Related Requirements: REQ#1, REQ#2

Actor:

- Employees

Participating Actor:

- Database

Preconditions:

- Employee is logged in

Postconditions:

- User is clocked in/out of their shift
- Database logs information for Manager

Flow of Events:

1. ← The system prompts employee to select their role and log in
2. → Employee logs in to application
3. ← System displays employee page to employee
4. → Employee selects “Clock-in” option
5. ← System tracks time of clock in
6. ← System checks employee’s location to ensure that employee is at restaurant
7. → When the employee is done working, employee will select “Clock-out” option
8. ← System tracks time of clock out
9. ← System checks employee’s location to ensure that employee is at restaurant

d. System Sequence Design

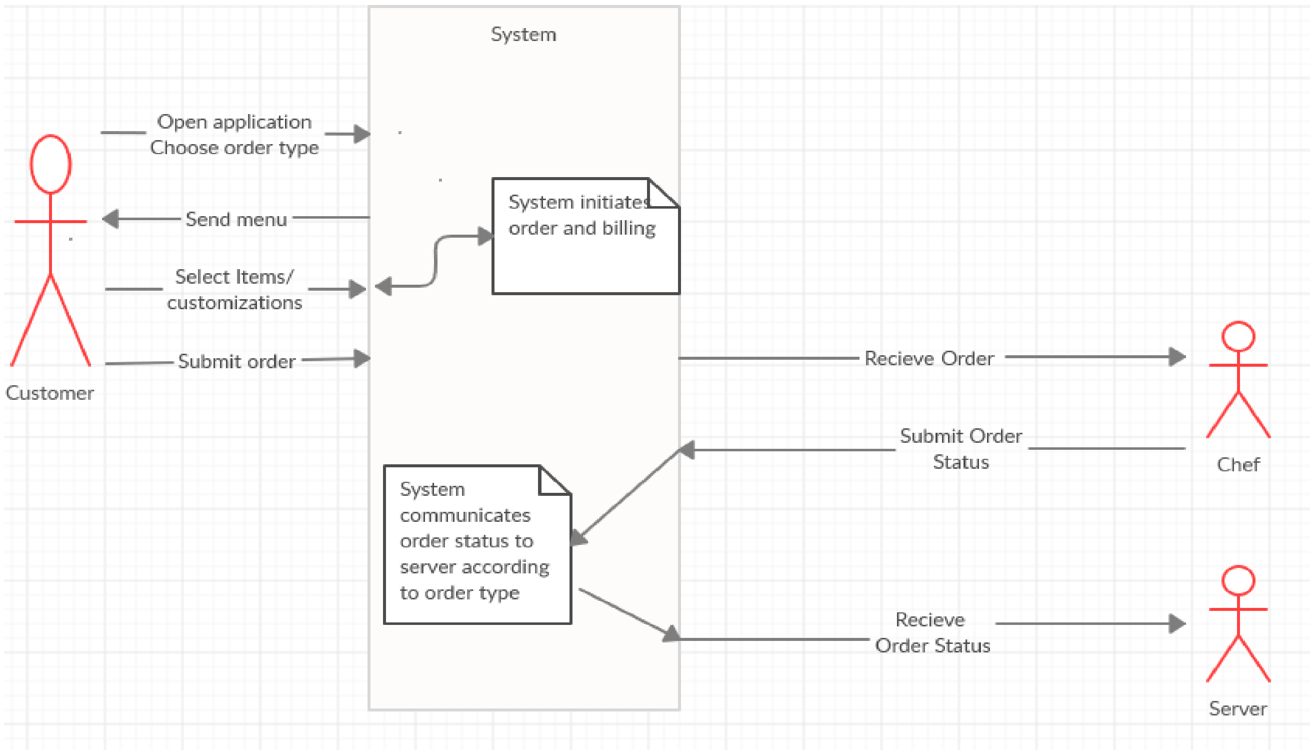


Figure 2: Placing an Order: From Customer to Server

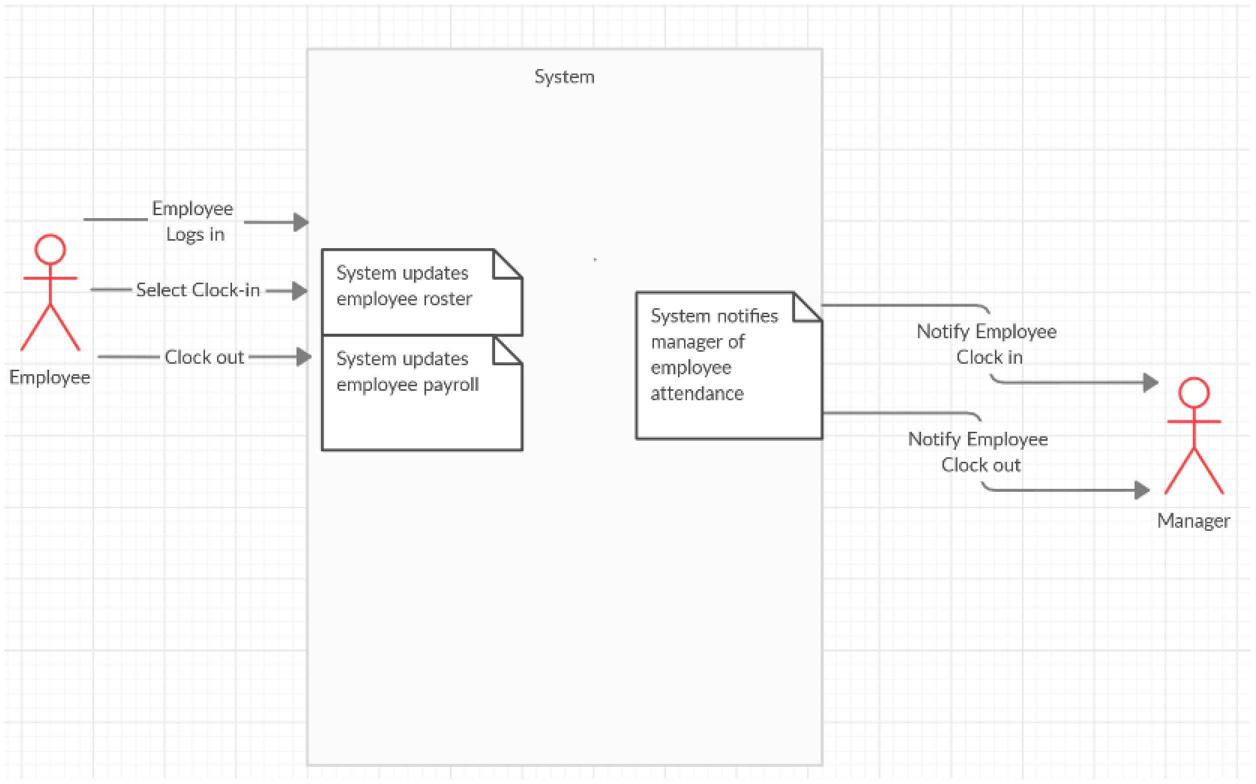

System
Employee Logs in
Select Clock-in
Clock out
Employee
System updates employee roster
System updates employee payroll
System notifies manager of employee attendance
Notify Employee Clock in
Notify Employee Clock out
Manager

Figure 3: Employee Clock In

V. Effort Estimation using Use Case Points

Customer Takeout Navigation: Total 3 mouse clicks. • Click “Customer Login”. • Enter Login Credentials. (Data Entry) • Click “Login”. • Click “Takeout”. • Will be prompted with a menu and will add desired items to a cart. (Data Entry) • Click "Place takeout”. • Select a time for pickup and enter payment method. (Data Entry) • Click “Confirm Order”. Data Entry: Varying amount of mouse clicks and keystrokes depending on many factors such as Login credentials length, Order length, and Payment information length. Approximately 20 inputs for Customer Login, 5 – 30 inputs for Order information, and 50 inputs for payment method if no previous payment information is saved. • Customer Login - Enter Username and Password. • Add to cart – Will click on and choose items from the menu and select any specifications. • Payment and Pick Up time – Need to enter Payment information and Pick up time.
Customer Delivery Navigation: Total 3 mouse clicks. • Click “Customer Login”. • Enter Login Credentials. (Data Entry) • Click “Login”. • Click “Delivery”. • Will be prompted with a menu and will add desired items to a cart. (Data Entry) • Click "Place Delivery”. • Enter payment method. (Data Entry) Data Entry: Varying amount of mouse clicks and keystrokes depending on many factors such as Login credentials length, Order length, and Payment information length. Approximately 20 inputs for Customer Login, 5 – 30 inputs for Order information, and 50 inputs for payment method if no previous payment information is saved. • Customer Login - Enter Username and Password. • Add to cart – Will click on and choose items from the menu and select any specifications.

- Payment and Pick Up time – Need to enter Payment information and Pick up time.

Customer Reservation
Navigation: Total 3 mouse clicks.

- Click “Customer Login”
- Enter login credentials. (Data Entry)
- Click “Login”
- Select time and party size for reservation as well as any special requests. (Data Entry)
- Click “Confirm Reservation”

Data Entry: Varying amount of mouse clicks and keystrokes depending on login credentials length. Approximately 20 inputs for Customer Login and about 4 inputs for reservation information if no special requests.

- Customer Login - Enter Username and Password
- Reservation Information – Click on a date and choose from one of the available times. Input reservation size and any special requests.

Employees Clocking in/out
Navigation: Total 3 mouse clicks.

- Employee will select “Login”
- Enter Login Credentials (Data Entry)
- Click “Login”
- Click “clock-in” or “clock-out”

Data Entry: Varying amount of mouse clicks and keystrokes depending on login credentials length. Approximately 20 inputs for Employee Login.

- Employee Login - Enter Username and Password

Hostess
Navigation: Total 3 mouse clicks.

- Employee will select “Login”
- Enter Login Credentials (Data Entry)
- Click “Login”
- Will have access to take-out, deliveries and reservations orders. Floor tables, check-out, Menu

Data Entry: Varying amount of mouse clicks and keystrokes depending on login credentials length. Approximately 20 inputs for Employee Login.

Waiter
Navigation: Total 3 mouse clicks.

- Employee will select “Login”
- Enter Login Credentials (Data Entry)

- Click “Login”
- Will have access to tables that are assign to employee, Menu, check-out, and place customers' order

Data Entry: Varying amount of mouse clicks and keystrokes depending on login credentials length. Approximately 20 inputs for Employee Login.

Fulfilling an order (Chef)

Navigation: Total 2 mouse clicks.

- Will already be on the orders screen which will display any pending orders.
- When order is completed, select appropriate order.
- Click “Order Ready”

Data Entry: None

Manager

Navigation: Total 3 mouse clicks.

- Will be on Manager screen which will display options such as schedule, employees, inventory, deliveries.
- Click “schedule” will display which employees are work and future dates
- Click “deliveries” will display upcoming deliveries
- Click “floor” will display the floor plan of all table at restaurant
- Click “menu” will display a menu where items can be added or remove
- Click “inventory” will display the logs for current inventory amounts

Data Entry: Varying amount of mouse clicks and keystrokes depending on login credentials length. Approximately 20 inputs for Employee Login.

VI. Domain Analysis

1. Domain Model

a. Concept Definitions

Responsibility	Type	Concept
R1: Verify username and password match	K	User Check
R2: Permission based data base accesses	K	DB Access
R3: Updates employees schedule, hours	D	Manager Profile
R4: Store customers information	D	Customer Profile
R5: Store customer rewards	D	Customer Profile
R6: Display order queue	D	Order Status
R7: Display take-out and delivery queue	D	Order Status
R8: Display available tables	D	Table Status

R9: Display status of tables	D	Table Status
R10: Display order to be serve	D OR K	Order Status
R11: Payment Process	D	Payment System
R12: Manage interaction with menu for Chef and Manager	K	Menu Alteration
R13: Display status of deliveries	K	Order Status
R14: Display status of take-outs	K	Order Status
R15: Tracks table's bill and other expenses	K	Payment System
R16: Tracks amount of ingredients	K	Statistics

b. Association Definitions

Concept Pair	Association Description	Association Name
User Check ↔ DB Connection	Checks for correct information to match that specific customers	Verify User
Customer Profile ↔ DB Connection	Obtains customer information	Get User Data
DB Connection ↔ Manager Profile	Allows manager to modify employee schedules, wages, adding employees.	Manager Actions
Payment System ↔ Order Status	Total will be charge base on the food order place and a calculate tip will also be available.	Get Bill
Statistics ↔ Order Status	A tracking system will be keep depending on the amount of food sold.	Food Data
Menu Alteration ↔ Manager Profile	Allows manager to add or remove items on menu.	Manager Actions
Statistics ↔ DB Connection	Inventory will be kept in the database	Inventory Data
Chef ↔ Server	Chef will be able to notify server of Order Status	Order Update
Manager ↔ Server	Manager will be able to assign tables to server	Manager Actions

c. Attribute Definitions

Concept	Attribute	Description
User Check	ProfileC	Will check that the username and password match
DB Connection	StoreData	Stores information
Manager Profile	EmployeeSchedule	Display which days an employee will work
	EmployeeHours	Display which hours an employee will work
	EmployeeWages	Display and edit wages
	AddEmployee	Create new Employee Profile
Customer Profile	AcccountUsername	Identifies username with customers' email
	AcccountPassword	Identifies password with customers username
	RewardPoints	Quantity of rewards on account
Order Status	DeliveryStatus	Shows user estimated time till delivery
	TakeOutStatus	Shows user estimated time for takeout order

	OrderStatus	Shows user estimated time till order is ready
Table Status	viewFloor	Display floor plan of restaurant with availability of tables
	viewTable	Displays table status of (in service, or open)
Payment System	TotalOrder	Calculates total amount due for a specific table
	PercentTip	Calculates 10%,15% and 20% of the total for tip.
Menu Alteration	AddItem	Increment of menu item
	DeleteItem	Remove of menu item
Statistics	UseItems	Keeps a total of the Food ingredients used
	NeedItems	Display ingredient that need to be order

d. Traceability matrix

	Domain Concepts								
Use Cases	User Check	DB Access	Manager Profile	Table Status	Payment System	Menu Alteration	Statistics	Order Status	Customer Profile
UC-1								X	
UC-2				X					
UC-3					X				
UC-4								X	
UC-5								X	
UC-6								X	
UC-7	X	X							
UC-8	X	X	X						X
UC-9		X	X						
UC-10		X					X		
UC-11		X	X						
UC-12						X			
UC-13							X		
UC-14			X						
UC-15		X		X					

2. System Operation Contracts

Operation	Ordering: Allows an employee or a customer to place an order.

Use Case	UC – 1
Preconditions	• The user is logged into their account or is logged in as a guest. • The user will fill in order details such as items, dine in or takeout, etc.
Postconditions	• The user will be taken to the receipt. • The user will be taken to the payment service. • Once order confirmed, order will be sent to the kitchen. • Customer can order all course meal at once but can also add additionally items as they continue through their meal.

Operation	Reservations
Use Case	UC – 2
Preconditions	• The user is logged into their account. • Will display available reservation times. • The user will fill in reservation details such as time, party size, etc. • User will confirm reservation.
Postconditions	• The reservation will be scheduled. • The user will have a reservation receipt. • Will update the reservation time availabilities.

Operation	Payment
Use Case	UC – 3
Preconditions	• The user is logged into their account • The user has placed and completed their order
Postconditions	• The user filled in their payment details • Once payment is accepted, the order will be archived • Receipt will be provided

Operation	Clocking in/out
Use Case	UC – 7
Preconditions	• Employee is logged in • Employee will choose either to clock in or clock out.
Postconditions	• User is clocked in/out of their shift • Database logs information for Manager

VII. Interaction Diagrams

Diagrams

FDD: Ordering Dine-in

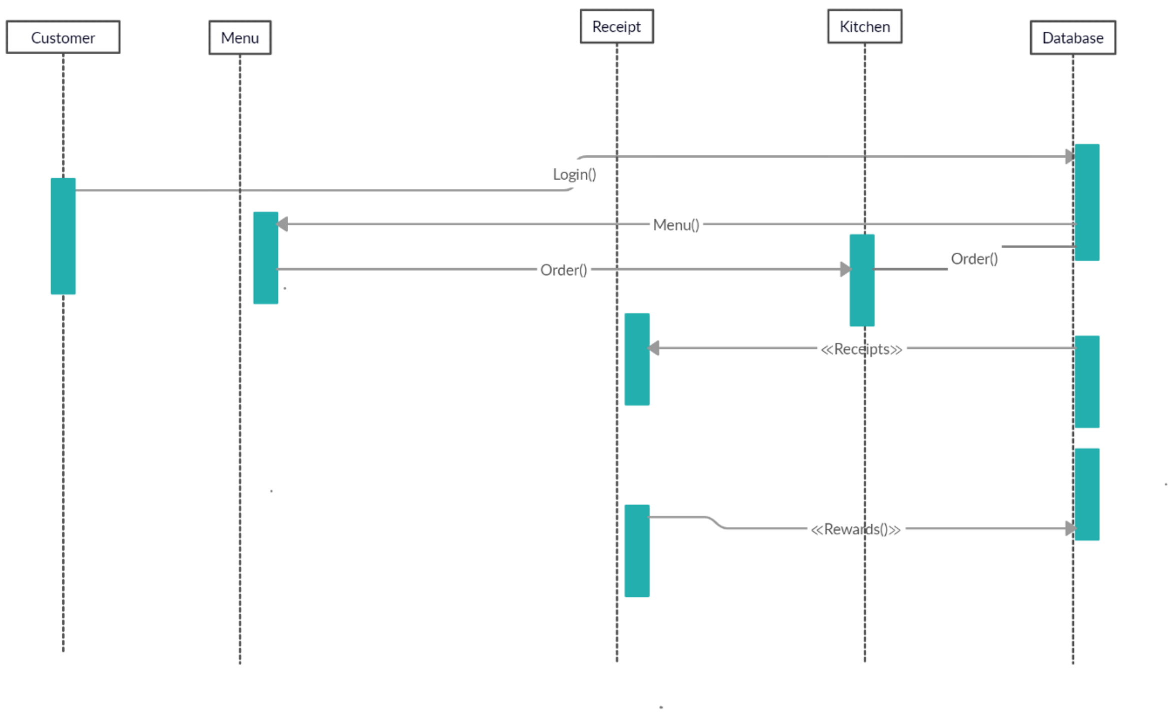


Figure 5

This interaction outlines how customers will be able to dine in. Either the server or the customer will log onto the application and select the dine in option. The system will then display the menu and then the customer will fill in all the order information. They will then confirm the order and the order will be stored in the database as well as be sent to the kitchen. Customer will order food items by selecting an item then a screen will be display where they will be able to type any dietary restrictions, or preferences, after the customer will confirm the order and will have the possibility of adding other items.

FDD: Reservation

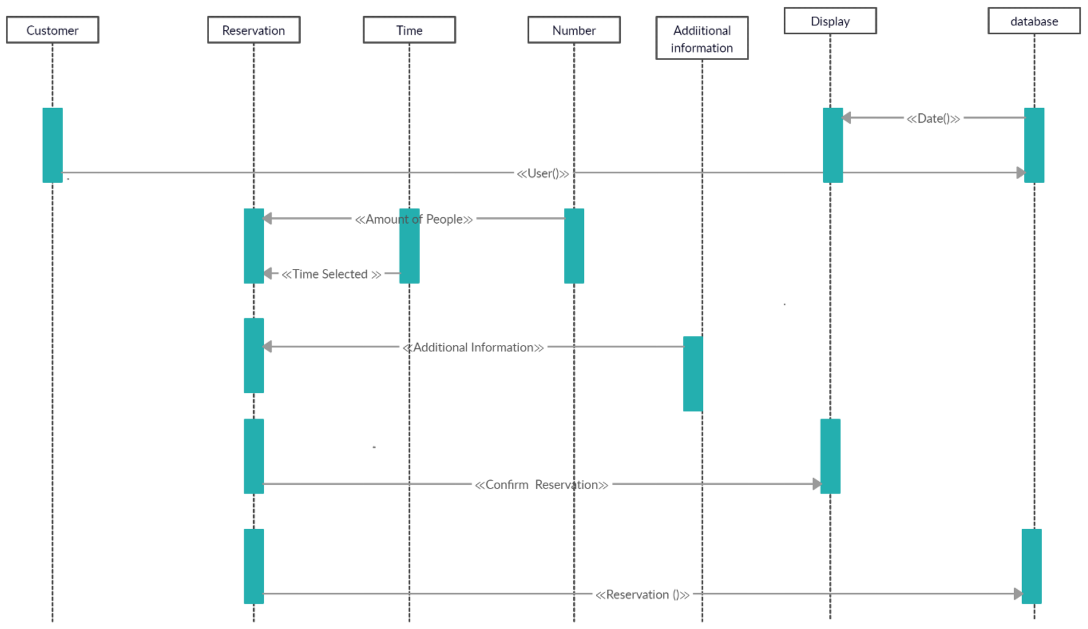


Figure 6

This interaction details how customers will be able to make online reservations. The customer must be logged into their account or create a new account. Once logged in, they will select the reservation option in order to reserve a table. The system will display available reservation times and the customer must select from one of the displayed times. The customer will also need to submit their party size along with any other additional information. The customer will then have to confirm their reservation and this reservation will then be sent to the database. The database will also update the available reservation times based on the reservation.

FDD: Payment

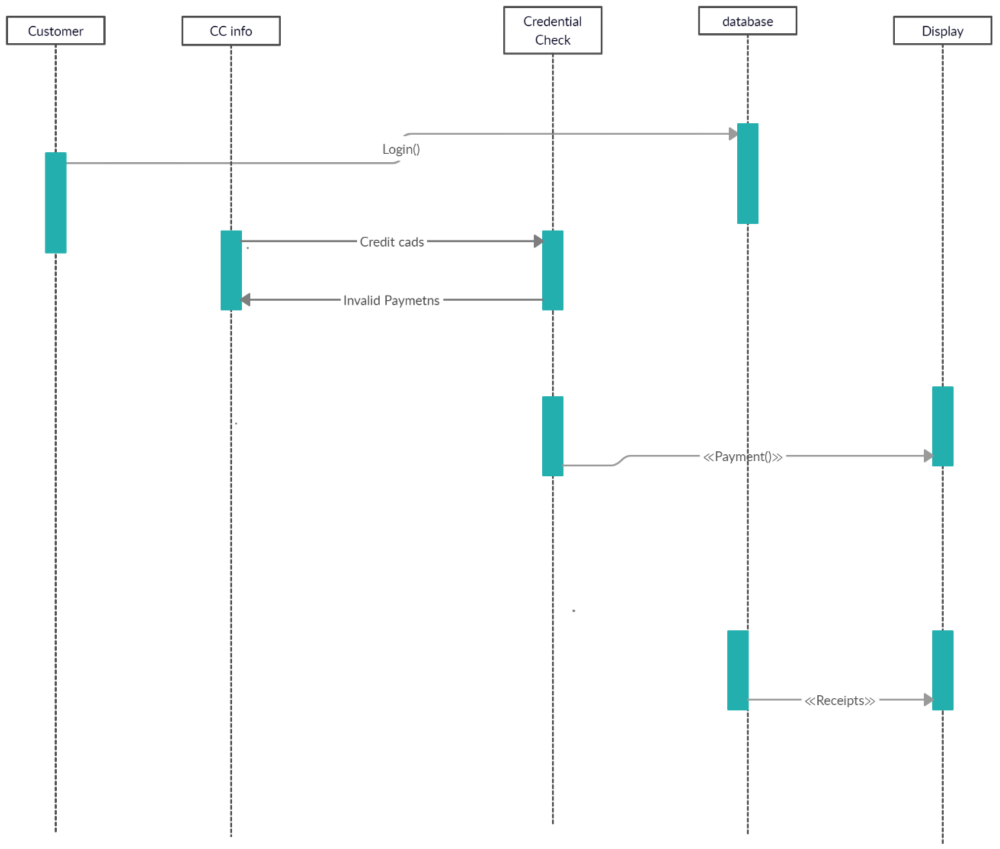


Figure 7

This interaction is on the payment process, A customer is log to their account, they enter their Credit card details, if the payment is not successful then it will return a message of invalid payment method, otherwise if the payment is successful then the information will be store, and a receipt will be display to the customer.

FDD: Clock in/out

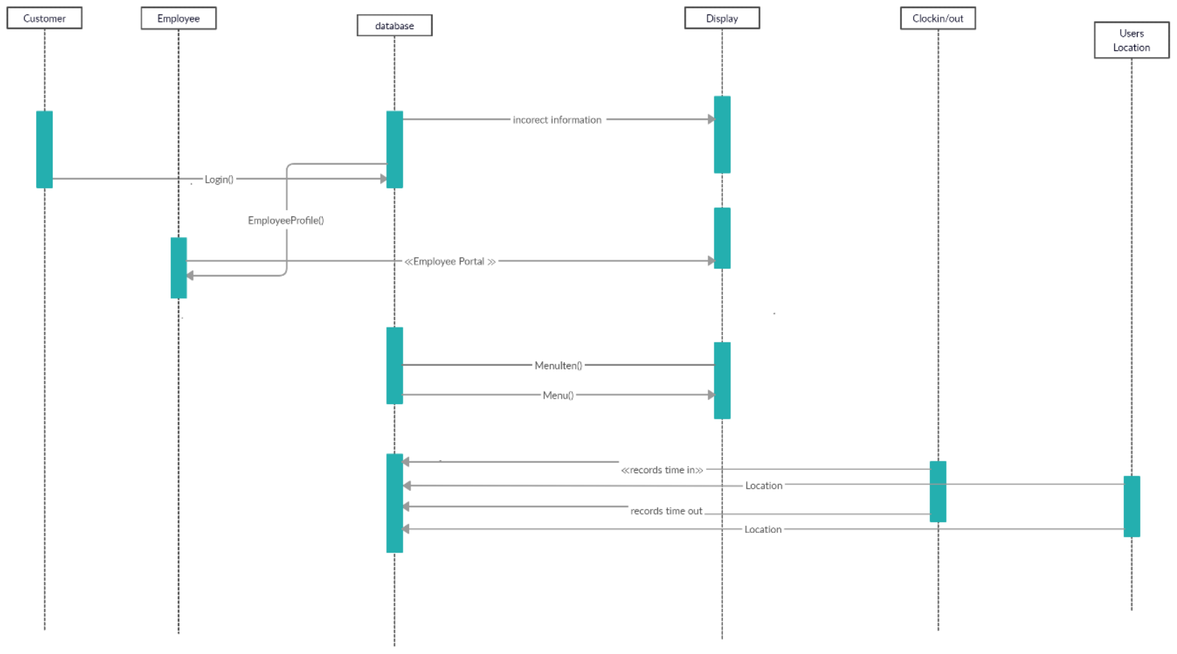


Figure 8

This interaction is the way employees' clock in or clock out hours. Employee's will have to log in to the system. In the case that they input the incorrect information; a message will be displayed letting them know their credentials are incorrect. If the employee has logged in correctly then the employee screen will be displayed, they will be able to clock in the hour which they are starting, the system will also check if the location of the employee is in the restaurant. At the end of the employee's shift, they will be able to clock out, again the system will check if the location of the employee is in the restaurant.

VIII. Class Diagram and Interface Specification

Class Diagrams:

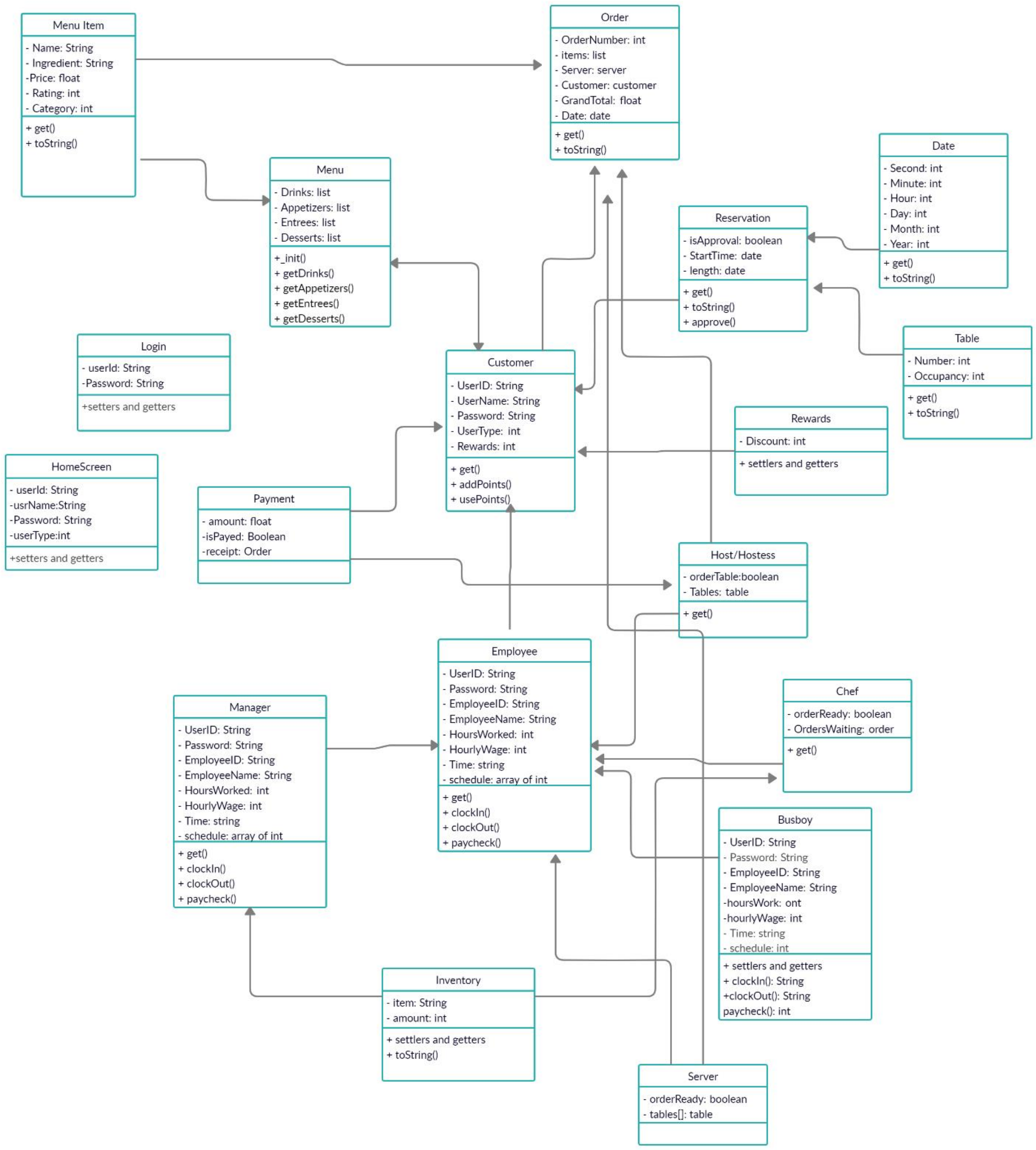


Figure 9

Data Types and Operation Signatures:

Class: Login

All Users
-UserID: String

-Password: String
+setters and getters

Class: Menu

Manager, Chief, Server, Customer, Host
– drinks: menuItem[]; list of the drinks available – appetizers: menuItem[]; list of the appetizers available – entrees: menuItem[]; list of the entrees available – desserts: menuItem[]; list of the desserts available
+__INIT__(String[] locationDrinksFile, String[] locationAppatizersFile, String[] locationEntressFile, String[] locationDessertsFile) +getDrinks(); retrieves all drinks on the menu +getAppetizers(); retrieves all appetizers on the menu +getEntrees(); retrieves all entrees on the menu +getDesserts(); retrieves all desserts on the menu +ToString()
On creation and a menu will be generated from 4 text files.

Class: menuItem

Manager, Chief
-name: String; name of the menu item -ingredients: String[]; list of ingredients in the menu item -price: float; price of the menu item -rating: int; number that other customers rated the item -category: int; category that the number is in
+getters for all items; retrieves all the attributes +ToString()
For category; 0 = drinks, 1 = appetizer, 2 = entrees, 3 = desserts

Class: Order

Server, Manager
-orderNumber: int; ID number of the order -items: menuItem[]; what items are in the order -server: Server; the server that is taking care of the order -customer: Customer; which customer the order is for -grandTotal: float; the total price of all the items in the order -date: Date; the date of the order
+getters for all items; retrieves all the attributes +ToString()
A guest account will be assigned to customers that are not logged in

Class: Date

Abstract class
__init__(int second, int minute, int hour, int day, int month, int year) +Second: int +Minute: int +Hour: int +Day: int +Month: int +Year: int
+Getters for all +ToString()

Class: TakeOut

Customer, Chef, Host
- Order: order; order that is being placed for take out -Customer: customer; customer that is ordering food for takeout -Payment: payment; payment method for the order
+setters and getters
Customer will have to order all items at once, otherwise food items will be take out at different times

Class: Table

Server, Busboy, Manager
-Number: int; table number -occupancy: int; number of people that can be seated at table
+setters and getters; retrieving attributes +orderNumber: int; retrieving order number for table

Class: Reservation

Host, Customer
-isApproval: Boolean; true if host/hostess approved reservation -startTime: date; what time the reservation starts -length: date; how long the reservation is for
+setters and getters; retrieving all attributes +toString() +approve(); host/hostess can approve reservation

Class: User

-userID: String; userID of user -userName: String; username of user -password: String; password of user -userType: int; type of user (customer or employee) -rewards: int; how many rewards the user has
+setters and getters; retrieving all attributes +addPoints(); add reward points to user +usePoints(); user can use reward points

Class: EmployeeProfile (type of user)

-userID: String; userID of employee

-password: String; password of employee -employeeID: String; id of employee -employeeName: String; name of employee -hoursWorked: int; how many hours employee worked -hourlyWage: int; how much employee gets paid per hour -time: String; the current time -schedule: int[]; list of time the employee is scheduled to work
+setters and getters +clockIn(): String; employee can clock in to work, returns time +clockOut(): String; employee can clock out of work, returns time +paycheck(): int; return how much employee is due to get paid
Extends to chef, busboy, server, host/hostess, manager

Class: Chef

-orderReady: boolean; true when order being worked on is ready -ordersWaiting: order[]; list of orders to make

Class: Server

-orderReady: boolean; true when order is ready -tables[]: Table; list of tables being served

Class: Host/Hostess

-openTable: boolean; true if there is an open table -reservations: Reservation[]; list of reservations - Order: order
Will have a display of takeout orders, with the profile information of the user.

Class: Manager

-userID: String; -password: String; -employeeID: String; -employeeName: String; -hoursWorked: int; -hourlyWage: int -time: String -schedule: int[]
+setters and getters +clockIn(): String +clockOut(): String +paycheck(): int

Class: Busboy

-userID: String -password: String -employeeID: String -employeeName: String -hoursWorked: int -hourlyWage: int -time: String -schedule: int[]
+setters and getters +clockIn(): String +clockOut(): String +paycheck(): int

Class: Payment

Server, Host, Manager
-amount: float; amount to be payed -isPayed: Boolean; true when transaction is complete

39

-receipt: Order; order is saved to receipt
Takeout payment will be done when order is place. Dine in payment will be done at the end of the customers meal

Class: Inventory

Chief, Host, Manager
-item: String; name of the item in inventory -amount: int; amount of item in inventory
+setters and getters +toString()

Class: Rewards

Manager, Customer, Host, Server
-Discount: int; amount of discount to be taken off of order
+setters and getters

Traceability Matrix:

	Domain Concept								
Class	User Check	DB Connection	Manager Profile	Customer Profile	Order Status	Table Status	Payment System	Menu Alteration	Statistics
Login	X								
Menu								X	
MenuItem								X	
Order		X			X		X		
TakeOut									
Table		X				X			
Reservation		X		X					
User		X	X	X			X		
EmployeeProfile			X						

Chef					X				
Server					X	X			
Host/Hostess									
Manager			X			X			
Busboy						X			
Payment							X		
Inventory									X
Rewards				X					

Domain Concept: User Check

Login: Allows user to log in based on their credentials

Domain Concept: DB Connection

Order: Orders are sent to the database once the customer completes the order

Table: Table availability/capacity will be stored in the database

Reservation: Reservation time/party size will be stored in database

User: Account information will be stored in database

Domain Concept: Manager Profile

User: Manager can update their own/their employees' account specifications

Manager: Managers are able to log on to portal and access the data they need

EmployeeProfile: Managers are able to edit employee info and schedule them

Domain Concept: Customer Profile

Reservation: Customers are able to make/view their reservations

User: Customers can update their account specifications

Rewards: Customers can view their reward points available and put current order towards reward points

Domain Concept: Order Status

Order: When an item gets ordered, its status is updated as it is created

41

Server: Server is updated on the status of their tables' orders

Chef: Chef updates the status of an order as it is created

Domain Concept: Table Status

Table: Layout and size of tables in the restaurant

Server: Server can view and edit status as customers come in/leave restaurant

Manager: Manager can view and edit status as customers come in/leave restaurant

Busboy: Busboy can view table status to see when the table needs clearing

Domain Concept: Payment System

Payment: User selects payment method and inputs information

User: User can choose to use saved payment methods and see previous transactions

Order: Calculates total payment owed based on food ordered

Domain Concept: Menu Alteration

Menu: Customer can view menu and the dietary warnings associated with them

MenuItem: Customers can view menu items' ingredients and add the items to order w/ various alterations

Domain Concept: Statistics

Inventory: User can view what items the restaurant currently has in the inventory and what items are still needed

IX. System Architecture and System Design

Architectural Styles:

Architectures styles are incorporated in Software to construct structures and systems which will become efficient at catching early problematic decisions on the system design and will allowed for a solution. Our design architecture will be composed of 2 Architectures Styles

One of the architectures styles we will use is Layered Style. By implementing this style, we are simplifying our design to be user friendly, since the consist of layers which are connected hierarchically it gives us the advantage of efficiently dividing work. Also, the ability to have levels that can change without affecting others, gives the freedom to manipulated and improve other layers independently. The use of layered Style is the structure of our Design.

Our second architectures style which our application will use is Client-Server Architecture Model. In this model client and servers communicate with each other through internet, both on them are on different hardware, clients are considered the staff of the restaurant and the customers. By using Client-Server

Model, the client will only have to initiate a request to use the services, then the server will provide services depending on the request of the client. In the case of many clients sending request to the server, the server relies on a scheduling system which will limit the number of requests a client can request.

Identifying Subsystems:

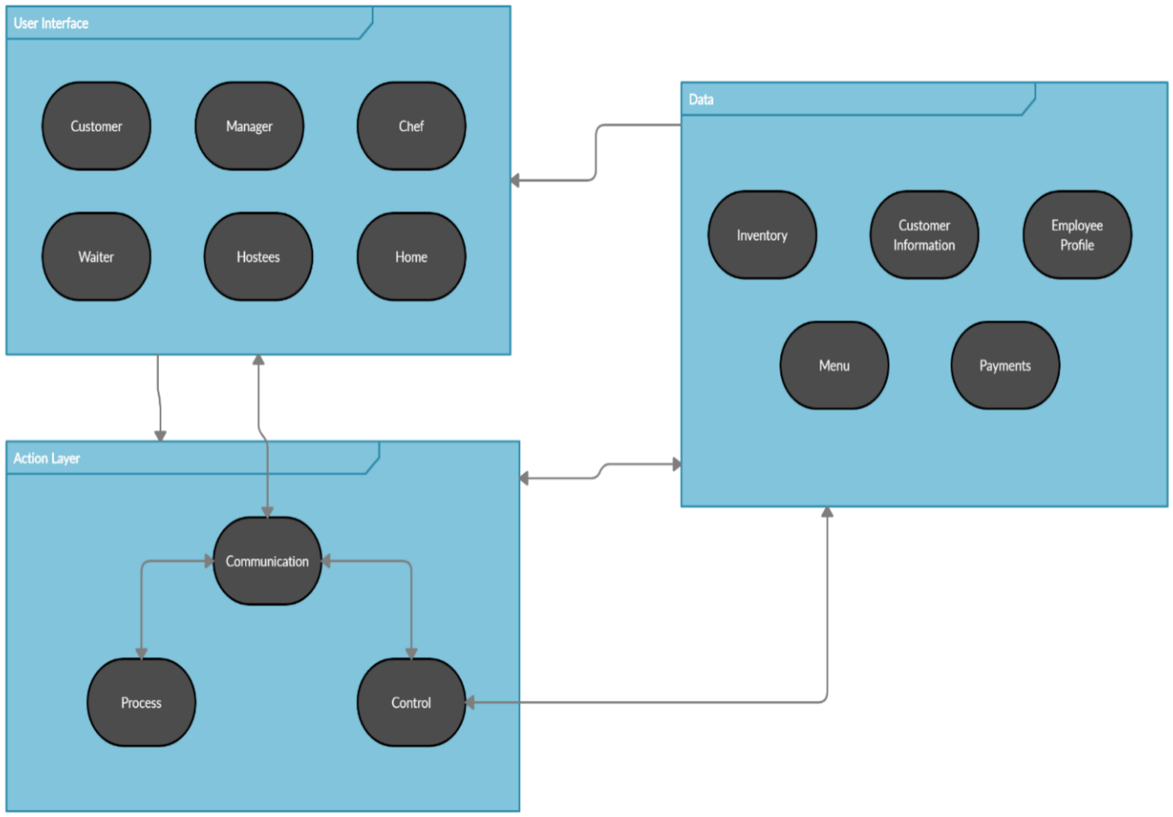


Figure 10

The system above is the subsystem of system, as we mention on the previous sections, we are implementing Layered Style, which is divided into 3 layers, User interface being the top layer, and Data being our lowest layer, each layer has also been decomposed into some layers to decrease the complexity, and our program runs smooth.

Our top layer is divided into 6 different subsystems which are the interfaces that our user will have display, the top layer is also connected to Data which is bottom layer this is because the Data layer's job is to communicate with the participating users, if the user needs anything to be display for example the menu, inventory or the employees profile the Data layer will have all the information to be display.

Now the middle layer which we are calling the Action layer also known as the brains of the application. While the other layers do logic, they tend to do generic logic while the Action layer does more or a specific logic, which gives control that each use case scenario is executed.

Even though all layers are connected to each other, and some of their subsystems are also connected, every layer can be evolved independently form the others. This a perfect scenario since you can test each layer with having the others implemented.

Mapping Subsystems to Hardware:

Our product will need to be run on multiple clients at the same time on any device with an established internet connection and a compatible web browser. Our application will use the Client-Server Architecture Model so the clients will be accessing the web server which will in turn send back the relevant web pages based on the client requests and the corresponding data from the database. The database will be stored on the MySQL server and will contain all the data regarding the restaurant. For the 3 different layers described above, the User Interface layer will be what the clients will be presented with and what the clients will interact with. The action layer will be the back end of the website and will control what is displayed to the user as well as process information from the user and input this information into the database. The Data layer will contain the database and will send and receive data from the clients.

Persistent Data Storage:

Data does need to be saved to outlive more than a single execution. Such examples would be an employee’s user ID, password, employee ID, name, hours worked and wages. For the customer side, the data that needs to be saved would be the customer’s user ID, username, and rewards as well. When an order is placed, the transaction along with the payment details need to be saved as well.

The storage management strategy that would be used is the Relational Model. This data strategy makes sense as each user (employee or customer) is linked to their own ID and password. The format of the tables will be in MySQL opposed to SQL. This will indeed help lower the cost to store needed data as MySQL is open source and free compared to SQL which is not free nor open source.

Network Protocol:

We will have all our users interact with our system from a web client. This means that almost all our networking will be over the internet. In order to ensure security all, the data would need to be encrypted asymmetrically to prevent anyone from intercepting the packets.

All our data will be stored on a cloud server. This server will be responsible for processing and storing input along with generating output. Users will connect to the server using HTTPS. We chose HTTPS because it is a secure way to allow universal access to the server and all the potential vulnerabilities will be in the design of the system, not the protocol for data transfer.

Because the sever contains mission critical information it would need to be constantly backed up onto a backup server that will remain off the network when not updating. We also would need to keep the server under lock and key because we will be storing user's private information and passwords.

Global Control Flow:

System will be built on an event-driven process. Users will have ability to generate or perform different actions in a dynamic order. Ordering of user events will not be linked directly to procedures of the system’s functionality. Dashboard interface will allow users to traverse through the different events of the applications by the users’ front-end actions. Object oriented framework will also allow for unique control flow based on the type of user that is accessing system.

Our system is strictly live event-response oriented. Therefore, at this moment in time no internal timers will be necessary for the overall functionality of our application. There will be live time bases posting within our clock-in and clock-out functionality for the employee end users, however this will be a live display and will not be a prerequisite for other events.

Hardware Requirements:

Our application is going to be designed to run on multiple devices such as smartphones, tablets, and computers. The interfaces of these devices vary from mouse and keyboard to touch input. The application will be browser based so users will need to have a compatible browser for full access of the application. In order to achieve connectivity between these devices we will have to utilize mobile and WIFI networks. All devices will have communication with servers using the Client-Server model so in addition to the devices we want to run the application on, we will have servers that help to update and relay this information to all necessary parties. Our application will be able to scale to native resolutions of different devices. The minimum resolution will be 640x480 pixels. The required bandwidth to properly use the application will be 56kbps. In order to confirm the employee locations, there must be GPS access for employee devices.

X. Algorithms and Data Structures

Algorithms

Wages Calculation

The application will calculate the total wages for each worker based off their total hours worked. The calculation can be represented by *totalWages = hoursWorked*wage*. The value of wage can be changed only by the manager. All these numbers can be viewed by anyone with the required cridentials (such as a potential accountant and the worker themselves)

Rewards Calculation

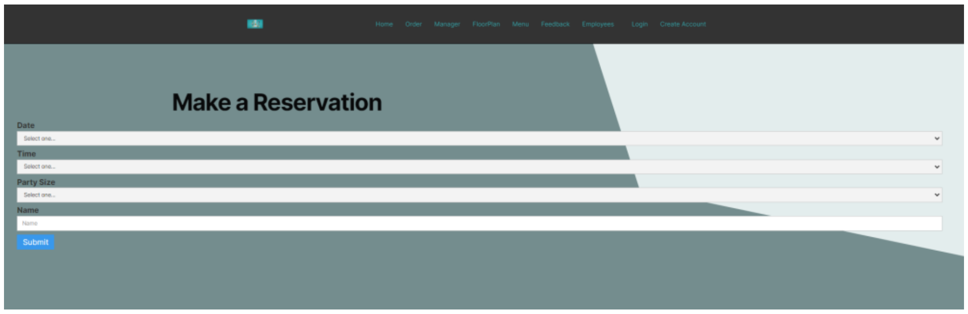
Our application will calculate the rewards points accrued by a user during each transaction through a simple percentage of the total price. The rewards will be calculated by *y = ordercost*x* where *x* is a percentage set by the manager, *ordercost* is the transaction cost prior to taxes or fees, and y is the total points added to the user's account. The default value of x for every purchase will be x = 0.1 but for special purchases or occasions the value of the accrued rewards may be changed. This algorithm can vary from client to client.

Data Structures

The system we propose will use multiple kinds of data structures including queues and arrays. When an employee or a customer commence an order, the item information will be collected in a queue. Thus, it uses the first in first out principle, allowing chefs to work on meals in the order they were received. The menu will be stored in a multidimensional array, allowing us to store not only the name of the item, but also the ingredients of the item. In order to keep track of the various ingredients used, we implemented a two-dimensional array which stores the ingredient name and the inventory level of that ingredient.

XI. User Interface Design and Implementation

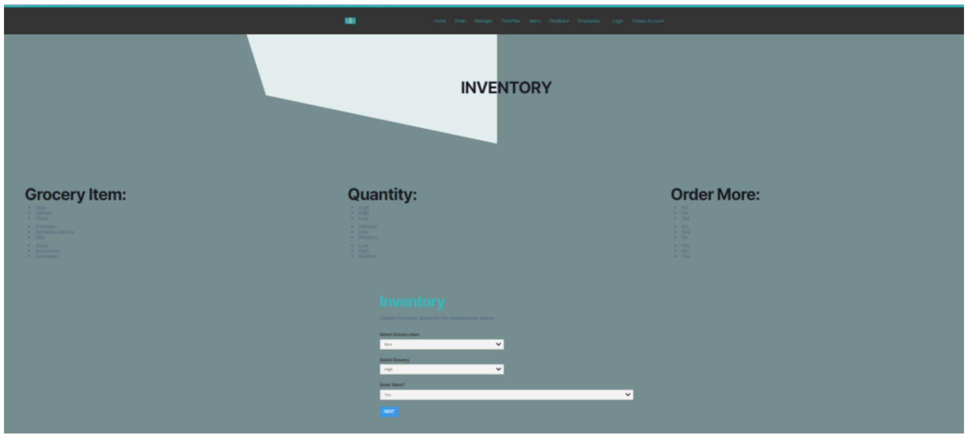
Our design for the user interface is not the same as in our report #1. We decided to scrap the original design because we changed our method of designing the UI. We are going to be redesigning the front end based off of our test website used in the demo. The original design was a bit crude and only served to show the different buttons and functions different parts of the application will have. We are going to be using a minimalist style UI that is able to run on multiple devices and their web browsers. Our User Interface will include a design that minimizes amount of typing the user needs to do within the application. We will be designing it using HTML. Our current design is subject to change throughout the development process.



46

Figure 11

Customers will be able to make a reservation by inputting date, time, party size and name



INVENTORY

Grocery Item:	Quantity:	Order More:
Rice	High	No
Salmon	High	No
Pasta	Low	Yes
Potatoes	Medium	No
Romaine Lettuce	Low	Yes
Milk	Medium	No
Sugar	Low	Yes
Mayonaise	High	No
Ice cream	Medium	Yes

Inventory

Update inventory based on the requirements above.

Select Grocery Item: Rice

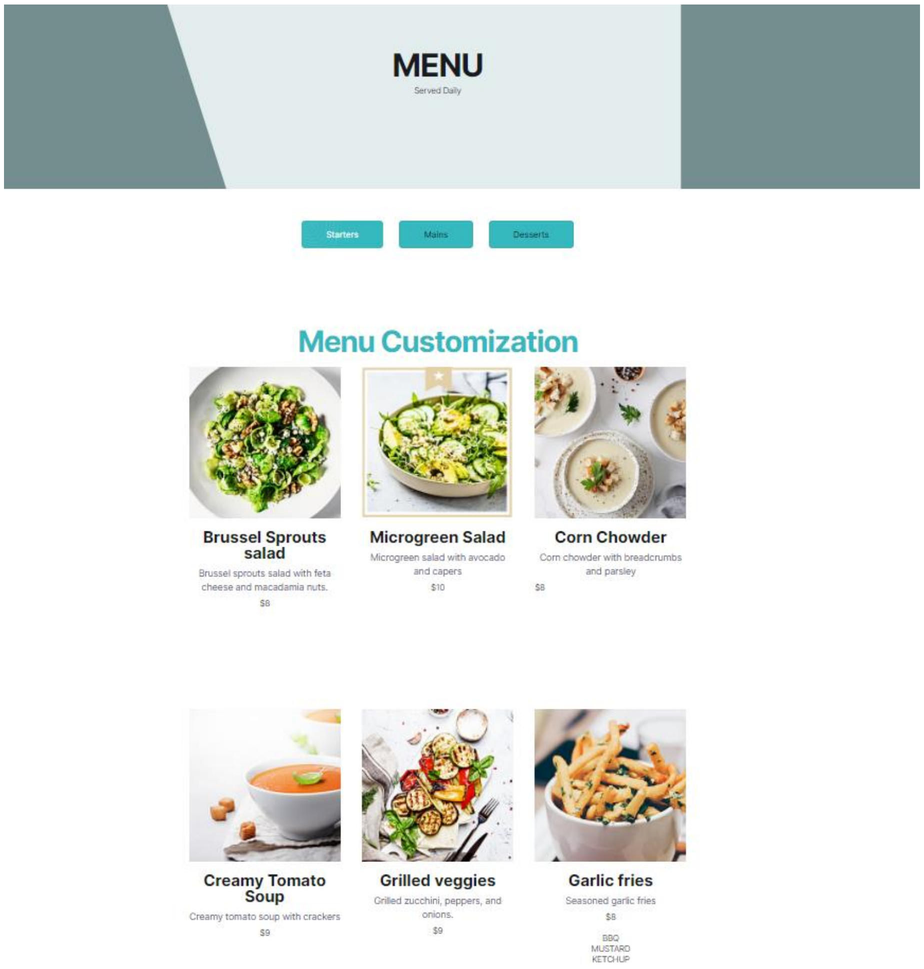
Select Grocery: High

Order More?: Yes

NEXT

Figure 12

Managers will have a screen that displays an inventory, which let them know what items are low in inventory. And will also allowed them to update the inventory by asking them what items they are adding and will also ask them for their credentials, to update the inventory.



MENU

Served Daily

Menu Customization

Brussel Sprouts salad
Brussel sprouts salad with feta cheese and macadamia nuts.
$8

Microgreen Salad
Microgreen salad with avocado and capers
$10

Corn Chowder
Corn chowder with breadcrumbs and parsley
$8

Creamy Tomato Soup
Creamy tomato soup with crackers
$9

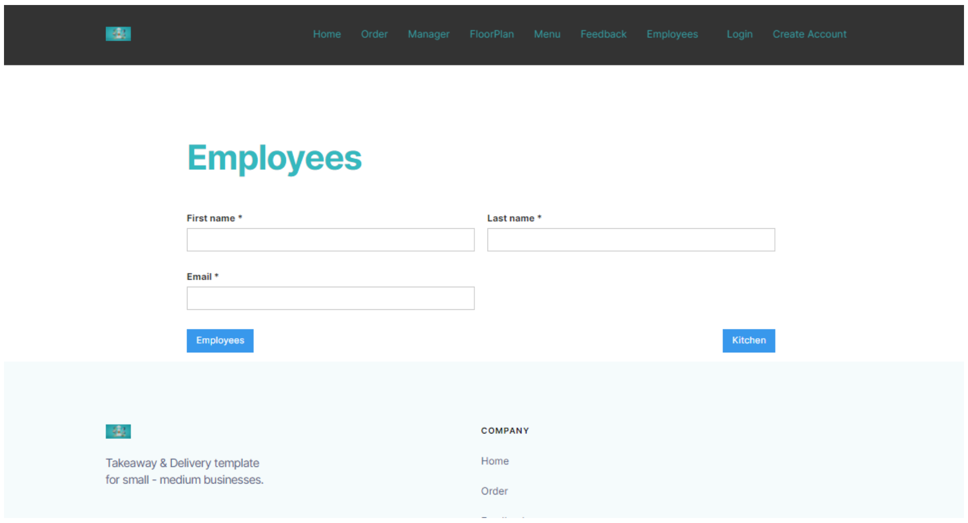
Grilled veggies
Grilled zucchini, peppers, and onions.
$9

Garlic fries
Seasoned garlic fries
$8
BBQ
MUSTARD
KETCHUP

47

Figure 13

Menu will be customized depending on restaurant Diversity.



Employees

First name *

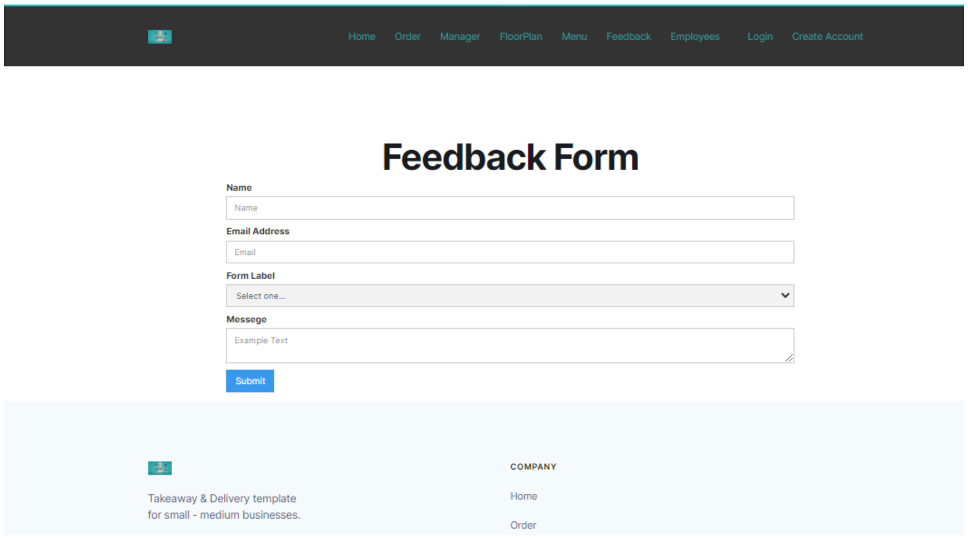
Last name *

Email *

Takeaway & Delivery template
for small - medium businesses.

Figure 14

Employee section where they will be able to login using their email, name.



Feedback Form

Name

Name

Email Address

Email

Form Label

Select one...

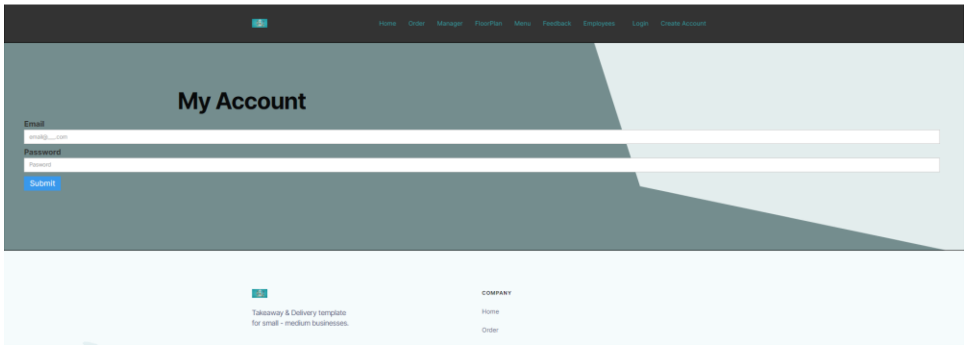
Messege

Example Text

Takeaway & Delivery template
for small - medium businesses.

Figure 15

This the feedback form where customers will be able to send messages to the restaurant, based on their experience.

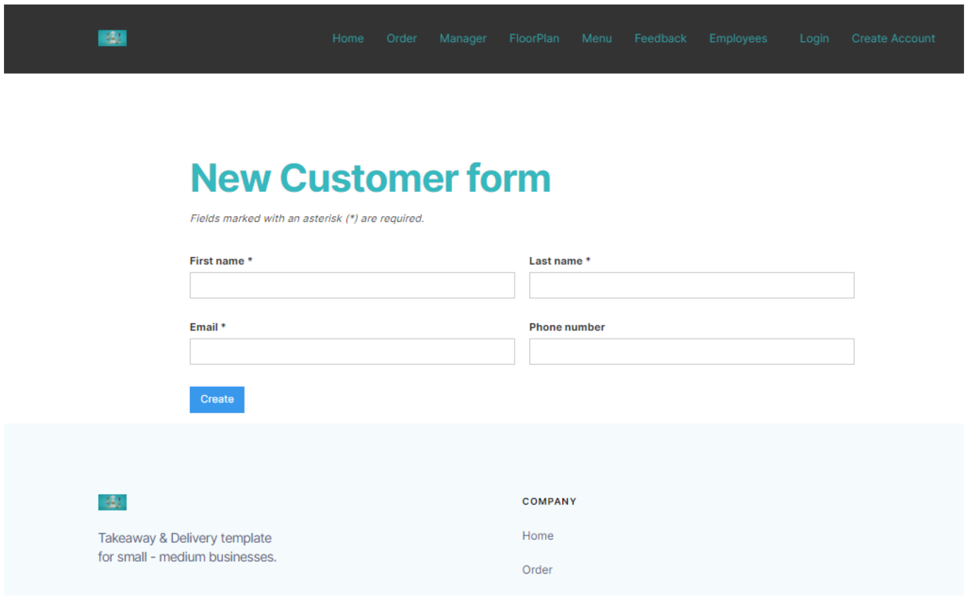


My Account

Takeaway & Delivery template for small - medium businesses.

Figure 16

This is where the customers will access their account by inputting their email and password.



New Customer form

Fields marked with an asterisk (*) are required.

Takeaway & Delivery template for small - medium businesses.

Figure 17

If a customer does not have an account and wants to create one, this screen will ask them for their information such as names, email, phone, and pasword.

49

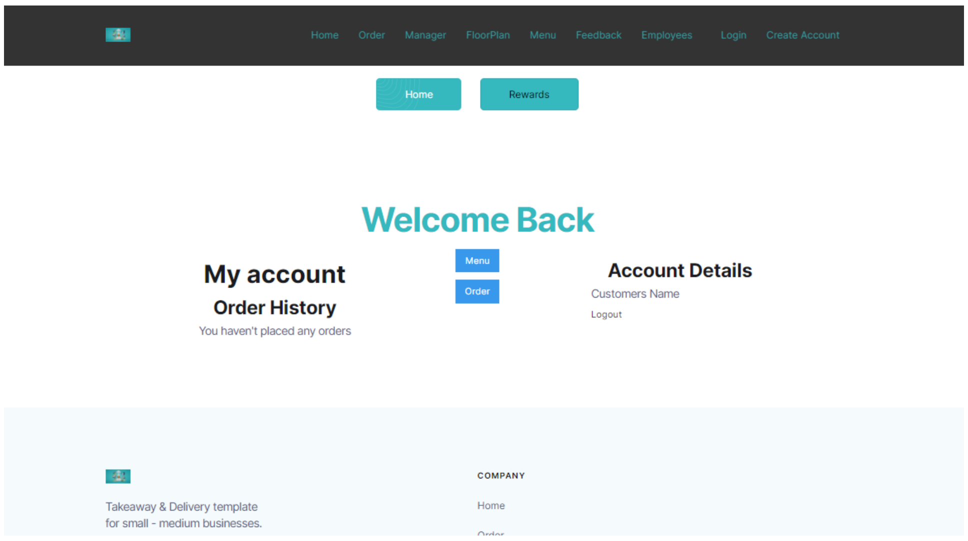


Figure 18

This is the account section where customers will be able to check their account order history, and how much rewards they have save or available.

XII. Design of Tests

Test cases

Use Cases: UC-1, UC-2, UC-4, and UC-9

Test Case Identifier: TC-1 Use Case: UC-1 (Ordering)		
Test Procedure: 11. User selects an item from the menu 12. User submits special requests for the item(s) ordered from the menu 13. User tries to place an order without selecting any items.	Expected Results: 14. Employees see the newly submitted order details on their dashboard. 15. Employees can view special request submissions attached to respective order, and option to pass requests to chef employee. 16. System notifies user to select items before submitting order.	Pass/Fail Criteria: The test will pass if the user is able to successfully order their designed item from the menu, and the employee is able to view newly submitted orders with any special requests if applicable. The test case will fail is the employees cannot see the newly submitted orders, or if the system does not notify the user to select items when zero items were selected when placing an order.

Test Case Identifier: TC-2 Use Case: UC-2 (Reservations)		
Test Procedure: 17. User selects a date, time, and party size and	Expected Results: 19. Server checks if the reservation time is still available for the party	Pass/Fail Criteria: The test will pass if the user is able to make a valid reservation. The test will fail if the user is

then submits the reservation. 18. User does not select one of the three fields and then submits the reservation.	size and then confirms to the user that the reservation was successful. 20. The System notifies the user that one of the fields was not specified.	unable to select all three fields or if the system does not properly notify the user if the fields are in anyway incomplete.

Test Case Identifier: TC-3 Use Case: UC-4 (Take Out)		
Test Procedure: 21. User places an order but does not select a takeout option 22. User places an order for takeout for a specific time	Expected Results: 23. The order is placed but takeout action is not proceeded 24. The order is placed successfully with desired time for pick up	Pass/Fail Criteria: This test will pass if the customer is able to place an order online successfully. The test will fail if the order cannot be taken for takeout or the order cannot go through

Test Case Identifier: TC-4 Use Case: UC-9 (Schedule Employees)		
Test Procedure: 25. User creates a schedule for employee (this is only a manager role) 26. User, not a manager, creates a schedule for employee	Expected Results: 27. User can create a schedule for employee when logged in as manager 28. User is not able to create schedule for employee	Pass/Fail Criteria: The test will pass if and only if the manager is able to create a schedule for employee. The test will fail if user that is not an employee is able to create a schedule.

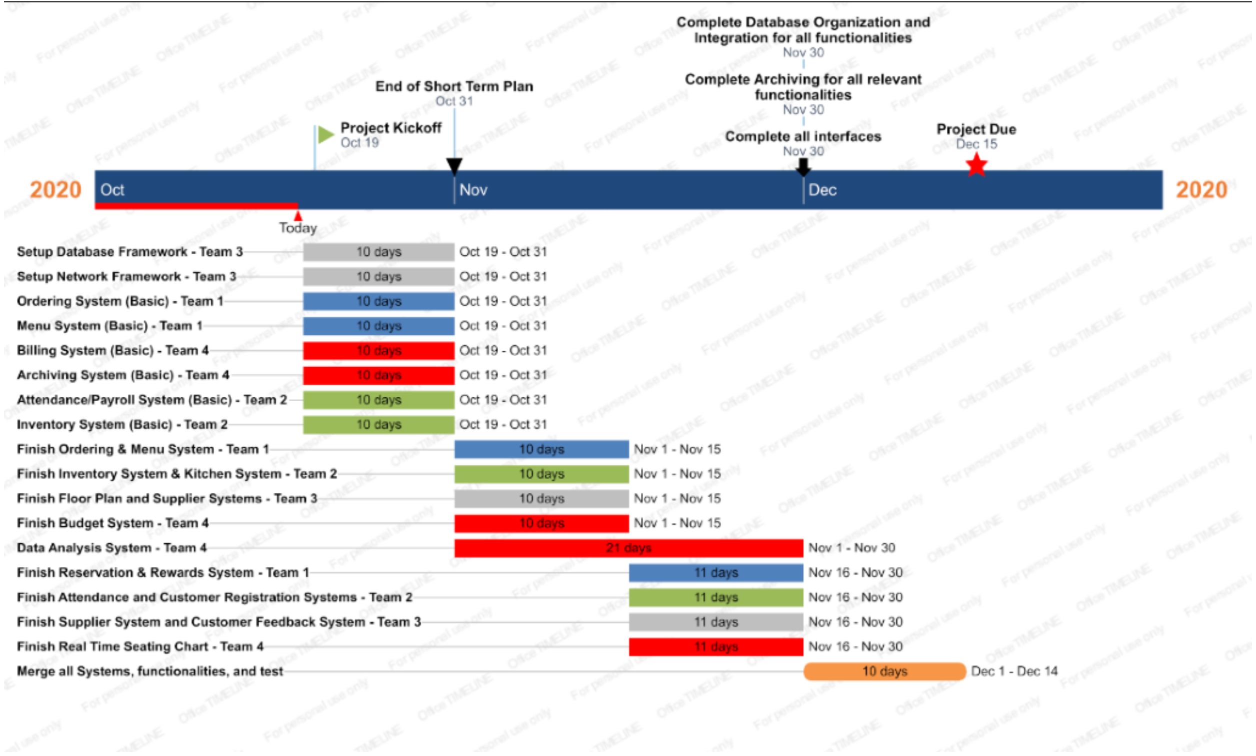
Test Coverage

The test cases cover the basic implementation or our design, some of our other cases have not been included this is because they are on the progress of being implemented. The cases which we implemented will guide the user through an understanding of some function of our system, such as button pressing, navigation through different screens, and pop-up screens. Other cases can be implemented as well, and cases can be modified based of new requirements. From our testing Ordering, the user will have to be on the order tab, a menu will display, the user will only have to click on the item a window will pop-up, in this window the user will be able to add any special request. They will have to click add to my order. Otherwise, the item will not be included. User will be able to see the items which they have already included.in Ordering testing the user input is through buttons, and special request. Which make it simple and user friendly. Now for Reservation testing user input will consist of data, time frame, and amount of people in the party. This will be done through drop down menu selection which the user will select and them they will summit their reservation; user will not be able to input any other type of input. Test cases take out, if user place an order but the option take-out is not selected the order will fail and the order will not be place. Fail will continue if user does not select take-out. Test case for schedule of employees. Managers will be the only authorize user to have access to creating and editing of schedules, any other user will not have access to this case.

Integration Testing

We decided the best method to use for integration testing would be to use the Sandwich Testing, other known as Hybrid Integration Testing; this is a combination of testing the lower-level components first prior to integration with the higher-level components and testing the higher-level components with the lower-level components already integrated. Some high-level components are much easier to create and do not have many components involved, making it easier to test, while other high-level components are composed of several smaller building blocks, thus, it is it better to test the smaller building blocks as pieces rather than a whole to find any bugs that may exist. By using the hybrid method, it enables us to be able to identify bugs in smaller components prior to integrating them to the high-level components, making it easier to construct the high-level components.

XIII. History of Work



Team 1 represents 🟦

Team 2 represents 🟩

Team 3 represents ⬜

Team 4 represents 🟥

All Teams represents 🟧

Figure 19

This Gnatt chart was our original plan of work and includes our originally planned deadlines and milestones. While working on our application, we realized that we could not keep up with all the deadlines we had originally projected. Even though, some functionalities in our project took less time than anticipated, for the most part, most of the systems took more time than anticipated and this caused us to push our deadlines and projected milestones back. Factors such as debugging, school and personal work, and learning new technologies and languages were reasons why we had to push back many deadlines. Even though we were not able to meet our original deadlines, we were still able to accomplish many of the things we set out to do as described below in the key accomplishments' subsection.

Key Accomplishments

- Successfully setup and connected a MySQL database for our application using .NET framework.
- Created multiple working systems using Angular.
    - Ordering System
    - Menu System
    - Billing/Payment System
    - Archiving System
    - Payroll System
    - Inventory System
    - Kitchen System
    - Login System
    - Reservation System
    - Rewards System
    - Registration System
- Created a home page that can access all the different systems.
- Tested the completed systems.

Future Work

Need to create different layouts based on type of user (Customer vs Manager). Need to finish the data analysis system so that it can work for all types of data and is compatible with the inventory, kitchen, ordering, and supplier systems.

XIV. References

1) The Software Engineering Textbook by Ivan Marsic:
    a) https://www.ece.rutgers.edu/~marsic/books/SE/book-SE
2) Description of Use case diagrams and examples of use case Diagrams
    a) Ambler, Scott W. *UML 2 Use Case Diagrams: An Agile Introduction*, www.agilemodeling.com/artifacts/useCaseDiagram.htm.
3) Explanation of System Sequence diagrams.
    a) "The #1 Development Tool Suite." *Ideal Modeling & Diagramming Tool for Agile Team Collaboration*, www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-di.
4) *Purpose of Use case Diagram, with examples and way of identifying use cases.*
    a) *What Is Use Case Diagram?* www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/.
5) Spring 2019 Semester Team #13
    a) https://www.ece.rutgers.edu/~marsic/books/SE/projects/Restaurant/2019-g13-report3.pdf
6) Understanding of Traceability Matrix and what requirements are included.
    a) Guru99 2020. "What Is Requirements Traceability Matrix (RTM)? Example Template." *Guru99*, www.guru99.com/traceability-matrix.html.
7) Creation of the Preliminary User Interface.
    a) https://www.canva.com/
8) Creation of Use Case Diagram and System Sequence Diagram.
    a) https://creately.com/