Streamline Dining

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

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

Table of Contents 2

Individual Contributions Breakdown 3

Interaction Diagrams 4

Diagrams 4

FDD: Ordering Dine-in 4

FDD: Reservation 4

FDD: Payment 5

FDD: Clock in/out 6

Class Diagrams and Interface Specification: 7

Class Diagrams: 7

Data Types and Operation Signatures: 7

Traceability Matrix: 13

System Architecture and System Design: 14

Architectural Styles: 14

Identifying Subsystems: 15

Mapping Subsystems to Hardware: 16

Persistent Data Storage: 16

Network Protocol: 16

Global Control Flow: 16

Hardware Requirements: 17

User Interface Design and Implementation 17

Design of Tests ..... **Error! Bookmark not defined.**

Test cases 19

Test Coverage 20

Integration Testing 20

Project Management: 20

Merging the Contributions from Individual Team Members 20

Project Coordination and Progress Report 21

Plan of Work 21

Breakdown of Responsibilities 21

References 22

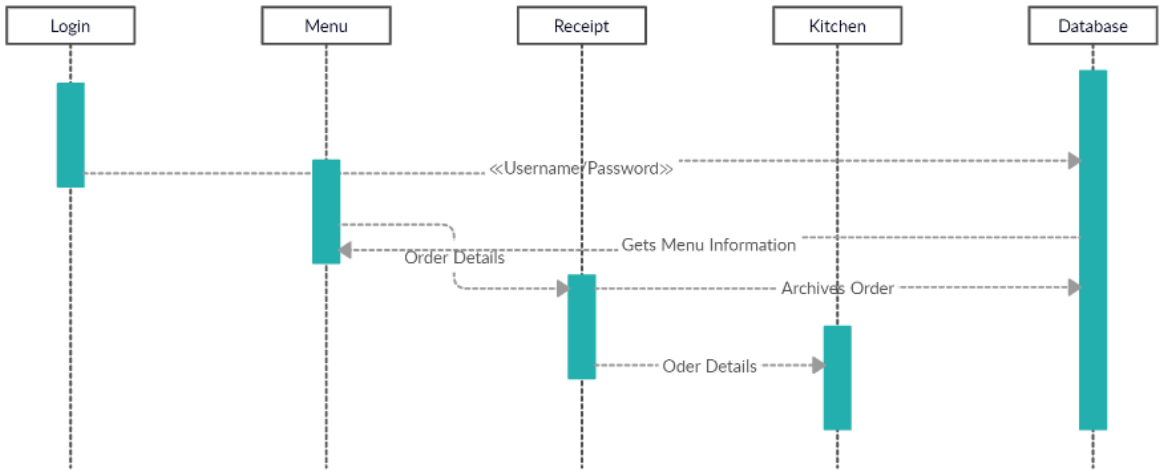
Individual Contributions Breakdown

Topic	Harman	Eugene	Max	Joel	Justice	Karneet	Pablo	Srinu	Talya
Individual Contributions	11.1%	11.1%	11.1%	11.1%	11.1%	11.1%	11.1%	11.1%	11.1%
Interactions Diagram	5%	5%	5%	5%	5%	5%	35%	35%	5%
Class Diagrams and Interface Specification			30%			30%	2.5%	2.5%	35%
System Architecture and System Design	16.6%		16.6%	16.6%	16.6%		16.6%	16.6%	
Hardware Requirements	100%								
User Interface Design and Implementation	40%		40%	5%		5%		5%	5%
Design of Tests	5%		5%	15%	15%	15%	15%	15%	5%
Project Management	5%	5%	5%	5%	5%	5%	15%	50%	5%

Interaction Diagrams

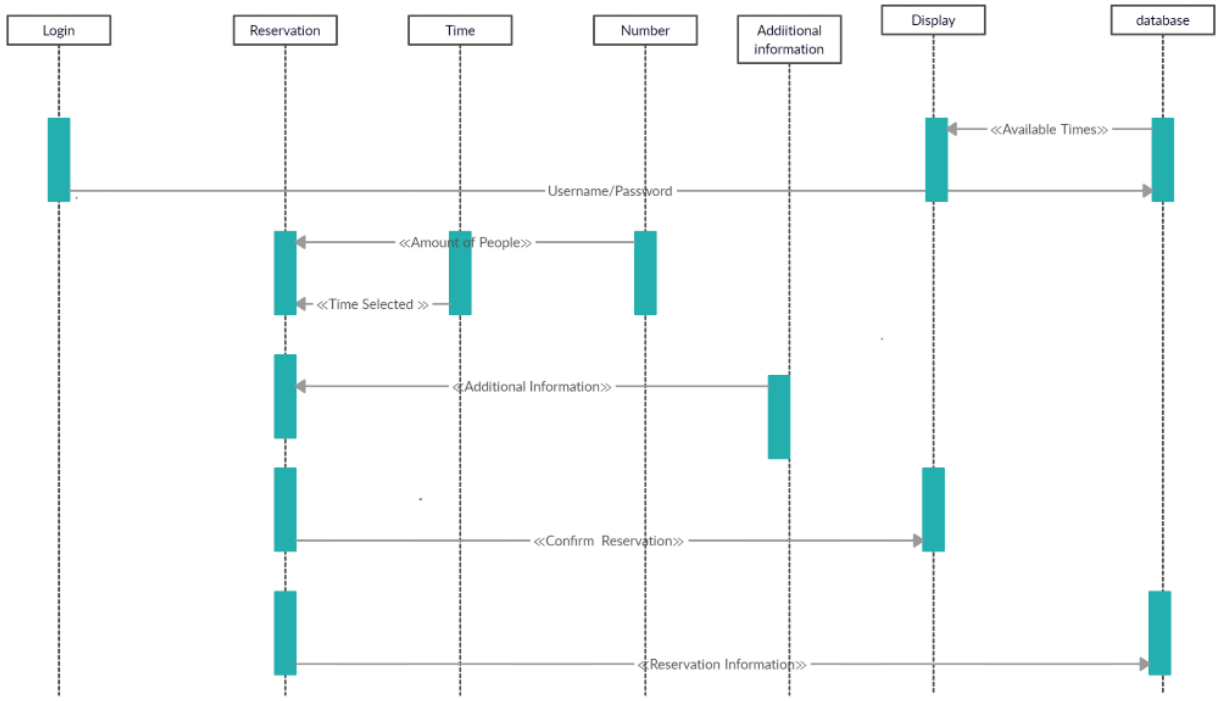
Diagrams

FDD: Ordering Dine-in



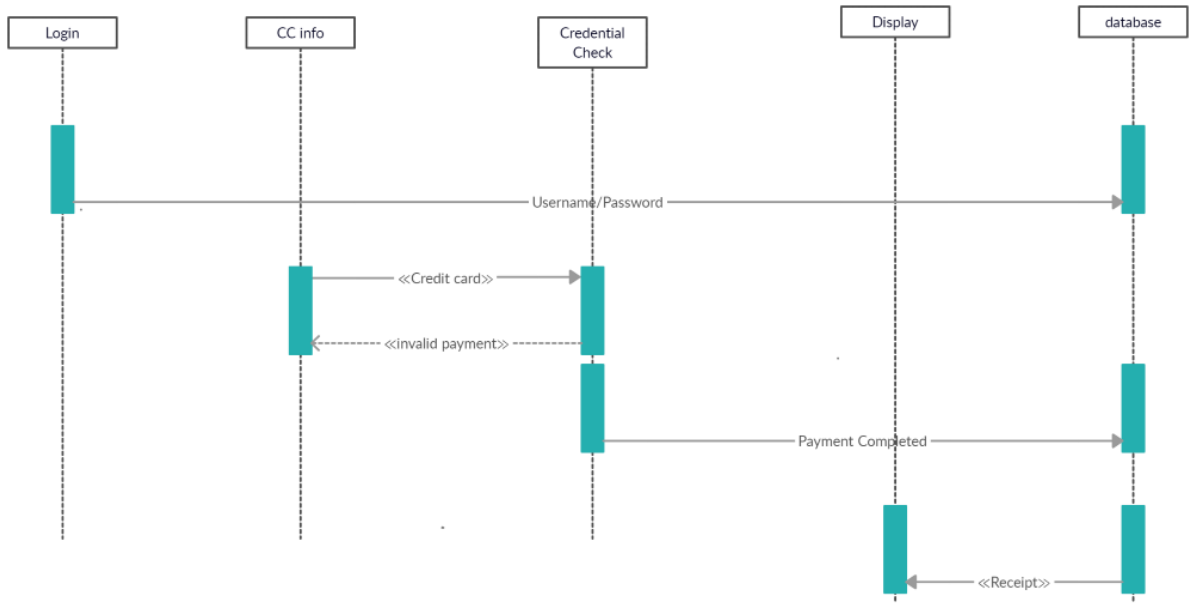
This interaction is how customers will be able to dine in. Either the server or the customer will log onto the application and select the dine in option. The system will then display the menu and then the customer will fill in all the order information. They will then confirm the order and the order will be stored in the database as well as be sent to the kitchen.

FDD: Reservation



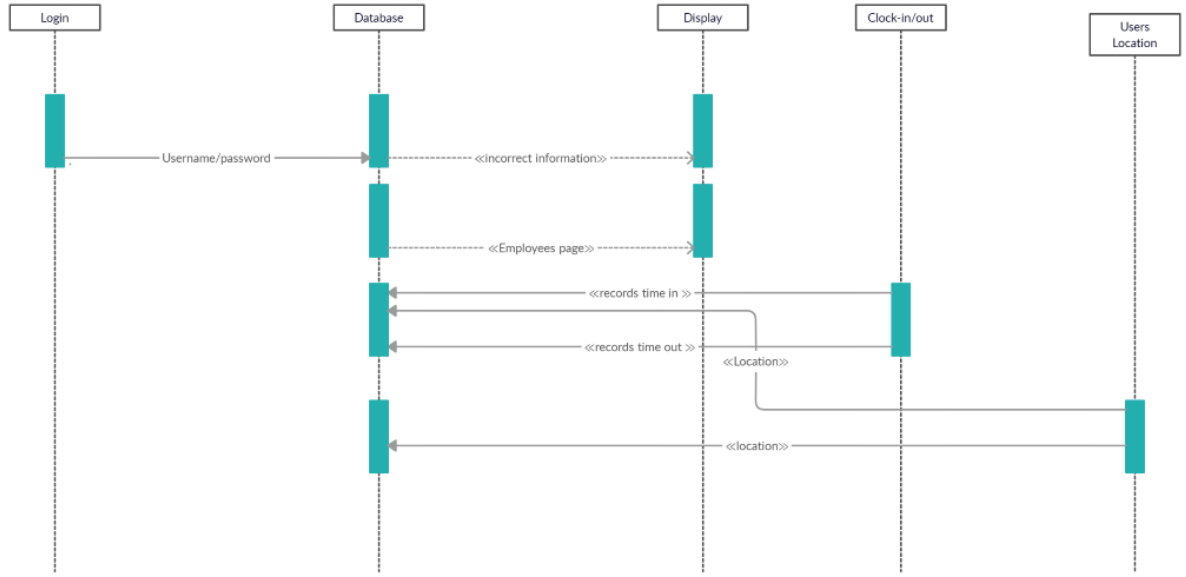
This interaction details how customers will be able to make online reservations. The customer must be logged into their account or create a new account. Once logged in, they will select the reservation option. The system will display available reservation times and the customer must select from one of 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



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.

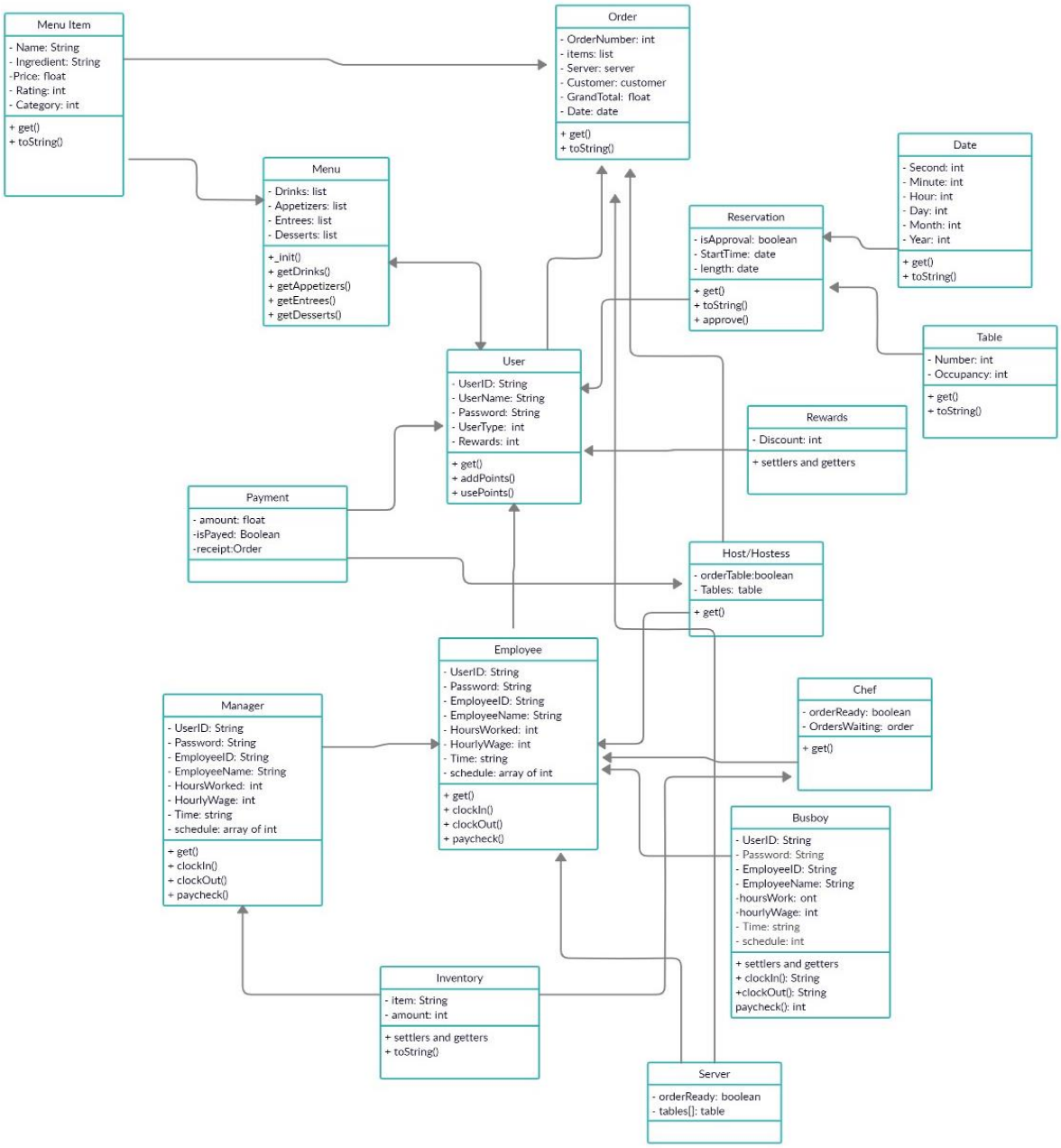
FDD: Clock in/out



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.

Class Diagrams and Interface Specification:

Class Diagrams:



Data Types and Operation Signatures:

Class: Login

All Users
-UserID: String -Password: String

+setters and getters

Class: HomeScreen

All Users
-userID: String -userName: String -password: String -userType: int
+setters and getters

Class: Menu

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

Class: menuItem

Manager, Chief
-name: String -ingredients: String[] -price: float -rating: int -category: int
+getters for all items +ToString()

For category; 0 = drinks, 1 = appetizer, 2 = entrees, 3 = desserts

Class: Order

Server, Manager
-orderNumber: int -items: menuItem[] -server: Server -customer: Customer -grandTotal: float -date: Date
+getters for all items +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 -Customer: customer -Payment: payment
+setters and getters

Class: Table

Server, Busboy, Manager
-Number: int -occupancy: int
+setters and getters

Class: Reservation

Host, Customer
-isApproval: Boolean -startTime: date -length: date
+setters and getters +toString() +approve()

Class: User

-userID: String -userName: String -password: String -userType: int -rewards: int
+setters and getters +addPoints() +usePoints()

Class: EmployeeProfile

-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: Chef

-orderReady: boolean -ordersWaiting: order[]

Class: Server

-orderReady: boolean -tables[]: Table

Class: Host/Hostess

-openTable: boolean -reservations: Reservation[]

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 -isPayed: Boolean -recept: Order

Class: Inventory

Chief, Host, Manager
-item: String -amount: int
+setters and getters +toString()

Class: Rewards

Manager, Customer, Host, Server
-Discount: int
+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								
Homescreen									
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
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

System Architecture and System Design:

Architectural Styles:

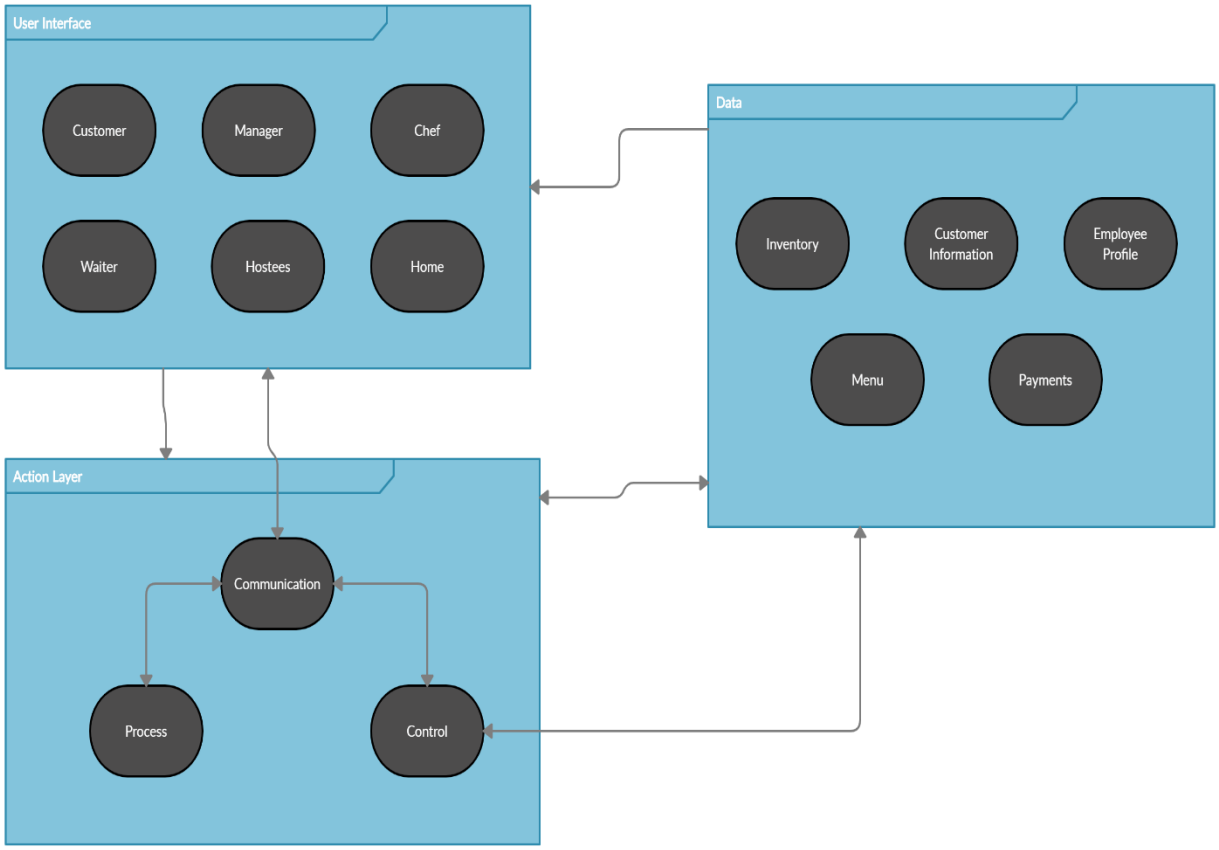
Architectures styles are incorporated in Software to construct structures and systems which will become efficient at catching early problematic decisions on the system design and will allowed for a solution. Our design architecture will be composed of 2 Architectures Styles

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:



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 need 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 of the data would need to be encrypted asymmetrically to prevent anyone from intercepting the packets.

All our data will be stored on a server inside the restaurant. 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.

This server will be on a separate subnet from the internet that clients will be allowed to access, and it will have its own firewall to ensure maximum security. 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.

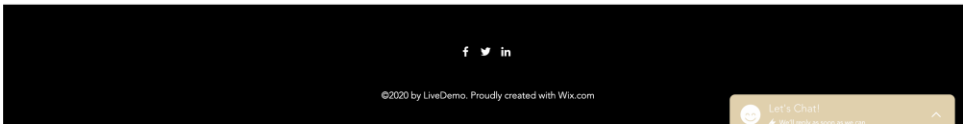
Hardware Requirements:

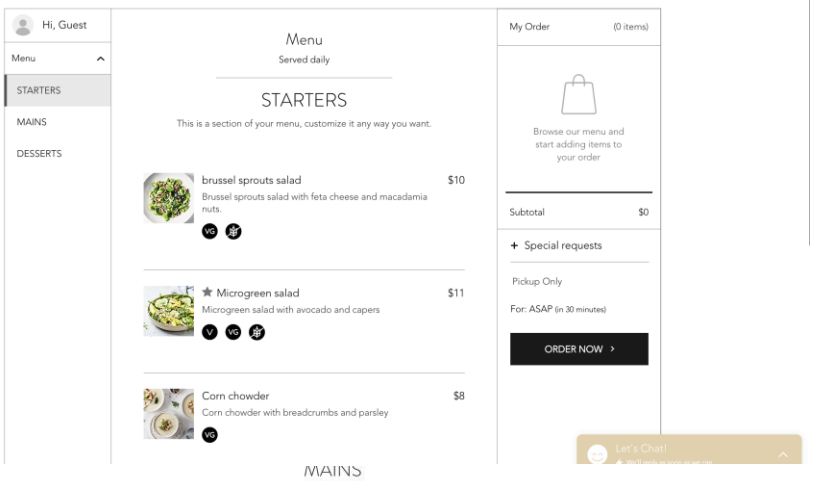
Our application is going to be designed to run on multiple devices such as smartphones and tablets. The interfaces of these devices vary from mouse and keyboard to touch input. In order to achieve connectivity between these devices we will have to utilize mobile and WiFi networks. All devices will have 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.

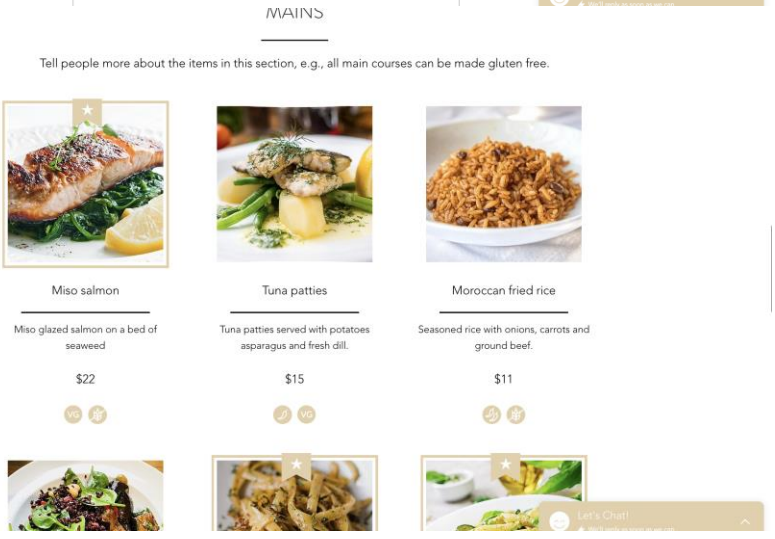
User Interface Design and Implementation

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.


LIVEDEMO
Home
Menus
Orders
Employees
Make a Reservation
Date
Time
Party Size
Submit


©2020 by LiveDemo. Proudly created with Wix.com
Let's Chat!


Hi, Guest
Menu
STARTERS
MAINS
DESSERTS
Menu
Served daily
STARTERS
This is a section of your menu, customize it any way you want.
brussel sprouts salad
$10
Brussel sprouts salad with feta cheese and macadamia nuts.
Microgreen salad
$11
Microgreen salad with avocado and capers
Corn chowder
$8
Corn chowder with breadcrumbs and parsley
My Order
(0 items)
Browse our menu and start adding items to your order
Subtotal
$0
+ Special requests
Pickup Only
For: ASAP (in 30 minutes)
ORDER NOW
Let's Chat!


MAINS
Tell people more about the items in this section, e.g., all main courses can be made gluten free.
Miso salmon
Miso glazed salmon on a bed of seaweed
$22
Tuna patties
Tuna patties served with potatoes asparagus and fresh dill.
$15
Moroccan fried rice
Seasoned rice with onions, carrots and ground beef.
$11
Let's Chat!

Design of Tests

Test cases

For the first demo we are showcasing the following Use Cases: UC-1, UC-2, UC-4, and UC-9

Test Case Identifier: TC-1 Use Case: UC-1 (Ordering)		
Test Procedure: 1. User selects an item from the menu 2. User submits special requests for the item(s) ordered from the menu 3. User tries to place an order without selecting any items.	Expected Results: 1. Employees see the newly submitted order details on their dashboard. 2. Employees can view special request submissions attached to respective order, and option to pass requests to chef employee. 3. 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: 1. User selects a date, time, and party size and then submits the reservation. 2. User does not select one of the three fields and then submits the reservation.	Expected Results: 1. Server checks if the reservation time is still available for the party size and then confirms to the user that the reservation was successful. 2. The System notifies the user that one of the fields was not specified.	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 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: 1. User places an order but does not select a takeout option 2. User places an order for takeout for a specific time	Expected Results: 1. The order is placed but takeout action is not proceeded 2. 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:	Expected Results:	Pass/Fail Criteria:

1. User creates a schedule for employee (this is only a manager role) 2. User, not a manager, creates a schedule for employee	1. User can create a schedule for employee when logged in as manager 2. User is not able to create schedule for employee	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 case 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.

Project Management:

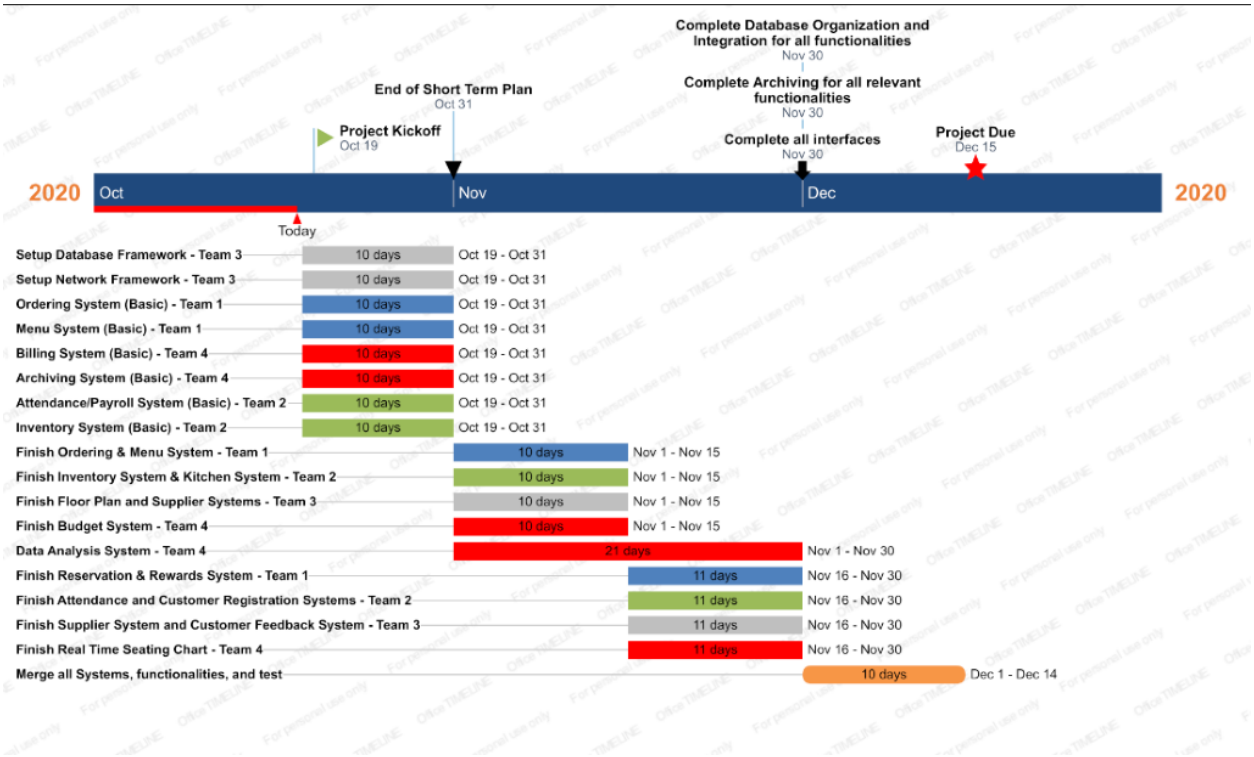
Merging the Contributions from Individual Team Members

Issues we have occurred is integrating and organizing the database so that it will work with all the sub-systems in our project. We are currently formatting the database to be adaptable to new systems as well as work with different systems.

Project Coordination and Progress Report

Almost finished setting up our web-based database and network framework. Created the basic functionalities for the ordering, menu, billing, archiving, payroll, and inventory systems. These correspond to use cases 1, 3, 7, 8, 10, and 12. We are currently working on integrating and completing these systems. We are also planning on starting work on the kitchen, budget, and floor plan systems.

Plan of Work



Breakdown of Responsibilities

Team 2 (Talya & Eugene) will be coordinate integration and will also oversee integration testing. Whoever developed a unit will assist team 2 in the integration and testing for their unit.

Every team will be responsible for collecting, organizing, and processing data for their functionalities as well as developing basic user interfaces. In addition to these tasks, each team will be responsible for managing and supervising global tasks.

Team 1: Harman, Max

- **Functionality**
  - Virtual Ordering
  - Menu system
  - Reservation system
  - Customer Reward System
- **Qualitative property**
  - User Interfaces/Graphic Design

Team 2: Talya, Eugene

- **Functionality**
  - Inventory Tracker
  - Attendance System
  - Kitchen system
  - Customer Registration System
- **Qualitative property**
  - Merging systems and functionalities

Team 3: Joel, Pablo

- **Functionality**
  - Feedback System
  - Floor Plan
  - Schedule
  - Supplier System
- **Qualitative property**
  - Performance and unit testing

Team 4: Srinu, Karneet, Justice

- **Functionality**
  - Budget System
  - Real Time Seating Chart
  - Data Analysis System
- **Qualitative property**
  - Database interactions

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/

(9) Creation of Use Case Diagram and System Sequence Diagram.
   a. https://creately.com/

(10) Interaction Diagram
   a. https://www.youtube.com/watch?v=Kpy2zKC21wg

(11) Gannt Chart
   a. http://www.ganttchart.com/

(12)Architectural Styles
   a. Paganini, Catherine. “Primer: Understanding Software and System Architecture.” *The New Stack*, 12 Dec. 2019, thenewstack.io/primer-understanding-software-and-system-architecture/.