Restaurant Automation
Report #2 Part 2

Software Engineering - 14:332:452

Group #14: Jan Matthew Miranda, Kevin Dai, Eric Jiang, Peter Luo, Christian Remolado, Leonardo Roman, Mohammad Sadiq Rehan

URL:

https://github.com/leonardoARoman/RestaurantAutomationMainSystem

Effort Breakdown

	Jan	Kevin	Eric	Peter	Christian	Leonardo	Sadiq
Problem Statement	40%		30%		15%	15%	
Glossary			60%	40%			
Functional Requirements	20%	30%				30%	20%
Nonfunctional Requirements		50%				50%	
UI Requirements	25%			25%	25%		25%
StakeHolders & Actors	30%		30%	20%			20%
Use Case Casual	100%						
Use Cases Full Description		33.3%		33.3%		33.33%	
System Sequence Diagram		25%			50%		25%
Preliminary Design	60%			25%		15%	
Effort Estimation	20%		60%		20%		
Concept Definitions		20%	40%		20%		20%
Association Definitions			20%				80%
Attribute Definitions			20%		80%		
Traceability Matrix						100%	
Domain Model		25%				75%	
Traceability Matrix							100%
System Operation Contracts	100%						
Plan of Work	60%		40%				
Interaction Diagrams					*	*	*
Class Diagram	*					*	
Data Types and Operation Signatures	*					*	
Traceability Matrix			*				
System Architecture and System Design			*		*		*
References			33.33%		66.67%		
Project Management	100%						

Table of Contents

Customer Statement of Requirements

Problem Statement

As more and more people decide to eat out instead of cooking at home, restaurants are continuing to grow in size and numbers. As they continue to expand, their vulnerabilities become more and more exposed. Restaurants have been around since the 11th century some historians say, and although foods, equipment and aesthetics have changed, the process of running a restaurant hasn't. As restaurants continue to gain more patrons, some are looking at expanding staff and floor space to house more patrons and serve them. However this solution doesn't truly solve the issue, it's just multiplying the inefficiencies they had since they opened. However we believe our Restaurant Automation Software is the solution for expanding restaurants how want to truly become efficient in the food catering business. We aim to solve problems restaurants have such as table management, communications between table orders and the kitchen, easy archiving of sales, workers recorded hours worked and more through technology. We believe restaurants who are expanding and plan on taking a larger amount of patrons should adopt our Restaurant Automation Software to increase their efficiency, save money and time.

Listed Problems

1.1 Difficulty with Accessing Restaurant Information

Problem:

Many customers come to a restaurant wondering about many minute details that take up time to ask hosts and hostesses upon their visits. Although visitors' questions may be small and quick to answer, being constantly asked the same questions can lead to lost time that could have been used for other host tasks such as checking reservation statuses and checking up on table statuses. Because time is money is the big restaurant industry, this ultimately leads to inefficiency.

Solution:

The website displays very transparently the menu, contact information, current offers/discounts. This way, the website provides a map of tables with availability and a login account with user information to check-in conveniently online.

1.2 Maintaining Table Availability Status

Problem:

When customers first enter a restaurant, they are usually greeted and asked how many people are within a customer's party so that they can be put in a paper-to-pen queue for a table. After that, hosts have to keep an eye out for when a table opens up physically around the restaurant in order to update a whiteboard, diagrammed layout of the restaurant which shows all tables and statuses. This is problematic as hosts spend a lot of time trying to update and transcribe the queue list and the whiteboard, which can lead to errors. On top of that, the hosts don't have a lot of time greeting the guests to make them feel welcome to the establishment.

Solution:

To reduce the amount of mental and writing work for the hosts/hostesses, a computerized algorithm will be introduced to find available tables and seat guests accordingly. The computer will ask for two simple inputs: the customer's name and how many people in the customer's party. Then the algorithm will place parties in separate queues based on how big the party is (how many people are in the party). Then, the queues will be prioritized by who arrived and got the host/hostess to input their party information first. Then, when a table is ready, a notification will come up on the host or hostess' device, displaying which table is ready for which customer party in the appropriate queue. At that point, the host or hostess can confirm on the device if they will seat the customer's party at the recommended table.

This also beneficially allocates social and emotional interactions for hosts and hostesses to have with their customers to make the latter feel welcome, rather than potentially stressing hosts and hostesses with laborious writing and constant organizing.

1.3 Customers Waiting to Pay the Check

Problem:

Most of the time, customers are looking to pay the bill immediately after they finish their meal. However, waiters often prioritize taking orders and serving dishes, putting the check in the backburner of their minds. Furthermore, for the check to be finalized, it has to go through a gruesomely long process. First, the check has to be prepared by the waiter, then brought to the customer, stalled by the customer's decision on how to pay, then stalled again until the waiter comes back to take the check back to the register This process can be even more prolonged if the waiter is waiting tables during high-traffic hours, making the transporting times for the check even longer.

Solution:

Instead of taking the orthodox route of transporting checks, a tablet-based payment system can be introduced to cut down on the time customers have to wait for the check. In this system, a tablet is introduced at each table so that when a party is all finished, the customers can then choose when and how to pay. The tablet would then be able to access and display the table's ordered items with their individual prices, a subtotal, the total tax, and an option to tip the waiter electronically. Finally, the electronic payment system will ask for cash, credit, or debit payments and respond accordingly with correct change and a physical receipt.

1.4 Attendance Punctuality

Problem:

Obsolete clock in/out methods such old punch cards, or ID number typing, or even swiping badges can not keep employees from cheating the time they clocked in/out. There are cases in which employees who are running late ask their fellow/friend employee to clock them in so there can not be any record that such employee was late for work.

Solution:

Fingerprint recognition would solve this integrity problem. Every employee must use their fingerprint to clock in/out as part of self identification.

Proposed Solution

With our solution, we aim to address the problem of restaurants being inefficient and held down by the non-optimized systems workers use to operate the restaurant. One of the biggest problems we wanted to address was the wait time between a customer's meal and the time they pay the check. Our system seeks to improve the traditional system through automating the systems through electronic means; the customer will be seated based off an electronically maintained table status window, the customer's orders will be sent electronically to the kitchen, meal status will be electronically notified to waiters and payment can be handled electronically by the table if through credit/debit.

Our proposed automated system will be available to restaurant owners through a software suite that contains a main desktop application, a website and a mobile application. Each application will be responsible for different systems running in the restaurant. The main application will be charge of maintaining table status, assigning waiters to tables, viewing archived orders, and clocking in workers' hours. The mobile application will handle sending of the meal orders to the kitchen via electronic notifications, notifying waiters of meals ready to serve, and paying checks through credit cards. The website will be a portal to captivate potential customers through previewing the menu, displaying photos of the food, and providing a simple reservation system.

Through this software suite, restaurant owners will be able to increase overall efficiency of the restaurant from all systems. Not only will it increase the speed at which a customer can sit down, eat, and pay but it will also increase the efficiency of managers and accountants. The archival capabilities of clocking in workers' hours and recording sold meals will allow for easy payment to employees as well as accurate information of revenue gained, respectively.

A restaurant's main purpose is to give customers a place to sit and eat delicious meal giving the customers a sense of convenience of just having to eat and pay. Our app will increase efficiency in and outside of the kitchen, which in turn will increase customer satisfaction, reduce workers responsibilities, and generate the restaurant more revenue.

Glossary of Terms

Technical Terms	
Database	The file where the menu items, inventory, scheduling and orders are stored.
Employee Portal	Schedule accessible by employees that lists their respective shifts and any open shifts they may cover.
Restaurant Automation	Use of an internal restaurant management application to automate and carry out major operations within a restaurant establishment
Graphical User Interface (GUI)	A type of user interface that allows users to interact with electronic devices through graphical icons and visual indicator
Table Availability Schedule	A database that shows the availability of the tables in the restaurant
Order Progress Queue	A priority queue that shows the progress of each order and what will be prepared first.

Non-Technical Terms	
Foodies	A person who has an ardent or refined interest in food and alcoholic beverages. Seeks food experiences as a hobby.
Bartender	Serves beverages and maintains the supply and inventory of the bar
Bill	A statement which contains details of the menu items that have been purchased and the money that is owed to them
Busboys	Clears tables, takes dirty dishes to the dishwasher, and sets the tables

Chef	Responsible for creating and planning menus, overseeing food preparation, and supervising the kitchen staff
Chef Hotline	The functionality that allows customers to contact the chef from the waiter's tablet to inquire about menu items
Customer	Any person that orders an item from the menu or walks into the restaurant. He/She can order food and place reservations online.
Customer Satisfaction	Measurement of how food preparation, customer service, and overall experience meets or surpasses customer expectation
Dine-in	When a customer would like to be seated and eat in the store
Host/Hostess	The person to greet and seat a party
Manager	The person that is responsible for inventory management, employee scheduling, payroll and customer satisfaction
Menu	A list of food and beverage offered to the customer
Profit	Revenue subtracted by expenses
Party	The customer and their guests, if they have brought any. A party is sat together.
Reservation	An arrangement made ahead of time for the party to dine at the restaurant
Take-out	Orders that are prepared for customers to be eaten outside of the restaurant
Waiter/ Waitress	Employee that takes customers' orders, brings completed orders to customers, and marks recently vacated tables

Division of Labor

Subgroup A - Main Application

- Leonardo Roman
- Kevin Dai
- Eric Jiang

In charge developing main application. Main functionality will include, a login page for users, a window up the restaurant floors with tables to maintain table status, a window for workers to clock-in and clock-out for their hours, a window that show pasts orders of patrons, a window that allows you to see workers hours.

Subgroup B - Mobile Application

- Mohammad Sadiq Rehan
- Christian Remolado
- Jan Matthew Miranda

In charge of developing mobile application to be used on tablets. Main functionality will include, a login page to associate user with a floor and set of tables, a window to view table user is currently serving, a list of menu items that can be sent to the kitchen, a notification system to notify user when food can be served from kitchen and a feature to update table status.

Subgroup C - Website Development

- Peter Luo

In charge of developing website. Main functionality will include, displaying restaurant information to potential patrons and allowing patrons to set reservations.

System Requirements

Functional Requirements

Identifier	User Story	Story Point
REQ-1	As a hostess, I can search and update available tables quickly	4
REQ-2	As a hostess, I can update reservations if requested by guest	2
REQ-3	As a waiter/waitress, I can quickly send table orders to the kitchen without having to go their myself	6
REQ-4	As a waiter/waitress, I can promptly be notified of when meals are ready to be sent from the kitchen to the table	4
REQ-5	As a waiter/waitress, I can easily let the hostess know of when tables are available for new customers without going to the hostess directly	3
REQ-6	As a waiter/waitress, I can keep track of which meals are to be served at the respective tables	2
REQ-7	As a waiter/waitress, the bill can be provided quickly and accurately as the order history and cost will be stored on the system without errors.	4
REQ-8	As a cook, I can see in queue orders and prepare them accordingly	4
REQ-9	As an employee, I can clock in and out of work	4
REQ-10	As a manager, I can make changes on orders if needed	3
REQ-11	As a manager, I can post staff work schedules	4
REQ-12	As a manager, I can see all transaction reports such as daily sales, profits, payments etc	4
REQ-13	As a manager, I can see how long each waiter worked by the amount of time they were logged into their tablet	3
REQ-14	As a customer, I can make reservations	3
REQ-15	As a customer, I can find information about the restaurant such as menu, hours of operations, etc	2

Nonfunctional Requirement

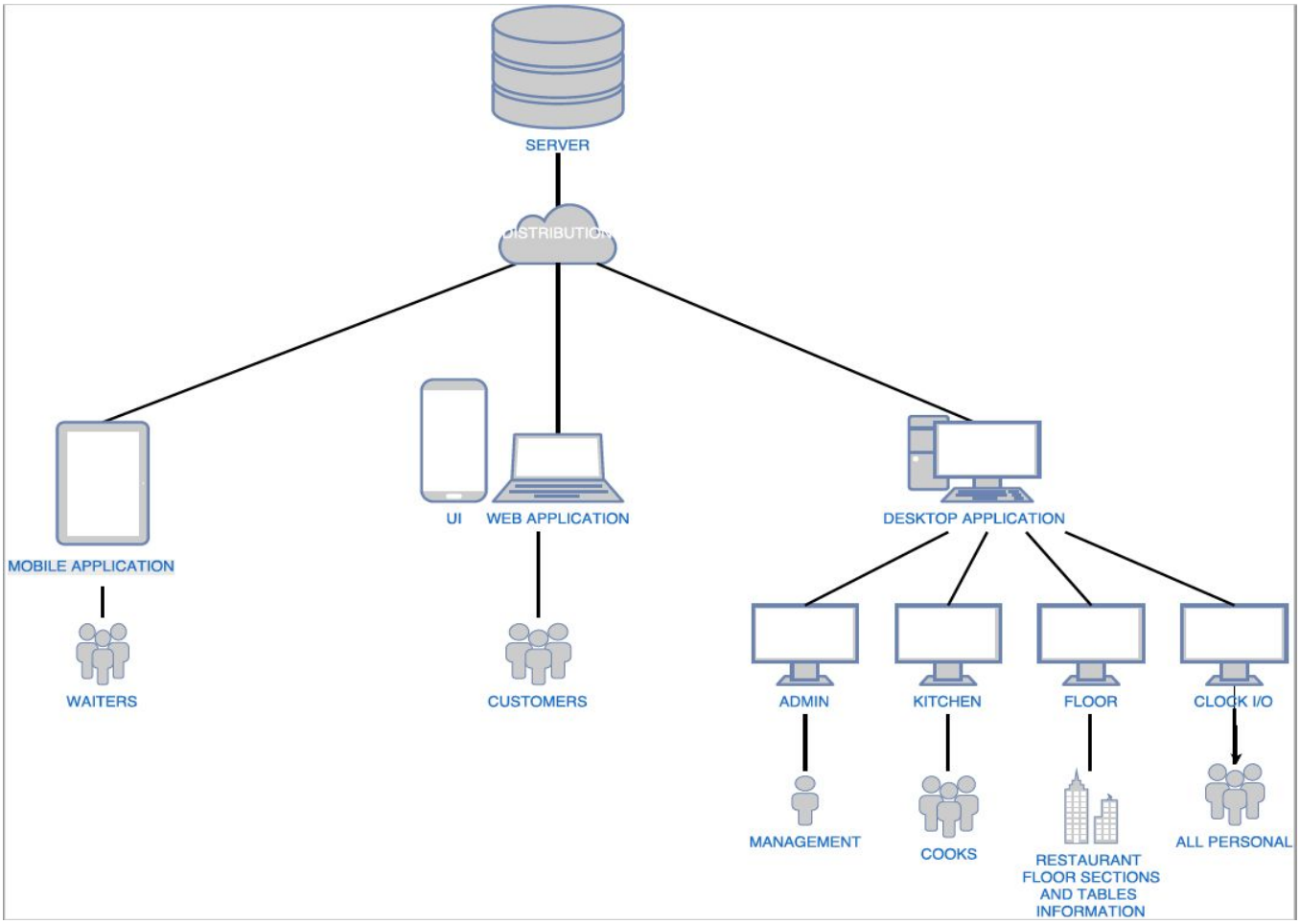


Figure 1.1 The System Architecture of the Restaurant Software Suite

FURPS	
Functionality	To divide the system-to-be into subsystems independently of each other in such way that each subsystem will meet all requirement satisfaction.
Usability	For targeted users such as customers, staff and management.
Reliability	For all users with their respective request such as making reservations, staff scheduling, table designation etc.
Performance	Input/Output transactions such as payments, table state, orders, queries, etc.
Supportability	Client/Server interprocess communication via cloud/server

Ideology of System Architecture

Before designing the system-to-be, architectural decision was made, such as dividing and conquer. The main system was divided into three subsystems to target all subdomains(subproblems) and solve them, accordingly to all requirements previously gathered, in order to deliver the final product. Furthermore, desktop application was divided even further in order to solve different challenges in the restaurant domain. As we can see in picture 1.1, desktop subsystems are kitchen, floor, administration and clock in/out devices and each subsystem have their respective architecture.

- Model View Controller(MVC) architecture for mobile devices such as waiter's tablets and desktop app.
- Client/Server architecture for web app and kitchen/waiter intercommunication for meal orders.
- Central repository database architecture(Admin) for keeping a log of data such as clocking in/out times, daily sells, and other essential data.

Although every system device shares same server in other to maintain communication, each subsystem is independent of other subsystems. Different architectural decision was made to target different aspects of challenges a restaurant might face.

UI Requirements

UI REQ-1	The first screen of the system will be the login screen.
UI REQ-2	The tablet interface will be used by waiters/waitresses and will be focused on taking orders.
UI REQ-3	The tablet has a table button to show the table layout of the restaurant. User can see table availability and be able to select the tables that are his/her responsibility.
UI REQ-4	Once a table is selected, a menu will open up where menu items that the customer requests can be selected to add to the customer order.
UI REQ-5	Once the order is final, it can be sent to the kitchen by selecting the place order button. When placing the order, the cost of the meal is also displayed and stored.
UI REQ-6	The tablet has a notification system to notify users when meals are ready to be sent from the kitchen to table.
UI REQ-7	The tablet has a notification system to notify users when tables need to be cleaned.
UI REQ-8	The tablet has a login system to associate each tablet with their respective user.
UI REQ-9	If a manager logs in, the screen presented to him/her will have more functionality such as options to view transaction reports, employee information as well as change employees schedules and make changes to the menu
UI REQ-10	The interface on the kitchen side will allow cooks to view orders and an option to notify the waiter/waitress when a dish is ready.
UI REQ-11	The website will allow customers to view the available reservations calendar and subsequently to create a reservation with a phone verification step included.
UI REQ-12	The website will have basic restaurant information such as the phone number, location, and hours of operation
UI REQ-13	The website will have an interactive menu where pictures of each item are included

Functional Requirements Specification

Stakeholder

Several types of stakeholders exists in our system from restaurants who wish to adopt our system to the actual workers who will follow the new system in it. The system will help speed up efficiency with organization utility for the staff. Those who will specifically be working with the system would be hosts, managers, waiters, kitchen staff and busboy. In addition, customers However customers would be another stakeholder who does not directly make use of the system other than through the website.

Actors and Goals

Initiating Actors:

Busboy		Chef	
Role	Employee who keeps cleanliness of the restaurant	Role	Employee who prepares meals and maintains the kitchen
Goal	To help maintain table status and clean tables marked as dirty	Goal	Manage Inventory of ingredients, manage menu and prepare ordered meals
Customer		Manager	
Role	A patron who orders a meal at the restaurant	Role	Employee who manages the other employees at the restaurant
Goal	To order, eat and pay for a meal	Goal	Manage scheduling, tracking of profits/losses and inventory management
Host		Waiter	
Role	Employee who greets and sits down customers	Role	Employee who interacts with and serves customers
Goal	Seats customers to an appropriate table and assigns waiters to table	Goal	To take orders from table to kitchen and serve the food back to table, also handles customer’s payment

Participating Actors:

Database

Role: To store all information needed for the restaurant system.

Goal: Store, modify and query needed data for Actors to perform their roles.

Employee Tablets

Role: To enable interoperability between the database and waiters.

Goal: Retrieve data from appropriate Actors during their roles.

Main Desktop

Role: Central system to interact with database, tablets and website.

Goal: To have access and control of main systems of restaurant automations, to be controlled by admins (managers).

Website

Role: Provide information about the restaurant to Customers

Goal: Make it easier for customers to find the menu, hours and open reservations.

Use Cases (Casual Description)

Use Case	Name	Description
UC-1	Order	Waitress can create an order of menu items for customers and send it to the kitchen via mobile application
UC-2	Serve	Cooks can electronically notify waitress of when their meal orders are ready to be served
UC-3	Seat	Host/waitress can queue up patrons and seat them to a table using an electronic diagram of the tables to quicken the process
UC-4	Reservation	Customers can use the website to setup reservations for when they visit the restaurant. Hosts can then confirm upon visit
UC-5	Clock in/out	Any Restaurant staff member can log their attendance of work
UC-6	Admin	Managers can have full access to restaurant software functionality(change menu items, check reservations, check worker attendance, check receipts, etc)

Use Cases Full Descriptions

UC-1: Order

Related Requirements:	REQ3, REQ8, REQ10
Initiating Actors:	Any of: Waitress, Cook, Manager
Actor's Goal:	To place the order of the customer as fast as possible, and make sure the cook gets the order correctly
Participating Actors:	Database, Employee Tablets
Preconditions:	*The set of valid keys (orders) in the system is initially 0 *The system queues each of the orders in chronological order and lists the orders made in groups as an array
Postconditions:	*The system starts dequeuing the orders as the cook serves them out and the waitress approves of the order
Flow of Events for Main Success Scenario:	1. Customer enters the restaurant and selects from the menu via an 'order' option 2. System via waitress a) signals to cook to begin order b)signals to manager to oversee order 3. Cook creates order and signals to waitress to retrieve order

Extracting the Attributes

Concept	**Attributes**	**Attribute Description**
Communication	Order request	Used to send orders to the kitchen.
Communication	Order display	Used to display order list on the screen.
Notifier	Tracking order	To display list after updating orders.
Archiver	Tracking number	To stored record of orders in database.

Use Case UC-2: Serve

Related Requirements:	REQ4, REQ6, REQ7
Initiating Actors:	Any of: Waitress, Cook
Actor's Goal:	To serve the order from the kitchen to the respective customer's table as fast as possible
Participating Actors:	Database, Employee tablets
Preconditions:	*The set of valid keys (orders) in the system is NOT zero *The set of valid keys are dequeued in chronological order to be served to the respective customers
Postconditions:	*The set of valid keys at the end of the day is zero (all orders have been completed)
Flow of Events for Main Success Scenario:	1. Cook receives order from waitress and produces meal 2. System notifies a)waitress that the meal is prepared and ready to serve 3. Waitress serves meal to respective customer's table 4. System signals to waitress when a customer has completed their meal 5. Waitress serves bill to respective customer's table

Extracting the Attributes

Concept	**Attributes**	**Attribute Description**
Communication	Order state	To notify waiter when order is ready.
Notifier	Tracking order	Used to show which order is ready to go.
Archiver	Tracking number	To add order in database after is being successfully delivered.

Use Case UC-3: Seat

Related Requirements:	REQ1, REQ5
Initiating Actors:	Any of: Hosts, Waitress
Actor's Goal:	To seat the waiting customers as fast as possible
Participating Actors:	Database, Employee tablets
Preconditions:	*The set of valid keys (tables) in the system is initially 0 *The system queues each of the customers in chronological order and lists the orders made in groups as an array
Postconditions:	*The system starts dequeuing the orders as the waitress begins seating them
Flow of Events for Main Success Scenario:	1. Customer enters the restaurant and selects the 'seating' option 2. Host enters customer party onto the database queue 3. Database calculates the party size with the seating options to optimize availability. 4. The system notifies the host that a table is available so they can direct the party to the table. 5. After seating the party, the host confirms to the system that the party has been seated. 6. The database then moves the party to seated party list

Extracting the Attributes

Concept	**Attributes**	**Attribute Description**
Search request	Table availability	To find which tables are available.
Investigating request	Records list	To query database to process search request.
Communication	Table state	Used for communication between hostess and waiters.

Use Case UC-4: Reservation

Related Requirements:	REQ 2, REQ 14, REQ 15
Initiating Actors:	Any of: Waiters, Managers, Hosts, Customers
Actor's Goal:	To create and manage restaurant reservations
Participating Actors:	Database, Employee tablets, Website
Preconditions:	*The desired table is not reserved by another person for the desired time. *The restaurant is open at the desired time *The Actor must have permission to manage a reservation after it has been created. Therefore, a customer can not delete a different customer's reservation but a manager could.
Postconditions:	*The reservation doesn't interfere with any other reservation and prevents other reservations from being made at this table and time. *Reservation has verified contact information attached to it
Flow of Events for Main Success Scenario:	1. Customer goes on website to the reservations page 2. Customer finds open reservation 3. Customer enters email or phone to verify their identity

Extracting the Attributes

Concept	**Attributes**	**Attribute Description**
Archiver	Update reservation	Used to make any changes on a reservation.
Archiver	Make reservation	Prompts the user to make a reservation.
Search request	Records list	Used to display restaurant information.

Use Case UC-5: Clock in/out

Related Requirements:	REQ 9
Initiating Actors:	Any of: restaurant staff
Actor's Goal:	To keep a log of employees attendance.
Participating Actors:	Database system: To store and keep records of each staff member past working hours.
Preconditions:	*Fingerprint recognition constraint in order to clock in and out by staff member.
Postconditions:	*Clocked in/out time log and storage.
Flow of Events for Main Success Scenario:	1. Employee hits clock in/out button. 2. Employee places finger for fingerprint recognition. 3. Clock in/out granted. 4. Time is recorded and given to respective staff member

Extracting the Attributes

Concept	**Attributes**	**Attribute Description**
Search request	User's identity	Prompts user to enter credentials.
Investigating request	Records list	Used to query database for credentials matching.
Archiver	Log time(in/out)	Used to send information to database server in order to be saved.
Notifier	Display time(in/out)	Displays user's time in or out on the screen.

Use Case UC-6: Admin

Related Requirements:	REQ 10, REQ 11, REQ 12, REQ 13
Initiating Actors:	Any of: Management personal(Chef, head manager, supervisors)
Actor's Goal:	To make management changes such as menu, staff schedules, payments, orders.
Participating Actors:	Database, Main Desktop
Preconditions:	*Login as administrator. *Username and password.
Postconditions:	*Save any changes made to database system.
Flow of Events for Main Success Scenario:	1. Manager enters to administration system. 2. Manager enters username and password. 3. Changes may be made. 4. Changes may be saved.

Extracting the Attributes

Concept	**Attributes**	**Attribute Description**
Archiver	Set orders	To fix any discrepancies in a particular order.
Archiver	Set schedules	Used to upload staff working schedules.
Search request	Get data	To get lists of interesting records selected for further investigation.
Notifier	Table status	Used to estimate how long a table has been occupied.

Traceability Matrix

System Requirements to Use Cases

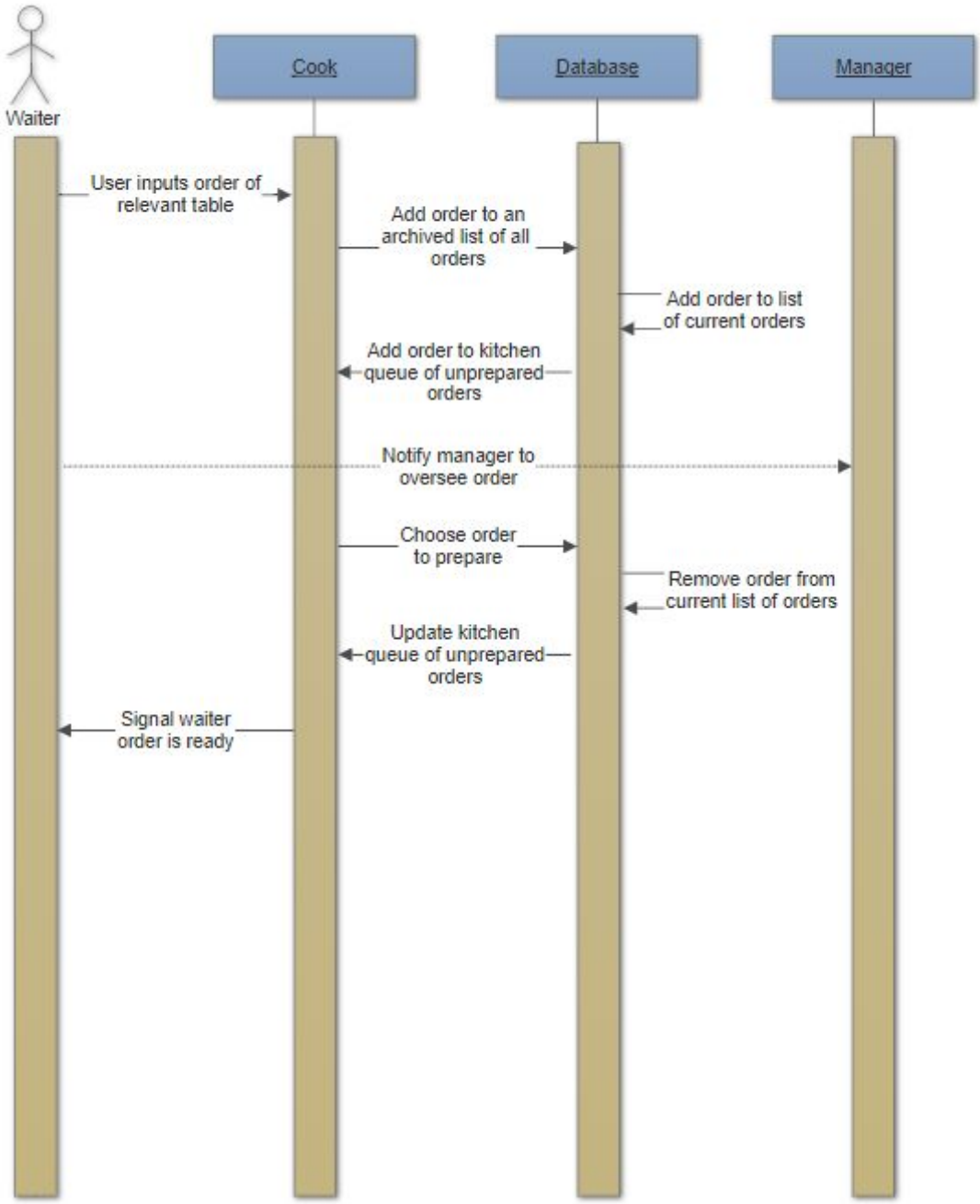
Requirements	**PW**	**UC1**	**UC2**	**UC3**	**UC4**	**UC5**	**UC6**
REQ1	4			*			
REQ2	2				*		
REQ3	6	*					
REQ4	4		*				
REQ5	3			*			
REQ6	2		*				
REQ7	4		*				
REQ8	4	*					
REQ9	4					*	
REQ10	3	*					*
REQ11	4						*
REQ12	4						*
REQ13	3						*
REQ14	3				*		
REQ15	2				*		
Max PW		**6**	**4**	**4**	**3**	**4**	**4**
Total PW		**13**	**10**	**7**	**7**	**4**	**14**

Use Cases to Domain Model

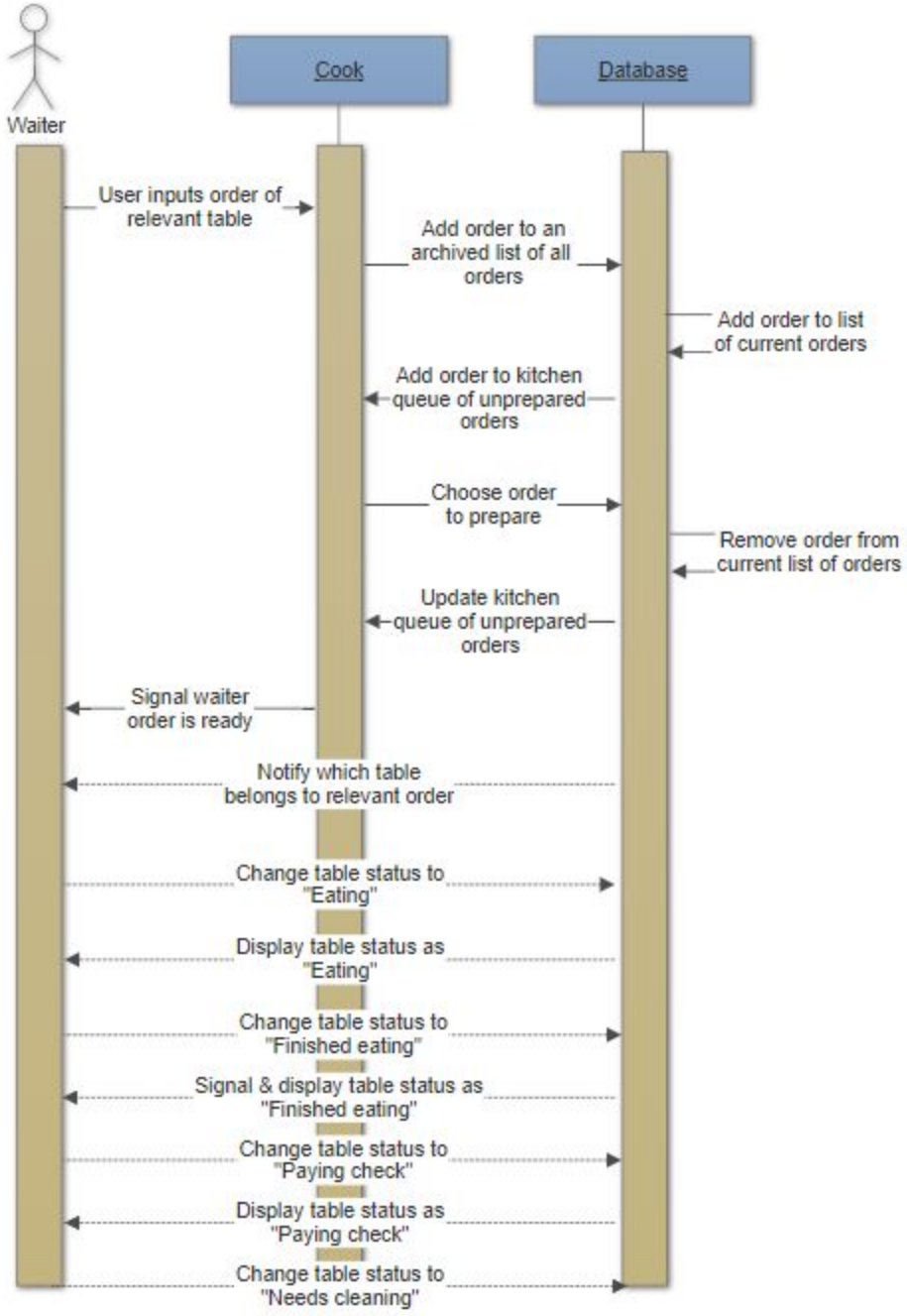
Use Case	**PW**	Communication	Search request	Investigating request	Archiver	Notifier
UC1	13	*			*	*
UC2	10	*			*	*
UC3	7	*	*			
UC4	7		*		*	
UC5	4		*	*	*	*
UC6	14		*		*	*

System Sequence Diagram

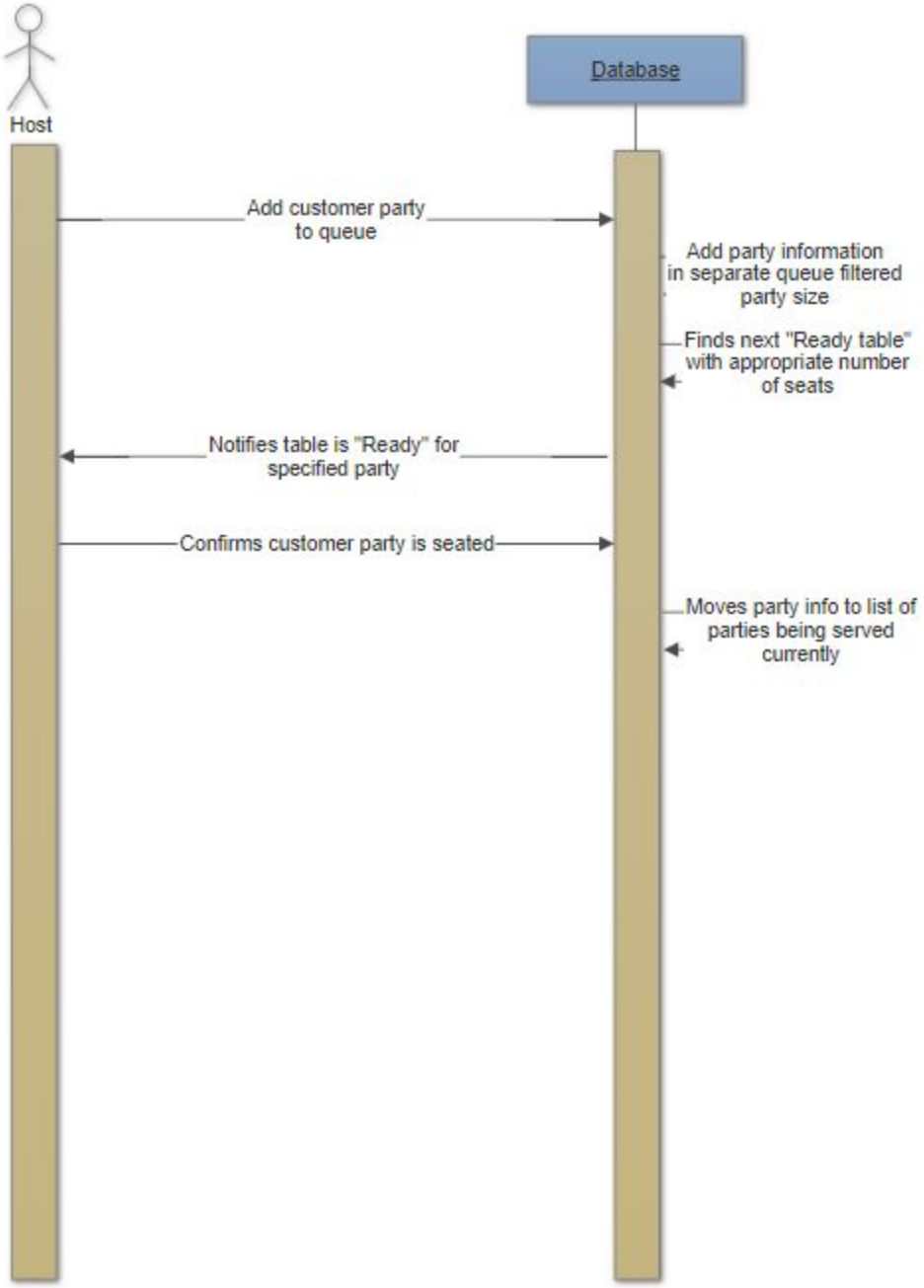
Use Case 1: Order



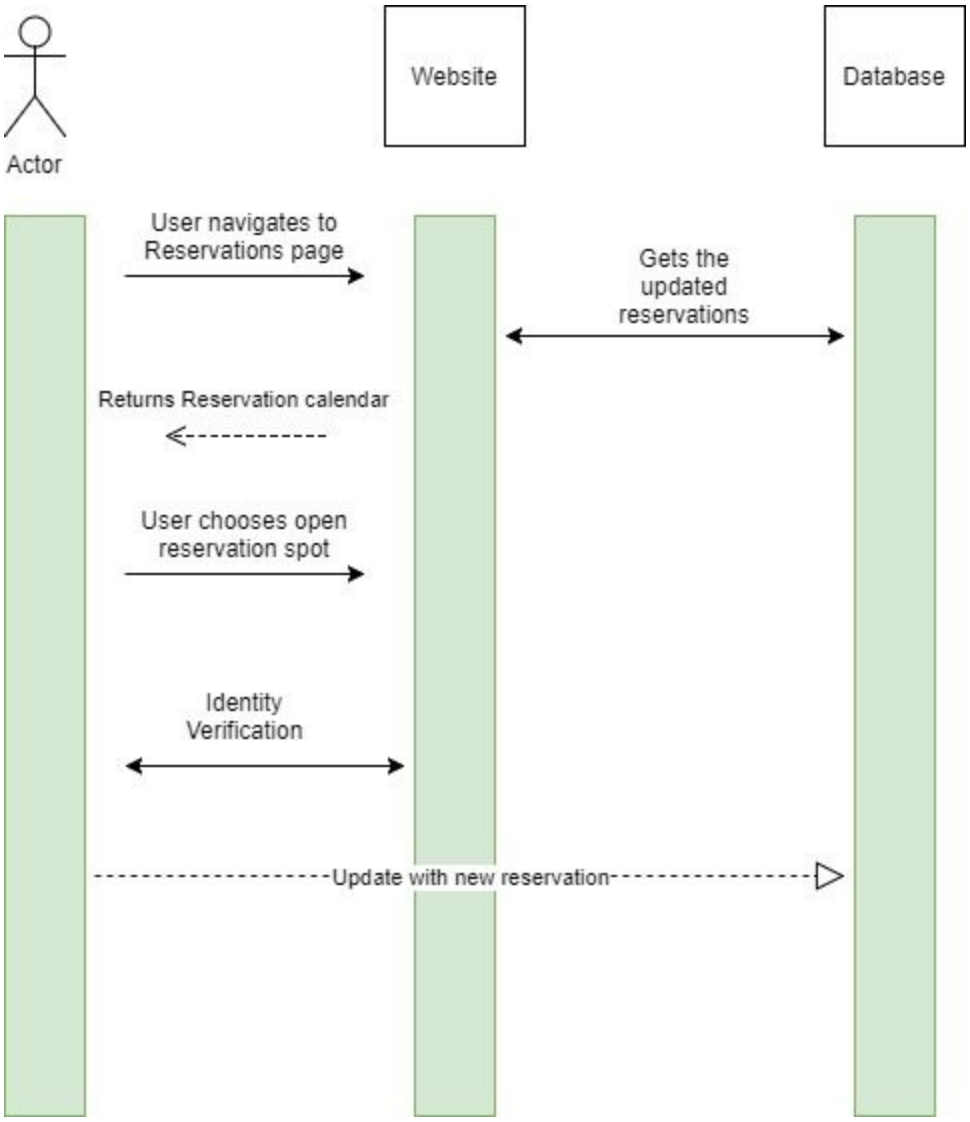
Use Case 2: Serve



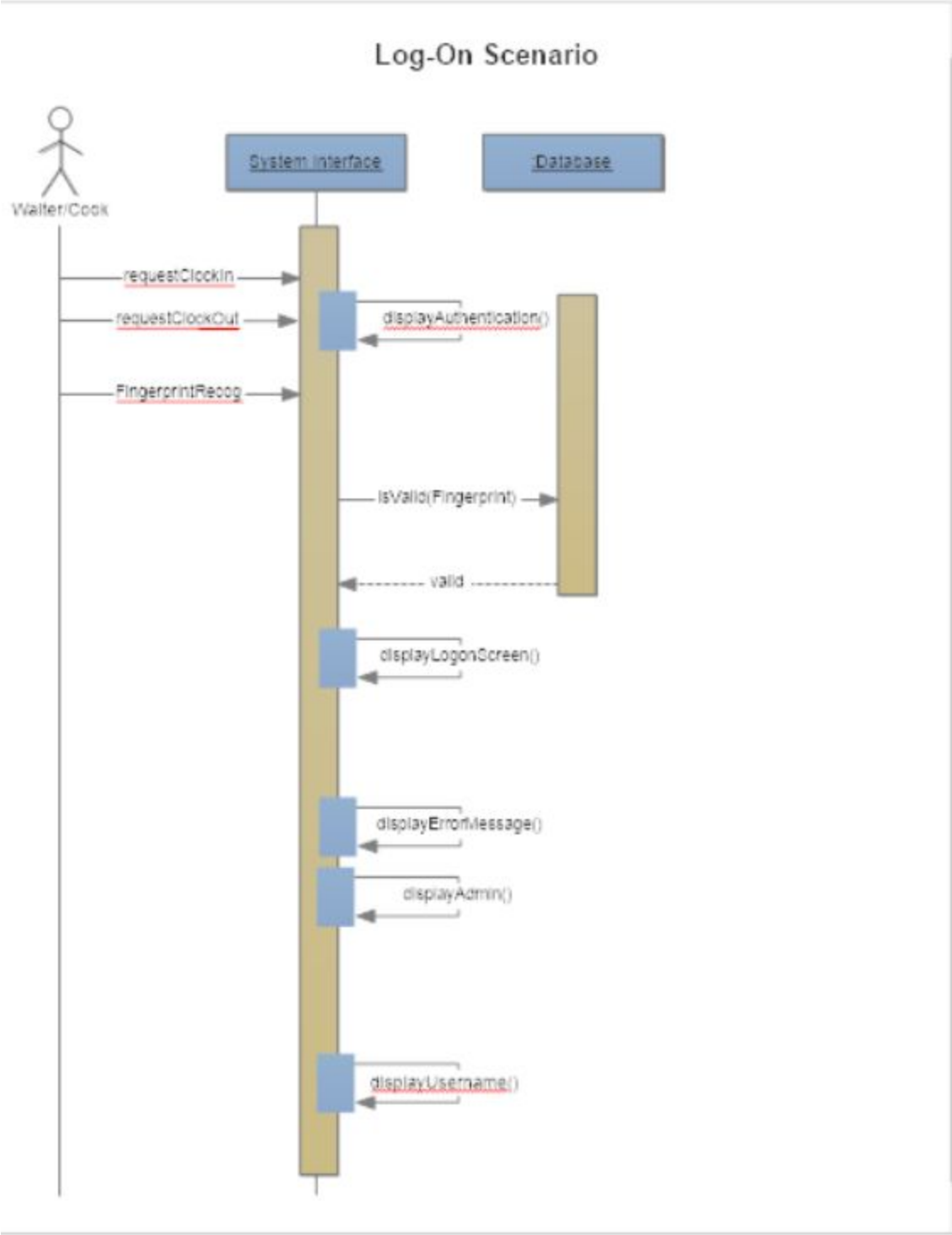
Use Case 3: Seat



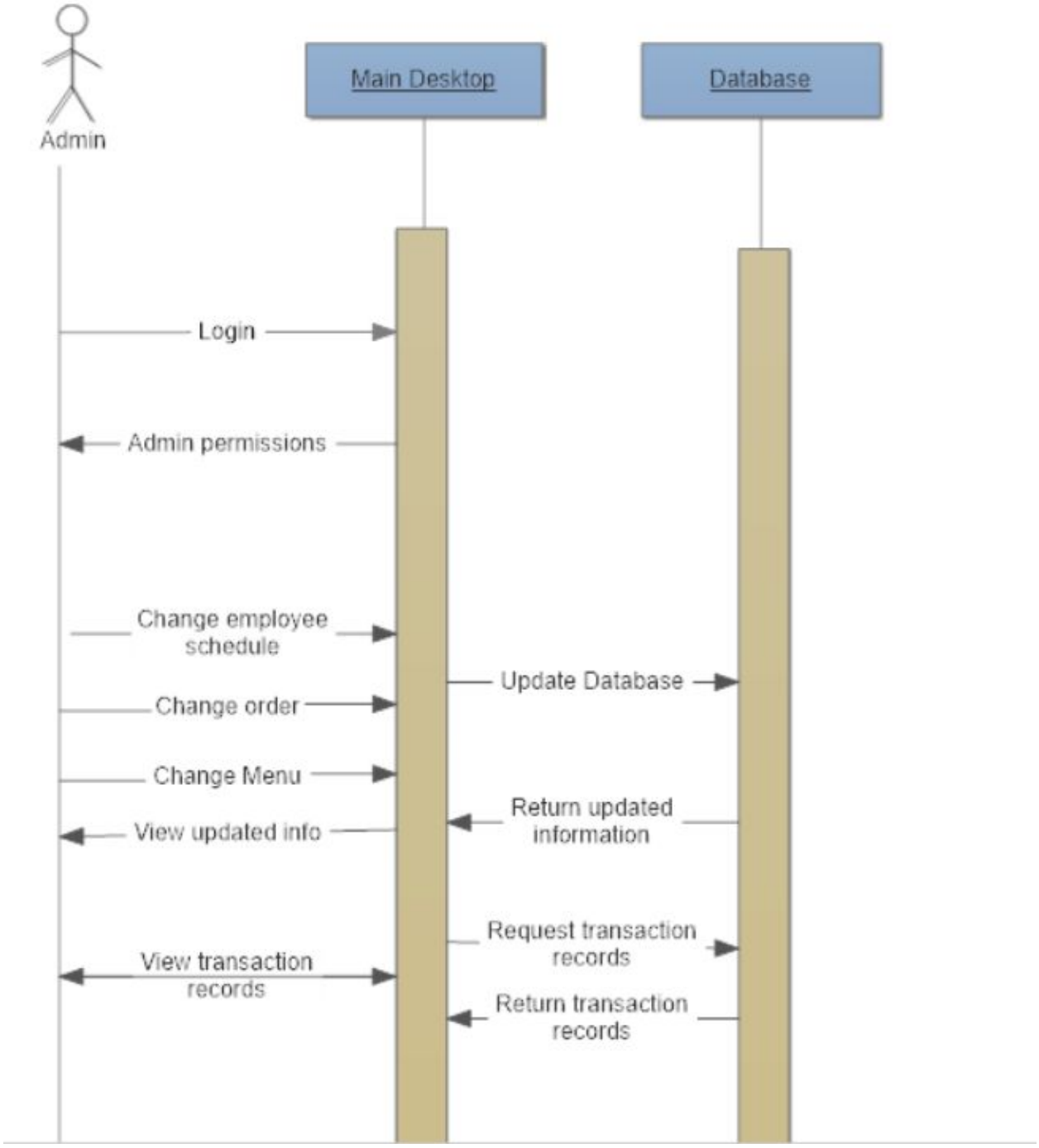
Use Case 4: Reservation



Use Case 5: Clock In/Clock Out

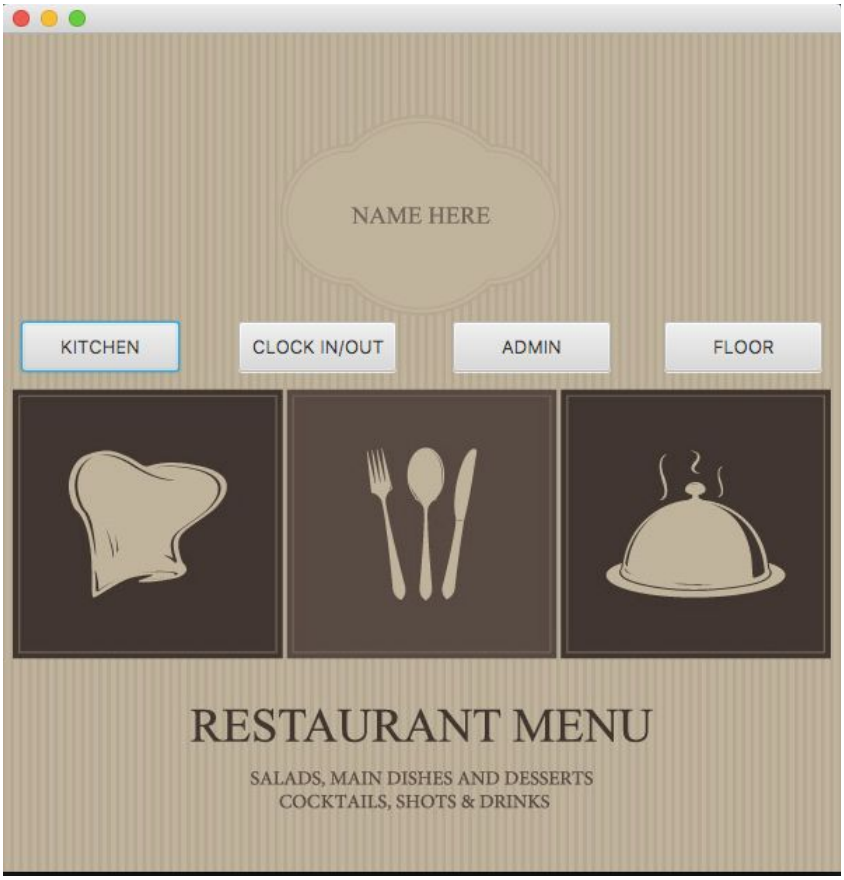


Use Case 6: Admin



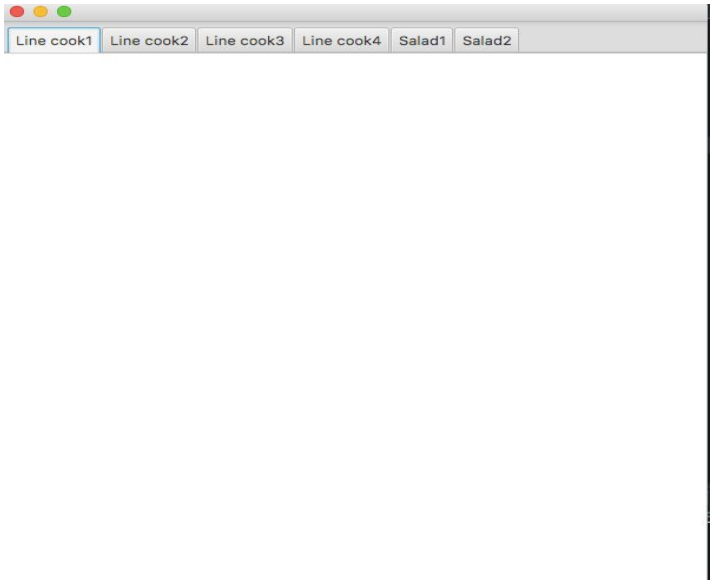
User Interface Specification

Preliminary Design
Restaurant Desktop GUI



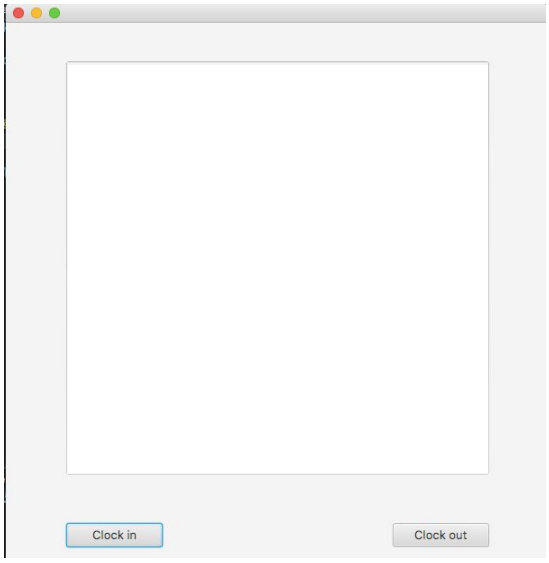
This picture is the current main page of the desktop restaurant automation application. From here the user can either enter the Kitchen, Clock In/Out, Admin, or Floor window.

Kitchen



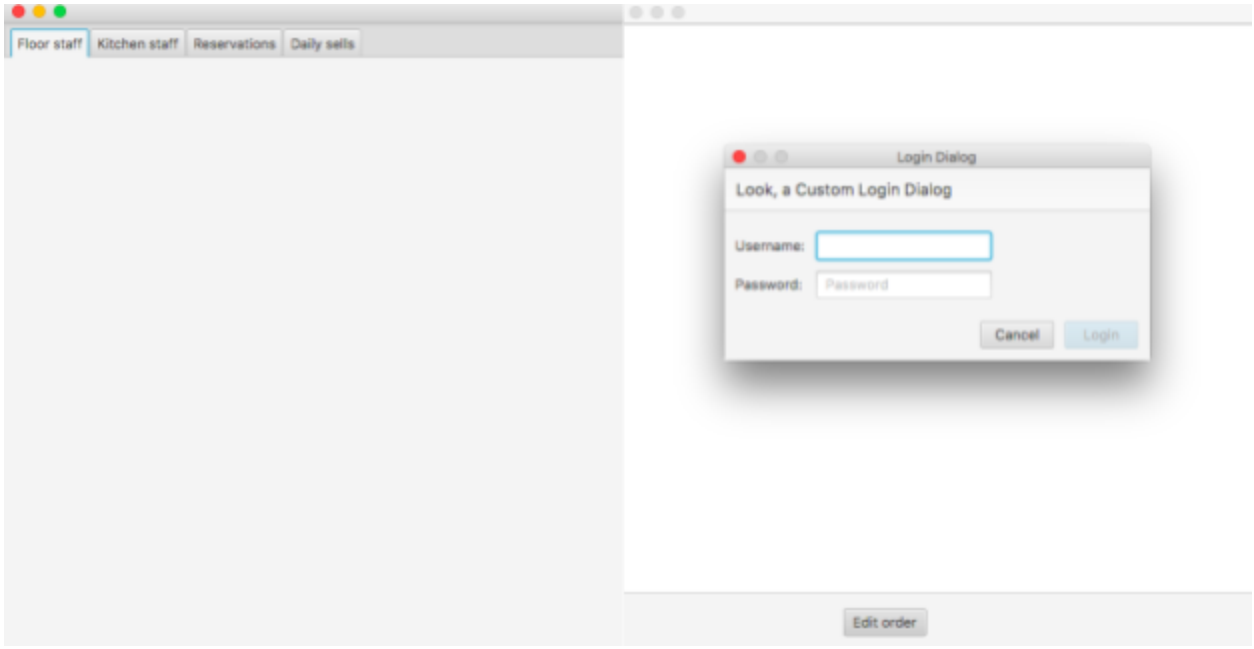
This is the current Kitchen window. Here line cooks have their own tab that will populate with meal orders as their are sent from the waitress.

Clock in/out



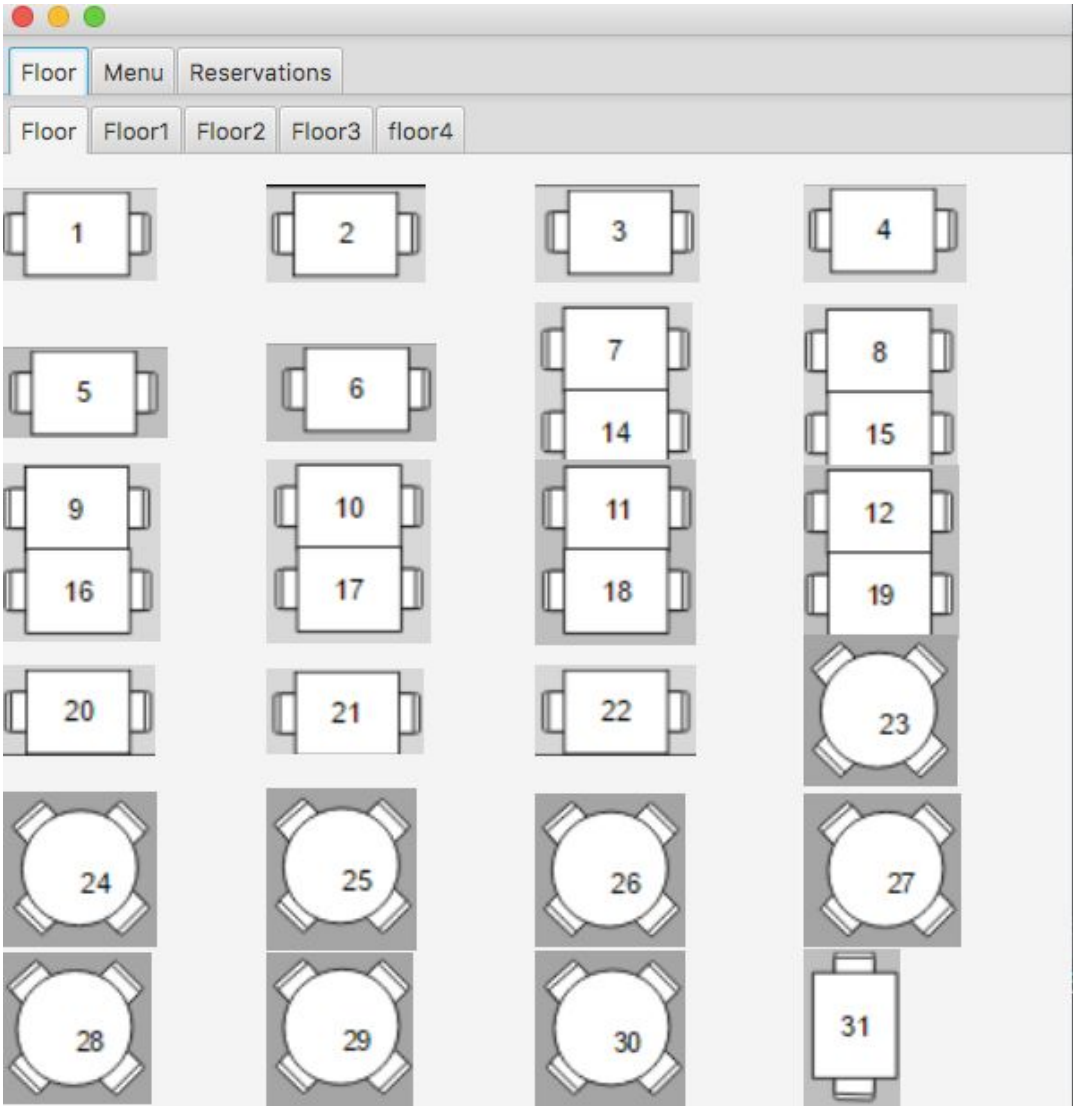
This is the current Clock in/out window. Here employees can log their attendance for work.

Admin



This is the current Admin window. Here users will be prompted to login so that only managers can enter the information and functions available through this page.

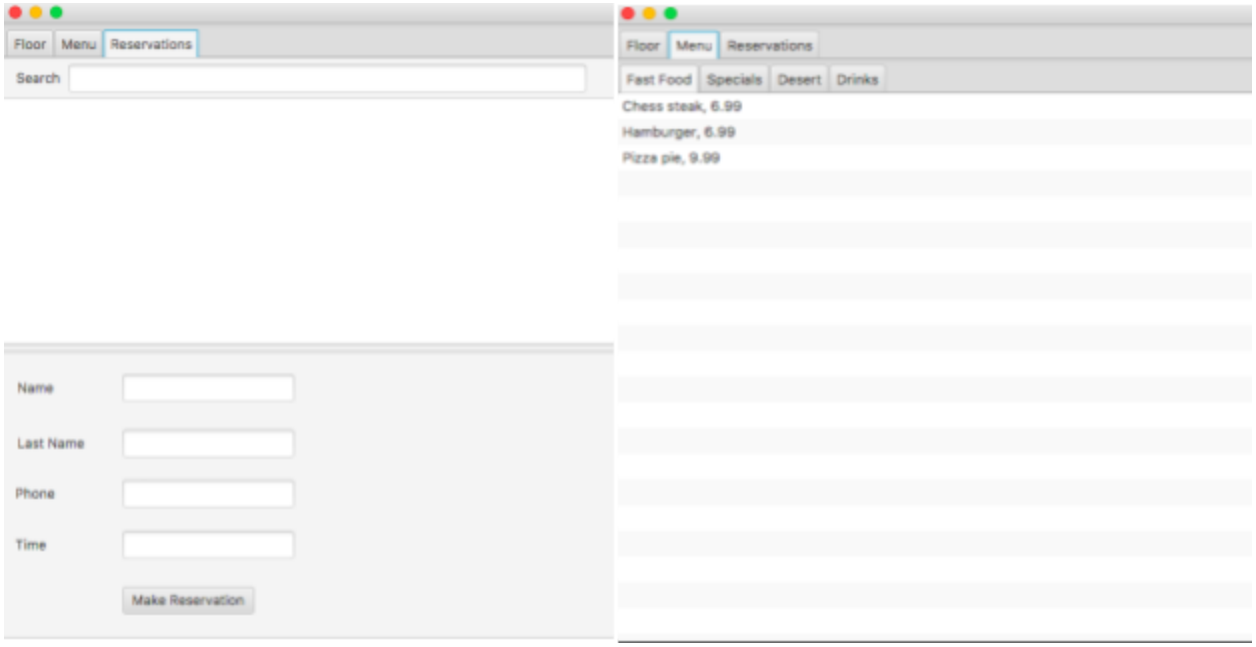
Floor



Here is the Floor menu page. Here users can check the table status of all the tables in the restaurant. In the following images below, green represents ready, blue represents dirt, and red represents occupied.

Table State (green: ready, blue: dirty, red: occupied)

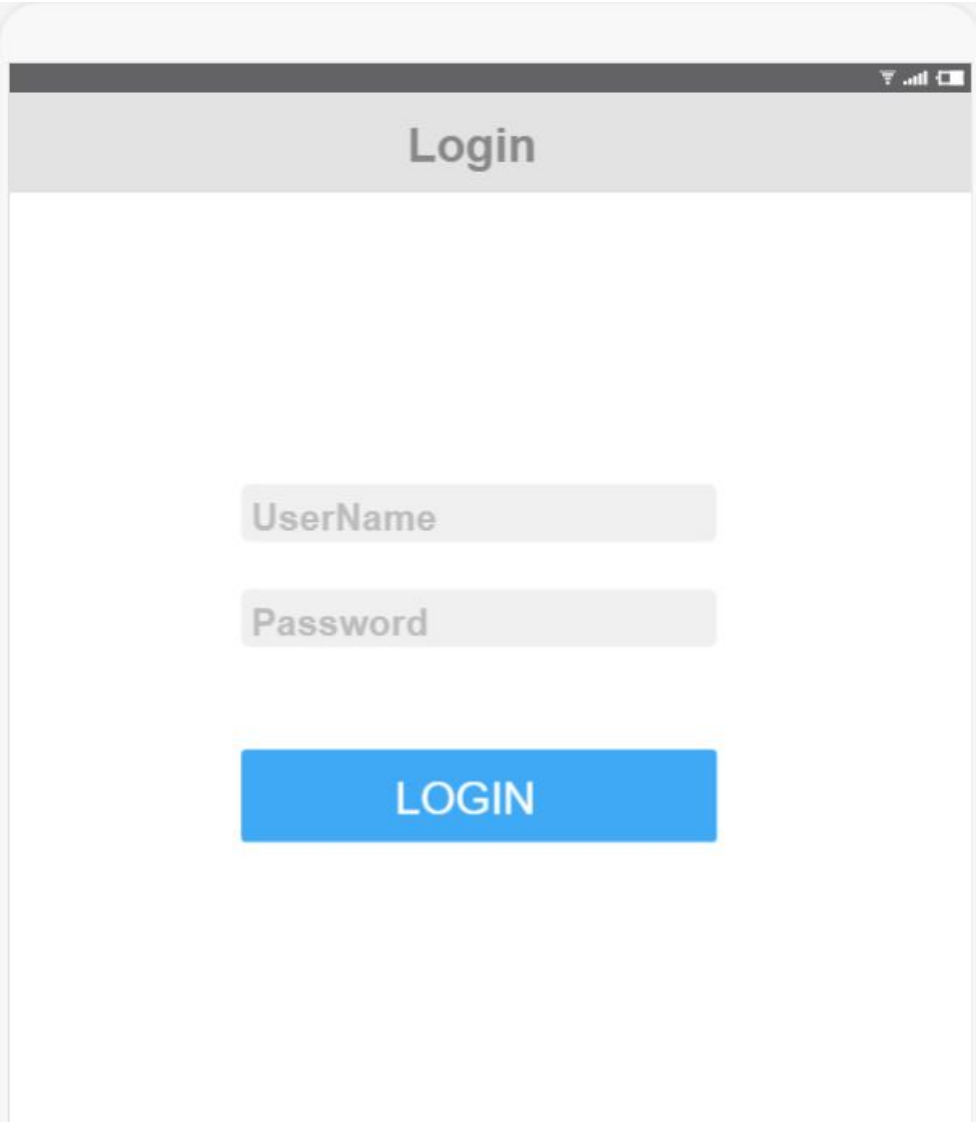
Reservation and Menu Interface.



This is the Reservation and Menu Interface window. For reservations, users can either look for an already made reservation or create a new one. For Menu Interface, users can enter new menu items or update already existing menu items.

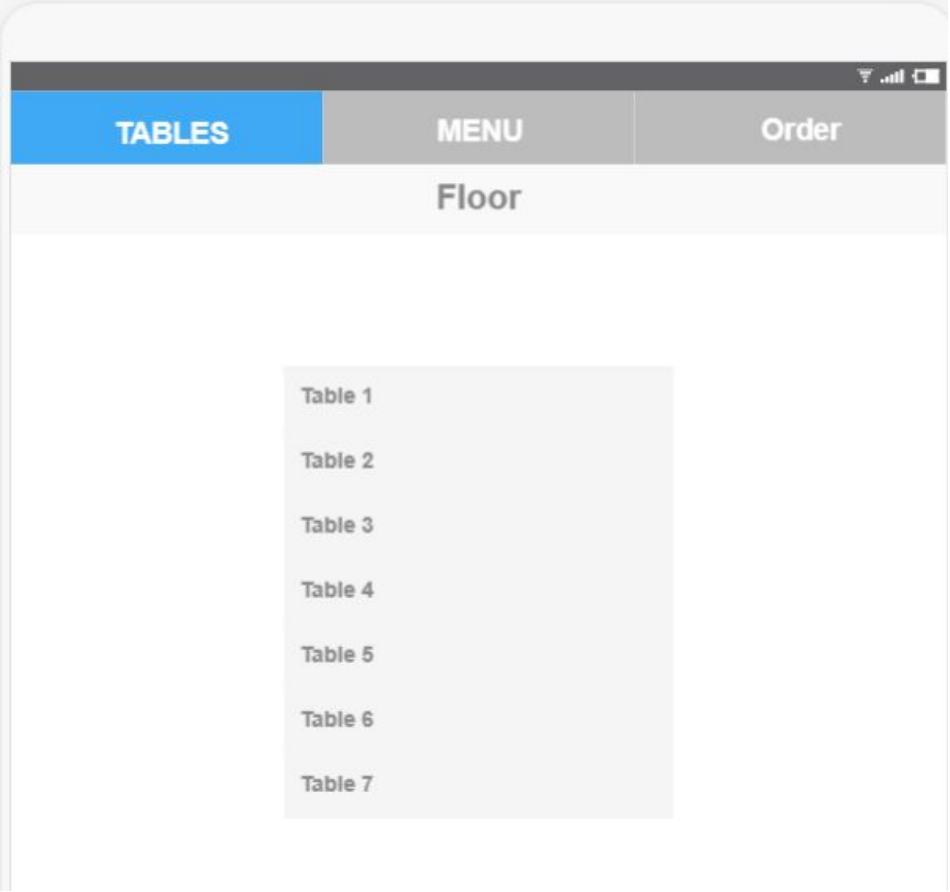
Mobile Application UI Diagrams

Mobile App Login Page



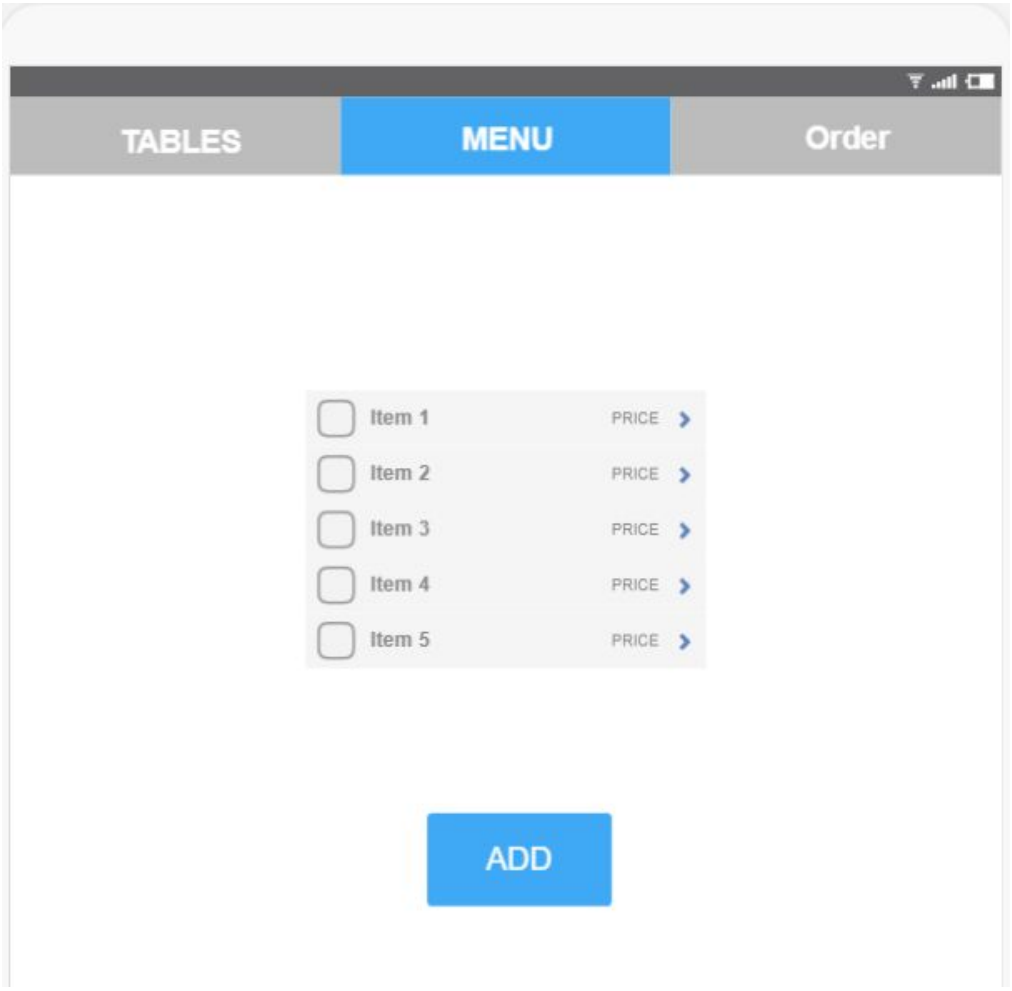
This is the Mobile Application login page. Here Waitresses can login to begin taking orders. By using a login system we can secure the application to only allowing users with access to order items.

Mobile App Tables Page



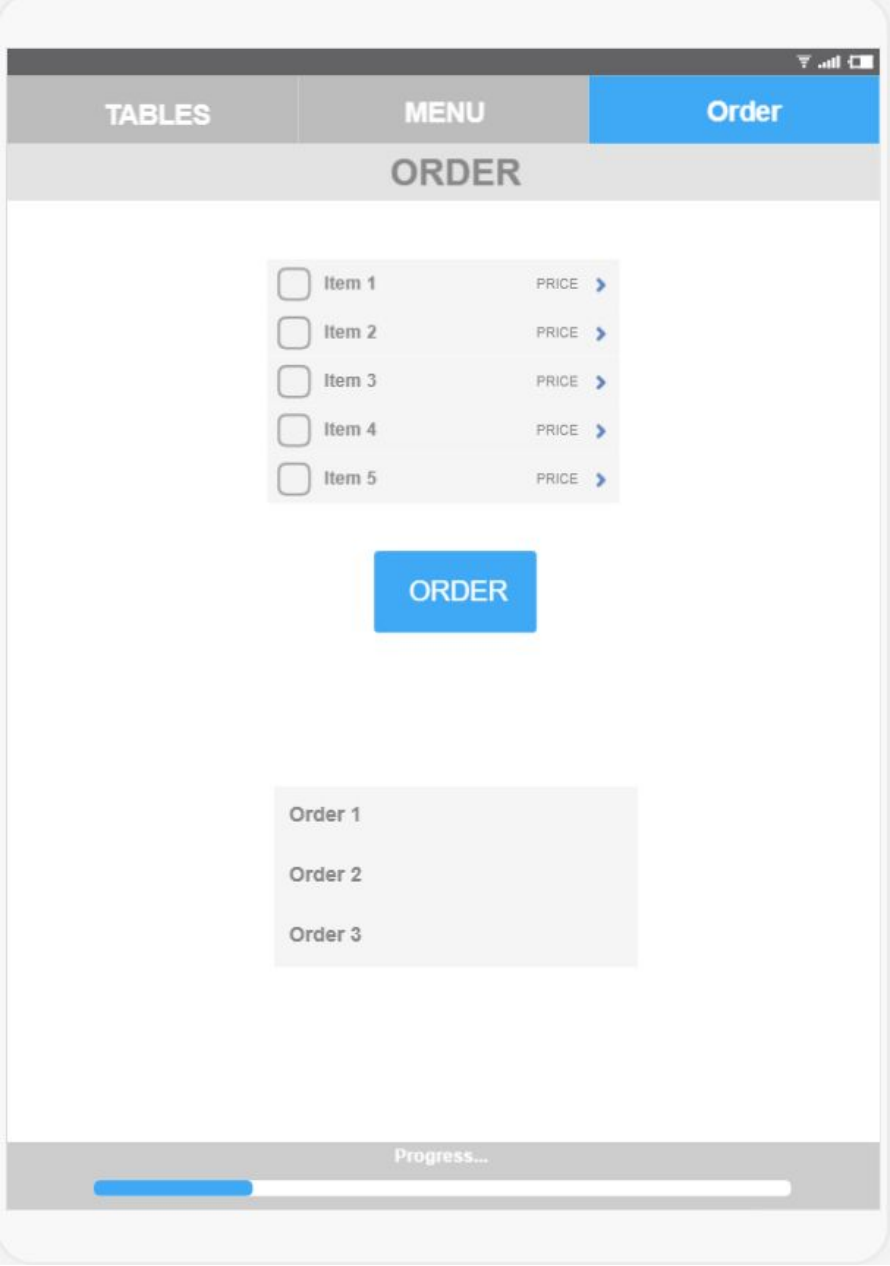
The Mobile App Tables Page will have a drop down list for the floor selection. There users can choose which floor they will be servicing from a drop down list. Under that element is a List View of the tables on the floor. Each table item will be clickable and will lead into the Menu Page.

Mobile App Menu Page



The Menu page has a list view of all the menu items available with the the price on the side. The check box allows the user to select the items and the ADD button will add these items to a order list in the Order page.

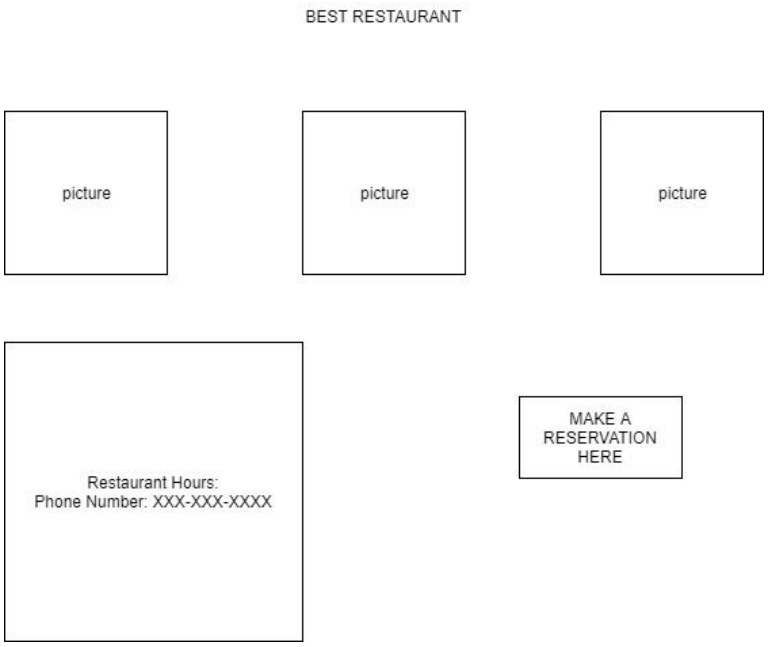
Mobile App Order Page



The Order page allows users to see what items are queued up before sending the order to the kitchen. The top list view will be populated with menu items that were selected and added from the menu page. The ORDER button allows the user to send the order list to the Kitchen electronically. The bottom listview will be populated by the orders already sent to the Kitchen. Clicking on any of the order items from this list will show the progress of it on the bottom of the page through the Progress bar.

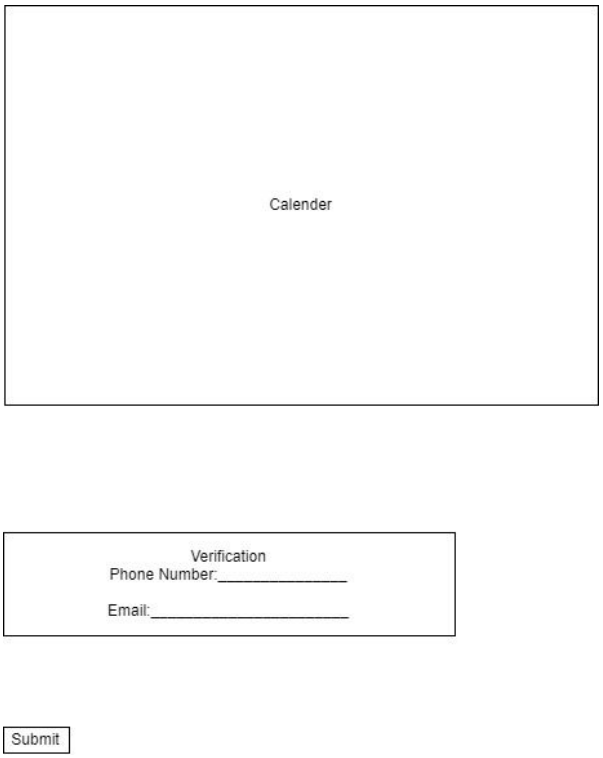
Website

Homepage



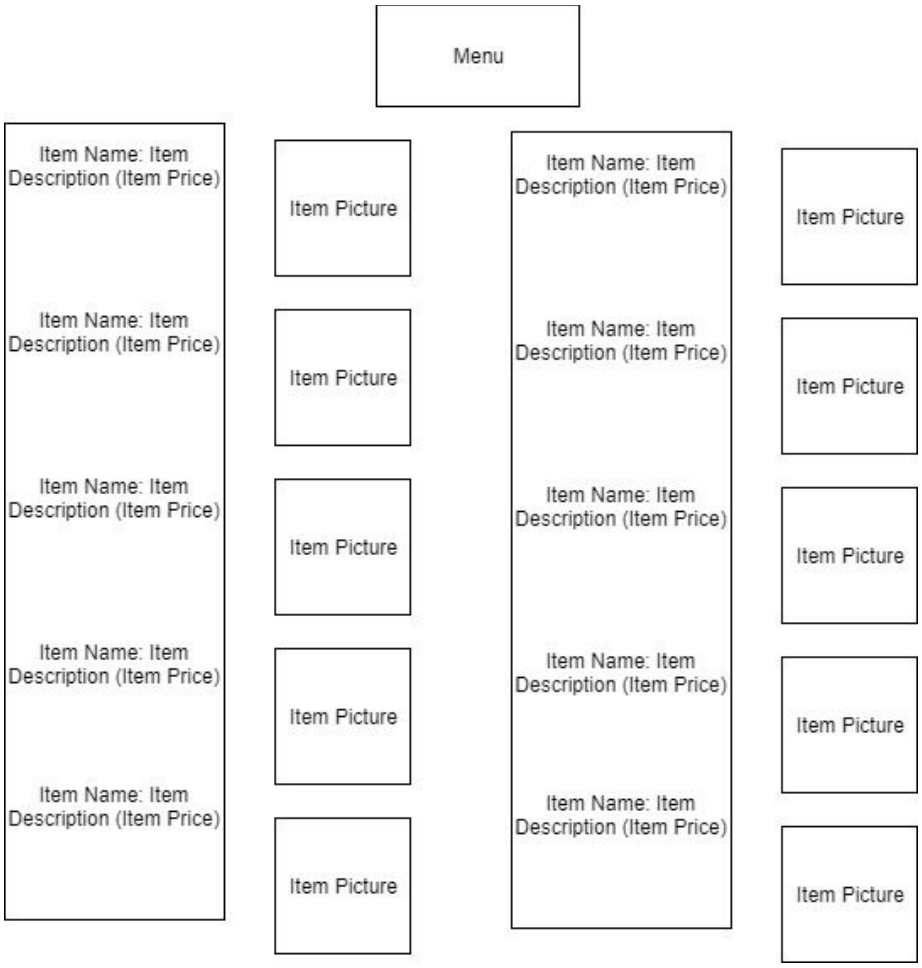
The homepage contains some general information about the restaurant. There are three pictures at the top to showcase some of the restaurant's decor. There are also the restaurant's hours and the phone number. There is a button that directs you to the reservations page.

Reservation Page



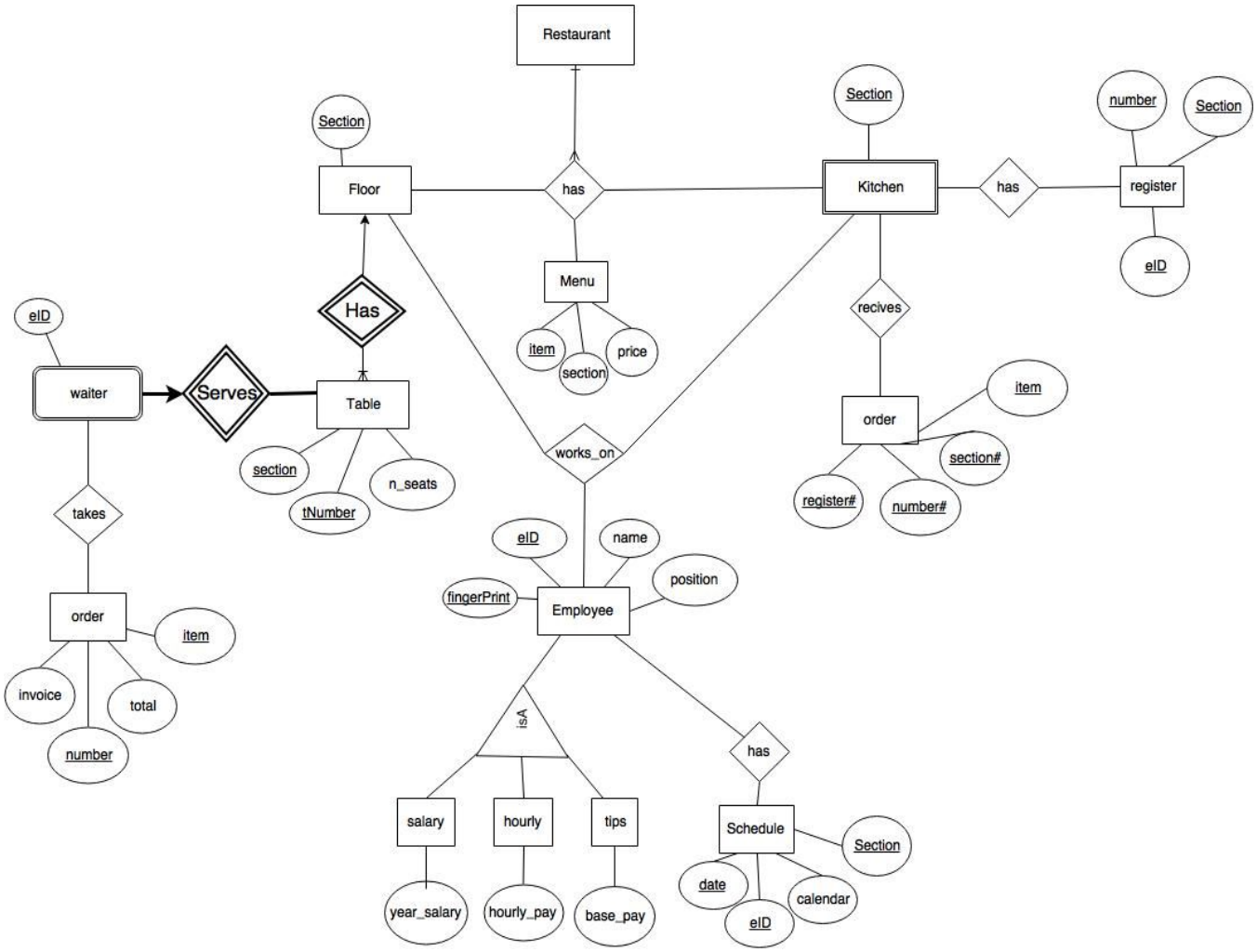
A calendar of all the dates is shown. The availability is shown on the days in the calendar. Then a verification entry is needed, whether phone number or email.

Menu



The menu page contains all the menu items with their descriptions, name, price, and picture.

Restaurant Automation ER diagram



This is the entity relation database system diagram created for the Restaurant Automation System. Here you can see the classes, functions, objects and how they work together.

User Effort Estimation

Scenario 1: Waiter Places Order

1. Navigation: From main activity press Menu button (3 taps minimum)
    a. Scroll up and down to find desired item
    b. Tap once to add to order, tap again to add another
    c. Once all desired items are added to order, press Order button to send to kitchen

Scenario 2: Host Seats Patrons

1. Navigation: From main window, enter table status tab (3 taps)
    a. Search through floors for available tables by pressing floor tabs
    b. Select table by clicking on desired table and setting name to patron's name
    c. Select an available waiter from list to service patron's party

Scenario 3: Busboy Cleans Table

1. Navigation: Select the option to look at the table view (3 taps)
    a. Waiter/busboy selects a table that has status called "needs to be cleaned" and clicks it to change the status to "cleaning table"
    b. Once the waiter/busboy is done cleaning the table, the waiter/busboy clicks the same table again to change the status to "ready table"

Scenario 4: Chef Prepares Order

1. Navigation: Select the option to go to the item queue (4 taps)
    a. Chef chooses an order from the list to start preparing. The queue is listed in order of when the item was inputted by a waiter, alongside all appropriate ingredients for the items (depending if customer changes ingredients)
    b. Chef selects "begin preparing" option and starts to prepare the order
    c. Chef selects "complete" option when the order is ready

Scenario 5: Customer makes reservation online

1. Navigation: Select the "Make Reservation" button (3 clicks, 3 info entries)
    a. Customer clicks the reservation page button
    b. Customer enters their information
    c. Customer presses submit

Unadjusted Use Case Points:

Use Case Weight

Type	**Description**	**Weight Amount**
Simple	One participating actor. 3 or less concepts	5
Average	Two or more participating actors 4 concepts	10
Complex	Three or more participating actors 4+ concepts	15

Use Case	**Use Case Weight**
UC-1: Order	15
UC-2: Serve	10
UC-2: Seat	10
UC-4: Reservation	5
UC-5: Clock in/out	5
UC-6: Admin	15

UUCP = 60

Technical Complexity Factor:

Technical Complexity Factor	Technical Complexity Factor Weight	TCF perceived complexity	Complexity Factor
1 - Distributed System	2	4	8
2 - Performance	1	4	4
3 - Ease of use	.5	5	2.5
4 - User Efficiency	1	4	4
5 - Ease of install	.5	3	1.5
6 - Reusability	1	0	0
7 - Portability	2	2	4

TCF = 0.6 + 0.01(Total Complexity Factor)

TCF = 0.6 + .24

TCF = 0.84

Environmental Factor:

Environmental factor	Weight	Perceived Impact	Complexity Factor
1 - System Familiarity	1.5	4	6
2 - Work Experience	.5	5	2.5
3 - Motivation	1	3	3
4 -	2	1	2
5 - Part-time staff	-.5	3	-1.5
6 - Language barrier	-1	2	-2

ECF = 1.4 + -.03*(Total Environmental Complexity Factor)

ECF = 1.4 + -.03*(10)

ECF = 1.1

User Complexity Points & Duration

UCP = UUCP * TCF * ECF

UCP = 60 * 0.84 * 1.1

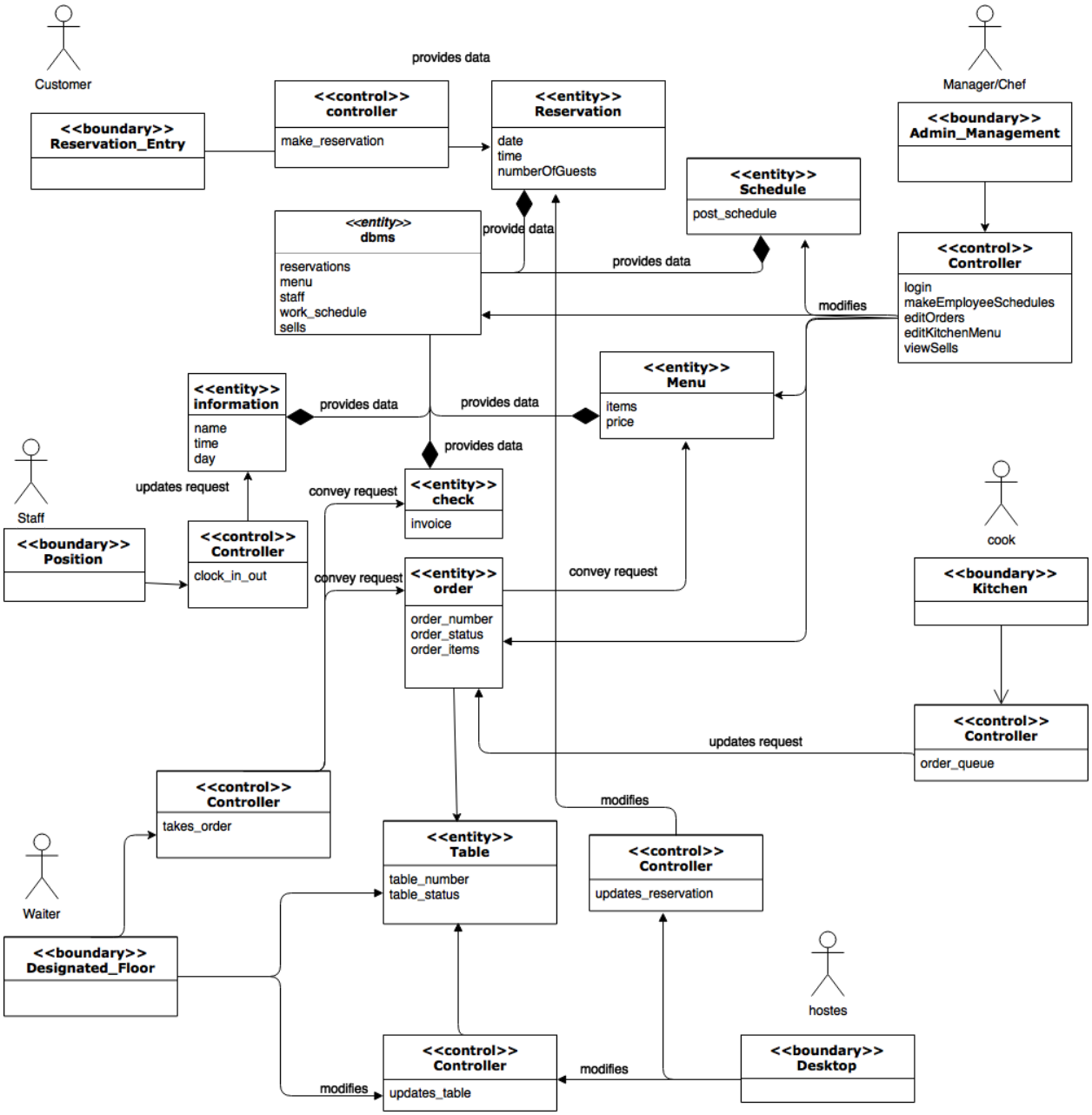
UCP = 55.44 = 55 use case points

Duration = UCP * Productivity Factor

Duration = 55 * 24 = 1320 Hours

Domain Analysis

Domain Model:



Concept Definitions:

C	Concept	Type	Responsibility Description
C1	Employee	D	Staff working at the restaurant that is registered to the system
C2	Manager	D	Manages the employees time schedules and work tasks
C3	Host/Hostess	D	Greets and seats the guests at their tables. Manages the incoming reservations as well.
C4	Waiter/Waitress	D	Takes orders from customers and sends them in
C5	Kitchen	D	Prepares food and notifies waters when orders are ready
C6	Floor	K	System shows the floor occupancy
C7	Table	D	Waiter assigns to guests when occupied
C8	Menu	D	Displays the food options
C9	MenuItem	D	Kitchen updates the items in the menu
C10	tableStatus	K	System updates status when being used by guests
C11	MenuModifier	D	
C12	Reservation	D	A customer makes a reservation to assure a table at an agreed upon time.
C13	ReservationCalendar	K	Keeps track of all reservations
C14	ReservationModifier	D	Modify the reservation calendar
C15	OrderItem	D	System sends order to kitchen
C16	OrderQueue	K	Keeps track of all the unfinished orders
C17	Check	K	Keeps track of all items ordered at a table with associated price, subtotal, total, and calculated tips
C18	CheckModifier	D	Allows user to change any of the check's associated price, subtotal, total, and calculated tips. Allows to add and remove items from check

C19	Users	K	Keeps track of all employee users, passwords, and permissions
C20	UserChecker	D	Checks if the given username and password are valid, provides entryway for employees to system
C21	Clock	K	Displays current day and time for employees to check shifts
C22	ClockSchedule	K	Displays all weekly employee schedules with associated dates and times
C23	ClockIn	K	Keeps track of when employees start and stop work
C24	WebPageController	D	Central intermediary between InferfacePage and PageMaker, and all reservation web-page actions
C25	PageMaker	D	Collects information needed to create page interface
C26	InterfacePage	D	Acts as the GUI communicator between the user (customer) and system

ii) Association Definitions:

Concept Pair:	**Association Description:**	**Association:**
Manager ⇔ ClockSchedule	Manager assigns work schedule employees	Modifies
ClockIn ⇔ Employee	Provides data to system when employees clock in to work	Provides data
Manager ⇔ MenuModifier	Manager updates menu	Modifies
Host/Hostess ⇔ Customer	Host seats customer to unoccupied table	Conveys Requests
Host/Hostess ⇔ Reservation	Host takes reservations from customers to assign tables on date	Conveys Requests
Host/Hostess ⇔ Floor	Host updates the floor occupancy	Provides Data
Host/Hostess ⇔ Table	Host updates the table status when reserving/assigning table	Provides data
Customer ⇔ Menu	Customer gets information from the menu	Provides data
Customer ⇔ Waiter/Waitress	Customer gives waiter/waitress order information	Conveys request
Customer ⇔ Website	Customer receives restaurant information	Provides Data
Customer ⇔ Reservation Modifier	Customer can book and edit their own reservation	Conveys request
Waiter ⇔ Kitchen	Waiter relays orders to Kitchen	Conveys request
Waiter ⇔ OrderQueue	Waiter checks for updates from order queue	Requests updates
Kitchen ⇔ OrderQueue	Kitchen updates the order status for the order queue	Modifies
Waiter ⇔ CheckModifier	Waiter updates the check of orders into the system	Modifies/Provides Data
Employee ⇔ Clock	Employee informs the schedule of clock in/clock out times	Provides Data

iii) Attribute Definitions:

Category:	**Attribute:**	**Description:**
Employee	userID	String representing each individual employee in the software suite
Employee	password	String stored and associated with each userID
Employee	shiftStart	Time when employee starts shift
Employee	shiftEnd	Time when employee ends shift
Manager	managerName	Name of manager
Manager	managerID	Unique number representing a manager
Manager	managerPermissions	Manager permissions include: changing orders, posting staff schedules, seeing daily transaction reports, log-in time of employees, all waiter permissions
Host/Hostess	hostName	Name of host/hostess
Host/Hostess	hostID	Unique number representing a host/hostess
Host/Hostess	hostPermissions	Host permissions include: creating reservations, updating floor occupancy, modifying table status.
Waiter	waiterName	Name of waiter
Waiter	waiterID	Unique number representing a waiter
Waiter	waiterPermissions	Waiter permissions include: seeing which tables associated with, see status of

		tables
Waiter	tableAssociation	Bitmap showing all tables assigned to waiter
Floor	floorLevel	Assigns each floor a level to display the array of tables
Floor	floorStatus	Shows the occupancy percentage of each floor inside the restauarant
Table	freeTables	Array of all available tables open for customers in queue for a table
Table	occupiedTables	Array of all tables in use by a customer party, cannot be given to queued customer
Table	tableStatus	Shows if a current table is: in use, waiting for order, eating, waiting for check, needs cleaning, is ready
MenuItem	itemName	Menu item’s name
MenuItem	itemDescription	Menu item’s description
MenuItem	itemPrice	Menu item’s price
Order	order	Order of a party at a table, list of items for chef to prepare
Order	orderID	Unique number representing table’s order
Order	isOrderReady	Boolean showing status of order
OrderQueue	orderQueue	List of all orders needed to be prepared by chefs
Check	checkID	Unique number representing table’s check
Check	checkSubtotal	Overall subtotal table has to

		pay on the bill
Check	checkTotal	Overall total after tax table has to pay on bill
Check	checkTip	Calculated possible tips after total for: 15%, 18%, 20% tips
TransactionReport	Day	Signifies transaction report is daily
TransactionReport	totalGain	Sum of all methods of revenue for the day
TransactionReport	totalLoss	Sum of all methods of losses for the day
TransactionReport	netGain	Difference of the gains and losses
Schedule	employeeName	Name of employee
Schedule	date	All of the dates specified employee is working
Schedule	time	Time frames specified for each date employee is working

Traceability Matrix:

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23
UC1	*			*	*	*	*	*	*						*								
UC2	*			*	*	*	*			*						*	*						
UC3	*		*	*		*	*			*		*											
UC4		*				*				*		*	*	*									
UC5	*	*	*	*	*																*	*	*
UC6		*									*			*				*	*	*		*	*

System Operation Contracts

Name	Customer Places Order
Responsibilities	Customers place their meal orders through the waiters
Use Cases	UC-1
Exception	None
Preconditions	Customers are already seated and Waiters are logged into the mobile app.
Postconditions	The order is placed and sent to the Kitchen, status will be updated on mobile app.

Name	Manage Meal Orders
Responsibilities	Kitchen staff will prepare placed orders and notify waiters when to serve
Use Cases	UC-2
Exception	None
Preconditions	Meal orders have already been placed through the mobile app
Postconditions	On the Mobile app, waiters can see the change of progress of orders, thus knowing when to pick up and serve ready meals

Name	Logging Work Hours
Responsibilities	Restaurant Staff are able to clock in and out their work hours using the desktop app
Use Cases	UC-5
Exception	None
Preconditions	Users are employees and logged in to desktop application
Postconditions	After clocking out, the total time worked is saved in the database

Name	Seating Customers
Responsibilities	As customers come in, host are able to find appropriate seating for them
Use Cases	UC-3
Exception	None
Preconditions	Host is logged in to desktop app and viewing the floor page
Postconditions	Host leads party to table and hands off responsibility to the waiter of the floor to serve them

Name	Reservation Management
Responsibilities	Hosts can create or verify customers reservations
Use Cases	UC-4
Exception	None
Preconditions	Hosts are logged into the desktop app and viewing the Reservations page
Postconditions	Through the reservation page, hosts can verify name of customer to see if they have a reservation or create a new one for a future visit

Name	Manage Menu Items
Responsibilities	User can update the Menu either adding, deleting or updating menu items
Use Cases	UC-6
Exception	None
Preconditions	User is an admin/chef logged into the desktop app and viewing the admin page
Postconditions	Menu items are updated with user’s modification

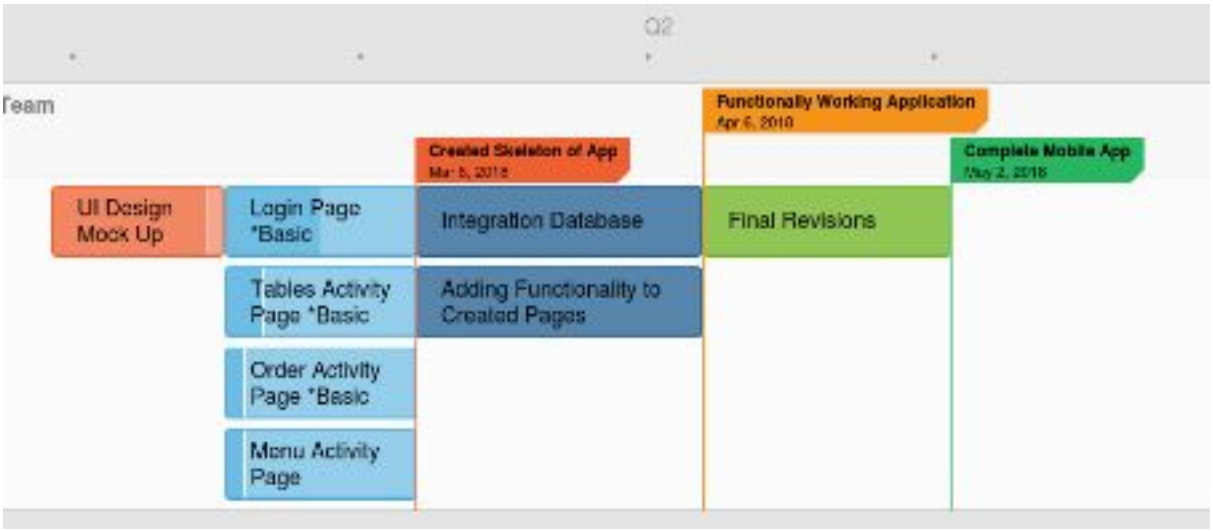
Name	View Worker Hours
Responsibilities	Check all worker hours
Use Cases	UC-6
Exception	None
Preconditions	Admins are logged into desktop application and viewing Admin page
Postconditions	View logs of hours worked by each employee

Name	View Receipts
Responsibilities	View a log of all receipts of customers
Use Cases	UC-6
Exception	None
Preconditions	Admins are logged into desktop application and viewing Admin page
Postconditions	View logs of all receipts made

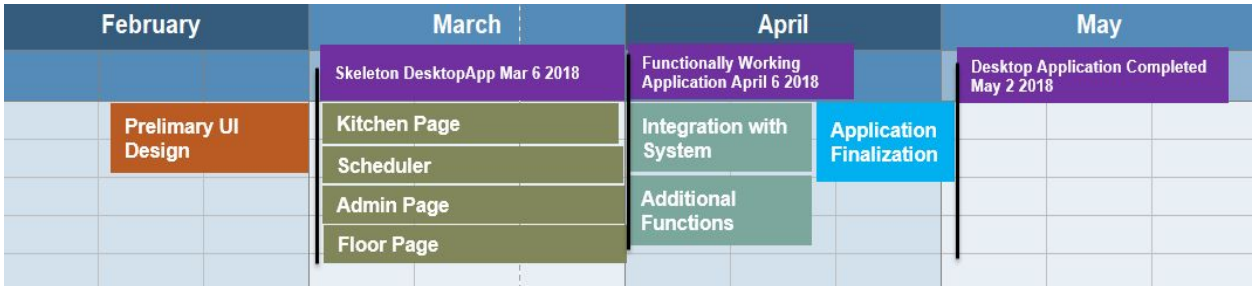
Plan of Work

At the end of Report #1, our group has set the foundations of what the Restaurant Automation Software should be and who it's for. Moving on we will continue with the development of the software which has been started since the start of the groups creation and the documentation the comes along with it. Below is a roadmap of the development of our software.

Mobile Application Roadmap

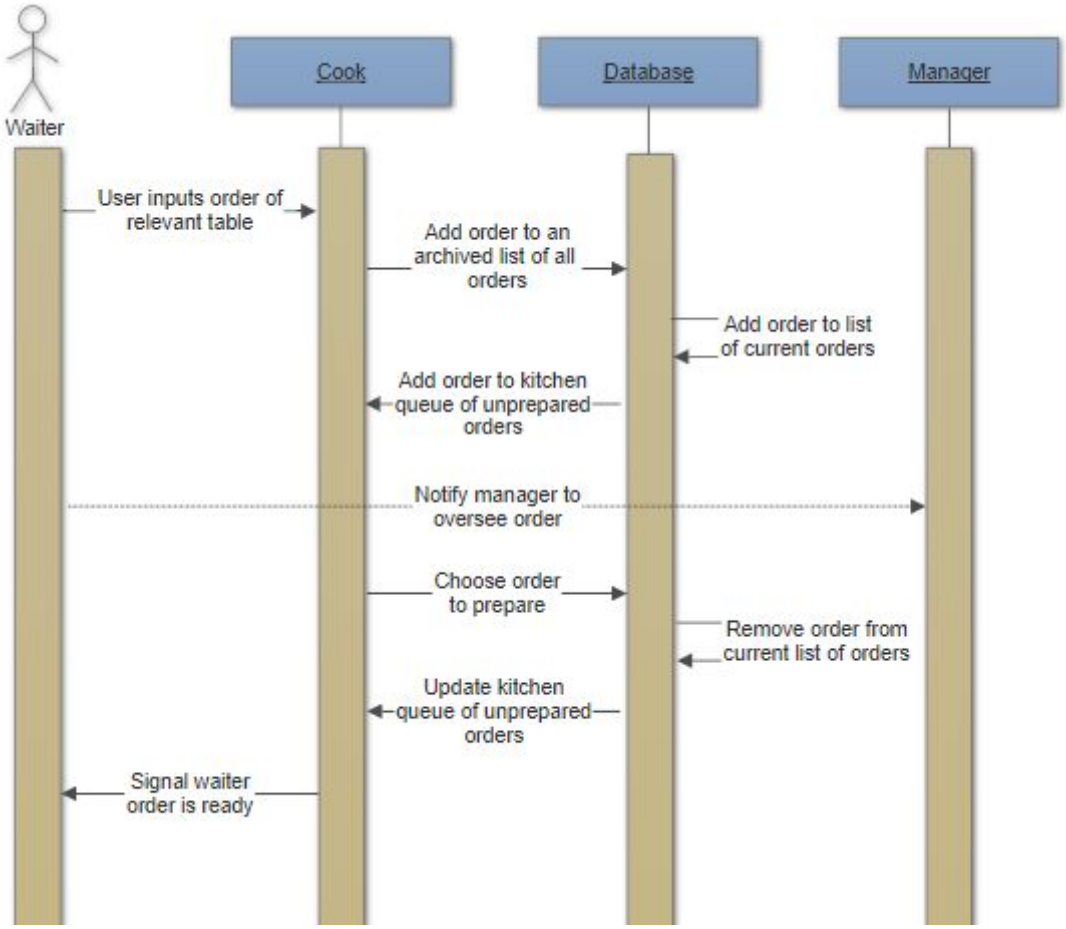


Desktop Application Roadmap



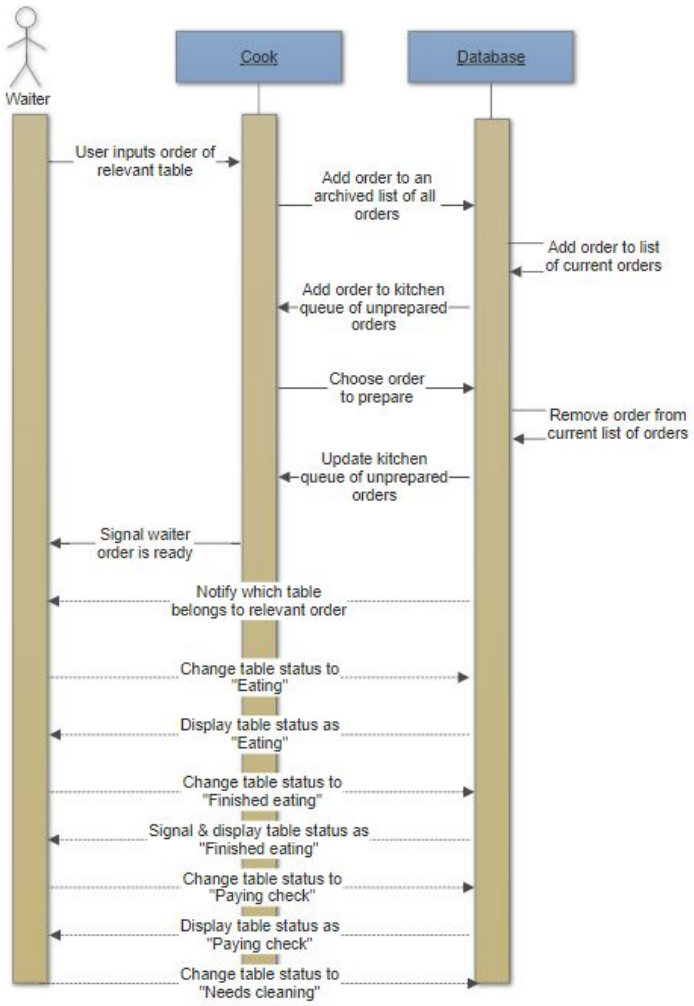
Interaction diagrams

Use Case 1: Order



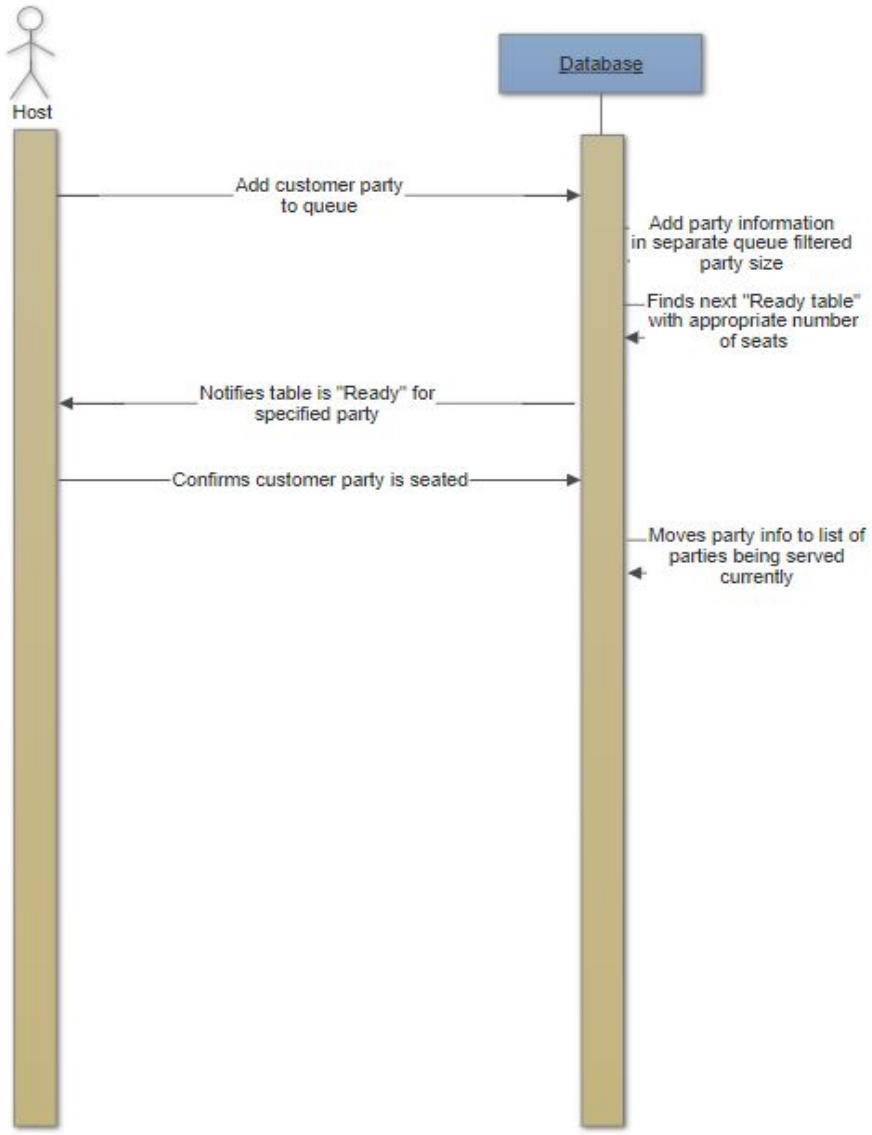
In this Use Case, the waiter begins by placing an order using the mobile device which sends to the Cook via the Database. The Database will place the order into a list of all current orders and send the list to the Cook. Once the Cook selects the order they are preparing, it's removed from the order list and once completed, the Waiter will be notified on their mobile device.

Use Case 2: Serve



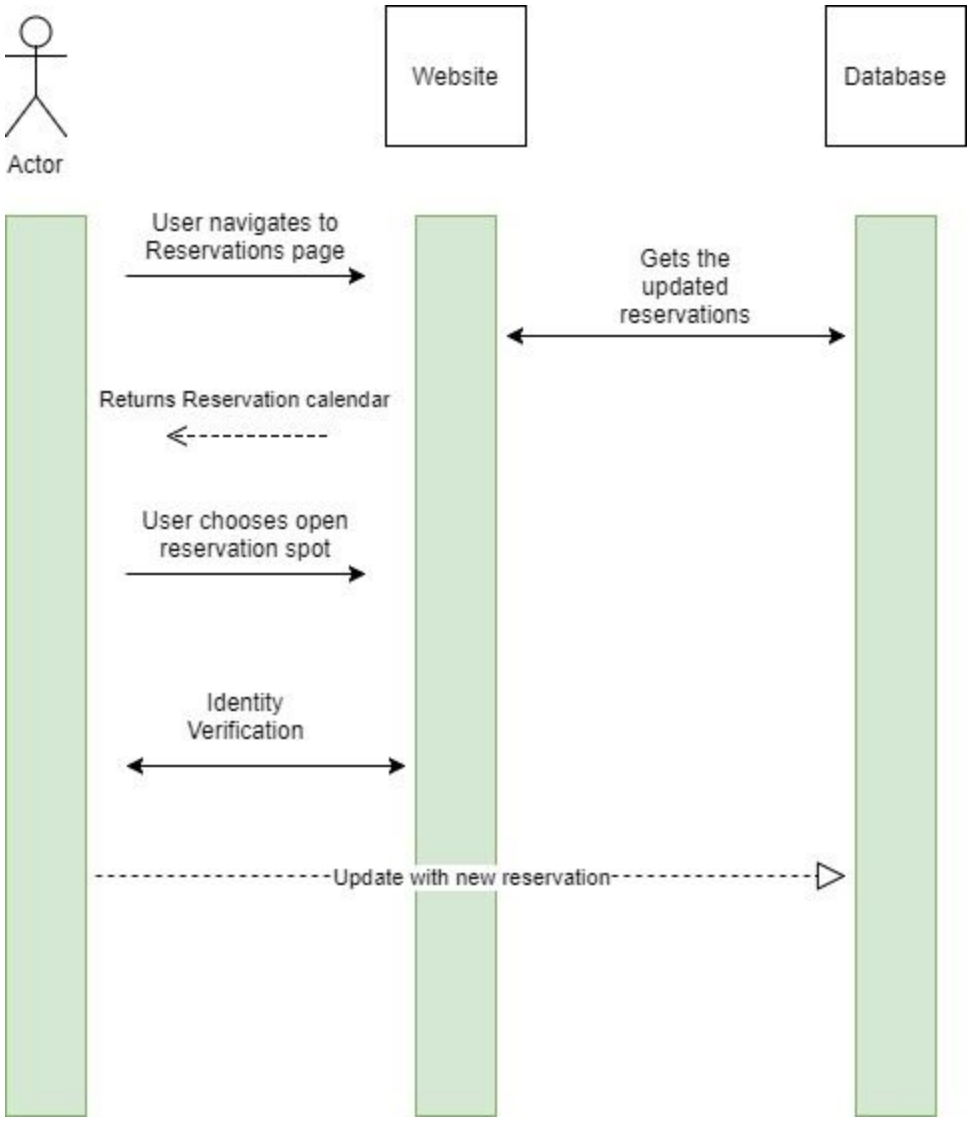
In this use case, the waiter is notified when the cook has finished an order and is ready to send the order to the table. The system has a database that keeps track of all of the orders in a queue and subsequently notifies the waiter when an order is ready, thus dequeuing it. When the order is served to the respective table, the table's status is then changed to "eating." Afterwards, once the table is finished with their meals, the table's status is changed to "finished eating" and the waiter is notified to collect the check. Finally, once the check is paid to the system, the table's status is changed to "needs cleaning" to set up for the next customer(s). In the main success case for an authorised user, the system will accurately assess this data over time and keep a precise check on each table's status.

Use Case 3: Seat



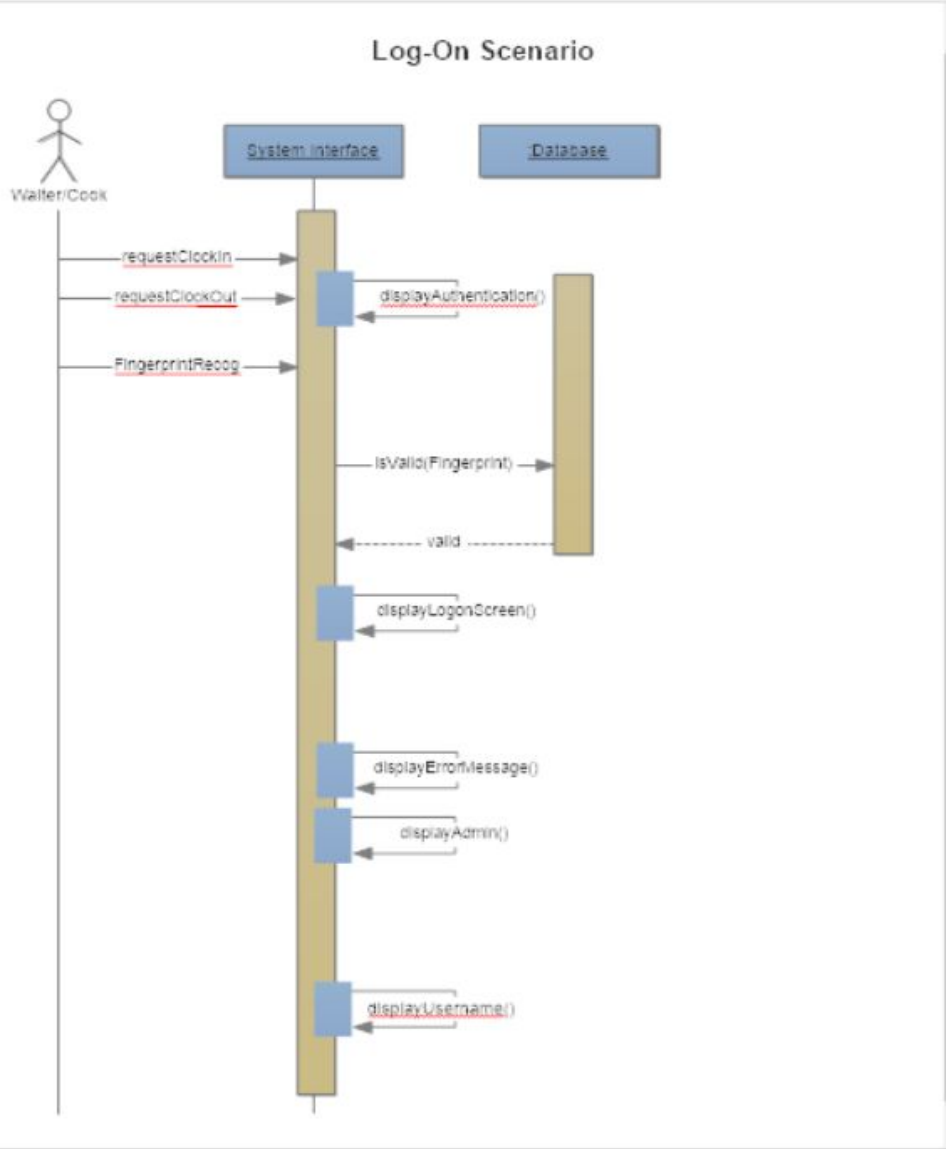
For the seating use case, the customer enters the restaurant and selects the ‘seating’ option. The host adds the customer to a seating queue on the database. The party size is calculated with the current seating options to find an optimal available table. When the table is open, the database notifies the customer that they are ready to be seated. The host directs the party to the table and confirms to the database that the customer is seated. The database then moves the customers to a list of parties currently being served.

Use Case 4: Reservation



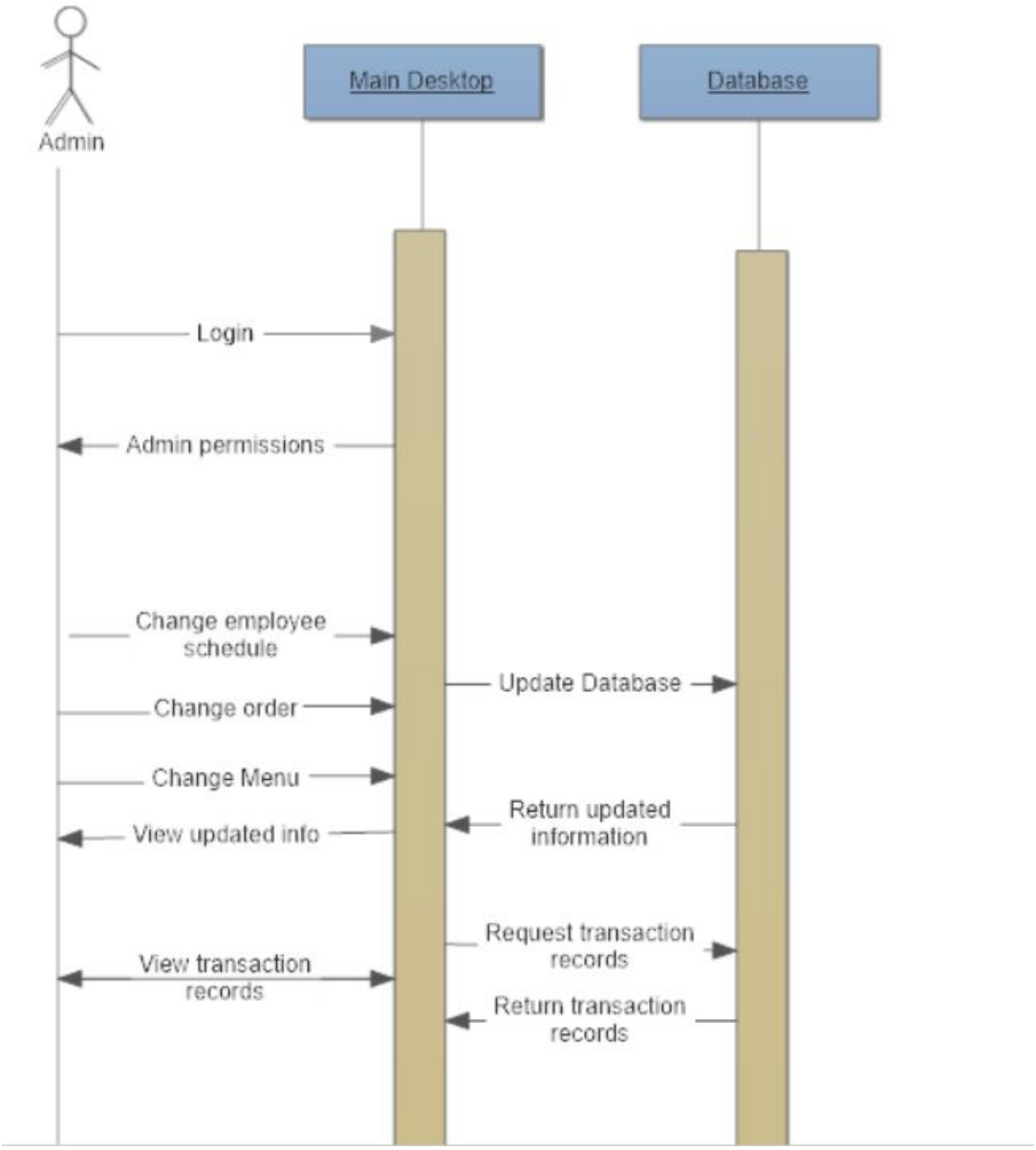
In this use case, there are two type of users that may interact with reservation UI, customer and staff member (hostess or manager). Reservation navigation may be access through restaurant website for customers and also via desktop by restaurant staff. All information given by the customer will be stored in the database server. Once reservation is made, user will get a confirmation. In case a user might need help on a existent reservation, a staff member can access this information via desktop queries to the database server. In short, reservation UI gives users access to secured data in the database system for management purposes.

Use Case 5: Clock In/Clock Out



In this use case the employee will attempt to clock in for the purpose of providing proof of their arrival to work and for recording the duration of which they were at work. The system has a biometric verification system in the for of a fingerprint scanner ensuring only authorised employees will access the system. If an unauthorised user attempts to clock in they will be presented with an error message. In the main success case for an authorised user, the system will allow the user to log on after obtaining the valid fingerprint and storing their clock in, clock out and hours worked for use of an administrator when he/she need to access that data.

Use Case 6: Admin

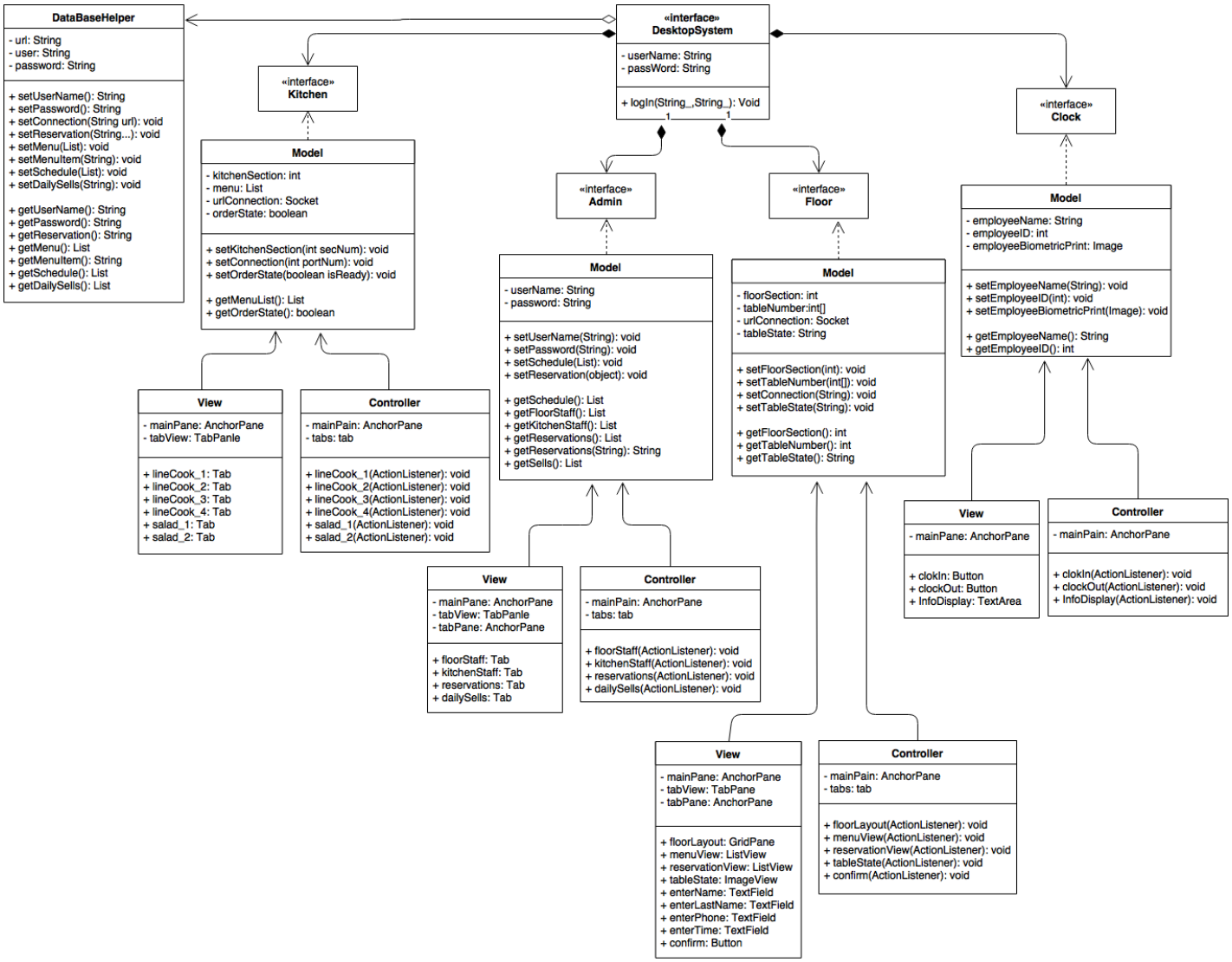


In this use case, there is one type of user that initially interacts with the serving use case: the administrator. It main series of events will include that the admin logs in to the main desktop, granting appropriate admin permissions to the user. The admin can choose to change the employee schedule, a specific order, the menu, and look at daily transaction records. All of these functions revolve around the main desktop being the controller that will manipulate the information as needed, while the database houses all the information.

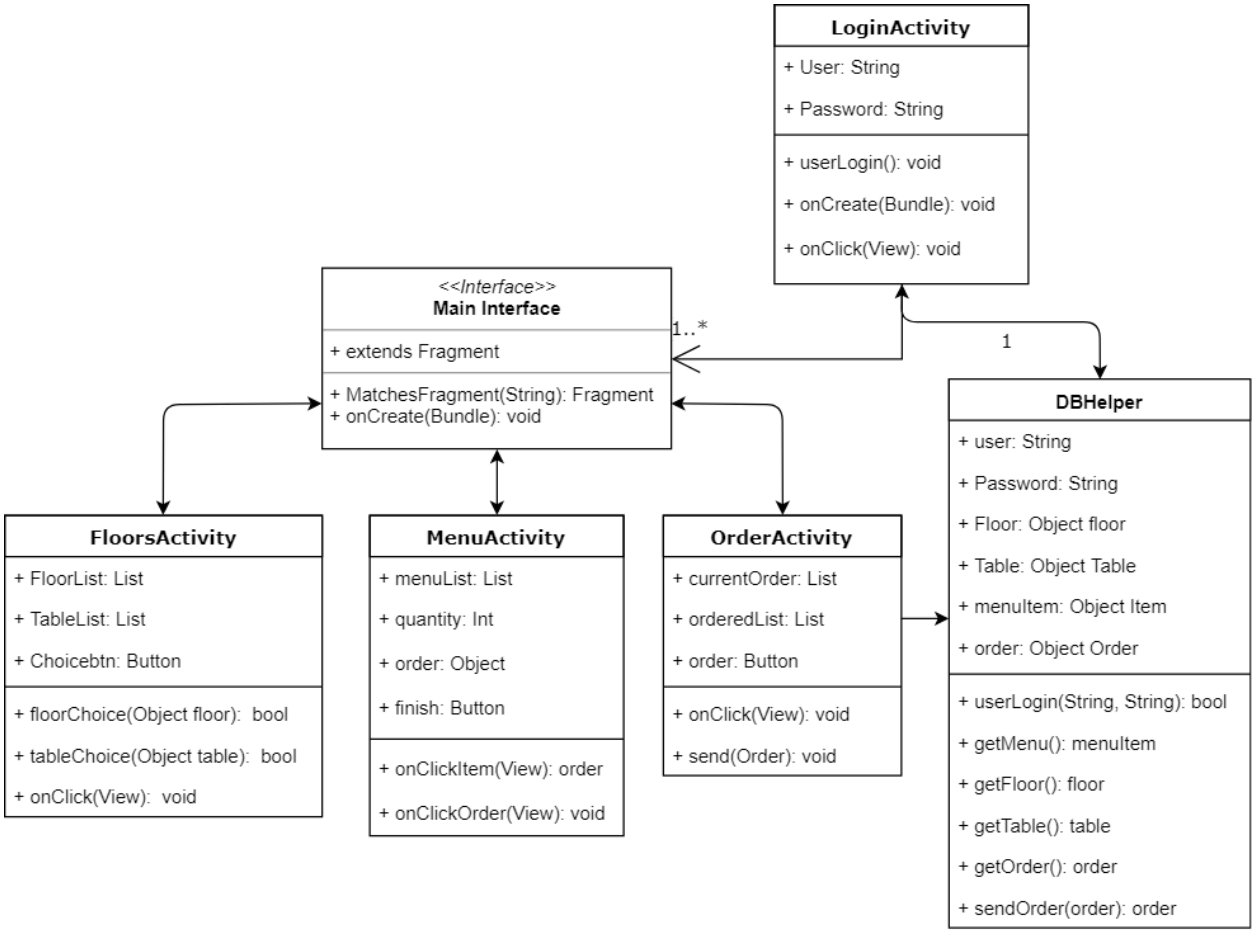
Class Diagram and Interface Specification

Class Diagrams

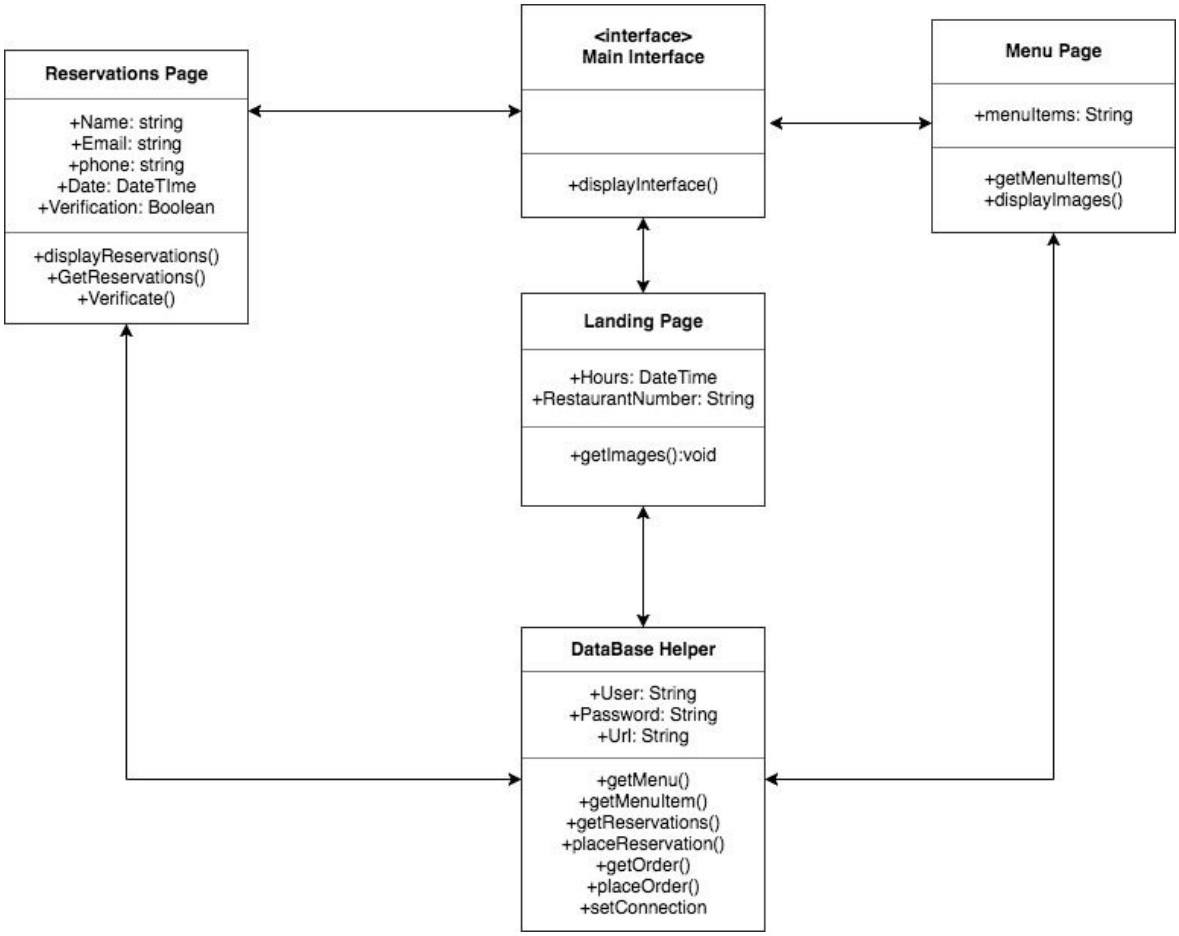
Desktop Application Class Diagram



Mobile Application Class Diagram



Website Class Diagram



Data Types and Operation Signatures

Desktop data types and operation signatures

Kitchen interface:

Kitchen Model Class	
Attributes	**Description**
- kitchenSection: int	To store respective kitchen section as part of a data structure of kitchen sections.
- Menu: List	To store/display menu list for cooks references.
- urlConection: socket	To establish bidirectional communication between kitchen and waiter applications.
- orderState: boolean	To show whether order is ready or not.

Kitchen Model Class	
Methods	**Description**
+ setKitchenSection(int):void	To store/update kitchen section.
+ setUrlConnection(int): void	To create the connection via designated port number.
+ setOrderState(boolean): void	To update order status.
+ getMenuList(): List	To display all menu items in the screen as requested.
+ getOrderState(): boolean	To check whether order is ready or not.

Admin Interface:

Admin Model Class	
Attributes	**Description**
- userName: String	Instance variable to store manager name.
- password: String	Instance variable to store a password.

Admin Model Class	
Methods	**Description**
+ setUserName(String): void	Setter function to store/update a new/existing name in data structure.
+ setPassword(String): void	Setter function to store/update a new/existing password in data structure.
+ setSchedule(List): void	Setter function to store/update a new/existing work schedule in data structure.
+ setReservation(object): void	Setter function to store/update a new/existing reservation in data structure.
+ getSchedule(): List	Getter function to display an existing work schedule.
+ getFloorStaff(): List	Getter function to display all employees working in restaurant floor.
+ getKitchenStaff(): List	Getter function to display all employees working in restaurant kitchen.
+ getReservations(): List	Getter function to display all existing reservations in the restaurant.
+ getReservations(String): String	Getter function to display an specific reservation.
+ getSells(): List	Getter function to display all existing sells.

Floor Interface:

Floor Model Class	
Attributes	**Description**
- floorSection: int	Instance variable to store floor section number.
- urlConnection: Socket	Instance variable open a new connection between host and waiter.
- tableState: String	Instance variable to hold current table state(ready, dirty, occupied).
- tableNumber: int	Instance variable to store a table number.

Floor Model Class	
Methods	**Description**
+ setFloorSection(int): void	Setter function to store/update a new/existing floor section in data structure.
+ setTableNumber(int): void	Setter function to store/update a new/existing table number in data structure.
+ setConnection(String): void	Setter function to create a connection via designated port number.
+ setTableState(String): void	Setter function to update the current status of a table.
+ getFloorSection(): int	Getter function to display all existing floor sections and floor layout.
+ getTableNumber(): int	Getter function to display table number.
+ getTableState(): String	Getter function to display current table status.

Clock in/out Interface:

Clock Model Class	
Attributes	**Description**
- employeeName: String	Instance variable to store employee name.
- employeeID: int	Instance variable to store employee identification number.
- employeeBiometricPrint: Image	Instance variable to store employee fingerprint.

Clock Model Class	
Methods	**Description**
+ setEmployeeName(String): void	Setter function to store/update a new/existing employee name in data structure.
+ setEmployeeID(int): void	Setter function to store/update a new/existing employee ID number in data structure.
+ setEmployeeBiometricPrint(Image): void	Setter function to store/update a new/existing employee fingerprint in data structure.
+ getEmployeeName(): String	Getter function to display employee name.
+ getEmployeeID(): int	Getter function to display employee ID number.

DataBase Interface:

DataBase Class	
Attributes	**Description**
- url: String	Instance variable to store DBMS url.
- user: String	Instance variable to store DBMS user name.
- password: String	Instance variable to store DBMS password.

DataBase Class	
Methods	**Description**
+ setConnection(String url): void	Setter function to create connection between DBMS and desktop application.
+ setReservation(String...): void	Setter function to store/update a new/existing reservation in database(reservation table).
+ setMenu(List): void	Setter function to store/update a new/existing menu items in database(menu table).
+ setMenuItem(String): void	Setter function to store/update a specific menu item in database(menu table).
+ setSchedule(List): void	Setter function to store/update a new/existing work schedule in database(schedule table).
+ setDailySells(String): void	Setter function to store/update a new/existing sells in database(sells table).
+ getUserName(): String	Getter function to get specific employee name from DBMS for identification(login table).
+ getPassword(): String	Getter function to get specific employee password from DBMS for identification(login table).
+ getReservation(): String	Getter function to get specific reservation from DBMS (reservation table).
+ getReservations(): List	Getter function to get all existing reservations from DBMS (reservation table).
+ getMenu(): List	Getter function to get all existing menu items from DBMS (menu table).
+ getMenuItem(): String	Getter function to get a specific item in the menu from DBMS (menu table).
+ getSchedule(): List	Getter function to get all existing work schedules from DBMS (schedule table).
+ getDailySells(): List	Getter function to get all past sells from DBMS (sells table).

Mobile data types and operation signatures

LoginActivity:

LoginActivity Class	
Attributes	**Description**
- user: String	Instance variable for user
- password: String	Instance variable for user's password

LoginActivity Class	
Methods	**Description**
+ userLogin(): void	Function to check with DB to check if user is a valid user of system
+ onCreate(Bundle): void	Standard Android function that will create elements to be shown on screen
+ onClick(View): void	Event Handler function for when a button is pressed to call userLogin()

FloorsActivity:

FloorsActivity Class	
Attributes	**Description**
- FloorList: List	Android on screen list element for selecting floor
- TableList: List	Android on screen list element for selecting table
- Choicebtn: Button	Android on screen button element for triggering events

FloorsActivity Class	
Methods	**Description**
+ floorChoice(List): bool	Event handler function to set Floor when user chooses from list
+ tableChoice(List): bool	Event handler function to set Table when user chooses from list
+ onClick(View): void	Event handler function for when floor and table are selected to begin order

MenuActivity:

MenuActivity Class	
Attributes	**Description**
- menuList: List	Android on screen list element for selecting menu items
- quantity: int	Instance variable to increase quantity of selected menu items
- order: Object	Object that holds menu item and quantity
- finish: Button	Android on screen button element for triggering events

MenuActivity Class	
Methods	**Description**
+ onClickItem(View): order	Event handler function for when an item is clicked to create and order object and store it in there with chosen quantity
+ onClickerOrder(Order): void	Event handler function for when finish is pressed to send order object to OrderActivity class

OrderActivity:

OrderActivity Class	
Attributes	**Description**
- currentOrder: List	Android on screen list element for passed order object to verify order before being sent
- orderedList: List	Android on screen list element for viewing sent orders to Kitchen
- order: Button	Android on screen button element for sending order to DB

OrderActivity Class	
Methods	**Description**
+ onClick(View): void	Event handler function to view past sent orders
+ send(Order): void	Event handler function for order button to send order to DB for Kitchen

DBHelper:

DBHelper Class	
Attributes	**Description**
- user: String	Instance variable for user
- password: String	Instance variable for user's password
- Floor: Object	Instance object for restaurant floors
- Table: Object	Instance object for restaurant table
- menuItem: Object	Instance object for restaurant menu items
- order: Object	Instance object that holds menu items and quantities

DBHelper Class	
Methods	**Description**
+ userLogin(String...): bool	Check function for if given user and password vars and valid in the DB
+ getMenu(): menuItem	Getter function that returns menuItems from the DB
+ getFloor(): floor	Getter function that returns floors from the DB
+ getTable(): table	Getter function that returns tables from the DB
+ getOrder(): order	Getter function that returns orders from the DB
+ sendOrder(order): order	Function that sends order to DB to be sent to the Kitchen

Website data types and operation signatures

DBHelper Class	
Attributes	**Description**
- user: String	Instance variable for user
- password: String	Instance variable for user's password
- Url: String	Instance variable to store DBMS url.

DBHelper Class	
Methods	**Description**
+ setConnection()	Setter function to create connection between DBMS and desktop application.
+ placeReservations()	Function that places a new Reservation into the DB
+ getReservations()	Getter function that returns Reservations from the DB
+ getMenu()	Getter function that returns Menu from the DB
+ getMenuItem()	Getter function that returns specific Menu Item from the DB
+ sendOrder()	Function that sends order to DB to be sent to the Kitchen
+ getOrder()	Getter function that returns Order from the DB

Landing Page Class	
Attributes	**Description**
+ Hours: DateTime	The hours of operation that the restaurant is open
- RestaurantNumber: String	The restaurant's phone number

DBHelper Class	
Methods	**Description**
+ getImages()	Getter function that returns the images of the Items from the DB

Menu Page Class	
Attributes	**Description**
+ MenuItems:String	All the menu items available at the restauarnt

DBHelper Class	
Methods	**Description**
+ getMenuItems()	Getter function that returns the images of the Items from the DB
+ displayImages()	Displays the images of the items after getting them

Reservations Page Class	
Attributes	**Description**
+ Name: String	The name of the customer that wants to place a reservation
+ Email: String	The email of the customer that wants to place a reservation
+ Phone: String	The phone number the customer that wants to place a reservation
+ Date: DateTime	The time that the customer wants to place a reservation
+ Verification: Boolean	Whether the verification returns valid or not

Reservations Class	
Methods	**Description**
+ getReservations()	Getter function that returns the Reservations from the DB
+ displayReservations()	Displays the returned Reservations
+ Verificate	Verify the customer for the reservation

Traceability Matrix:

	Software Classes								
Domain Concepts	DatabseHelper	Kitchen	Admin	Floor	Clock	FloorsActivity	LoginActivity	MenuActivity	OrderActivity
Employee		*	*		*	*	*	*	*
Manager			*						
Host/Hostess			*	*		*			
Waiter/Waitress			*						*
Kitchen		*	*						*
Floor	*		*	*		*			
Table	*			*		*			
Menu	*	*						*	
tableStatus	*			*					
Reservation	*		*	*					
OrderItem		*							*
OrderQueue		*							*
Check									*
Clock					*				
ClockSchedule	*		*		*				
ClockIn					*				

Traceability Matrix: Concepts/Class relationship

The DatabaseHelper is able to modify/retrieve the menu, reservations, schedules, table/floor status.

Employee revolves around all the staff and as a whole has access to the entire system's activites.

The kitchen class has control over all the food operations including orders and menu.

Admin is accessed via the manager and can view all the staff information. It can also modify reservations and clocking.

Floor intakes the table and reservation activity overseen by the host/hostess.

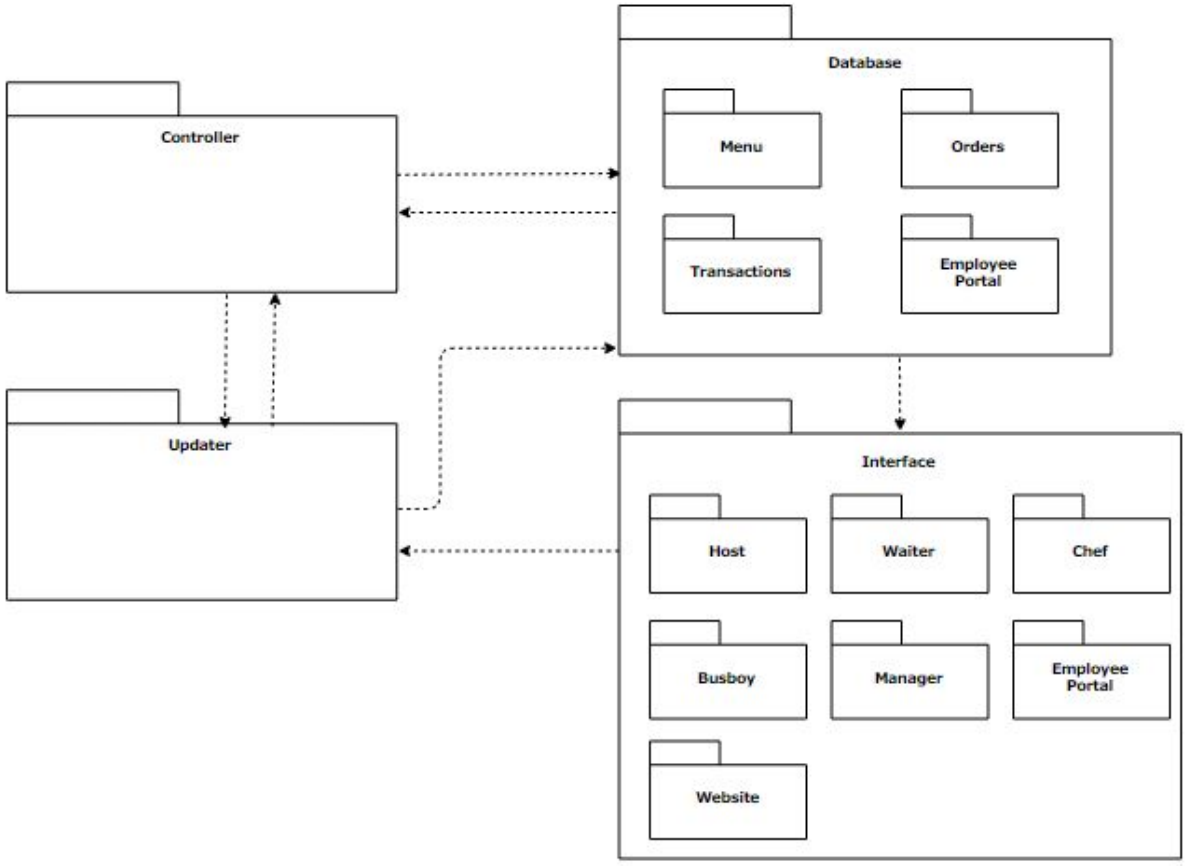
Clock class encompasses all of the scheduling/clocking in and out.

System Architecture and System Design

Architectural Styles:

The architecture style that we are using for our system is client server based system. The server stores data and executes functions which are initiated by the clients which would consist of the desktop and tablets. The server holds the database consisting of all the relevant information for the restaurant's operation such as order information, transaction information, employee information and table status. The functions to update,change or view this information is initiated by the client. The functions that the clients can access changes depending on the user permissions of who is associated with the client. The manager who is the highest ranking therefore having the most user permissions, has the most access to the server functions through the client. As an example, considering the manager as client will have permissions to do things that no other employee can such as changing the menu, viewing/changing the work schedule of an employee worked and checking transaction records. The manager therefore can make a request to the server to execute these function which will update the database and return the relevant information to the client. A client with less user permission such as a waiter will only be able to make limited server request related to his job such as requesting a change in table status and adding new orders. Since we will be using a cloud based server all our operations will be dependent on a good network connection to allow quick communication between server and its clients for efficient restaurant operation. Delays in communication will result in delayed relay of information across the system from waiter to kitchen to manager which will be detrimental to the functioning of the restaurant. A cloud based system however has the advantage of not requiring to run,maintain and secure your own local server which will increase hardware investment costs.

Identifying Subsystems:



The interface subsystem will house everything all employee types (host, waiter, chef, busboy, manager) need to see, alongside the employee portal which will allow the the interface to talk to the updater. The interface subsystem will also have the website interface. The updater can then update any information in the database an employee would like to change through their devices. The database will display all necessary information to the interfaces of each employee, while housing the menu, orders, and transactions. All at the center of the system will be the controller subsystem, which will act as an intermediary between the updater and database.

Mapping Subsystems to Hardware:

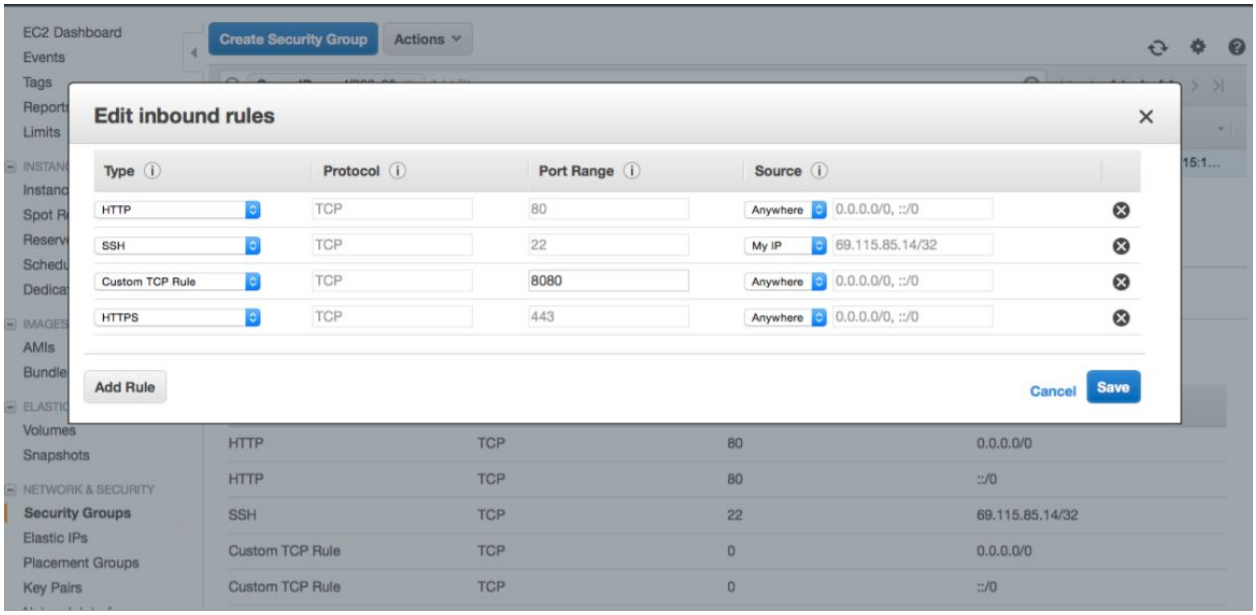
Because Restaurant Automation is a client-server architecture, multiple computers will be needed for the system. The server which has the database will be on a master computer, holding all the menu, order, transaction, and employee portal information from a remote location. A desktop app which will require another computer located at the restaurant will have the employee portal, manager interface, and host interface subsystems. The desktop app will also have all of the controller and updater subsystem functions. Tablets, which by definition are computers, will have the employee portal, water, chef, and busboy subsystems. The website subsystem can be accessed from any computer or mobile device to reserve tables or look at the menu.

Persistent Data Storage:

Our system requires data to be stored for an extended period of time for the purpose of record keeping and analysis by the manager to make adjustments to improve business as well as for the daily functioning of the restaurant. The orders, work schedules, clock in and out time, and menu are all stored in the database. The storage of the menu is necessary as the orders will be entered in the tablet and sent to the order section of the database based on what was present in the menu section of the database. One of the main benefits of using a cloud based database is we can have as much storage as we desire without having to worry about the space, cost and technical expertise required in installing,running and maintaining a local server to do all the tasks.

Network Protocol:

The system will be based on the Amazon Web Services Cloud for the desktop and web application. There will be a TCP/IP protocol that's configured with AWS for accessing the network. When SSHing onto the server (port 22), only the creator IP can access the online system to maximize security. There is a custom TCP Port 8080 that serves as the main source for uploading any JAR files through Tomcat's Apache as the applications will be Java, Javascript, and MySQL based.



Global Control Flow:

Execution Orderliness:

Restaurant Automation will execute in an event driven fashion initially, then in a linear fashion for the remainder of the procedures. The initial events would be if the customer is reserving a table for their party, they would input their party information and time through the website. Then whether or not the customer reserved, all customers would walk into the restaurant and tell the host their party's name and number of people in the party. If the customer did not reserve their table, the host would input the party's information, adding them to the queue.
The remainder of the global flow would be linear. Once the optimal table is ready for the customer's party, the party will be seated. After that, the server will input the customers' orders into the server's tablet, notifying the chefs of a new order. Once a chef takes the order and prepares it, he/she can notify the corresponding server when the order is ready. After the meal is sent to the customers, when the customers are ready to pay the bill, the server can update the tablet to let the system know the table is paying. Once they're done paying, the server can update the system that the table is dirty and ready for cleaning, letting the busboys know to clean the table. Finally, when the busboy is done cleaning the table, he/she can update the table in their tablet as a "ready to be seated" table.

Time Dependency:

Restaurant Automation uses clocks to keep track of how long an employee has worked during their shift between logging into the system and logging off. Other than that, the system is an event-response type since the control flow is dependent on if the user working on the current part of the flow is done and notifies the next part of the flow.

Concurrency:

Restaurant Automation will use multiple threads since there will be multiple systems going independently at the same time. One way multiple threads may spawn is if there are multiple customers reserving a table, waiting for a ready table, or putting in orders. This is controlled by putting the customers table or order requests in a chronological queue for the fairness of customers who acted first. Another instance of multiple threads is if the admin has to edit any information of a customer's order, menu, or transactions through the database. This won't create conflict with the other threads interacting with the customers as the thread is only interacting with database. There is no synchronization between threads as each thread handles different, independent actions.

Hardware Requirements:

The hardware components needed for the our system to operate are:

- A central desktop PC which will have the most functionality in using/changing the system via the desktop app with a monitor display
- Android tablets for the employees to carry out their duties via the mobile app.
- Customer web browsing device to access the restaurant website
- One router per floor for sufficient WiFi connection. Each terminal should have a minimum bandwidth of 1.5mbps download speed and 768 kbps upload speed.

Desktop-

Hardware Component	**Minimum Requirements**
Processor	Dual Core CPU
RAM	4GB
HDD	128GB
Network Connection	Ethernet/Wifi card

Tablet-

Hardware Component	**Minimum Requirements**
Processor	Quad Core CPU
RAM	2GB
HDD	16GB
Screen	Touchscreen display
Network Connection	Wifi card

Customer Web Browsing Device

Any device that can access the internet using a web browser.

References

1. **Group 3's Report 2015 in formatting of document:**
    a. http://www.ece.rutgers.edu/~marsic/books/SE/projects/Restaurant/2015-g3-report3.pdf
2. **Group 4's Report 2014 in formatting of document:**
    a. http://www.ece.rutgers.edu/~marsic/books/SE/projects/Restaurant/2014-g4-report3.pdf
3. **UML Diagrams designed in:**
    a. http://www.draw.io
    b. https://www.lucidchart.com
4. **User Story Points:**
    a. https://www.atlassian.com/agile/project-management/estimation
    b. The website was used to determine how many story points should be awarded for each problem
5. **User Effort Estimation:**
    a. https://www.tutorialspoint.com/estimation_techniques/estimation_techniques_use_case_points.htm
    b. Website was used to determine how to properly calculate user estimations for technical and environmental factors.
6. **Gantt Charts:**
    a. http://www.gantt.com/
    b. The website was used to figure how to create and a Gantt chart and how to use the tool effectively
7. **UI Mobile Application Mockup**
    a. https://www.fluidui.com

8. Hardware Requirements

https://support.revelsystems.com/hc/en-us/articles/204251049-What-are-the-bandwidth-requirements-

Determining the bandwidth and router usage for an internet dependent restaurant.