Software Engineering

14:332:452

Group #11

Project URL: https://sites.google.com/site/softwareengineeringspring2012/

Restaurant Automation

Report 1

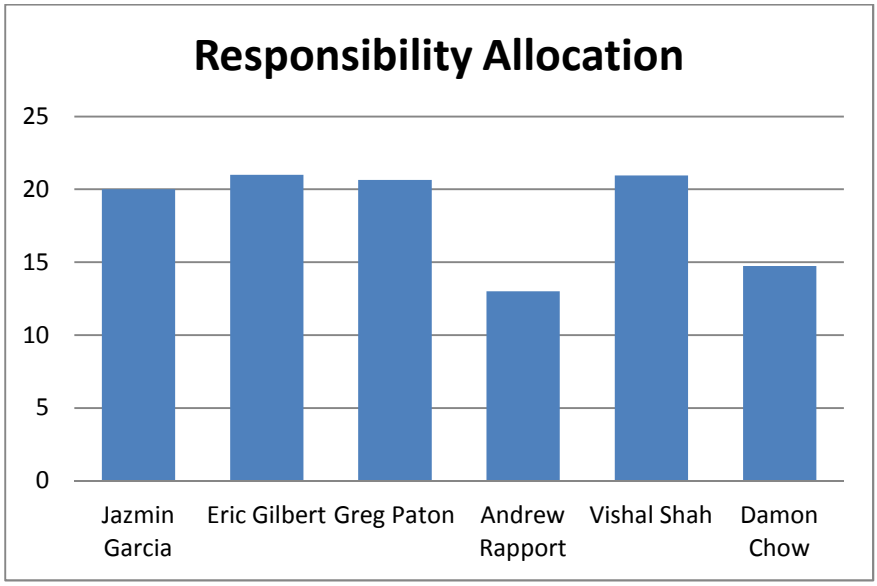
February 17, 2012

Jazmin Garcia
Greg Paton
Eric Gilbert
Andrew Rapport
Vishal Shah
Damon Chow

Individual Contributions Breakdown

Team Member Names

	Jazmin Garcia	Eric Gilbert	Greg Paton	Andrew Rapport	Vishal Shah	Damon Chow
Project Management	0.6			0.4		
Sec.1: Customer Statement of Requirements			1			
Sec.2: System Requirements	0.1	0.2	0.1	0.2	0.1	0.1
Sec.3: Functional Requirements Specification	0.28	0.26	0.16	0.2	0.22	0.18
Sec.4: User Interface Specs		0.8				0.25
Sec.5: Domain Analysis			0.25		0.55	0.2
Sec.6: Plan of Work	1					



1. Customer Statement of Requirements

a. Problem Statement

Running a restaurant incurs a great deal of overhead. Anyone looking to open and run a successful restaurant in today's economy needs to minimize cost to stay competitive. Therefore, a modern system must be implemented that can automate tasks that were once heavily time consuming, increasing efficiency and reducing operating costs. Implementing a computer system from the start of a new restaurant will have the proprietor showing profit sooner and increasing the chances of staying in business. Converting to a computer system in a currently running restaurant, while requiring a short period of adjustment to train employees, will eventually lead to greater efficiency and higher profit margins.

Basic Features

- Increase efficiency
- Reduce overhead
- Minimize customer dining time

Touch Screen Monitors

- Easy and quick method for inputting orders
- Decreases order placement time

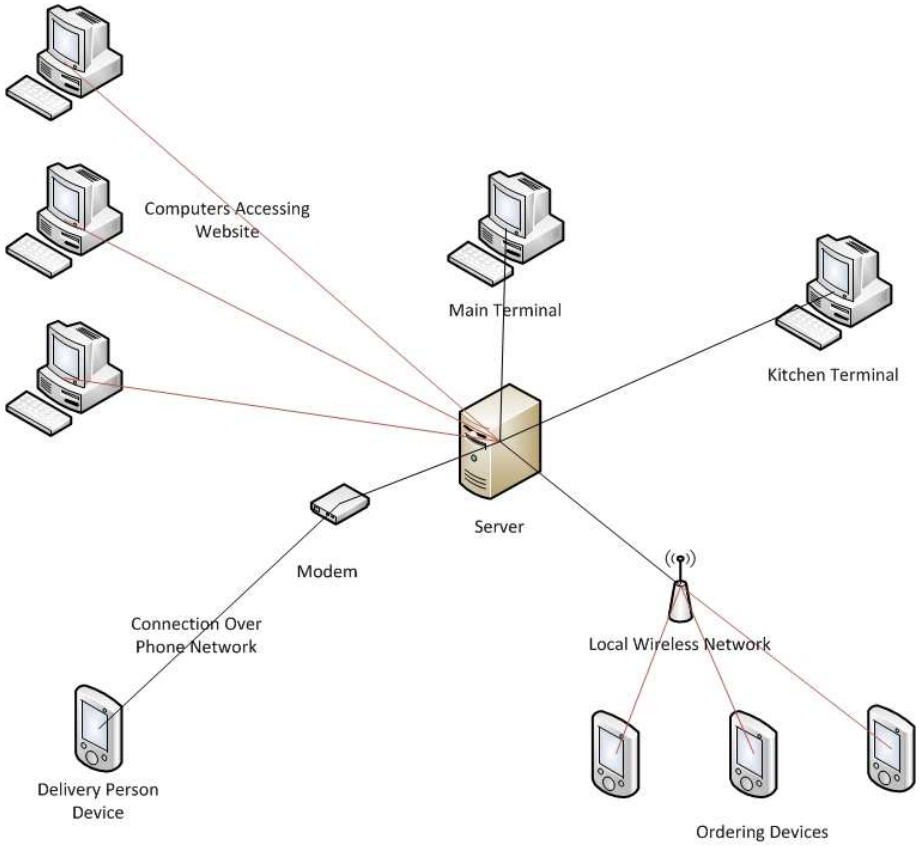
Wireless Order Placement

- Allow waiter to input order using a wireless iOS or Android Device
- Allow proprietor to embed iOS or Android device in tables to allow customers to place their own orders

Mobile Ordering Updates

- Allow updates to be sent to a delivery person using a mobile device running iOS or Android to alert him/her of upcoming orders

System Network



System Actors' Roles

Management Tools

- Statistical analysis tools: graphical and numerical breakdown of customer dining time, menu item popularity, stock of ingredients, etc.
- Employee management permissions:
    - Add/Remove employees
    - Customize employee permissions
    - Create/Customize employee schedules
    - Update restaurant floor plan layout
    - Access/Modify database records

Host

- Assign customers to table
- Check the wait time for available tables
- Schedule reservations for customers

Waiter/Waitress

- View/Update status of tables the server is assigned to
- Place/Edit/Remove order for specific tables

Bartender

- Place/Edit/Remove orders
- Receive drink orders placed by waiter/waitress

Chef

- Receive order placed by waiter/waitress
- Update completion of order
- Update stock of ingredients

Customer

- Online order/reservation placement
- In restaurant ordering from embedded iOS or Android device

The Old System

While the old pen and paper system may be suitable for smaller operations, any restaurant that receives a large throughput of customers can't be run using these outdated methods. The updated computer system increases efficiency, giving the restaurant a competitive edge by reducing overhead. The use of paper is virtually eliminated. In conjunction with the ability to base ordering of ingredients on the computer analysis of stock and item popularity, operating costs are drastically reduced. This system also allows the proprietor the option of running a "no waiter" restaurant that allows the customer to place their own order on devices embedded in the tables, reducing the number of employees and lowering cost. Overall, implementing a computer system for a restaurant will result in higher efficiency and higher profits.

b. Glossary of Terms

Manager – manages all aspects of the restaurant. Given the largest amount of permissions to access parts of the system restricted to the average user.

Employee – any of the users currently hired by the restaurant. Consists of manager, host, waiter/waitress, chef, busboy, delivery person, and bar tender.

Host – seats the customers to a specific table when a table is available. Also handles making reservations.

Waiter/Waitress – attends to customer. Place order either on the main terminal or through use of a mobile device.

Chef – cooks orders in the order which they are received. Updates the order status when they are complete.

Busboy – clears off tables that are marked as dirty. Updates the table status to available once cleaning is completed.

Delivery Person – delivers orders. Also has ability to place orders if need be. Receive updates to a mobile device about future orders as well as sends notification back to the restaurant when orders have been delivered.

Bar Tender – receives drink orders and makes them. Updates drink order status as completed once done. Also has the ability to place food orders.

Customer – patron of the restaurant. Gives order to waiter/waitress or places order on an embedded device in the table. Also able to place reservation using online application.

Main Terminal – the central terminal in the restaurant that all employees have access too. Able to place orders, clock in/out, update table status, etc. May be multiple depending on the size of the restaurant.

Kitchen Terminal – terminal in the kitchen that notifies the chef of food orders and allows chef to update the order status.

Mobile Device – device used to place orders by the waiter/waitress, or, in the case of the delivery driver, a device used to receive future orders and update delivery order status.

2. System Requirements

a. Enumerated Functional Requirements

	Rank	Description
REQ1	5	The system shall allow managers to control the payroll. This portion of the system should be accessible only by management via the web application. The payroll portion of the system should allow managers to add employees, remove employees, and update employee information in the database storing employee information. It should also allow managers to pay employees (discussed in more detail in REQ7)
REQ2	3	The system should allow managers to view statistics showing the overall performance of the restaurant. This will include a display that allows managers to query about how much a particular employee has worked, figures showing the state of the inventory, what items are on the menu and how much of them have been sold, and overall financial figures on either a weekly or monthly basis. The statistical analysis will only be availabe to managers through the web application.
REQ3	2	The system shall have an inventory list that specifies what items are in stock, out of stock, and close to being out of stock. The inventory system shall be populated by the manager initially and whenever new items are delivered to the restaurant. It should then be automatically updated to reflect any changes to the menu that have been made. The inventory list should adjust based upon customer orders and will remove items from the menu on the mobile applications of the customer when stock runs low Please refer to the mathematical model section for more detail). Additionally, the managers should have the capability to reconcile inventory at the end of the day or week to reflect spoiled food or other unforeseen circumstances.
REQ4	4	The system shall allow the manager to alter menu items via the web application. When a menu item is added, it should be reflected in the inventory list of the statistical management system as having no stock. Thus, it should not appear on the menus of customer mobile devices until stock is ordered and the manager updates the inventory list to reflect new stock. When a menu item is deleted, it will be removed from the menu display on the customer mobile application.

REQ5	4	The system shall allow personnel to view the layout of the restaurant as a 2D diagram that mimics an aerial view of the floor. The layout will show tables and their configuration, and the tables will be colored to show the their status. Green will indicated an open table, yellow will indicate a vacant table that must be cleaned and preperad for another customer, and red will indicate and table currently in use. The host will be capable of marking a table red via her terminal application. The server will be capable of marking the table yellow via his mobile application. The busboy will be capable for marking a table green via his terminal application. The system will allow alteration of the floor plan by managers via a web application and a hostess via the hostess terminal application.
REQ6	2	The system shall allow managers, waiters, and hostesses to both physically move the tables in the restaurant and change the diagram showing table layout through their respective devices describe in the table layout application requirement.
REQ7	5	The system shall allow managers to properly pay employees. Employees shall be able to clock in and out of the system, with the time work being logged. If an employee fails to clock out of the system after a maximum threshold is crossed, he will automatically be logged out. This threshold, along with hourly pay and overtime pay, will be stored database along with the name, address, and social security number of the employee. The payroll should allow the manager to release payment calculated through hours logged and employee information once every two weeks via the web application. Upon payment, the employee hours shall be populated into database for statistical analysis and the pay information shall be reset.
REQ8	5	The system shall allow both customers and waiters to place orders that are sent directly to the screens of the chef terminal in the kitchen. Customers will be capable of ordering through a mobile device located at their table, while waiters will be able to place order through the mobile device they carry on their person at all times. Alcoholic drink orders can not be placed via the customer application, and servers will check for identification at the table if alcohol is ordered.
REQ9	4	The system should populate delivery orders on the screens of the chef terminal and the deliveryman mobile application. The chefs can then remove the order from their screen once it has been prepared, and the deliveryman can remove the order from his mobile application when it has been delivered. If the order does not make it to the customer, he or she can call to complain or submit a complaint online via the web application.
REQ10	3	The system should allow customers to beckon a waiter via the mobile application located at their table. To limit the complexity of the system and opportunity for customer tampering, the customer beckoning will simply show up on the table layout diagram as a dot layered on top of the red table. This can be viewed by anyone with access to the table layout diagram through the devices described in that requirement.
REQ11	5	The system shall have a menu screen on the mobile application of customers for customer ordering. The menu screen should allow the customer to place his order through the customer mobile application.

REQ12	1	The system should allow reservation to be made online via the web application. Online reservations will only be allowed if the reservation is placed 24 hours in advance of the time requested. Otherwise, the customer can call the host and the system should allow the host to input the reservation if it is available via the host terminal application. Regardless of online or phone reservation, the system shall prompt the customer or host respectively for name and phone number. If the customer is logged in via the web application, this information will be retained.
REQ13	1	The system should allow accounts to be made for future login by the same customer. This feature is predominatly for delivery order, so that customer information such as name, phone number, address, and credit card information can be retained. Thus, the customer won't be forced to re-enter information each time an order is placed.
REQ14	5	The system should allow customers to pay for their check at the table via their mobile application. Payment information should be entered via a prompt on the screen. Only electronic payment will be allowed via the customer mobile application. The customer can alert the server either through the mobile application or personally if another payment method is desired. The server can make payment through the server mobile application (credit cards only) or terminal (all acceptable forms of payments).
REQ15	2	The system should allow an order change or cancellation to the customer via her mobile application. The system should post the proposed change or cancellation on the screens of the chef terminal in the kitchen. There, the request can either be accepted or denied. The chef terminal should then relay the appropriate message back to the customer mobile application. If the customers feels that they have been wronged, they can log a complaint via the web or seek a manager personally.
REQ16	4	The system shall provide the host with the tools and information to either seat patrons in a reasonable amount of time or provide customers with an estimate of wait time. A reasonable amount of time will be defined as 3 minutes. Thus, the host terminal will show the table layout, reservations made via phone or online, and the customer wait list. All of these items can be manipulated via the host terminal to alter the restaurant layout for larger parties, change reservation statuses for new/cancelled reservations, and add/remove customers from the waitlist. Wait time will be estimated by the hostess rather than by mathematical formula.
REQ17	5	The system shall have a group of screens controlled by a computer terminal in the kitchen for chefs where the orders are listed in the order that they came in. The screens will be run by the chef terminal application located on the computer. 10 orders will be able to fit on a screen, with the number of total screens to be determined by the total number of tables in the restaurant divided by 10. After an order has been on the screen for 20-30 minutes, it will blink or become a shade of red to indicate urgency. Because the screen will be the primary avenue for the chef to view what to cook and is in proximity to the grill/oven/etc., the blinking order will notify the chefs complete the order or face customer complaint.

b. Enumerated Non-Functional Requirements

Non-functional requirements are those requirements that articulate the goals of the software development beyond the core necessities of the project. Such pieces of information work to explain how choices have been made to best allow the program to make the jump from a testing environment into the wild. The five categories of non-functional requirements are functionality, usability, reliability, performance, and supportability.

	Rank	Description
Functionality		
REQ18	3	The system should be designed is such as way that it could be extended to a different restaurant with a different business model in less than a month
REQ19	3	The system should utilize technology that is flexible enough to work on various platforms and new hardware. In the event that the software does not work on a particular piece of hardware, it should take no more than a month to port the software to the different hardware configuration.
Usability		
REQ19	4	The various interfaces of the host terminal (table layout, reservation, waitlist) will be available in one click from the root host menu
REQ21	5	The UI of the chef terminal should always show the current order queue. The chef should be able to notify the server that food is ready in 3 clicks. The order queue should update automatically on the placement of an order or the completion of notification of food completion.
REQ22	5	The screens of the chef terminal should be in view of the chefs and text should be large enough to read from 10 ft. away.
REQ23	4	There should be enough server terminals such that it doesn't take a server longer than 15 seconds to reach his terminal from the farthest table he is serving.
REQ24	2	The manager should be capable of adding, removing, or updating employee information in less than 3 minutes.
REQ25	2	The manager should be capable of viewing pay information and releasing payment to employees in less than 3 minutes.
REQ26	5	The user interfaces shall be aesthetically pleasing both to customers and to employees. Aesthetically pleasing will be generally defined as having a favorable appearance to 80% of 10 customers and 10 employees asked about the appearance of the program at random.
Reliability		
REQ27	4	The hostess terminal shall be reliable to the extent that it does not crash more than once every 2 months.
REQ28	5	The chef terminal shall be reliable to the extent that it does not crash more than once every 2 months
REQ29	5	The customer mobile application should reliable to the extent that it does not crash more than once every 6 months. Because the customer application will connect to a local wireless router, lack of internet access should not serve as an impediment to the customer mobile application.

REQ30	3	The web application should be reliable to the extent that it does not crash more than once per month.
REQ31	5	The system shall function in the absence of an internet connection. While the web application would not function in such a case, all the standard operations of the restaurant could occur. Once internet access is restored, the manager could update any necessary parts of the system.
REQ32	2	The server shall have the built-in capability such that networking it with a central server or backing up any data on it would take less than 3 days.
REQ33	1	The server shall have the capability of updating code dynamically to keep uptime above 99% in the event that updates or fixes are required.
Performance		
REQ34	4	There should be a delay of no longer than 400ms for all operations on all devices except for statistical queries run from the web application on the database. A 1 second delay is acceptable for database queries with mathematical calculations.
Supportability		
REQ35	3	The system design and implementation should be documented and simple to the extent that a capable IT company could perform updates and fixes within 3 days

c. On-Screen Appearance Requirements

Since our project is a multi-platform design (android-based, computer terminal-based, and web-based), we will have many on screen requirements we have to handle. Designing in just android brings up many requirements. Being that android is a multi-device platform, we will have to design our UI in a way that will accommodate all different types of devices. For example, there are android devices in the form of tablets and phones that can be used with our system. Both android tablets and phones comes in a wide variety of screen sizes and pixel densities. Since it easy to port an Android app to run on a PC(and thus a web interface) using porting programs such as "YouWave" , we will only focus on the on screen requirements for Android devices. As well as hardware requirements, we also have to address usability requirements. Our system should not only have a clean and polished look, but also be extremely easy to use for experienced users and new users alike.

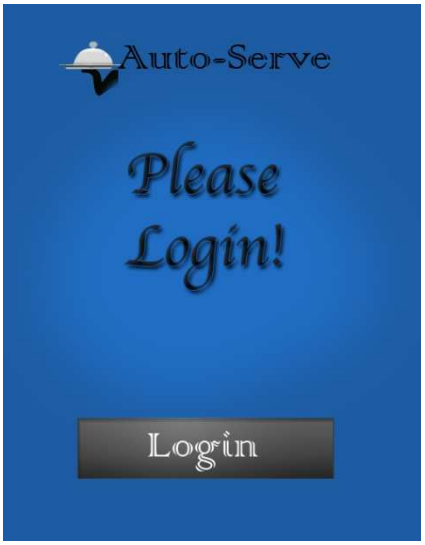
Android suggests making three different versions of your UI to accommodate screens that are “Low Density”, “Medium Density”, and “Large Density”. But what about the different sized screens within these categories? The best way to deal with this problem is so design “Fluid” layouts, meaning that no matter what size screen one has, the program will automatically re-size the UI to fit the screen. Luckily, the Android framework has this feature built-in. There are various kind of layouts available (LinearLayout, FrameLayout etc.) that will gracefully fit images based on screen resolution of the device.

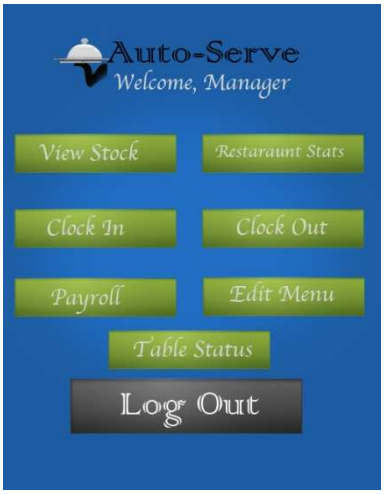
To accommodate on screen usability requirements, we will design a UI that will require a minimal amount of button presses to get to important features (5 or less), an intuitive design flow, big buttons that can be ready from a distance of at least 2 feet away, and buttons that have icons that show what the feature does in the case that someone that is not using the system does not speak English as their first language.

Below is a table with a prioritized table of on screen requirements:

Identifier	Priority	Requirement
UIREQ-1	5	The UI shall be able to re-size itself to be able to fit on different sized screens and devices.
UIREQ-2	5	The UI shall have multiple sized versions for different screen densities.
UIREQ-3	5	The UI shall be designed in a way that is easily ported to a PC and web-based application.
UIREQ-4	4	The UI shall take no more than 5 button presses (besides typing characters to log in) to reach desired program features.
UIREQ-5	3	The UI should have buttons that can be seen from at least 2 feet away
UIREQ-6	4	The UI shall have buttons with icons that easily describe program features to users that do not speak English as a primary language.
UIREQ-7	3	The UI shall have an intuitive flow that can be easily used by new users as well as experienced users.
UIREQ-8	1	The UI should have multiple versions in different languages
UIREQ-9	5	The UI should have a clean and polished look that produces a positive user experience

Below are some screen shots showing preliminary designs on how we plan on overcoming UIREQ-2,UIREQ-4, and UIREQ-5. These images depict a UI that has big, easily read buttons, a navigation path that only take 3 button presses to choose a feature from the options menu, and multiple sized screens to accomadate a phone and a table android device.





3. Functional Requirements Specification

a. Stakeholders

Usually, programming projects have five main stakeholders: the project managers, software testers, program architects and developers, system analysts, and lastly clients. For the purpose of this project, our group as a whole will assume the role of project managers, software testers, program architects and developers and system analysts. Thus, the main focus for our stakeholders will be the clients.

Our stakeholders are divided into two main categories:

1. Actors
    a. This includes all of our program actors: customer, waitress, hostess, bartender, manager, and delivery boy.

b. These actors will be interacting directly with our system.

2. Companies that will buy the system
    a. This would include Owners of the restaurant/restaurant chain.
    b. These people generally will not be interacting directly with our system.

b. Actors and Goals

Initiating Actors:

1. Customer
    a. This actor places a new order through the system at the table. The order that is placed is relayed to the kitchen (UC-3: PlaceOrder)
    b. This actor can edit a current order as long as the order has not been completed (UC-5: EditOrder)
    c. This actor can request a reservation (UC-1: MakeReservation)
2. Chef
    a. This actor clocks in and clocks out when entering/leaving work (UC-25: ClockIn, UC-26: ClockOut)
    b. This actor reports on the progress of a particular order (UC-10: UpdateFoodProfress)
    c. This actor cancels an item from a current order in the case of a mistake (UC-6:CancelOrder)
    d. This actor has access to the statistics of the inventory of the restaurant (UC-16: ViewStockStats)
3. Server
    a. This actor clocks in and clocks out when entering/leaving work (UC-25: ClockIn, UC-26: ClockOut)
    b. This actor logs into the system to their specific interface (UC-31: AuthenticateUser)
    c. This actor adds/removes items from an order (UC-5: EditOrder)
    d. This actor places orders for customers when a customer is having difficulty (UC-4: PlaceOrder)
    e. This actor collects payment from tables (UC-7: PayBill)
    f. This actor changes the status of a table when a table is either occupied or empty (UC-11: TableOccupied, UC-12: TableClean, UC-13: TableDirty)
4. Host
    a. This actor clocks in and clocks out when entering/leaving work (UC-25: ClockIn, UC-26: ClockOut)
    b. This actor logs into the system to their specific interface (UC-31: AuthenticateUser)
    c. This actor seats a Customer at a particular table when requested by the customer (UC-28: SeatCustomer)
    d. This actor can alter the floor plan of the restaurant for situations such as large parties (EditLayout)

5. Manager
    a. This actor clocks in and clocks out when entering/leaving work (UC-25: ClockIn, UC-26: ClockOut)
    b. This actor logs into the system to their specific interface (UC-31: AuthenticateUser)
    c. This actor adds/removes employees from the database as well as updating an employee's information (UC-22: AddEmployee, UC-23: RemoveEmployee, UC-24: UpdateEmployee)
    d. This actor updates the menu by adding/removing items (UC-18: AddtoMenu, UC-19: DeleteFromMenu)
    e. This actor has access to the statistics of the inventory of the restaurant (UC-16: ViewStockStatus)
    f. This actor can view the restaurant statistics, which includes financial statistics (UC-17: ViewStats)
    g. This actor can alter the floor plan of the restaurant for situations such as large parties (UC-3: EditLayout)
    h. This actor can issue payment to employees (UC-20, ReleasePayment)
6. Delivery Boy
    a. This actor clocks in and clocks out when entering/leaving work (UC-25: ClockIn, UC-26: ClockOut)
    b. This actor logs into the system to their specific interface (UC-31: AuthenticateUser)
    c. This actor notifies the system when an order has been completed (UC-21: CompletedDelivery)

Participating Actors:

1. Chef:
    a. Based on the queue, this actor prepares food.
2. Server
    a. This actor brings food and drinks to a table when ready.
    b. This actor also responds to Customer's calls.
3. Busboy
    a. This actor is responsible for cleaning tables at the restaurant once the server marks the table as dirty.

c. Uses Cases

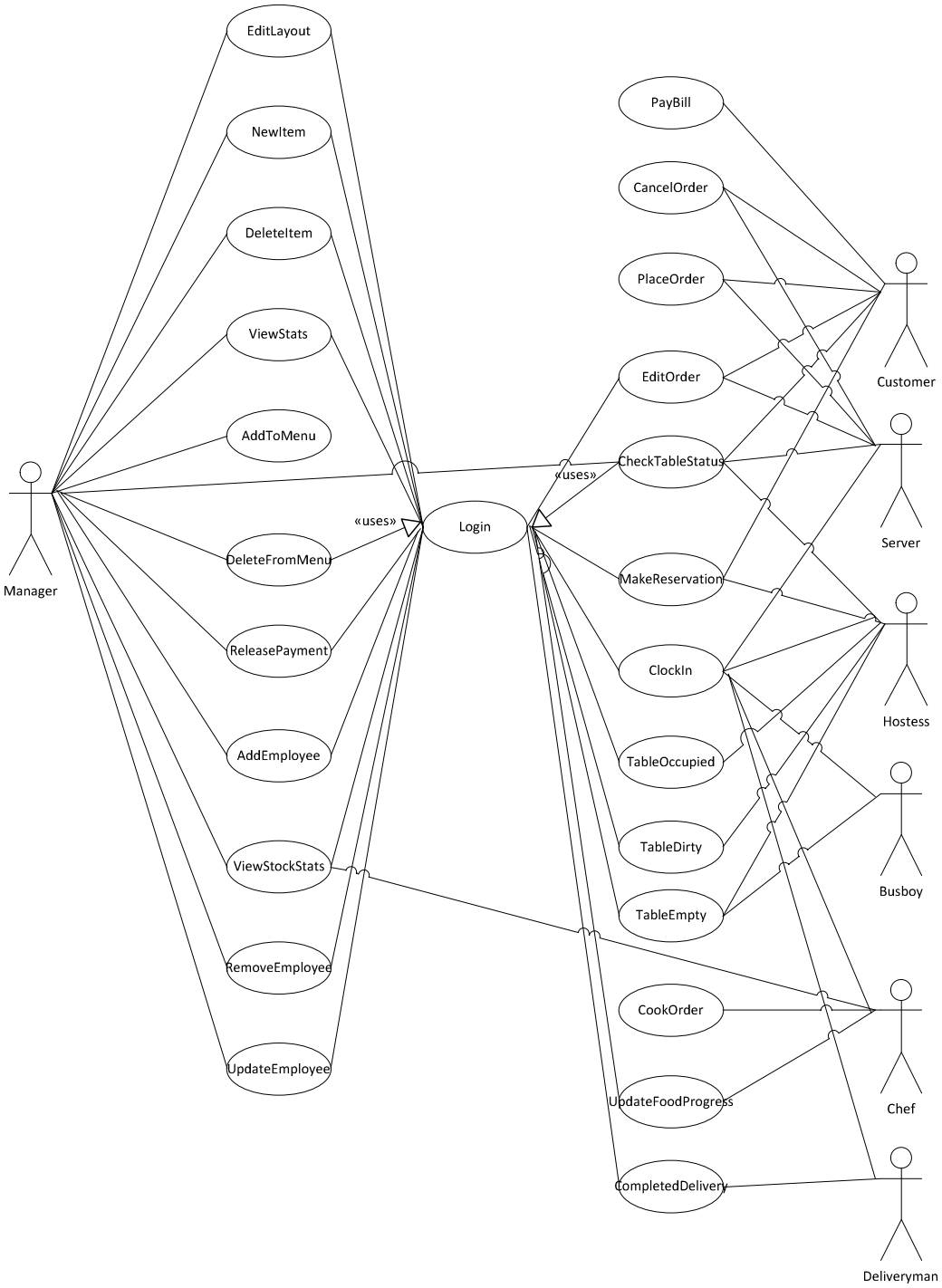
i. Casual Description

USE CASE	DESCRIPTION
MakeReservation (UC-1)	Allows a *customer/hostess* to place a reservation.
CheckTableStatus (UC-2)	Allows *all personnel* to see whether a restaurant table is empty, occupied, or dirty.
EditLayout (UC-3)	Allows the *hostess/manager* to change the layout of restaurant tables, including the combination of several tables into one to accommodate larger groups.
PlaceOrder (UC-4)	The *customer/waiters* place an order through the mobile application system for their respective tables.
EditOrder (UC-5)	Allows a *customer/waiter* to add/modify an order that has been placed.
CancelOrder (UC-6)	A *customer/waiter* cancels an order that has already been placed.
PayBill (UC-7)	Allows a *customer* to beckon the *waiter* for a check/receipt.
CookOrder (UC-8)	The *chef* prepares the food according the customers' orders.
UpdateFoodProgress (UC-9)	The *chef* starts/updates the food progress meter with the push of a button.
(UC-10)	Intentionally omitted to preserve numbering system
TableOccupied (UC-11)	Allows the *hostess* to denote a table occupied.
TableClean (UC-12)	Allows the *hostess/busboy* to change the table status to clean to denote a clean table ready for new customers.
TableDirty (UC-13)	Allows the *hostess* to change the table status to dirty and notify the

	busboy.
NewItem (UC-14)	Allows the *manager* to create a new inventory item.
DeleteItem (UC-15)	Allows the *manager* to delete an item from the inventory.
ViewStockStats (UC-16)	Allows the *chef/manager* to check the current inventory stocks in the restaurant.
ViewStats (UC-17)	Allows the *manager* to see the financial and ordering statistics.
AddtoMenu (UC-18)	Allows the *manager* to add an item to the restaurant menu.
DeleteFromMenu (UC-19)	Allows the *manager* to delete an item from the restaurant menu.
ReleasePayment (UC-20)	Outputs the pay of *restaurant personnel* on a monthly basis, resetting hours worked back to 0 after payout.
CompletedDelivery (UC-21)	Once the *delivery boy* finishes a delivery, the order is removed from the list of queued items.
AddEmployee (UC-22)	Allows the *manager* to create a new employee account and place it in the database.
RemoveEmployee (UC-23)	Allows the *manager* to delete an employee account from the database.
UpdateEmployee (UC-24)	All the *manager* to update employee information in the database
ClockIn (UC-25)	The *restaurant personnel* notify the system of when they started working.
ClockOut (UC-26)	The *restaurant personnel* notify the system of their departure time and adds the total hours clocked that day to their monthly totals.
RequestTable (UC-27)	The *customer* asks to be seated at a certain table or for an open table.

SeatCustomer (UC-28)	The *hostess* assigns a table to the *customer*.
GetPartySize (UC-29)	Notifies the *hostess* of the number of people to arrive or be seated with a *customer's* request.
CleanTable (UC-30)	The *busboy* is responsible for cleaning a particular table when a party gets up and leaves the restaurant.
AuthenticateUser (UC-31)	The *hostess, server,* and *manager* will be verified that they have the authorization to enter a particular interface.

ii. Use Case Diagram



iii. Fully Dressed Description

Use Case UC-16	ViewStockStats (UC-16)
Related Requirements:	REQ2, REQ3, REQ17
Initiating Actor:	Any of: Manager, Chef
Actor's Goal:	To be able to view the level of stock of each inventoried item
Participating Actors:	Any of: Manager, Chef
Precondition:	There must be menu items to view the stock of.
Postcondition:	
Flow of Events for Main Success Scenario:	→ 1. Manager/Chef logs in to system 2. include::*AuthenticateUser* → 3. Manager/Chef chooses from menu "View Stock Statistics." ←5. System displays stats to Manager/Chef
Flow of Events for Extensions (Alternate Scenarios):	2a. *AuthenticateUser* unsuccessful. 1. System signals *AuthenticateUser* was unsuccesful. 2. Manager/Chef re-enters password.

Use Case UC-17	ViewStats
Related Requirements:	REQ1, REQ2, REQ3, REQ4, REQ7
Initiating Actor:	Any of: Manager
Actor's Goal:	To be able to view the statistics of menu items. How much of each item is ordered when its ordered, etc
Participating Actors:	Any of: Manager, Chef
Precondition:	There must already be statistical data for each item.
Postcondition:	
Flow of Events for Main Success Scenario:	→ 1. Manager logs in to system 2. include::*AuthenticateUser* → 3. Manager chooses from menu "View Restaraunt Statistics" ← 5. System displays stats to Manager
Flow of Events for Extensions (Alternate Scenarios):	2a. *AuthenticateUser* unsuccessful. → 1. System signals *AuthenticateUser* unsuccessful. ← 2. Manager re-enters password.

Use Case UC-31	AuthenticateUser
Related Requirements:	All non-customer requirements
Initiating Actors:	Any of: Manager, Hostess, Waiter, Busboy, Delivery Person.
Actor's Goal:	To be positively identified by the system and to access the program tools after successfully logging into the system
Participating Actors:	
Precondition:	System is connected to server. Valid users are in database.
Postcondition:	User successfully logs into the system
Flow of Events for Main Success Scenario:	→ 1. User selects *AuthenticateUser* screen on the main screen. ← 2. System prompts the actor to enter username and password. → 3 User enters information and submits to the system. ← 4. System (a) validates data in the database and confirms correct username and password (b) logs in the actor to appropriate user interface.
Flow of Events for Extensions (Alternate Scenarios):	4a. User entered invalid information ← 1. System could not validate username and password based on database entries. ← 2. System sends an error message to the actor indicating that the data entered is invalid → 3. User enters information and submits to the system. 4. Same as step 4.

Use Case UC-4	PlaceOrder
Related Requirements:	REQ8, REQ9, REQ17
Initiating Actor:	Any of: Customer, Waiter
Actor's Goal:	To record the orders to be relayed to the kitchen for preparation.
Participating Actors:	Any of: Chef
Precondition:	The system is connected to the server. The table has been designated as occupied.
Postcondition:	An order is placed, and the chef is notified.
Flow of Events for Main Success Scenario:	→ 1. Customer/Waiter selects "Add Item" on system interface. 2. include::*AuthenticateUser* ← 3. System provides the actor with an electronic menu. → 4. Customer/Waiter selects and item on the menu. ← 5. System provides specific selection criteria for item chosen. → 6. Customer/Waiter placed the order and the item is added to the bill. ← 7. System notifies the Chef that the order is placed.
Flow of Events for Extensions (Alternate Scenarios):	3a. Customer asks Waiter for help on ordering → 1. Waiter orders for customer on a portable interface ← 2. System relays the ordered item to the kitchen/Chef ← 3. Same as step 6 above 6a. Item is out of stock.

	← 1. System notifies actor that item is unavailable and shows main menu. → 2. Customer/Waiter selects “Add Item” on system interface. 3. Same as step 4. 6b. Customer/Waiter cancels order. ← 1. System returns to main menu. → 2. Customer/Waiter selects “Add Item” on system interface. 3. Same as step 4.

Use Case UC-2	CheckTablesStatus
Related Requirements:	REQ5, REQ6
Initiating Actor:	Any of: Hostess
Actor's Goal:	Determine what tables are clean, occupied, dirty or reserved.
Participating Actors:	Any of: Waiter, Busboy
Precondition:	There are tables in the restaurant.
Postcondition:	System returned status on particular table.
Flow of Events for Main Success Scenario:	→ 1. Customer requests a table by walking in, over the telephone, or online. 2. include::*RequestTable* → 3. Hostess checks the status of the tables in the restaurant. 4. include::*AuthenticateUser* ← 5. System returns the status of the tables in the restaurant. 6. include::*SitCustomer*
Flow of Events for Extensions (Alternate Scenarios):	3a. System returns that the table does not exists. ← 1. System asks the Hostess/Waiter/Busboy to choose a different table. 2. Same as step 3 above

Use Case UC-14, UC-15	EditMenuItems
Related Requirements:	REQ3, REQ3, REQ4, REQ11
Initiating Actor:	Any of: Manager
Actor's Goal:	To add or delete an item from the menu.
Participating Actors:	
Precondition:	The system is connected to the server.
Postcondition:	Menu is altered and changes are successfully made.
Flow of Events for Main Success Scenario:	→ 1. Manager wishes to alter the menu by adding or deleting a menu item. 2. include::*AuthenticateUser* ← 3. System returns a prompt with a list of the menu items as well as an *AddItems* option. → 4. Manager chooses the item he wishes to delete or presses the *AddItems* option. ← 5. System returns that the change was possible, and the menu has been updated.
Flow of Events for Extensions (Alternate Scenarios):	5a. System states that the item to be added is already on the menu. ← 1. System asks the Manager to add a different item. 2. Same as step 4 above

Use Case UC-25	ClockIn
Related Requirements:	REQ1, REQ2, REQ7
Initiating Actor:	Any of: Manager, server, host(ess), busboy, chef, or bartender
Actor's Goal:	Establish the starting time for an employee shift
Participating Actors:	
Precondition:	System connected to the server. User logged into the system. User not currently clocked in. User is an employee of the restaurant on present payroll list. User has not worked above maximum hours allowable for day/week/month. ClockIn time during open restaurant hours && employee scheduled to work.
Postcondition:	ClockIn time stored in system UI graphics updated
Flow of Events for Main Success Scenario:	→ 1. Employee notifies system system of wish to clock in. ← 2. System updates active user list in the system, populates any appropriate user interface graphics, and returns affirmative ClockIn signal to the user.
Flow of Events for Extensions (Alternate Scenarios):	← 2a. System returns a negative ClockIn signal to the user if a precondition is not met and notifies manager.

Use Case UC-26	ClockOut
Related Requirements:	REQ1, REQ2, REQ7
Initiating Actor:	Any of: Manager, server, host(ess), busboy, chef, or bartender
Actor's Goal:	Enter concluding time for a work shift
Participating Actors:	
Precondition:	User is clocked into the system.
Postcondition:	Update active user. UI graphics updated. Book wages to payroll.
Flow of Events for Main Success Scenario:	→ 1. Employee notifies system system of wish to clock out ← 2. System updates active user list in the system, populates any appropriate user interface graphics, books appropriate wages, and returns affirmative ClockOut signal to the user.
Flow of Events for Extensions (Alternate Scenarios):	→ 1a. System requests auto-clockOut if end of scheduled work interval is breached. ← 2a. Same procedure as 2, but manager notified.

Use Case UC-22	AddEmployee
Related Requirements:	REQ1, REQ2, REQ 7
Initiating Actor:	Any of: Manager
Actor's Goal:	Add employee to payroll system
Participating Actors:	
Precondition:	System is connected to the server Manager is logged into the system
Postcondition:	New employee and information added to employee list
Flow of Events for Main Success Scenario:	→ 1. Manager notifies system of wish to add a new employee ← 2. System prompts manager for employee information through UI form → 3. Manager inputs employee information ← 4. System returns that employee has been succesfully added to the payroll
Flow of Events for Extensions (Alternate Scenarios):	← 4a. System returns that information was entered incorrectly and returns to step 3

Use Case UC-23	RemoveEmployee
Related Requirements:	REQ1, REQ2, REQ 7
Initiating Actor:	Any of: Manager
Actor's Goal:	Remove an employee from the payroll system
Participating Actors:	
Precondition:	System is connected to the server Manager is logged into the system Employee is clocked out
Postcondition:	Employee is removed from employee list
Flow of Events for Main Success Scenario:	→ 1. Manager notifies system of wish to remove an employee from the payroll ← 2. System prompts manager for employee to remove through UI form → 3. Manager inputs employee information ← 4. System returns that employee has successfully been removed
Flow of Events for Extensions (Alternate Scenarios):	← 3a. Manager inputs incorrect information

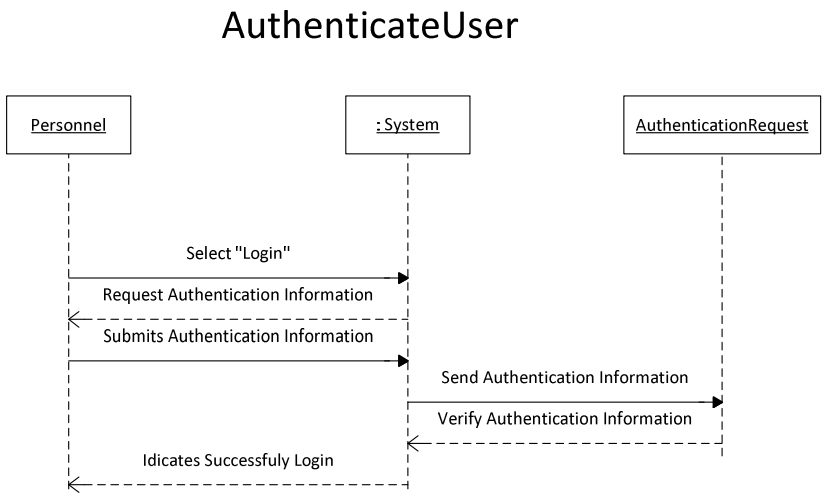
Use Case UC-24	UpdateEmployee
Related Requirements:	REQ1, REQ2, REQ 7
Initiating Actor:	Any of: Manager
Actor's Goal:	Update employee information in the payroll system
Participating Actors:	
Precondition:	System is connected to the server Manager is logged into the system Employee exists in payroll database
Postcondition:	Employee information updated
Flow of Events for Main Success Scenario:	→ 1. Manager notifies system of wish to update employee information ← 2. System prompts manager for employee information on UI form → 3. Manager inputs updated employee information ← 4. System returns that employee has successfully been updated
Flow of Events for Extensions (Alternate Scenarios):	← 3a. Manager inputs incorrect information

Use Case UC-20	ReleasePayment
Related Requirements:	REQ1. REQ2, REQ7
Initiating Actor:	Any of: Manager
Actor's Goal:	Pay employees and update the payroll
Participating Actors:	
Precondition:	System is connected to the server Manager is logged into the system Payroll information exists
Postcondition:	Paystubs generated Statistics updated Payroll reset
Flow of Events for Main Success Scenario:	→ 1. Manager notifies system of wish to release payment ← 2. System generates paystubs, updates statistics, and resets payroll data to initial conditions.
Flow of Events for Extensions (Alternate Scenarios):	

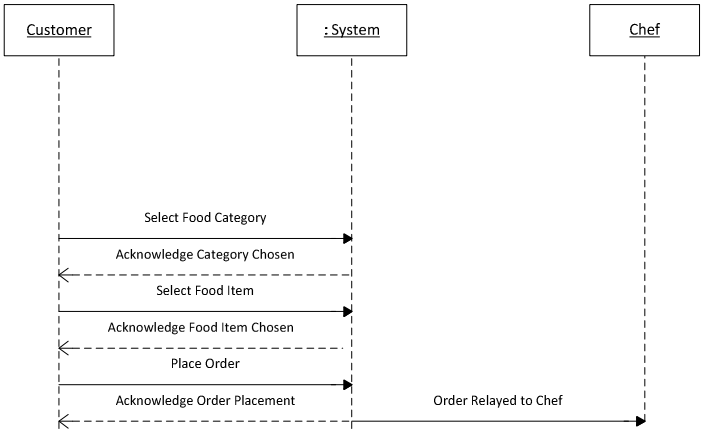
iv. Traceability Matrix

		USE CASE																													
REQ	**PW**	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8	UC-9	UC-10	UC-11	UC-12	UC-13	UC-14	UC-15	UC-16	UC-17	UC-18	UC-19	UC-20	UC-21	UC-22	UC-23	UC-24	UC-25	UC-26	UC-27	UC-28	UC-29	UC-30	UC-31
REQ1	5																X	X			X		X	X	X	X	X					X
REQ2	3							X					X				X	X			X		X	X	X	X	X		X			X
REQ3	2								X						X	X	X	X		X												X
REQ4	4														X		X		X													X
REQ5	4	X	X	X								X	X	X														X	X	X	X	X
REQ6	2	X	X	X								X	X	X														X	X	X	X	X
REQ7	5																	X			X		X	X	X	X	X					X
REQ8	5				X	X	X			X																						X
REQ9	4				X	X	X			X												X										X
REQ10	3																															X
REQ11	5				X	X	X			X																						X
REQ12	1	X	X									X																				X
REQ13	1																															X
REQ14	5							X																								X
REQ15	2					X																										X
REQ16	4	X	X	X								X	X	X														X	X	X	X	X
REQ17	5				X	X	X		X	X																						X
MAX PW		4	4	4	5	5	5	5	5	5	0	4	4	4	4	2	5	5	4	2	5	4	5	5	5	5	5	4	4	4	4	5
Total PW		11	11	10	19	21	19	8	7	19	0	11	13	10	6	2	14	15	4	2	13	4	13	13	13	13	13	10	13	10	10	59

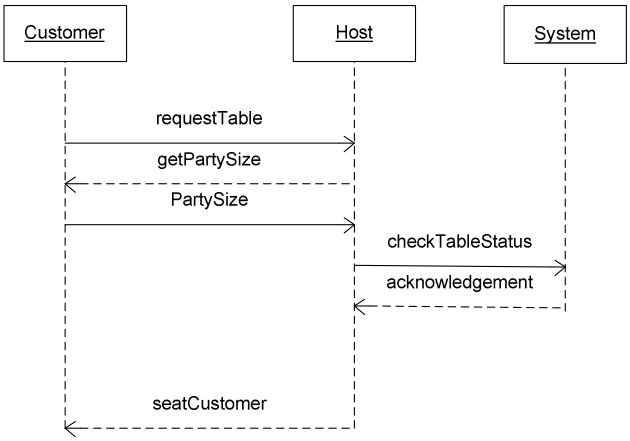
d. System Sequence Diagrams



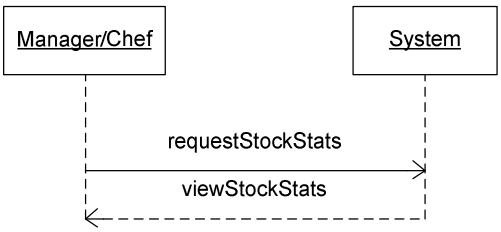
PlaceOrder

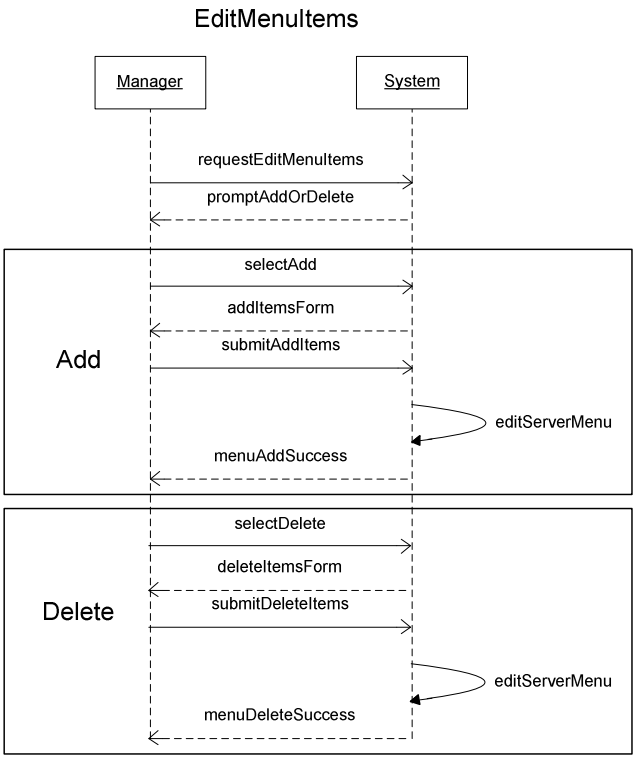


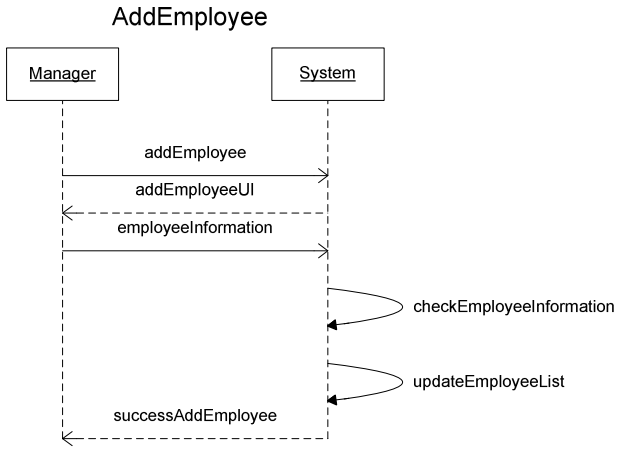
SeatCustomer

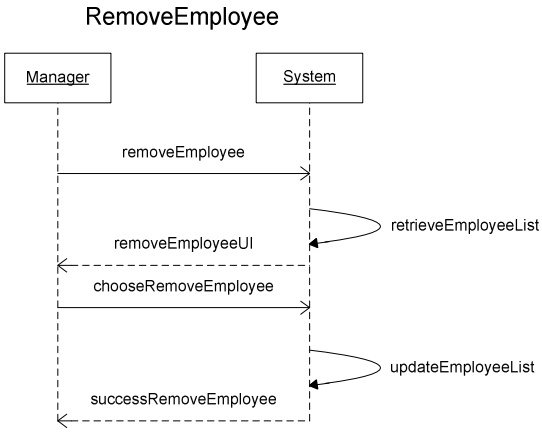


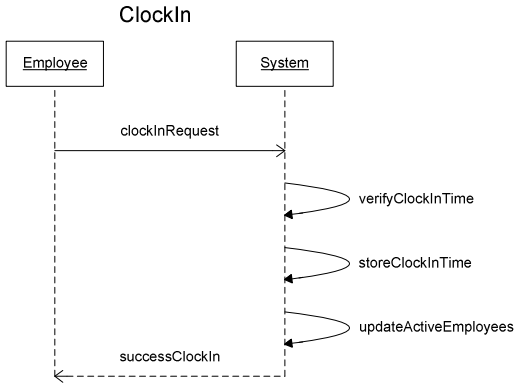
StockStats

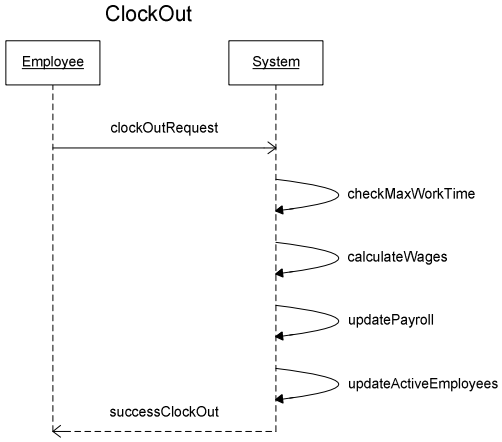


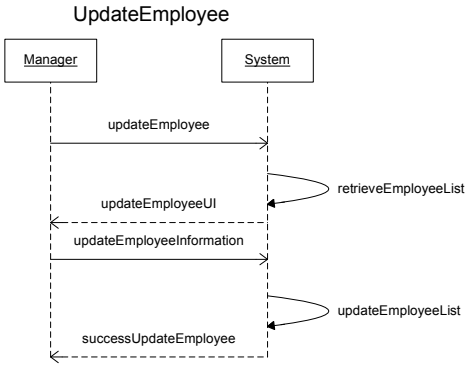

EditMenuItems
Manager
System
requestEditMenuItems
promptAddOrDelete
selectAdd
addItemsForm
submitAddItems
Add
editServerMenu
menuAddSuccess
selectDelete
deleteItemsForm
submitDeleteItems
Delete
editServerMenu
menuDeleteSuccess

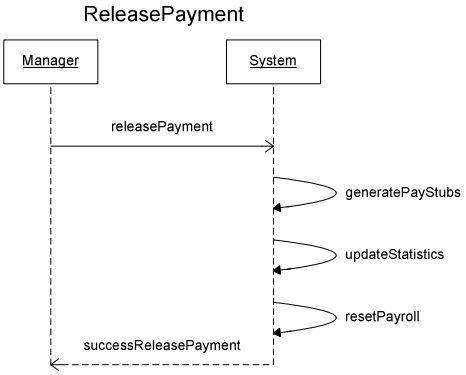

AddEmployee
Manager
System
addEmployee
addEmployeeUI
employeeInformation
checkEmployeeInformation
updateEmployeeList
successAddEmployee


RemoveEmployee
Manager
System
removeEmployee
retrieveEmployeeList
removeEmployeeUI
chooseRemoveEmployee
updateEmployeeList
successRemoveEmployee


ClockIn
Employee
System
clockInRequest
verifyClockInTime
storeClockInTime
updateActiveEmployees
successClockIn


ClockOut
Employee
System
clockOutRequest
checkMaxWorkTime
calculateWages
updatePayroll
updateActiveEmployees
successClockOut





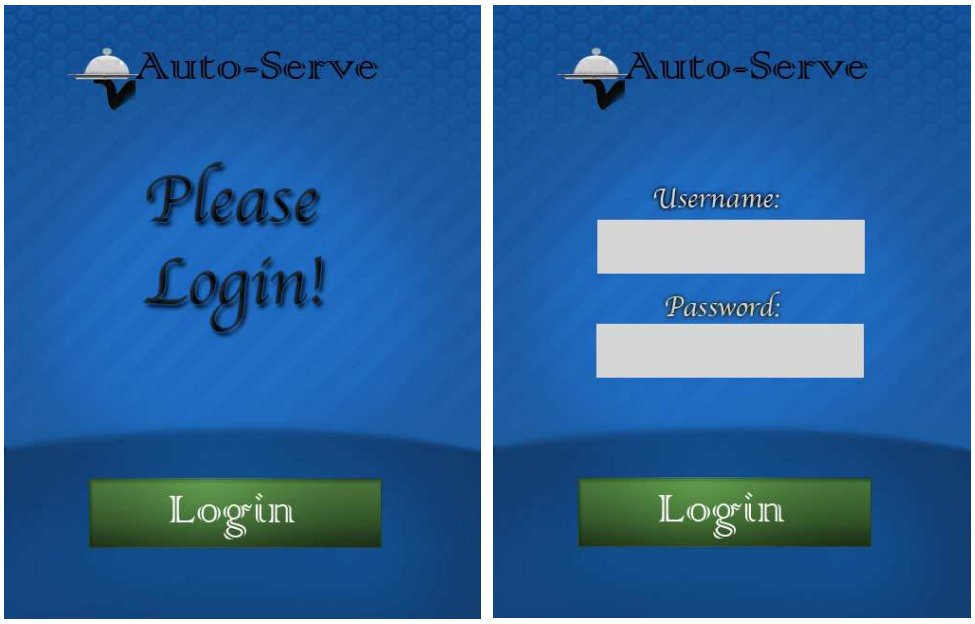
4. User Interface Specification

a. Preliminary Design

Our program is primarily design to be used on Android devices (phones, tablets, etc). This means that built in to the phone hardware is a home, back, menu, and search button. Our design of the user interface can also easily be ported to an online application, as well as a desktop terminal since it will be programmed in Java.

Our interface is specifically designed to be user friendly. Firstly, we tried to create an aesthetically pleasing environment. This should create a feeling that doing their job is more enjoyable. Also, we have implemented big buttons with clear icons and labels. This feature makes it easy for employees to use the touchscreen in a expedient manner. Our user interface is also designed to have an intuitive flow, which allows for an almost zero learning curve for a new hire.

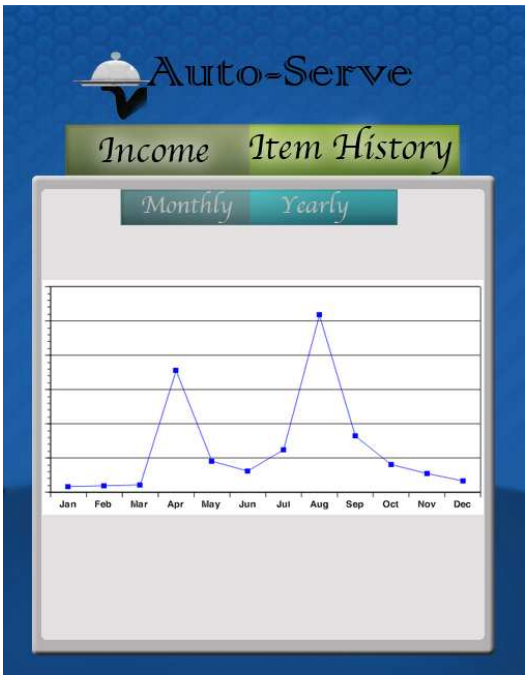
When one opens our system they are met with a welcome screen that prompts them to log in. Once they press the log-in button on the bottom they are met with a screen that prompts them to input their employee user-name and password.



This Log-in procedure not only allows the system to verify who is using the system, but also allows it to know what system feature privileges to give to the user. For examples, when a manager logs in the system, our system will know it is a manager and thus display the manager options screen.

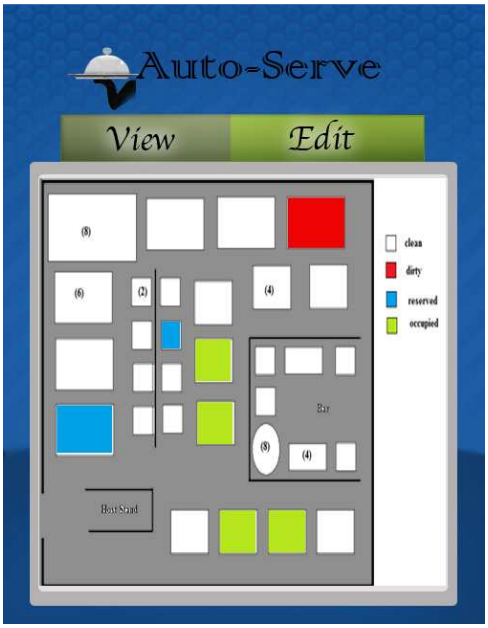
The “View Stock” button allows the manager to view the status of all the restaurants ingredients(low stock, high stock, etc).

The “Restaurant Stats” button shows the manager various restaurant statistics such as what items are selling the best, business reports, and more.



The “Timeclock” button lets the manager clock in or out of their shift.

The “Table Status” button lets the user view and edit a layout of the restaurant that shows which tables are open, clean, dirty, etc.

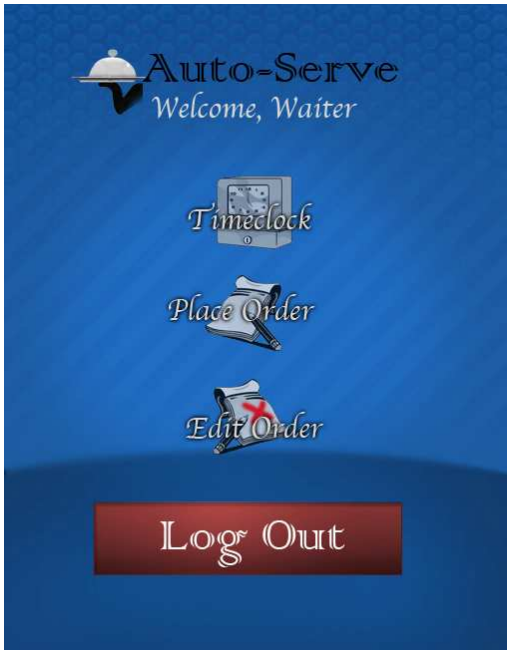


The “Payroll” button allows the manager to add and delete employees, as well as alter employee time cards in case of a mistake.

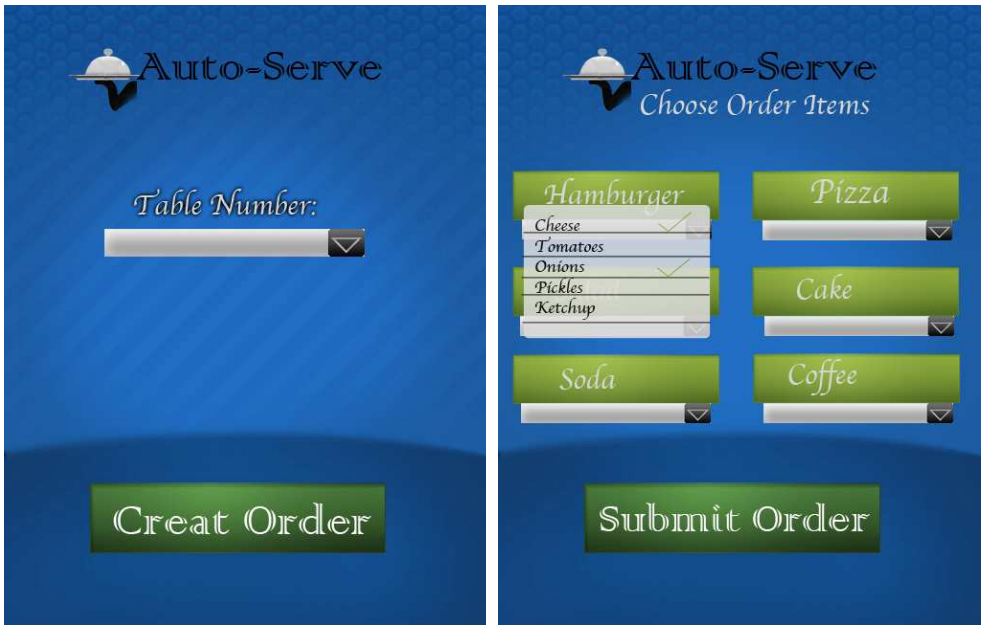
The “Edit Menu” button lets the manager make changes to the restaurant menu. Lastly, the “Logout” button logs the user of out of the system and exits.

Similarly, an options screen comes up after each other type of employee logs in; waiter, hostess, and busboy.

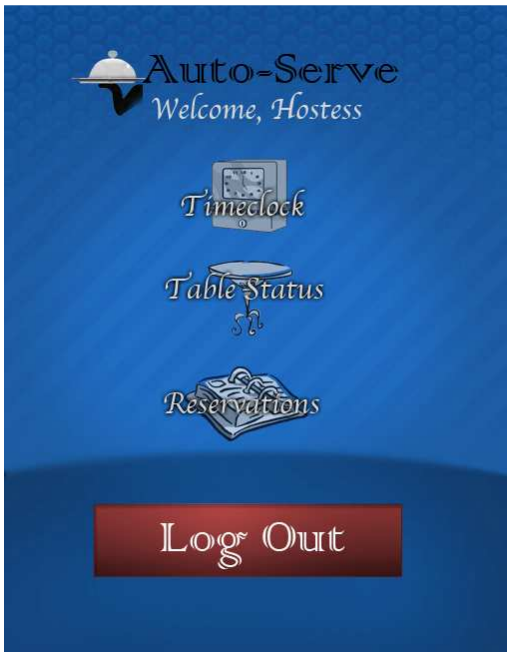
This is the screen that appears when a waiter logs in. You’ll notice that it only has options specific to a waiter’s duties.



The “Place Order” button allows the waiter to input an order to the kitchen. This is a multi-step process. First, the waiter must select from the system which table will be making a new order. Second, they will have options to add certain items to the order (burger, pizza, etc.) with a drop down menu under each to specify which topping, if any.

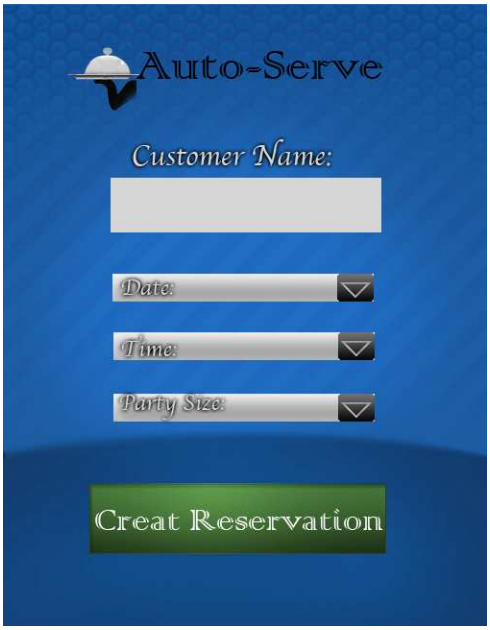


When a hostess logs in to the system they will be shown the hostess options screen:

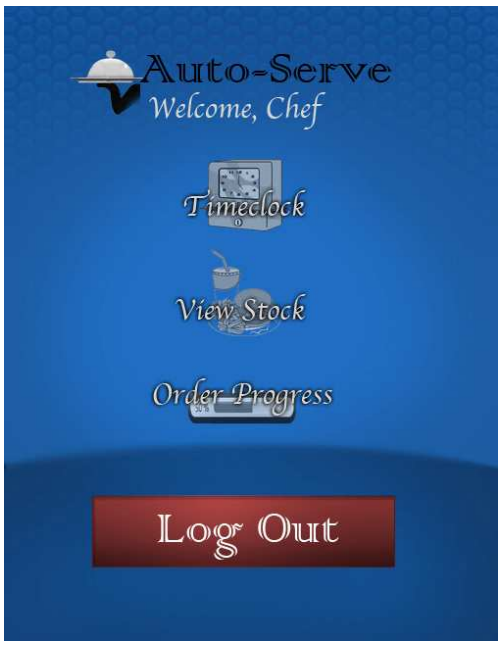


The “Timeclock” and “Table Status” features are the same as the other users, however the hostess is given the privilege of using the “Reservations” feature. When the hostess presses the “Reservations”

button they open a screen that displays a text box to input the customers name who is making a reservation and three drop down boxes to input the date, time, and party number of the reservation.

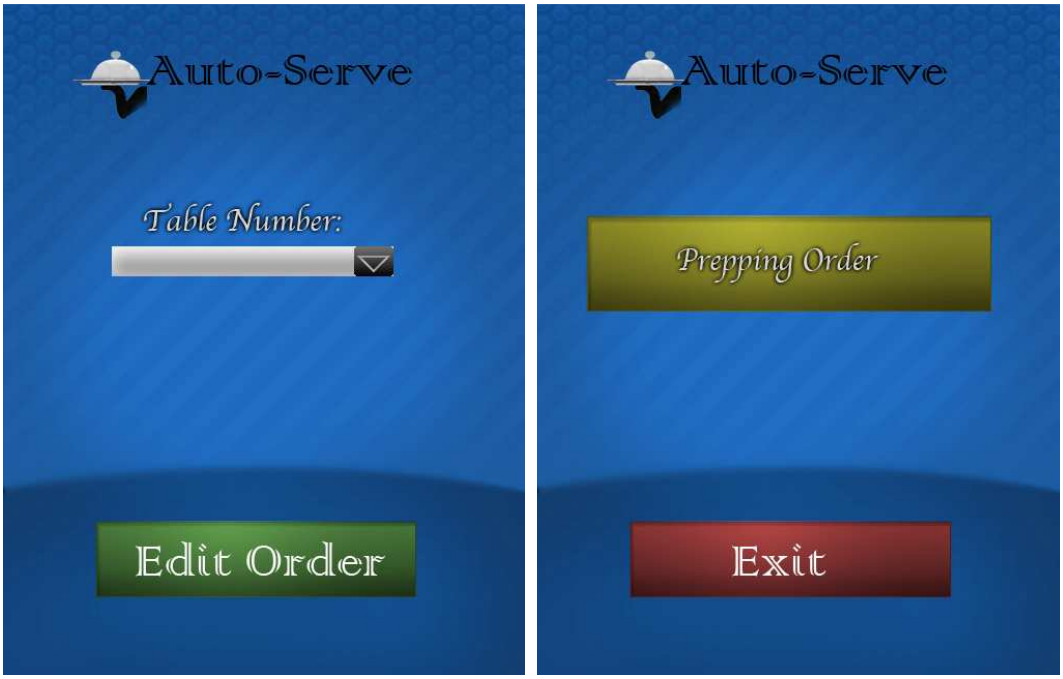


The Chef's options screen will be similar to the other employees:

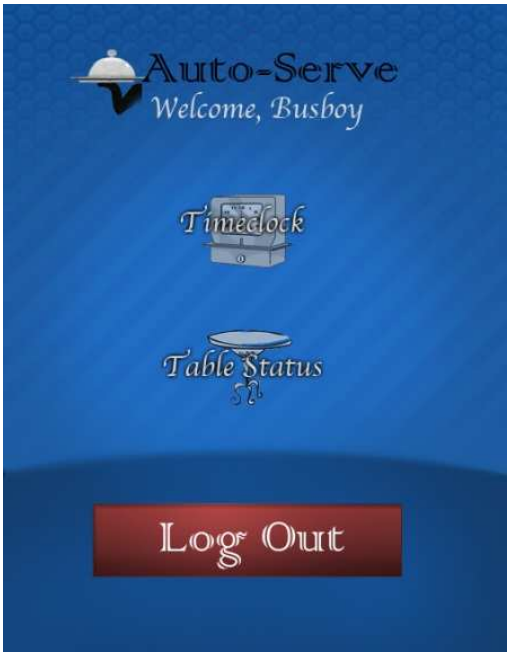


The chef has an option to edit the status of an order by pressing the "Order Progress" button. This is a two step process. The first screen prompts the chef to choose from a drop down box which table's order he wants to change the status of. The second screen displays one of three buttons "Prepping", "Cooking", or "Order Ready for Customer". After one button is pressed, it will go away and be replaced

by the next step of the order process, making this task easier and less prone errors in which button to press.



Lastly, we have the Busboy options screen. The busboy is only given the option to clok in and out, and to view and edit the status of a table (cleaned, dirty, etc).



To exit the program, the user would simply need to just press the “Log Out” button, and then the “Home” physical button on the device.

b. User Effort Estimation

The most used and typical usage scenario is the log in of the restaurant personnel. This requires input of information for authentication by the system. As a result, there is little to no navigational input requirements as seen below.

Logging In (Applies to all restaurant personnel)

1. Navigational: total 1 mouse click as follows [after completing data entry as shown below]
    a. Click “Login” button to log into the system
2. Data Entry: total 2 mouse clicks and a maximum of 50 keystrokes as follows
    a. Click the “Username” text field
    b. Enter a user name for the restaurant personnel
    c. Click the “Password” text field
    d. Enter a corresponding password to the entered user name

Customer

The customer user interface (UI) will include 3 prominant options in the main menu for ease-of-use and reduced complexity. This includes a “Menu” button for ordering food, a “Request Check” button to ask for a check and receipt once dining is finished, and a “Call Waiter” button for further assistance or beckoning a waiter to order for the table instead. A “Home” button will bring the customer back to the main menu after visiting 1 of the three menus.

As a result, this user interface is more demanding of clerical data entry for menu orders as shown below.

Placing an Order

1. Navigation: total 3 mouse clicks as follows
    a. Click “Menu” button [after completing data entry as shown below]
    b. Click “Submit Order” button to place order
2. Data Entry: total 3 mouse clicks as follows
    a. Click drop-down menu under order wanted
    b. Click option from drop-down menu
    c. Click “Add to Order” button

Chef

The user effort for the chefs’ user interface will be simplified for quick and efficient use in the kitchen setting. The buttons for progress are made large and made to appear in sequence with only one progress status update on the screen at a time. This allows for more concentration and time for cooking an order well than spending time updating the progress of cooking.

Given this layout, there is little navigational input needed compared to clerical data entry for updating the food completion status.

Updating Order Progress

1. Navigation: total 1 mouse click as follows
    a. Click “Order Progress” button [after completing data entry as shown below]
    b. Click “Exit” button to finish
2. Data Entry: total 5 mouse clicks as follows
    a. Click drop-down menu for table orders
    b. Click order to edit from drop-down menu
    c. Click “Prepping” button
    d. Click “Cooking” button
    e. Click “Order Ready” button

Busboy

The busboy mainly needs to check the table status. Therefore, the prominent functionality allows the busboy to know which tables need cleaning as efficiently as possible.

The navigational input required is minimized here, with more clerical data entry for changing table statuses.

Viewing Table Status / Cleaning Tables

1. Navigation: total 2 mouse clicks as follows
    a. Click “Table Status” button [after completing data entry as shown below --
    b. Click “Exit” button to return to main menu
2. Data Entry: total 3 mouse clicks as follows [after cleaning a dirty table]
    a. Click “Edit” tab
    b. Click on the table just cleaned
    c. Click “Clean” to change status to clean

Hostess

Aside from checking the table status to know which table to direct to customers to, the hostess will also be allowed to make reservations. As a result, most of the inputs will be within the clerical data entry.

A majority of the inputs for the hostess are for recording a reservation into the system, which is demands more clerical data entry than navigation.

Make a Reservation

1. Navigation: total 2 mouse clicks as follows

    a. Click “Reservations” button [after completing data entry as shown below]
    b. Click “Create Reservation” to make a reservation
2. Data Entry: total 7 mouse clicks and a maximum of 25 keystrokes as follows
    a. Click the “Customer Name” text field
    b. Enter customer’s name
    c. Click the “Date” drop-down menu
    d. Click the specified date of the reservation
    e. Click the “Time” drop-down menu
    f. Click the time of day the reservation was made
    g. Click the “Party Size” drop-down menu
    h. Click the quantity that applies

Manager

Since the manager overlooks most, if not all of the restaurant operations, he/she will have the greatest number of menu options. There is a greater navigational effort for the manager to view restaurant statistics and operations. Some functionalities will have more emphasis on data entry, including changes to menu options and employee/payroll statuses.

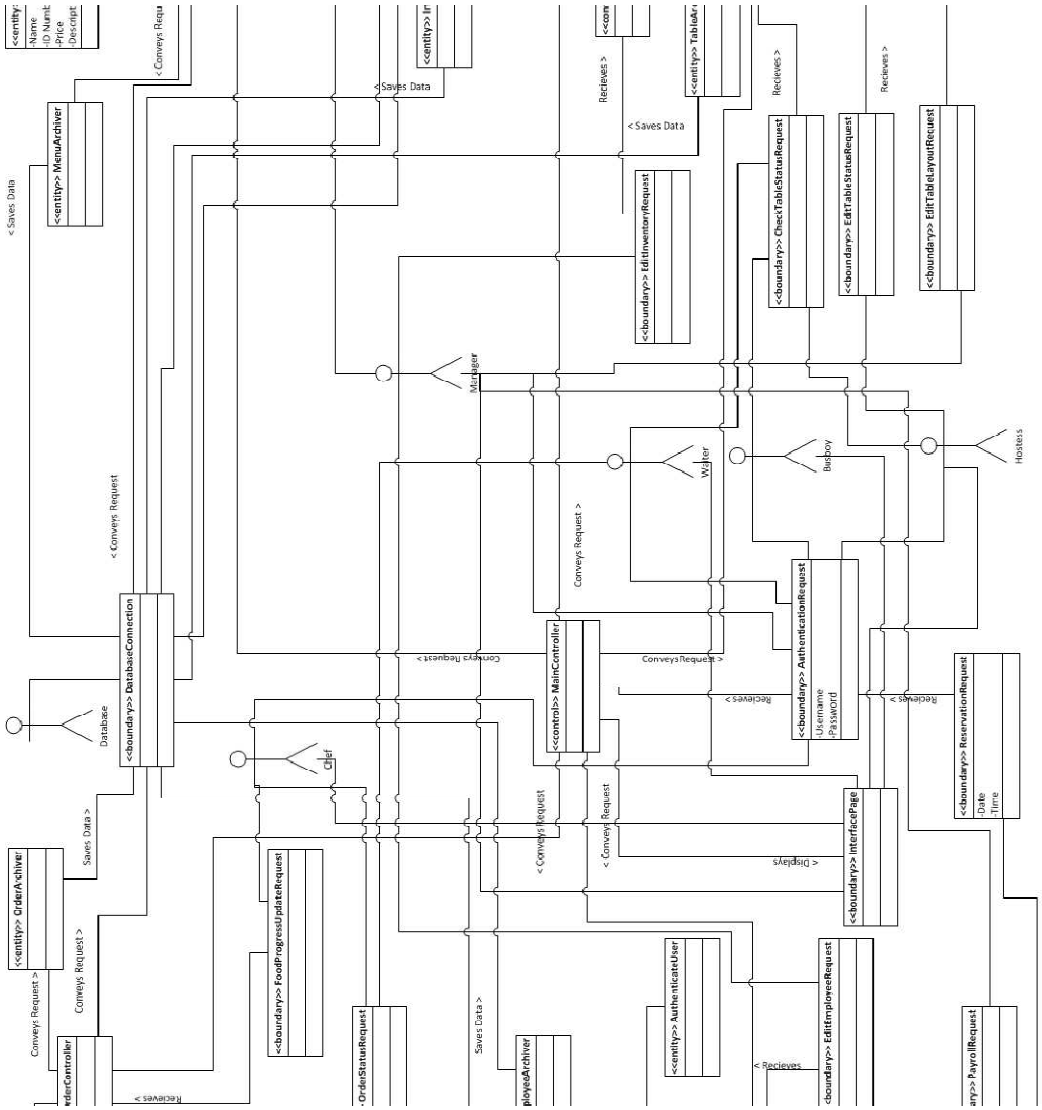
The following only requires navigational input, which has been minimized for ease-of-access.

<u>Viewing Popularity of Menu Options</u>

1. Navigation: total 3 mouse clicks as follows
    a. Click “Restaurant Stats” button
    b. Click “Item History” Tab
    c. Click “Exit” button to return to main menu
2. Data Entry: no data entry required

5. Domain Analysis

a. Domain Model



i. Concept Definitions

Responsibility Description	Type	Concept Name
Container holding FoodItems that customers order.	K	Order
Coordinate actions of the ordering sub-system and delegate the work to other concepts.	D	OrderController
Archive a request in the database in the appropriate table.	D	OrderArchiver
Form specifying the order a customer places that is placed in the database.	D	OrderRequest
Form specifying a status update of an order that is placed in the database.	D	FoodProgressUpdateRequest
Returns the current status of a specified order that is present in the database.	D	OrderStatusRequest
Archive a request in the database in the appropriate table.	D	EmployeeArchiver

Verify user credentials that were entered with those in the profile of a specific employee.	D	AuthenticateUser
Coordinate actions of the employee sub-system and delegate the work to other concepts.	D	EmployeeController
Form specifying changes to an employee profile that is updated in the database.	D	EditEmployeeRequest
Container holding contact and payroll information about employees.	K	EmployeeProfile
Container holding information about food that can be served.	K	FoodItem
Container holding a list of FoodItems.	K	Menu
Coordinate actions of the menu sub-system and delegate the work to other concepts.	D	MenuController
Archive a request in the database in the appropriate table.	D	MenuArchiver
Form specifying changes to the list of items and the items that is updated and saved in the database.	D	EditMenuRequest
Container holding information about food items in inventory.	D	InventoryItem
Archive a request in the database in the appropriate table.	D	InventoryArchiver
Coordinate actions of the inventory sub-system and delegate the work to other concepts.	D	InventoryController
Form specifying changes to inventory that are saved in the database.	D	EditInventoryRequest
Container holding information about the tables in the establishment.	K	Table
Archive a request in the database in the appropriate table.	D	TableArchiver
Coordinate actions of the table sub-system and delegate the work to other concepts.	D	TableController
Returns the current status of a specified table that is present in the database.	D	CheckTableStatusRequest
Form specifying changes to the status of a specified table that is currently in the database.	D	EditTableStatusRequest
Form specifying changes to the physical layout of the tables in the establishment.	D	EditTableLayoutRequest
Form specifying log in credentials that is later verified.	D	AuthenticationRequest
Form that shows actor current context, the actions that can be done, and the outcomes of previous actions.	K	InterfacePage
Coordinate actions of the entire system and delegate the work to other concepts.	D	MainController
Form specifying modification and calculation of payroll.	D	PayrollRequest
Form specifying a reservation for a certain date and time.	D	ReservationRequest

ii. Association Definitions

Concept Pair	Association Description	Association Name
OrderController <-> OrderArchiver	OrderController conveys request to OrderArchiver to send data to the database.	conveys request
OrderController <-> OrderRequest	OrderController receives order from	receives

	OrderRequest and sends order to OrderArchiver	
OrderController <-> OrderStatusRequest	OrderController receives request from OrderStatusRequest and returns status of order from databse.	receives
OrderController <-> FoodProgressUpdateRequest	OrderController receives request from FoodProgressUpdateRequest and sends the modified data to OrderArchiver.	receives
OrderController <-> DatabaseConnection	OrderController conveys request to DatabaseConnection and receives requested data.	conveys request
OrderController <-> MainController	MainController conveys request to OrderController, which delegates task to other concepts.	conveys request
EmployeeController <-> PayrollRequest	EmployeeController receives a request from PayrollRequest, which is then processed and sent to the database.	receives
EmployeeController <-> EditEmployeeRequest	EmployeeController receives a request from EditEmployeeRequest and sends data to EmployeeArchiver.	receives
EmployeeController <-> AuthenticateUser	EmployeeController conveys request to AuthenticationUser and receives validity of authentication.	conveys request
EmployeeController<-> EmployeeArchiver	EmployeeController conveys request to EmployeeArchiver to send data to the database.	conveys request
EmployeeController <-> DatabaseConnection	EmployeeController conveys request to DatabaseConnection and receives requested data.	conveys request
EmployeeController <-> MainController	MainController conveys request to EmployeeController, which delegates task to other concepts.	conveys request
TableController <-> EditTableLayoutRequest	TableController receives request from EditTableLayoutRequest and sends data to TableArchiver	receives
TableController <-> EditTableStatusRequest	TableController receives request from EditTableStatusRequest and sends data to TableArchiver	receives
TableController <-> CheckTableStatusRequest	TableController receives request from CheckTableStatusRequest and returns status.	receives
TableController <-> TableArchiver	TableController conveys request to TableArchiver to send data to the database.	conveys request
TableController <-> DatabaseConnection	TableController conveys request to DatabaseConnection and receives requested data.	conveys request

TableController <-> MainController	MainController conveys request to TableController, which delegates task to other concepts.	conveys request
InventoryController <-> EditInventoryRequest	InventoryController receives request from EditInventoryRequest and sends data to InventoryArchiver	receives
InventoryController <-> InventoryArchiver	InventoryController conveys request to InventoryArchiver to send data to the database.	conveys request
InventoryController <-> DatabaseConnection	InventoryController conveys request to DatabaseConnection and receives requested data.	conveys request
InventoryController <-> MainController	MainController conveys request to InventoryController, which delegates task to other concepts.	conveys request
MenuController <-> EditMenuRequest	MenuController receives request from EditMenuRequest and sends data to MenuArchiver.	receives
MenuController <-> MenuArchiver	MenuController conveys request to MenuArchiver to send data to the database.	conveys request
MenuController <-> DatabaseConnection	MenuController conveys request to DatabaseConnection and receives requested data.	conveys request
MenuConroller <-> MainController	MainController conveys request to MenuController, which delegates task to other concepts.	conveys request
MainController <-> InterfacePage	MainController displays InterfacePage based on actor's position.	display
MainController <-> ReservationRequest	MainController receives request from ReservationRequest and returns feedback of reservation.	receives

iii. Attribute Definitions

Concept	**Attributes**	**Attribute Description**
EmployeeProfile	Name	Used to determine the name of the employee for convenient search and reference.
	ID	Holds the identification number of the employee to specify a unique employee.
	Position	Used to determine the employee's position in the infrastructure of the establishment and determine the interface the user is authorized to view.

	Address	Used in order to determine the address at which information will be sent to the user.
	Wage	Used to determine the payment that will be made during payroll calculation.
	Username	Unique phrase used by the user as the log in name and half of the authentication protocol.
	Password	Unique phrase used in conjunction with Username to validate user into system.
	HoursWorked	Holds the total time the user is clocked in.
FoodItems	Name	Used to determine the name of an item for convenient reference.
	ID Number	Holds the identification number for an item in order to distinguish unique items.
	Price	Used to determine the cost of an order including an item.
	Description	Holds the detailed information about a menu option.
Order	FoodItem	Holds the menu option chosen to be prepared.
InventoryItem	Name	Used to determine the name of an item in stock within the inventory.
	ID Number	Holds the identification number to determine specific items and organizing.
	Quantity	Holds the amount of an item remaining.
	Description	Holds detailed information on an item.
AuthenticationRequest	Username	Unique phrase stored in the system to match with corresponding entered log in information.
	Password	Unique phrase corresponding to a Username stored for matching with entered keystrokes.
ReservationRequest	Date	Holds the month, day, and year.
	Time	Holds the time of day.

Table	ID Number	Holds the identification number to distinguish between different tables.
	Location	Holds information of the physical placement of tables in the restaurant.

iv. Traceability Matrix

Use Case	PW	Order	OrderController	OrderArchiver	OrderRequest	FoodProgressUpdateRequest	OderStatusRequest	EmployeeArchiver	AuthenticateUser	EmployeeController	EditEmployeeRequest	EmployeeProfile	FoodItem	Menu	MenuController	MenuArchiver	EditMenuRequest	InventoryItem	InventoryArchiver	InventoryController	EditInventoryRequest	Table	TableArchiver	TableController	CheckTableStatusRequest	EditTableStatusRequest	EditTableLayoutRequest	AuthenticationRequest	MainController	PayrollRequest	InterfacePage	ReservationRequest
UC1																																X
UC2																									X							
UC3																						X	X	X			X					
UC4		X	X	X	X									X																		
UC5		X	X	X	X									X																		
UC6		X	X	X	X																											
UC7																																
UC8						X	X																									
UC9						X	X																									
UC10		X	X	X	X									X																		
UC11																						X	X	X		X						
UC12																						X	X	X		X						
UC13																						X	X	X		X						
UC14																		X	X	X	X											
UC15																		X	X	X	X											
UC16																		X	X	X												
UC17																		X	X	X	X								X			
UC18													X	X	X	X	X															
UC19													X	X	X	X	X															
UC20																													X			
UC21		X	X	X																												
UC22								X		X	X	X																				
UC23								X		X	X																					
UC24										X	X	X																				
UC25																													X			
UC26																													X			
UC27																														X		
UC28																									X					X		
UC29																									X					X		
UC30																									X							
UC31									X																			X				

b. System Operation Contracts

Name:	**CheckTableStatus**
Responsibilities:	Determines which tables are clean, dirty, occupied, or reserved
Cross References:	Use Cases: CheckTableStatus
Exceptions:	None
Preconditions:	There are tables.
Postconditions:	The current conditions of the tables were updated and displayed.
Name:	PlaceOrder
Responsibilities:	Records customers' orders.
Cross References:	Use Cases: PlaceOrder
Exceptions:	None
Preconditions:	There are menu items to order.
Postconditions:	Orders were recorded by the system.

Name:	**AuthenticateUser**
Responsibilities:	Checks the user logging into the system is an employee.
Cross References:	Use Cases: AuthenticateUser
Exceptions:	None
Preconditions:	There are users in the system database.
Postconditions:	**User was identified as either an employee or not.**

Name:	StockStats
Responsibilities:	Returns the quantities of items in the inventory.
Cross References:	Use Cases: StockStatcs
Exceptions:	None
Preconditions:	There is a list of inventory items.
Postconditions:	The amounts of each inventory item was displayed.

Name:	**ViewRestaurantStats**
Responsibilities:	Displays the popularity of menu items along with menu and restaurant statistics.
Cross References:	Use Cases: ViewRestaurantStats
Exceptions:	None
Preconditions:	There are menu items to be ordered.
Postconditions:	The frequency of each menu item ordered was displayed.

Name:	**EditMenuItems**
Responsibilities:	Adds and/or deletes menu options.
Cross References:	Use Cases: EditMenuItems
Exceptions:	None
Preconditions:	There are menu items already in the system.
Postconditions:	The items to be added to the menu and deleted from the menu were added and deleted.

Name:	**ClockIn**
Responsibilities:	Records the time an employee begins working.
Cross References:	Use Cases: ClockIn

Exceptions:	None
Preconditions:	An employee is authenticated and logged in.
Postconditions:	The time the employee clocked in was recorded in the system.

Name:	**ClockOut**
Responsibilities:	Records the time an employee ends a work shift.
Cross References:	Use Cases: ClockOut
Exceptions:	None
Preconditions:	An employee is clocked in.
Postconditions:	The employee's working hours were adjusted to with the recorded clock out time, with wages adjusted.

Name:	**AddEmployee**
Responsibilities:	Creates a new employee profile in the system.
Cross References:	Use Cases: AddEmployee
Exceptions:	None
Preconditions:	The employee is not already in the system and the manager is logged in.
Postconditions:	A new profile was added to the employee roster.

Name:	**RemoveEmployee**
Responsibilities:	Deletes an existing employee profile from the system.
Cross References:	Use Cases: RemoveEmployee
Exceptions:	None
Preconditions:	The employee's profile to be deleted is already in the system.
Postconditions:	The employee's profile was removed from the list of employees in the system

Name:	**UpdateEmployee**
Responsibilities:	Updates the information for the employee profiles in the system.
Cross References:	Use Cases: UpdateEmployee
Exceptions:	None
Preconditions:	There are employee profiles in the system.
Postconditions:	Old employee information was replaced by newer information.

Name:	**ReleasePayment**
Responsibilities:	Pays employees the amount earned and updates their payroll
Cross References:	Use Cases: ReleasPayment
Exceptions:	None
Preconditions:	There are employees with wages and payroll information.
Postconditions:	The payroll was reset after a payment to the respective employees and statistics recorded.

c. Mathematical Model

The application requires some very basic algorithms in order to calculate various restaurant statistics. Those are as follows.

Payroll

The system will calculate payroll statistics. The system will access the hours the employee worked, multiply the number of hours by their hourly wage, and then record how much the employee was paid and reset the employee's hours to zero.

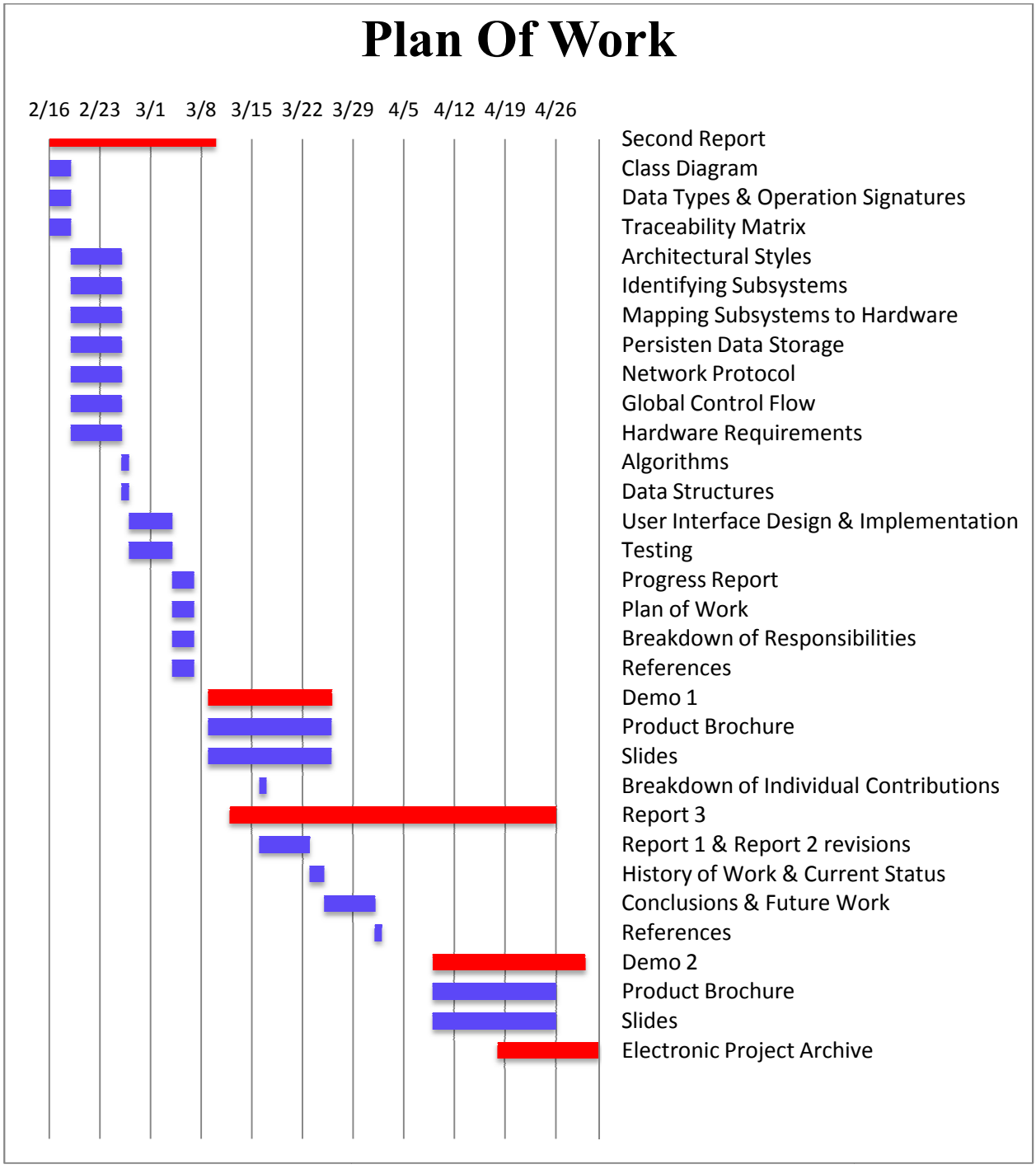
Check Statistics

The manager will be allowed to check the menu trend and most popular hours that that customers visit the restaurant. The system will return the most popular food items by returning their sales per week. Also, the system will return the average customers per hour on any given day.

Check Stock

The system will simply access the current amount of stock and return the values. It will also give recommended amounts to order based on the trends in menu items and the ingredients they are composed of.

6. Plan of Work



For our group's plan of work, we have created a Gannt Chart where we have listed the Tasks we have to do as well as the date when to begin a particular task as well as when the task should be completed by. The bars in Red indicate one item while the bars in purple indicate a sub item. For example, the Second Report, Demo 1 and 2, and Report 3 would be considered items while a sub item would be Network

Protocol. Below we have also included the table so as to be more specific if the Gannt Chart is not clear to some.

Second Report	**Start Date**	**Days till Due Date**	**End Date**
Class Diagram	2/17/12	23	3/9/12
Data Types & Operation Signatures	2/17/12	3	2/20/12
Traceability Matrix	2/17/12	3	2/20/12
Architectural Styles	2/17/12	3	2/20/12
Identifying Subsystems	2/20/12	7	2/27/12
Mapping Subsystems to Hardware	2/20/12	7	2/27/12
Persisten Data Storage	2/20/12	7	2/27/12
Network Protocol	2/20/12	7	2/27/12
Global Control Flow	2/20/12	7	2/27/12
Hardware Requirements	2/20/12	7	2/27/12
Algorithms	2/20/12	7	2/27/12
Data Structures	2/27/12	1	2/28/12
User Interface Design & Implementation	2/27/12	1	2/28/12
Testing	2/28/12	6	4/5/12
Progress Report	2/28/12	6	4/5/12
Plan of Work	3/5/12	3	3/8/12
Breakdown of Responsibilities	3/5/12	3	3/8/12
References	3/5/12	3	3/8/12
Demo 1	3/5/12	3	3/8/12
Product Brochure	3/10/12	17	3/27/12
Slides	3/10/12	17	3/27/12
Breakdown of Individual Contributions	3/10/12	17	3/27/12
Report 3	3/17/12	1	3/18/12
Report 1 & Report 2 revisions	3/13/12	45	4/27/12
History of Work & Current Status	3/17/12	7	3/24/12
Conclusions & Future Work	3/24/12	2	3/26/12
References	3/26/12	7	4/2/12
Demo 2	4/2/12	1	4/3/12
Product Brochure	4/10/12	21	5/1/12
Slides	4/10/12	17	4/27/12
Electronic Project Archive	4/10/12	17	4/27/12
	4/19/12	14	5/3/12

7. References

1. Marsic, Ivan; *Software Engineering, http://www.ece.rutgers.edu/~marsic/books/SE/book-SE_marsic.pdf*
2. Project #3 Examples; http://www.ece.rutgers.edu/~marsic/books/SE/projects/Restaurant/2007-g13-report3.pdf
3. FURPS article; http://en.wikipedia.org/wiki/FURPS