QUICKBYTES

BY MMPK

OCTOBER 13, 2020

MEMBERS:
Patrick Carra, Mark Norfolk, Kegan Ronholt, Clint Ryan, Mila Hose

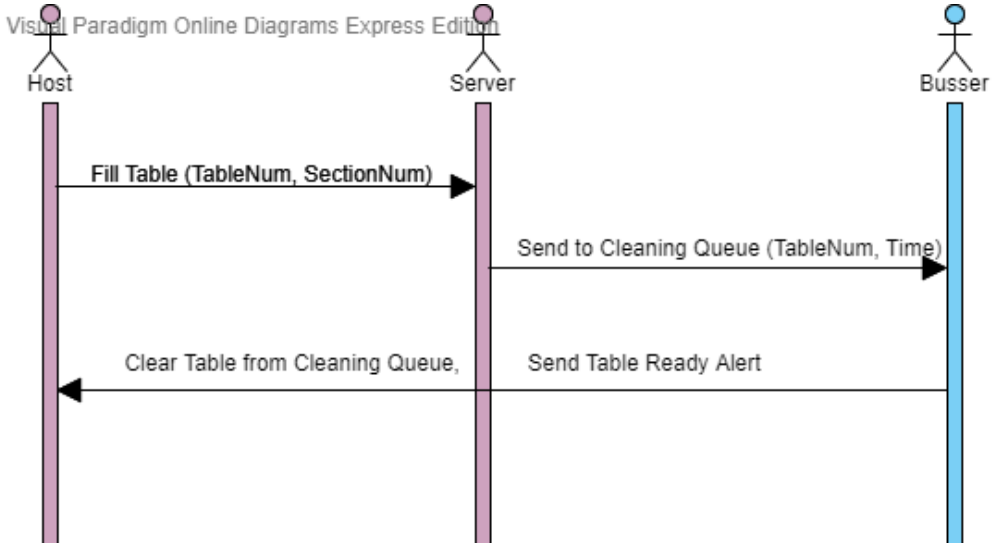
	Requirements Specification	Software Design	Coding	Debugging	Report Preparation	Other:
Patrick Carra	20%	20%	20%	20%	20%	20%
Mark Norfolk	20%	20%	20%	20%	20%	20%
Kegan Ronholt	20%	20%	20%	20%	20%	20%
Clint Ryan	20%	20%	20%	20%	20%	20%
Mila Hose	20%	20%	20%	20%	20%	20%

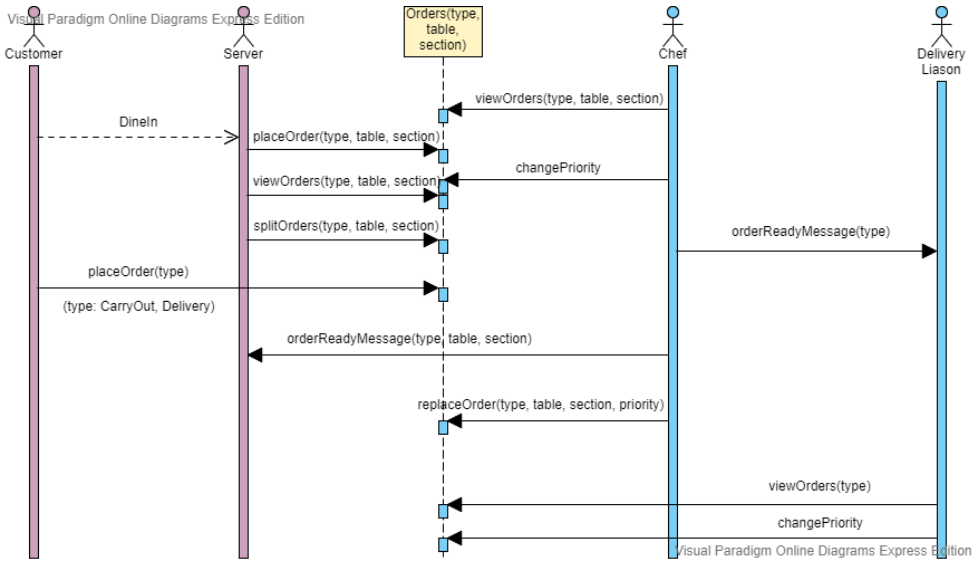
Table of Contents

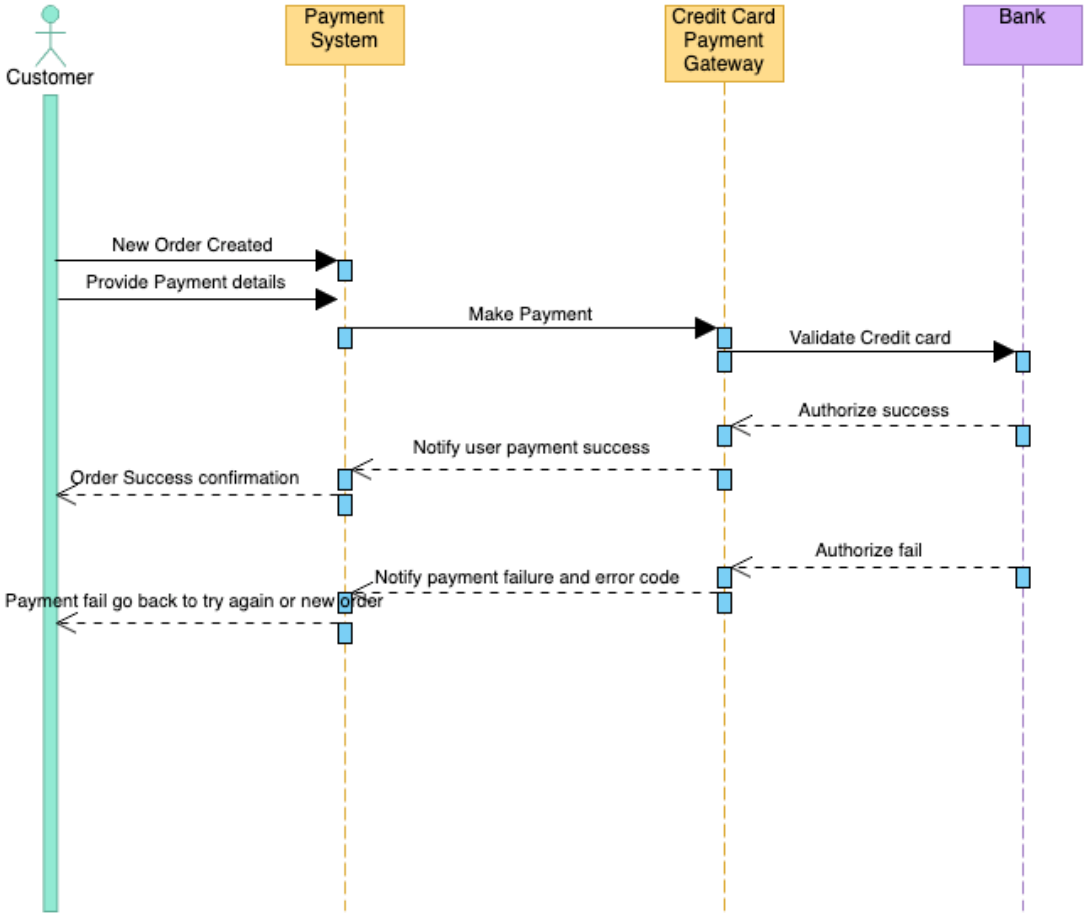
Report #2

Section 1:

Section 1.1: Interaction Diagrams





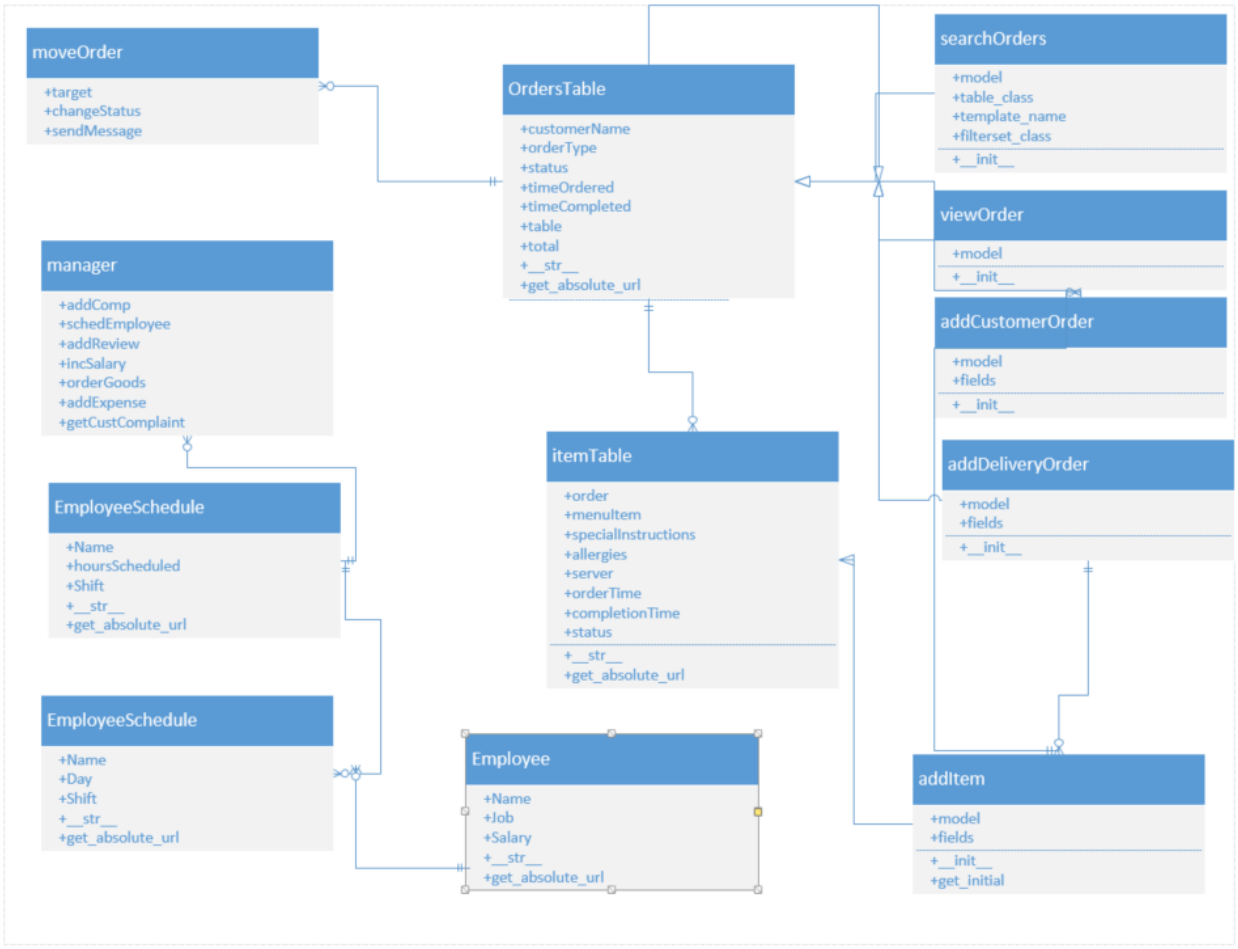


Section 1.2: Project Management

Section 2:

Section 2.1: Class Diagram and Interface Specification

Section 2.1.1: Class Diagrams



Section 2.1.2: Data Types and Operation Signatures

Class Orderstable:

1. Attributes:

- customerName - Name of customer
- orderType - Type may be delivery, dine-in, or carry-out
- status - The order status waiting, preparing, etc
- itemsOrdered - A list of items ordered
- timeOrdered - The time at which the order was started
- timeCompleted - Completion time of order
- table - The table number if the customer is dining in

- total - The total cost of an order

2. Methods:

- __str__ - string representation of the object
- Get_absolute_url - provides a redirection upon successful completion of an edit function

Class itemTable:

1. Attributes:

- Order - the associated order number for this item
- menuItem - the associated menuItem for this item
- specialInstructions - user supplied instructions for prepping this item
- Allergies - user supplied allergies if they exist
- Server - associated server if one exists
- orderTime - time the item was added to an order
- completionTime - time the item was completed in the kitchen
- Status - the status of an item waiting, preparing, completed, delivered

2. Methods:

- __str__ - the string representation of the object

Class addCustomerOrder:

1. Attributes:

- Model - the class item from which this class inherits its attributes
- Fields - the pertinent fields to include for this order

2. Methods:

- __init__ - constructor for the class

Class addDeliveryOrder:

3. Attributes:

- Model - the class item from which this class inherits its attributes
- Fields - the pertinent fields to include for this order

4. Methods:

- __init__ - constructor for the class

Class viewOrder:

5. Attributes:
   - Model - the class item from which this class inherits its attributes
6. Methods:
   - __init__ - constructor for the class

Class searchOrders:

7. Attributes:
   - Model - the class item from which this class inherits its attributes
   - Table_class - the associated table for this set of orders
   - Template_name - the html template to include with this view
   - Filterset_class - the filters to include with the searchOrder class
8. Methods:
   - __init__ - constructor for the class

Class addItem:

9. Attributes:
   - Model - the class item from which this class inherits its attributes
   - Fields - the pertinent fields to include for this order
10. Methods:
   - __init__ - constructor for the class
   - Get_initial - provides initial values for order to provide association

Class moveOrder:

11. Attributes:
   - Target - what status this order will have
12. Methods:
   - changeStatus - changes status of order
   - sendMessage - sends a message to the appropriate person in the restaurant if necessary

Class moveOrder:

13. Methods:

- addComp - will discount some or all of an order
- schedEmployee - will schedule an employee for a shift
- addReview - provide the employee with the periodic review
- incSalary - will allow a manager to give an employee a raise
- orderGoods - will allow the manager to order more food or beverage items
- addExpense - will allow the manager to add expenses for tracking
- getCustomerComplaint - allows a manager to receive customer complaints

Class employeeSchedule:

14. Attributes:
    - Name - employee name
    - hoursScheduled - total number of hours scheduled for the week
    - Shift - shifts that an employee is scheduled for
15. Methods:
    - __str__ - string representation of employee schedule
    - Get_absolute_url - allows for redirection upon successful completion of editing a schedule

Class Employee:

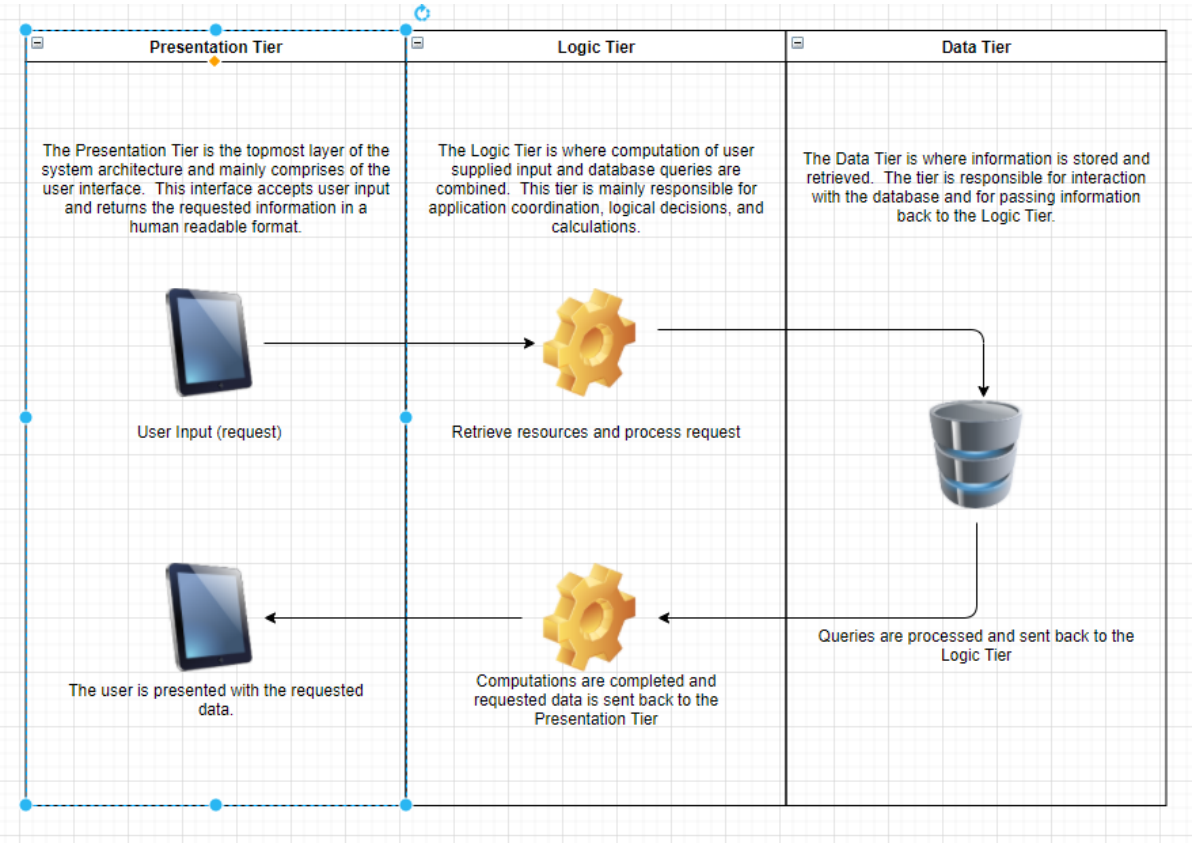
16. Attributes:
    - Name - employee name
    - Job - the associated job title
    - Salary - the salary of the employee
17. Methods:
    - __str__ - string representation of employee schedule
    - Get_absolute_url - allows for redirection upon successful completion of editing a schedule

Section 2.1.3: Traceability Matrix:

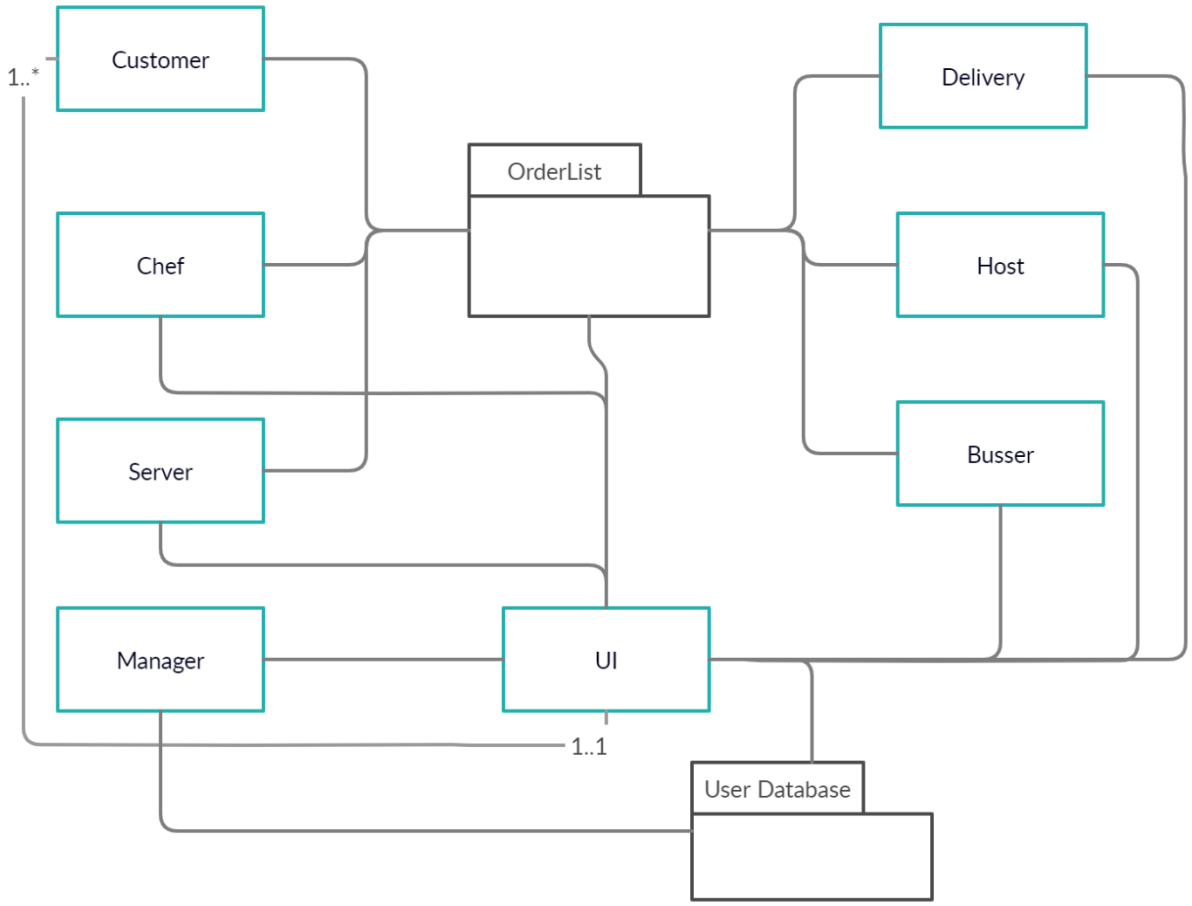
	Domain Concepts		
Software Classes	Employee Interface	Customer Interface	Manager Interface
ordersTable	x		x
itemsTable	x		x
addCustomerOrder	x	x	x
addDeliveryOrder	x	x	x
addItem	x	x	x
viewOrders	x	x	x
searchOrders	x		x
moveOrders	x		x
manager			x
employeeSchedule			x
employee			x

Section 2.2: System Architecture and System Design

Section 2.2.1: Architectural Styles



Section 2.2.2: Identifying Subsystems



Section 2.2.3: Mapping Subsystems to Hardware

The system will be active on multiple devices. The client is able to access our product via web browser and once connected, they access our server that processes information on their behalf. The server subsystem should be able to maintain the information that is provided from external devices such the computers, phones, and point-of-sale (POS) systems. It is responsible for making sure the information collected is valid and is properly stored into the database.

Section 2.2.4: Persistent Data Storage

Data that needs to outlive a single execution of the system includes the following:

- Usernames and Passwords
- Orders and Order History
- Comments and Complaints
- Business Income and Expenditure
- Customer Payment information (For easier payment access)

- Discounts and rewards
- Notifications
- Meal Ratings
- Table Reservations
- Customer Order time

Section 2.2.5: Network Protocol

Our Program is run over the web, so it utilizes HTTP as its network protocol. All the users are connected through a single website with each user having access to a different interface that each can signal to other users of different kinds. Such as a server who can place and track orders on their interface while sending a message to the busser when a table needs cleaned or sends an order to the chef.

Section 2.2.6: Global Control Flow

- Execution orderliness:
- Time Dependency
- Concurrency

Section 2.2.7: Hardware Requirements

Since the program is run remotely, the users must simply have stable access to the internet as well as a device capable of opening and running a website.

Section 3:

Section 3.1: Algorithms and Data Structures

Section 3.1.1: Algorithms

No algorithms used in our project.

Section 3.1.2: Data Structures

We are using a series of queues for order placement, because it allows for orders to be placed in a first in first out sequence. It also allows us to change the order of items in the queue if necessary, for remake orders, deliveries, or special circumstances.

Section 3.2: User Interface Design and Implementation

To make things easier for all users, we combined the sign-in and register page for all users. So, every customer, employee, and manager will make an account through the same place though they may need different information depending on which job they are fulfilling. When they sign in, they will be directed to their job's interface. The employee interfaces will be relatively similar but will be slightly specialized to their individual job. The customer interface will be simple and easy to navigate with menu that allows that to easily switch through the menus, call for service, or view their account.

Section 3.3: Design of Tests

Test Case Identifier: TC-1 **Use Case Tested: UC-1** **Pass/Fail Criteria: Test passes if user can initiate and complete an order** **Input Data: customerName, orderType, status, table, menuItem, specialInstructions, allergies, server**	
Test Procedure	**Expected Result**
Step 1:User initiates a new order filling in customerName and orderType.	**Upon successful completion the user is redirected to their main order page.**
Step 2: User selects addItem	**The user is then taken to the addItem page.**
Step 3: User fills in menuItem, specialInstructions, allergies, server, and status	**The user upon successful completion will be redirected to the main order page.**

Test Case Identifier: TC-2 **Use Case Tested: UC-2** **Pass/Fail Criteria: Test passes if the user can view active orders and chef can change order status. Test fails if query does not return table status or changes cannot be made to order status** **Input Data: customerName or table**	
Test Procedure	**Expected Result**
Step 1: User enters the customerName or table. Chef enters change to order status	**Upon successful query the customer shown the status of their order.**
Step 2: Order status is displayed	**Upon success, the order status changed by chef is updated in the database**

Test Case Identifier: TC-3 **Use Case Tested: UC-3** **Pass/Fail Criteria: Test passes if customers can view tables, hosts/hostesses can view and assign tables, servers and busboys can change status of table.** **Input Data: table**	
Test Procedure	**Expected Result**
Step 1: Host views available tables and assigns table to customer **Step 2: Server marks guests have left and table needs to be cleaned** **Step 3: Busboy cleans and marks that table is cleaned**	**Upon success, available tables should be able to be queried by customers and employees.** **Upon success, Hosts/Hostesses and servers can assign tables to customers** **Upon success, servers and busboys can change the status of tables from dirty to clean or vice verse.**

Test Case Identifier: TC-4 **Use Case Tested: UC-4** **Pass/Fail Criteria: Test passes if payment is processed by system for customer or server. Test fails if payment is not processed by bank** **Input Data: price, customerName, phone, email, cardNumber, expirationDate, securityCode, zip**	
Test Procedure	**Expected Result**
Step 1: Bill amount is entered into system **Step 2: financial information of guest is entered or card is swiped.** **Step 3: Verification is returned from bank that transaction was successful**	**Upon Success, system will receive confirmation from bank that transaction was successfully processed.**

Test Case Identifier: TC-5

Use Case Tested: UC-5

Pass/Fail Criteria: Test passes if employee or customer can successfully log in. Fails if employee or customer is unable to log in to the system.

Input Data: email or username, password

Test Procedure	Expected Result
Step 1: User enters username and password **Step 2: User submits information**	**If successful, the user is directed to the appropriate landing page.** **If not successful, the user is asked to try again and given option to reset password.**

Test Case Identifier: TC-6

Use Case Tested: UC-6

Pass/Fail Criteria: Low contact mode is able to be engaged

Input Data: Simple true or false value from user

Test Procedure	Expected Result
Step 1: User selects low contact mode **Step 2: System alerts the restaurant, and guest orders and pays exclusively through application**	**If successfully, instructions for low contact mode will appear to customer, and they will be walked through the ordering and payment processes.**

Test Case Identifier: TC-7

Use Case Tested: UC-7

Pass/Fail Criteria: Passes if the user can put in request to the service queue.

Input Data: table, needHelp(true)	
Test Procedure	**Expected Result**
Step 1: customer selects that they desire service, and enters table number **Step 2: Their table is placed in service queue for server to see** **Step 3: After receiving assistance, the server takes the table out of the queue.**	**If successful the customer can select the request service button and their table will appear in the service queue.** **Server will be able to mark that table was served and they will be popped out of queue**

Test Case Identifier: TC-8 **Use Case Tested: UC-8** **Pass/Fail Criteria: Passes if user can see the waiting list, view the position of a customer in line and estimate the time until they are seated** **Input Data: Customer Name, present time**	
Test Procedure	**Expected Result**
Step 1: User enters the customer's name and current time **Step 2: User verifies the customers place in line and estimated wait time.**	**If successful the user will see the customer placed on the wait list.** **The user will be able to see the time the customer was placed on the wait list and estimate the time until they are seated**

Test Case Identifier: TC-9 **Use Case Tested: UC-9** **Pass/Fail Criteria: Manager may schedule an employee for a shift and view the schedule** **Input Data: Employee, Schedule date and time**

Test Procedure	Expected Result
Step 1: Manager will select an employee	**The app will accept the user entered.**
Step 2: Manager will set a time and date for their shift	**The app will set the time and date for the scheduled shift and redirect to the schedule.**
Step 3: Manager will confirm that the employe has been scheduled	**The newly entered shift will be reflected on the schedule.**

Test Case Identifier: TC-10 **Use Case Tested: UC-10** **Pass/Fail Criteria: Users will be able to track and order ingredients for the restaurant** **Input Data: Ingredient, quantity**	
Test Procedure	**Expected Result**
Step 1: User will select the ingredient needed.	**The app will accept the ingredient selected.**
Step 2: User will select the quantity needed.	**The app will accept the quantity needed.**
Step 3: User will verify the amount and type of ingredients ordered	**The user will be able to verify that the correct amount and type of ingredients have been ordered.**

Test Case Identifier: TC-11 **Use Case Tested: UC-11**

Pass/Fail Criteria: Host/Hostess or customer is successfully able to place table reservation **Input Data: guestName, guestPhone, guestEmail, table, date, partyCount**	
Test Procedure	**Expected Result**
Step 1: User supplies customer information **Step 2: User enters table choice and date** **Step 3: User received confirmation or denial message**	**The app will accept the user information and add the table reservation to the DB**

Test Case Identifier: TC-12 **Use Case Tested: UC-12** **Pass/Fail Criteria: Successful if employee can clock in and out of system** **Input Data: employee id, password, startTime, endTime**	
Test Procedure	**Expected Result**
Step 1: employee clocks in to work using id and password **Step 2: employee clocks out of work using id and password**	**Employee clocks in using id and password, employee clocks out using id and password. System tracks how many hours employee worked**

Test Case Identifier: TC-13 **Use Case Tested: UC-13** **Pass/Fail Criteria: successful if new account is created successfully. Fail if account is unable to be created** **Input Data: username, password, repeated password, email, repeated email**

Test Procedure	Expected Result
Step 1: User enters chosen username, email, and password **Step 2: User is requested to re-enter email and password** **Step 3: If the email and password match, the account is created and added to DB**	**User is able to enter required information, and their account is added to the DB.**

Section 3.4: Project Management and Plan of Work

Section 3.4.1: Merging Contributions from Individual Team Members

We originally split our work up between which actor in our program that we were going to be responsible for. Each actor being the Manager, Chef, Host, Server, Busser, Delivery Liaison, and Customer. For most of the project report, the work was split up to fill in information for each aspect of the report for each one of the actors we were responsible for.

This way of working changed a bit when we started coding. After Patrick set up the database. The coding has mostly been a group effort, with several people working to get a single page or part of the system working. There were a couple connection issues to the database for a few of us, but after enough google searches the problem was fixed.

Section 3.4.2: Project Coordination and Progress Report

The main website, the login, and registration pages are hosted. We're working on functionality of the login/registration pages. Use cases will be implemented as soon as the login functionality is available. Several of the employee pages have been created but are still in the process of debugging. For the rest of the employee pages, we have basic templates of what they are going to be but don't yet have them implemented.

Section 3.4.3: Plan of Work

- **Chef Pages Online - 10/7/2020**
- **Delivery Liaison Pages Online - 10/12/2020**
- **Busboy Pages Online - 10/16/2020**
- **Host Page Online - 10/23/2020**
- **Begin Functionality Coding - 10/24/2020**

Section 3.4.4: Breakdown of Responsibilities

Patrick: Manager, Login/Register

Mark: Delivery Liaison, Log-in/Register

Keagan: Host, Chef

Mila: Server, Busser

Clint: Customer