Restaurant Automation System

Leader: Khaled Ashraf Elgreitly / 20100237
Member: Mostafa Walid Zayed / 20100391
Member: Ahmed Abdelazeem Emam / 20100359
Member: Abdelrahman Abdelmasod Orz / 20100241

Table of Contents

Problem Describtion

Many restaurants are still operated using pen and paper methods, with little or no automation. This project develops a computerized system to help restaurant personnel coordinate ,
This idea will help restaurant owners handle online food orders, table booking, inventory control, generating bills, managing menus, and various customer services.
their activities and improve their services, and for the management to track business growth and create future plans.

The goal for this project is to introduce automation in privately-owned restaurants, This system will be an automated food ordering system. We are working to facilitate the process for the customer, make everything automatic, and facilitate the payment process as well.

This project computerizes restaurant operation so that all information pertaining to patron's
Orders and staff activity will be conveniently shared and stored over the restaurant's intranet.

Hosts will be able to view table status with a click of a button. The wait staff will be able to enter the patron's orders quickly and efficiently and then have it electronically delivered to the kitchen.

The kitchen staff will be able to view the incoming orders and notify the proper wait staff when the food is ready.

1- Customer Statement of Requirements

Problem Statement

Chef:

Working in a kitchen can be strenuous and it can be difficult to keep track of everything that needs to be done. Orders need to be filled in a reasonable amount of time, while simultaneously ensuring that any special dish accommodations are accounted for. This needs to be done with accuracy because it can be a disaster if an order is made incorrectly or even worse, a person with a food allergy could get hurt as a result of a waiters poor handwriting. We want to get our orders out with accuracy and speed to make sure we have a happy, returning customer. When dishes are complete and the server does not pick it up for a few minutes, it can be a pain because the food gets cold and it takes up space on the counter that could be used otherwise.

It would be a great help if I had something that clearly displays all incoming orders. This can save me more time in the kitchen and make the cooking process more efficient. Finally, I would love a way to signal to the servers that the order is finished and ready to be taken to the table to reduce the time the food sits on the counter.

How will restaurant automation system help me? The application will generate a queue for the chef to follow so that they are aware of what dishes they should be making and provide them with a general idea of how old the order is so that they may stay on track. The application will also allow chefs to ping waiters when a certain order is complete so that they know when to come and pick the meals up so that they may be delivered to the tables.

Host/Hostess:

Whenever I am welcoming customers into our restaurant, I aim to make the start of their experience as flawless as possible. As parties of multiple people arrive, they tend to have different requests and requirements based on how many people they have and where they want to sit. Sometimes you can have a party of 8 looking for a round table in a corner, where as other times you can get a family of 4 wanting a booth. I love to find them their preferred seating, but sometimes there just isn't an available spot, and we aren't really sure how long it will be for an appropriate spot to open up.

Finding an available table for guests can sometimes be a challenge, especially when I'm unsure of which tables are ready to go. When I'm welcoming customers in, it would be great if there was a way to easily keep track of which tables are currently cleaned and empty and which are occupied, without having to go through endless reams of paper. At the same time, it'd be nice to keep track of how long each table has already been occupied. This will be a big help in providing guests with accurate estimations on when a valid seating arrangement may become available.

How will restaurant automation system help me? Using a handy tablet/phone I can keep on me, I can mark tables as occupied, vacant, clean, etc. as the day goes on. Whenever I seat new guests, I can mark the table as occupied. Once the guests leave, that table can be marked as vacant, notifying a busboy that the table is ready to be cleaned. Once the busboy is finished, they can mark it as cleaned which lets me know that the table is ready to be used once again.

Customer :

I love going out to eat, but I don’t love how slow things can be. When I’m seated, it’d be really great if I could place an order for drinks or appetizers without waiting for a waiter to get to me, especially if the restaurant is incredibly busy. This could really cut down on time spent waiting. If I have a special request immediately after my initial order or need to notify the waiter for anything, it can be rough getting someone’s attention. Waiting for a server can be a major inconvenience while dining out. It’s usually very difficult to get their attention unless they walk right past you, and if no one ever walks by, you might be waiting extended periods of time before anyone notices you. It can also be difficult to split a bill while out with friends, often times it doesn’t get mentioned until the very end of the outing, and it becomes an awkward experience determining with the server what the best way to split the bill is. Some people prefer to split it by itemizing and some by simply splitting the total bill at the end. When this hasn’t been properly planned it can be difficult to figure out how to fix it.

How will restaurant automation system help me? A tablet on the table with a menu that my party could use to place orders immediately after being seated would be a great idea. It would also be a great way to have more interactive menus that could give me more information on each item, since it can take a while to call a server over to my table. The ability to simply press a button in an app to call the server over to my table would be an easy solution to this problem. It would also be fantastic if I could even order on the app.

Server:

Working as a server is an extremely demanding job. Servers have to run around the restaurant taking customers orders, keeping track of which table ordered what, and returning orders. Servers have to interact with chefs/cooks and get food to people right away so the food does not get cold. They also have to enter checks when a customer or group is ready to pay. Since there are many tables in this restaurant, it is somewhat difficult to connect a certain order to a certain table, and also to determine whether a table needs to be cleaned and set up for the next customer. Unfortunately, sometimes a server may have to send a dish back if the customer does not like it or finds something wrong.

How will restaurant automation system help me? Having an application that could help take care of customer's orders so that they can take their time would be delightful. It would be amazing if the software would be able to send the orders directly to the kitchen and remember which table and which person ordered each item. Sometimes customers need help however I'm not currently in the same area as them due to other job responsibilities. A way for my customers to ping me whenever they need my help would be here in time I would also love to be able to receive notifications from the chef once orders are finished getting prepared so the food isn't standing out for so long.

Busboy:

Working as a busboy can be a very time consuming job. At times it's hard to keep track of all the tables currently being used at the restaurant and whether or not they are vacant for me to clean.
How will restaurant automation system help me? Having an easy way to look at the current status of tables would be a huge advantage. Being able to see which tables are ready to be cleaned and prepared at any given time would be a huge help in efficiently getting stuff ready. Once I'm done, I'd love to be able to update the status of the table is ready so it can beused for incoming customers

Terms Definition s.s:

Application - A software designed to perform a group of coordinated functions, tasks or activities for the benefit of the user.

Busboy - Clears tables, take dirty dishes to the dishwasher and set up the tables to be occupied again.

Chef - Makes food that is requested by the customers.

Customer - Someone that comes to the restaurant as a guest to be waited on and served food.

Host/Hostess - Greets incoming customers and assigns seats to each customer.

Menu - A list of food available to be prepared, cooked, and presented, including pictures and ratings.

Manager - Supervises all staff on board and makes sure that everything is in working order.

Employee Portal - Allows employees to clock into work from our system.

Manager Account - A user account that allows access to the data collected by the application as well as having extra features that allows management of employees and the restaurant.

Filter (food) - To remove meals that have certain ingredients that are unwanted by customers.

Operating System - System software on which the application will run.

Rating - A position or standing of something determined by a customer's feedback.

Reservation - An arrangement made in advance to secure a table.

payment – Sum of money that will be paid after the order and checking out.

Restaurant Automation - Makes a restaurant flow more efficiently and more easily. Eliminates certain tasks that servers would normally have to do. Uses devices with preloaded software that manages several tasks which helps eliminate many of the required tasks that are normally done via employees.

Queue - A first in, first out way of handling orders to ensure that food arrives in order that it is requested.

System Requirements

Functional Requirements

Identifier:	**Requirement:**
REQ-1	The application will allow customers to select an open table based on an interactive seating chart, after the user has logged in. Once selected, the table is marked as “Occupied”.
REQ-2	The application will present an interactive graphical menu once the customer is seated at the table.
REQ-3	The application will provide employees with an Employee Portal.
REQ-4	The application will keep track of all employee hours for payroll based on clock-in / clock-out reports .
REQ-5	The application will notify the busboy when a customer has paid their bill to indicate that they are leaving, so the busboy can clean the table.
REQ-6	The application will allow for customers to conjoin two tables together if desired.
REQ-7	The application will allow for customers to split the bill.
REQ-8	The application will provide the option to make a reservation at the restaurant ahead of time.
REQ-9	The application will notify and send the incoming orders to the chef, adding it to the end of the queue in order of time placed.
REQ-10	The application will allow a “Take-Out” option where the take-out orders are placed and taken out.
REQ-11	The application will allow the customer to sign-up to be a part of a Rewards program.

REQ-12	The application should add points earned every time a customer makes a purchase.
REQ-13	The application should deduct points and apply them appropriately to the total bill when the customer decides to use them.
REQ-14	The application will notify the server when a customer’s food is ready, so they can deliver it to the designated table.
REQ-15	The application will allow the option for the menu to be translated, if needed.
REQ-16	The application will allow customers to filter menu items according to dietary restrictions and notify the chef when the order is sent.
REQ-17	The application will allow the customer to rate and leave comments on the order at the end of the meal.
REQ-18	The application will allow for servers to mark tables as “Occupied”.
REQ-19	The application will allow the customer to alert the server, using a button on the application.
REQ-20	The application will allow a table to be marked as “Ready” once the busboy has cleaned it.
REQ-21	The application will allow the customer to pay on the spot with a credit-card, if desired.
REQ-22	The application will give the option of receipt choice (i.e. paper, e-mail, none).

Nonfunctional Requirements

Identifier:	**Requirement:**
REQ-23	The application should be compatible with iOS and Android operating systems.
REQ-24	The application should be easy for the typical person to use.
REQ-25	The application should be aesthetically pleasing.
REQ-26	The application should be able to startup and shut down quickly.
REQ-27	The application should have smooth transitions from page to page.
REQ-28	The application should support a functioning restaurant.

Customer Interface Requirements

Identifier:	**Requirement:**
CSREQ-1	Customer gets to choose among “Log In”, “Continue As Guest”, and “Sign Up”.
CSREQ-2	Clicking on the “Login” button takes the customer to the Login Page where he/she inputs her account username and password. If entered correctly, the user can next see the Tables page.
CSREQ-3	Clicking on the “Continue As Guest” button takes customer to a page where he/she gets to choose between Dine-In and Take-Out.
CSREQ-4	Clicking on the “View Menu” button as guest takes the customer to a Menu Page **without** “Add to Cart” option to prevent checking out without either logging in or continuing as a guest.
CSREQ-5	Clicking on Dine-In takes the customer to the Tables page so that he/she can choose a table to sit at.

CSREQ-6	Clicking on a ready table (colored in green) takes the customer to the Menu Exclusion page where he/she can filter out the menu according to his/her dietary restrictions/allergies. The user is also allowed to skip this page simply by pressing on Continue. Clicking on an occupied or dirty table (colored in red or coral) is forbidden.
CSREQ-7	Clicking on the Take-Out button will take the customer straight to the Menu page.
CSREQ-8	After the confirmation of the customers Dietary Restriction/Allergies, the customer can head to the Menu page which is already filtered out.
CSREQ-9	After selection from the menu, the customer gets a summary/overview of her selections in the summary page and he/she can proceed to Progress page.
CSREQ-10	In the Progress Page, the customer can see the progress of his/her order and Order More or Checkout/Pay.
CSREQ-11	Clicking on “Payment” takes customer to the Payment page where he/she can either pay using a Credit/Debit Card or Cash.
CSREQ-12	After payment, the Receipt/Feedback page appears where the customer can choose how they want their receipt or not and rate his/her experience of ordering via the app.
CSREQ-13	Clicking on the Home page, the customer can go back to the dine-in/take-out page where she can place another order - let’s say, for takeout.
CSREQ-14	Clicking on the “Sign Up” button takes the customer to the Sign Up screen where he/she is able to input the required information and then use the account to log in.

Manager Interface Requirements

Identifier:	**Requirement:**
MSREQ-1	Owner/Manager logs in with his/her username and password.
MSREQ-2	The default page of the Manager Interface holds options of Employee Info, Orders, Tables, Menu, and Logout.
MSREQ-3	The bar at the bottom of the screen will have options to navigate to the following pages: Employees Info, Orders, Tables, Menu, and Logout.
MSREQ-4	Employee Info takes the owner to view a list of all his employees and their information.
MSREQ-5	Clicking on Orders takes the owner to the Orders page where he/she can see a list of active/past orders and their statuses.
MSREQ-6	Logout takes the owner back to the Login Page.
MSREQ-7	Menu takes the manager to either view the current menu and/or update the current menu.

Employee Interface Requirements

Identifier:	**Requirement:**
ESREQ-1	Employee logs in with his/her username and password.
ESREQ-2	The Employee clocks in and the time, location of clock-in, IP address, and current clocking status gets logged into the server under the employee’s account.
ESREQ-3	Default page of Tables appears after login to show which tables are ready (green), occupied (red), or dirty (coral).

ESREQ -4	The bar at the bottom of the screen will have options to navigate to the following pages: Orders, Tables, Clock Out and Logout.
ESREQ-5	The Employee gets to Clock Out. This is only possible if the employee has previously clocked in. Otherwise, the employee will be prompted with an appropriate error message.

Sub Systems used in Use Cases Diagram

UC-1: Reservation - Allows customers to make reservations online before they come to the restaurant to reserve a table.

From: REQ-8

UC-2: Payment - Allows customers to split the bill, choose option of receipt (email, paper) and allow customers to pay on the spot with a credit-card reader (if desired).

From: REQ-5, REQ-7, REQ-12, REQ-13 REQ-21, REQ-22, CSREQ-10, CSREQ-11, CSREQ-12

UC-3: View Menu - Allows customers to view the entire menu with our system.

From: REQ-2, REQ-15, REQ-16, CSREQ-4, CSREQ-8

UC-4: Meal Prep - Allows chefs to be updated on the status and requirements of their orders.

From:: REQ-9, REQ-16, ESREQ- 4

UC-5: Rate Food - Allows customers to rate their overall restaurant experience.

From:: REQ-17, CSREQ-12

UC-6: Food Filters - Allows customers to filter out food according to dietary restrictions or preferences.

From:: REQ-16, CSREQ-8

UC-7: Clocking In/Out - Allows employees to clock in when they come to work and clock out when they go on break/go home.

From:: REQ-3, REQ-4, ESREQ-1, ESREQ-2, ESREQ-4, ESREQ-5, MSREQ-4

UC-8: Serving - Allows waiters/waitresses to keep track of their current orders and customer status.

From:: REQ-6, REQ-14,REQ-18, REQ-19, REQ-20, ESREQ-4

UC-9: Placing an Order - Allows servers or customers to send their orders to the kitchen.

From:: REQ-9,ESREQ-4, CSREQ-10

UC-10: Table Marking - Allows servers to mark a table as occupied when customers sit down and allows busboys to mark a table as ready once it is cleaned.

From: REQ-1, MSREQ-5, ESREQ-3, ESREQ-4

UC-11: Earning Rewards - When the customer makes a purchase they earn rewards. With a certain amount of rewards the customer can receive discounts or a free drink as an example.

From:: REQ-11, REQ-12, REQ-13

UC-12: Redeeming Rewards - When the customer makes a purchase, they can use points earned toward the cost of their meal.

From:: REQ-12, REQ-13

UC-13: Take-Out - An option for the customers that allows them to order food from the restaurant for pick up to take home.

From:: REQ-10, CSREQ-7

UC- 14: Table Selection - Allows the customer to select a table after either logging in or continuing as a guest if dining in by viewing the floor plan, which will have the status of the tables shown and allowing them to select from all of the open tables.

From:: REQ-1, REQ-18, CSREQ-5

UC-15: **Table Plan Status -** Allows the manager and the servers to change the status of a table from Ready to Occupied to Dirty and back to Ready as desired. Tables conjoined together will change status in unison and have matching Order IDs.

From:: REQ-6, MSREQ-2, MSREQ-3, MSREQ-5

UC-16: **Login -** Allows users to login which will dictate which interfaces (Employee or Customer) they will view and the different potential functions that they may access.

From:: REQ-3, REQ-11, CSREQ-1, CSREQ-2, CSREQ-3, MSREQ-1, ESREQ-1

UC-17: Create Account - Allows users to create an account so that they may get the benefits of being an account holder.

From:: REQ-11, CSREQ-1

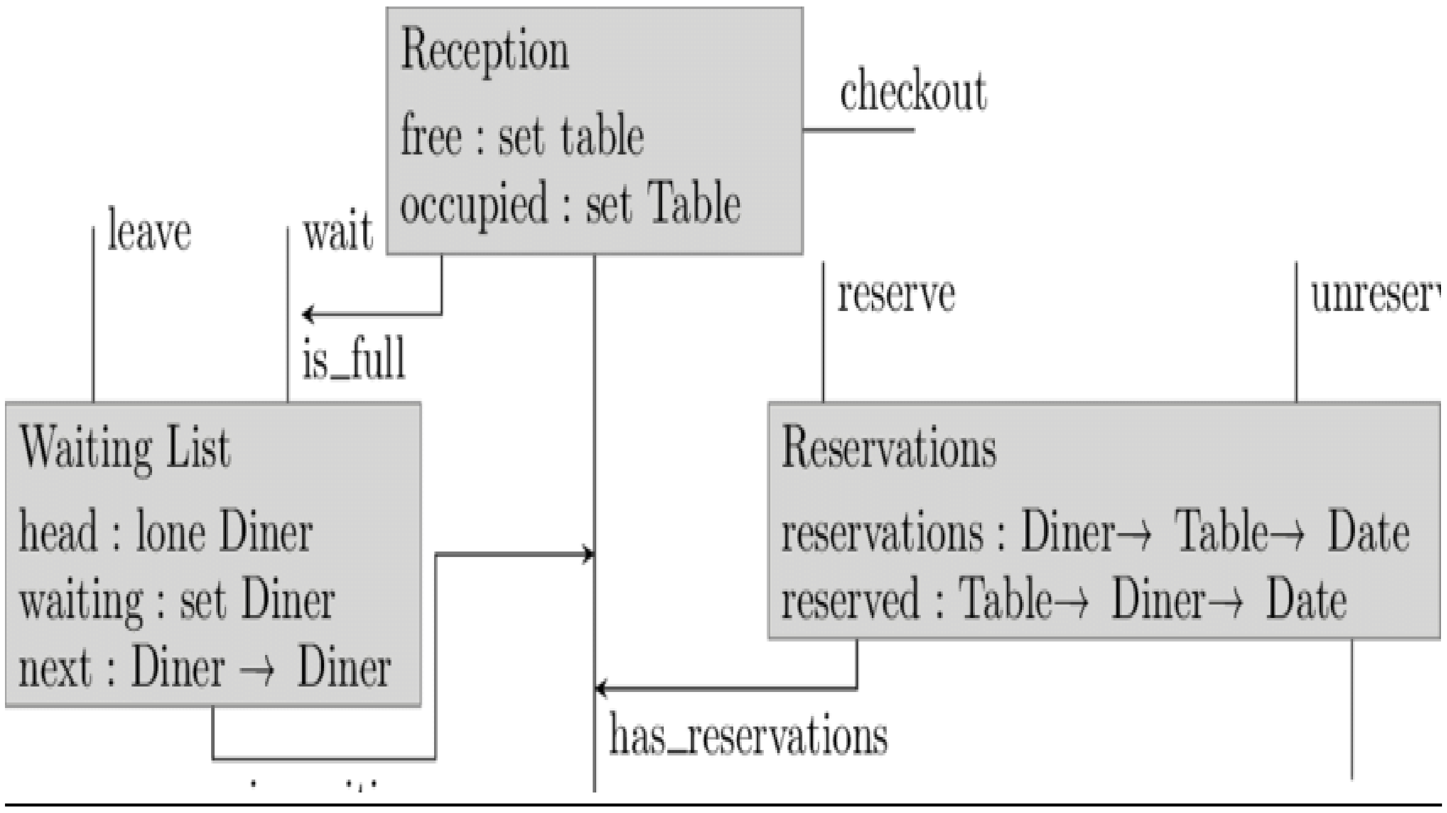
UC-18: Translation - Allows user to have the option to translate the menu into another language, so they do not need a translator.

From:: REQ-15

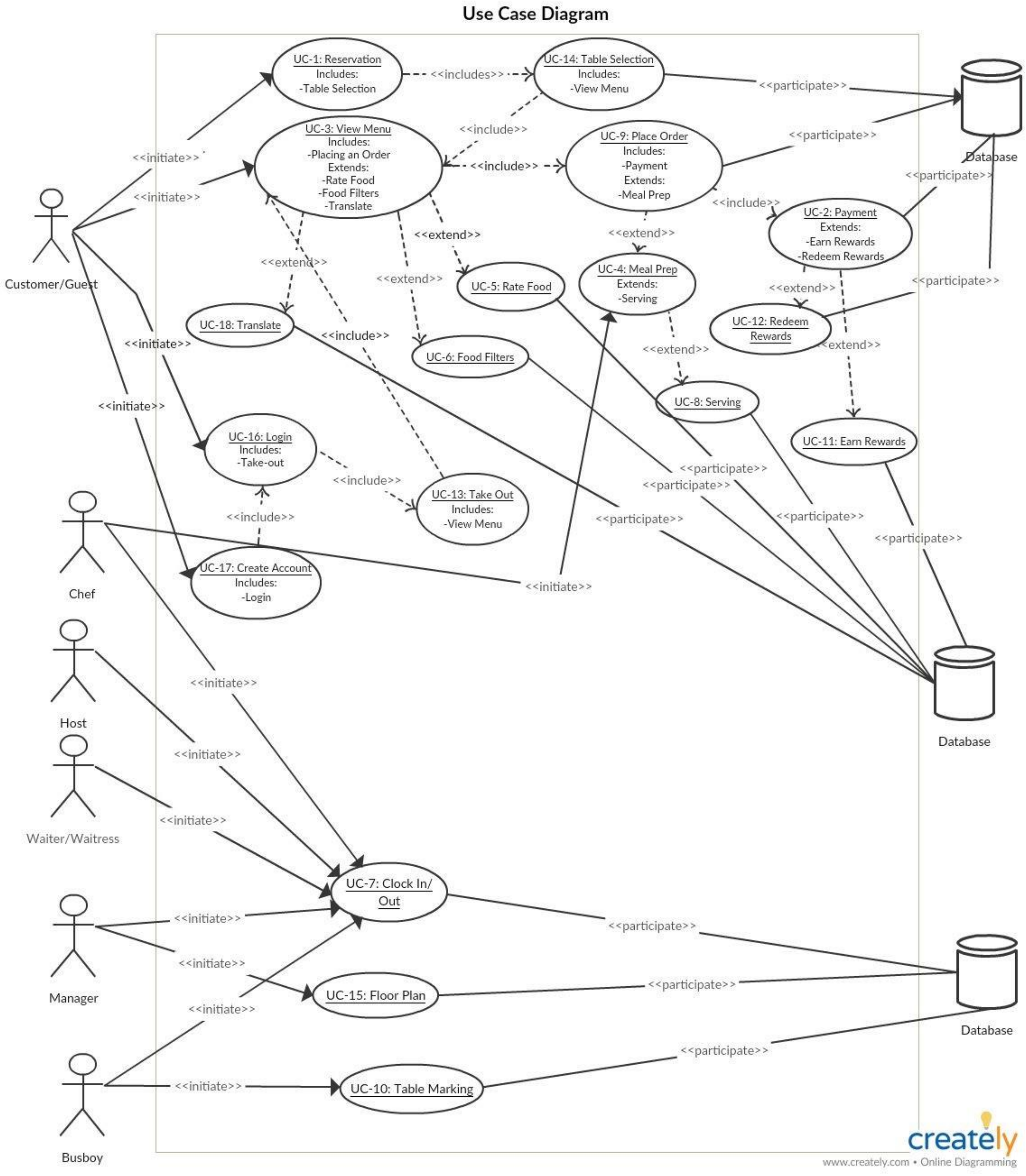
UC-19: Menu Changes - Allows the manager to easily be able to update and edit the menu.

From:: MSREQ-7

SubSystem model:



Use Case Diagram


Use Case Diagram
UC-1: Reservation Includes: -Table Selection
UC-14: Table Selection Includes: -View Menu
UC-3: View Menu Includes: -Placing an Order Extends: -Rate Food -Food Filters -Translate
UC-9: Place Order Includes: -Payment Extends: -Meal Prep
UC-2: Payment Extends: -Earn Rewards -Redeem Rewards
UC-4: Meal Prep Extends: -Serving
UC-5: Rate Food
UC-6: Food Filters
UC-8: Serving
UC-11: Earn Rewards
UC-12: Redeem Rewards
UC-18: Translate
UC-16: Login Includes: -Take-out
UC-13: Take Out Includes: -View Menu
UC-17: Create Account Includes: -Login
UC-7: Clock In/ Out
UC-15: Floor Plan
UC-10: Table Marking
Customer/Guest
Chef
Host
Waiter/Waitress
Manager
Busboy
Database
<<includes>>
<<include>>
<<extend>>
<<initiate>>
<<participate>>
creately
www.creately.com • Online Diagramming

Traceability Matrix::

Req's	1	2	3	4	5	6	7	8	9	10	11	12	3	14	15	16	17	18	19
REQ-1										X				X					
REQ-2			X																
REQ-3							X									X			
REQ-4							X												
REQ-5		X																	
REQ-6								X							X				
REQ-7		X																	
REQ-8	X																		
REQ-9				X					X										
REQ-10													X						
REQ-11											X					X	X		
REQ-12		X									X	X							
REQ-13		X									X	X							
REQ-14								X											
REQ-15			X															X	
REQ-16			X	X		X													
REQ-17					X														
REQ-18								X						X					
REQ-19								X											
REQ-20								X											
REQ-21		X																	
REQ-22		X																	
CSREQ-1																X	X		
CSREQ-2																X			

CSREQ-3																X			
CSREQ-4			X																
CSREQ-5														X					
CSREQ-6																			
CSREQ-7													X						
CSREQ-8			X			X													
CSREQ-9																			
CSREQ-10		X							X										
CSREQ-11		X																	
CSREQ-12		X			X														
CSREQ-13																			
CSREQ-14			X																
MSREQ-1																X			
MSREQ-2															X				
MSREQ-3															X				
MSREQ-4							X												
MSREQ-5										X					X				
MSREQ-6																			
MSREQ-7																			X
ESREQ-1							X									X			
ESREQ-2							X												
ESREQ-3										X									
ESREQ-4				X			X	X	X	X									
ESREQ-5							X												

Use Case Description

UC-2: Payment
Related Requirements: REQ-5, REQ-7, REQ-12, REQ-13, REQ-22, CSREQ-10, CSREQ-11, CSREQ-12
primary Actor: Customer
Actor's Goal: To pay for a completed order
Secondary Actors: Database Waiter/Waitress
Preconditions: The user has loaded the system The user has logged in as a Customer and has the menu opened
Postconditions: The user is prompted on payment method The user is prompted on receipt option
Flow of Events for Main Success Scenario: 1. The customer clicks on the "Checkout" button on the menu screen. 2. The system displays the payment menu. 3. The system displays payment method options. 4. The customer chooses the "Cash" button in order to pay by cash. 5. The system prompts the customer with receipt options. 6. The customer chooses the "Print Receipt" button. 7. The system notifies the waiter/waitress to print the receipt and bring it back to the customer.
Flow of Events for Alternate Success Scenario: 1. The customer clicks on the "Checkout" button on the menu screen. 2. The system displays the payment menu. 3. The system displays payment method options. 4. The customer chooses the "Credit" button in order to pay by credit. 5. The system prompts the customer with receipt options. 6. The customer chooses the "E-Mail Receipt" button. 7. The system prompts the user for an email address. 8. The customer types in an email address and receives a receipt through email.

UC-3: View Menu

Related Requirements:
REQ-2, REQ-15, REQ-16, CSREQ-4, CSREQ-8

primary Actor: Customer

Actor’s Goal:
To view the menu

Secondary Actors: Database

Preconditions:
The user has loaded the system

Postconditions:
The user is shown the available menu

Flow of Events for Main Success Scenario:

1. ← The system gives options for “Login”, “Continue As Guest”, and “Create Account”.
2. → The customer selects to login to their account.
3. ← The system prompts the customer to login with their information.
4. → The customer inputs information for account.
5. ← The system gives options for “Dine-In” or “Take-Out”.
6. → The customer selects option, “Dine-In”.
7. ← The system allows the customer to select a table from the seating chart.
8. → The customer sits at the table.
9. ← The system provides options to select Dietary Restrictions.
10. → The customer chooses to not filter the menu.
11. ← The system displays the available menu.

Flow of Events for Alternate Success Scenario:

1. ← The system gives options for “Log-In”, “Sign-Up”, or “Continue as Guest”.
2. → The customer selects “Continue as Guest”.
3. ← The system gives options for “Dine-In” or “Take-Out”.
4. → The customer selects option, “Dine-In”.
5. ← The system allows the customer to select a table from the seating chart.
6. → The customer sits at the table.
7. ← The system provides options to select Dietary Restrictions.
8. → The customer chooses to not filter the menu.
9. ← The system displays the available menu.

UC-7: Clocking In/Clocking Out
Related Requirements: REQ-3, REQ-4, ESREQ-1, ESREQ-2, ESREQ-4, ESREQ-5, MSREQ-4
primary Actor: Employee
Actor’s Goal: To accurately record the amount of time that they spend while on the job and to make sure the employee is actually at the restaurant when they clock-in and out
Secondary Actors: All Employees (i.e. Server, Busboy, etc.)
Preconditions: The user has downloaded the application
Postconditions: The user has successfully clocked in/out The database logs the information for future use
Flow of Events for Main Success Scenario: 1. → The employee opens the application and chooses to login as a regular employee. 2. ← The system prompts the employee for their login information. 3. → The employee logins in with their information. 4. ← The system presents the employee with an Employee Portal. 5. → The employee selects the clock-in button. 6. ← The system keeps track of the employee’s clock-in data. 7. ← The system verifies that the employee is actually in the restaurant by checking the IP address and GPS coordinates. 8. → When the employee has completed the shift for the day, the employee will select the clock-out button. 9. ← The system keeps track of the employee’s clock-out data. 10. ← The system verifies that the employee is actually in the restaurant by checking the IP address and GPS coordinates.

Flow of Events for Alternate Success Scenario:

1. → The employee opens the application and chooses to login as a manager.
2. ← The system prompts the manager for their login information.
3. → The manager logins in with their information.
4. ← The system presents the manager with a manager interface.
5. → The manager selects the clock-in button.
6. ← The system keeps track of the manager's clock-in data.
7. ← The system verifies that the manager is actually in the restaurant by checking the IP address and GPS coordinates.
8. → When the manager has completed the shift for the day, the manager will select the clock-out button.
9. ← The system keeps track of the manager's clock-out data.
10. ← The system verifies that the manager is actually in the restaurant by checking the IP address and GPS coordinates.

.

UC-11: Earning Rewards
Related Requirements: REQ-11, REQ-12, REQ-13
primary Actor: Customer
Actor's Goal: To receive coupons/discounts based off the amount of purchases at restaurant
Secondary Actors: Database
Preconditions: The user has loaded the system The user intends to make a purchase
Postconditions: The user has earned an appropriate amount of points for how much was spent
Flow of Events for Main Success Scenario: 1. ← The system asks the customer at the beginning to login/create account. 2. → The database will retrieve the user's current point balance. 3. ← The customer chooses all the food items and confirms order. 4. → The customer finishes payment. 5. ← The database updates points based on transaction.

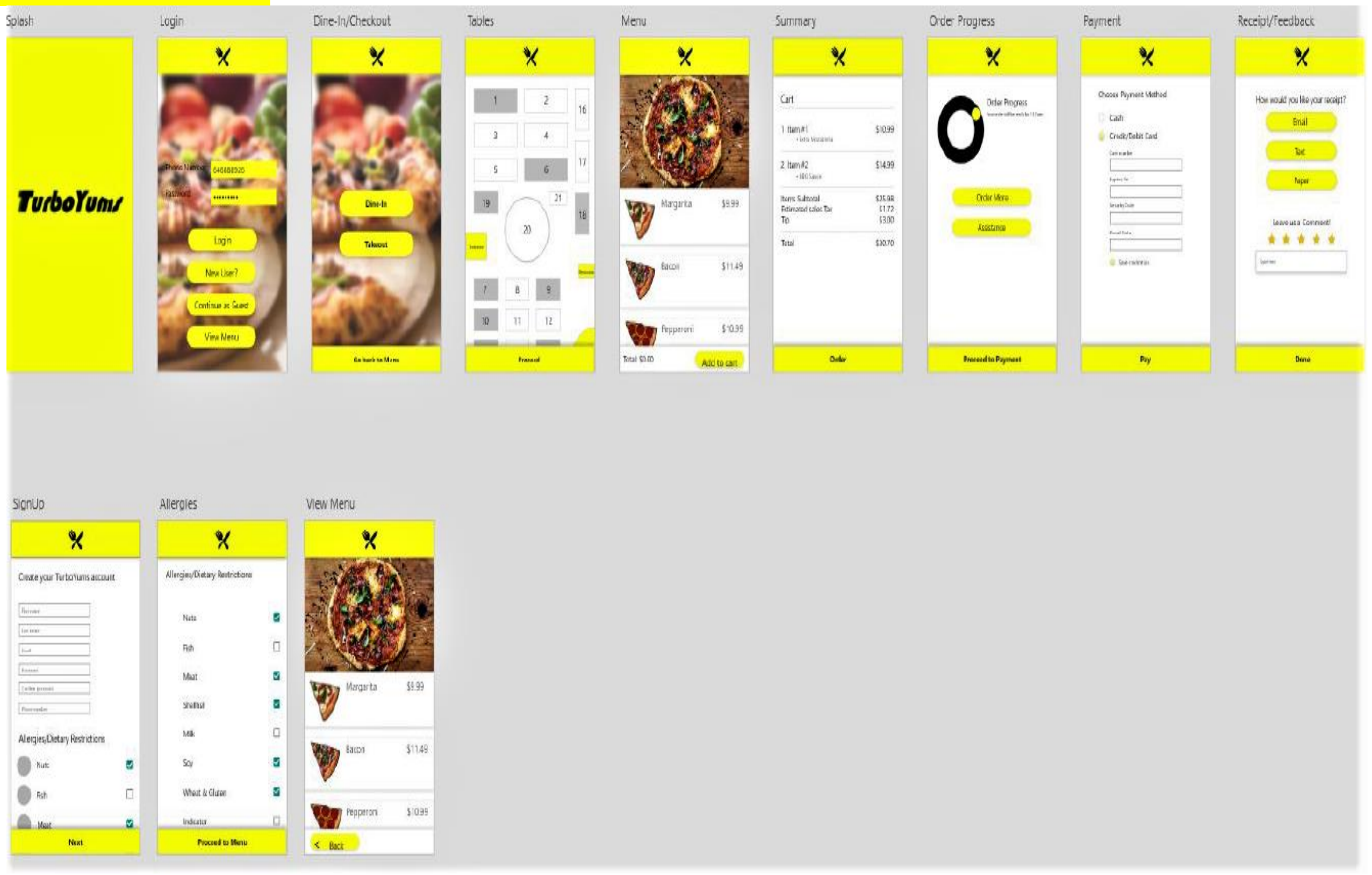
Flow of Events for Alternate Success Scenario:

1. ← The system prompts the customer at the beginning to login/create account.
2. → The database will retrieve the user’s current point balance.
3. → The customer confirms order.
4. → Customer does not complete payment.
5. ← Database returns that transaction is incomplete, points not added.

Graphical user interface (GUI)

*Screen appearance requirements :

(customer interface):

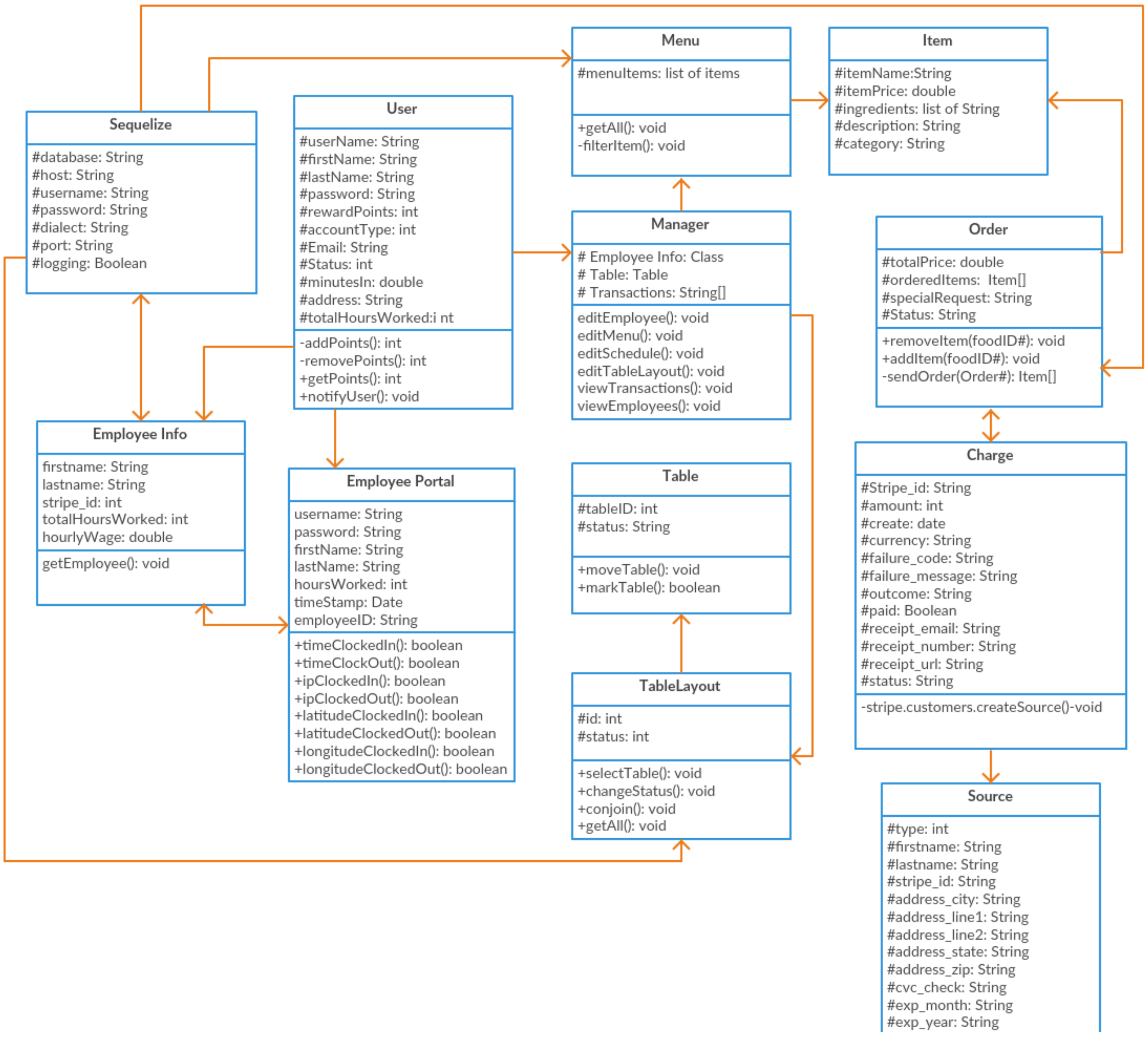


List of internal and external Interfaces definition:

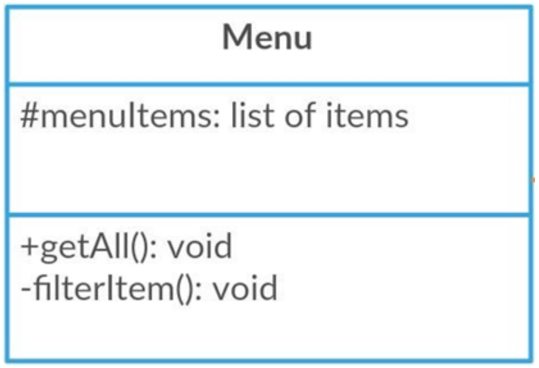
Interfaces	attributes	definitions
Customer Profile	accountUsername	Associated username of the customer (email). Guest account is assigned if no account.
	accountPassword	Password of user account
Interface	confirmOrder	Allows the user to confirm order after choosing food items
	receipt	Allows the user to choose which way they would like to receive the receipt (email, text, paper)
	rateMeal	Allows the user to rate a meal, and then have that rating stored and displayed
	tableStatus	Provides the user with the up-to-date status of each table in the restaurant. Tables can exist in 3 states: available, occupied and dirty
Payment System	paymentMade	Updates system depending on whether the payment has been made.
Food Status	orderStatus	Allows the chef to update the status of the current order being cooked
	orderReady	Allows chef to signal to waiter/waitress and customers that the order is finished
Order Queue	chefQueue	Keeps track of submitted food orders in the order that they were submitted for the chef to follow
Controller	tableList	Shows the customer the current tables available
	tableConfirm	Table is greyed out once it is picked by the customer and stays grey until cleaned
	paymentMade	Once the customer pays, the table is then confirmed and the order is initiated in the kitchen
Communicator	customerMeal	Each meal that the customer orders is stored in the database
	customerMealOrder	Meals in the queue are ordered and sent to a specific chef, to balance the chef work load
Floorplan	viewPlan	Displays the current layout of the restaurant
	tableAdjust	Allows the manager/server to adjust a table in the restaurant layout
Employee Profile	hoursWorked	Displays hours a particular employee has worked in a particular payment cycle
	tablesAssigned	Shows what tables the employee has been assigned recently (threshold to be determined)

	role	Role of the employee: food server, chef/cook, caterer etc.
Table Status	tableNumber	Displays the table number
	seatCount	Max number of people that can be seated at a particular table
	currTableStatus	Is table empty(white), occupied(grey) or uncleaned(red)?
Reward System	currUserPoints	Displays present user reward points balance
	rewardsRedeemable	Allows user to redeem available rewards
DB Connection	dbAuthenticationCreds	Stores the DB login for authorized personnel

Class Diagrams and Interface Specification:



Data Types and detailed definitions:



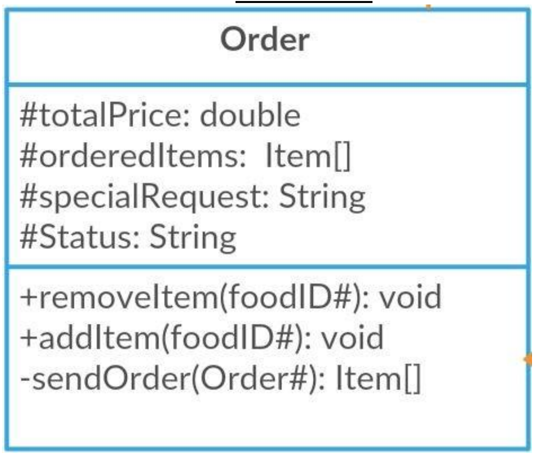
Class: Menu

Attributes:

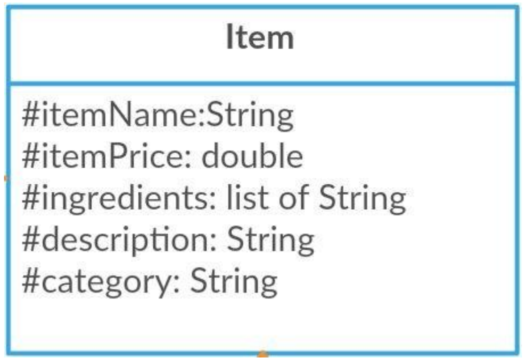
- menuItems: list of Item - The food items available to the customer.

Methods:

- getAll() - The database retrieves all of the menu items.
- filterItem() - The database filters out menu items, according to preference. **Class: Order** Attributes:



- totalPrice:double - The total price of all items in the order.
- orderedItems: Item - The list of food items in the cart picked by the customer.
- specialRequest: String - A place for customers to express any request not available in the UI.
- Status: String - Maintain the status of the order Methods:
- removeItem(foodID#)- removes food from existing order- will take a food ID and quantity.
- addItem(foodID#)-adds food/beverage indicated by the customer to the current order.
- sendOrder(Order)- sends the order to the kitchen to be made. **Class: Item** Attributes:

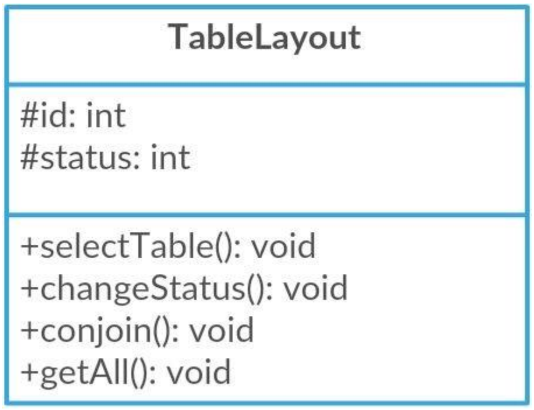


- itemName: String - The name of an item.
- itemPrice: double - The price of each item.
- ingredients: list of String - a list of the different ingredients needed to make the item.
- description: String - each item is described with a few short sentences.
- category: String - a string to keep track of what category an item may be included in **Class: TableLayout** Attributes:

- id: int - The ID of a table.
- status: int - The current state/status of a table - ‘green’ is available, ‘red’ is occupied, and ‘coral’ is dirty.

<u>Methods:</u>

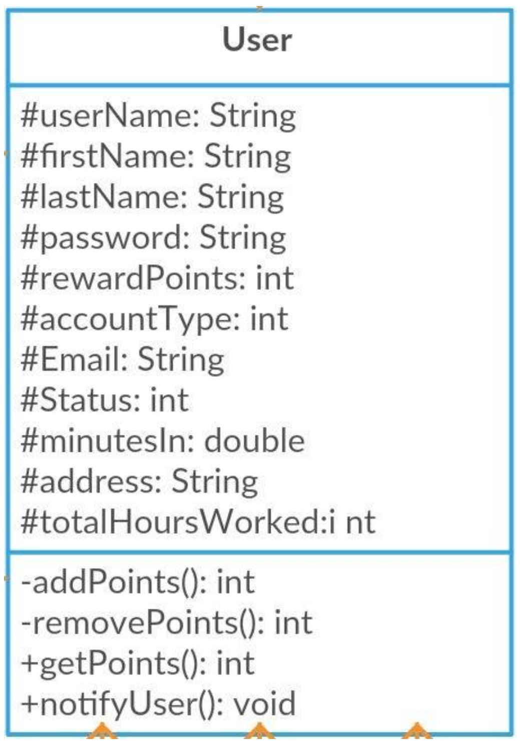
- selectTable() - The customer selects a table.
- changeStatus() - The status of a table will change status upon selection.



conjoin() - The customer is able to select two tables to conjoin them.

getAll() - The database retrieves all of the tables in the restaurant. **Class: User** <u>Attributes:</u>

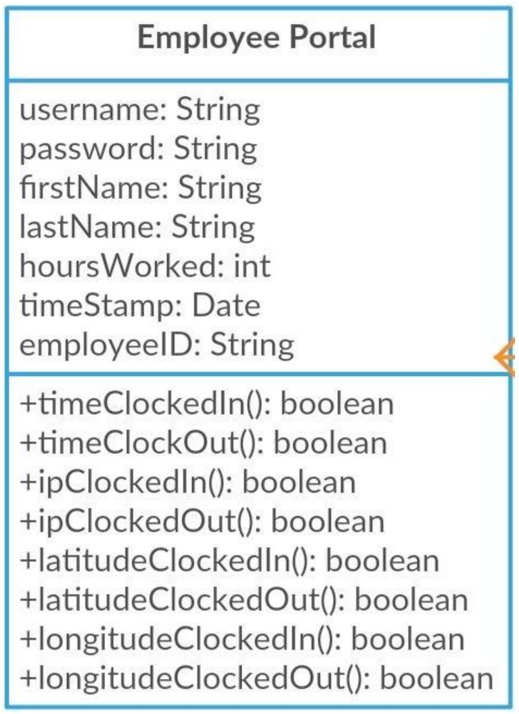
- userName: String - The username of the user.
- firstName: String - The first name of the user.
- lastName: String- The last name of the user.
- password : String - the password of the user.
- rewardPoints: int - How many rewards points the user has.
- rewardBalance: double - The amount of rewards the user has (1 point is not 1 reward)
- accountType: int - The type of user account.
- Email: String - Email address of the user.
- Status: Int - an integer value to keep track of whether or not an employee is clocked in.
- minutesIn: Double - record the amount of minutes that a user has spent clocked in
- address: String - The user’s address.
- totalHoursWorked: int - total hours worked by user if the user is of the employee account type.



<u>Methods:</u>

- addPoints() - Add an integer of reward points to the user’s balance.
- removePoints() - Remove an integer of reward points to the user’s balance.
- getPoints() - Get the user’s point balance.
- notifyUser() - If the user has an active session, send them a message.

Class: Employee Portal - This class is made to allow employees to clock-in and clock-out of their shifts (including breaks).



<u>Attributes:</u>

- username: String - This is the individual employee's self-made username to be able to have an account.
- password: String - This is the individual employee's self-made password to keep their profile protected.
- firstName: String - This is the individual employee's first name.
- lastName: String - This is the individual employee's last name.
- hoursWorked: int - This will keep track of the amount of time that the individual employee has worked during the current shift.
- timeStamp: Date - The time and date of the clock-in/clock-out.
- employeeID: String - This is an employee specific ID, so that the system and manager can identify the employee's role and have the ability to edit and track.

timeClockedIn: String - Employee has the ability to clock in when beginning their shift, time is recorded.

timeClockedOut: String - Employee has the ability to clock out when ending their shift, time is recorded.

- ipClockedIn: String - Employee clocks in, IP is recorded so employee cannot use a non company owned computer or log from home.
- ipClockedOut:String -Employee clocks out, IP is recorded so employee cannot use a non company owned computer or log from home.
- latitudeClockedIn: String - Employee clocks in, latitude is recorded so employee cannot use a non company owned computer or log from home.
- latitudeClockedOut: String - Employee clocks out, latitude is recorded so employee cannot use a non company owned computer or log from home.
- longitudeClockedIn: String - Employee clocks in, longitude is recorded so employee cannot log from home.
- longitudeClockedOut: String - Employee clocks out, longitude is recorded so employee cannot use a non company owned computer or log from home.

<u>Methods:</u>

- clockedIn(): boolean - Employee has the ability to clock in when beginning their shift.
- clockedOut():boolean - Employee has the ability to clock out when ending their shift.

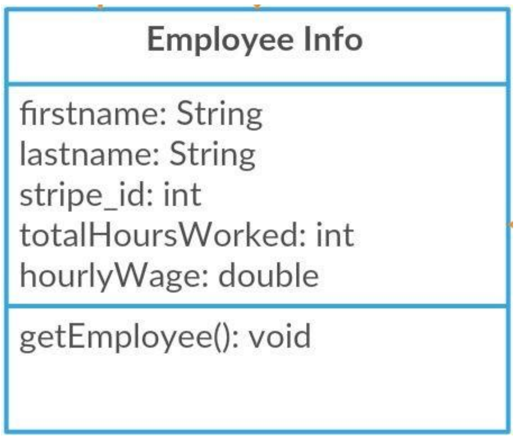
Class: Employee Info - This class is made to hold employee information, if the employee wishes to access it.

<u>Attributes:</u>

- firstname: String - The first name of the employee.
- last name: String - The last name of the employee.
- stripe_id: int - The unique ID of the employee.
- email: String - The email of the employee.
- totalHoursWorked: int - This will keep track of the total amount of time that the individual employee has worked during the payment cycle.
- hourlyWage: double - This will keep track of the individual employee’s working hourly wage.

<u>Methods:</u>

- getEmployee() - This will display the employee’s information.



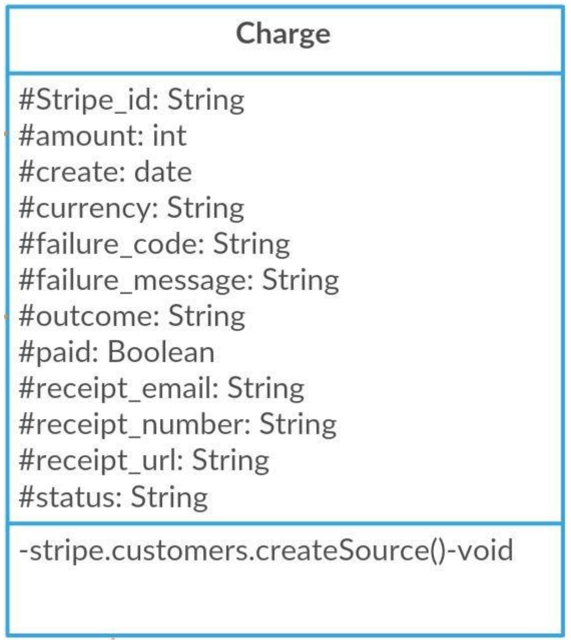
Class: Charge - This class is created to help maintain the information that is associated with a transaction occurring.

<u>Attributes:</u>

- Stripe_id: String - The unique ID of the charge
- amount: int- The amount being charged
- created: date- The time and date that the transaction was enacted.
- currency: String, - The type of currency being used
- failure_code: String - The code associated with the reason for failure
- failure_message: String,- The message displayed when there is a failure

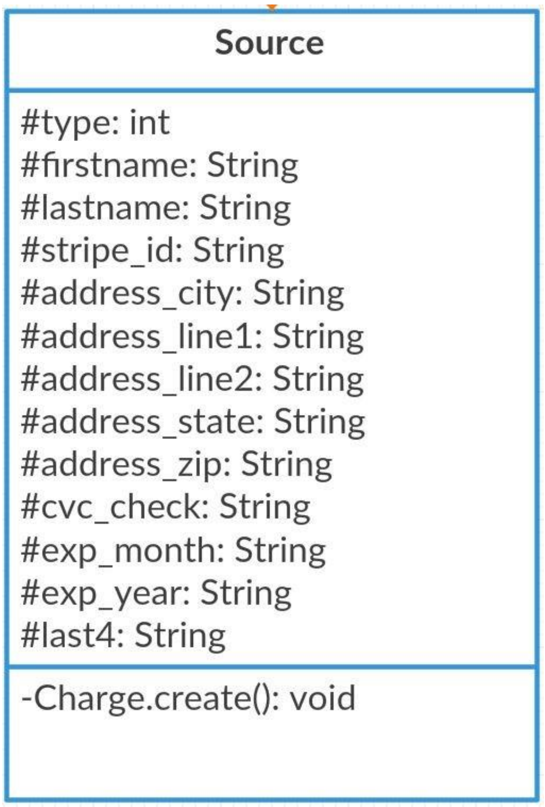
  outcome: String, - The outcome of the charge

  paid: Boolean - used to tell if the order has been paid for or not
- receipt_email: String, - The email the receipt will be sent to
- receipt_number: String, -The unique receipt number
- receipt_url: String, - url associated with the receipt
- status: String, - associated with the status of a charge



<u>Methods:</u>

- stripe.customers.createSource() - Creates a new Source and defines all associated variables **Class: Source -** This class will keep track of all of the details about the method of payment, if it was done by cash or card, and if the transaction is completed by card then the card information will be stored.



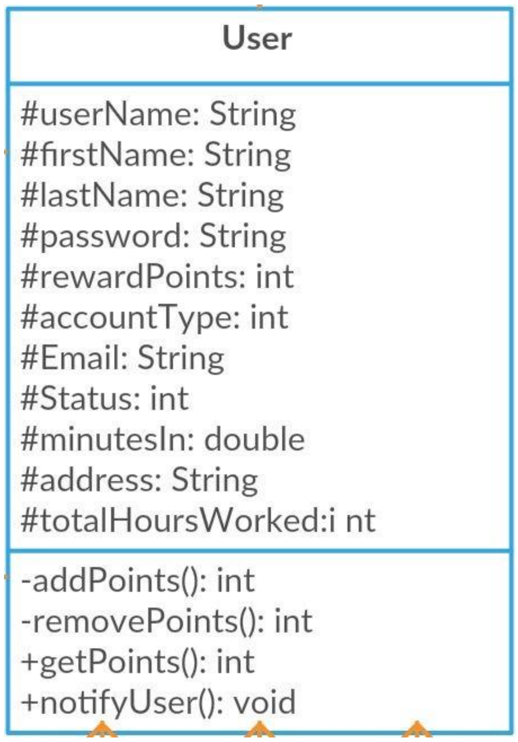
<u>Attributes:</u>

- firstname: String, - The first name of the person being charged
- lastname: String, - The last name of the person being charged
- stripe_id: - The charge's unique stripe ID
- address_city: String, - The city of the billing address
- address_country: String, -The country of the billing address
- address_line1: String, - The address of the billing address
- address_line2: String, - The alternate address of the billing address
- address_state: String, - The state of the billing address
- address_zip: String, - The zip code of the billing address
- cvc_check: String, - The check to make sure the security code of the payment card is correct
- exp_month: String, - The expiration month of the payment card
- exp_year: String, - The expiration month of the payment card
- last4: String, - The last 4 digits of the payment card <u>Method:</u>
- Charge.create() - Creates a new charge for the table and defines all associated variables

conjoin() - The customer is able to select two tables to conjoin them.

getAll() - The database retrieves all of the tables in the restaurant. **Class: User** Attributes:

- userName: String - The username of the user.
- firstName: String - The first name of the user.
- lastName: String- The last name of the user.
- password : String - the password of the user.
- rewardPoints: int - How many rewards points the user has.
- rewardBalance: double - The amount of rewards the user has (1 point is not 1 reward)
- accountType: int - The type of user account.
- Email: String - Email address of the user.
- Status: Int - an integer value to keep track of whether or not an employee is clocked in.
- minutesIn: Double - record the amount of minutes that a user has spent clocked in
- address: String - The user's address.
- totalHoursWorked: int - total hours worked by user if the user is of the employee account type.



Methods:

- addPoints() - Add an integer of reward points to the user's balance.
- removePoints() - Remove an integer of reward points to the user's balance.
- getPoints() - Get the user's point balance.
- notifyUser() - If the user has an active session, send them a message.

Traceability Matrix:

	Software Classes										
Domain Concepts (interfaces)	Menu	Order	Item	TableLayout	User	Employee Portal	Employee Info	Transaction	PaymentMethod	Charge	Source
Customer Profile		X			X			X	X	X	X
Interface	X	X	X	X	X	X			X		
Payment System					X			X	X	X	X
Food Status		X									
Order Queue		X									
Controller	X			X				X	X		
Communicator		X			X						
Table Layout				X							
Employee Profile					X	X	X				
Table Status				X							

Reward System					X			X			
DB Connection		X	X		X		X	X	X		

- <u>Customer Profile:</u>
    - User: Allows customer to view and edit account specific data.
    - PaymentMethod: Allows customer to select preferred payment method.
    - Source: Allows customer to have a saved payment method.
    - Charge: Allows customer to pay for his order.
- <u>Interface:</u>
    - Menu: The restaurant’s menu is accessible via the interface.
    - Order: Orders can be placed through the interface.
    - Item: All available items are presented on the interface.
    - TableLayout: Table selection is done via the interface.
    - User: User data is presented via the app interface.
    - Employee Portal: The portal is accessible via the app interface.
    - PaymentMethod: Chosen through the interface.
- <u>Payment System:</u>
    - User: The purchase is logged to the user’s account after the payment is made.
    - Transaction: The system attempts to validate and confirm payment has been made.
    - Payment Method: The system uses the selected payment method in order to complete payment.
    - Source: The system allows the customer to have a saved payment method.
    - Charge: The system allows the customer to pay for his order.
- <u>Food Status:</u>
    - Order: Food Status initiates once an order is placed.
- <u>Order Queue:</u>
    - Order: Ordering a food item adds it to the queue.

- Controller:
  - Menu: Controller requires that items be chosen from the menu.
  - TableLayout: Controller requires that a valid table is selected.
  - PaymentMethod: Controller requires a valid Payment Method from customer.
- Communicator:
  - Order: Communicator allows the customer’s order to be communicated to the chef via queue.
  - User: Allows accounts to validate information from server, such as login.
- Floorplan:
  - TableLayout: Allows the table to be conjoined to be selected.
  - User: Users will be able to select a table and change the status.
- Employee Profile:
  - User: Contains employee personal information.
  - Employee Portal: Allows employees to clock in/out and view work status.
  - Employee Info: Allows employees to view their current earnings and status.
- Table Status:
  - TableLayout: Allows users to select a table and view its status.
  - User: Can be viewed and edited by a user.
- Reward System:
  - User: All rewards points are saved to a user’s profile.
- DB Connection:
  - Order: Orders will be sent to the database once sent by the customer.
  - TableLayout: Table information is stored on the database.
  - User: User account information will be stored on the database.
  - Employee Info: Employee information will be stored on the database.
  - Payment Method: Saved and preferred payment methods will be stored on the database.

Traceability matrix (between use cases and designed classes):

Look in page 18:

Use Cases	Designed classes (domain concepts)									
	Customer Profile	Interface	Table Status	Payment System	Food Status	Order Queue	Controller	Communicator	Employee Profile	Floor Plan
UC-1	X	X	X					X		X
UC-2	X	X		X				X		
UC-3		X						X		
UC-4					X	X				
UC-5		X								
UC-6		X					X			
UC-7									X	
UC-8					X	X				
UC-9	X									
UC-10			X							X
UC-11	X			X						
UC-12	X			X						
UC-13	X			X						
UC-14	X		X							X

UC-15	X									X
UC-16	X	X							X	
UC-17	X	X							X	
UC-18		X								
UC-19		X					X			

UC-2 - Payment: Customer uses payment so Customer Profile is involved. Payment information is collected and displayed with interface. Once payment is complete the table status can be updated. Communicator communicates information between different parts of payment system. Once payment happens the floor plan can be updated.

UC-3 - View Menu: Menu is displayed to users through the Interface. The Communicator communicates information between the interface and database.

UC-4 - Meal Prep: When food is being ordered and prepared its status is updated. Further when food is ordered it is added to the Order Queue.

UC-6 - Food filter: Selections on filtering are made on the interface and changes made are done by the

Controller.

UC-7 - Clocking In/Clock out: Information stored by Employees clocking in/out is contained in the

Employee’s Profile .

UC-8 - Serving: Once food is served is status is updated and thus its Food Status is changed.

UC-9 - Placing an Order: When someone places an order it is added to the Order Queue and its Food Status is changed.

UC-10 - Table Marking: When marking a table, its Table Status and Floor Plan is updated.

UC-11 - Earning Rewards: A customer may earn rewards and it is done while the customer is paying for their meal.

UC-12 - Redeeming Rewards: A customer may redeem rewards if they have enough points and it is done while the customer is paying for their meal.

UC-13 - Take-out: A customer may get take out and does not need to select a table, but does need to make a payment.

UC-14 - Table Selection: Utilizes the floor plan and interface to display it.

UC-15 - Floor Plan Status: Utilizes the floor plan and interface to display it.

UC-16 - Login: Any use must log in order to access their accounts, so customer and employee profiles are involved, in addition to this user login by typing on the screen and reading the information displayed by the interface.

UC-17 - Create Account: Customers and employees have the ability to create an account and therefore customer and employee profiles are involved. The user must interact with the application and the information that is displayed through the interface.

UC-19 - Menu Changes: In order to change the menu, a manager must interact with the application and the display using the interface and will then update the information in the database via the controller.