CookIt!

IS 385 Fall 2018
Hoan Nguyen #013422030
Victor Le #013159664
Felix Huang #013427386
John Nguyen #016218330
Joji Maldonado #018626801
Josue Cruz #010057110

Table of Contents

Vision Document

Problem Description

According to the Washington Post, the percentage of Americans who eat homemade suppers was about 60% in 2014. 30 years ago, that number was closer to 75%. Across all levels of socio-economic status, there has been a drop in the amount of meals prepared at home; high income households decreased from 88% to 65%, mid-level income households decreased from 92% to 69%, and low income households decreased from 95% to 65%. The Business Insider also published an article stating that food accounts for 12.5%, or just over $7,000 of the average American household's annual expenditures. Eating out accounts for 43% of food expenditures, or about $3,000. By cooking at home, households can make their dollars go a little further.

These decreases can be attributed to the increase in already prepared foods that can be made quickly. Americans are becoming more reliant on prepared and processed foods, and as a result, many grocery stores are increasing their inventory for such goods. However, many celebrities and foodie personalities such as Martha Stewart and Tamir Adler are trying to encourage people to get back into the kitchen.

Today's social norms include diet fads, unhealthy healthy foods, and improper diet advice. People still think that the extreme is what you need to do, and in fact that is rarely the case. The choices we have in our diet are larger than ever due to the largest cookbook in the world, the Internet. We are busier than ever and look for solutions and the faster and easier the solution the better. The world is fast moving, and this is also the case with the way we want our food.

With the expansion of mobile technology and the spike in active users on the Internet, preparing a home cooked meal should be something that should integrate into the technology that people use everyday. Because handheld and mobile devices are being used all the time, a system should be developed that leverages our use of technology and assists in reducing the oddities and time constraints of preparing a meal in the modern world. The system should allow users to select a meal they want to cook and provide them some instruction on how to make that meal a reality.

System Capabilities

This system should be capable of:

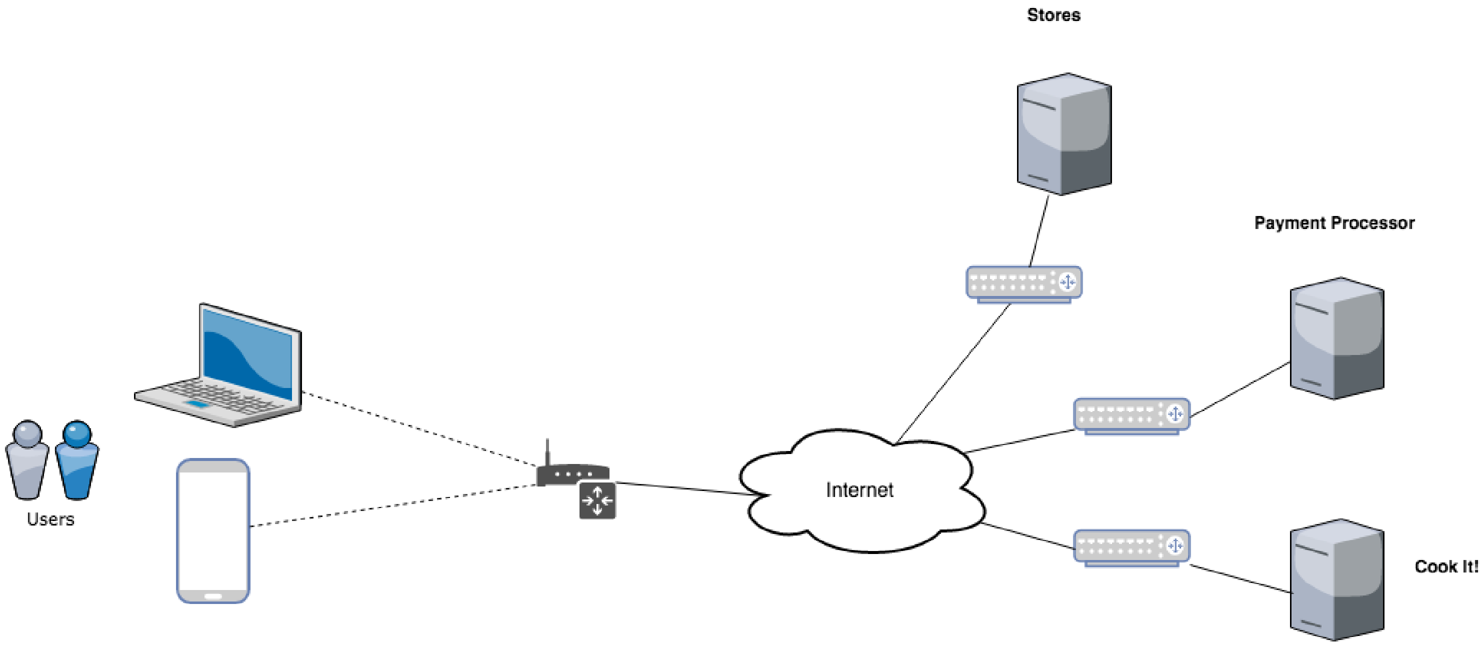
- Allowing users to register and create an account where they can input personal information
- Showing nearby stores and locations where food can be picked up and providing the user the option of selecting a location they want
- Supporting selection of a meal option, displaying nutritional information, and a price for the corresponding meal order
- Showing the progress of the a meal order after it has been submitted
- Displaying instructions on how to prepare the meal based on the gathered ingredients
- Adding and modifying payment information

Benefits

The development of a system like the one described above will greatly benefit is users in the following ways:

- Being able to see the possible meals available allows puts all the power in the hands of the user, thereby reducing any misinformation and indecision
- Viewing the progress of a meal order will increase efficiency because a user can determine at what point in time their order will be ready, thus reducing unwanted waiting times
- Seeing nearby locations where meal order can be picked up makes the ordering experience more personal

Proposed Application Architecture



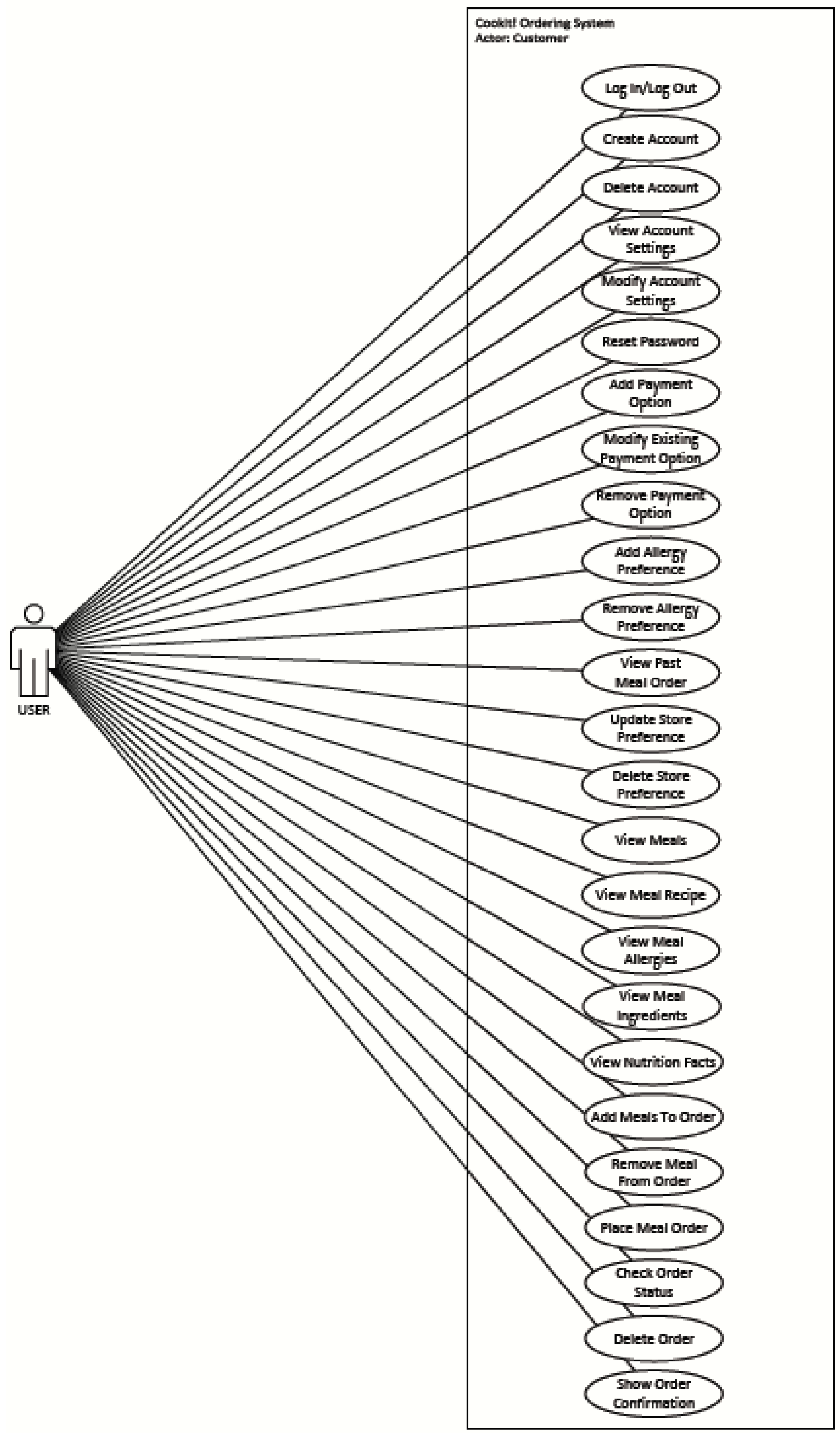
Brief Use Case Descriptions

Use Case	**Brief use case description**
Login	The user inputs their login credentials (username and password). The system checks if the username and passwords are valid. If they are not, the system prompts the user of the error.
Create account	The user inputs their account information: username and password, customer address, zip code, and allergy information. Based on the zipcode the user provides, the system will show a list of nearby grocery stores. The user will also provide their payment information, and choose a customer membership option. Once the user has input all this information, the system prompts them of their registration success and creates a new customer record.
Add meal to order	After the user has logged in, the system retrieves and displays the different food items available to them. The user can select a meal to see more in-depth information and the system will display the Nutrition Facts and Recipe Instructions. After selecting the meal, the user has the option to add the meal to their order. If the meal is added, the system adds the meal to the current order cart, and prompts the user to answer if they want to continue browsing for meals.
Pay for meal	When the user has selected the option to checkout, they are asked to confirm their payment method. The system will display the default payment method which will be the payment method the user entered during their registration. The system allows the user the option to add a new payment method. If the user decides to add a payment method, they are prompted to input the credit card information which the system will store.
View Past Meal Orders	When the user is logged in, they will have the option of viewing their past orders. The system retrieves and displays the user's past orders associated with their account. If the user selects a specific order, the system will display the meal items and total cost of the order.
View Account Settings	The user chooses the option to view their account settings. The system retrieves the customer's settings, and displays the data.
Modify Account Settings	The user changes the value of 1 or more of their settings. When the user has finished with their changes, the system saves these changes in the existing customer record.
Delete Account	The user chooses to deactivate their Cook It! account. The system removes records of the customer.
Reset Password	The user selects the "Forgot My Password" option on the login screen, and the system prompts the user to verify who they are. The system then sends a password reset email to the email that was in the associating customer record.

Add Payment Option	The user provides new payment information (credit card number, CCV, expiration). The system creates a new payment-type record with the data the user provided.
Modify Existing Payment Option	The user modifies a value (credit card number, CCV, expiration) from one of their existing payment options. The system stores the modified values in the existing payment-type record.
Remove Payment Option	The user chooses to delete an existing payment option. The system removes the corresponding payment-type record.
Add Allergy Preference	The user modifies their existing allergy selections. The system stores the values and creates new customer allergy records for new selections.
Remove Allergy Preference	The user chooses to delete an existing allergy preference. The system removes the corresponding customer allergy record.
Update Store Preference	The user inputs their zipcode. The system returns and displays a list of nearby grocery stores corresponding to the zipcode. The customer selects the store that they want to add, and the system creates a new customer-store record for the selection.
Delete Store Preference	The user chooses to delete one of their previous store selections. The system removes the corresponding customer-store record.
View Meals	The system loads the Meal Homepage when the user logs in. The user selects a meal, and the system retrieves and displays information corresponding to that meal.
View Meal Recipe	The user selects a meal to view. The system retrieves and displays the information corresponding to that meal. As part of this data, the system displays the meal recipe.
View Meal Allergy	The user selects a meal to view. The system retrieves and displays the information corresponding to that meal. As part of this data, the system displays allergies the meal has.
View Ingredients	The user selects a meal to view. The system retrieves and displays the information corresponding to that meal. As part of this data, the system displays the meal ingredients.
View Nutrition Facts	The user selects a meal to view. The system retrieves and displays the information corresponding to that meal. As part of this data, the system displays the meal nutrition facts.
Remove Meal from Order	The user selects the option to view the current shopping cart and the system returns the meals in the current order. The user chooses the option to remove a meal. The system removes the meal-order record and the previously selected meal is no longer displayed.
Place Meal Order	After the payment method is selected, the user confirms their order. The system displays an order confirmation and a progress indicator showing the status of their order.
Check Order Status	The user selects a current meal order. The system retrieves and displays time remaining.
Show Order Confirmation	The user selects a current meal order. The system retrieves and displays the confirmation code.

Delete Order	The user selects the option to cancel their order. The system removes the order record, along with the meal order record(s) associating the meals the user purchased to the order.

Use Case Diagram



CRUD Matrix

The Store table will hold all the information regarding grocery stores. The customer would never create or modify rows in this table (ie. the customer would never define or modify store data). They would only read from this table (ie. view the stores).

The Allergy table will hold all the information regarding food allergies that our system will support. The customer would not create or modify existing allergies. They would only read from this table.

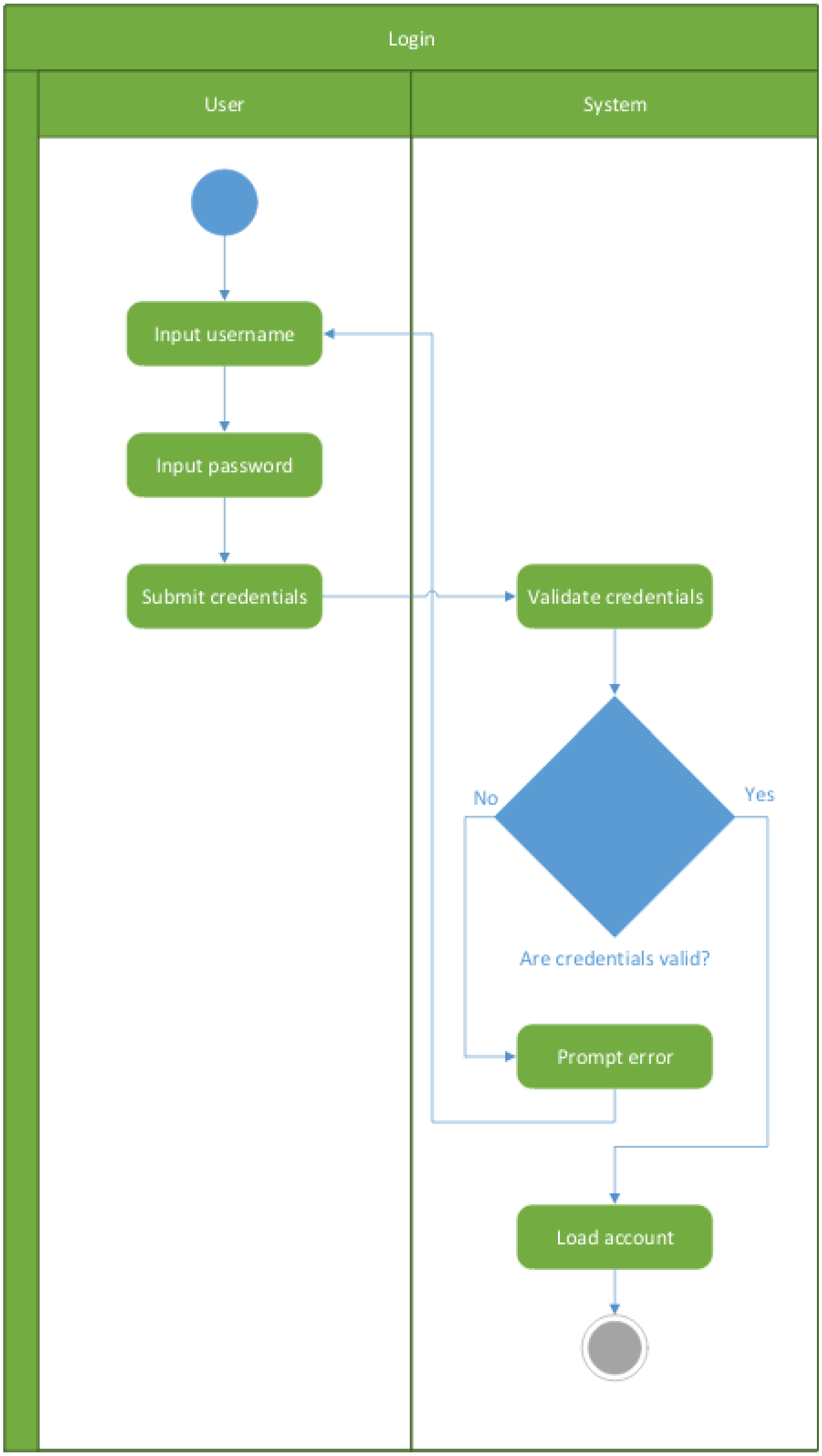
The Meal table and its related tables - Recipe, Nutrition Facts, Ingredient, Meal Allergy - are only read by the customer. The customer does not add or define their own meals.

	Customer	Store Instance	Store	Allergy Instance	Allergy	Payment	Order	Order Instance	Meal	Recipe	Nutrition Facts	Ingredient	MealAllergy
Login / Logout	R												
Create Account	C	C	R	C	R	C							
Delete Account	D	D		D		D							
View Account Settings	R	R		R		R							
Modify Account Settings	U												
Reset Password	R,U												
Add Payment Option	R					C							
Modify Existing Payment Option	R					U							
Remove Payment Option	R					D							
Add Allergy Preference	R			C,U	R								
Remove Allergy Preference	R			D									

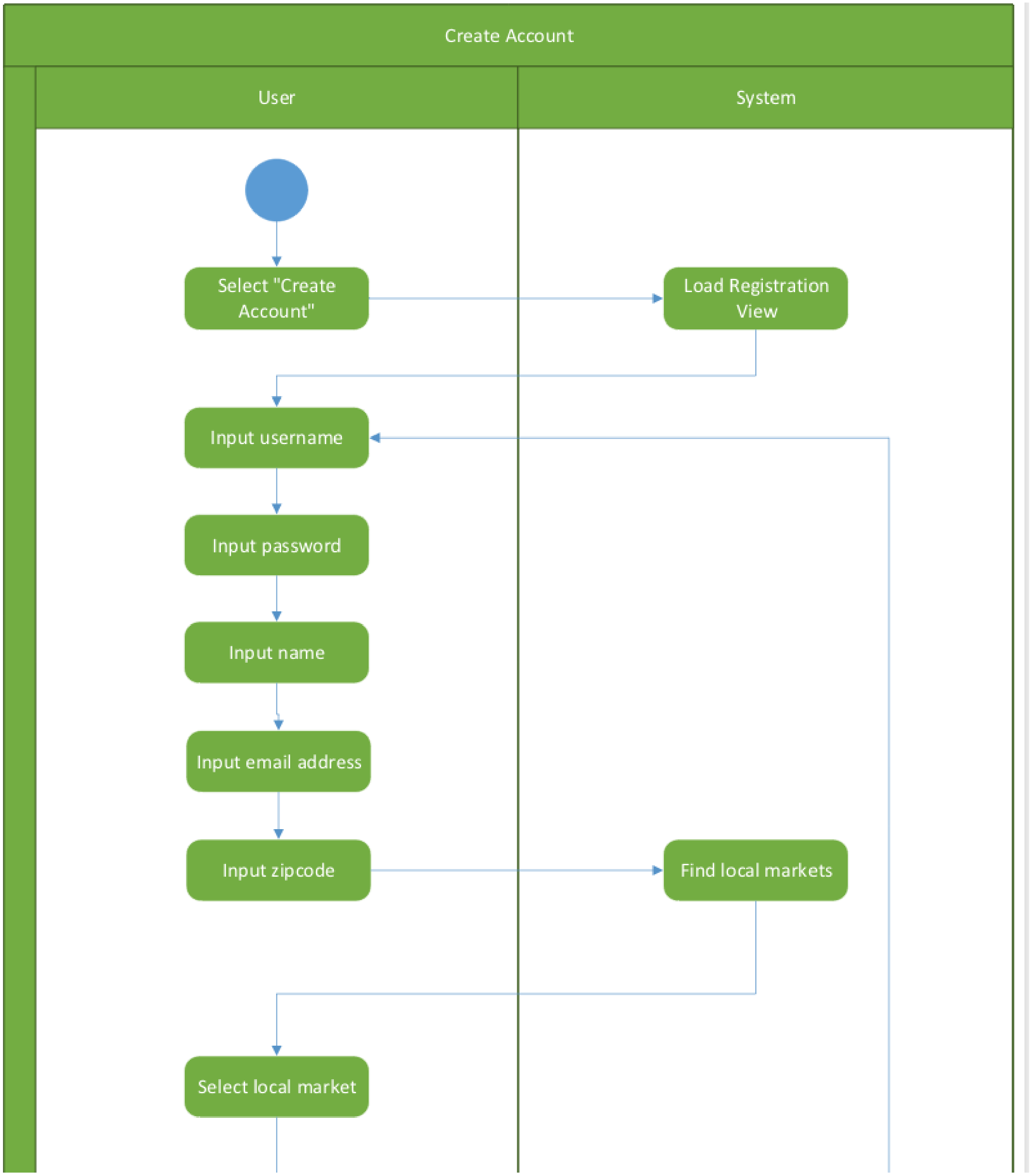
View Past Meal Orders							R	R					
Update Store Preference	R,U	U	R										
Delete Store Preference	R	D											
View meals									R				
View meal recipe										R			
View meal allergies													R
View meal ingredients												R	
View nutrition facts											R		
Add meals to order								C,U	R				
Remove meal from order								D					
Place Meal Order		R				R	C	C					
Check Order Status							R						
Delete Order							D						
Show Order Confirmation							U						

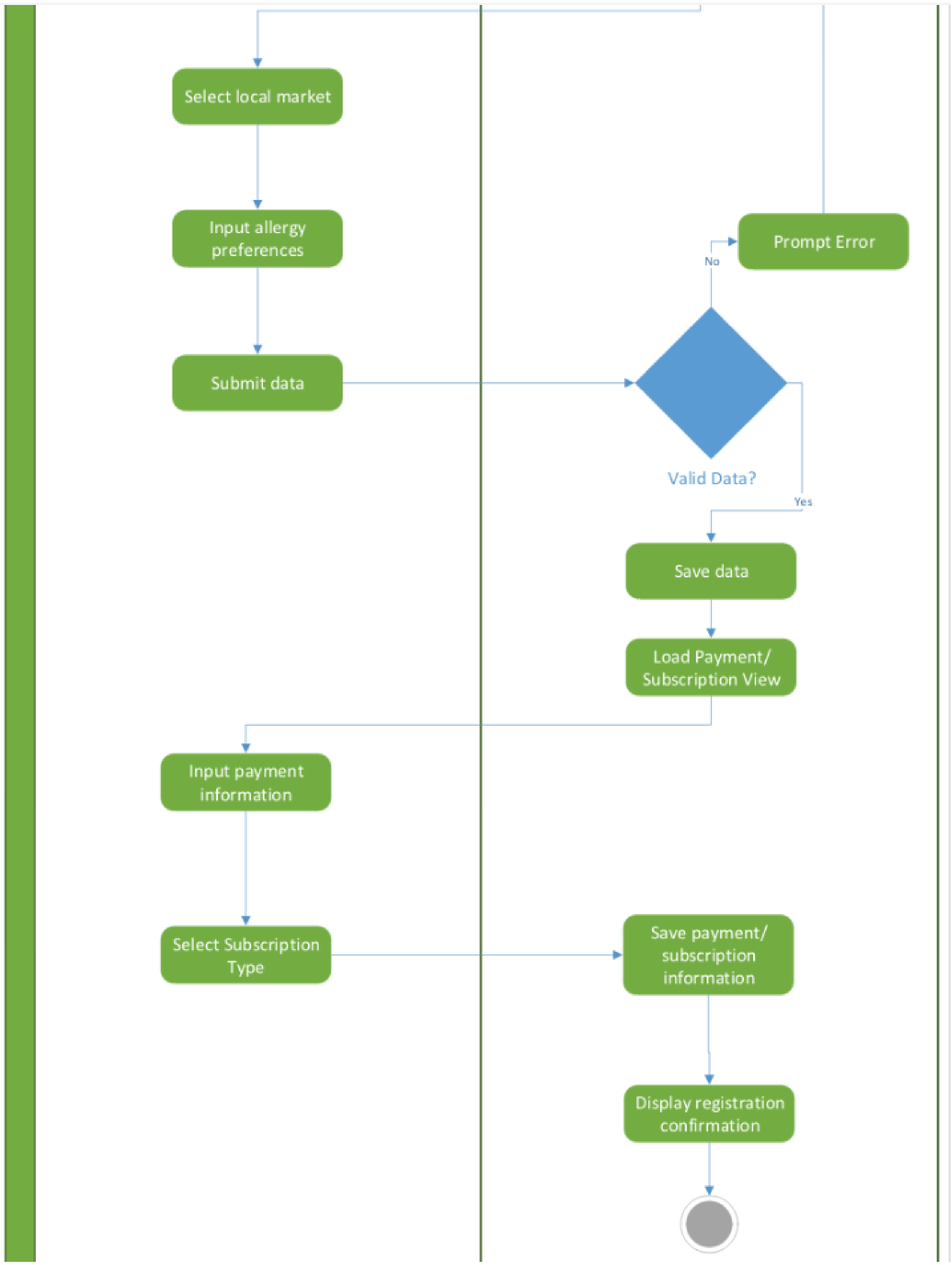
Activity Diagrams

Login

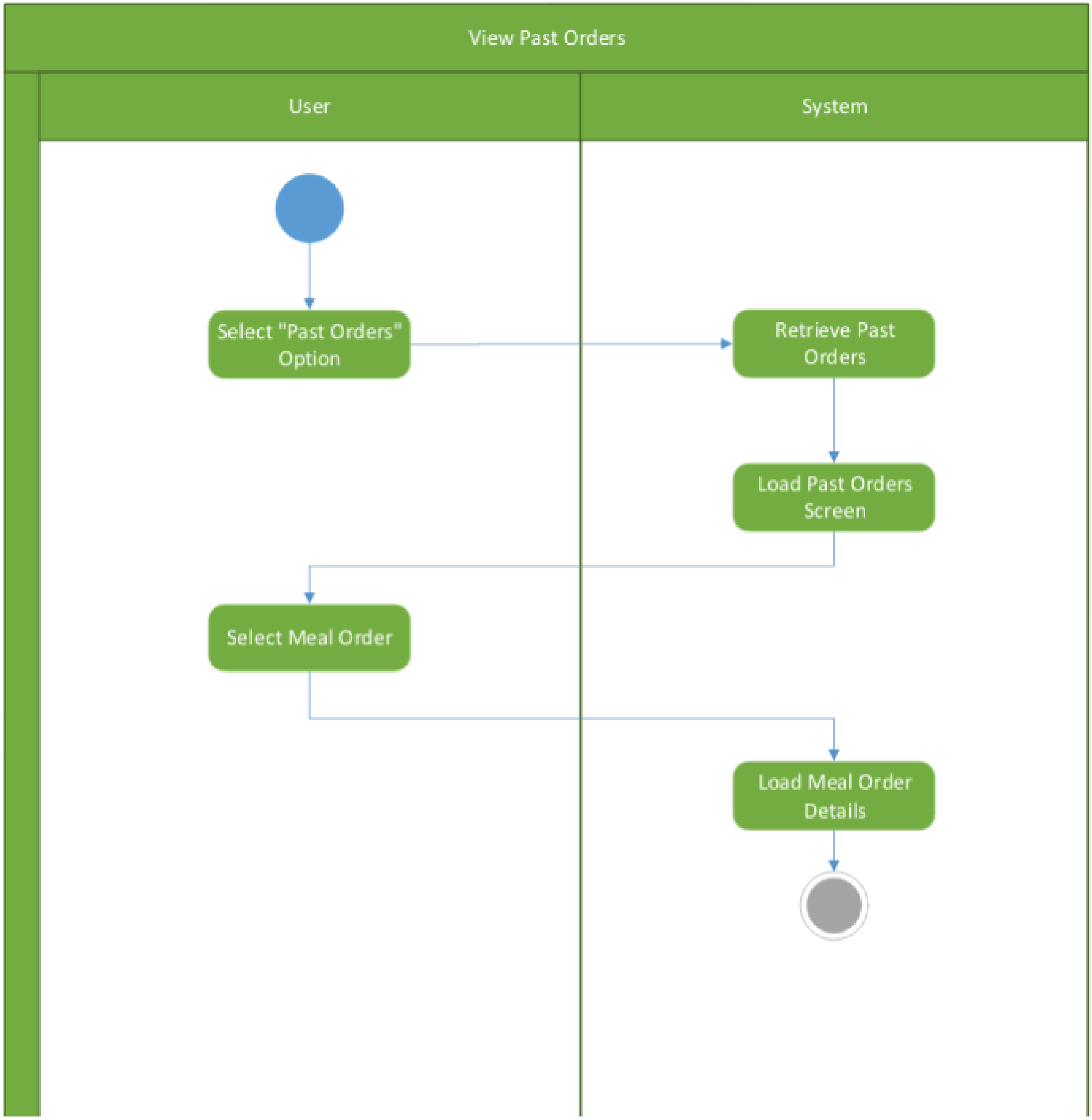


Create Account

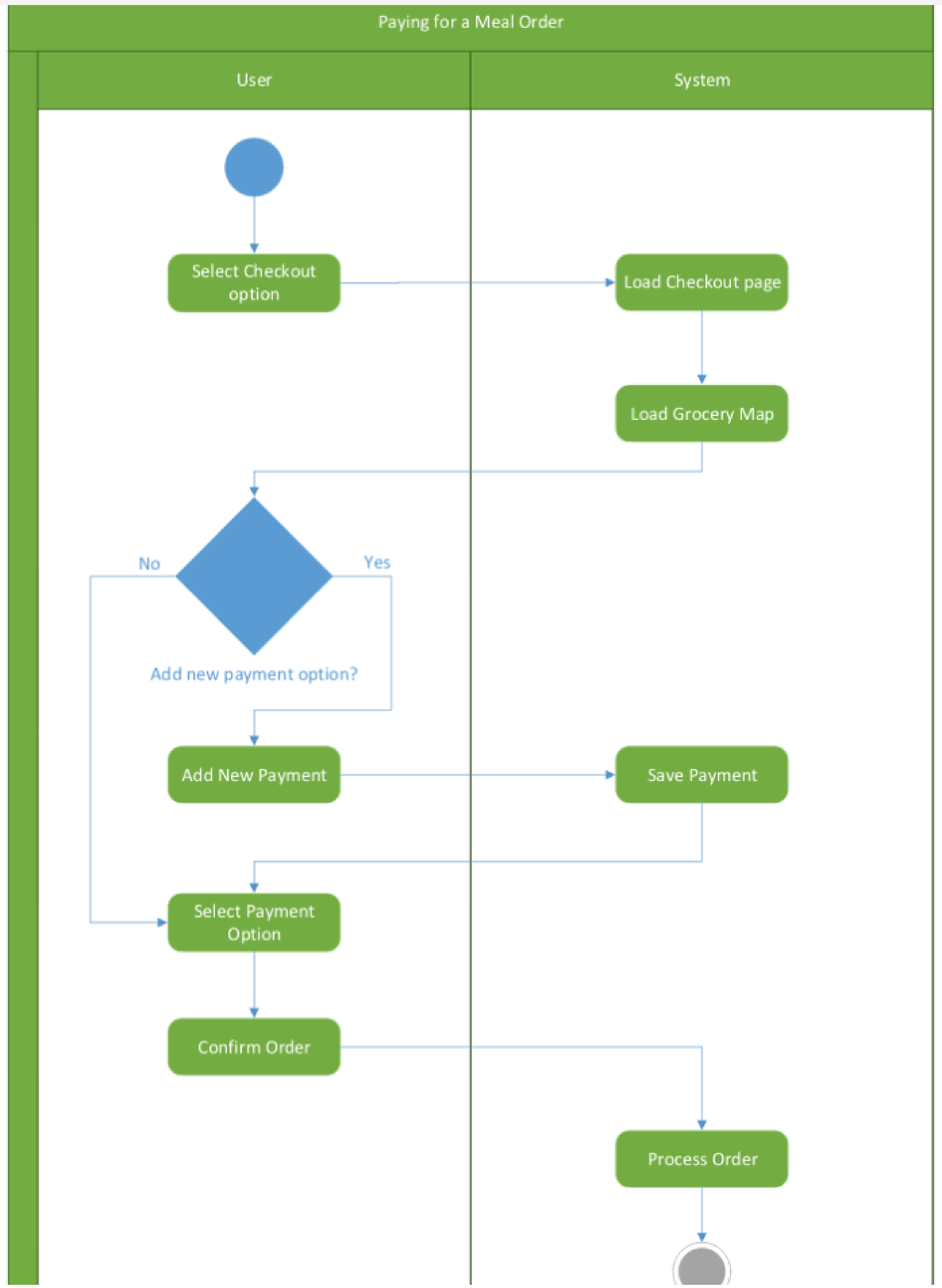



Select local market
Input allergy preferences
Submit data
Prompt Error
No
Valid Data?
Yes
Save data
Load Payment/ Subscription View
Input payment information
Select Subscription Type
Save payment/ subscription information
Display registration confirmation

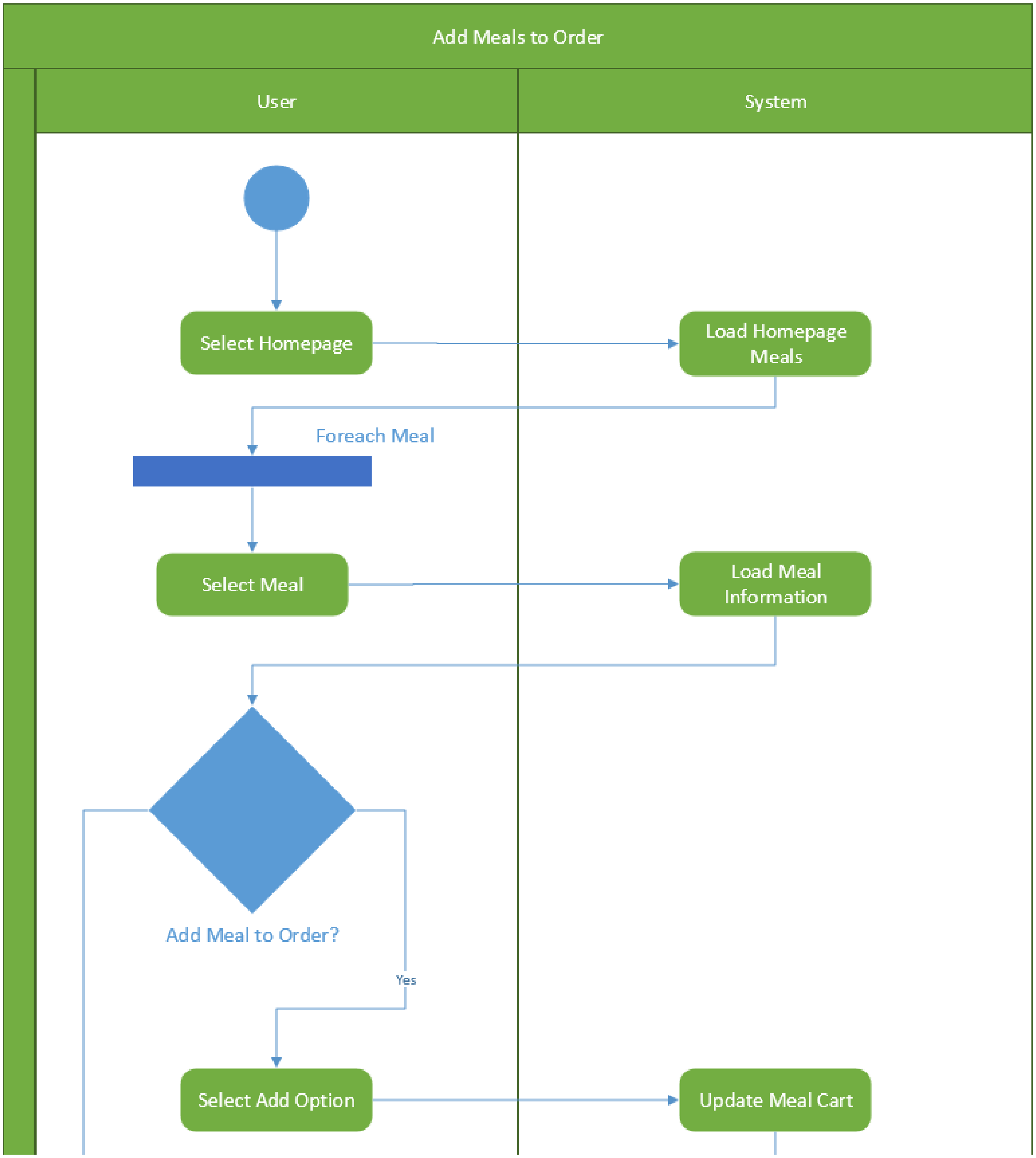
View Past Orders

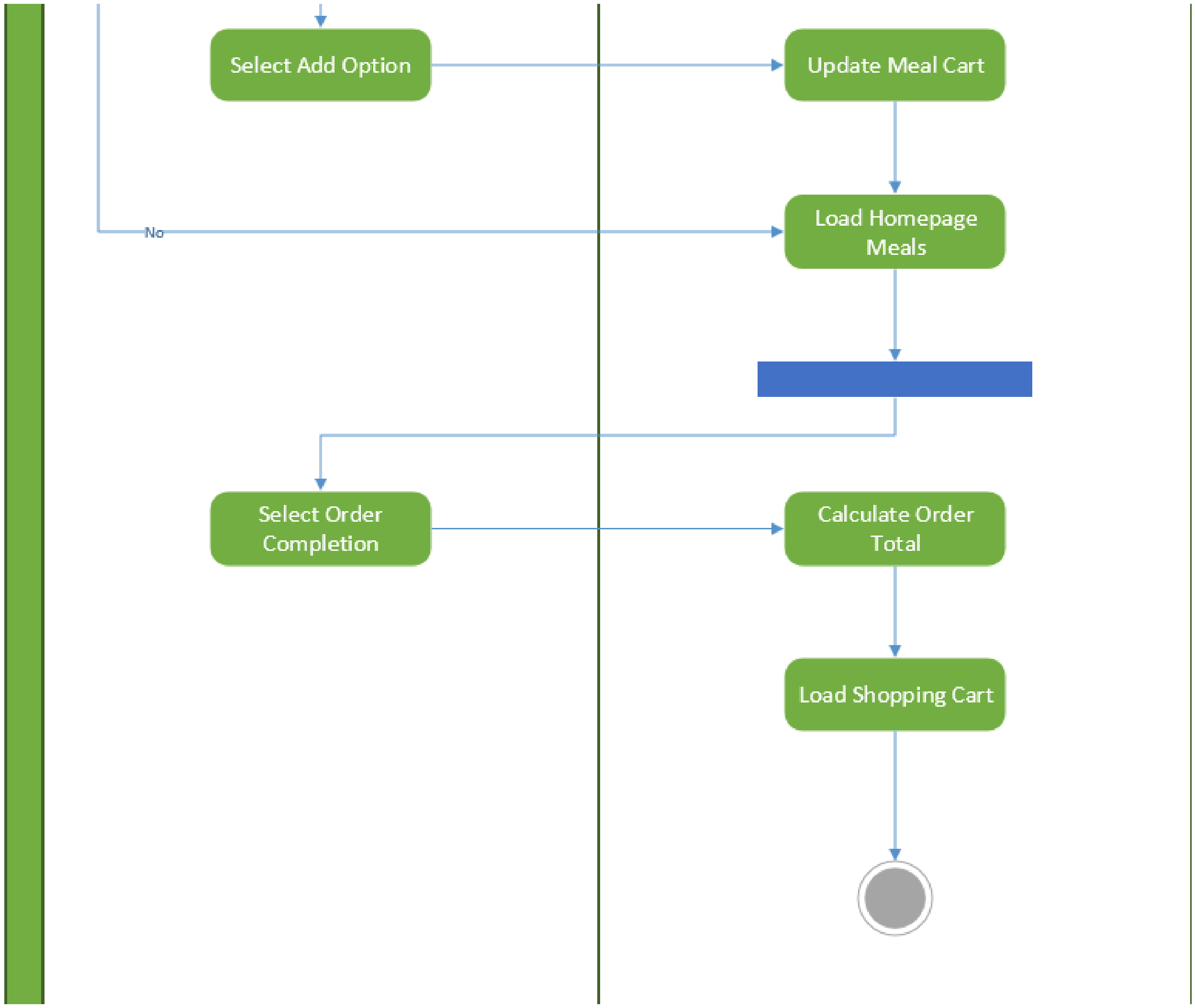


Paying For A Meal



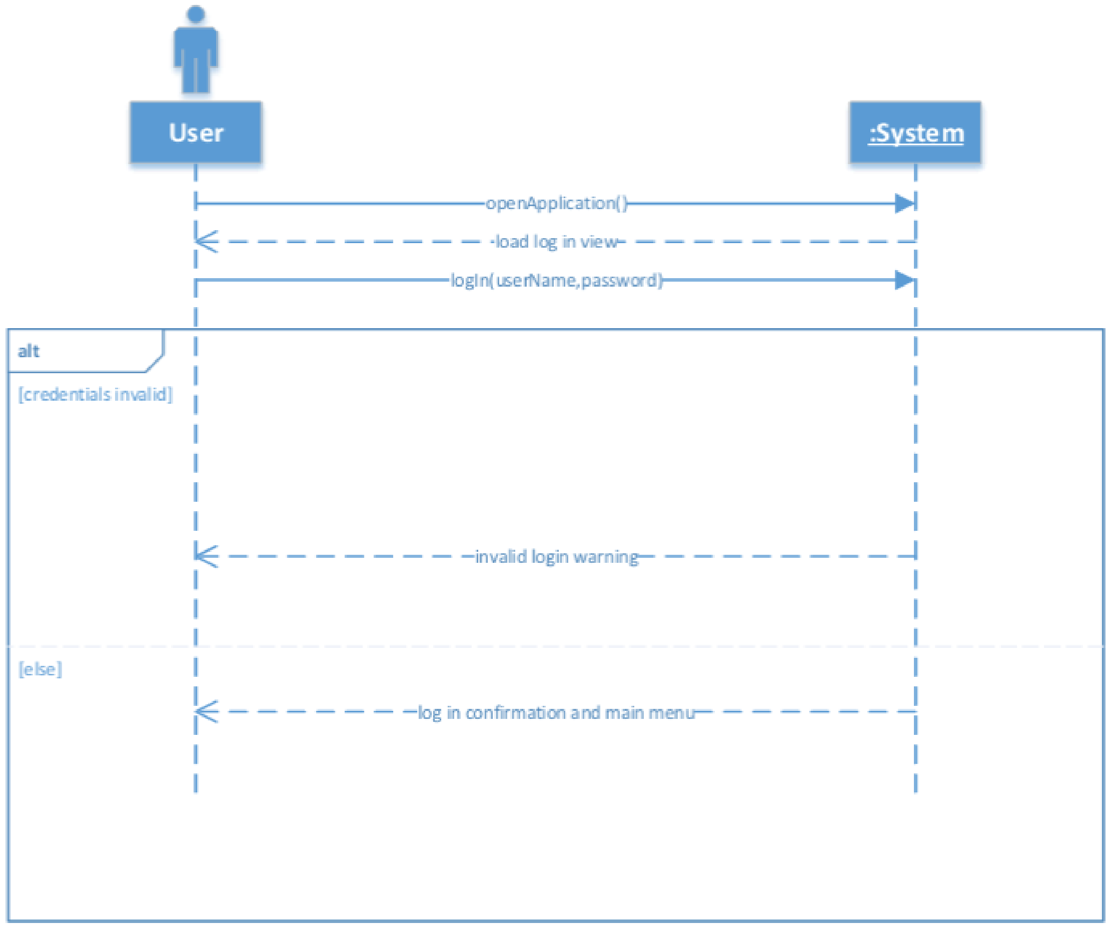
Add Meals to Order



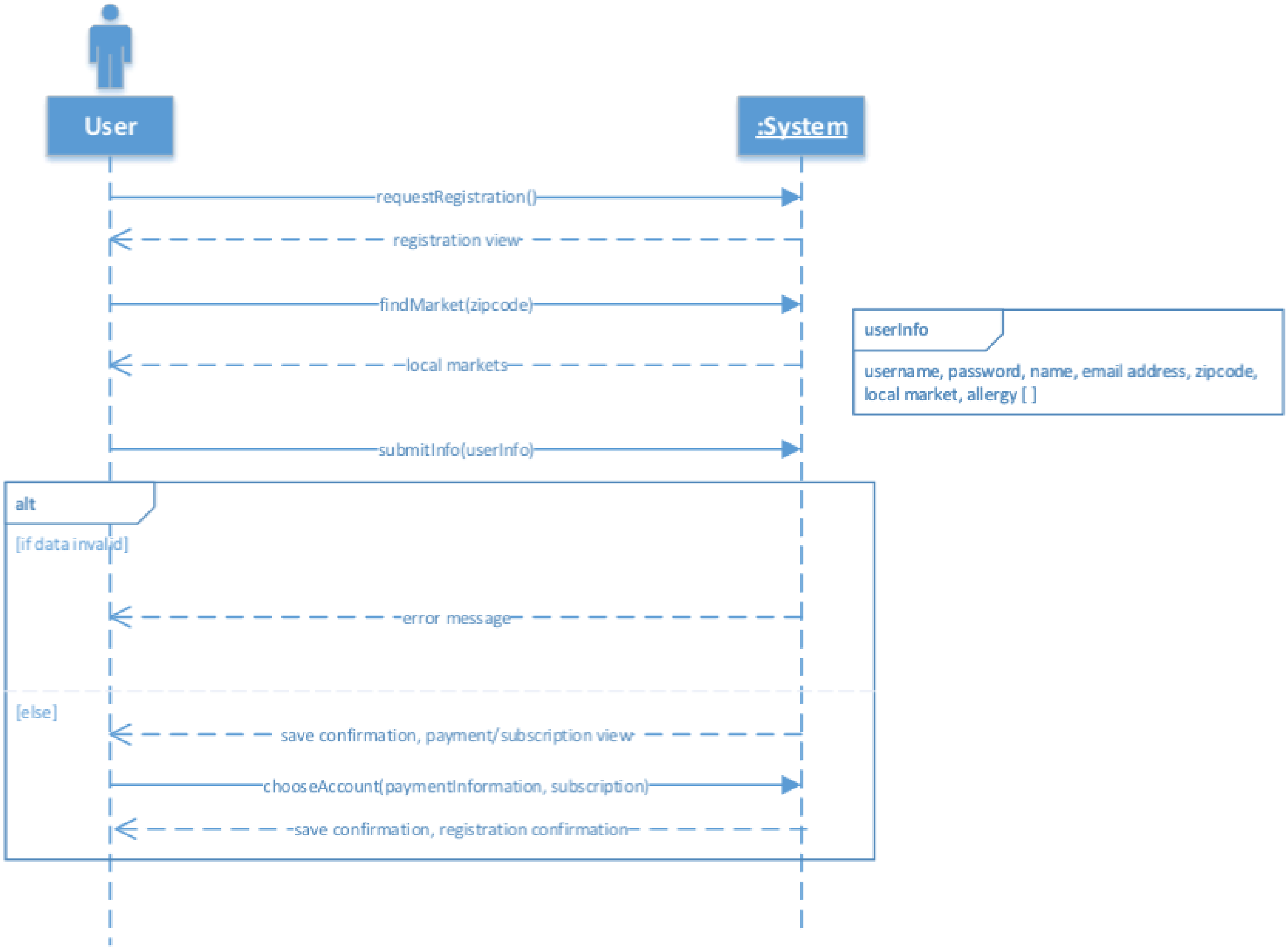

Select Add Option
Update Meal Cart
No
Load Homepage Meals
Select Order Completion
Calculate Order Total
Load Shopping Cart

System Sequence Diagrams

Login

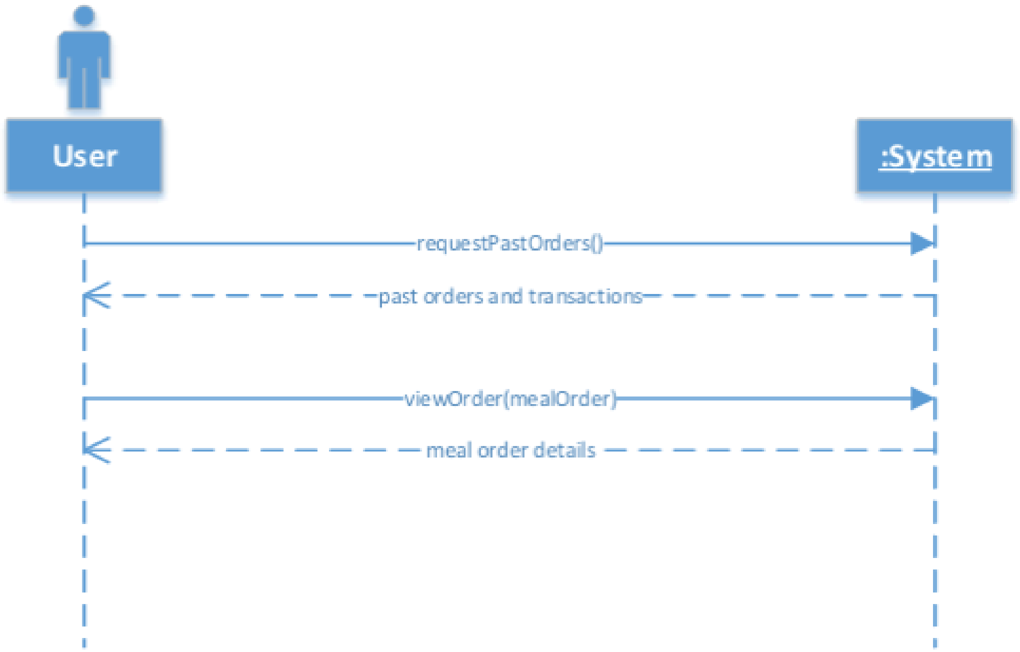


Create Account

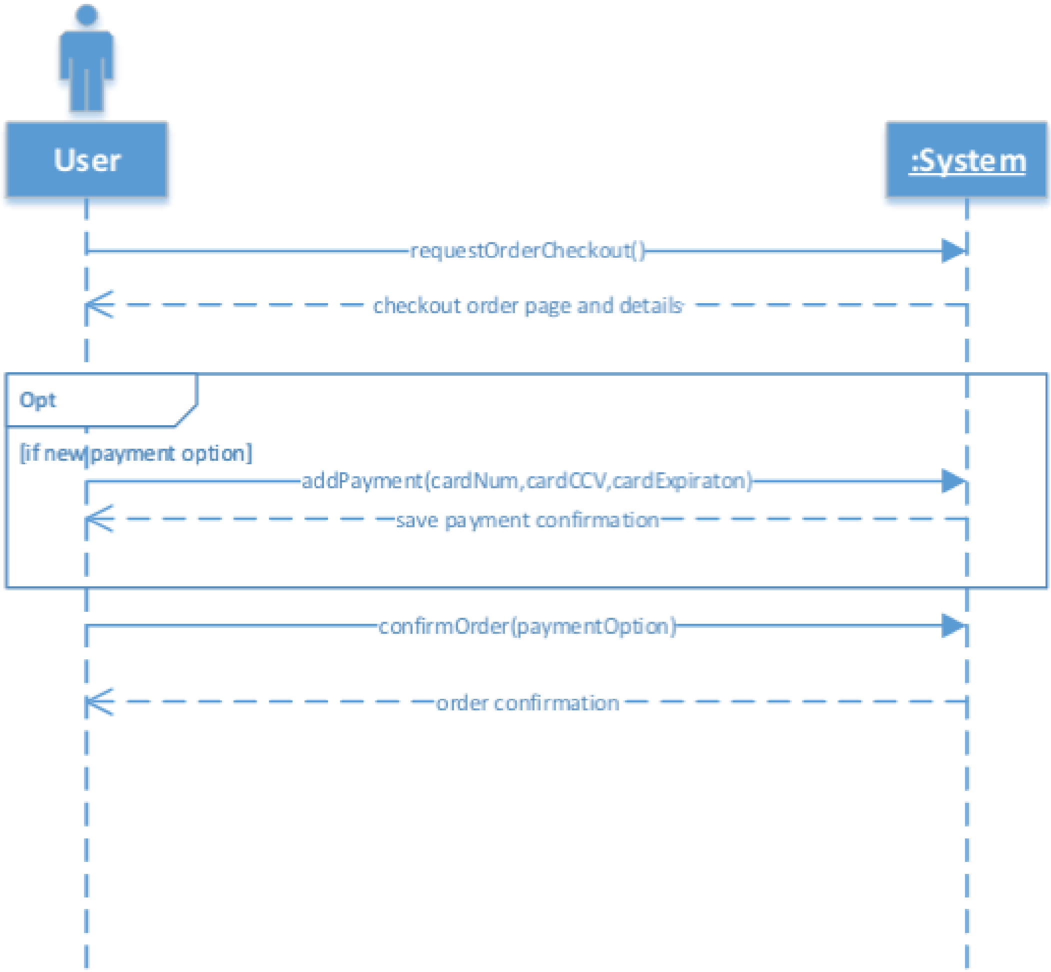


User
:System
requestRegistration()
registration view
findMarket(zipcode)
local markets
userInfo
username, password, name, email address, zipcode, local market, allergy [ ]
submitInfo(userInfo)
alt
[if data invalid]
error message
[else]
save confirmation, payment/subscription view
chooseAccount(paymentInformation, subscription)
save confirmation, registration confirmation

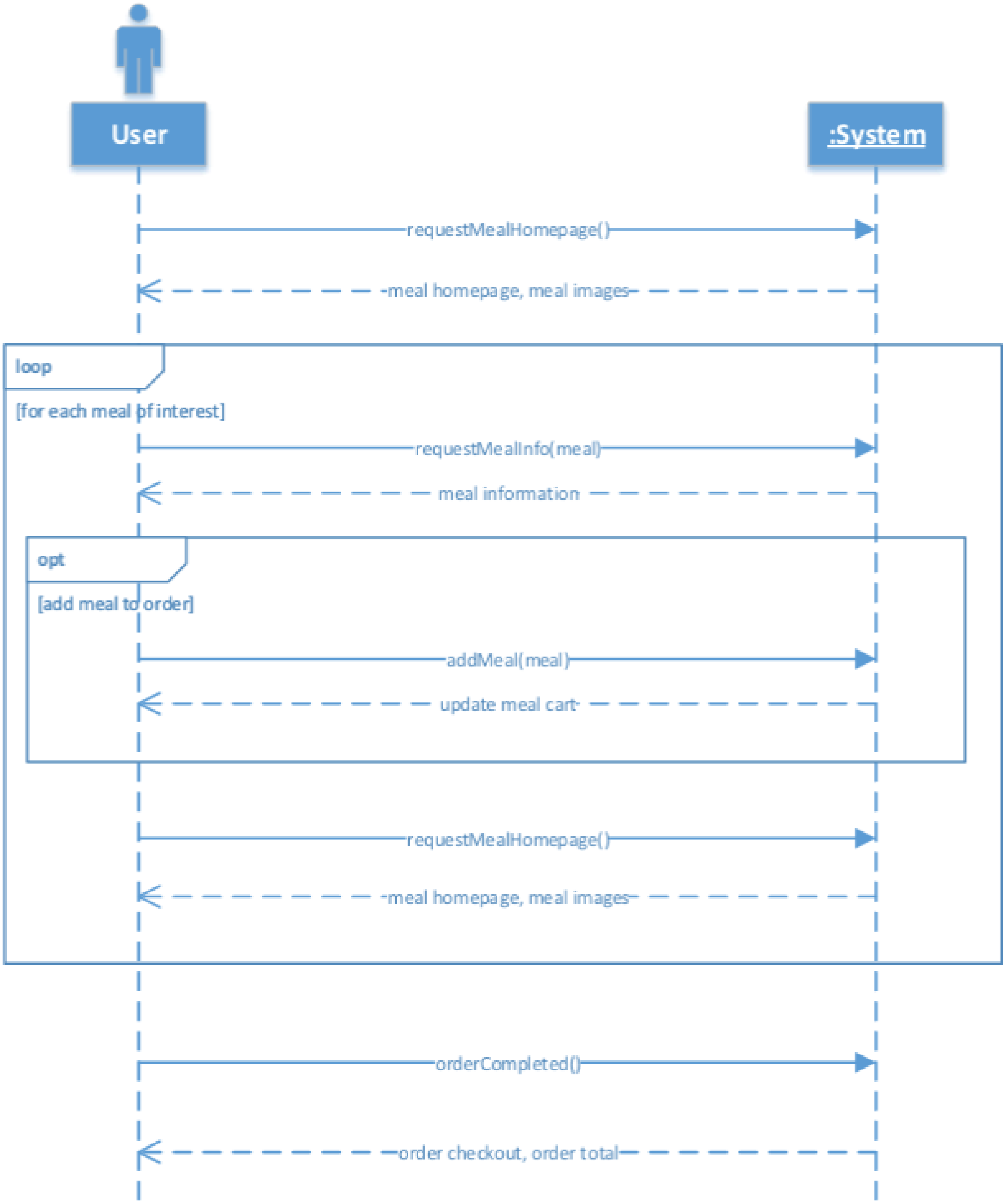
View Past Orders



Paying For A Meal


User
:System
requestOrderCheckout()
checkout order page and details
Opt
[if new payment option]
addPayment(cardNum,cardCCV,cardExpiraton)
save payment confirmation
confirmOrder(paymentOption)
order confirmation

Add Meals to Order



Use Case Scenarios

<table>
<tr><td><b>Use case name:</b></td><td><i>Login</i></td></tr>
<tr><td><b>Scenario:</b></td><td>Login to access the system</td></tr>
<tr><td><b>Triggering event:</b></td><td>Customer wants to login.</td></tr>
<tr><td><b>Brief description:</b></td><td>The customer inputs their login credentials so that they can access the system</td></tr>
<tr><td><b>Actors:</b></td><td>Customer</td></tr>
<tr><td><b>Related use cases:</b></td><td>Create Account</td></tr>
<tr><td><b>Stakeholders:</b></td><td>Customer</td></tr>
<tr><td><b>Preconditions:</b></td><td>The customer has already downloaded the Cook It! Application.
The customer has already created an account.</td></tr>
<tr><td><b>Postconditions:</b></td><td>The customer gains entry to the system.</td></tr>
<tr><td><b>Flow of activities:</b></td><td><table>
<tr><th>Actor</th><th>System</th></tr>
<tr><td>1. Customer inputs their username and passwords and submits it.</td><td>1.1 System validates the username and password.
1.2 System loads customer account.</td></tr>
</table></td></tr>
<tr><td><b>Exception conditions:</b></td><td><b>1.1</b> Customer username or password are invalid.</td></tr>
</table>

<table>
<tr><td><b>Use case name:</b></td><td><i>Create Account</i></td></tr>
<tr><td><b>Scenario:</b></td><td>Create an account for the customer</td></tr>
<tr><td><b>Triggering event:</b></td><td>The customer wants to create an account</td></tr>
<tr><td><b>Brief description:</b></td><td>Customer creates an account by entering basic registration information, payment information, and selecting a payment type.</td></tr>
<tr><td><b>Actors:</b></td><td>Customer</td></tr>
<tr><td><b>Related use cases:</b></td><td>None.</td></tr>
<tr><td><b>Stakeholders:</b></td><td>Customer</td></tr>
<tr><td><b>Preconditions:</b></td><td>The customer has already downloaded the Cook It! Application.
The customer has already created an account.</td></tr>
<tr><td><b>Postconditions:</b></td><td>A customer account is created.</td></tr>
<tr><td><b>Flow of activities:</b></td><td>
<table>
<tr><th>Actor</th><th>System</th></tr>
<tr><td>1. Customer selects the option to create an account on the main page</td><td>1.1. System returns “Create Account” page</td></tr>
<tr><td>2. User fills in form with all the required information and submits it</td><td>2.1. System verifies all information required is entered</td></tr>
<tr><td>3. User enters credit card payment information and selects subscription account type</td><td>3.1. System verifies that credit card information entered is valid and waits for confirmation from credit card issuer
3.2. System receives confirmation from credit card server and creates payment information for user</td></tr>
<tr><td>4. User selects to sign up</td><td>4.1 System sends confirmation link via email</td></tr>
<tr><td></td><td></td></tr>
</table>
</td></tr>
<tr><td><b>Exception conditions:</b></td><td>2.1 System detects missing or incomplete information; returns an error and highlights areas that need attention.</td></tr>
</table>

	3.1. User does not have a valid credit card information; system prompts user to enter new credit card information

<table>
<tr><td><b>Use case name:</b></td><td><i>Add Meals to Order</i></td></tr>
<tr><td><b>Scenario:</b></td><td>Adding Meals that will be part of an order</td></tr>
<tr><td><b>Triggering event:</b></td><td>Customer is browsing a meal and they have selected to add it to the current order</td></tr>
<tr><td><b>Brief description:</b></td><td>After the user has logged in, they are able to see the different food items available to them. The user can select a meal to see more in-depth information about it such as Nutrition Facts and Recipe Instructions. After selecting the meal, the user has the option to add the meal to their order. The user is then prompted to answer if they want to continue browsing meals.</td></tr>
<tr><td><b>Actors:</b></td><td>Customer</td></tr>
<tr><td><b>Related use cases:</b></td><td>Paying For Meal, Place Meal Order</td></tr>
<tr><td><b>Stakeholders:</b></td><td>Customer</td></tr>
<tr><td><b>Preconditions:</b></td><td>The customer is browsing a meal.</td></tr>
<tr><td><b>Postconditions:</b></td><td>The meal is added to the current order.</td></tr>
<tr><td><b>Flow of activities:</b></td><td>
<table>
<tr><th>Actor</th><th>System</th></tr>
<tr><td>1. Select Homepage</td><td>1.1 Load Homepage Meals</td></tr>
<tr><td>2. Select Meal.</td><td>2.1 Load Meal Information</td></tr>
<tr><td>3. Add Meal to Order</td><td></td></tr>
<tr><td>4. Select Add Option</td><td>4.1 Update Meal Cart
4.2 Load Homepage Meals</td></tr>
<tr><td>5. Select Order Completion</td><td>5.1 Calculate Order Total
5.2 Load Shopping Cart</td></tr>
</table>
</td></tr>
<tr><td><b>Exception conditions:</b></td><td>4.1 Customer selects not to add more meals.</td></tr>
</table>

<table>
<tr><td><b>Use case name:</b></td><td><i>View Past Orders</i></td></tr>
<tr><td><b>Scenario:</b></td><td>Viewing an order that the customer previously completed</td></tr>
<tr><td><b>Triggering event:</b></td><td>Customer wants to view an order that they previously made</td></tr>
<tr><td><b>Brief description:</b></td><td>Customer wants to view an order from their past. The customer selects a particular order to see more details about it</td></tr>
<tr><td><b>Actors:</b></td><td>Customer</td></tr>
<tr><td><b>Related use cases:</b></td><td>None.</td></tr>
<tr><td><b>Stakeholders:</b></td><td>Customer</td></tr>
<tr><td><b>Preconditions:</b></td><td>Customer previously made an order to see order details.</td></tr>
<tr><td><b>Postconditions:</b></td><td>The system displays order details for a particular order</td></tr>
<tr><td><b>Flow of activities:</b></td><td><table>
<tr><th>Actor</th><th>System</th></tr>
<tr><td>1. Select Past Orders
2. Select Order</td><td>1.1 Load Past Orders Page
2.1 Load Meal Order Details</td></tr>
</table></td></tr>
<tr><td><b>Exception conditions:</b></td><td><b>2.1</b> The customer has not placed any orders</td></tr>
</table>

<table>
<tr><td><b>Use case name:</b></td><td><i>Paying For Meal</i></td></tr>
<tr><td><b>Scenario:</b></td><td>The customer has finalized their shopping cart and are going to pay for their meal in the checkout.</td></tr>
<tr><td><b>Triggering event:</b></td><td>The customer wants to finalize payment for meal.</td></tr>
<tr><td><b>Brief description:</b></td><td>Customer confirms payment information and location of pickup, and pays for the meal.</td></tr>
<tr><td><b>Actors:</b></td><td>Customer</td></tr>
<tr><td><b>Related use cases:</b></td><td>Add Meals to Order, Place Meal Order</td></tr>
<tr><td><b>Stakeholders:</b></td><td>Customer</td></tr>
<tr><td><b>Preconditions:</b></td><td>Customer inserts credit card information and finalizes store location.</td></tr>
<tr><td><b>Postconditions:</b></td><td>Meal(s) have been bought and estimated wait time is initiated to designated grocery store.</td></tr>
<tr><td><b>Flow of activities:</b></td><td><table>
<tr><th>Actor</th><th>System</th></tr>
<tr><td>1. Select checkout option</td><td>1.1 Load checkout page
1.2 Load grocery map</td></tr>
<tr><td>2. Add new payment/Select current payment</td><td>2.1 Save payment</td></tr>
<tr><td>3. Confirm order</td><td>3.1 Process order</td></tr>
</table></td></tr>
<tr><td><b>Exception conditions:</b></td><td>Credit card number is invalid.</td></tr>
</table>

Domain Model Class Diagram

Table Definitions

Customer - An individual user of the Cook It! Application
Allergy - A medical food aversion type that an individual may have
Store - A location that an individual selects to pick up their food.
Order - A request for food that an individual selects.
Meal - A combination of food that is part of an Order that an individual selects. (Synonymous with Entree)
Nutrition Facts - A set of metrics detailing how healthy a particular meal is
Recipe - A list of instructions detailing out how a meal is prepared
Ingredient - A single item that is needed to prepare a meal
Payment - Information representing a credit/debit card that allows an individual to purchase something.

Domain Relationships

A Customer has 0 to Many Allergies. An Allergy can be had by 0 to Many Customers. This Many to Many is broken up by the association class AllergyInstance. AllergyInstance will show an allergy that a customer has.

A Customer selects Many Stores. A Store can be selected by Many Customers. This Many to Many is broken up by the association class Store Instance. StoreInstance is used to keep track of the store that a customer has selected.

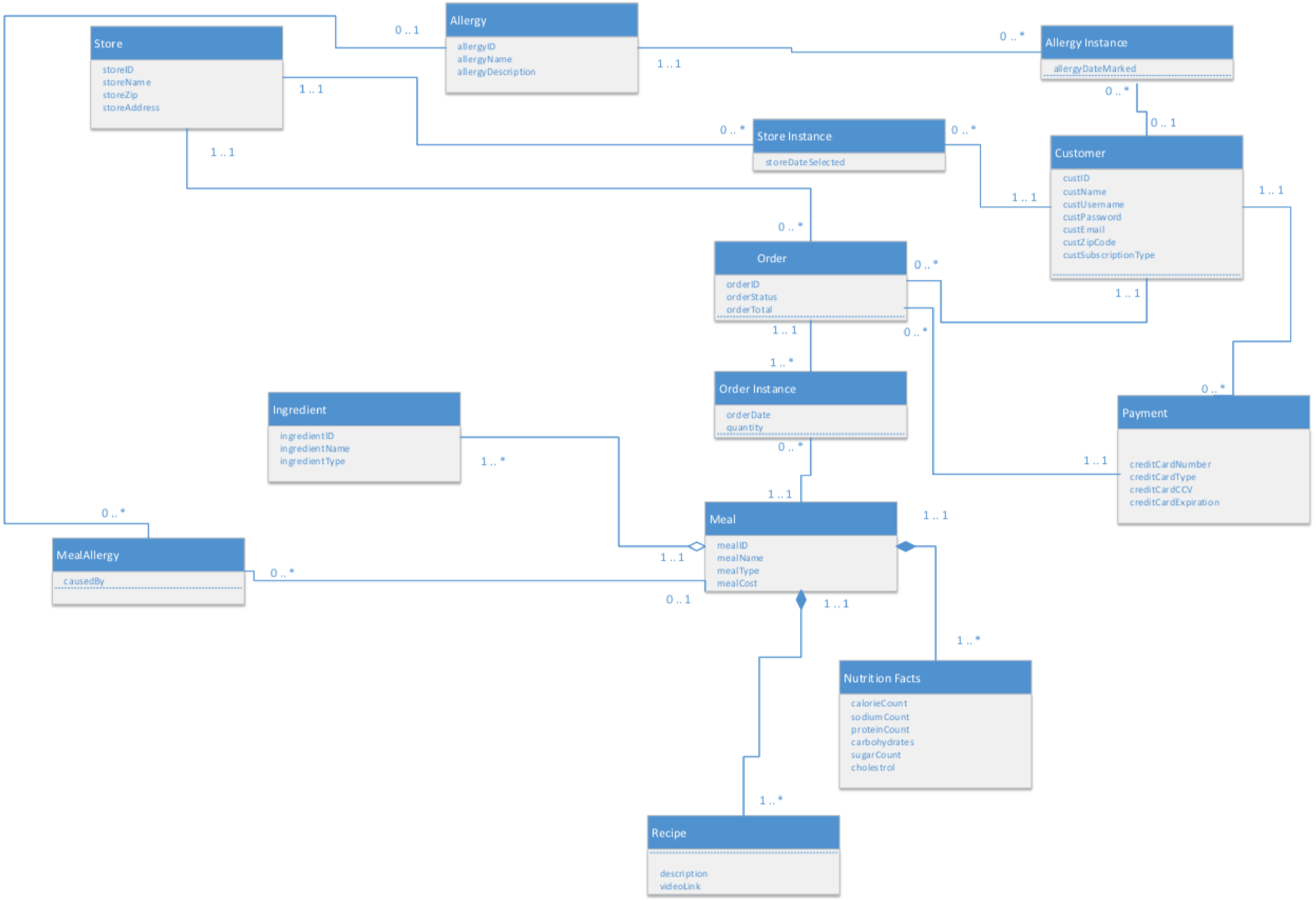
A Customer places 0 to Many Orders. An Order is placed by a single Customer.

A Store handles Many Orders. An Order is handled by a single Store.

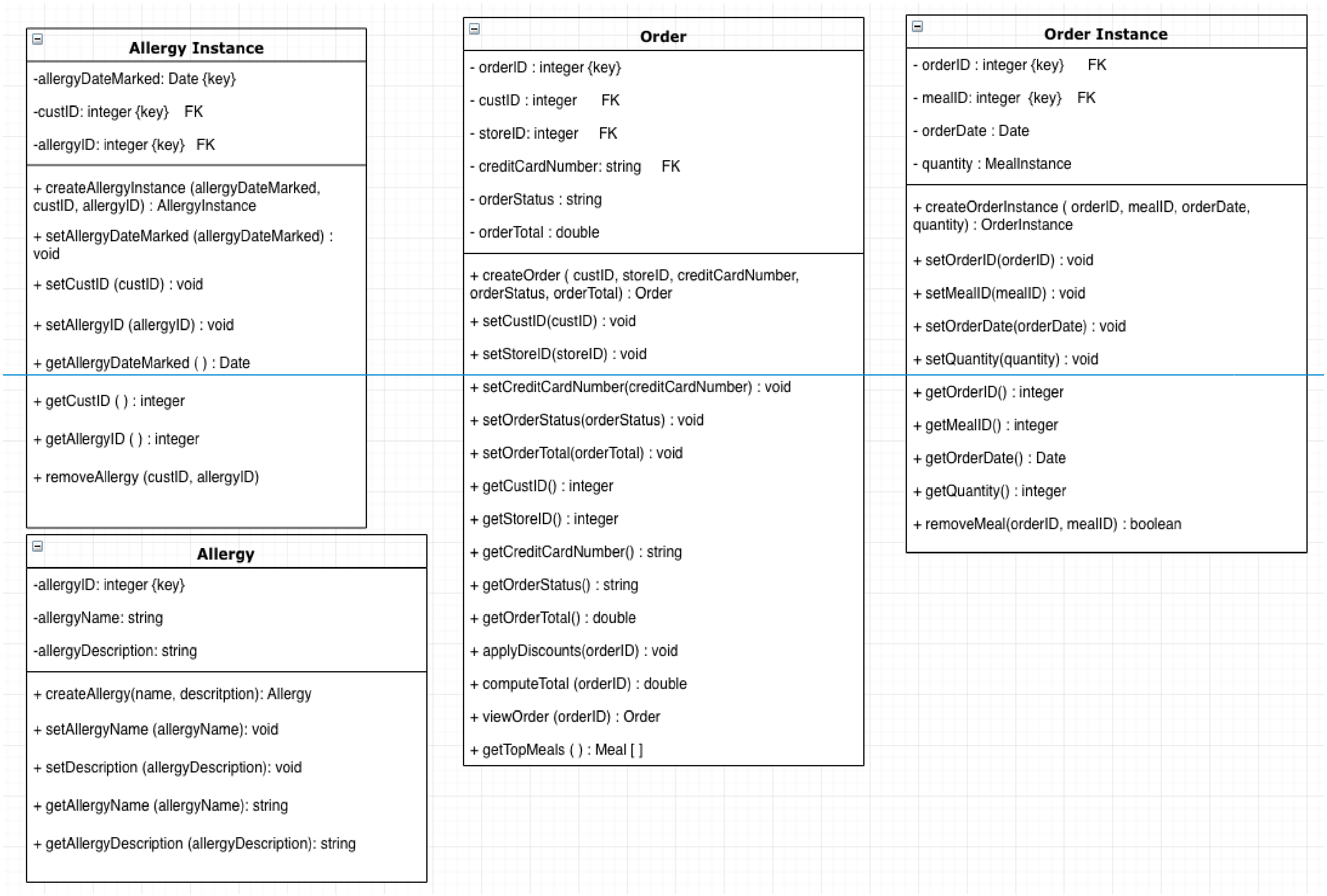
An Order contains 1 to Many Meals. A single Meal can be contained in Many Orders. This Many to Many is broken up by Order Instance. OrderInstance is used to keep track of the Meals that are part of an Order.

A Meal can contain Many Allergies, and an Allergy can be contained in Many Meals. This Many to Many is broken up by MealAllergy. Meal Allergy is used to keep track of the allergies that a certain meal may have.

A Meal has Nutrition Facts and Recipes, where the Nutrition Fact and Recipe cannot exist without the Meal. Therefore this is an example of aggregation.
An Ingredient is part of a Meal, but it can exist separately of the Meal. Therefore, this is an example of composition.


Store
storeID
storeName
storeZip
storeAddress
Allergy
allergyID
allergyName
allergyDescription
Allergy Instance
allergyDateMarked
Store Instance
storeDateSelected
Customer
custID
custName
custUsername
custPassword
custEmail
custZipCode
custSubscriptionType
Order
orderID
orderStatus
orderTotal
Order Instance
orderDate
quantity
Payment
creditCardNumber
creditCardType
creditCardCCV
creditCardExpiration
Ingredient
ingredientID
ingredientName
ingredientType
Meal
mealID
mealName
mealType
mealCost
MealAllergy
causedBy
Nutrition Facts
calorieCount
sodiumCount
proteinCount
carbohydrates
sugarCount
cholestrol
Recipe
description
videoLink
0 .. 1
1 .. 1
0 .. *
1 .. 1
0 .. *
0 .. 1
0 .. *
0 .. *
1 .. 1
1 .. 1
1 .. 1
0 .. *
0 .. *
1 .. 1
1 .. 1
0 .. *
1 .. *
0 .. *
0 .. *
1 .. 1
1 .. *
1 .. 1
1 .. 1
0 .. *
1 .. 1
0 .. *
0 .. 1
1 .. 1
1 .. *
1 .. *

Design Class Diagram



Allergy Instance

-allergyDateMarked: Date {key}

-custID: integer {key} FK

-allergyID: integer {key} FK

+ createAllergyInstance (allergyDateMarked, custID, allergyID) : AllergyInstance

+ setAllergyDateMarked (allergyDateMarked) : void

+ setCustID (custID) : void

+ setAllergyID (allergyID) : void

+ getAllergyDateMarked ( ) : Date

+ getCustID ( ) : integer

+ getAllergyID ( ) : integer

+ removeAllergy (custID, allergyID)

Allergy

-allergyID: integer {key}

-allergyName: string

-allergyDescription: string

+ createAllergy(name, descritption): Allergy

+ setAllergyName (allergyName): void

+ setDescription (allergyDescription): void

+ getAllergyName (allergyName): string

+ getAllergyDescription (allergyDescription): string

Order

- orderID : integer {key}

- custID : integer FK

- storeID: integer FK

- creditCardNumber: string FK

- orderStatus : string

- orderTotal : double

+ createOrder (custID, storeID, creditCardNumber, orderStatus, orderTotal) : Order

+ setCustID(custID) : void

+ setStoreID(storeID) : void

+ setCreditCardNumber(creditCardNumber) : void

+ setOrderStatus(orderStatus) : void

+ setOrderTotal(orderTotal) : void

+ getCustID() : integer

+ getStoreID() : integer

+ getCreditCardNumber() : string

+ getOrderStatus() : string

+ getOrderTotal() : double

+ applyDiscounts(orderID) : void

+ computeTotal (orderID) : double

+ viewOrder (orderID) : Order

+ getTopMeals ( ) : Meal [ ]

Order Instance

- orderID : integer {key} FK

- mealID: integer {key} FK

- orderDate : Date

- quantity : MealInstance

+ createOrderInstance (orderID, mealID, orderDate, quantity) : OrderInstance

+ setOrderID(orderID) : void

+ setMealID(mealID) : void

+ setOrderDate(orderDate) : void

+ setQuantity(quantity) : void

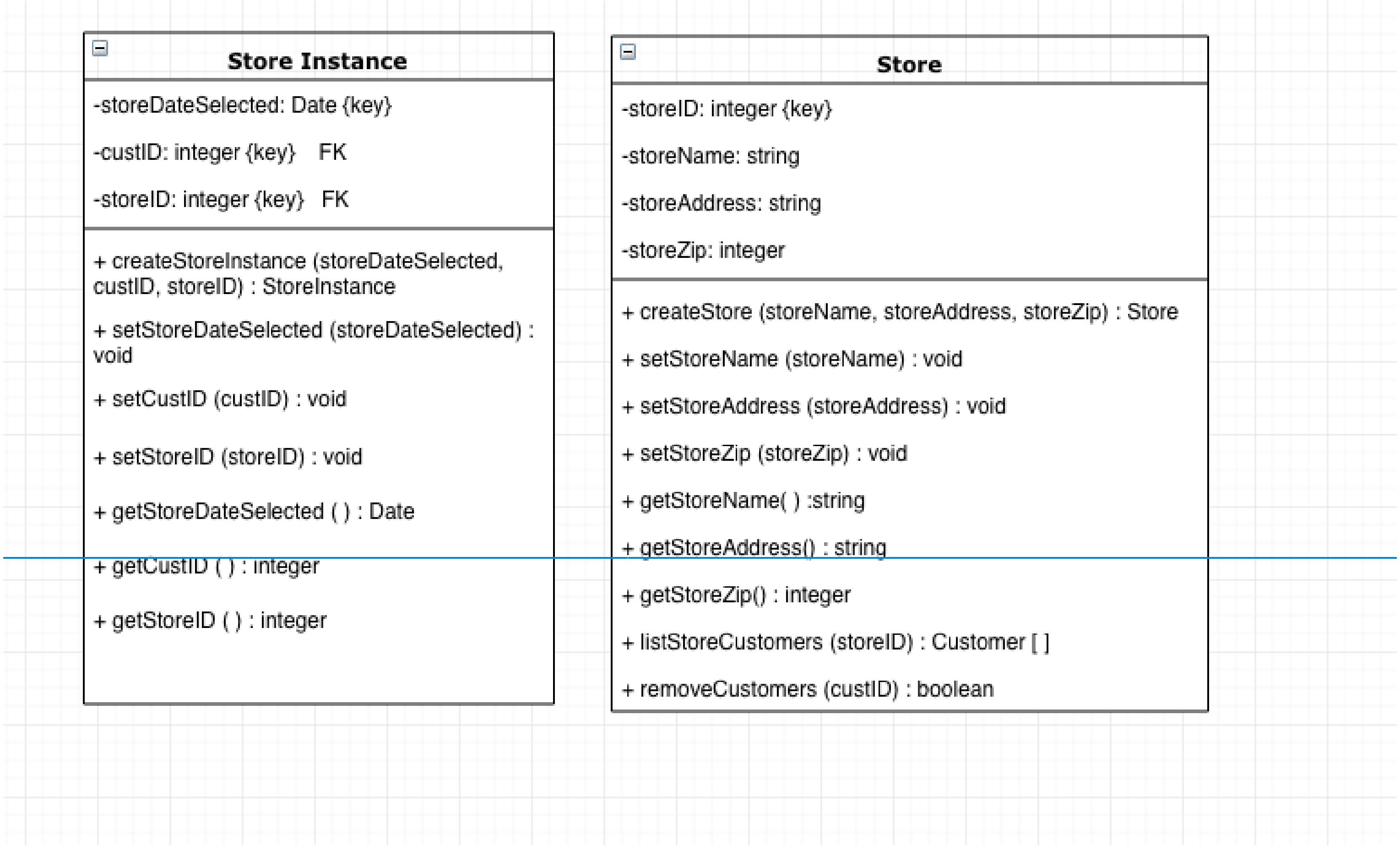
+ getOrderID() : integer

+ getMealID() : integer

+ getOrderDate() : Date

+ getQuantity() : integer

+ removeMeal(orderID, mealID) : boolean



Store Instance

-storeDateSelected: Date {key}

-custID: integer {key}   FK

-storeID: integer {key}   FK

+ createStoreInstance (storeDateSelected, custID, storeID) : StoreInstance

+ setStoreDateSelected (storeDateSelected) : void

+ setCustID (custID) : void

+ setStoreID (storeID) : void

+ getStoreDateSelected ( ) : Date

+ getCustID ( ) : integer

+ getStoreID ( ) : integer

Store

-storeID: integer {key}

-storeName: string

-storeAddress: string

-storeZip: integer

+ createStore (storeName, storeAddress, storeZip) : Store

+ setStoreName (storeName) : void

+ setStoreAddress (storeAddress) : void

+ setStoreZip (storeZip) : void

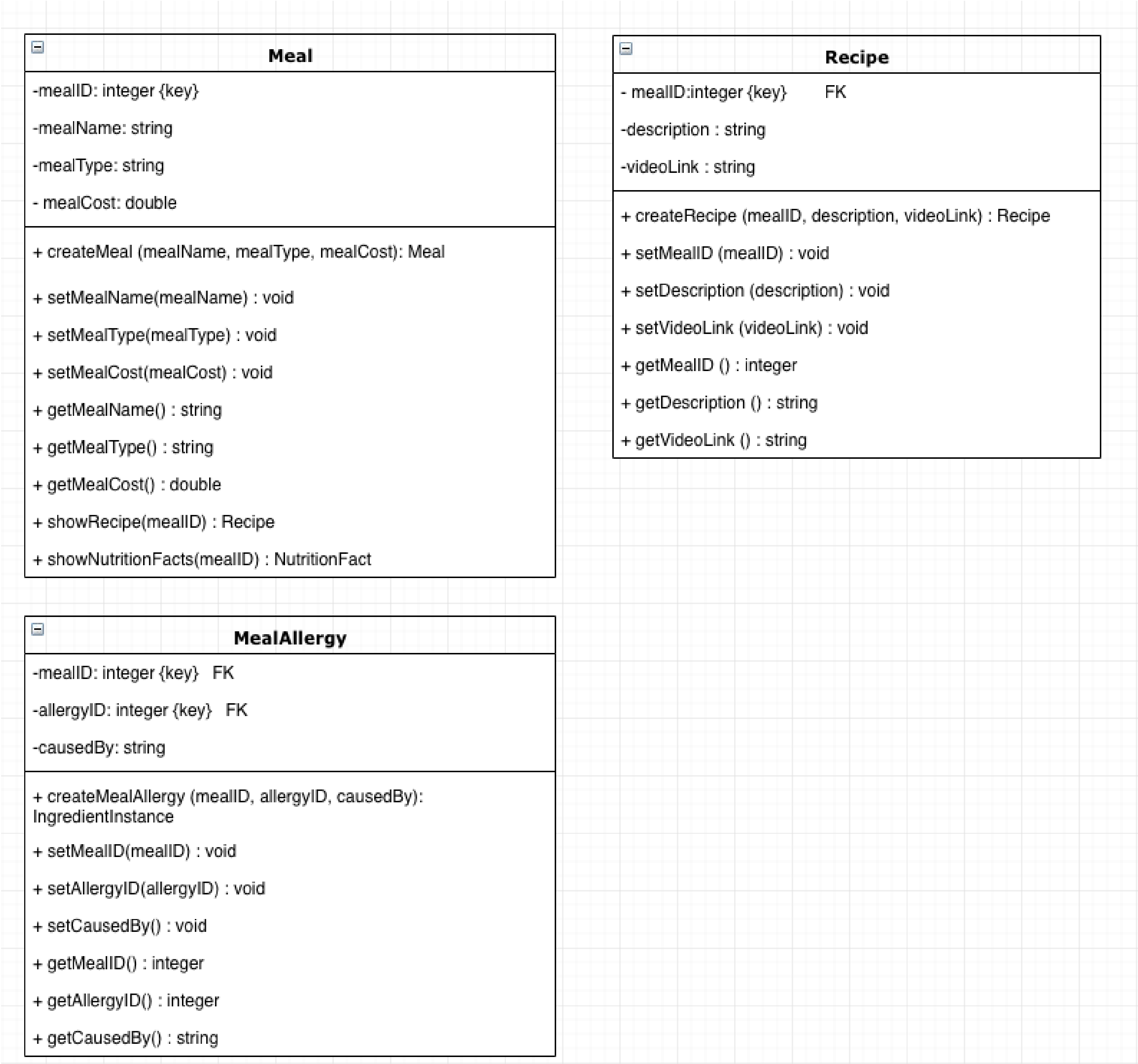
+ getStoreName( ) :string

+ getStoreAddress() : string

+ getStoreZip() : integer

+ listStoreCustomers (storeID) : Customer [ ]

+ removeCustomers (custID) : boolean



Meal

-mealID: integer {key}

-mealName: string

-mealType: string

- mealCost: double

+ createMeal (mealName, mealType, mealCost): Meal

+ setMealName(mealName) : void

+ setMealType(mealType) : void

+ setMealCost(mealCost) : void

+ getMealName() : string

+ getMealType() : string

+ getMealCost() : double

+ showRecipe(mealID) : Recipe

+ showNutritionFacts(mealID) : NutritionFact

Recipe

- mealID:integer {key} FK

-description : string

-videoLink : string

+ createRecipe (mealID, description, videoLink) : Recipe

+ setMealID (mealID) : void

+ setDescription (description) : void

+ setVideoLink (videoLink) : void

+ getMealID () : integer

+ getDescription () : string

+ getVideoLink () : string

MealAllergy

-mealID: integer {key} FK

-allergyID: integer {key} FK

-causedBy: string

+ createMealAllergy (mealID, allergyID, causedBy): IngredientInstance

+ setMealID(mealID) : void

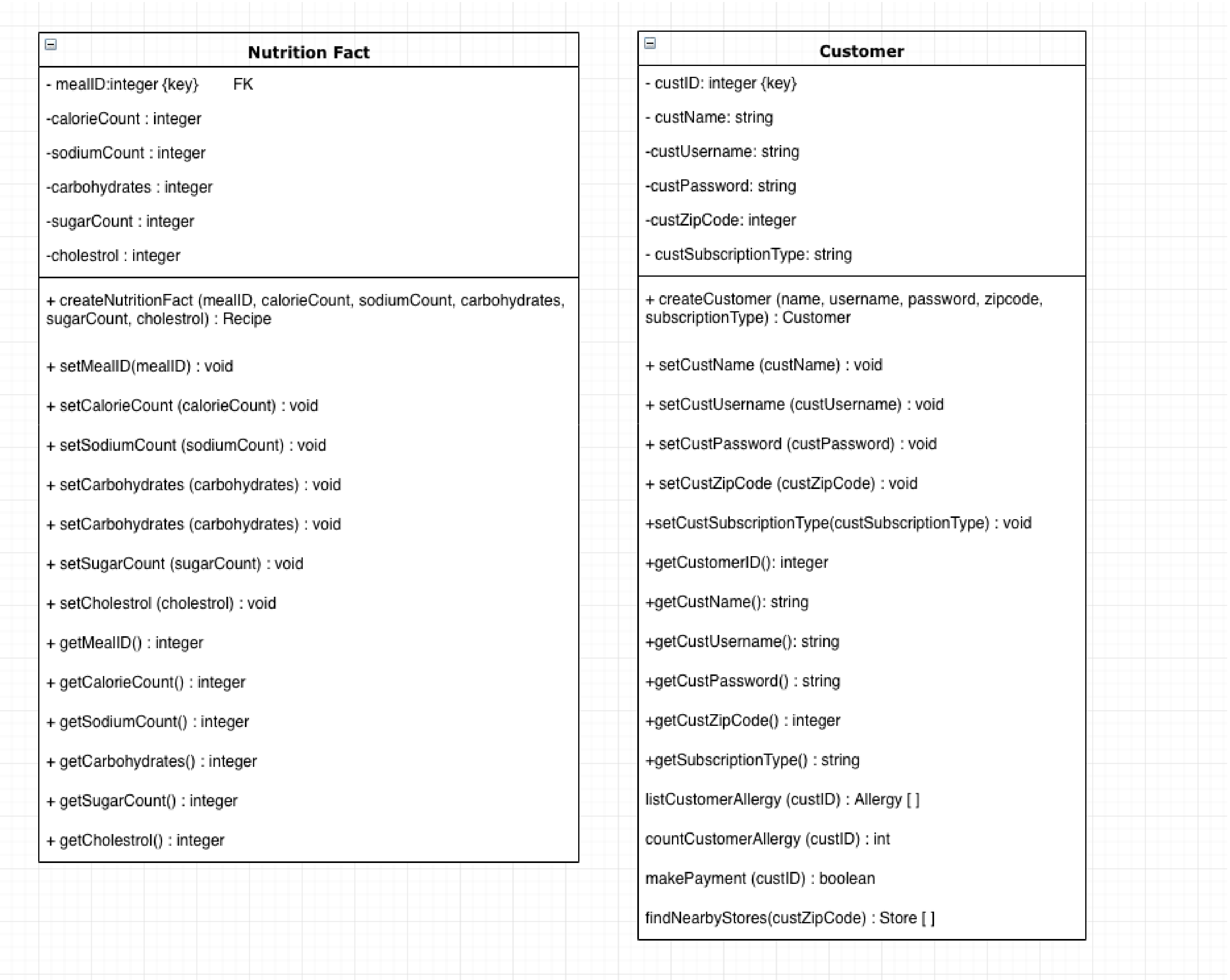
+ setAllergyID(allergyID) : void

+ setCausedBy() : void

+ getMealID() : integer

+ getAllergyID() : integer

+ getCausedBy() : string



Nutrition Fact

- mealID:integer {key}   FK

-calorieCount : integer

-sodiumCount : integer

-carbohydrates : integer

-sugarCount : integer

-cholestrol : integer

+ createNutritionFact (mealID, calorieCount, sodiumCount, carbohydrates, sugarCount, cholestrol) : Recipe

+ setMealID(mealID) : void

+ setCalorieCount (calorieCount) : void

+ setSodiumCount (sodiumCount) : void

+ setCarbohydrates (carbohydrates) : void

+ setCarbohydrates (carbohydrates) : void

+ setSugarCount (sugarCount) : void

+ setCholestrol (cholestrol) : void

+ getMealID() : integer

+ getCalorieCount() : integer

+ getSodiumCount() : integer

+ getCarbohydrates() : integer

+ getSugarCount() : integer

+ getCholestrol() : integer

Customer

- custID: integer {key}

- custName: string

-custUsername: string

-custPassword: string

-custZipCode: integer

- custSubscriptionType: string

+ createCustomer (name, username, password, zipcode, subscriptionType) : Customer

+ setCustName (custName) : void

+ setCustUsername (custUsername) : void

+ setCustPassword (custPassword) : void

+ setCustZipCode (custZipCode) : void

+setCustSubscriptionType(custSubscriptionType) : void

+getCustomerID(): integer

+getCustName(): string

+getCustUsername(): string

+getCustPassword() : string

+getCustZipCode() : integer

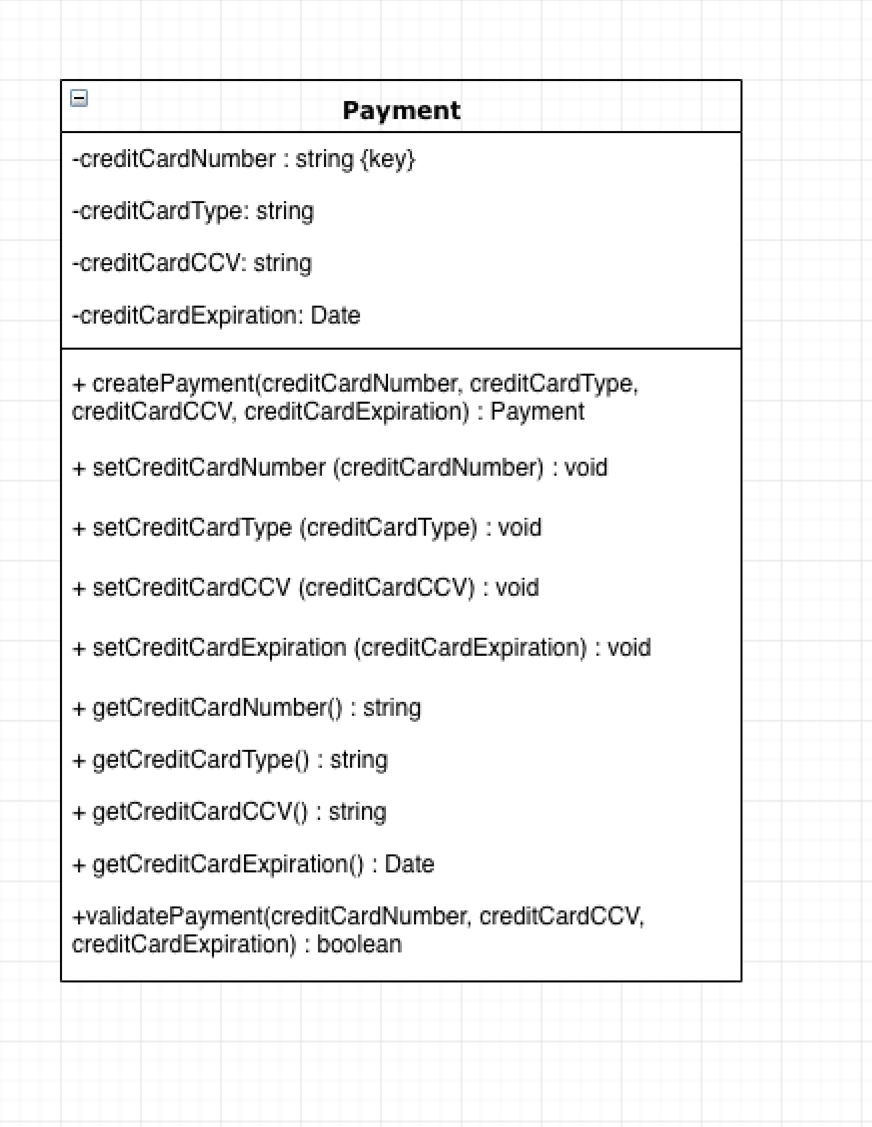
+getSubscriptionType() : string

listCustomerAllergy (custID) : Allergy [ ]

countCustomerAllergy (custID) : int

makePayment (custID) : boolean

findNearbyStores(custZipCode) : Store [ ]



Payment

-creditCardNumber : string {key}

-creditCardType: string

-creditCardCCV: string

-creditCardExpiration: Date

+ createPayment(creditCardNumber, creditCardType, creditCardCCV, creditCardExpiration) : Payment

+ setCreditCardNumber (creditCardNumber) : void

+ setCreditCardType (creditCardType) : void

+ setCreditCardCCV (creditCardCCV) : void

+ setCreditCardExpiration (creditCardExpiration) : void

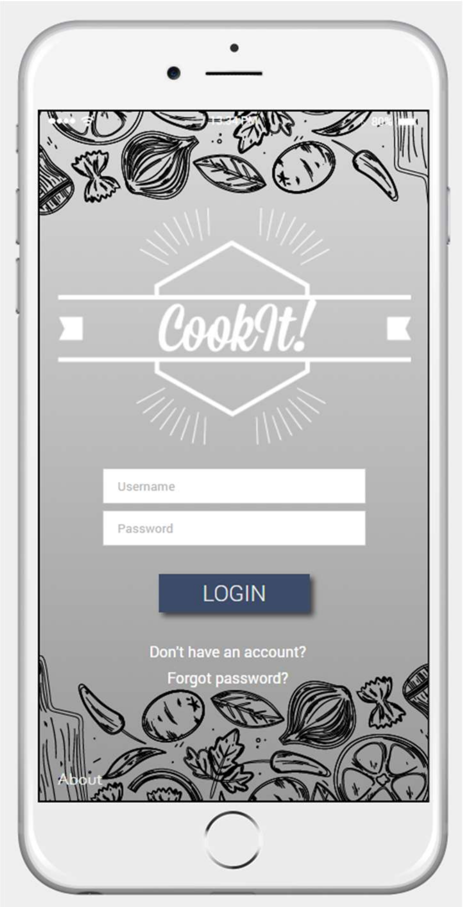
+ getCreditCardNumber() : string

+ getCreditCardType() : string

+ getCreditCardCCV() : string

+ getCreditCardExpiration() : Date

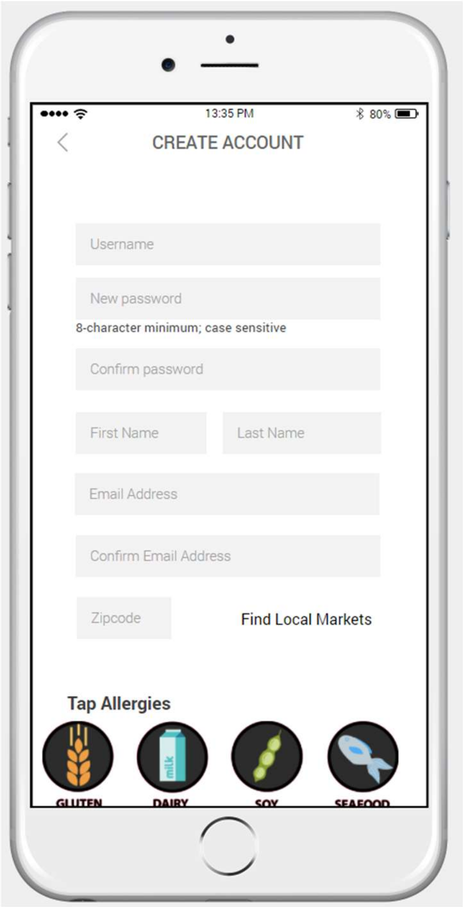
+validatePayment(creditCardNumber, creditCardCCV, creditCardExpiration) : boolean



LOGIN SCREEN

Figure 1

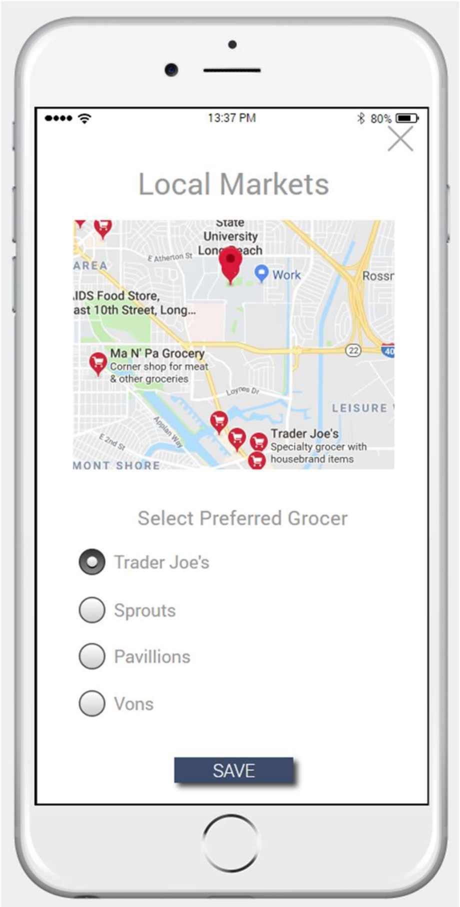
Upon opening the app, users can enter their account information to login. They may also choose to create or recover an existing account if they are new or unable to login.



CREATE ACCOUNT SCREEN

Figure 2.1

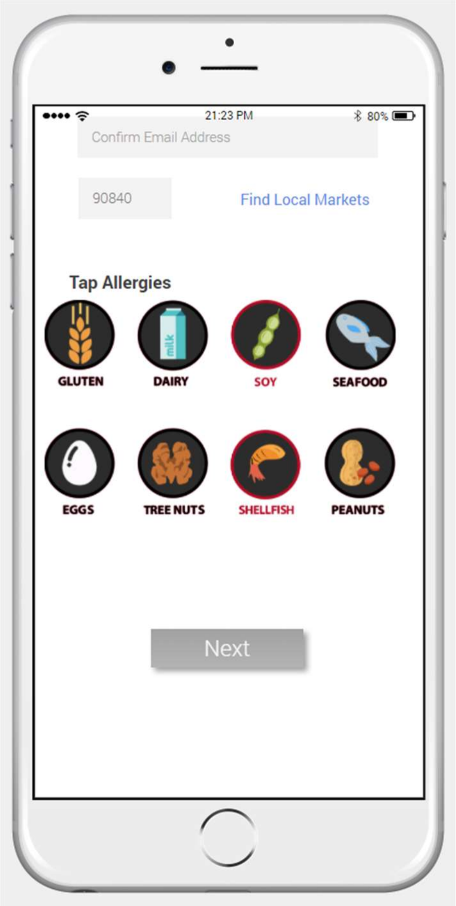
Users that choose to create a new account from the login screen will be greeted with the Create Account screen seen in Figure 2.1. Users will be asked to input their personal information first and upon the zip code input, users will tap 'Find Local Markets' to get a pop-up screen seen in Figure 2.2.



CREATE ACCOUNT SCREEN

Figure 2.2

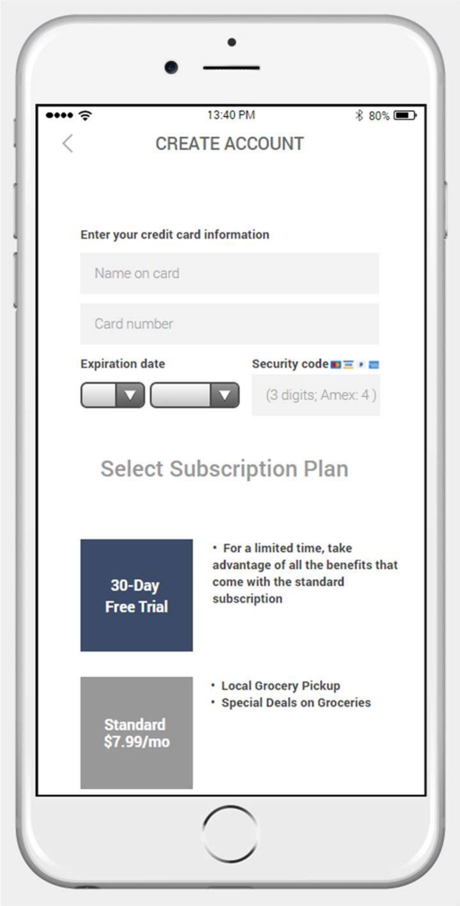
From the 'Local Markets' pop-up screen, the detected user location will be marked, and nearby grocers will be shown on the map. Users will then be able to pick their preferred grocery store for meal pick-up from a radio list of participating grocers.



CREATE ACCOUNT SCREEN

Figure 2.3

After selecting their preferred grocer, users have the option to select any possible allergen that they may have. These allergies were selected from a list of the most common allergens. These icons will be shown on every meal recipe as seen in Figure 5.1.

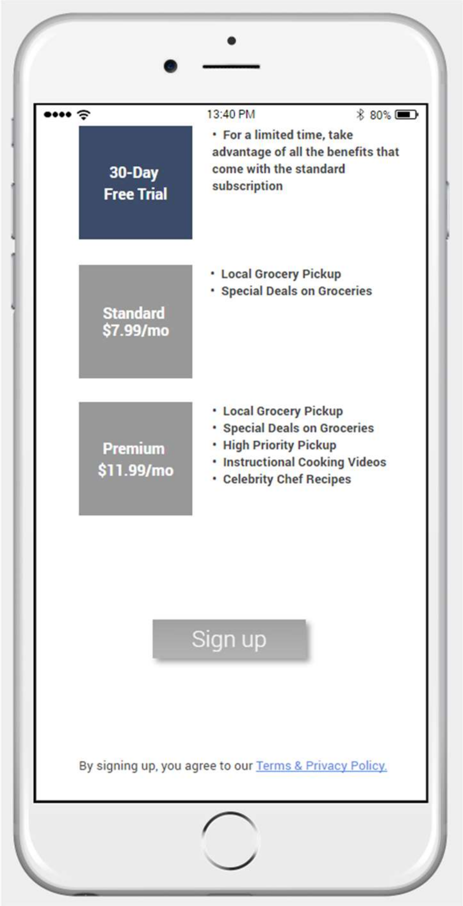


- For a limited time, take advantage of all the benefits that come with the standard subscription
- Local Grocery Pickup
- Special Deals on Groceries

CREATE ACCOUNT SCREEN

Figure 2.4

After users fill out their personal information and preferences, users will be asked to submit their payment information as well as select from one of our available subscription plans.



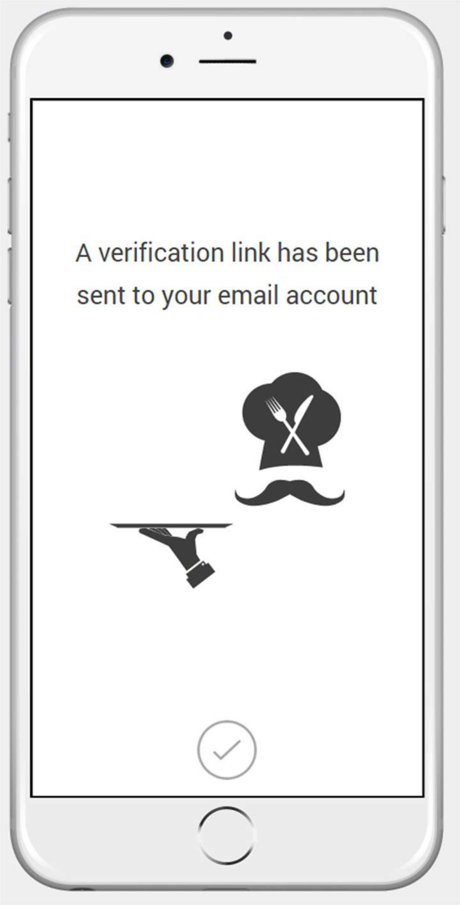
- For a limited time, take advantage of all the benefits that come with the standard subscription
- Local Grocery Pickup
- Special Deals on Groceries
- Local Grocery Pickup
- Special Deals on Groceries
- High Priority Pickup
- Instructional Cooking Videos
- Celebrity Chef Recipes

By signing up, you agree to our Terms & Privacy Policy.

CREATE ACCOUNT SCREEN

Figure 2.5

Users can then view our terms and conditions should they wish and tap 'Sign up' to submit all their information. A verification screen (Figure 3) will then be displayed to prompt users to verify their email and complete the account creation process.

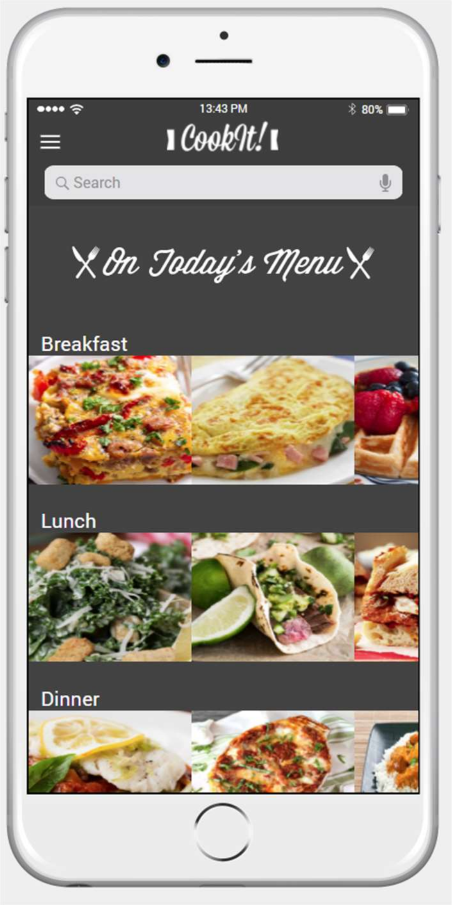


A verification link has been sent to your email account

VERIFICATION SCREEN

Figure 3

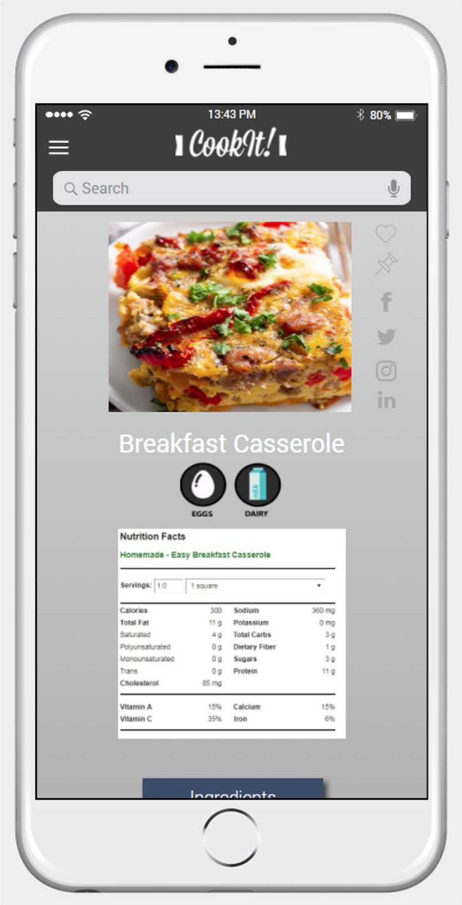
The screen that is displayed after submitting account creation information.



HOME SCREEN

Figure 4

Once users log into their account from the Login Screen (Figure 1) they will be greeted with a gallery of meal recipes. From here, users can search, browse, and select any meal recipes that they may find interesting. Meals will be updated per availability of ingredients in stores and dependent on user preferences.



Nutrition Facts

Homemade - Easy Breakfast Casserole

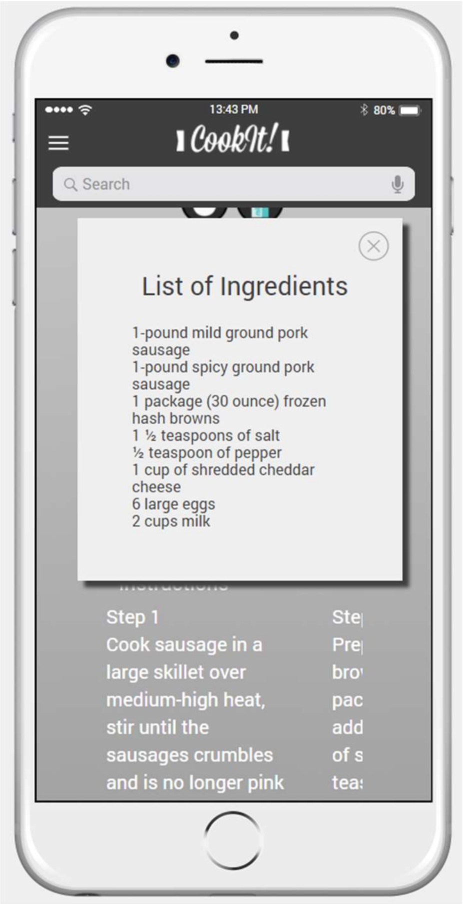
Servings: 1.0 1 square

Calories	300	Sodium	360 mg
Total Fat	11 g	Potassium	0 mg
Saturated	4 g	Total Carbs	3 g
Polyunsaturated	0 g	Dietary Fiber	1 g
Monounsaturated	0 g	Sugars	3 g
Trans	0 g	Protein	11 g
Cholesterol	85 mg		
Vitamin A	15%	Calcium	15%
Vitamin C	35%	Iron	6%

RECIPE SCREEN

Figure 5.1

Once a meal recipe is selected, users will see the name of the meal, the allergens within the recipe, nutrition facts, list of ingredients, and step-by-step cooking instructions.



List of Ingredients

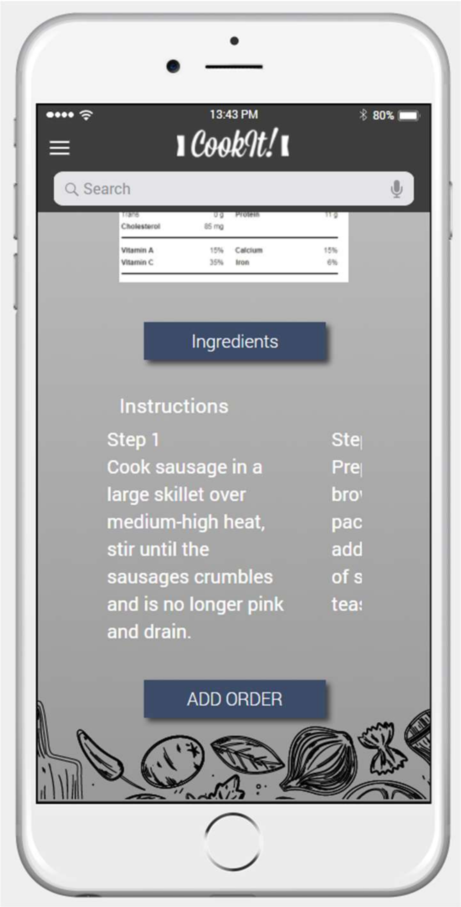
1-pound mild ground pork sausage
1-pound spicy ground pork sausage
1 package (30 ounce) frozen hash browns
1 ½ teaspoons of salt
½ teaspoon of pepper
1 cup of shredded cheddar cheese
6 large eggs
2 cups milk

Step 1
Cook sausage in a large skillet over medium-high heat, stir until the sausages crumbles and is no longer pink

RECIPE SCREEN

Figure 5.2

Here we see what the list of ingredients pop-up would look like if the user taps on the ingredients button seen in Figure 5.3.



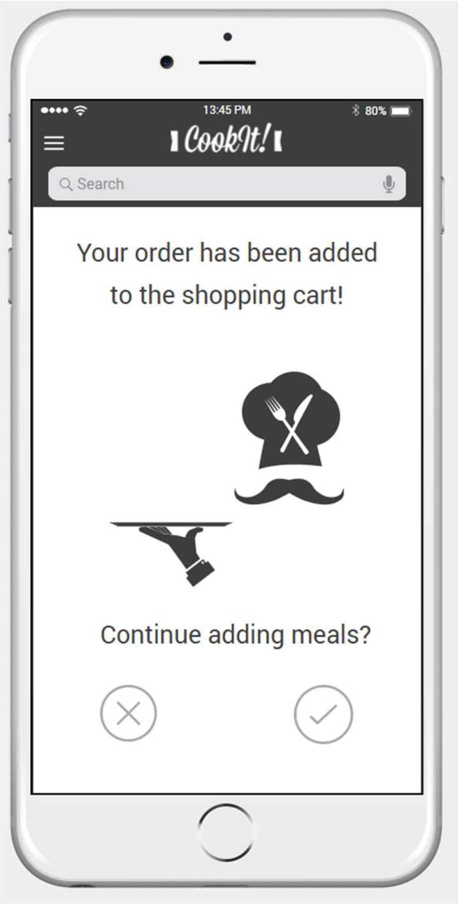
Instructions

Step 1
Cook sausage in a large skillet over medium-high heat, stir until the sausages crumbles and is no longer pink and drain.

RECIPE SCREEN

Figure 5.3

Users can horizontally scroll to read through the cooking instructions. The bottom of the screen as shown above, is where users can order the meal's ingredients to their shopping cart.



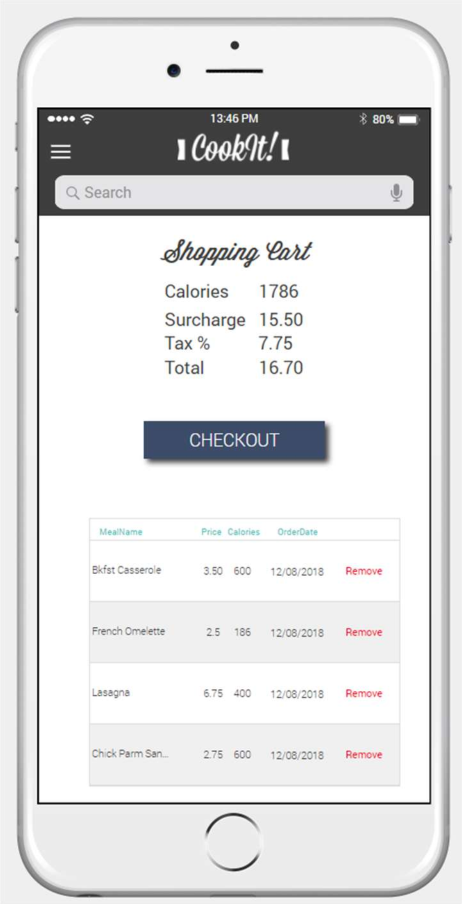
Your order has been added to the shopping cart!

Continue adding meals?

ADDED ORDER SCREEN

Figure 6

Once the user adds an order, they will be prompted by this screen. From here users can choose to continue adding meals, which leads them back to the home screen (Figure 4) or stop adding meals and view their shopping cart (Figure 7).



Shopping Cart

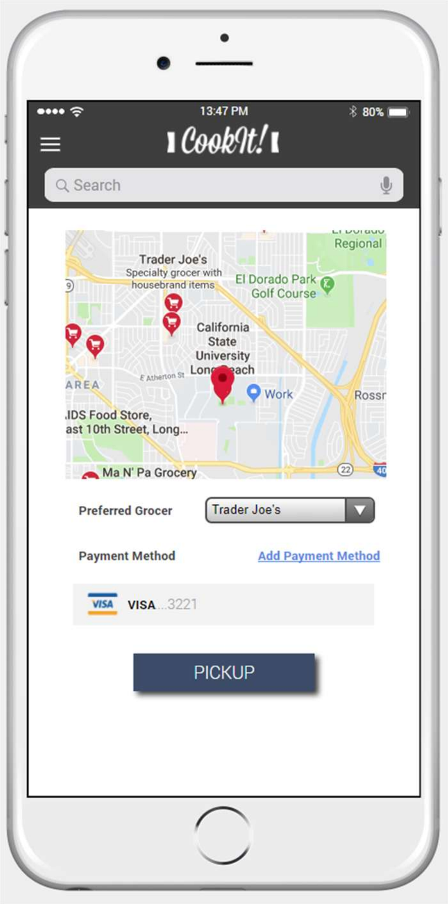
Calories	1786
Surcharge	15.50
Tax %	7.75
Total	16.70

MealName	Price	Calories	OrderDate	
Bkfst Casserole	3.50	600	12/08/2018	Remove
French Omelette	2.5	186	12/08/2018	Remove
Lasagna	6.75	400	12/08/2018	Remove
Chick Parm San...	2.75	600	12/08/2018	Remove

SHOPPING CART SCREEN

Figure 7

In the shopping cart screen, users can see their added meals and remove them from the cart should they choose. The shopping cart also lists additional information about their order such as total calories and price. Users satisfied with their list of meals can then check out their shopping cart from this screen.


●●●● 13:47 PM 80%
CookIt!
Search
Trader Joe's
Specialty grocer with housebrand items
El Dorado Park Golf Course
Regional
California State University Long Beach
Work
Rossr
AREA
IDS Food Store, ast 10th Street, Long...
Ma N' Pa Grocery
PICKUP

Preferred Grocer: Trader Joe's

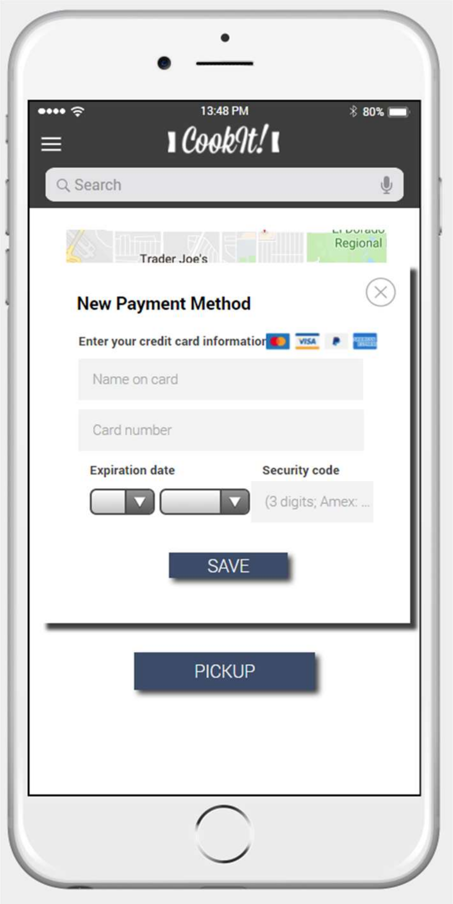
Payment Method — Add Payment Method

VISA VISA ...3221

PICKUP CONFIRMATION SCREEN

Figure 8.1

Tapping the 'Checkout' button in Figure 7 will display this screen. Here users are shown grocery stories nearby, their preferred grocer and the ability to change it, as well as their payment information and the option to add a different payment method.



New Payment Method

Enter your credit card information

Name on card

Card number

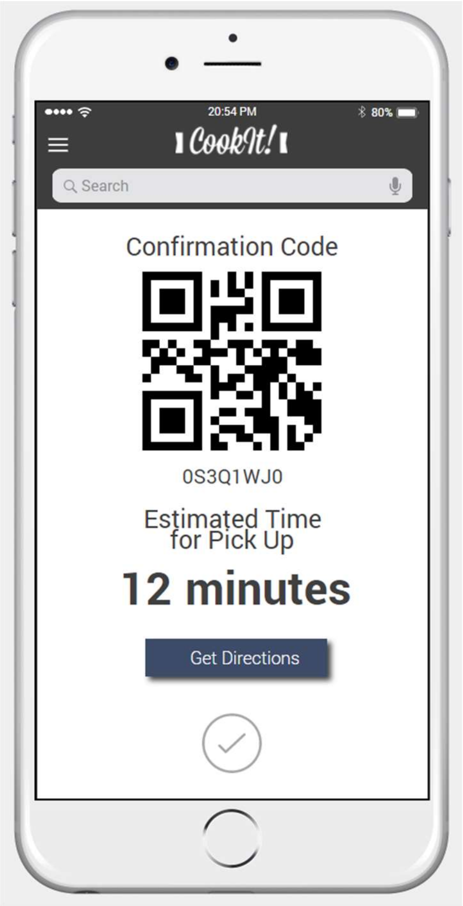
Expiration date

Security code (3 digits; Amex: ...

PICKUP CONFIRMATION SCREEN

Figure 8.2

Here we see what the pop-up would look like should the user choose to add a new payment method from the one currently saved as default.



Confirmation Code

0S3Q1WJ0

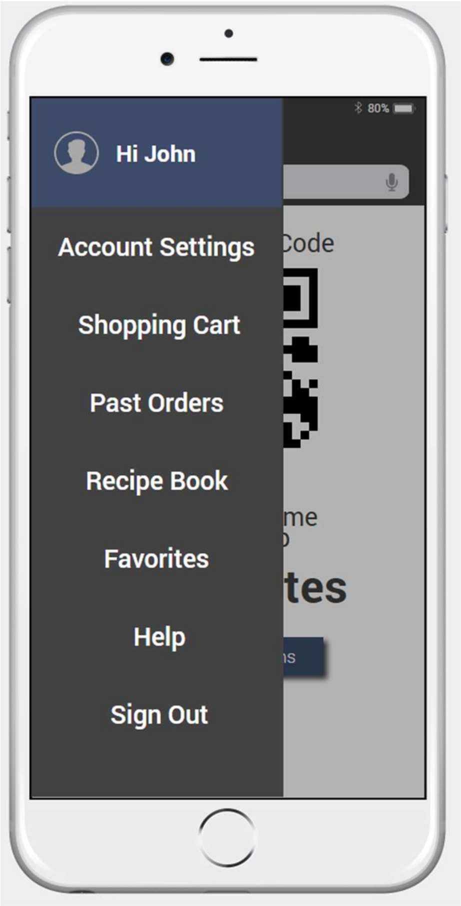
Estimated Time for Pick Up

12 minutes

PICKUP STATUS SCREEN

Figure 9.1

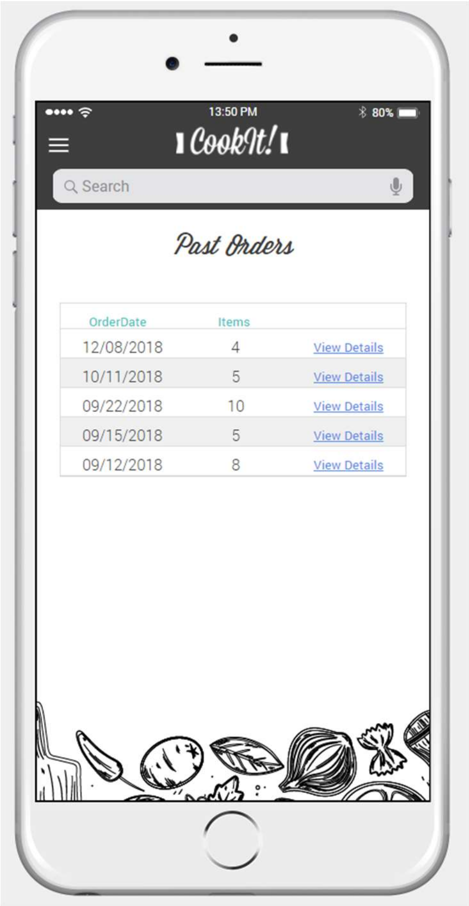
After confirming pickup, users will be taken to this screen where they can see their confirmation code, the estimated time for their order to be ready for pickup, as well as the ability to automatically be sent directions on their phone's default navigation app such as Google or Apple Maps.



SLIDE MENU

Figure 9.2

Users can also access additional screens from the slide menu which can be accessed from the header anywhere within the app once logged in.



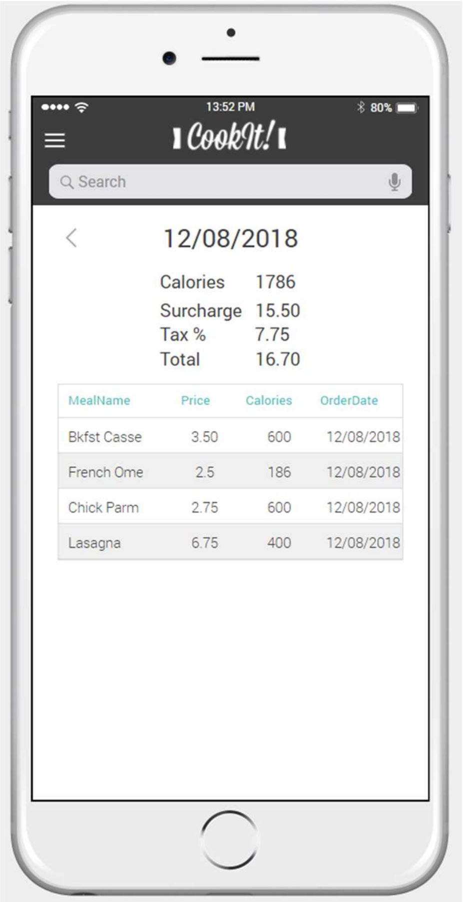
Past Orders

OrderDate	Items	
12/08/2018	4	View Details
10/11/2018	5	View Details
09/22/2018	10	View Details
09/15/2018	5	View Details
09/12/2018	8	View Details

PAST ORDERS SCREEN

Figure 10

In the Past Orders screen (accessible from the slide menu as shown Figure 9.2), users will be able to view the invoices for any of their previous orders.



12/08/2018

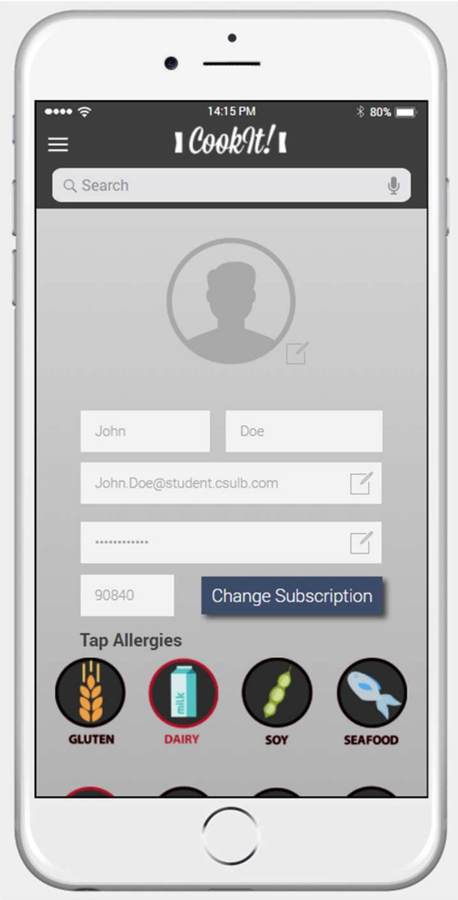
Calories	1786
Surcharge	15.50
Tax %	7.75
Total	16.70

MealName	Price	Calories	OrderDate
Bkfst Casse	3.50	600	12/08/2018
French Ome	2.5	186	12/08/2018
Chick Parm	2.75	600	12/08/2018
Lasagna	6.75	400	12/08/2018

ORDER INVOICE SCREEN

Figure 11

Here we see an example of what an order's invoice would look like.



John

Doe

John.Doe@student.csulb.com

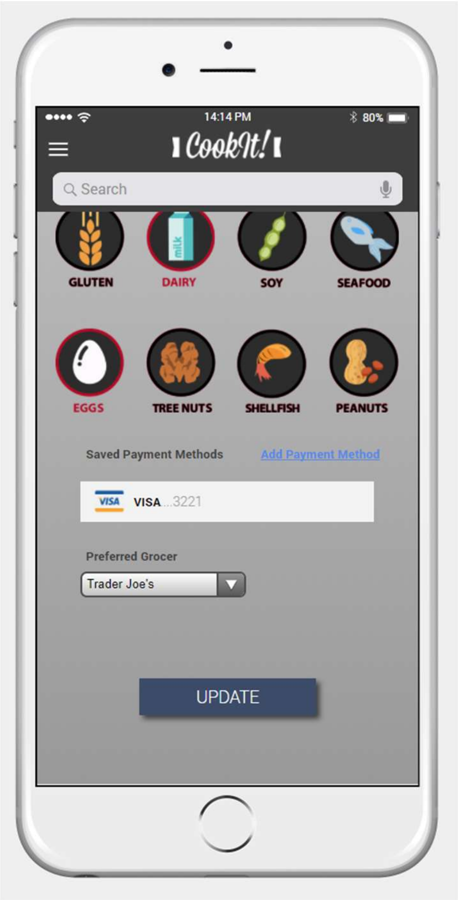
••••••••••••

90840

ACCOUNT SETTINGS SCREEN

Figure 12.1

Users can also make changes to their account information by going to the Account Settings screen (accessible from the slide menu as shown in Figure 9.2). Here we can see users are able to make changes to their profile photo, personal information, subscription plan, and allergies.



Saved Payment Methods

VISA ...3221

Preferred Grocer

Trader Joe's

ACCOUNT SETTINGS SCREEN

Figure 12.2

When users scroll vertically, they will see additional settings such as the payment methods they have saved on their account as well as their current preferred grocer.

Conclusion

This section will describe the direction that our team would like to take for the application in future enhancements. Our team will aim to have most, if not all, of these enhancements ironed out before the final project presentation date, should we have time.

Improving Accessibility: Since our application is within the food and consumer goods domain, it is important to ensure that it will be accessible to all types of users. For example, in our proposal we stated that we would like to develop this application on the Android and iOS platforms because these 2 platforms make up 98% of the U.S. mobile market share. Furthermore, in the current scope of our application, we tailored and designed the application with an American audience in mind. As a future enhancement, we would like to look into integrating international markets into our customer base. We feel that this may not be too difficult because the concept of grocery stores and mobile payments is widespread across developed markets, and we just have to integrate our solution into it.

Cook It! for Desktop Browser: One of the major pain points that we identified in our Vision Document that caused people to turn away from cooking meals at home was the convenience of eating out or ordering something to go. As a result, our team designed Cook It! which we feel addresses this problem by allowing users to order meals from their phones and pick them up from their local grocery stores. To further extends this level of convenience, we would also like to design Cook It! to work on desktop browsers. We feel that this will allow customers to use Cook It! Services when their phones aren't immediately available. Our common consensus is to design Cook It! as a progressive-web application to be inline with modern web standards. This means that we will also need to make sure that the web application scales across all types of screens. This will allow our all users to experience the same look and feel of the application regardless of the device that they are using, without having to download it.