

Healthy Eats

Team 5

Benjamin Gonzalez, Dan Mendoza, Xavier Byrd

# **Software Design Specification Document**

**Version: (2)**

**Date: (12/3/2019)**

## Table of Contents

<b>1 Introduction</b>	<b>5</b>
1.1 Goals and objectives	5
1.2 Statement of system scope	5
<b>2 Architectural design</b>	<b>8</b>
2.1 System Architecture	8
2.2 Design Rational	10
<b>3 Key Functionality design</b>	<b>10</b>
3.1 Log-In Authentication	10
3.1.1 Log-In Use Cases	10
3.1.2 Processing sequence for Log-In Authentication	12
3.1.3 Structural Design for Healthy Eats - Log-In	16
3.1.4 Key Activities	16
3.1.5 Software Interface to other components	17
3.2 Medical Condition	17
3.2.1 Medical Condition Use Cases	17
3.2.2 Processing sequence for Medical Condition	18
3.2.3 Structural Design for Medical Condition	18
3.2.4 Key Activities for Medical Condition	19
3.2.5 Software Interface to other components	20
3.3 The Favorites Function	20
3.3.1 Favorites Function Use Cases	20
3.3.2 Processing sequence for Favorites Function	21
3.3.3 Structural Design for Favorites	22
3.3.4 Key Activities for Favorites Restaurant Function	23
3.3.5 Software Interface to other components	23
3.4 Select a Restaurant	24
3.4.1 Select Restaurant Use Cases	24

3.4.2 Processing sequence for Favorites Function	25
3.4.3 Structural Design for Favorites Function	25
3.4.4 Key Activities	26
3.3.5 Software Interface to other components	27
<b>4 User interface design</b>	<b>27</b>
4.1 Interface design rules	27
4.2 Description of the user interface	27
4.2.1 Log-In Page	27
4.2.1.1 Log-In Screen Images	27
4.2.1.2 Log-In Objects and Actions	27
4.2.2 Sign Up Page	28
4.2.2.1 Sign Up Form Screen Images	28
4.2.2.2 Objects and Actions	28
4.2.3 Forgot Password Page	28
4.2.3.1 Forgot Password Screen Images	28
4.2.3.2 Forgot Password Objects and Actions	29
4.2.4 Forgot Username Page	29
4.2.4.1 Forgot Username Screen Images	29
4.2.4.2 Forgot Username Objects and Actions	30
4.2.5 Home Page	30
4.2.5.1 Screen Images	30
4.2.5.2 Objects and Actions	30
<b>5 Restrictions, limitations, and constraints</b>	<b>30</b>
<b>6 Testing Issues</b>	<b>31</b>
6.1 Types of tests	31
6.2 List of Test Cases	31
<b>7 Appendices</b>	<b>34</b>
7.1 Packaging and installation issues	34

7.2 User Manual	41
7.3 Open Issues	43
7.4 Lessons Learned	43
7.4.1 Design Patterns	43
7.4.3 Team Communications	43
7.4.4 Task Allocations	43
7.4.5 Desirable Changes	44
7.4.6 Challenges Faced	44

## 1 Introduction

Healthy Eats is an application that provides list of restaurants that serve healthy meals specific to the user's health condition and location. This document describes all data, architectural, interface, and component-level design for the software.

### 1.1 Goals and objectives

The goal of Healthy Eats is to have healthy eating choices according to the user's health condition. The objective of Healthy Eats is to make choosing healthy meals easier for user to have access to restaurants that have healthy options and compliment their medical condition and near their location.

### 1.2 Statement of system scope

Healthy Eats application is intended to be used by the general public and shall be readily available to any user in the United States using Windows computer. The primary Application Programming Interface (API) to develop the application is C# programming using Microsoft Visual Studio 2019 as the Integrated Development Environment (IDE). Microsoft SQL Server 2018 is used as the relational database to store the data of the application. Microsoft SQL Server database is run under Amazon Relational Database Service (RDS). This database has 20GB of storage and can auto scale up to 1,000GB when needed. The initial setup of the database is one CPU and 1GB of RAM to minimize initial maintenance cost. However, the database performance is scalable to ensure the user experience is not compromised during high traffic in data transaction. Microsoft SQL Server Management Studio 2018 is used to manage and develop the database architecture. ADO.NET is used to communicate between the C# programming and SQL database. Figure 1 through 4 display Use Case diagrams for each key feature of the system.

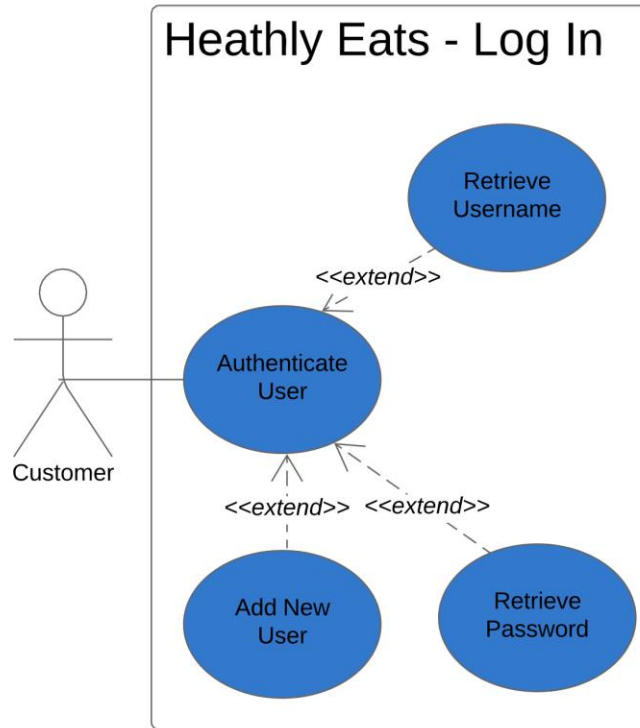


Figure 1: UML Use Case Diagram - Healthy Eats Log In

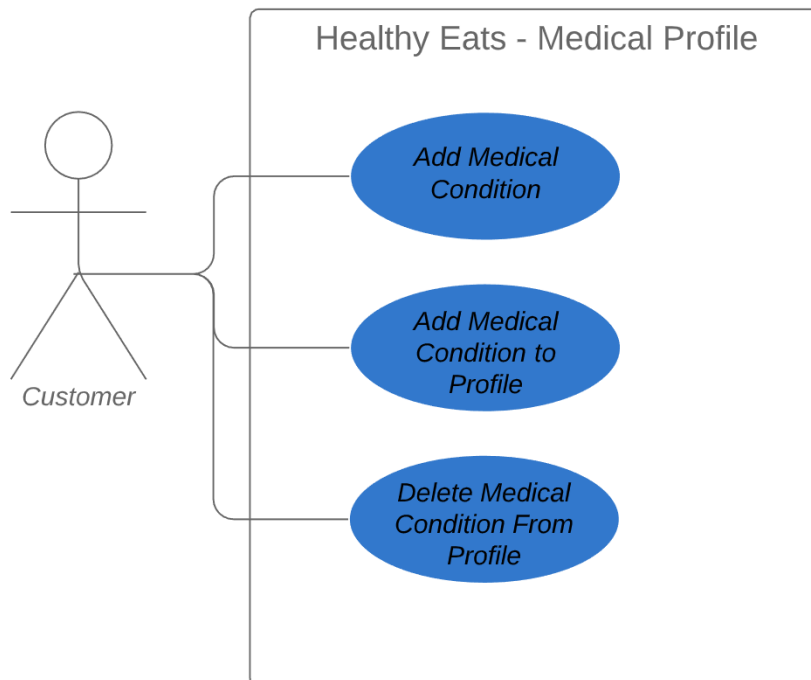


Figure 2: UML Use Case Diagram - Healthy Eats Medical Profile

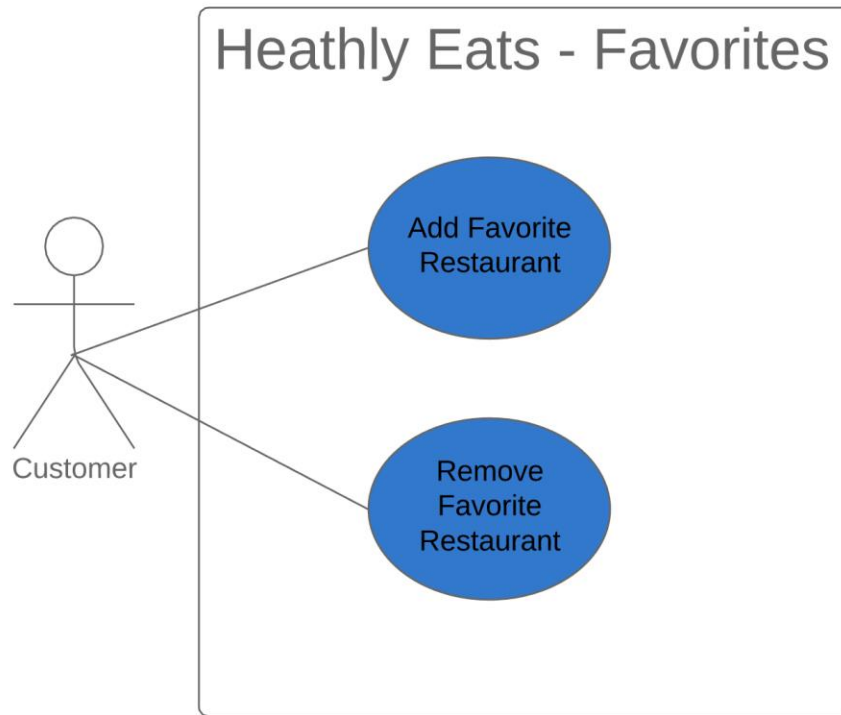


Figure 3: UML Use Case Diagram - Healthy Eats Favorites

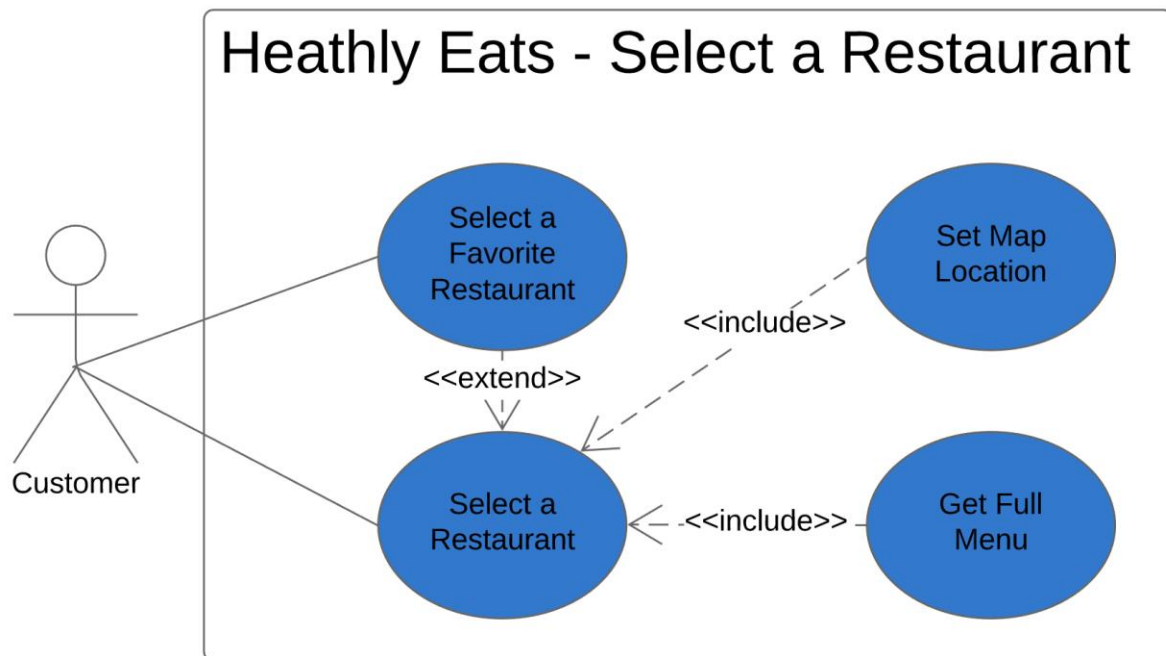


Figure 4: UML Use Case Diagram - Healthy Eats Select Restaurant

## 1.3 Definitions and Acronyms

Acronyms	Definition
API	Application Programming Interface
IDE	Integrated Development Environment
MVC	Model View Controller
RDS	Relational Database Service

## 2 Architectural design

### 2.1 System Architecture

Healthy Eats Application utilizes Client-Server and MVC system architecture. The Client package contains the View of the MVC and the Server package contains the Controller and Model of the MVC. The Controller uses the Google Map API package to display geological map of a restaurant and user's current location. Within Model, SQL Server Stored Procedures are used to send and retrieve data from the RDS SQL Server Database, and Models use ADO.NET to create a connection with the database. Figure 5 illustrates the relationships of each package and how the application operates as a system.



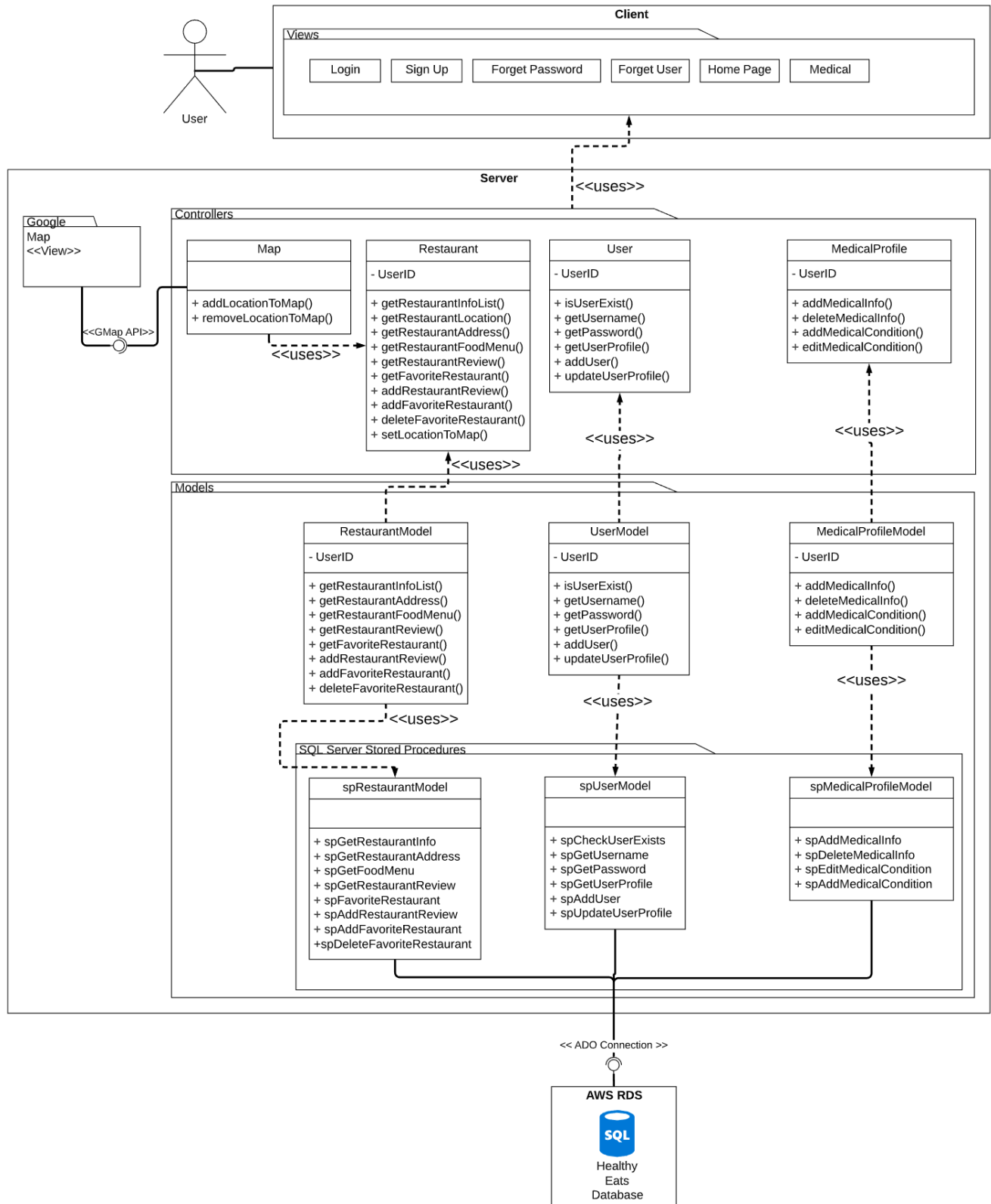


Figure 5: UML Package Diagram - Healthy Eats

## 2.2 Design Rational

Client-Server and MVC architectural design are chosen for this application because the application is expected to be accessed by multiple users with only one data source. Additionally, MVC design pattern allows the application to easily accommodate different User Interfaces such web and mobile UI without any changes from Controller and Model packages, which reduces development effort for scalability and future proofing.

## 3 Key Functionality design

### 3.1 Log-In Authentication

#### 3.1.1 Log-In Use Cases

The user launches the application and enters his or her credentials. The application verifies the user entry for user entry error, and once the input is valid, the application initiates log-in authentication which retrieves information from the RDS SQL database to authenticate the user credentials. Once the user has been authenticated, the application initiates the Home Page of the application with user profile information. Table 1 displays the Use Case of log-in authentication with alternative scenarios.

Use Case	#1 Use the Healthy Eats Application Log-In	
Goal in Context	Log In	
Scope	The Healthy Eats Application	
Level	Primary	
Primary Actor	Customer	
Preconditions	The application is not launched yet	
Minimal Guarantee	The application failed to launch	
Success Guarantee	The application successfully launched	
Trigger	The customer opens the application	
Success Scenario:	Condition: User already have an account in the system	
	Step	Action
	1	The customer launch the application
	2	The customer enters username and password
	3	The system authenticate the customer
	4	The system initializes the Home page
Extensions:	Branching Scenarios	
	Condition: Customer does not have an account yet in the system	
	Step	Action
1A	1	The customer selects the Sign-Up button
	2	The system launch the sign up form

	3	The customer create an account
	4	The system successfully create new account
	Condition: Customer already have an account in the system but forgot username	
	Step	Action
1B	1	The customer selects the Forgot Username button
	2	The system launch the Forgot Username form
	3	The customer enters first name, last name, and birthday
	4	The system retrieves the customer's username
	Condition: Customer already have an account in the system but forgot password	
	Step	Action
1C	1	The customer selects the Forgot Username button
	2	The system launch the Forgot Password form
	3	The customer enters username and birthday
	4	The system retrieves the customer's password

**Table 1: Use Case - Healthy Eats Log-In Authentication**

3.1.2 Processing sequence for Log-In Authentication

Figure 6 to 9 shows sequence diagrams for Healthy Eats **Login, Sign Up** and **Forget Password and Username**.

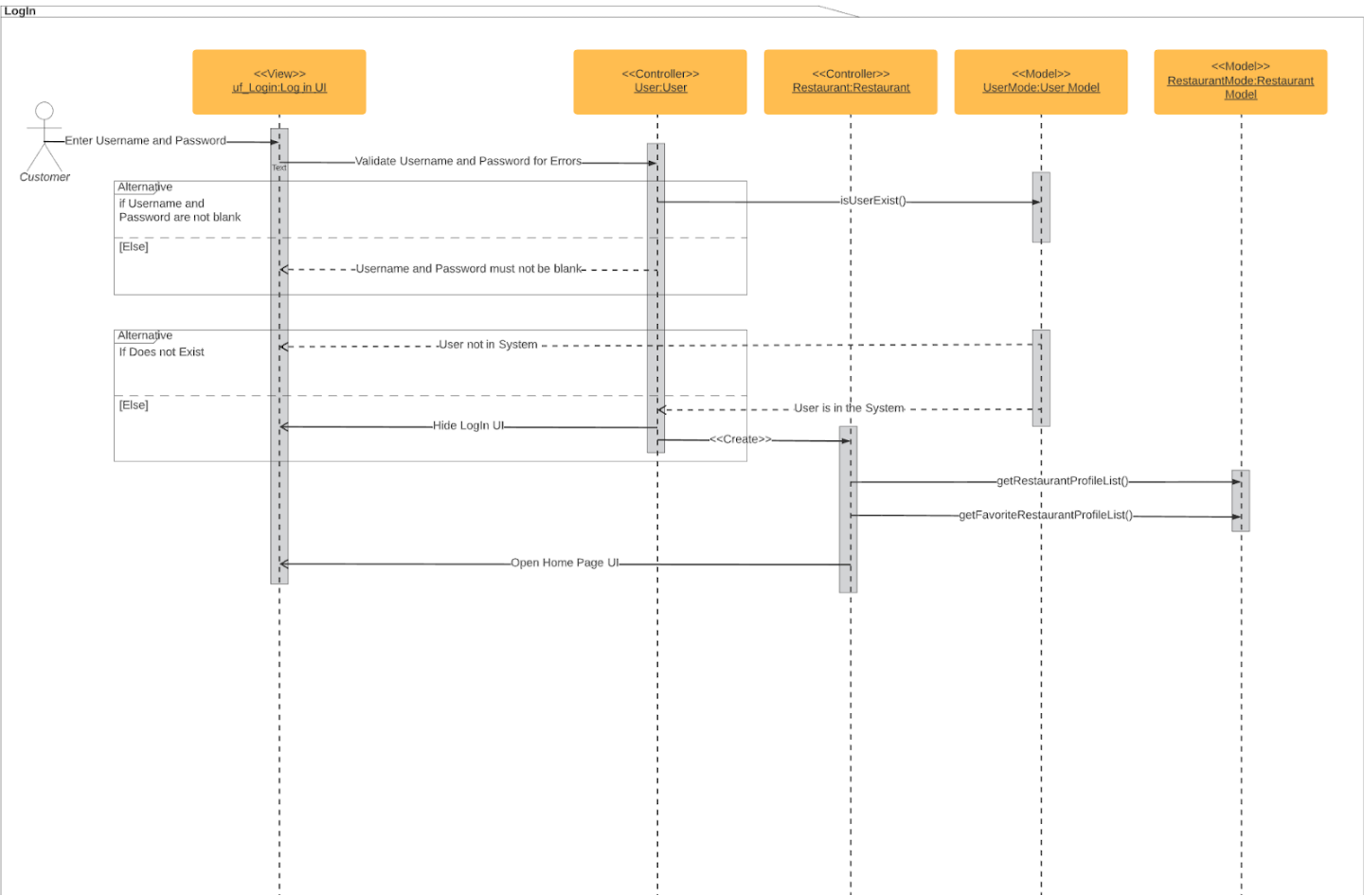
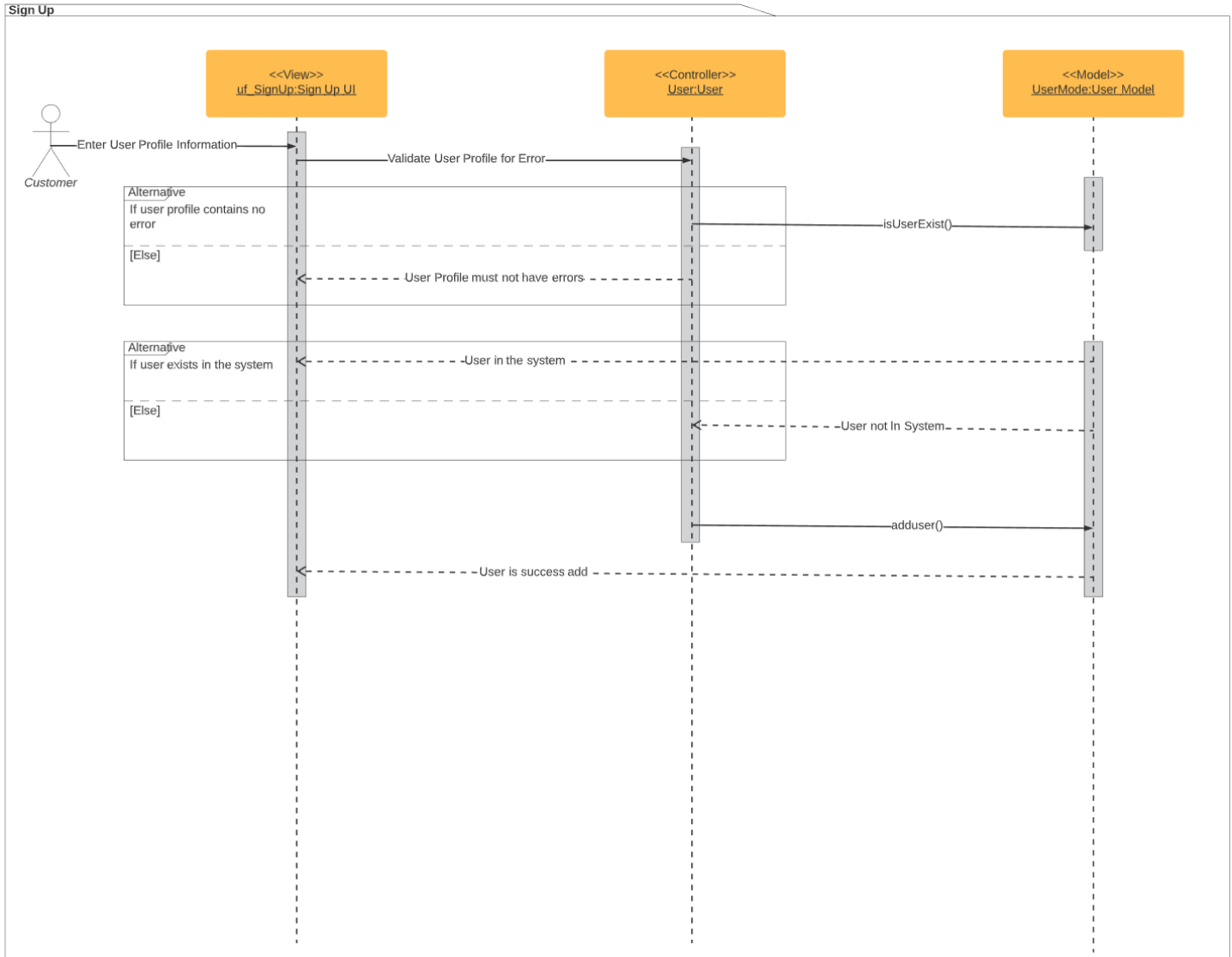


Figure 6: Sequence Diagram - Healthy Eats Log-In Authentication

**Figure 7: Sequence Diagram - Healthy Eats Sign Up**

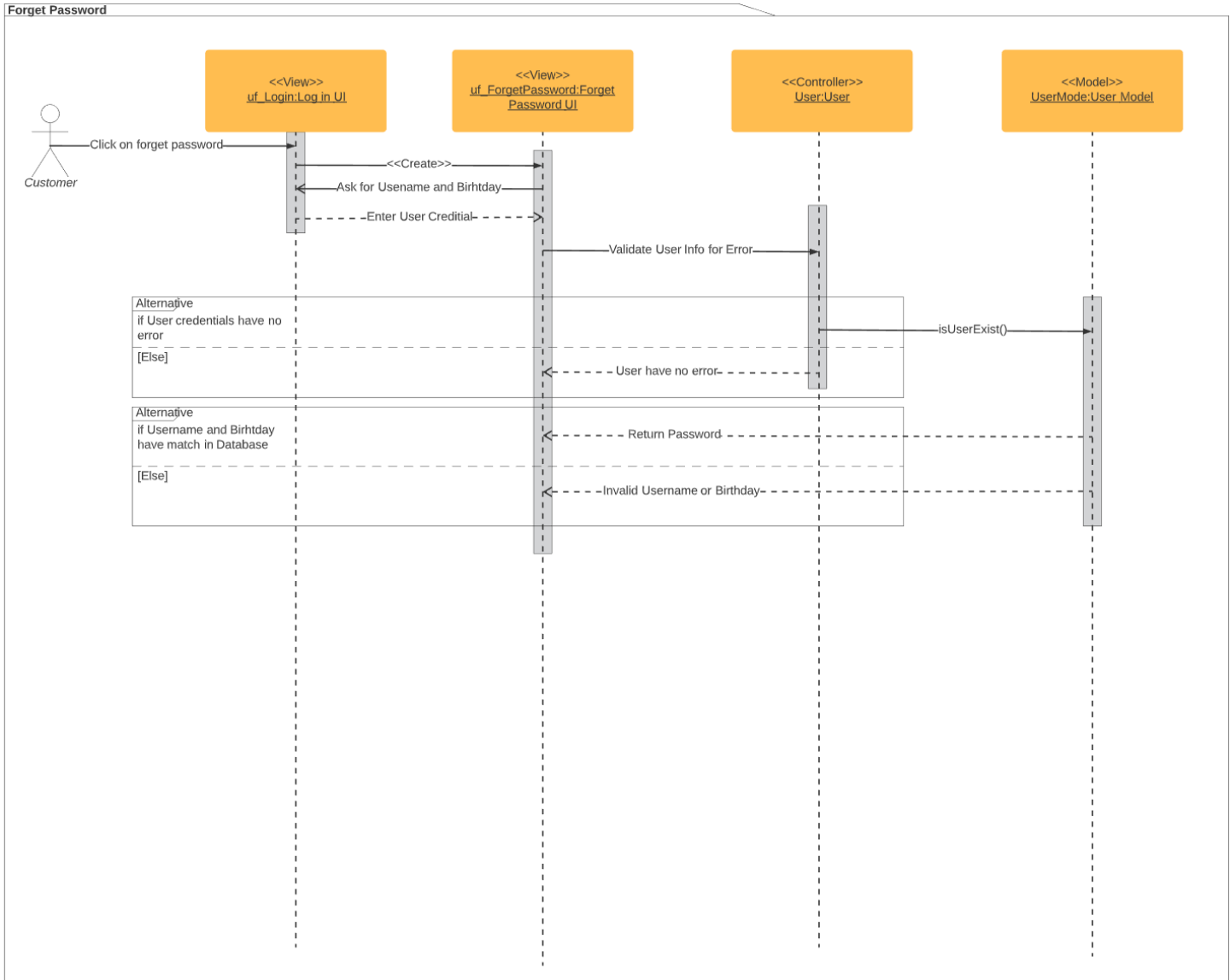


Figure 8: Sequence Diagram - Healthy Eats Forget Password

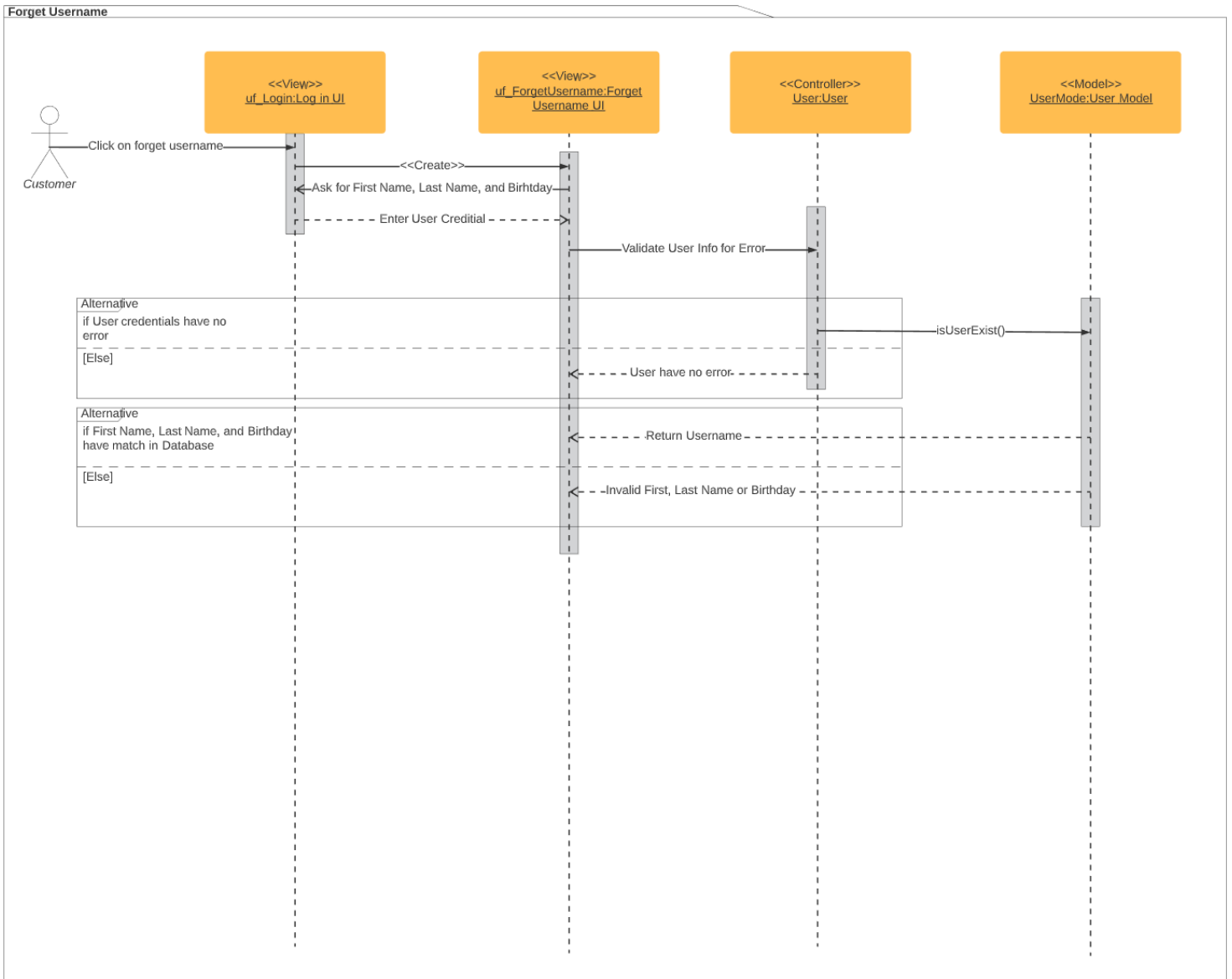
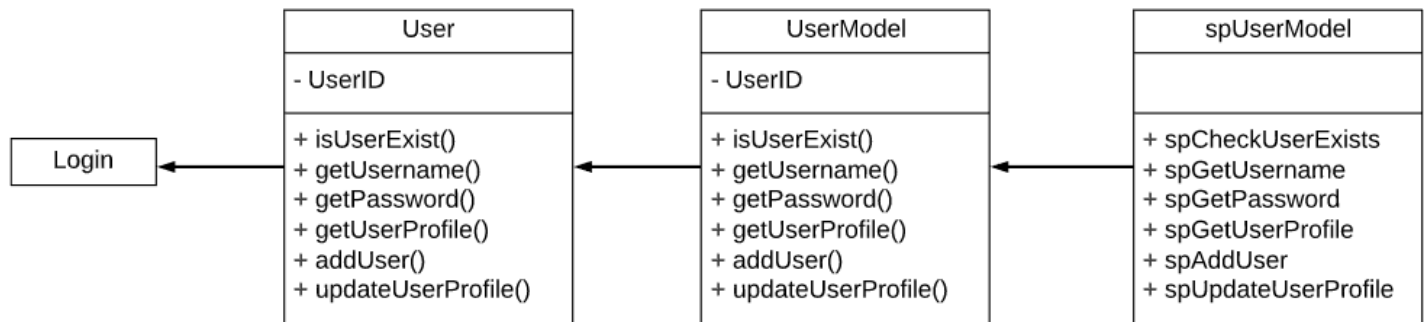


Figure 9: Sequence Diagram - Healthy Eats Forget Username

### 3.1.3 Structural Design for Healthy Eats - Log-In

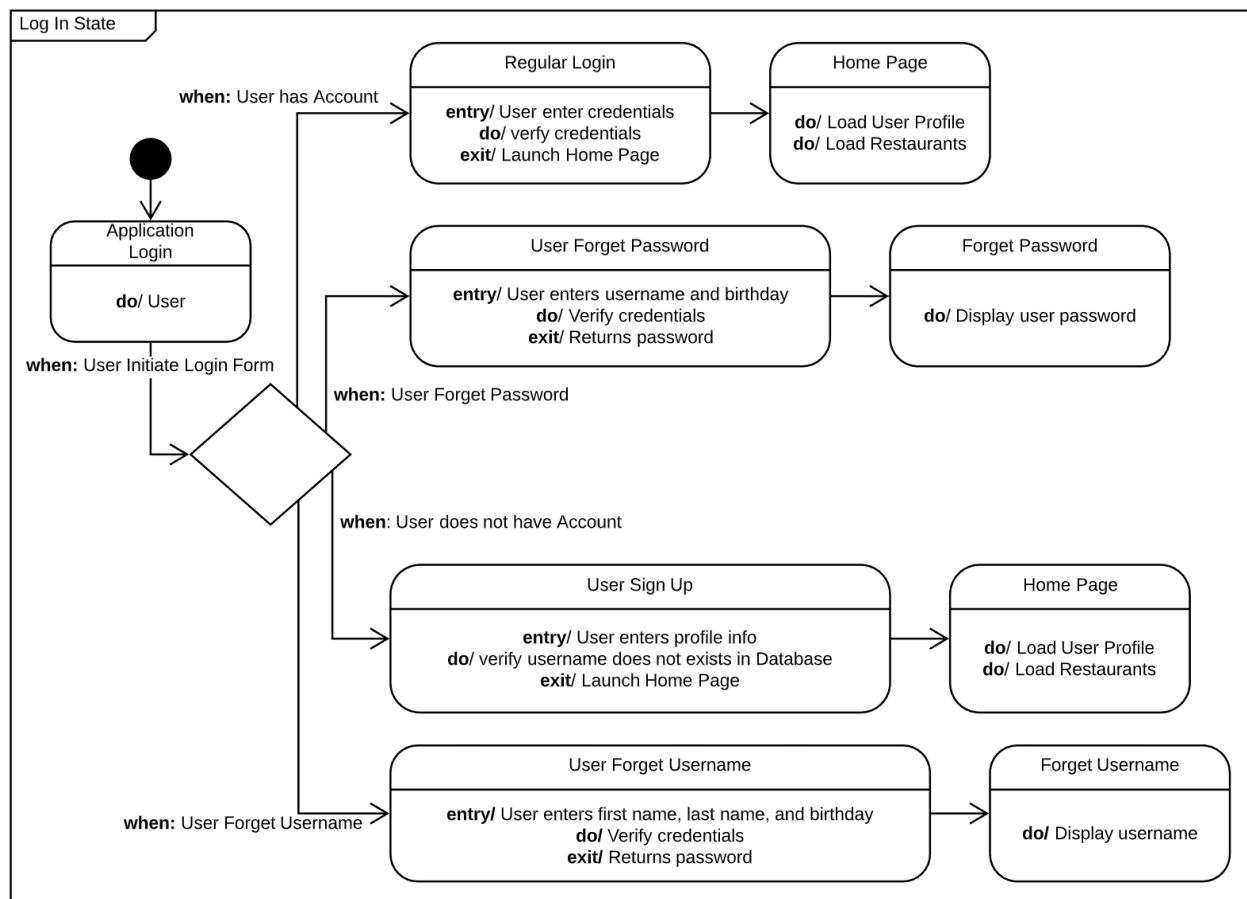
Figure 10 displays the system structure of the Healthy Eats - Log-In application.



**Figure 10: Class Diagram - Healthy Eats Log-In**

### 3.1.4 Key Activities

Figure 11 displays the interactions and events related to the Log-In authentication process.



**Figure 11: State Diagram - Healthy Eats Log-In**



### 3.1.5 Software Interface to other components

Log-In authentication interface uses User controller to verify information of the user, which then used to retrieve user profile by the UserModel and SQL Server Stored Procedure from the RDS SQL database. The User Controller initiates instances of the following interface depending on the user's need: Home Page, Forget Password, Forget Username, and Sign Up.

## 3.2 Medical Condition

### 3.2.1 Medical Condition Use Cases

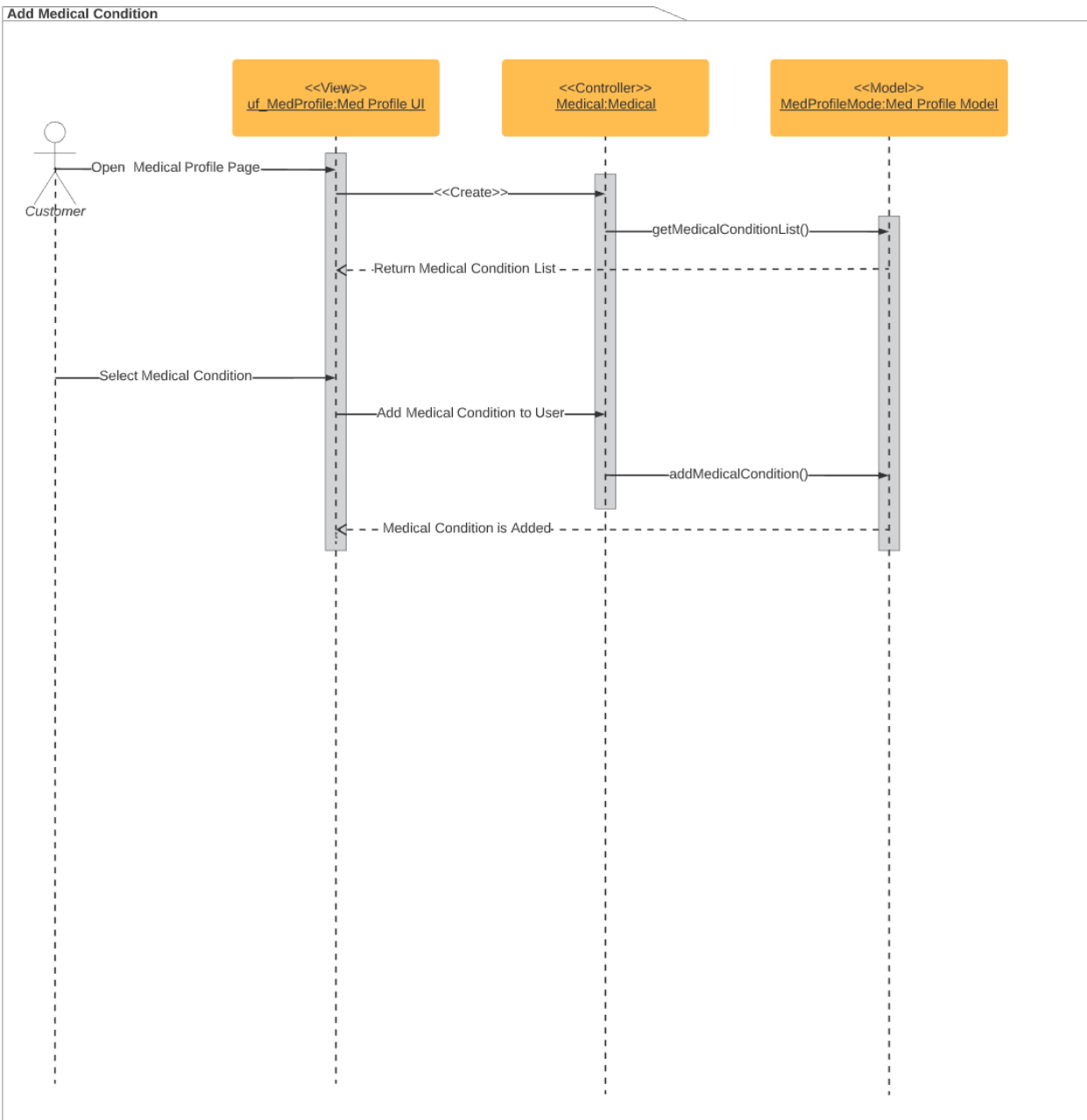
The user opens the Medical Condition user interface and selects medical condition from the dropdown list retrieved from the database. Once the user has selected a medical condition, the system then submits the transaction to the database and creates a record linked to the user's profile. The user may add new medical condition if it is not available from the list provided by the system. The system then adds the new medical condition to the database and refreshes the list in the application.

Use Case	#2 Use the Medical Profile Function	
Goal in Context	Adding new medical information	
Scope	The Healthy Eats Application	
Level	Primary	
Primary Actor	Customer	
Preconditions	The customer has medical condition	
Minimal Guarantee	No medical condition have been added	
Success Guarantee	Customer added at least one medical condition	
Trigger	The customer opens the medical profile form	
Success Scenario:	Condition: Customer already has some medical conditions	
	Step	Action
	1	The customer opens the medical tab
	2	The customer add a medical condition from the dropdown
	3	The system updates customer medical condition
Extensions:	Branching Scenarios	
	Condition: Delete Medical Condition	
1A	1	The customer delete medical condition
	2	The system deletes Medical Condition in user's Profile
	Condition: Add New Medical Condition	
1B	1	The customer types the new medical condition
	2	The system adds the new medical condition to the database
2B	3	The system refreshes the medical condition dropdown

**Table 2: Use Case - Healthy Eats Medical Profile**

### 3.2.2 Processing sequence for Medical Condition

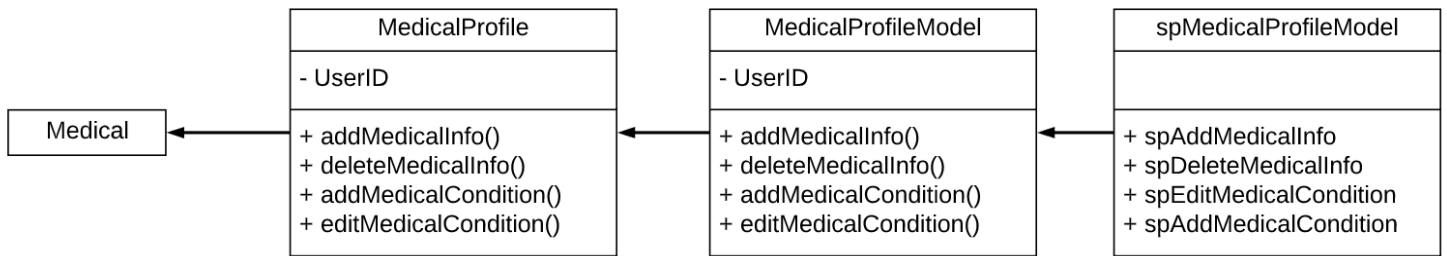
Figure 12 displays the process of adding a medical condition per user account.



**Figure 12: Sequence Diagram - Healthy Eats Medical Add Condition**

### 3.2.3 Structural Design for Medical Condition

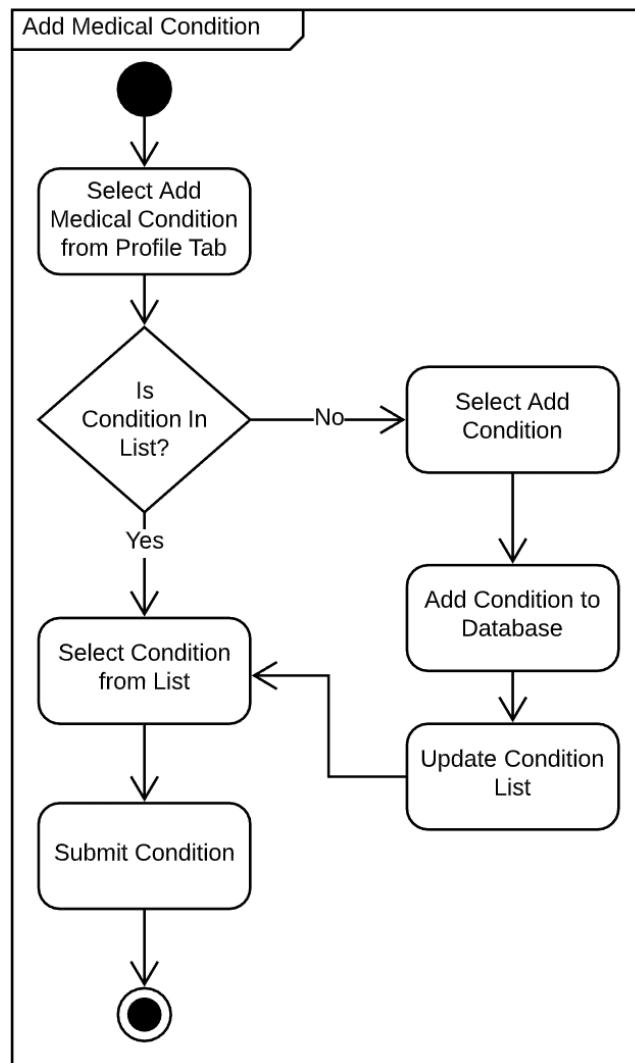
The User opens the medical profile tab and selects a medical condition. The system submits the medical condition to the medical condition controller which then submits the data through the medical condition profile to the database. Refer to figure 13 for the structure of the Medical Condition.



**Figure 13: Class Diagram - Healthy Eats Medical Profile**

### 3.2.4 Key Activities for Medical Condition

The Add Medical Condition in the profile tab have a list of conditions. When you add a condition, it also add into the database and update the condition list; then, submit it to the condition to the medical condition profile. See figure 14 for Activity Diagram for Medical Condition.



**Figure 14: Activity Diagram - Healthy Eats Medical Add Condition**

### 3.2.5 Software Interface to other components

Medical Condition interface uses User and Medical controllers to submit information to MedicalModel and SQL Server Stored Procedure from the RDS SQL database. The Medical Controller refreshes the Medical interface once the transaction has been added to the database.

## 3.3 The Favorites Function

### 3.3.1 Favorites Function Use Cases

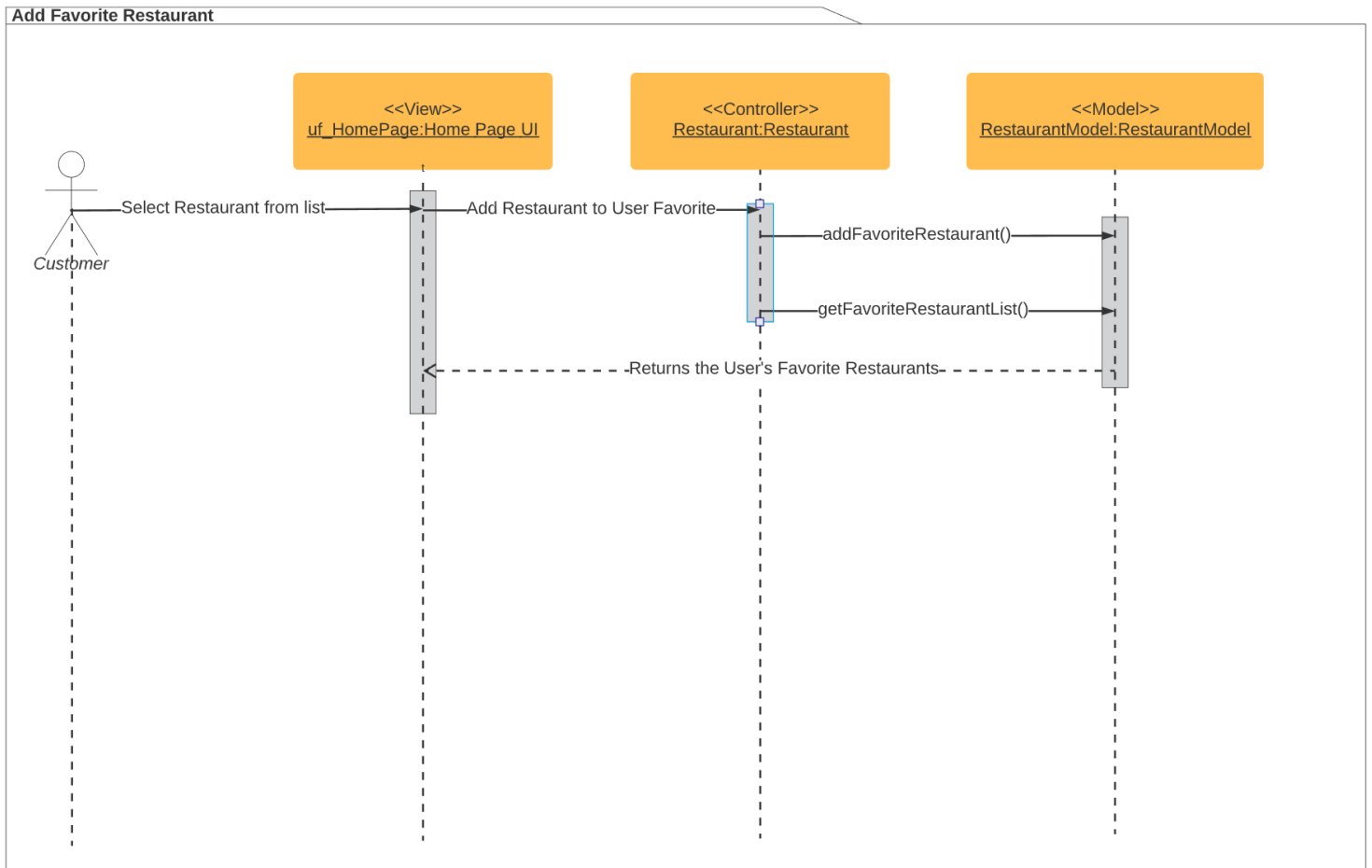
The user opens the Favorites user interface and selects a Restaurant from the user's Favorite list that is retrieved from the database. When the user has selected a Favorite Restaurant, the system then retrieves the restaurants profile to the user. The user may not have any favorite restaurants and will need to use the search user interface to find a restaurant to add to their favorites list. The added favorite restaurant is submitted to the database and the system updates the user's favorites list in the application.

Use Case	3 Use the Favorites	
Goal in Context	Add a Restaurant on the favorite list	
Scope	The Healthy Eats Application	
Level	Primary	
Primary Actor	Customer	
Preconditions	There are no favorite restaurants	
Minimal Guarantee	User has not added any restaurants to their favorite	
Success Guarantee	Customer found desired favored restaurant	
Trigger	The customer select a favorite restaurant from the favorite list	
Success Scenario:	Condition: Customer wants to add new restaurant to Favorites list	
	Step	Action
	1	The customer select a restaurant from the restaurant list
	2	The customer press the add favorites button
	3	The system adds the restaurant from to the user progile
	4	The system retrieves the restaurants profile
	Extensions:	
Extensions:	Branching Scenarios	
	Condition: Customer wants to remove a restaurant from Favorites list	
	Step	Action
	1	The customer select a restaurant from the favorites restaurant list
	2	The customer press the Remove from Favorites button
	3	The system removes the restaurant from the user profile
	4	The system refresh the Home Page interface

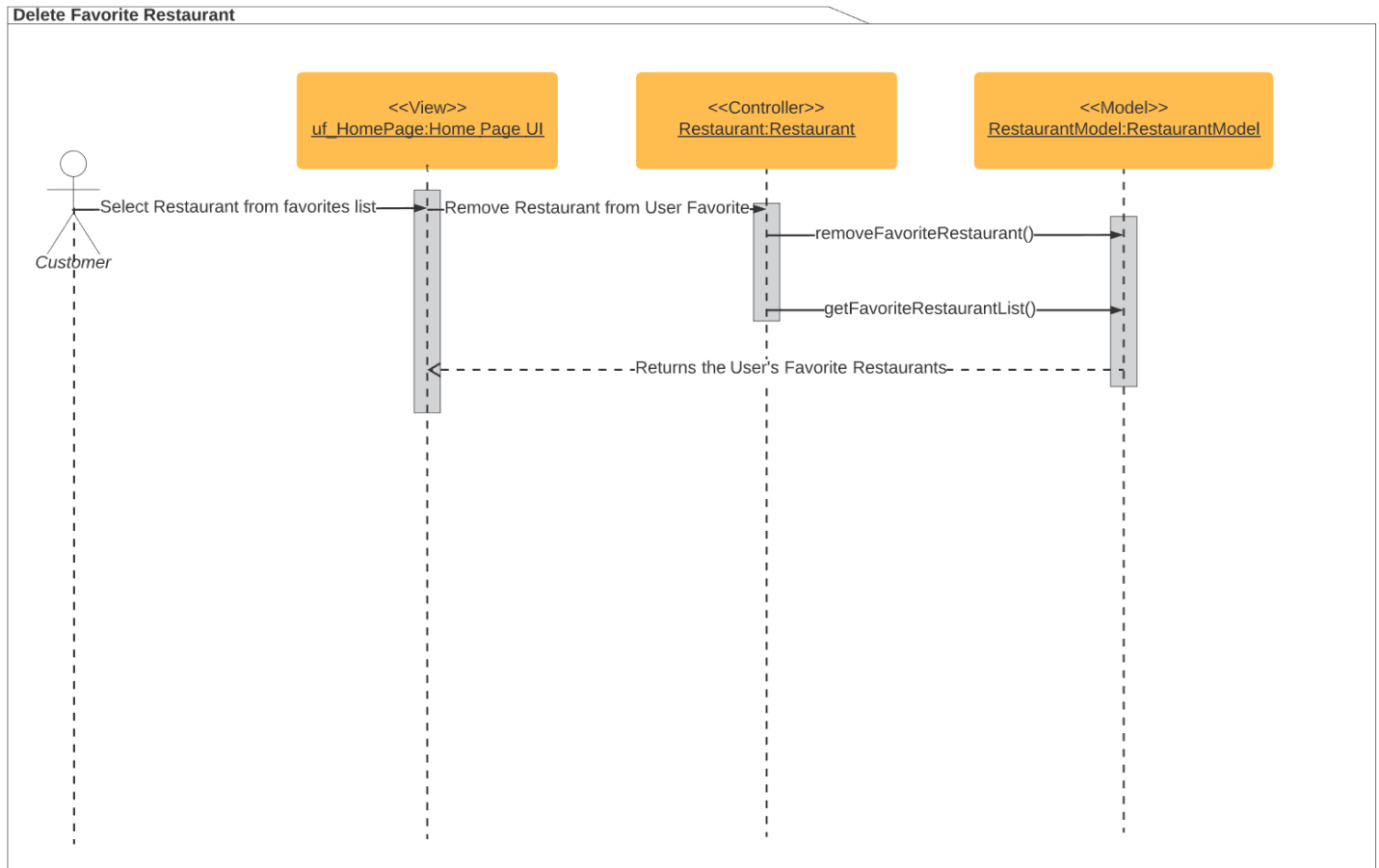
**Table 3: Use Case - Healthy Eats Favorites**

### 3.3.2 Processing sequence for Favorites Function

Figure 15 and 16 display the process of adding and deleting favorite restaurant to user profile.



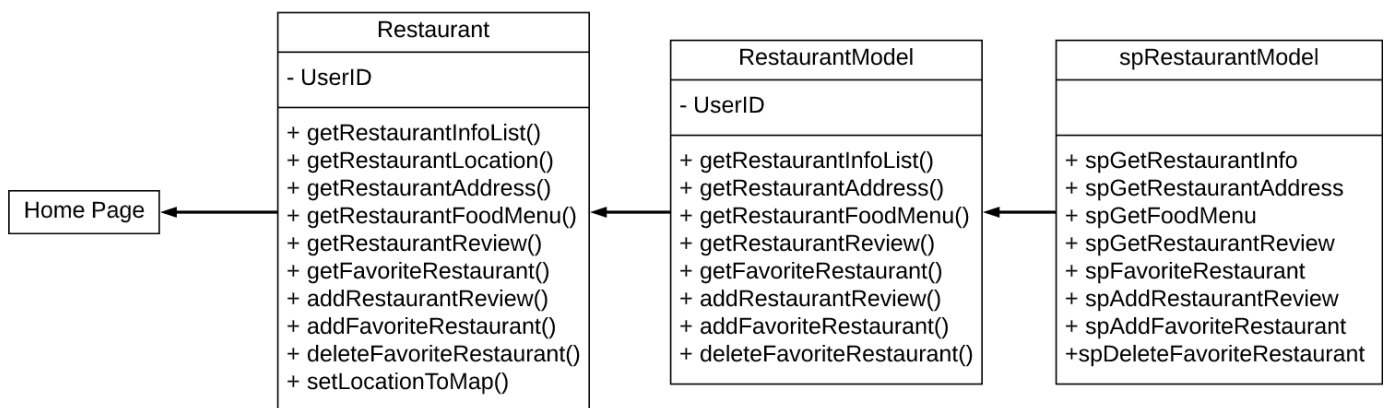
**Figure 15: Sequence Diagram - Healthy Eats Medical Add Favorite Restaurant**



**Figure 16: Sequence Diagram - Healthy Eats Delete Favorite Restaurant**

### 3.3.3 Structural Design for Favorites

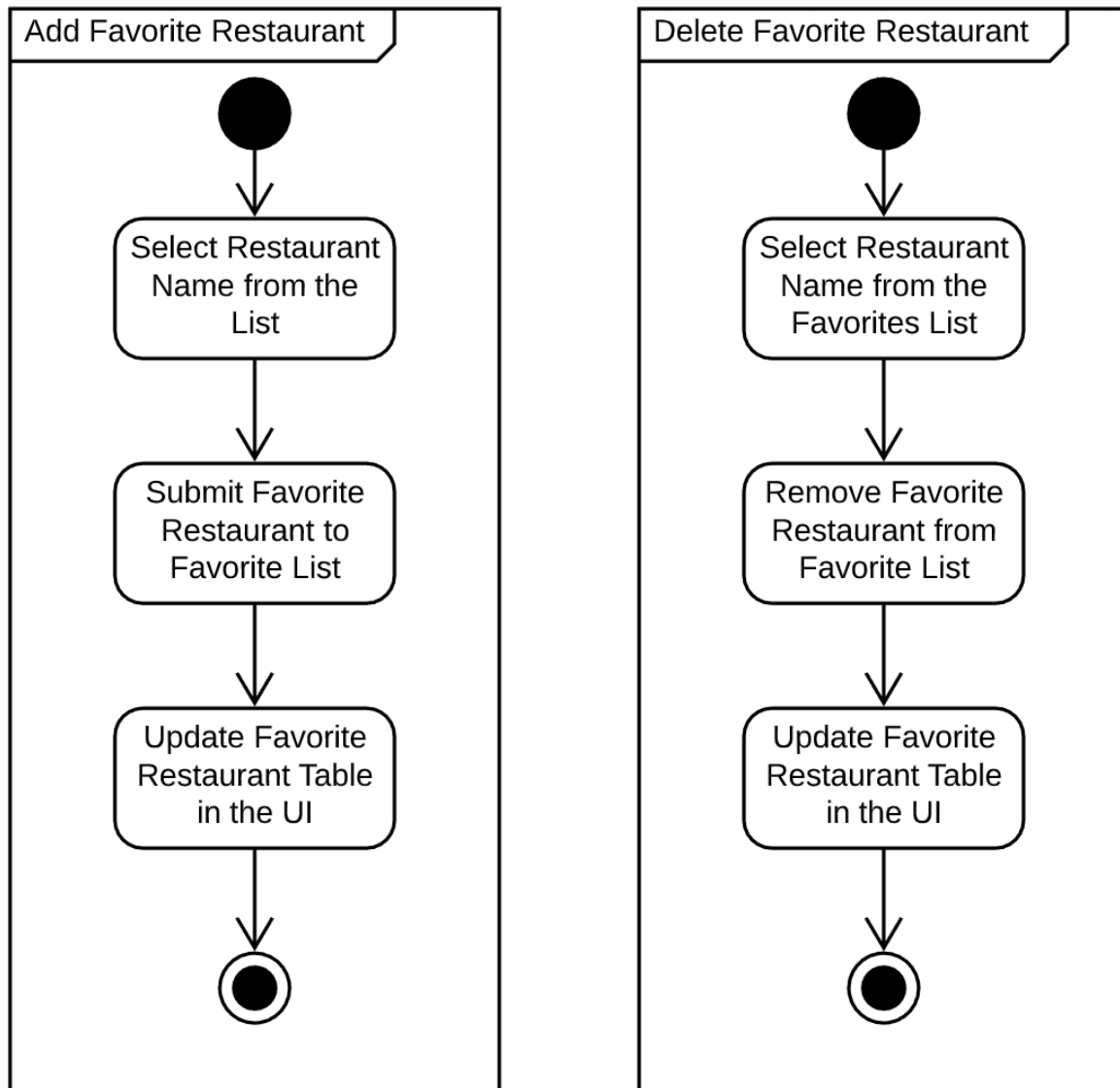
The User selects a restaurant from restaurant list and presses the Add to Favorites button. The system submits the restaurant to the Restaurant controller which then submits the data through the Restaurant Model to the database. Refer to figure 17 for the structure of the Favorites Function.



**Figure 17: Class Diagram - Healthy Eats Restaurant (Add/Remove Favorites)**

### 3.3.4 Key Activities for Favorites Restaurant Function

The Add Medical Condition in the profile tab have a list of conditions. When you add a condition, it also add into the database and update the condition list; then, submit it to the condition to the medical condition profile. See figure 14 for Activity Diagram for Medical Condition.



**Figure 18: Activity Diagram - Healthy Eats Add/Remove Favorite Restaurant**

### 3.3.5 Software Interface to other components

Home Page interface uses Restaurant controllers to submit information to RestaurantModel and SQL Server Stored Procedure from the RDS SQL database. The Restaurant Controller refreshes the Home Page interface once the transaction has been added to the database.

### 3.4 Select a Restaurant

#### 3.4.1 Select Restaurant Use Cases

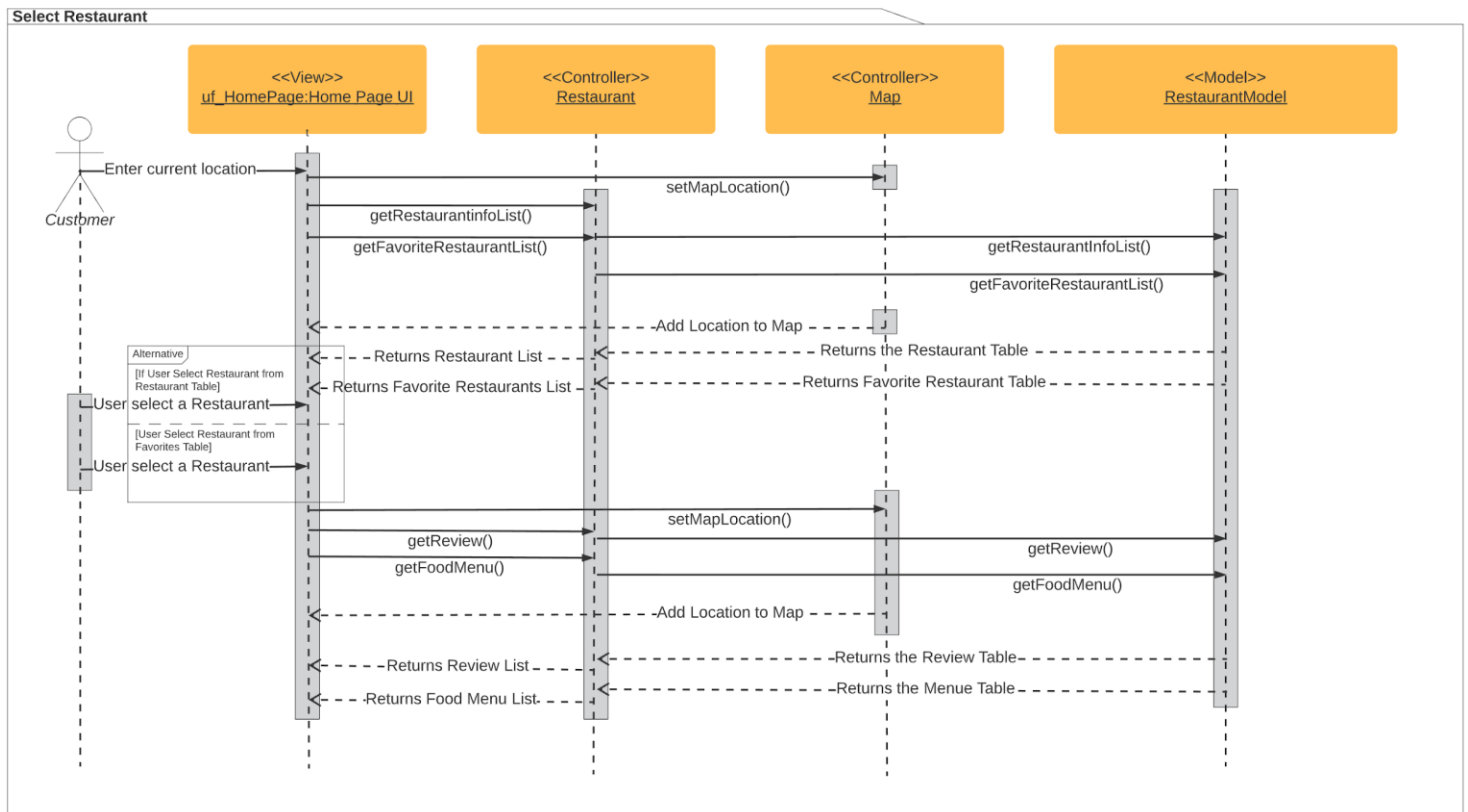
The user enters his or her current location in to the application. The application then retrieves all restaurants and favorite restaurants that are near the area as well as marking the current location on the map. The user then has the ability to select the desired restaurant and retrieve more information in the application. Table 4 displays the Use Case of Select Restaurant with alternative scenarios.

Use Case	#4 Select a Restaurant	
Goal in Context	Choose a Restaurant	
Scope	The Healthy Eats Application	
Level	Primary	
Primary Actor	Customer	
Preconditions	The customer has already logged in to the system	
Minimal Guarantee	The application failed to retrieve restaurant information	
Success Guarantee	The application successfully retrieves restaurant list and information	
Trigger	The customer submits current location	
Success Scenario:	Condition: Customer selecting a restaurant from Restaurant table	
	Step	Action
	1	The customer enters current location and press the Current Location button
	2	The system retrieves the list of restaurants near the area from the Restaurant Controller
	3	The system retrieves list of favorite restaurants near the area from the Restaurant Controller
	4	The system converts the current address to coordinates and input it to the map
	5	The customer selects a restaurant from the Restaurant Table
	6	The system retrieves and display the restaurant's review and rating
	7	The system retrieves and display the restaurant's menu
	8	The system converts the restaurant's address to coordinates and input it to the map
Extensions:	Branching Scenarios	
	Condition: Customer does not have an account yet in the system	
	Step	Action
5A	1	The customer selects a restaurant from the Favorites Table

**Table 4: Use Case - Healthy Eats Select a Restaurant**



### 3.4.2 Processing sequence for Favorites Function



**Figure 19: Sequence Diagram - Healthy Eats Select Restaurant**

### 3.4.3 Structural Design for Favorites Function

The User selects a restaurant from either the Restaurant table or Favorites table. The system submits the restaurant information to the Restaurant controller which then submits the data to the database. The Restaurant Model returns the data from the database which then returns to the Restaurant Controller to display the information. The address of the restaurant is sent to the Map Controller to display the location on the map. Refer to figure 20 for the structure of the Restaurant.

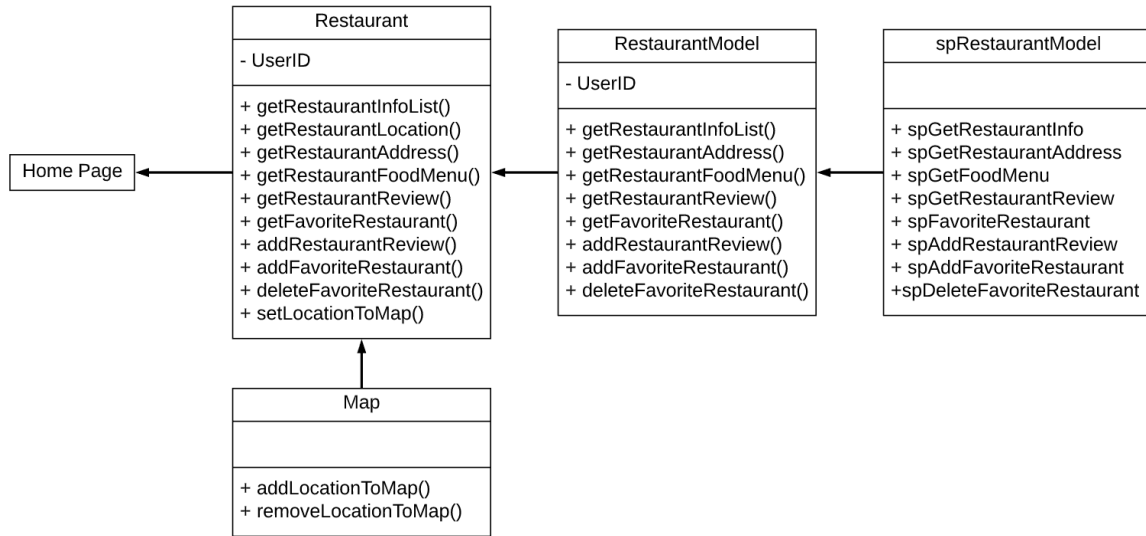


Figure 20: Class Diagram - Healthy Eats Restaurant (Select Restaurant)

#### 3.4.4 Key Activities

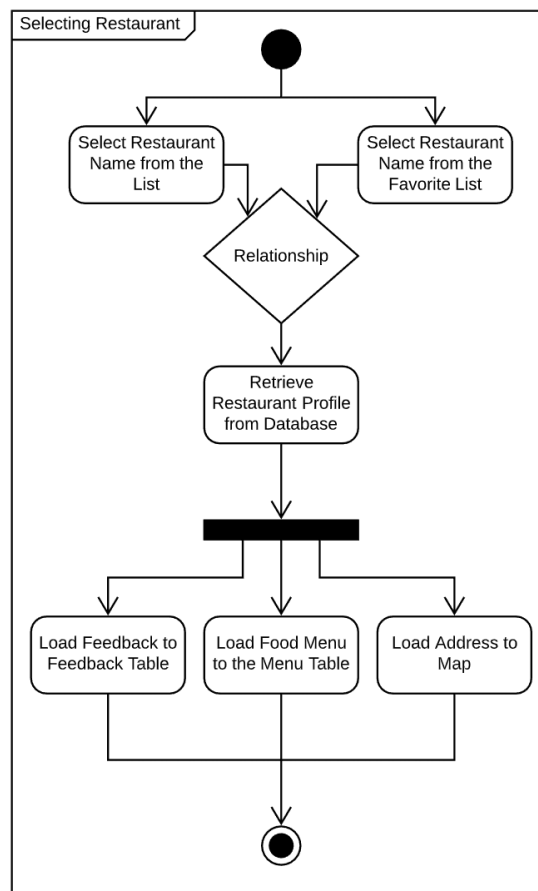


Figure 21: Activity Diagram - Healthy Eats Selecting Restaurant

### 3.3.5 Software Interface to other components

Home Page interface uses User, Restaurant, and Map controllers to submit information to RestaurantModel and SQL Server Stored Procedure from the RDS SQL database. The Restaurant Controller refreshes the Home Page interface once the transaction has been added to the database as well as the map.

## 4 User interface design

### 4.1 Interface design rules

Healthy Eats application user interface must comply to the color theme and font for all user interface.

### 4.2 Description of the user interface

Healthy Eats is a Windows application that uses Windows form for user interface. Log-in user interface is mainly used to authenticate user and create a new user profile.

#### 4.2.1 Log-In Page

Log-in user interface is composed of four events that a user can select: Sign Up, Log-in, Forgot Password, and Forgot Username.

##### 4.2.1.1 Log-In Screen Images

Figure 22 shows the interface screens of the Log-In page.



**Figure 22: Log-In User Interface**

##### 4.2.1.2 Log-In Objects and Actions

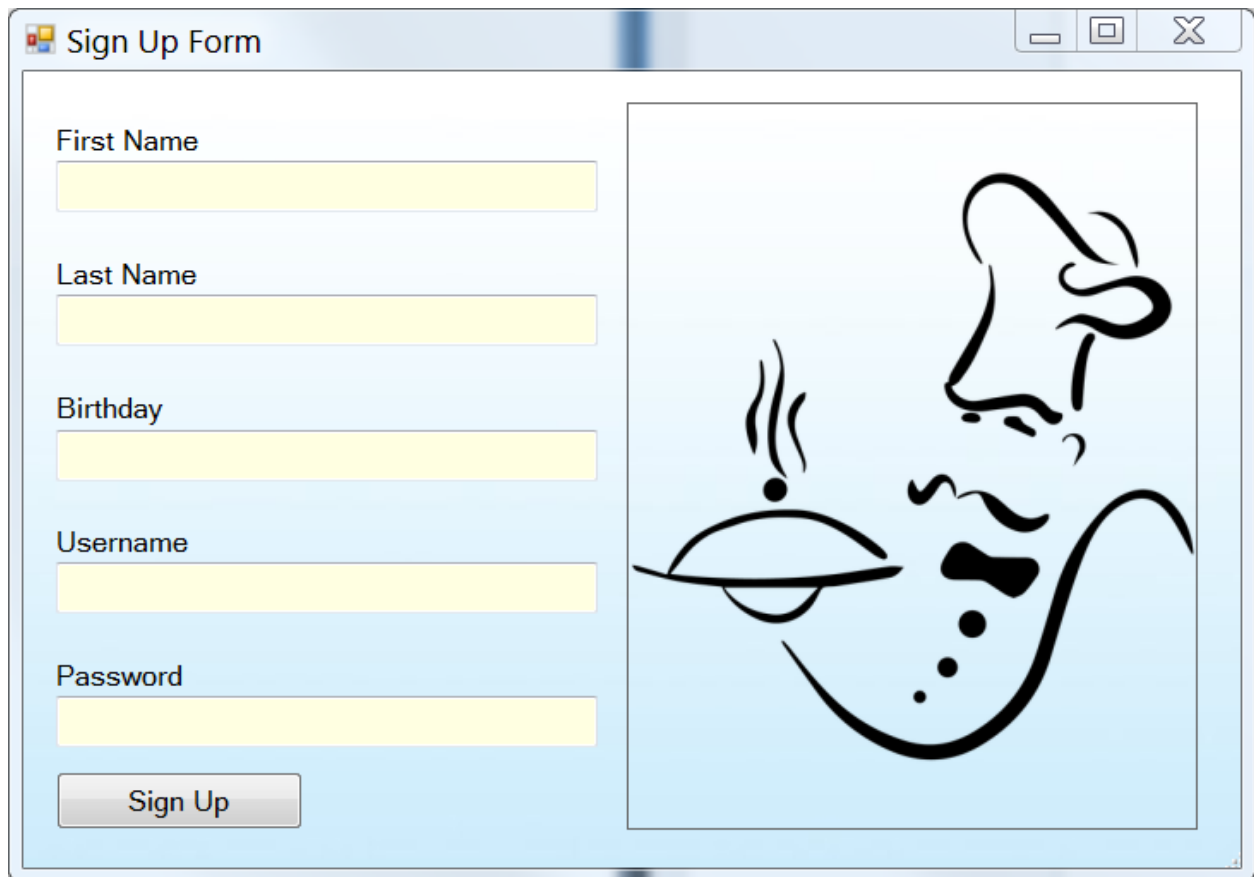
The goal of the Log-In user interface is to authenticate user credentials and provide a means for user to create a new profile. In addition, Log-In UI also provides a means for a user to retrieve username or password.

#### 4.2.2 Sign Up Page

The Sign Up User Interface is composed of entries that a user must fill in to create a profile. First Name, Last Name, select a Birthday, Username, Password along with the event, Sign Up that user can select after the user is done with filling in the information.

##### 4.2.2.1 Sign Up Form Screen Images

Figure 23 shows the interface screens of the Sign Up page.

The image shows a software window titled "Sign Up Form". On the left side, there are five text input fields stacked vertically, each with a label above it: "First Name", "Last Name", "Birthday", "Username", and "Password". Below these fields is a "Sign Up" button. On the right side of the window, there is a large rectangular area containing a stylized black line drawing of a person's face in profile, looking upwards and to the right. The drawing is minimalist, using only outlines and a few dots for features like the eye and nose.

**Figure 23 - Sign Up Form User Interface**

##### 4.2.2.2 Objects and Actions

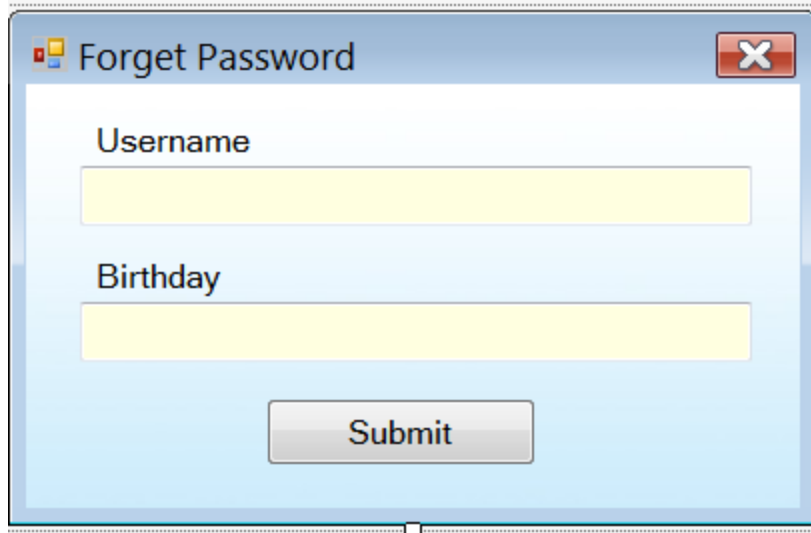
The goal of the Sign Up Form User Interface is to create a new user profile.

#### 4.2.3 Forgot Password Page

The Forgot Password User Interface is composed of the entries Username and Birthday along with the event, Submit, a user can select.

##### 4.2.3.1 Forgot Password Screen Images

Figure 24 shows the interface screen of the Forgot Password page.

A screenshot of a Windows-style dialog box titled "Forget Password". The dialog has a light blue background and a standard Windows window border with a close button (X) in the top right corner. Inside the dialog, there are two text input fields. The first field is labeled "Username" and the second field is labeled "Birthday". Both fields have a yellow background. Below the input fields is a single "Submit" button with a grey gradient and a 3D effect.

**Figure 24: Forget Password User Interface**

#### *4.2.3.2 Forgot Password Objects and Actions*

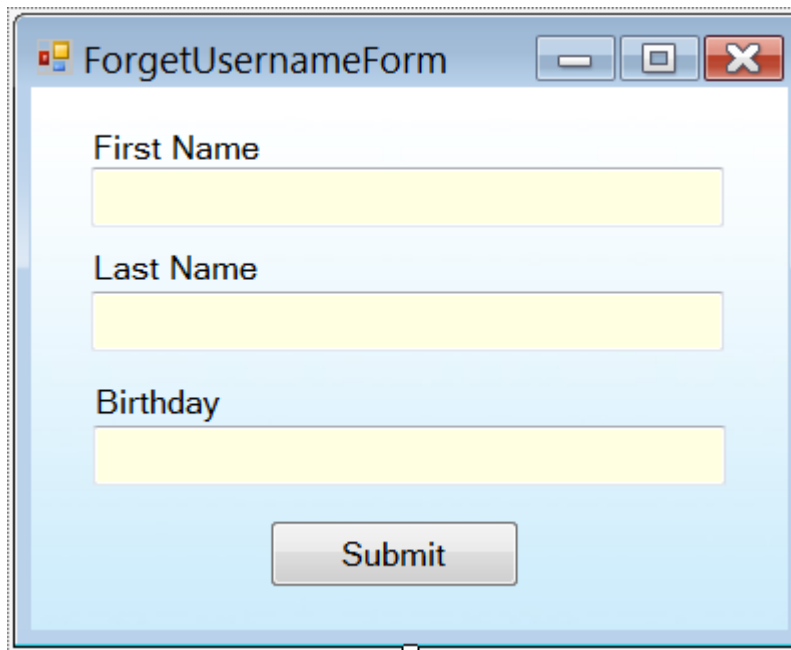
The goal of the Forget Password User Interface is to retrieve the users' password.

#### *4.2.4 Forgot Username Page*

The Forget Username User Interface is composed of the entries First Name, Last Name and Birthday along with the event, Submit, a user can select.

##### *4.2.4.1 Forgot Username Screen Images*

Figure 25 shows the interface screen of the Forget Username page.

A screenshot of a Windows-style window titled "ForgetUsernameForm". The window has a light blue background and a standard Windows window border with minimize, maximize, and close buttons in the top right corner. Inside the window, there are three text input fields. The first field is labeled "First Name", the second is labeled "Last Name", and the third is labeled "Birthday". All three fields have a yellow background. Below the input fields is a single "Submit" button with a grey gradient and a 3D effect.

**Figure 25: Forget Username Interface**

#### 4.2.4.2 Forgot Username Objects and Actions

The goal of Forgot Username User Interface is to retrieve the users' username.

#### 4.2.5 Home Page

The Home Page User Interface is composed of four sections of information including Restaurant list, Favorites list, Restaurant Feedback and Restaurant Menu.

##### 4.2.5.1 Screen Images

Figure 26 shows the interface screen of the Home page.

The screenshot shows a web application window titled "Home Page". It features a "Settings" link in the top right corner. The main content area is divided into several sections:

- Current Location:** A text input field and a "Current Location" button.
- Restaurant:** A table with columns "Restaurant", "Type", "Web Page", and "Address". An "Add to Favorites" button is located to the right of the table.
- Favorites:** A table with columns "Restaurant", "Type", "Web Page", and "Address". A "Remove From Favorites" button is located to the right of the table.
- Reviews:** A table with columns "Review" and "Rating".
- Menu:** A table with a single column "Food Menu".
- Map:** A large, empty rectangular area on the right side of the page.
- Footer:** An "Add Feedback" text input field, a "Rating" dropdown menu, and a "Submit" button.

**Figure 26: Home Page User Interface**

##### 4.2.5.2 Objects and Actions

The goal of the Home Page User Interface is to provide a restaurant search, restaurant list, a users favorite restaurant list, restaurants nearby on the map, restaurant feedback and the restaurants menus to the user.

## 5 Restrictions, limitations, and constraints

- Application uses C# language to develop the application.
- The database used in the application is controlled by Amazon Web Service which is a monthly subscription for service.

- The application only operates in Microsoft Windows 7 or later.
- The map provided by the application does support direction and location distance due to the limitation of the Google Map Application Console monthly subscription.

## 6 Testing Issues

Test strategy and preliminary test case specification are presented in this section. Additional test cases are located in the Appendix section in spreadsheet format.

### 6.1 Types of tests

The following list below are the definition of application testing type:

- (1). **Performance Test** – for example, to ensure that the response time for information retrieval is within an acceptable range. You typically should provide a specific performance bounds. For example, the search process should not take longer than 30 seconds.
- (2). **Accuracy Test** – for example, to determine if queries return the expected results.
- (3). **User Interface Test** – for example, to make sure the user interface is clear and easy to use with all types of users. Unfamiliar user can use the interface with minimal instruction and achieve the desired results.
- (4). **Security test** – for example, to ensure that users can only perform the tasks specified for their user group
- (5). **Repeatability Test** – for example, the software returns the same result for repeated queries.

### 6.2 List of Test Cases

Healthy Eats application testing specifications listed in **Table 5** are designed to meet the following testing types: Accuracy, Security, and Repeatability. The testing specifications are performed and verified by the tester(s) that the Healthy Eats application has met the user requirements with highest quality.

Test #	Type	Test Range	Test Input	Procedure	Expected Result	Pass /Fail	Tester (Initial)
TS1	Security	User Log-in	Username and Wrong Password	Enter username and wrong password to the Log-In form and press the Log-In button	Message inform user wrong password.	Pass	Ben/Dan /Xavier
TS2	Security	User Log-in	Username and Password	Enter username and wrong password to the Log-In form and	Launch Home page form	Pass	Ben/Dan /Xavier

				press the Log-In button			
TS3	Accuracy	Forgot Password	Username and Birthday	Select the Forgot Password from the Log-In form. Enter the username and birthday in the Forgot Password Form and press the Submit button.	Application returns user password through message box.	Pass	Ben/Dan /Xavier
TS4	Accuracy	Forgot Username	Firstname, LastName and Birthday	Select the Forgot Username from the Log-In form. Enter the first name, last name, and birthday in the Forgot Username Form and press the Submit button.	Application returns user username through message box.	Pass	Ben/Dan /Xavier
TS5	Accuracy	Sign Up	First Name, Last, Birthday, Username, Password	Select the Sign Up from the Log-In form and enter first name, last name, birthday, username, and password in the Sign Up form and press the Sign Up button.	Message inform user that it was successful.	Pass	Ben/Dan /Xavier
TS6	Repeatability	Current Location Entry	Address	Enter address into current location field and press the Current Location button.	Show the address on the map.	Pass	Ben/Dan /Xavier
TS7	Repeatability	Restaurant Selection	None	Shows restaurant Info for the menu, reviews and map.	The restaurant menu, the reviews and the location on the map.	Pass	Ben/Dan /Xavier
TS8	Repeatability	Favorite	None	Select a favorite	The favorite restaurant	Pass	Ben/Dan



		Restaurant Selection		restaurant from the favorite restaurant list.	menu, the reviews and the location on the map.		/Xavier
TS9	Accuracy	Add a Restaurant to Favorites	None	Select a restaurant from the restaurant list and press the Add to Favorites button.	The user should see the restaurant in the favorites box.	Pass	Ben/Dan /Xavier
TS10	Accuracy	Remove Restaurant from Favorites	None	Select a favorite restaurant from the favorite restaurant list and press the Remove From Favorites.	The user should see the restaurant out of the favorites box.	Pass	Ben/Dan /Xavier
TS11	Accuracy	Add Review	Feedback, Rating	Select a restaurant from the restaurant list and enter the feedback comment in the Add Feedback field and enter the restaurant rating.	The review should show in the review box for the restaurant.	Pass	Ben/Dan /Xavier
TS12	Accuracy	Add New Medical Condition	New Medical Condition	Select the Medical Profile in Settings from Home Page, and enter new medical condition and press the Add New Med Condition button.	Condition will be added to the medl condition drop down menu.	Pass	Ben/Dan /Xavier
TS13	Accuracy	Add Medical Condition to User Profile	Medical Condition	Select the Medical Profile in Settings from Home Page, and select medical condition from dropdown and press the Add Med Condition to Profile button.	Show a drop down menu of the user med conditions.	Pass	Ben/Dan /Xavier

TS14	Accuracy	Remove Medical Condition from User Profile	Medical Condition	Select the Medical Profile in Settings from Home Page, and select medical condition from dropdown and press the Remove Med Condition to Profile button.	The med condition should not show in the med condition drop down menu.	Pass	Ben/Dan /Xavier
TS15	Accuracy	Update User Profile	Password	Select the User Profile in Settings from the Home Page, update password, and press the Update button.	The user password should be updated to the new password and give user a message of success.	Pass	Ben/Dan /Xavier

Table 5: Test Specification Table

## 7 Appendices

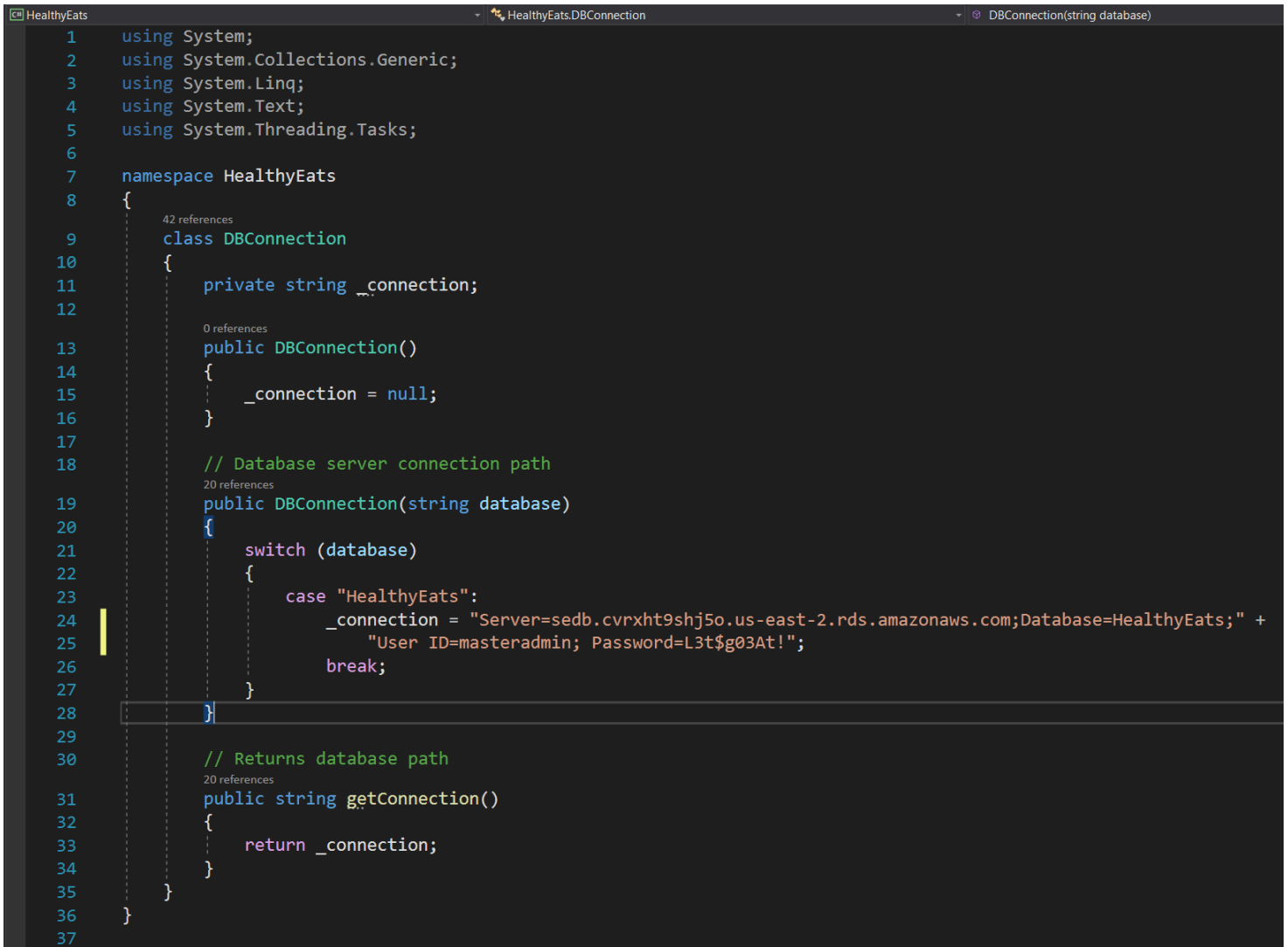
### 7.1 Packaging and installation issues

Healthy Eats application is developed using C# programming language that includes Google Map libraries and SQL connection to utilize map feature and SQL Server Database. The instruction below explains how the application is developed.

#### Database Setup:

- 1. Acquiring Server:** the team has opened an account on Amazon Web Service (AWS) to acquire free one year database subscription or 730hrs of usage.
- 2. Build Database:** in AWS, the team created a Relational Database Service (RDS) for Microsoft SQL Server Express Edition database, which provided database server and created a username and password.
- 3. Database Connection:** once the database has been created, the security group rules for CIDR/IP-Inbound and CIDR/IP-Outbound rule were set to 0.0.0.0/0 IP Address. This allowed the server to accept any IP address that will connect to the database. Although, the database was in a vulnerable position from unwanted clients, this was the best that the team can do due to the lack of knowledge in Networking Systems. In addition, System.Data.SQLite.Core in Visual Studio were downloaded in Visual Studio as Nuget Package for SQL Server Connection libraries, System.Data.SqlClient.
- 4. Database Management:** the team used the Microsoft SQL Server Management Studio to manage the database and build necessary tables, views, and stored procedures.
- 5. Database Connection Code Example:** Healthy Eats Application utilizes the MVC architectural pattern to manage the code, including database connection. *DBConnection* class was created to manage the database server connection path, which was used in the all Model Classes. Figure 27 shows how the *DBConnection* class was structured in the Healthy Eats application. Figure 28 shows structure on how to

connect to the database using Stored Procedure without return value. Conversely, Figure 29 shows how to connect to the database using Stored Procedure with return Data Table.



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace HealthyEats
8 {
9     42 references
10    class DBConnection
11    {
12        private string _connection;
13
14        0 references
15        public DBConnection()
16        {
17            _connection = null;
18        }
19
20        // Database server connection path
21        20 references
22        public DBConnection(string database)
23        {
24            switch (database)
25            {
26                case "HealthyEats":
27                    _connection = "Server=sedb.cvrxht9shj5o.us-east-2.rds.amazonaws.com;Database=HealthyEats;" +
28                        "User ID=masteradmin; Password=L3t$g03At!";
29                    break;
30            }
31        }
32
33        // Returns database path
34        20 references
35        public string getConnection()
36        {
37            return _connection;
38        }
39    }
40 }
```

Figure 27: Database Connection Class

```
// Add restaurant review
1 reference
public void addRestaurantReview(int restaurantID, string comment, int rate)
{
    DBConnection db = new DBConnection("HealthyEats");
    string sql = "spAddRestaurantReview";

    try
    {
        using (SqlConnection conn = new SqlConnection(db.getConnection()))
        {
            SqlCommand cmd = new SqlCommand(sql, conn);
            cmd.CommandType = CommandType.StoredProcedure;

            cmd.Parameters.Add("@restaurantID", SqlDbType.Int).Value = restaurantID;
            cmd.Parameters.Add("@comment", SqlDbType.NVarChar).Value = comment;
            cmd.Parameters.Add("@rate", SqlDbType.Int).Value = rate;
            conn.Open();
            cmd.ExecuteNonQuery();
            conn.Close();
        }
    }
    catch (Exception)
    {
        throw;
    }
}
```

Figure 28: Database Connection Using Stored Procedure Without Return Values

```
// Gets favorite restaurant list
1 reference
public DataTable getFavoriteRestaurant(string city, string state, int userID)
{
    DBConnection db = new DBConnection("HealthyEats");
    string sql = "spGetFavoriteRestaurant";

    try
    {
        using (SqlConnection conn = new SqlConnection(db.getConnection()))
        {
            SqlCommand cmd = new SqlCommand(sql, conn);
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.Add("@city", SqlDbType.NVarChar).Value = city;
            cmd.Parameters.Add("@state", SqlDbType.NVarChar).Value = state;
            cmd.Parameters.Add("@userID", SqlDbType.Int).Value = userID;

            conn.Open();

            SqlDataAdapter da = new SqlDataAdapter(cmd);
            DataTable dt = new DataTable();
            da.Fill(dt);
            da.Dispose();
            conn.Close();

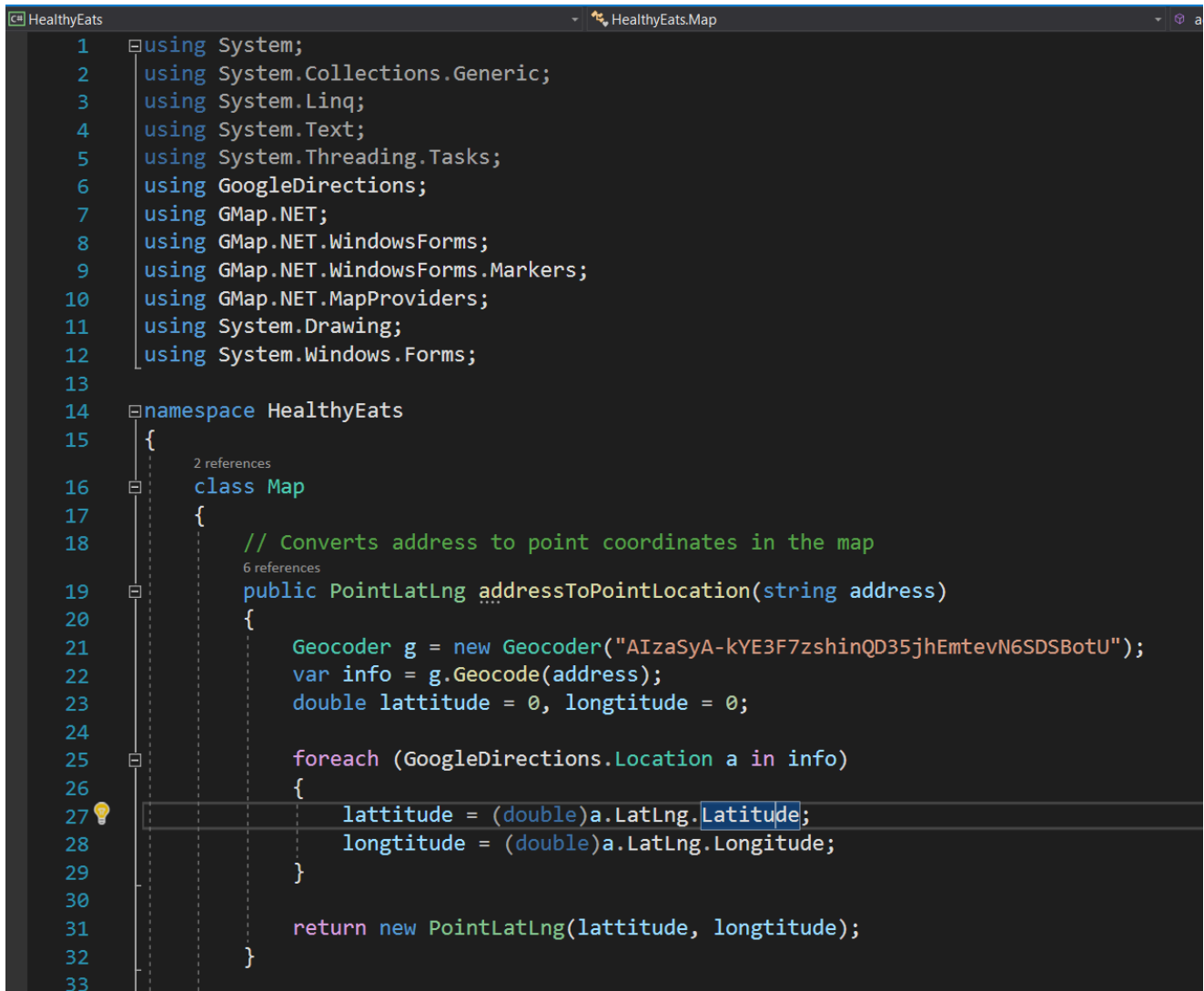
            return dt;
        }
    }
    catch (Exception)
    {
        throw;
    }
}
```

Figure 29: Database Connection Using Stored Procedure With Return Data Table

**Google Map Setup:**

1. **Acquiring API Connection:** the team has opened an account in Google Developer Console API and established a free limited time API service to be used for the Google Map feature. In addition, GoogleMapsAPI and GMap.NET.Windows were downloaded in the Visual Studio as Nuget Packages to utilize GoogleMap libraries.
2. **Using GMap Library:** GoogleMapsAPI package provided a built-in class to be used to create a map in a Windows form. Using Geocoder class, the API was passed in as parameter and this class created connection with Google Service to allow the application to use Google Map. Figure 30 shows how to convert address to map coordinates. Figure 31 shows how to set-up point marker on the map using

default pointers or custom point marker using an image converted into BitMap. Figure 32 shows how to input the coordinates and with a point marker into the map.



```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using GoogleDirections;
7  using GMap.NET;
8  using GMap.NET.WindowsForms;
9  using GMap.NET.WindowsForms.Markers;
10 using GMap.NET.MapProviders;
11 using System.Drawing;
12 using System.Windows.Forms;
13
14 namespace HealthyEats
15 {
16     class Map
17     {
18         // Converts address to point coordinates in the map
19         public PointLatLng addressToPointLocation(string address)
20         {
21             Geocoder g = new Geocoder("AIzaSyA-kYE3F7zshinQD35jhEmtevN6SDSBotU");
22             var info = g.Geocode(address);
23             double latitude = 0, longitude = 0;
24
25             foreach (GoogleDirections.Location a in info)
26             {
27                 latitude = (double)a.LatLng.Latitude;
28                 longitude = (double)a.LatLng.Longitude;
29             }
30
31             return new PointLatLng(latitude, longitude);
32         }
33     }
34 }
```

Figure 30: Converts Address to Map Coordinates

```
34 // Gets a pointer to the map
35 1 reference
36 public GMapOverlay getMarkerOverlay(string address1, bool primary)
37 {
38     PointLatLng p1 = addressToPointLocation(address1);
39     GMapOverlay markerOverlay = new GMapOverlay("marker");
40
41     if (primary == true)
42     {
43         // point in the map using marker
44         GMapMarker marker1 = new GMarkerGoogle(p1, GMarkerGoogleType.green_big_go);
45         markerOverlay.Markers.Add(marker1);
46     }
47     else
48     {
49         // marker setup
50         Bitmap icon = new Bitmap("Food Icon map.png");
51         Bitmap resized = new Bitmap(icon, new Size(icon.Width / 28, icon.Height / 28));
52
53         // point in the map using marker
54         GMapMarker marker1 = new GMarkerGoogle(p1, resized);
55         markerOverlay.Markers.Add(marker1);
56     }
57     return markerOverlay;
58 }
59
```

Figure 31: Set up a Point Marker in the Map



```
100 // Sets address to map
101 3 references
102 public void setMapLocation(string address, GMapControl gmMap, bool primary)
103 {
104     if (address == "") return;
105     //to remove markers
106     if (primary == true)
107     {
108         while (gmMap.Overlays.Count > 0)
109         {
110             gmMap.Overlays.RemoveAt(0);
111         }
112     }
113     else
114     {
115         while (gmMap.Overlays.Count > 1)
116         {
117             gmMap.Overlays.RemoveAt(1);
118         }
119     }
120
121     // map setup
122     gmMap.DragButton = MouseButtons.Left;
123     gmMap.MapProvider = GoogleMapProvider.Instance;
124     GMaps.Instance.Mode = AccessMode.ServerOnly;
125     PointLatLng p1 = addressToPointLocation(address);
126
127     gmMap.Position = p1;
128     gmMap.Overlays.Add(getMarkerOverlay(address, primary));
129     if (primary == true)
130         gmMap.Zoom = 15;
131     else
132         gmMap.Zoom = 13;
133
134     gmMap.Refresh();
135 }
```

Figure 32: Input Coordinate and Point Marker to the Map

## 7.2 User Manual

The following information below are the key instruction on how to operate the Healthy Eats application.

### **Sign-Up:**

1. Click the Sign Up button.
2. Enter your first and last name.
3. Enter your birthday.
4. Enter what you want your username and password to be.
5. Then click the Sign Up button

### **Login:**

1. Launch the Healthy Eats application.
2. Enter the username and password.
3. Click Forget Username to retrieve your username by entering your first, last name and birthday then click submit.
4. Click Forget Password to retrieve your password by entering your username and birthday then click submit.

### **Home Page Interface:**

1. Launch the Healthy Eats application.
2. Enter your username and password, and press the Log-In button to open Home Page.
3. Enter your current location in the Current Location field using comma to separate city and state, and then press the Current Location button.
4. Select a restaurant from the Restaurant table to retrieve more information about the restaurant.

### **Favorite Restaurant:**

1. From the Home Page, select the desired restaurant from the Restaurant Table and press the Add to Favorites button to add a restaurant to the favorites list.
2. Conversely, to remove a restaurant from the favorites list, select the restaurant from the Favorites Table and press the Remove from Favorites button to remove the restaurant from the favorites list.

### **Medical Profile:**

1. From the Home Page, select the Medical Profile in the Setting tab.
2. To add a medical condition, select a medical condition from the Medical Condition dropdown and press the Add Med Condition to Profile button.
3. To remove a medical condition, select a medical condition from the Medical Condition dropdown and press the Remove Med Condition to Profile button.
4. To add new medical condition, enter the new medical condition in to Add Medical Condition field and press the Add New Med Condition button. The new medical condition will be added to the list and can now be selected.

### **User Profile:**

1. From the Home Page, select User Profile in the Settings tab.
2. Update any of the user information and then press the Update button.

### 7.3 Open Issues

The following features for the Healthy Eats application were considered, but due to the time constraint those were not included in the application development and maybe a future enhancement of the application.

- Ability to search as restaurant
- Ability to inactivate a user profile
- Create a separate Favorite page
- Include image of the food in the table
- Ability to get food descriptions and food category
- Ability to see the price of the food
- Recommendations of the restaurant food base on your medical profile
- Ability to get directions to the restaurant

### 7.4 Lessons Learned

The following information below are the key items that the team found as lessons learned during the project development.

- **Team Collaboration and Synchronization:** we learned how to develop an application with multiple developers and established project scope and timeline.
- **Time Management:** we learned how to utilize different ways of communication and resolving conflicts.
- **Adaptability:** we learned how to quickly use new program languages (C# and SQL).
- **Formal Documentation:** we learned how to write formal project documentation and diagrams.
- **Database Utilization:** we learned how to build one from scratch and how to use it.
- **New Features:** we learned how to make a google map.

#### 7.4.1 Design Patterns

Healthy Eats application utilized Model-View-Controller (MVC) and Client-Server software architectural pattern. Client-Server was chosen because the application was using a database, and MVC was chosen because the application contained multiple user interface that required the same source of data, which was stored in a database.

#### 7.4.3 Team Communications

The team established twice a week one hour on-campus meeting to talk about project progress. In addition, the team used Trello for Project phasing, Github for storage, and Discord for communication. However, the team added Discord Live Stream and Screen Share, Google Drive, and Lucid Chart to work on the project, assignments, and application development. The Live Stream and Screen Share allowed the team to meet and result questions regarding the project quicker which helped speed up the project completion.

#### 7.4.4 Task Allocations

The team has established an understanding of balanced task contribution throughout the semester. This includes project prioritization, design, development, documentation, and presentation. This also includes group

assignments and diagram creation. Although we used Trello for project phasing, as a team, we could further utilize this tool to improve our task allocation and project management.

#### 7.4.5 Desirable Changes

- **Ben** - Given another month on the project I would have wanted to work on more of the less important components that were nice to have in the project. Help make the addition of a search for restaurants using the users medical conditions and food type preferences.
- **Dan** - I would like to improve the security of the database since the current status is open for everyone. In addition, I would like to add the map direction and distance between to places on the map to further the usage of the map in the application.
- **Xavier** - I would like to work more on the food menu so it can show an image of the food is and the description of the food to give the user more information on it.

#### 7.4.6 Challenges Faced

- **Ben** - Implementing has to be the hardest but time consumption can vary the same as the specifications and design. This being one of the first bigger projects I've been apart of with the amount of code we have so far has been harder than even very new requirements specification which seemed easier to understand in the long run.
- **Dan** - The hardest task for me during the project development was the system implementation because I did not have experience and knowledge on how to build an independent database that can be accessed publicly as well as using a map in an application. These two tasks required a lot of time researching online to build a working system.
- **Xavier** - I would say my hardest part was to format the code and implement it to the program specifies parts to make the program run. I had to learn about the models, controllers and views parts so that I know how to make a program. I also had to learn how to make a database and how to connect the two together.