FOOD REVIEW DEVELOPER GUIDE

VERSION 1.0

# Introduction and Background

* 1. **Introduction**

Food Review is food management software. Food Review was motivated through my frustration with modern food system software. In my experience in higher education, most of this software tends to either be overly complicated, or it fails to provide correct functionality that course management software should provide. Food Review was built with the intention to alleviate these issues. I have designed it so that there is as small a learning curve as possible. The user interface of Food Review has been designed so that it requires as little input as possible from the user.

Users can register themselves by using register us link in the login page, so that they can view the food review based on products list and ingredients once user login into the application. It is meant to delineate the features of the new application such as register us, Login Page and Food review page which help the to know the food review and health benefits for the category-based products based on the Ingredients, for this users needs to register themselves and login into the application with the details

* Login Page --- User can login into the application.
* Register Page---User need to register themselves to view the food review and comments.
* Food Review--- To view the food review either Good or Bad based on Product and Ingredients.

# Objectives

As state earlier, Food Review is food management software. That said, Food Review application implements most of the expected functionality of Food Review management software. It does not implement the functionality found in a product like provides the following functionality:

* + - Users can Login into their application:
      * Username is required.
      * Password is required.
    - Users can register themselves to get the login details:
      * First name of the user
      * Last Name of the user
      * Username to access the application.
      * Password to access the application.
      * Address of the User
    - Users can view/manage the food review system:
      * Category
      * Product
      * Ingredients
      * Food Review for the selected food ingredient
      * Health Benefits for selected food ingredient

# Risks

This section discusses project risks and the approach to managing them.

Project risks are events or circumstances that can potentially have a negative impact on a project's objectives, such as its scope, schedule, budget, or quality. Identifying and assessing project risks is a crucial part of project management.

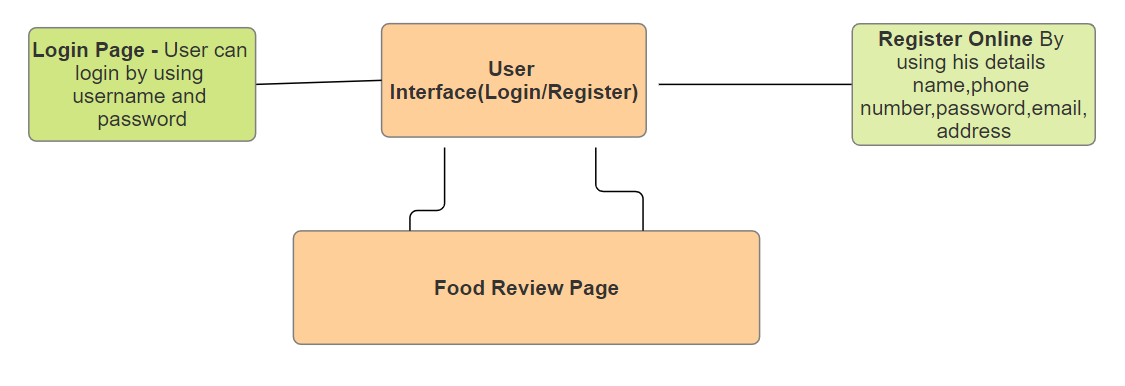
* Scope Creep
* Resource Constraints
* Schedule Delays
* Quality Issues
* Technical Challenges
* Communication Breakdown

|  |  |  |  |
| --- | --- | --- | --- |
| **Name of risk** | **probability** | **impact** | **RM3 pointer** |
| Scope Creep | less | Impact on Design of the requirement | Initial |
| Resource Constraints | less | Impact on Schedule Timings | Optimized |
| Schedule Delays | medium | Impact on Timely Deliverable | Defined |
| Quality Issues | medium | Impact on Output of the requirement | Optimized |
| Technical Challenges | medium | Impact on Proper Coding & testing | Initial |
| Communication Breakdown | medium | Impact on Design of the requirement | Initial |

# 2.0 Architecture

* 1. **Software Architecture**

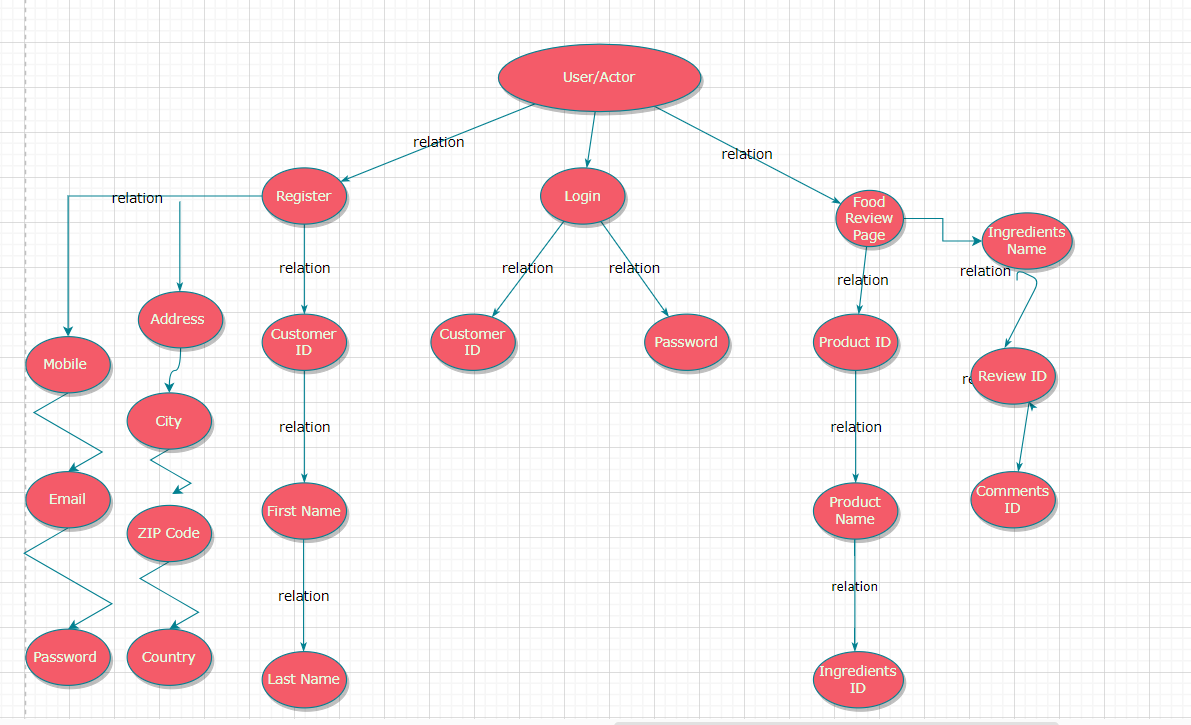
When designing Food Review Application, I explicitly chose to utilize a Layered architecture. I had briefly considered employing the MVC architecture pattern, however time constraints on the project alongside my inexperience with the pattern in ASP.NET led to me choosing a Layered architecture instead. This has led to some curious implementation details whereby the system mimics an MVC architecture (stressing that this is not an implementation of MVC).

The architecture diagram for Food Review Application is shown below. Proceeding the architecture diagram is a description of all the components that make up the diagram.

*Figure 1: food Review Architecture Diagram*

# 2.1 Database Design

Food Review follows a relational database model that approximates third-normal form. There were some decisions regarding certain tables where I thought it would be more prudent to just repeat data in a column rather than create a small (3-7 row) table and below shows:

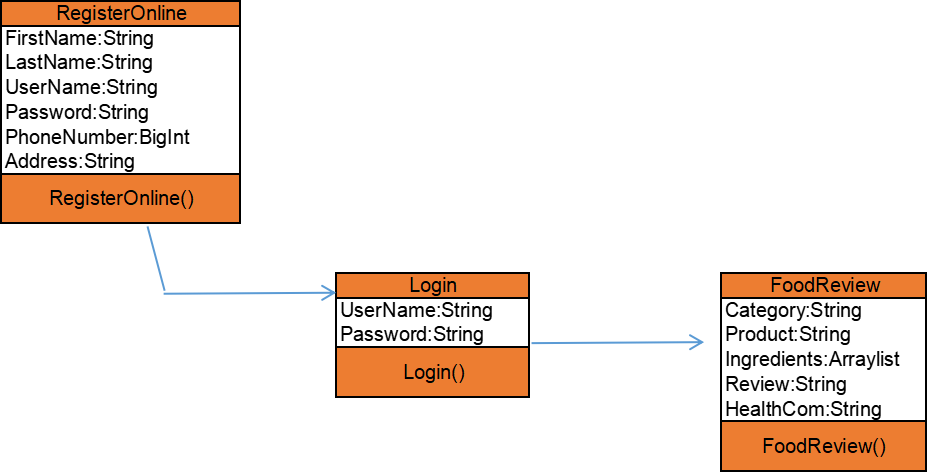


Description of the Tables Present in the Food Review Application ERD

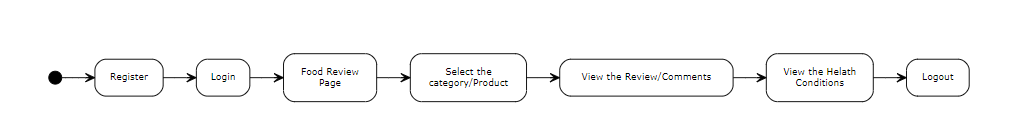
|  |  |
| --- | --- |
| **Users** | **FoodCategory** |
| Customer ID: Int | Product ID: Int |
| First Name: String | Product Name: String |
| Last Name: String | Ingredients ID: Int |
| Address 1: String | Ingredients name: String |
| City: String | Review:  String |
| ZIP Code: Int | Comments:String |
| Country: String | CategoryName : String |
| Mobile Number: BigInt |  |
| Email: String |  |
| Password : String |  |

# Class Level Design

This section explains every class present in Food Review, as well as the interactions between them. I will also provide rationale behind the design of several of the more complicated classes in the system. The classes listed below are organized as they appear in the Visual Studio solution: First by folder, then by class.



**Static model**



**Dynamic Model**

# 2.1 Cache

# This folder contains classes related to an in-memory cache.

# 2.1.1 Cache Item

Implements a key-value pair cache item. Both the key and value are generic for increased flexibility. A cache item implements methods for accessing and mutating both the key and value. It is the primary class utilized by the Memory Cache class.

# 2.1.2 Memory Cache

Implements an in-memory cache that stores key-value pair Cache Items. The Memory Cache class implements an aging mechanism for paging. It also utilizes the C# delegate design model to dynamically invoke a method handler for page events. This provides a direct benefit for classes that use the Memory Cache class in that they can catch objects as they are paged out of cache and handle them appropriately. The Memory Cache class employs all of the standard operations one would normally expect from a software-based cache.

# 2.2 Classes

This folder contains classes related to data bearing classes as well as some miscellaneous classes.

# 2.2.1 User

This class is the most important class utilized by Food Review. It provides information regarding users such as their name, email, username, and various other pieces of information. A user is registered with the system whenever they login and logout whenever they log off the system. This prevents a user from having more than one active session. It also prevents a user from manipulating the web controls client-side to view pages their role does not permit them to view.

# 2.2.2 Register

This class is the most important class utilized by Food Review. It provides information regarding users such as their name, email, username, and various other pieces of information. A register is used to get the login details into the food review application and can be stored in the database users table for tracking purpose, without this user can’t login into the application

# 2.2.3 Food Review

This class is the most important class to know the food review and food health benefits for the selected product It provides information regarding users such as their product, ingredients, food review and health benefits for the selected ones. A user needs to select the product to get the food review and health benefits for the selected ones.

# Software Interfaces

Food Review utilizes several outside software packages to accomplish its objectives.

When installing Food Review, the maintainer must be certain (in most cases) that the following software packages are installed on the hosting system:

* MY SQL Server Express: This software ships with Visual Studio (if the proper packages are installed with it). SQL Server Express was used during development as the database provider for Food Review. It is highly suggested that the maintainer transfers the database schema to a full-fledged SQL Server or Azure instance on dedicated hardware. It is not recommended to run Food Review on the same physical hardware that the database exists on.
* jQuery\*: Provides advanced JavaScript functionality to the system and supports the bootstrap theming system.
* Bootstrap\*: Provides basic theming capabilities to the system.
* NetBeans: Food Review utilizes Net beans for running the application, the maintainer must have latest version three installed if they wish to perform automated unit testing on the Food Review software.
* Tomcat Server: Used to host the application in the server which can be accessed by the local host URL.
* GitHub Version Control: Available at https://github.com, this web service hosts the project. The maintainer must be familiar with its operation.

# Conclusion

* 1. **Remarks on Implementation**

This section contains my remarks regarding the current implementation of Food Review available at: https://github.com. This implementation is nearly complete (I estimate that ~95% of the functionality outlined in the Requirements Spec. is met). For instance, advisors cannot currently override prerequisites for their advisees. Likewise, I ran out of time to complete the theming of the system. It has partially met the layout specified in the Design Specification, however the arrangement of the controls within the system are not as I had specified.

Also, due to the time constraints and limited time I personally had to work on this project, I made some implementation decisions that are not optimal from a performance standpoint. I have tried to outline these areas within the source code of the system.

However, I may have missed some areas. I have tried my best to provide developer comments for every method but there are some methods that I have missed (the stubs are there, they but they are not filled in).

# Possible Future Improvements

N/A