Section 1 - Project Description

1.1 Project

Smart Shop

1.2 Description

Brief overall description of the project

A computer generated shopping list that is personalized to each user's nutritional needs, financial situation, and food goals. Minimizing the time and hassle required to produce, create, and shop for a weekly meal plan.

1.3 Revision History

Date	Comment	Author
10/26/23	Added UML charts in section 8.,1 and added new sections 4.1, 6.5	Jad Kang
11/11/23	Edited Use Case Diagram and added Tech Stack (section 9.1)	Jad Kang
11/14/23	Updated System Architecture Diagram (section 3)	Ryan McArthur

Contents

Section 2 - Overview

2.1 Purpose

Brief description of the focus of this module of the overall project and its intended audience.

This project is meant to be a shopping assistor for anyone who doesn't enjoy coming up with meals or recipes. Our goal is to provide an individualized shopping list for each user that takes into account their purchase history, preferred type of food, and allergy information taking out much of the thought a person needs to put into what they plan to eat for the week. The intended audience ranges anywhere from young adults to the average adult looking to save time.

2.2 Scope

Describe the scope of the module to be produced

Our product is going to be able to come up with a meal plan and tell the user what ingredients it should purchase based on user given criteria such as allergy info, caloric information, and overall food preferences. Our product will not be able to order the products for the user at this given point, it will be unable to account for the exact price of the products to be purchased as prices vary based on store, location, and brand. With a teaming up of stores and other services our product could meet these areas however within our given design they will not be available.

2.3 Requirements

Your mileage may vary -- we typically break down the requirements to provide a ballpark estimate.
User interface requirements: The user interface intends to be simple and user friendly. Each page only has a few areas to input and every item on the page is clearly labeled. On the login screen there will be a short description of Smart Shop to inform users of what the website does. There will also be a one sentence guide on how to use it. The edit profile button will be represented with a simple filled outline of a person and will be on each page. The shopping list button will appear after selecting a meal plan and will remain on each subpage. It will be represented with a list icon.
Hardware interface requirements: Software requires an internet connection to be able to access. It can be run on any PC or mobile device that can load a webpage on an internet browser.Requires a mouse or substitute to be able to click on buttons and input boxes. Requires a keyboard to be able to type in input boxes.
Software interface requirements: Any web based application that can be accessed by web browser. Frontend/backend both use the .net framework. APIs that give information on recipes. Also APIs for grocery outlets and what products they have and the price of products.
Communication interface requirements: We will use embedded forms for communication between client and server. We will also need to communicate with Google's login services.

2.3.1 Estimates

#	Description	Hrs. Est.
1	Create UI based off of UX	20
2	Align backend processes with UI	20
3	Ensure database functionality with system	30

4	Integrate with third party recipe provider	20
5	Test product	30
	TOTAL:	120

2.3.2 Traceability Matrix

Cross reference this document with your requirements document and link where you satisfy each requirement

SRS Requirement	SDD Module
Req 1	Section 7
Req 2	Section 9
Req 3	Section 4/9
Req 4	Section 3/8.1

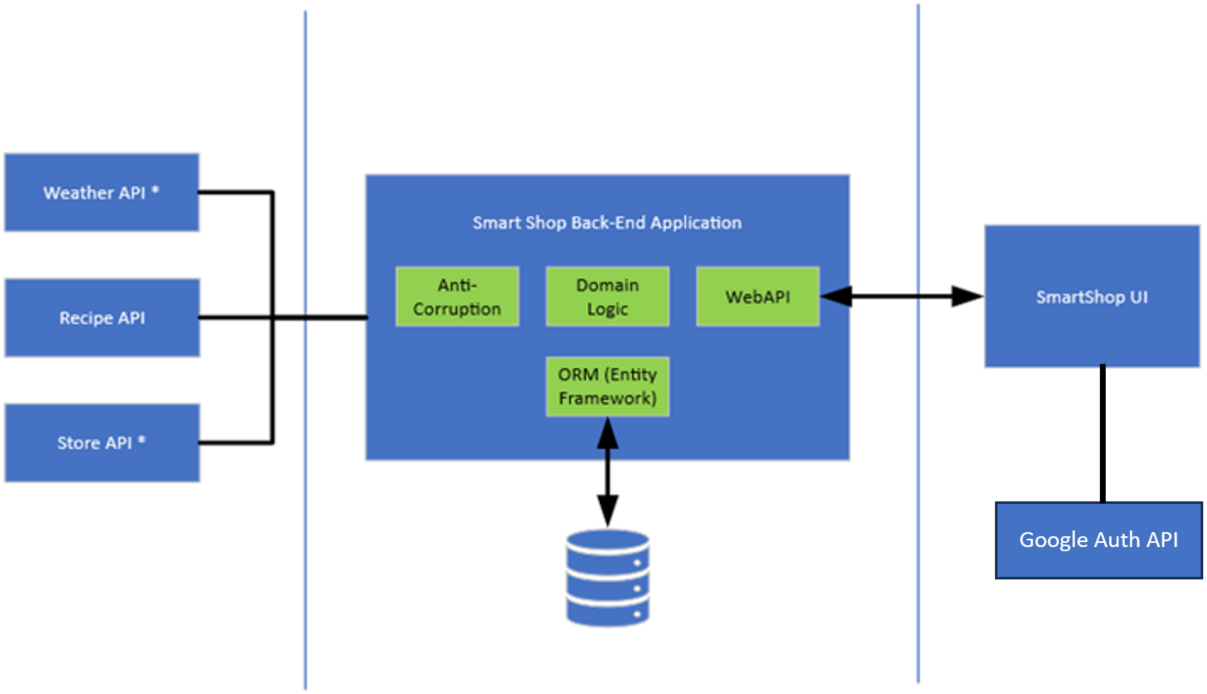
Section 3 - System Architecture

System will be developed using Microsoft .Net Core 6.0 (LTS). It will be generally segmented into three distinct areas: The back end, front end, and external services. External services will be linked into the back end using an anti-corruption layer which will translate a service into local domain models. The back end will manage local database transactions and provide endpoints for the UI to retrieve data as well as post data back for processing and potentially storage in our database.

The front end will handle interactions with the user. The typical page cycle will consist of retrieving information from the user, posting it to the back end through the API and then retrieving and displaying the result to the user. The front end will also be responsible for connecting to the Google Authentication provider for user authentication.

Communication between areas will be secured using the following mechanisms:

- UI to Google Authentication: Client Id and Shared Key
- UI to Back End Application: Shared Secret (key) - passed via request header.
- Back End to Recipe API: Client Id and Shared Key
- UI to user: Standard https protocol.



* Weather API and Store API are for future functionality. Store API would be used to send shopping lists to a store for instant purchase and monetization. Weather API would be used to correlate recipe choices with weather trends for refinement of suggested meals.

Section 4 - Data Dictionary

User (local database)

Field	Notes	Type
ID	Unique Identifier using GUID	UniqueIdentifier
Email	User Email Address (for lookup from google login)	VARCHAR
Allergies	List of allergies to avoid in recipes (stored as a JSON list)	VARCHAR
DietTypes	JSON list of users diets (Vegetarian, Carnivore, Balanced, etc)	VARCHAR
MealPlans	Reference to meal plans the user has	Reference

GoogleUser (from Google authentication)

Field	Notes	Type
ID	Unique Identifier	VARCHAR
Email	User Email Address	VARCHAR
Name	Users DisplayName	VARCHAR
GivenName	Users First Name	VARCHAR
SurName	Users Last Name	VARCHAR
Picture	URL to the users profile photo (hosted on Google)	VARCHAR

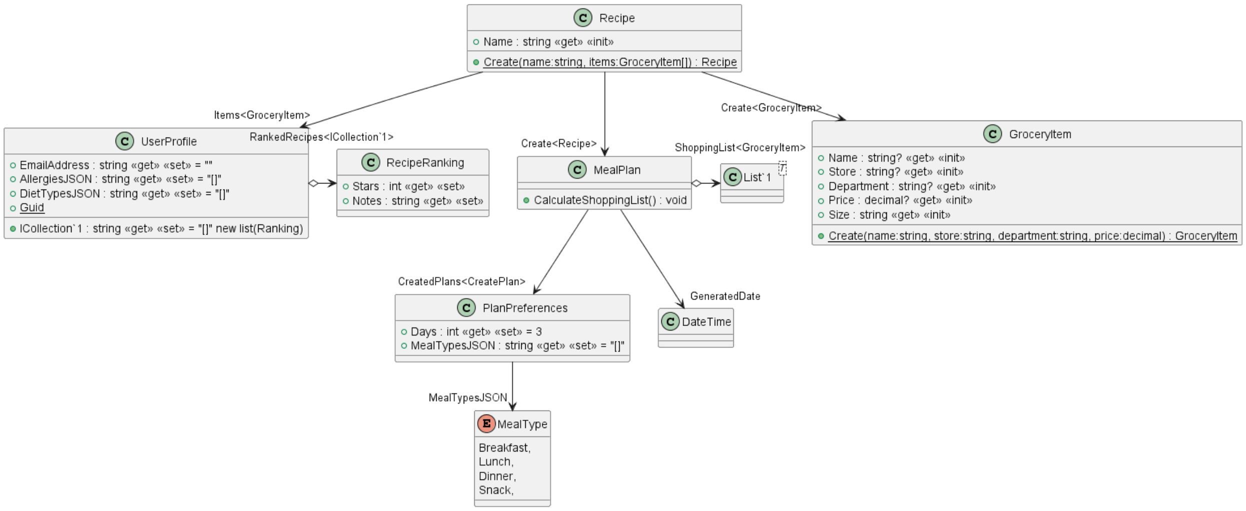
MealPlan (local database)

Field	Notes	Type
UserId	User who generated the meal plan	UniqueIdentifier
Date	Date the meal plan was created for	DateTime
Recipes	Reference to recipes used in the meal plan	Reference
Groceries	Calculated List of Groceries based on the recipes	

MealPlan_Recipe (local database)

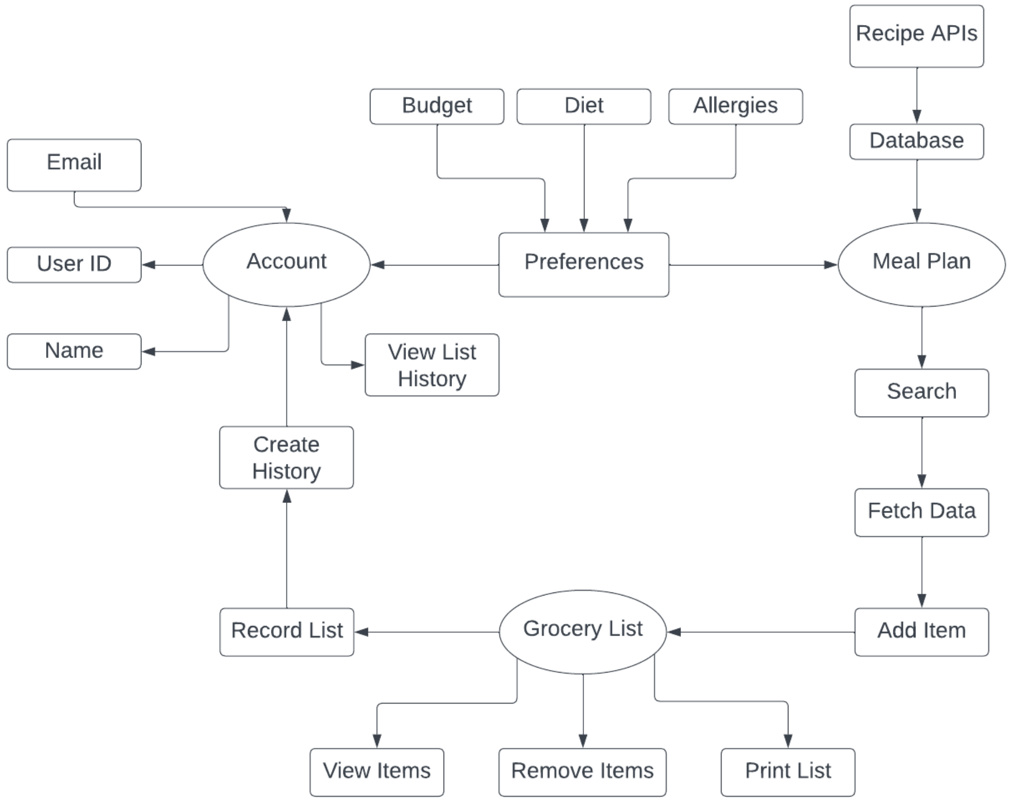
Field	Notes	Type
MealPlanId	Meal Plan this recipe is for	UniqueIdentifier
RecipeId	Reference to the recipe	VARCHAR
RecipeSource	Link to the recipe source (service provider)	VARCHAR
MealType	Breakfast, Lunch, Dinner or Snack	INT

4.1 - Class Diagram


Recipe
Name : string «get» «init»
Create(name:string, items:GroceryItem[]) : Recipe
Items<GroceryItem>
Create<GroceryItem>
UserProfile
EmailAddress : string «get» «set» = ""
AllergiesJSON : string «get» «set» = "[]"
DietTypesJSON : string «get» «set» = "[]"
Guid
ICollection`1 : string «get» «set» = "[]" new list(Ranking)
RankedRecipes<ICollection`1>
RecipeRanking
Stars : int «get» «set»
Notes : string «get» «set»
Create<Recipe>
MealPlan
CalculateShoppingList() : void
ShoppingList<GroceryItem>
List`1
T
GroceryItem
Name : string? «get» «init»
Store : string? «get» «init»
Department : string? «get» «init»
Price : decimal? «get» «init»
Size : string «get» «init»
Create(name:string, store:string, department:string, price:decimal) : GroceryItem
CreatedPlans<CreatePlan>
PlanPreferences
Days : int «get» «set» = 3
MealTypesJSON : string «get» «set» = "[]"
GeneratedDate
DateTime
MealTypesJSON
E
MealType
Breakfast,
Lunch,
Dinner,
Snack,

Section 5 - Software Domain Design

5.1 Software Application Domain Chart



5.2 Software Application Domain

Account: Holds information about the user to improve the functionality of the website.

Meal Plan: Helps the user find recipes and creates a grocery list based on account information.

Grocery List: Displays the items that are on the grocery list and prints a finalized version.

5.2.1 Domain X

Account Domain: Once an email is given by the user and stored, an account is created with a unique user ID and the user's full name. Preferences can also be given to help narrow down search results as the user is browsing the recipe database. Past grocery lists can be viewed, so that users can easily make past grocery lists without having to search for everything again.

Meal Plan Domain: Recipes are stored within the database. The search function looks through the database to find recipes that adhere to the user's preferences and the keywords used. Once a recipe is found, a function will extract data from the recipe like a description of what the recipe is, the ingredients used to make it, and the total calories of the recipe. If that recipe is what the user is looking for, an add function will add that recipe to the grocery list along with all the ingredients needed to make it.

Grocery List Domain: Users will be able to view the items that are on their grocery lists and remove any ingredients that they decide they don't need. Once a user is satisfied with their grocery list, they will be able to print a finalized grocery list. The grocery list will then be cleared and the user can then create a new grocery list whenever they are ready. Past grocery lists will be stored to a user's account and can be viewed at a later time.

5.2.1.1 Component Y of Domain X

Budget, diet, and allergy information is stored within preferences of the account domain.

Recipe APIs are stored within the database for the meal plan domain.

5.2.1.1.1 Task Z of Component Y1 of Domain X

Budget is how much money a user is willing to spend for their groceries. Diet consists of what the user is willing to eat with examples being keto, vegetarian, etc. Allergies are the types of food that the user is allergic to and filters out search results that contain anything with those items.

Recipe APIs are pulled from multiple websites to fill the database. Recipes will then be able to be searched for based on keywords and ingredients.

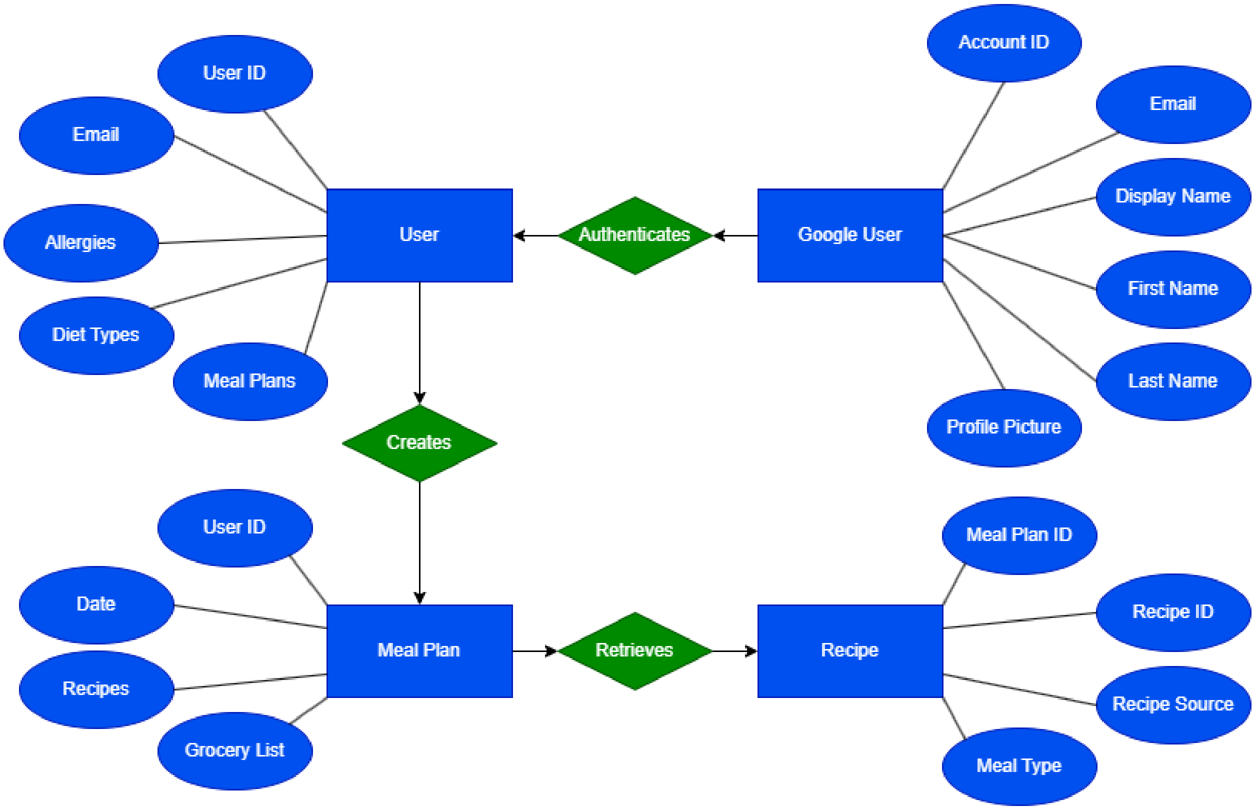
Section 6 – Data Design

Describe the data contained in databases and other shared structures between domains or within the scope of the overall project architecture

The data contained in databases include user account information such as food preferences, allergy restrictions, and dietary goals. The data also includes food data such as names of items, prices, nutrition information, and recipe information

6.1 Persistent/Static Data

Describe/illustrate the logical data model or entity relationship diagrams for the persistent data (or static data if static)



6.1.1 Dataset

Describe persisted object/dataset and its relationships to other entities/datasets

Google User:

Persisted data: “Account ID”, “Email”, “Display Name”, “First Name”, “Last Name”, “Picture”.

Relationship: Provides authentication and account information for “User” object.

User:

Persisted data: “User ID”, “Email”, “Allergies”, “Diet Types”, “Meal Plan”.

Relationship: Supplies information required for the creation of the “Meal Plan” object.

Meal Plan:

Persisted data: “User ID”, “Date”, ”Recipes”, ”Grocery List”.

Relationship: Retrieves information from “Recipe” object and holds main meal plan data.

Recipe:

Persisted data: “Meal Plan ID”, “Recipe ID”, “Recipe Source”, “Meal Type”.

Relationship: Retrieves recipe information from API and supplies data to “Meal Plan” object.

6.1.2 Static Data

Describe static data

Static data includes predefined, unchanging diet types, allergies, and meal types (e.g.breakfast, lunch, dinner).

6.1.3 Persisted data

Describe persisted data

Persisted data includes user account information, user inputted information such as specific dietary criteria, recipe information, and meal plan information such as the resulting grocery list.

6.2 Transient/Dynamic Data

Transient data includes temporary data that goes into meal plan calculation such as nutritional values calculated to meet a specific meal plan (e.g., caloric number restrictions).

6.3 External Interface Data

Any external interfaces’ data goes here (this is for the data, section 8 is for the interface itself)

External interface data:

User account information (account ID, email, display name, first name, last name, and picture) from Google Authentication.

Recipe information (Recipe Source: recipe name, ingredients, price, nutritional information) from Spoonacular API.

6.4 Transformation of Data

Describe any data transformation that goes on between design elements

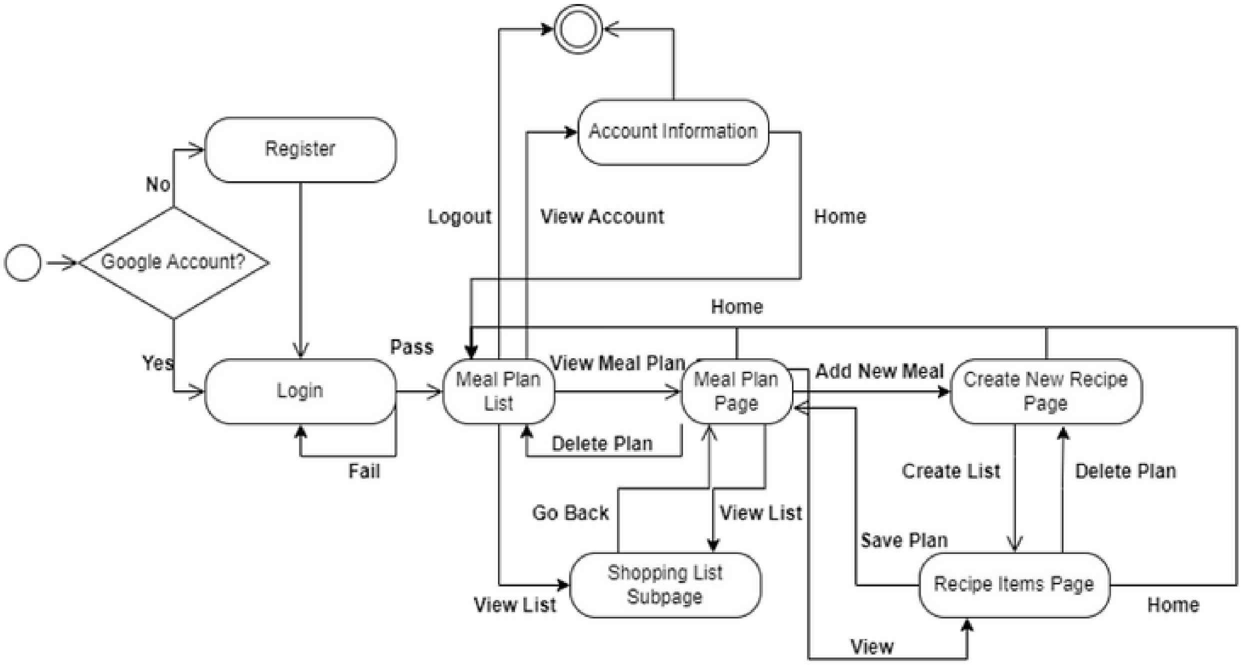
Data transformations:

“User” object data transforms upon receiving new or updated data from the user, adapting to fit any new limits or criteria applied to their diet.

“Meal Plan” object data transforms to reflect this user-specific criteria held in the “User” object. It will restructure its data to fit the user’s desired meal plan.

“Recipe” object data also transforms to meet user criteria. It will adapt and retrieve or modify recipe data to fit requirements.

6.5 Statechart Diagram



Section 7 - User Interface Design

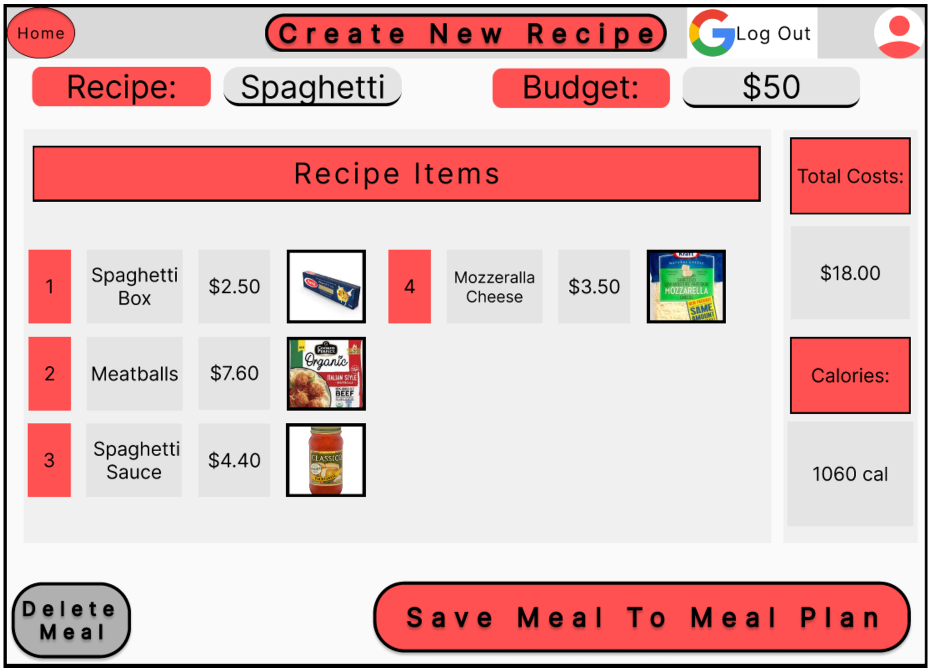
7.1 User Interface Design Overview

Pictures, high level requirements, mockups, etc.

- Link to the Figma Mockup of the website here.
- Link to a demo of editing a meal plan and adding a new recipe here.(Compatible with Desktop)

The software will run through a website for users, and can be accessed in different browsers, and will be responsive and user friendly. With google authentication, user information will be easy to store and access and have the user able to quickly log in and start meal planning. A base simple color palette will be used for the website, focused on warm colors. Users will be able to access the meal page and their user page from most areas of the website and be able to logout anytime back to the Google login page.

Image of Recipe list page:



Recipe: Spaghetti

Budget: $50

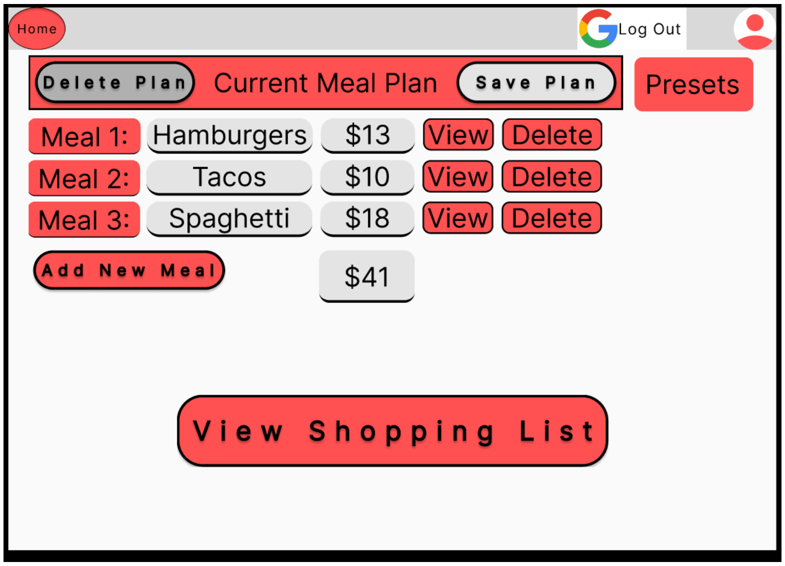
Recipe Items

1	Spaghetti Box	$2.50	4	Mozzeralla Cheese	$3.50
2	Meatballs	$7.60			
3	Spaghetti Sauce	$4.40			

Total Costs: $18.00

Calories: 1060 cal

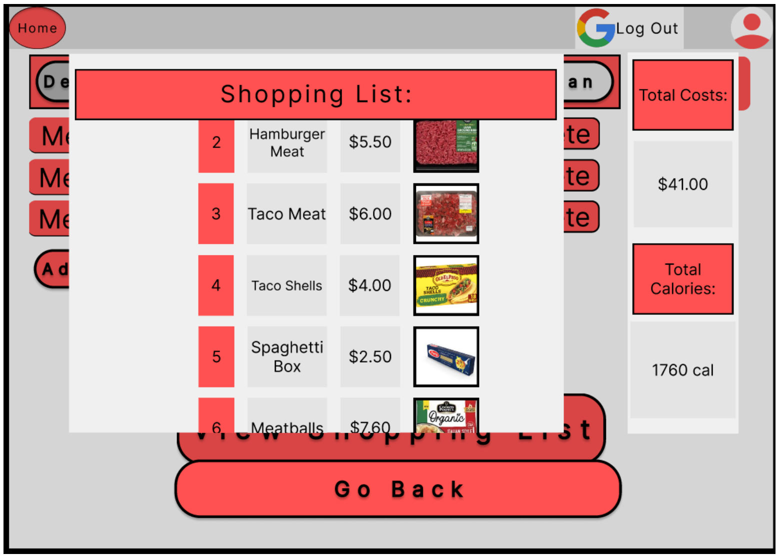
Image of Meal Plan page:



Current Meal Plan

Meal 1:	Hamburgers	$13	View	Delete
Meal 2:	Tacos	$10	View	Delete
Meal 3:	Spaghetti	$18	View	Delete
		$41		

Image of Shopping List page:



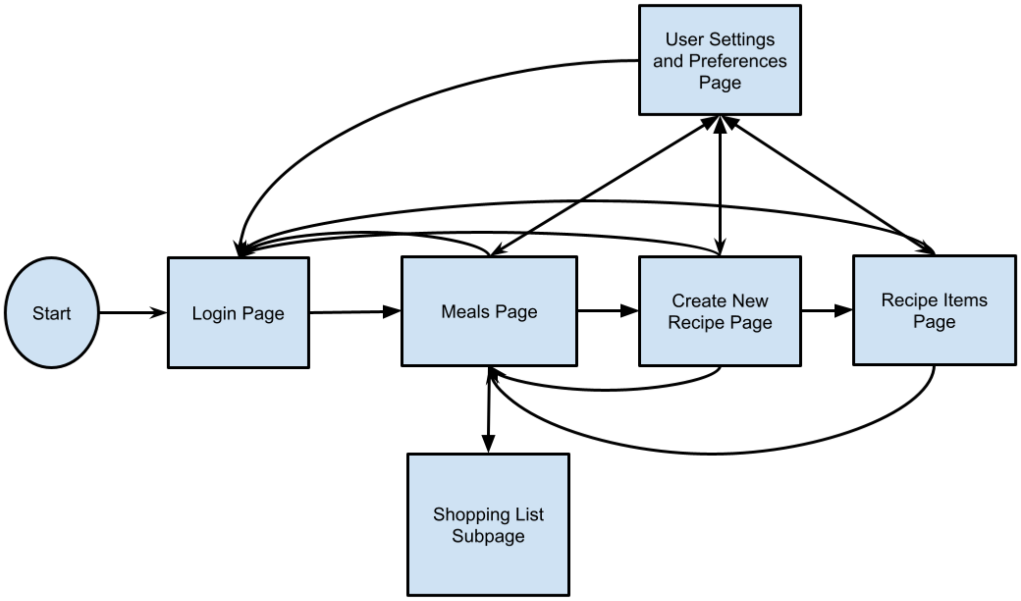
2	Hamburger Meat	$5.50
3	Taco Meat	$6.00
4	Taco Shells	$4.00
5	Spaghetti Box	$2.50
6	Meatballs	$7.60

Total Costs: $41.00

Total Calories: 1760 cal

7.2 User Interface Navigation Flow

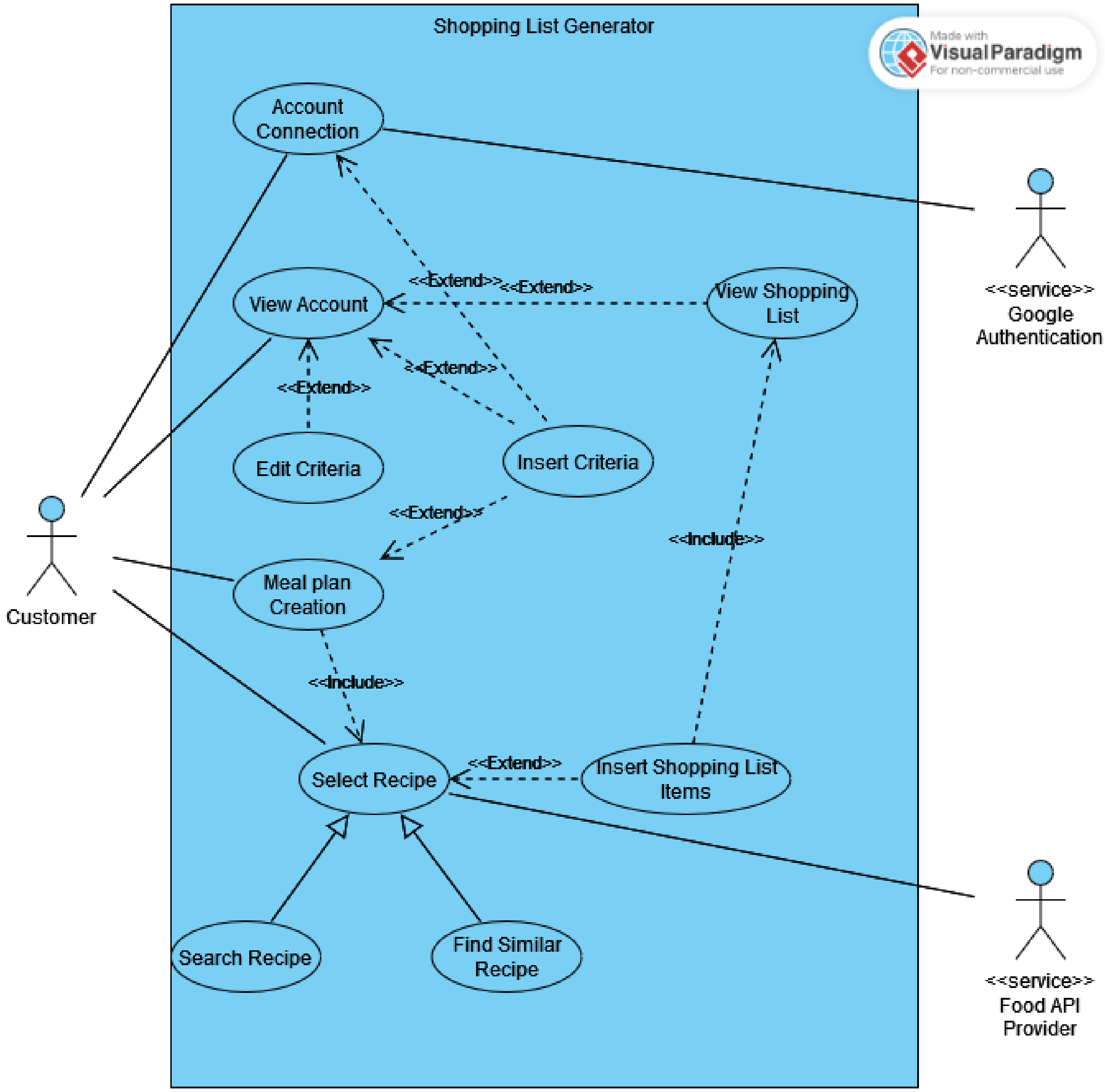
Diagram the flow from one screen to the next



This image illustrates how the user can navigate through the website to different main pages to view pages. From any page the user will be able to access the login page by logging out and the settings page. The main flow will involve adding new items to a saved meal plan and adding/removing items as desired.

7.3 Use Cases / User Function Description

Describe screen usage / function using use cases, or on a per function basis



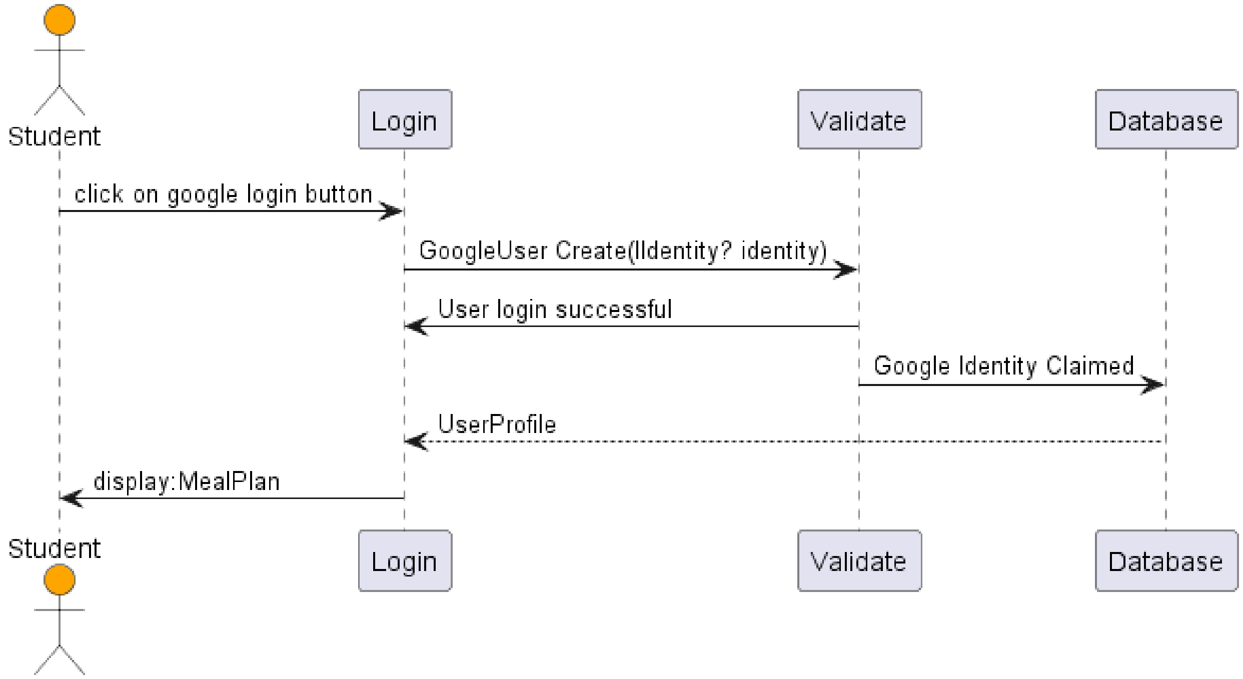
The customer will have access to account connection, meal plan creation, selecting a recipe, and viewing a shopping list. Account connection uses Google authentication as a third party service. The user can view their account at any time and insert or edit criteria, and view made shopping lists. Meal plan creation extends to inserting criteria for that meal plan, and then includes selecting a recipe. When selecting a recipe, the user either searches a recipe or finds similar recipes using items that they have in the shopping list. Recipe selection connects to the services provided by multiple food APIs. After selecting a recipe this extends into inserting the items into a shopping list.

Section 8 - Other Interfaces

Identify any external interfaces used in the execution of this module, include technology and other pertinent data

8.1 Google Authentication

OAUTH identity provider. Implemented using Google Authentication libraries for .net core. Application identification using clientId, shared secret and pre-registered endpoints upon successful authentication.



8.2 Spoonacular API

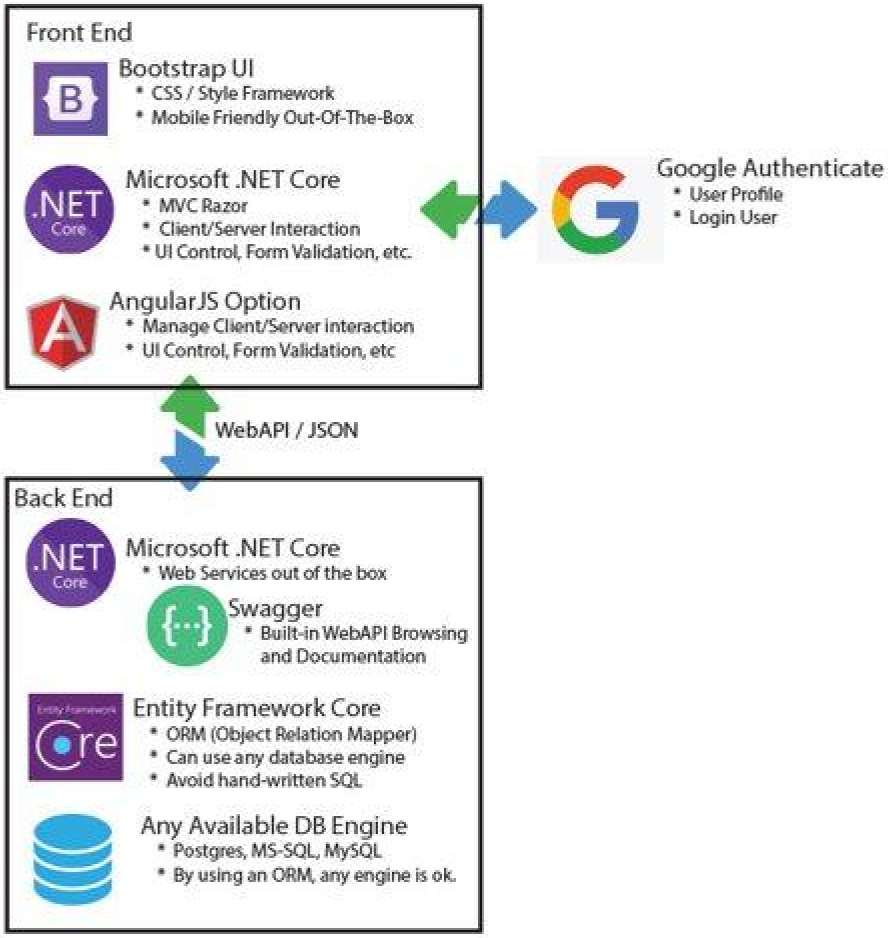
Recipe service. RESTful API, authentication using client Id and shared secret. Primary endpoints to be leveraged are the recipe search (for initial suggestions) and recipes by ingredients (for subsequent recipes).

Section 9 - Extra Design Features / Outstanding Issues

Does not fit anywhere else above, but should be mentioned -- goes here

- The user needs some sort of electronic device with internet access as well as the accessories to operate the device and open the web browser
- User can change preferences at any time
- Shopping lists can be edited after items are automatically inserted by the user

9.1 Tech Stack


Front End
Bootstrap UI
* CSS / Style Framework
* Mobile Friendly Out-Of-The-Box
.NET Core
Microsoft .NET Core
* MVC Razor
* Client/Server Interaction
* UI Control, Form Validation, etc.
AngularJS Option
* Manage Client/Server interaction
* UI Control, Form Validation, etc
Google Authenticate
* User Profile
* Login User
WebAPI / JSON
Back End
.NET Core
Microsoft .NET Core
* Web Services out of the box
Swagger
* Built-in WebAPI Browsing and Documentation
Entity Framework Core
* ORM (Object Relation Mapper)
* Can use any database engine
* Avoid hand-written SQL
Any Available DB Engine
* Postgres, MS-SQL, MySQL
* By using an ORM, any engine is ok.

Section 10 – References

Any documents which would be useful to understand this design document or which were used in drawing up this design.

Section 11 – Glossary

Glossary of terms / acronyms

UI: User Interface

API: Application Programming Interface
UX: User experience
OAUTH: Open authorization