Capstone Documentation

By Huy Nguyen, Thong Ton

Table of Contents

Project Description

Overview/Motivation

We implement an application that embraces multiple Computer Science concepts which include but are not limited to: Cloud Computing, Web Application, Database, Search Engines, APIs, front-end, back-end, and more to create an application that serves as a notifier on discounts or reduced prices on grocery item(s).

For example, if I want to buy some Fuji apples at Walmart for a reduced price, I can use this application to notify me whenever the apples I want are on sale. This application would help the user save extra money by finding the best grocery prices.

We wanted to make a useful application that can be a way to learn as well as solve some of the “real world” problems that one or many might have.

Key Features

Core objectives application that must be completed:

- Available on the internet
- Accessible cross-platform and cross-browsers
- User login/logout (authentication and authorization systems)
- User data and database operations
- User inputs desired grocery item(s)
- System displays different prices of the desired grocery item(s) at local supermarkets
- User picks the cheapest price
- User opt-in for notifications (using email) on desired grocery items that are on sale at local supermarkets.
- Eliminate "cloud lock-in"
- Create customized API(s) for application's related logical processing
- Full-stack development cycle using Agile methodology

Market Value

- Shoppers can now have the ability to pick the best prices for their grocery items. This application would give shoppers cheaper options if there are any when they go grocery shopping. When they check the prices between grocery stores by using the application, they know what is worth paying. The shoppers are able to save additional money
- Additionally, advertising in the application could be a great idea. It benefits both shoppers and businesses. It is a win-win situation if businesses have a sales campaign and appeal to shoppers

Stakeholders

The stakeholders of this project include:

- CSS 497 Advisor and Sponsor
- Frontend Developer
    - Create and implement the app functionalities and the interaction between the app and the user

- Backend Developer
    - Create and manage the app server, server system database, APIs of the app

- Project Manager
    - Manage resources for the project to stay on the right track and deliver the project before the deadline

- Testers
    - Test and verify if the app meets the requirement without any possible error and bug before it is released to the audience

Feasibility

Technical

- Cloud services providers (AWS, Azure) to take care of most web application processing → using the cloud would be ideal for most or all of the components for the project since cloud providers take care of the majority of the “building from scratch aspect”
- Data sharing between the APIs to the application
- Could use the native UI elements on each platform like HTML, CSS to build. The design doesn’t need to be a stylish
- Front-end framework like React or Angular
- Back-end framework like Django or Nodejs or Java Servlet
- Tech stack like MERN or MEAN
- Could use Figma (design tool) for creating application prototype
- Could use Github (collaboration tool) for managing change and keeping track of project progress

Organizational

- Agile SDLC will be used during the development
- Sprint cycles to implement new features
- The maximum size of the development team is two people. We work interchangeably between roles, however, there is a more specific focus in each member of the team:
    - Huy Nguyen as project manager. Also, focus on front-end development
    - Thong Ton as technical support. Also, focus on back-end development
- All members will be in charge of the testing

Operational

- The application will take user input data for retrieving the location and product name
- The application will return the prices of the product
- If the user chooses to receive a notification, the application will send a notification of the product’s price when it is on sale

Economics

- At the current stage, we are planning to make an application for free
- Cost
    - Maintaining cost
    - Cloud services cost
    - Server hosting cost
    - Database cost
- Benefit
    - Kickstarter

    - Developing experience
    - Project management experience
    - Cloud computing experience

Marketability

- Main audience
    - Must know how to use web applications
    - Student
    - Worker
    - Parents
    - Not restricted to age
- Free application
    - Monetize by adding advertisement if feasible
- Easy accessible
    - Since this application is a website application

Challenges

- The effort to find the right APIs
- The effort to maintain web availability
- Possible to maintain fetching data from JSON in particular between APIs and applications. Sometimes APIs could be changed by providers

Support/maintainability

- There is no user support at the current stage. Further notice will be announced.
- Further scaling for the site/application if enough experience and resources are available.

Tentative Schedule

(This schedule is subjected to change and update over the development life cycle)

1/20/2022 Date and planned work:	Initial research for tools, resources, and required services. Product brainstorming.
1/28/2022 Date and planned work:	Outline back-end development. Outline front-end design such as flowchart, and application structure. Targeted frameworks/services

	are AZURE and AWS. Targeted technologies are web APIs, SQL database, authentication framework (either from AZURE or AWS), web application architecture (e.g. MVC, client-server, 1-page web app, etc.)
2/03/2022 Date and planned work:	Back-end development: APIs (AWS and Kroger API) for data retrieval services.
2/5/2022 Date and planned work:	Back-end and front-end development: ASP.NET training and learning for web application development using the MVC pattern for back-end and front-end communication. Front-end will be made from starch using Razor Pages to integrate the front-end and back-end together. **The Back-end language will be C#.** Google Maps API will also be used to implement a feature that shows the user location of the store.
2/8/2022 Date and planned work:	Back-end and front-end development (cont.): Design back-end and front-end architecture using MVC (Model-View-Controller) and ASP.NET Framework version 4.7.x. Connect API services from AWS and Kroger to the web application.
2/12/2022 Date and planned work:	Back-end and front-end development (cont.): Continue on the development. Integrate local database framework from Microsoft (LocalDB and Entity Framework) for local development and testing.
2/16/2022 Date and planned work:	Back-end and front-end development (cont.): Continue on the development. Integrate local identity management framework from Microsoft (Identity Framework) for local development and testing.
2/22/2022 Date and planned work:	Back-end and front-end development (cont.): Continue on development. Core functionalities such as search functions and user identity management systems are implemented and working in conjunction. The local database is used for testing. More testing and debugging are needed, and more features will be implemented.
2/28/2022 Date and planned work:	Back-end and front-end development (cont.): More debugging and testing. Email service using Sendgrid is integrated for additional features such as “Weekly news sellers alerts via emails”.
3/03/2022 Date and planned work:	Back-end and front-end development (cont.): More bug fixes. Improvement on user interface and UX (user experience).

3/10/2022 Date and planned work:	Back-end and front-end development (cont.): More bug fixes and finalizing for deployment.
3/14/2022 Date and planned work:	Configure Azure Web App Services, SQL database server, and related resources for deployment.
3/16/2022 (updated, actual development date) Date and planned work:	Web application is deployed and live at https://sinhvienngheo.azurewebsites.net/
NEXT QUARTER	**POSTER AND PRESENTATION.**
…	…

Formal Software Specifications

Software Development Life Cycle

Sprint 1

Set Up

- Initial brainstorming and project setup.
- Meeting and discussion over Zoom and Discord. Meeting Dates:
- We decided to use 2 cloud service providers, AWS (Amazon Cloud Services) and Azure Cloud Services.

AWS

- Initial services used include but are not limited to AWS S3, AWS Lambda, AWS APT gateway, etc..
- More services will be added later.

Azure

- Initial services used include but are not limited to ASP.NET, Sendgrid emailing services, MVC, Entity Framework, and SQL database. Identity Framework, etc..
- More services will be added later.
- ASP.NET Framework (which uses C#) and we both had little to no experience with previously.

Prototype

- Using Azure Web App Services for setting up a URL. The initial template site can be temporarily accessed by the URL below: https://sinhvienngheo.azurewebsites.net/

  (Site has been updated and completed. If the link is inaccessible, the site has been paused to stop from overspending Azure free and limited computing resources).

Functional Requirements (FR)

These sections focus on the main functionalities of the application. The major requirements are listed below. This also provides the framework of development requirements

ID	Functional requirement

FR-01	Users shall be able to access the site via a unique URL
FR-02	Users shall be able to click main buttons
FR-03	Users shall be able to input location and product name

Non-functional: Requirements (NFR)

ID	Non-Functional Requirement
NFR-01	Web App uses a web framework such as ASP.NET MVC
NFR-02	App uses AWS or Azure authentication middlewares
NFR-03	Use 2 different cloud providers to prevent cloud "lock-in"

Sprint 1 Backlog

- For the next sprint, the functionality listed below will be implemented. (if any)
    - FR-04, FR-05, FR-06
    - NFR-04, NFR-04, NFR-04

Sprint 2

APIs

Gather all the available grocery APIs source to conduct research

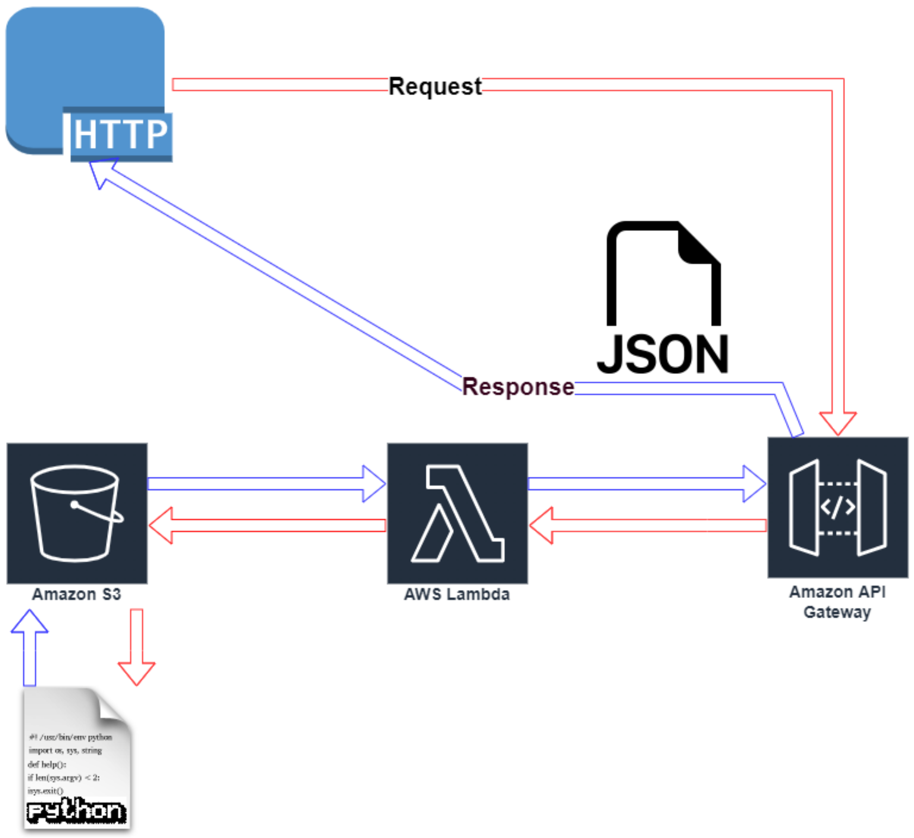
- Understand APIs; how to set up the request, and retrieve response through HTTPS.
- Set up an APIs account for retrieving data
- Re-organize the format of data; make it become compatible with the current project
- These are the groceries that are applied to request data

● FRED MEYER STORES ● FRED ● FOODSCO ● FOOD4LESS ● DILLONS ● QFC ● PICK N SAVE ● SMITHS ● RALPHS ● QUIK STOP	● OWENS ● METRO MARKET ● KROGER ● JAYC ● GERBES ● FRY ● COPPS ● CITYMARKET ● BAKERS

Cloud Provider

Using Amazon Website Services (AWS) to conduct Restful APIs

- Utilize the build-in services in AWS environment to make a fully Restful APIs
- Tech-stack:
    - **AWS S3**: use to store the python source codes
    - **AWS Lambda**: use to make serverless; which deploy the source code
    - **AWS API Gateway**: use to deploy and public Restful API links
- Flow of data



Programming Language

This part of the program is written completely in Python

- Set up a virtual environment which is the requirement for building run-time source code in AWS Lambda
- Install needed package for Python to invoke AWS dependencies
- All the source codes are stored in AWS S3 that being invoked in AWS Lambda

APIs Early Test Phase

Imply available APIs test service which Postman

- Set up credentials
- Test request/response behave unexpectedly

Functional Requirements (FR)

These sections focus on the main functionalities of the application. The major requirements are listed below. This also provides the framework of development requirements

ID	Functional requirement
FR-04	API must be able to return major values such as product name, product id, location, price, discount price
FR-05	User able to see the groceries names
FR-06	User able to see the groceries prices

Non-functional: Requirements (NFR)

ID	Non-Functional Requirement
NFR-04	APIs response time certainly less than 30 seconds
NFR-05	APIs must be Restful framework
NFR-06	APIs request must be through serverless; executable codes are not hosted by renting a server.

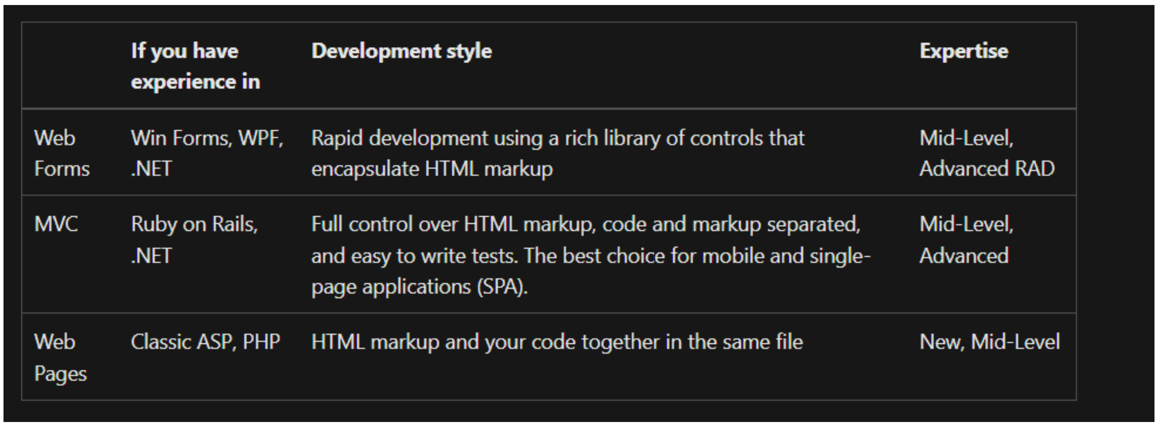
Sprint 2 Backlog

- For the next sprint, the functionality listed below will be implemented.
    1. FR-06, FR-07, FR-08, FR-09, FR-10, FR-11, FR-12, FR-13
    2. NFR-07, NFR-08, NFR-09

Sprint 3

ASP.NET (ASP.NET FRAMEWORK 4.7.2)

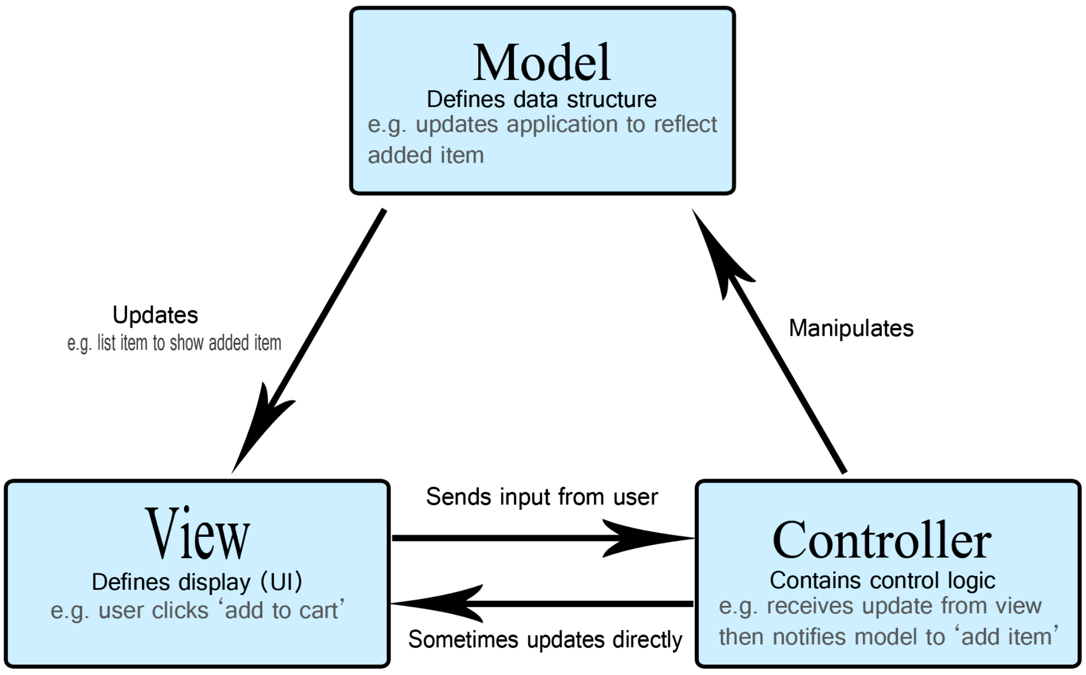
- ASP.NET is a free web framework for building great websites and web applications using HTML, CSS, and JavaScript.
- According to the documentation, ASP.NET MVC is a good choice for building web applications. (It’s also a pretty challenging framework to learn):



	If you have experience in	Development style	Expertise
Web Forms	Win Forms, WPF, .NET	Rapid development using a rich library of controls that encapsulate HTML markup	Mid-Level, Advanced RAD
MVC	Ruby on Rails, .NET	Full control over HTML markup, code and markup separated, and easy to write tests. The best choice for mobile and single-page applications (SPA).	Mid-Level, Advanced
Web Pages	Classic ASP, PHP	HTML markup and your code together in the same file	New, Mid-Level

- This ASP.NET Framework will be the underlying technology used to build the web application that can be accessed via any browser as long as there is internet connection. This framework enables integrating Database, front-end, back-end, and many different techs and services together to create our application.
- The web application is hosted on Azure Cloud.

MVC (Model-View-Controller) 5


Model
Defines data structure
e.g. updates application to reflect added item
Updates
e.g. list item to show added item
Manipulates
View
Defines display (UI)
e.g. user clicks 'add to cart'
Sends input from user
Sometimes updates directly
Controller
Contains control logic
e.g. receives update from view then notifies model to 'add item'

- The Model-View-Controller is the back-bone architecture used to design our project.
- This model enables the web application to process front-end input from user and integrate back-end login along with Database connection and other external services.

Google Maps API

- As an additional feature, this application uses the Google Maps API and it's Geolocation service to translate longitude and latitude to a human readable address and vice versa.
- Initially the default values for finding location on Google Maps is via Longitude and Latitude. But through Geocoding and Reverse Geo-Coding, the Long and Lat are converted to an actual human readable address such as the snippet below:

Secaucus
95
Union City
MANHATTAN
FDR
East River
COLLEGE P
ASTORIA
Grand Central Pkwy
FLUSH
Empire
277 Bedford Ave, Brooklyn, NY 11211, USA
ELMHURST
9
9A
78
278
495
440
New York
FOREST HIL
78
278
BUSHWICK
478
BROOKLYN

Functional Requirements (FR)

ID	Functional requirement
FR-06	User able to see the front page with user input text area for the "Search Grocery" function
FR-07	The front-page must provide text area input and validate the provided input from the user.
FR-08	User must be able to submit the input
FR-09	System must process the user input to call the external grocery API
FR-10	System must receive and clean response from the API call accordingly
FR-11	System must display the useful data (relevant data) back to the user
FR-12	System must handle user input errors (front-end validation) and system errors (back-end validation)
FR-13	System must integrate Google Maps API for the "Navigate To Location" feature

Non-functional Requirements (NFR)

ID	Non-Functional Requirement
NFR-07	Improve user UX and UI by making the site more responsive
NFR-08	System must process the retrieval data in less than 3 seconds for larger data
NFR-09	System should be able to validate user input on front-end and

Sprint 3 Backlog

- For the next sprint, the functionality listed below will be implemented:
    - NFR-07, NFR-08, NFR-09, FR-13.

The above NFRs will be implemented/incorporated into the program within the next print.

User Case

UC-1: User Navigates to Site

Goal: Display front-page and search option for the user to use

Actor(s): Anyone with internet access

Preconditions: The website is the online

Main scenario:

1. The website loads to the front page (Index page)
2. The front page will also display a nav-bar that has options for login, signup, contact/about page, and a home page (Some functionality such as login and signup will be implemented in the next sprint)
3. The user enters “Grocery Item Name” input, for example, milk
4. The user enters “Zipcode” input, for example, 98032
5. The user clicks the enter button (input validation should happen here)
6. The input gets delivered to the back-end for logical processing
7. The input gets processed successfully and returns output
8. The user is taken to a different page to see the search results
9. The user can go back to search or go to pages available on the navigation bar
10. The user can click on the option “Navigate to this store” to take them to the Google Map that shows the store location

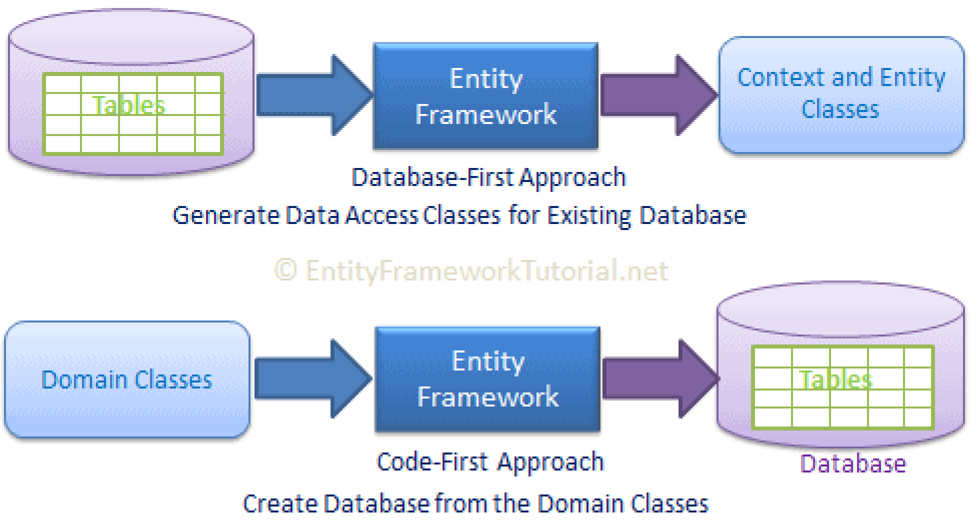
End conditions/goals:

1. The user can see formatted output that resulted from their input
2. The user can navigate (using navigation bar) around the site (initially only Search function, navigate function, and contact/about page will be available).

Sprint 4

ASP.NET Entity Framework (EF)

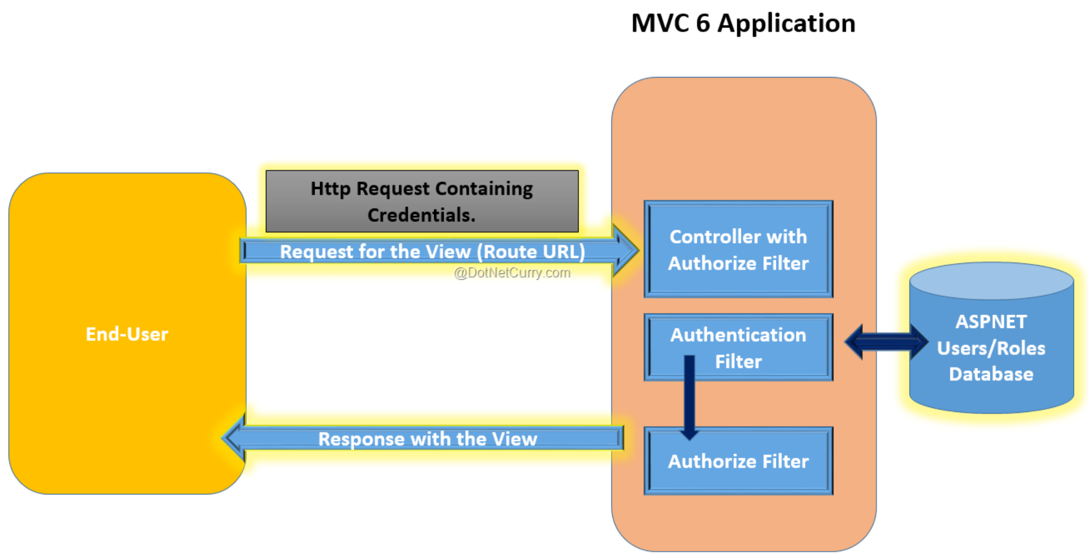
- Entity Framework (EF) is an object-relational mapper that enables. NET developers to work with relational data using domain-specific objects. It eliminates the need for most of the data-access code that developers usually need to write:


Tables
Entity Framework
Context and Entity Classes
Database-First Approach
Generate Data Access Classes for Existing Database
© EntityFrameworkTutorial.net
Domain Classes
Entity Framework
Tables
Code-First Approach
Database
Create Database from the Domain Classes

(More on Code-First Approach used in our project in order to use EF)

- SQL Database (LocalDB for development and SQL Server for deployment/production) is used in conjunction with the EF to provide CRUD operations (Create, Read, Update, Delete) on user-related data and applications data.
- This EF framework helps facilitate the database structure and operations which enables the use of the Identity Framework for user identity management.

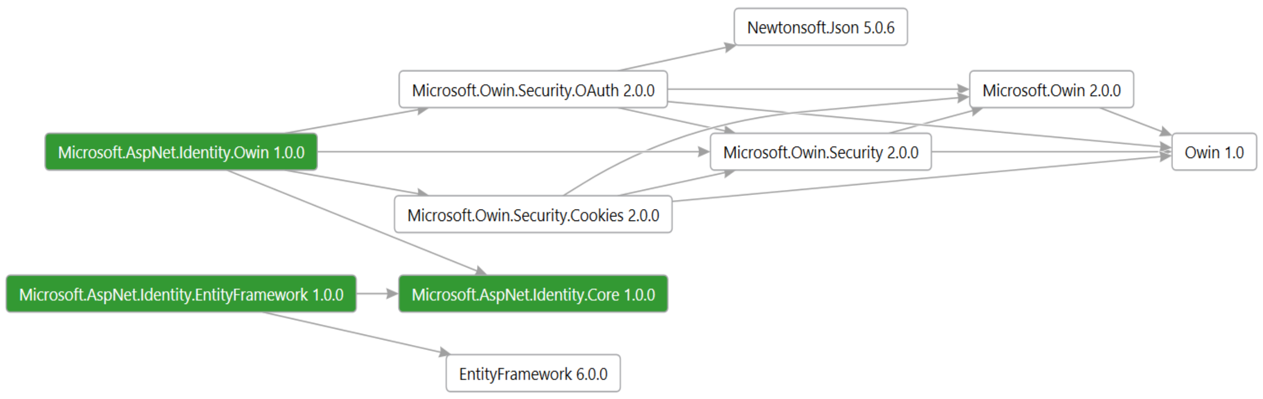
ASP.NET Identity Framework (IF)


MVC 6 Application
End-User
Http Request Containing Credentials.
Request for the View (Route URL)
@DotNetCurry.com
Controller with Authorize Filter
Authentication Filter
Authorize Filter
ASPNET Users/Roles Database
Response with the View

- ASP.NET Identity helps us manage the user accounts. It allows users to self-register on the site. It allows users to select user IDs and passwords. Users can also register using the social login providers such as Facebook, Google, Twitter, etc. (although external logins have not yet been implemented) ASP.NET MVC Identity uses the entity framework Code first and OWIN.

- ASP.NET Identity is implemented using the following components:
  - ***Microsoft.AspNet.Identity.EntityFramework***
    This package has the Entity Framework implementation of ASP.NET Identity which will persist the ASP.NET Identity data and schema to SQL Server.
  - ***Microsoft.AspNet.Identity.Core***
    This package has the core interfaces for ASP.NET Identity. This package can be used to write an implementation for ASP.NET Identity that targets different persistence stores such as Azure Table Storage, NoSQL databases, etc.
  - ***Microsoft.AspNet.Identity.OWIN***
    This package contains functionality that is used to plug in OWIN authentication with ASP.NET Identity in ASP.NET applications. This is used when you add sign-in functionality to your application and call into OWIN Cookie Authentication middleware to generate a cookie.

- The diagram below shows the components of the ASP.NET Identity system. The packages in green make up the ASP.NET Identity system. All the other packages are dependencies that are needed to use the ASP.NET Identity system in ASP.NET applications:


Newtonsoft.Json 5.0.6
Microsoft.Owin.Security.OAuth 2.0.0
Microsoft.Owin 2.0.0
Microsoft.AspNet.Identity.Owin 1.0.0
Microsoft.Owin.Security 2.0.0
Owin 1.0
Microsoft.Owin.Security.Cookies 2.0.0
Microsoft.AspNet.Identity.EntityFramework 1.0.0
Microsoft.AspNet.Identity.Core 1.0.0
EntityFramework 6.0.0

Functional Requirements (FR)

ID	Functional requirement
FR-14	User is able to navigate to the Login and Signup page via nav bar
FR-15	User can use the sign-up page to sign up for an account with password and email as username
FR-16	User must confirm their email address to finish account creation
FR-17	User will receive an email to confirm their address
FR-18	System must send email confirmation to user via their email account
FR-19	System must show the “resend confirmation” option if no email went through
FR-20	The email must include “confirm email link” and redirect user back to login page
FR-21	User must be able to login using their email address and password AFTER successful email confirmation
FR-22	System must show Profile Manager option after the user has logged in successfully
FR-23	Profile Manager must show “back to search” option, “change password” option, “create item list” option for first

	time user
FR-24	Back to search option must take user back to the search page (home page/front page)
FR-25	Password change option should reset user password via email confirmation similarly to the signup process
FR-26	Create item list option must enable first time user to create grocery items (shopping list)
FR-27	System must verify whether a user has created grocery items to display options “view items list” and “enable email alert for items” option
FR-28	View items list option must display available grocery items that user has added
FR-29	View items list option must create delete option for each item if user wants to remove an item
FR-30	System must show “add more items” option on the view items list page
FR-31	Enable email alert for items option must take in a Zipcode to send items that belong to local supermarkets to user via email
FR-32	The system must send emails for items in user grocery list and local and available supermarkets only
FR-33	System/Profile Manager must show disable email grocery alert after a user has enabled the service

Non-functional Requirements (NFR)

ID	Non-Functional Requirement
NFR-10	Refactor code to increase performance and readability
NFR-11	System should be more robust by reducing crashes by 99% and not expose the back-end code
NFR-12	System should display status messages from back-end operations to increase

	usability. The status messages must be marked/highlighted “yellow” and conspicuous
NFR-13	System must store keys and secrets in a secure way (e.g. in secure files or places and not hard-coded into source code)

Note: NFR-07, NFR-08, NFR-09, FR-13 from Sprint 3 have been completed at this stage.

Sprint 4 Backlog

- For the next sprint, the functionality listed below will be implemented:
    - FR-16, FR-17, FR-18, FR-19, FR-20, FR-21 (development will ignore the email confirmation process) these FRs will be implemented in the next sprint after an email service has been implemented.
    - FR-25, FR-27, FR-31, FR-32, FR-33 pertain to features that use an email service, thus, these FRs will be implemented in the next sprint after an email service has been implemented.
    - NFR-12, NFR-13 will be implemented in the next sprint.

The above FRs and NFRs will be implemented/incorporated into the program within the next print.

User Case

UC-2: User Gains Access to Profile Manager

Goal: User will be able to sign up, login and access Profile Manager which they can use additional services

Actor(s) : Anyone with internet access and a valid email address

Preconditions: The website is online, user has a valid email address

Main scenario:

1. The user clicks on sign-up page and enter their valid email address and password to sign-up
2. The user must confirm their email address (for deployed site version), email validation is bypassed for development or testing purpose
3. After the user has successfully logged in, they are taken to the front page and the Profile Manager option is now available in the navigation bar

4. The user goes to the Profile Manager, for first time users, they only see options for “go back to search/front page”, “change password” and “create items list” to create grocery items
5. For users that have created a grocery list, they will see more options in the Profile Manager. They will see “create more items”, “enable email alert” (“disable email alert” if they have enabled and now want to disable)
6. The user can add more items into their grocery list through “create more items” option
7. The user can “enable email alert” to receive news sellers on discounted items (only for items in the grocery list) delivered to their confirmed email address weekly (every sunday)
8. The user can change password via “change password” option, which requires a confirmation email
9. The user can go back to search, or go to other pages available on the navigation bar
10. The user can sign out of their account, they will lose access to the Profile Manager

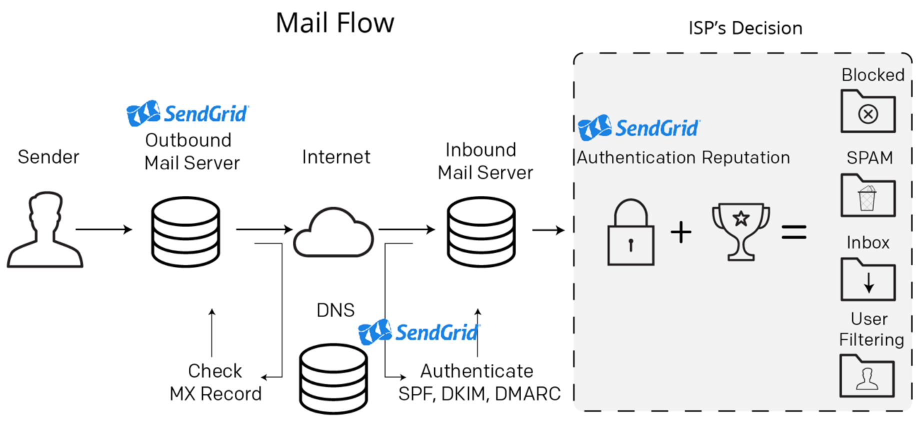
End conditions/goals:

1. The user can log in and out, access Profile Manager page to use additional features and services

Sprint 5

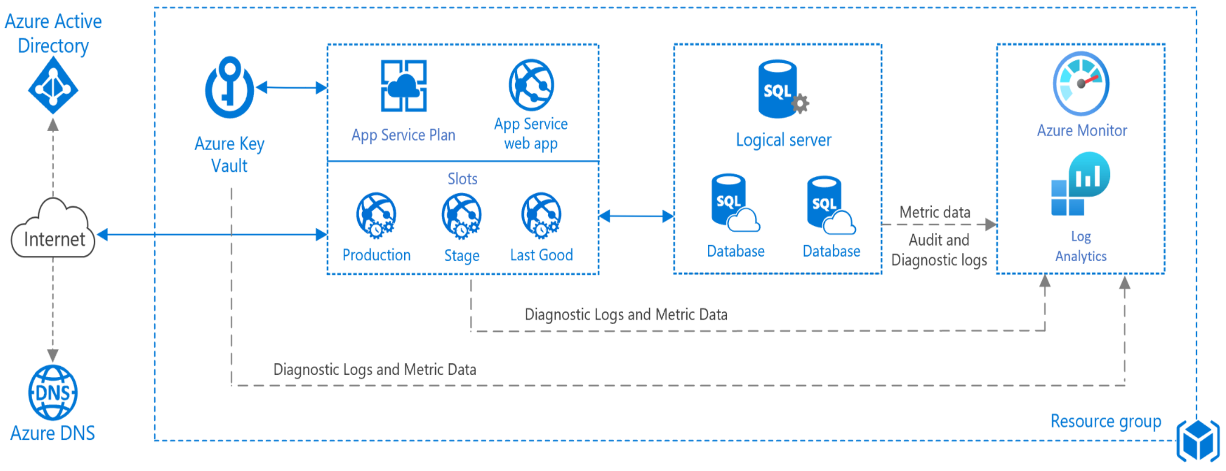
Sendgrid Email Service/API

- SendGrid provides a cloud-based service that assists businesses with email delivery.The service manages various types of email including shipping notifications, friend requests, sign-up confirmations, and email newsletters.
- We use SendGrid primarily for the News Seller feature and email confirmation activities in this project
- SendGrid API provides convenient to connect and integrate into our program without having to set up many middlewares:



Deployment to Azure Cloud

- **We use Azure Cloud to host and deploy our application via Visual Studio.** Visual Studio handles all of the dependencies and packages to send to Azure Web App Services.
- The services we are currently using on Azure Cloud are:
    - Azure App Service
    - Azure SQL Server
    - Azure SQL Database
- Azure App Service is used for hosting the application
- Azure SQL Server is used for facilitate storing user data (credentials) and app data
- Azure SQL Database is used for the actual storage of the user data and app data
- The diagram below shows roughly the architecture of the deployment process:



Note: Most if not all of the previous and new requirements if any have been implemented at this stage.

The site is live at ***https://sinhvienngheo.azurewebsites.net/***

Website Link

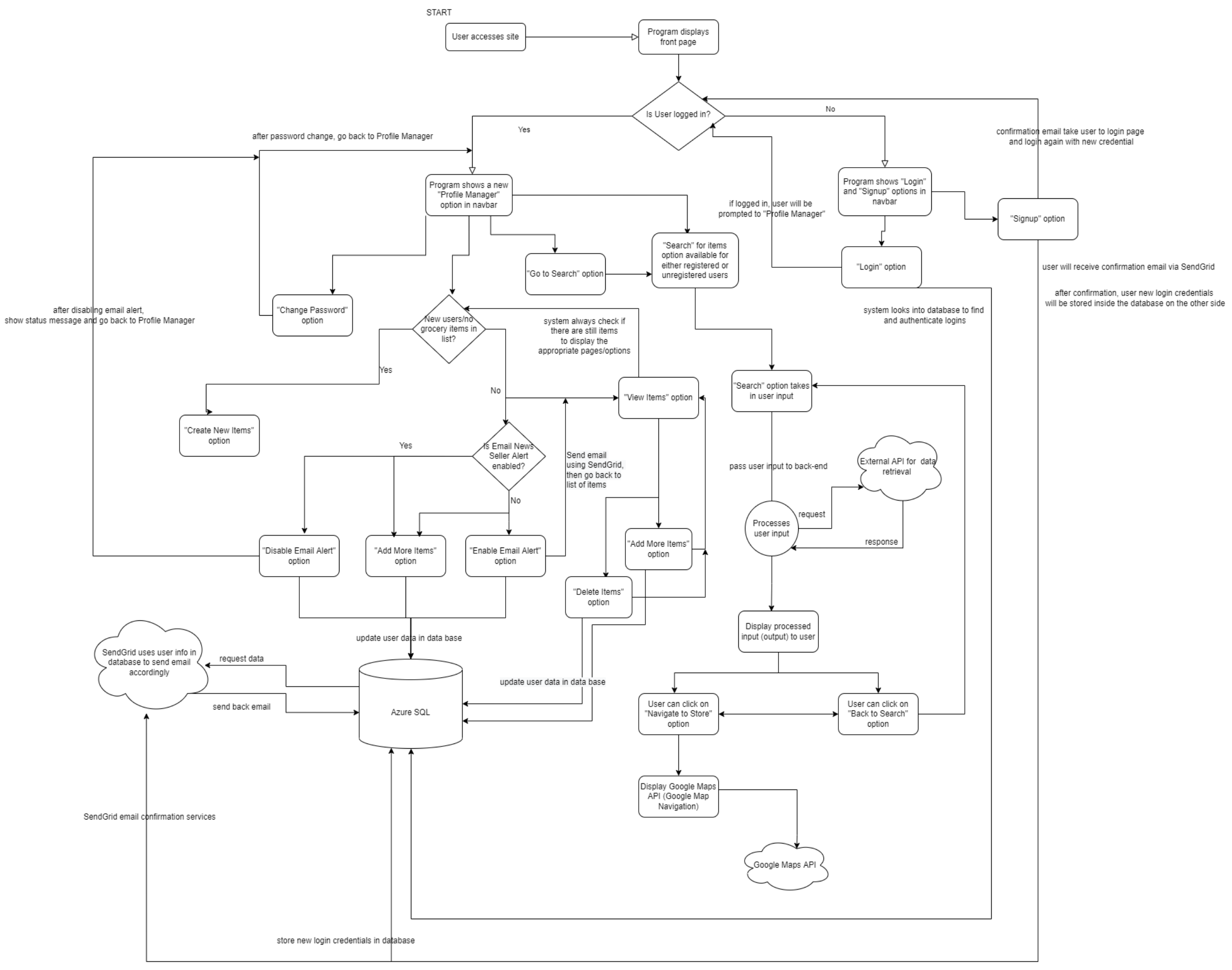
The site is now active and live at:

https://sinhvienngheo.azurewebsites.net

Class Diagram

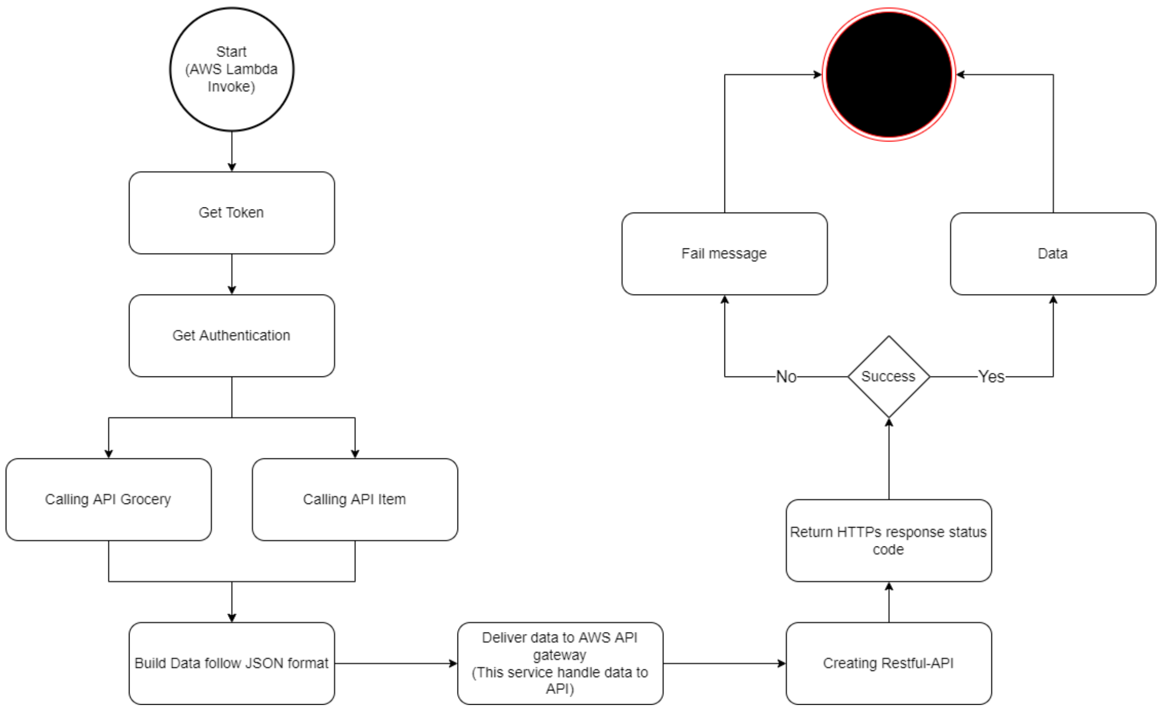
General Program Flow Diagram

LINK TO PROGRAM FLOW DIAGRAM (might take a few seconds to load, but much better clarity than the inserted png below)



START
User accesses site
Program displays front page
Is User logged in?
Yes
No
after password change, go back to Profile Manager
Program shows a new "Profile Manager" option in navbar
Program shows "Login" and "Signup" options in navbar
confirmation email take user to login page and login again with new credential
if logged in, user will be prompted to "Profile Manager"
"Signup" option
"Search" for items option available for either registered or unregistered users
"Go to Search" option
"Login" option
user will receive confirmation email via SendGrid
after confirmation, user new login credentials will be stored inside the database on the other side
after disabling email alert, show status message and go back to Profile Manager
"Change Password" option
New users/no grocery items in list?
system always check if there are still items to display the appropriate pages/options
system looks into database to find and authenticate logins
Yes
No
"View Items" option
"Search" option takes in user input
"Create New Items" option
Yes
Is Email News Seller Alert enabled?
Send email using SendGrid, then go back to list of items
pass user input to back-end
External API for data retrieval
No
request
Processes user input
response
"Disable Email Alert" option
"Add More Items" option
"Enable Email Alert" option
"Add More Items" option
"Delete Items" option
update user data in data base
Display processed input (output) to user
SendGrid uses user info in database to send email accordingly
request data
send back email
Azure SQL
update user data in data base
User can click on "Navigate to Store" option
User can click on "Back to Search" option
Display Google Maps API (Google Map Navigation)
SendGrid email confirmation services
Google Maps API
store new login credentials in database

APIs Design Diagram


Start (AWS Lambda Invoke)
Get Token
Get Authentication
Calling API Grocery
Calling API Item
Build Data follow JSON format
Deliver data to AWS API gateway (This service handle data to API)
Creating Restful-API
Return HTTPs response status code
Success
No
Yes
Fail message
Data

JSON Formatted Testing:

https://jsonformatter.curiousconcept.com/

From Kroger

Note: This is apply for only Kroger

```
# Item format return to front-end
#   {
#       'locationId' : 'locationId'
#       'address' : {'addressLine1',
#                   'city',
#                   'state',
#                   'zipcode',
#                   'county'
#               }
#       'chain' :  'Name'
#       'items' : [
#                   {
#                       'productId' : 'productId',
#                       'categories' : 'categories',
#                       'description' : 'description',
#                       'itemId' : 'itemID',
#                       'price' : {
#                           'regular' : 'regular'
#                    'promo' : 'promo'
#                  }
#              }
#              {
#              ...Second Item
#              }
#          ]
#  }
```

References:

- List of the famous grocery stores in us (possible provide APIs)
    - The Kroger Co. (QFC, Fred Meyer, etc… are Subsidiary)
        - Chain name:
            - FRED MEYER STORES
            - FRED
            - FOODSCO
            - FOOD4LESS
            - DILLONS
            - QFC
            - PICK N SAVE
            - SMITHS
            - RALPHS
            - QUIK STOP
            - OWENS
            - METRO MARKET
            - KROGER
            - JAYC
            - GERBES
            - FRY
            - COPPS
            - CITYMARKET
            - BAKERS

APIs Sources

Look for Groceries APIs: https://www.programmableweb.com/category/grocery/api

- Walmart (not feasible to implement)
    - https://walmart.io/docs/affiliate/product-lookup
    - https://rapidapi.com/apidojo/api/walmart/ (Walmart nearby)
- Kroger (Collection of many groceries API)
    - https://developer.kroger.com/reference
- Safeway
- Albertsons Cos. Inc
- Ahold Delhaize Usa
- Publix Super Markets Inc.
- H.E. Butt Grocery Co.
- Meijer Inc.
- Wakefern Food Corp.
- Aldi Inc.

- Whole Foods Market
- Trader Joe’s
- Amazon Fresh
- Albertsons
- Sprouts Farmers Market
- Winco Foods
- Grocery Outlet

Sources:

- Introduction to ASP.NET Identity - ASP.NET 4.x | Microsoft Docs
- Create a secure ASP.NET MVC 5 web app with log in, email confirmation and password reset (C#) | Microsoft Docs
- Deploying passwords and other sensitive data to ASP.NET and Azure App Service - ASP.NET 4.x | Microsoft Docs
- Configure apps - Azure App Service | Microsoft Docs
- Getting Started with ASP.NET MVC 5 | Microsoft Docs
- Tutorial: Use EF Migrations in an ASP.NET MVC app and deploy to Azure | Microsoft Docs
- How to Setup and Configure ASP.NET Core Identity
- Bootstrap List - JSFiddle - Code Playground
- https://jsonformatter.curiousconcept.com/#
- https://aws.amazon.com/s3/getting-started/
- https://aws.amazon.com/lambda/
- https://aws.amazon.com/api-gateway/
- https://docs.python.org/3/library/venv.html
- https://www.tutorialspoint.com/python/python_environment.htm
- https://docs.python-guide.org/dev/virtualenvs/
- https://developer.kroger.com/reference
-

Contact information

Huy Nguyen: huyn8@uw.edu

Thong Ton: tonthong@uw.edu