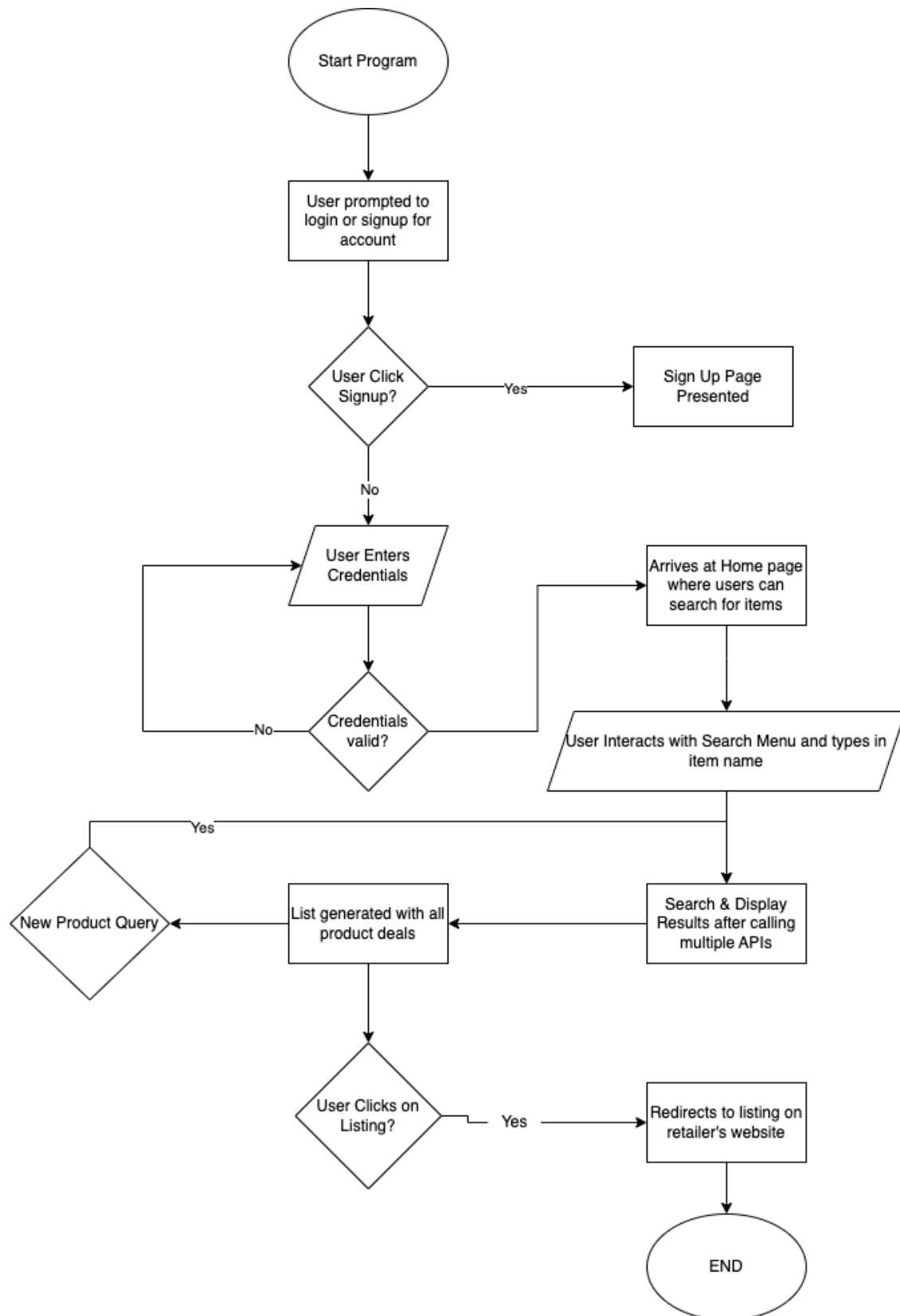# Criterion B: Design

Flowchart(s)
The following flowchart outlines the workflow of application when a user interacts with it.

Algorithms

I will be using a user-based application. Meaning, if users would like to set a price alert, they will be required to sign up for an account. My intent with this is so that I can send alert notifications to valid email addresses. To help me in this, I will be using Firestore's password management system. Also, I will need to read and write data, mainly to help me retrieve information from multiple retail sites about products. I will also be making use of recursion to sort the product prices from low to high.
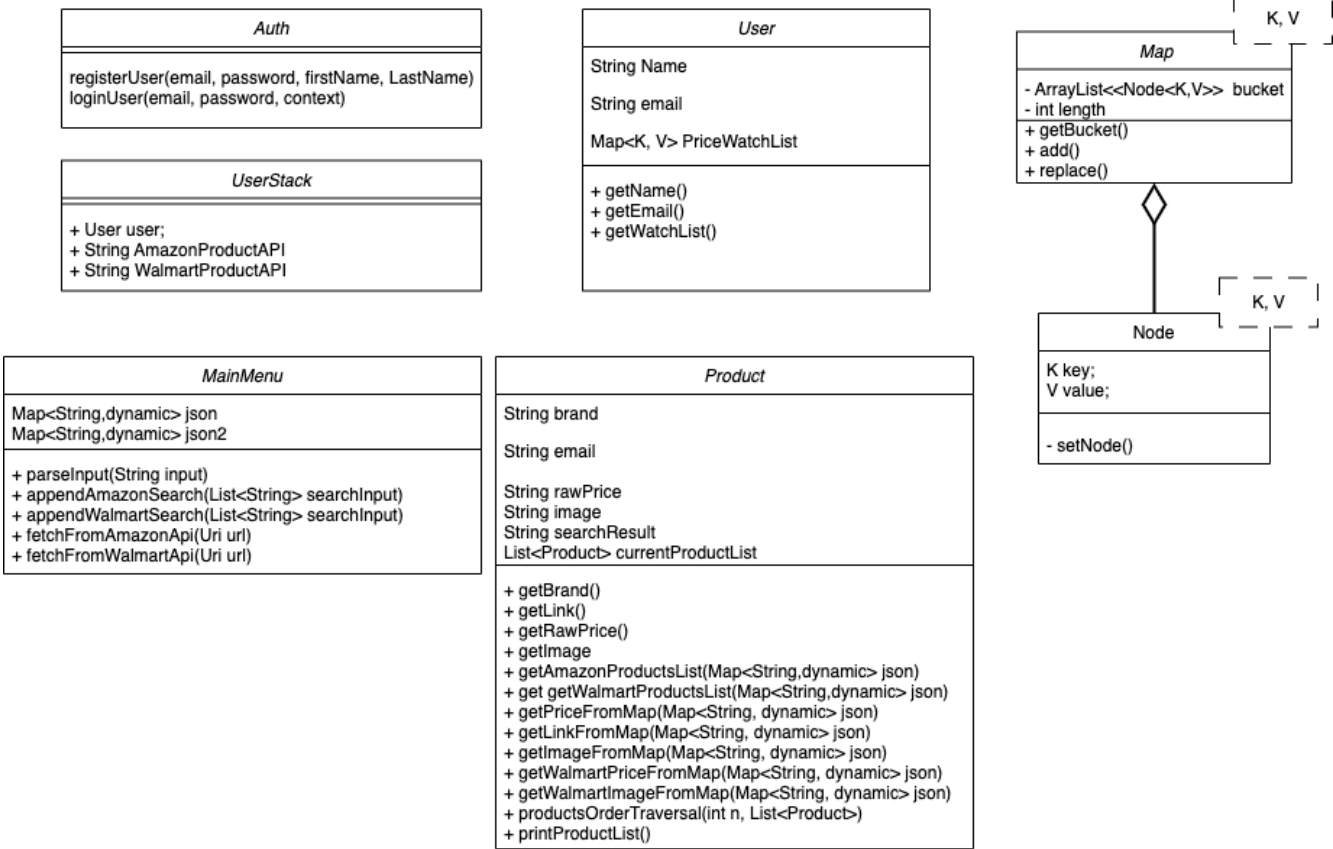
Data Structures

Since I am making use of price alerts, and having to store a lot of user information that will be stored in the cloud database Firestore, I will be employing the use of a Map. Using Firestore as my backend service will allow me to create new users with ease and write to the database without any issues. I will also be using a static list that holds Products.

Objects / UML diagram

I will be making use of a User class that stores basic information like email, name etc. This object-oriented program mainly consists of using the Product object, which will store the product listing information provided by the numerous APIs I plan to use.
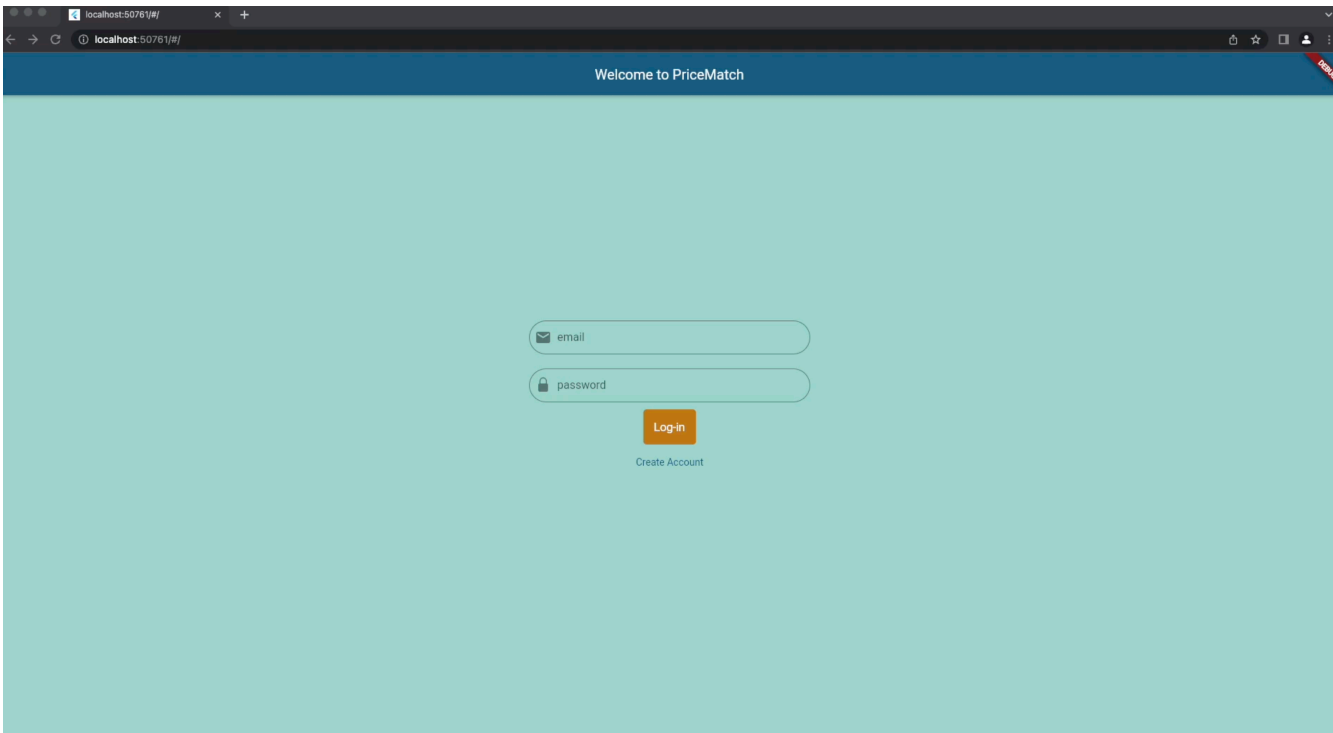
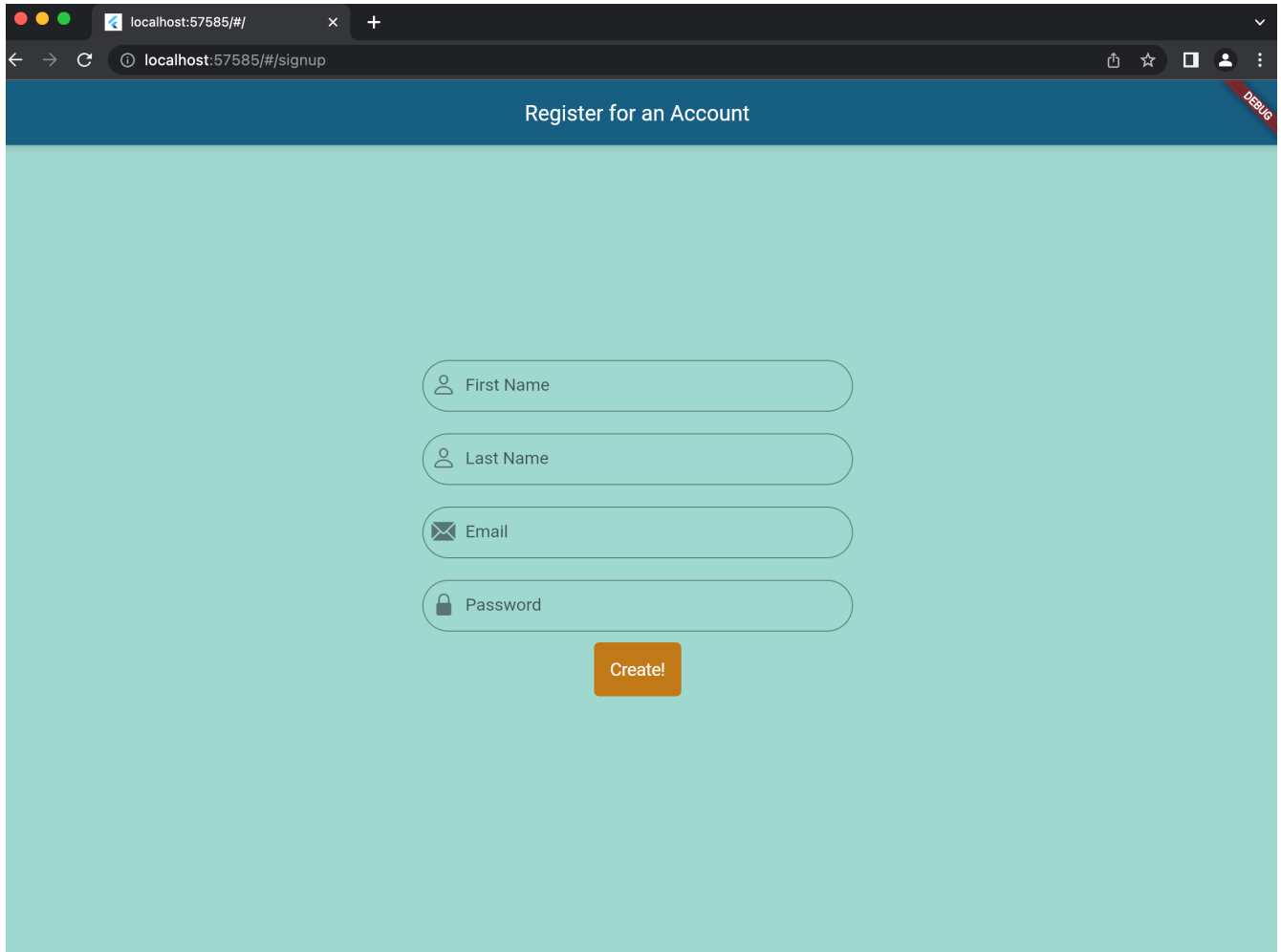| Class Name	Description
User	The standard user class will contain vital user information and will also hold the priceWatchList that users use
Product	Every product object stores data for an individual product listing retrieved by the API. The fields of the Product object are then used to display necessary information to Users regarding a product listing.
MainMenu	Data from the API is fetched and displayed here.
Map	Comprised up of a list of nodes that are used to store the key-value pair which relates the desired price point of an item to the product link the user wants to keep an eye on.

UML Class Diagram

Auth

registerUser(email, password, firstName, LastName)
loginUser(email, password, context)

UserStack

+ User user;
+ String AmazonProductAPI
+ String WalmartProductAPI

User

String Name

String email

Map<K, V> PriceWatchList

+ getName()
+ getEmail()
+ getWatchList()

Map

K, V

- ArrayList<<Node<K,V>> bucket
- int length
+ getBucket()
+ add()
+ replace()

Node

K, V

K key;
V value;

- setNode()

MainMenu

Map<String,dynamic> json
Map<String,dynamic> json2

+ parseInput(String input)
+ appendAmazonSearch(List<String> searchInput)
+ appendWalmartSearch(List<String> searchInput)
+ fetchFromAmazonApi(Uri url)
+ fetchFromWalmartApi(Uri url)

Product

String brand

String email

String rawPrice
String image
String searchResult
List<Product> currentProductList

+ getBrand()
+ getLink()
+ getRawPrice()
+ getImage()
+ getAmazonProductsList(Map<String,dynamic> json)
+ get getWalmartProductsList(Map<String,dynamic> json)
+ getPriceFromMap(Map<String, dynamic> json)
+ getLinkFromMap(Map<String, dynamic> json)
+ getImageFromMap(Map<String, dynamic> json)
+ getWalmartPriceFromMap(Map<String, dynamic> json)
+ getWalmartImageFromMap(Map<String, dynamic> json)
+ productsOrderTraversal(int n, List<Product>)
+ printProductList()

UI flows

1) Login

2) Signup

Register for an Account
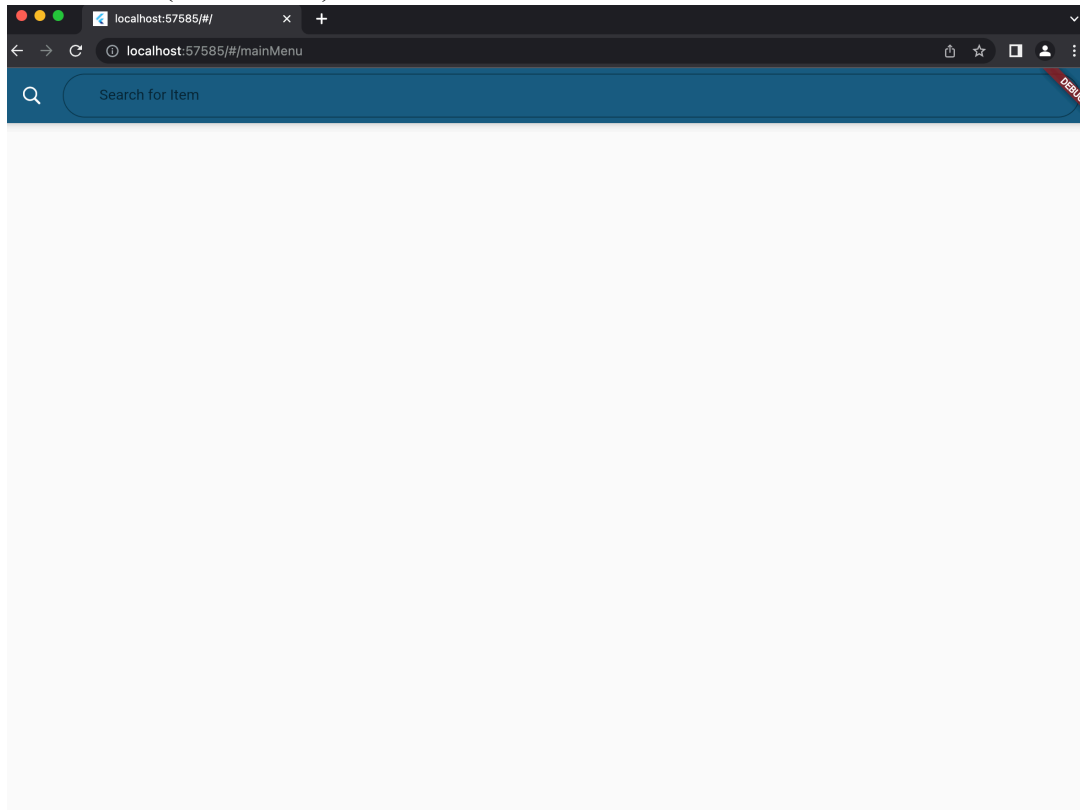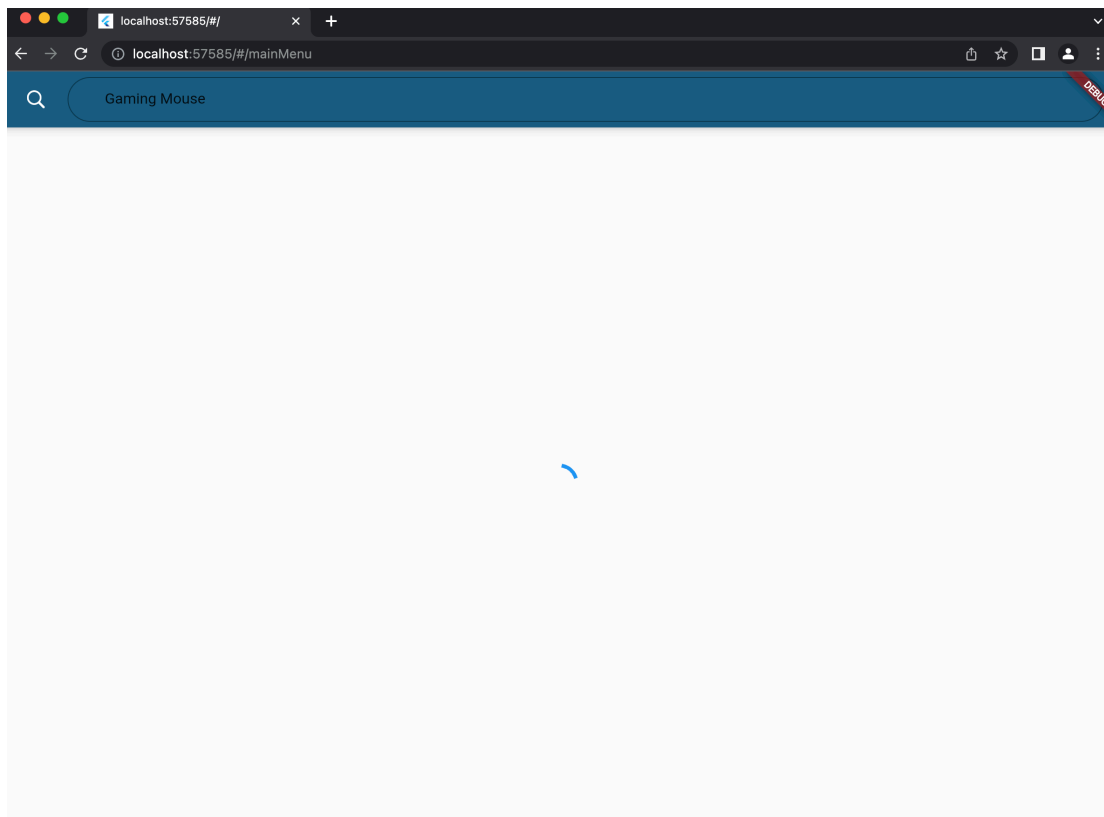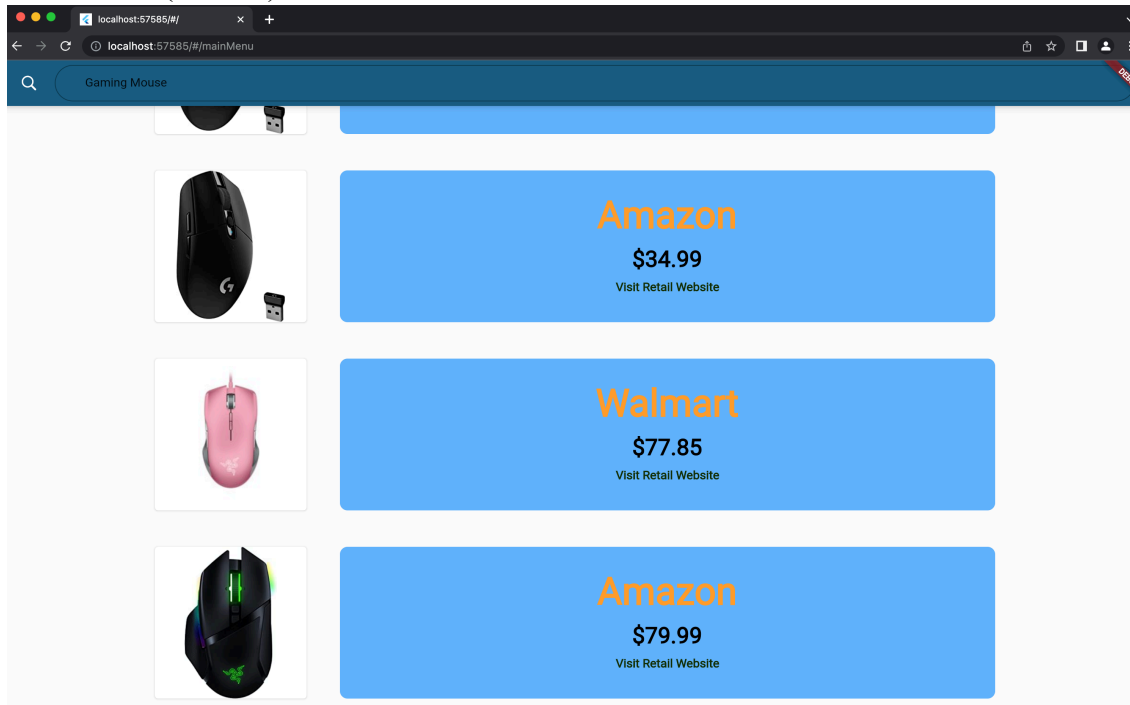
First Name

Last Name

Email

Password

Create!

3) Main Menu (Initial State)

4) Main Menu (Loading)

5) Main Menu (Results)

Test Plan

Success Criteria	Test Plan	Expected Outcome
1, 8	a) Upon start-up, test the security of the login page by entering random characters. b) Users signs up for an account when "create account" is pressed.	a) User stays on the login page until valid credentials are input. b) Users are redirected to sign up page and then back to the login after successful creation of an account.
2, 3, 6, 7	a) Search up common household products. b) Search products with typos in their name. c) Type part of a product name in search menu.	a) Listings from all online retailers are presented in a list view. The listings are presented in a hierarchical fashion where product listings at the top are the "best deals." All listings have a price next to it. b) If no product is found on retail sites, return feedback to user saying to type in a valid product name. c) Returns accurate listings for desired product.
4, 5	a) Search up common household products and click "set price alert"	a) Allows the user to select a price point for a product. When said price point is satisfied, an email notification is sent to the user's

	b) Search up another common household products and click "set price alert". Setting it to something very simple that can be met quite easily.	inbox. b) User should receive emails for all the products they are currently looking out for.
6	a) Search up another product that is offered at many different retailers.	a) When list view is presented, upon user clicking on a specific listing, they should be redirected to its respective web page where they can purchase it.

Word Count: 231 Words