Veggie Check Technical Specification

CA326 Third Year Project

Project Title:	Veggie Check
Student 1:	Róisín O’Rourke - 19360491
Student 2:	Amelia Grigoriev - 19348241
Supervisor:	Paul Clarke
Date Completed:	03/03/2022

Table of Content

1. Introduction

1.1 Overview

There are approximately 300,000 vegans or vegetarians in Ireland, with Ireland ranking third in the world for countries with the most vegans per capita. The product developed is an iOS application called 'VeggieCheck' that will allow users, while shopping, or at any other time, to check if a food product is vegan friendly or to check if certain ingredients are vegan friendly.

The application uses the smartphone camera and text recognition to capture the ingredients of the product. If there is an internet connection, an external API is used to check if the ingredients are suitable. If there is no internet, a database of food ingredients is used to check if the item is suitable for consumption. The system will then notify the user of the status of the food check on the screen.

1.2 Motivation

One of the members of the group is vegetarian which inspired this project. When researching other similar applications available on the app store, the most popular application's reviews showed that it recognised very few products and relied on users to input whether the product was vegan or vegetarian once they had scanned the barcode. We believed that we could create a more efficient application that would instead scan the list of ingredients on a given product and use an API and food database to check if the product is suitable, instead of relying on user input, which would frustrate the user.

2. Changes from Original Design

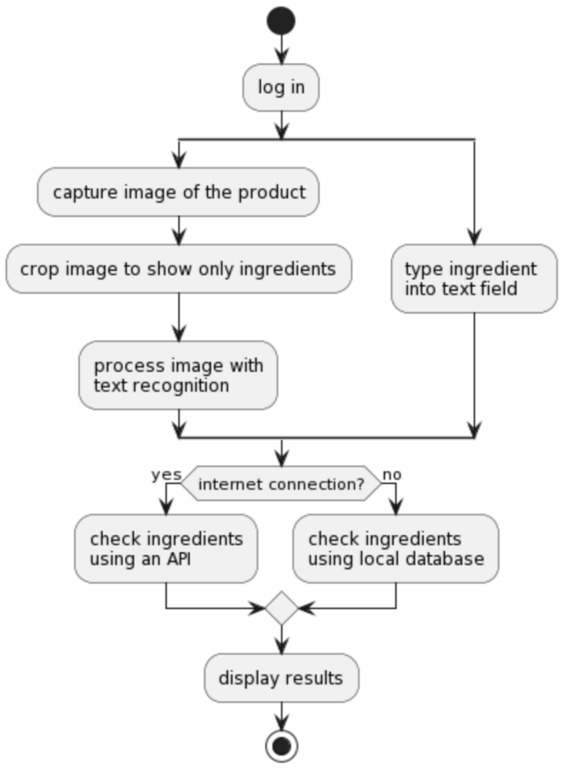
Several changes were made from the original design to the current design of the application. We decided to focus on querying for vegan friendly only ingredients instead of vegan and vegetarian as originally planned, as vegan friendly encompasses vegetarian friendly. If something is vegan friendly it is also vegetarian friendly, but the same is not true, vice versa. We also thought that it is more likely that it will state on a food product if it is vegetarian friendly than vegan friendly items.

Additional functionality was added with the page to view common non vegan ingredients. This works by accessing the local database of non vegan ingredients and displaying them on a list that the user can scroll through. This can allow the user to familiarise themselves with non vegan ingredients so that they can identify them easily on food packaging.

In the original plan for the application, a local database would be used to verify if ingredients were vegan friendly or not. However, in the final design, we decided to give priority to an external API for checking ingredients as the external API is more robust and there is more data available. The local database still exists, however it is only utilised if there is no internet connection available, otherwise the API is used.

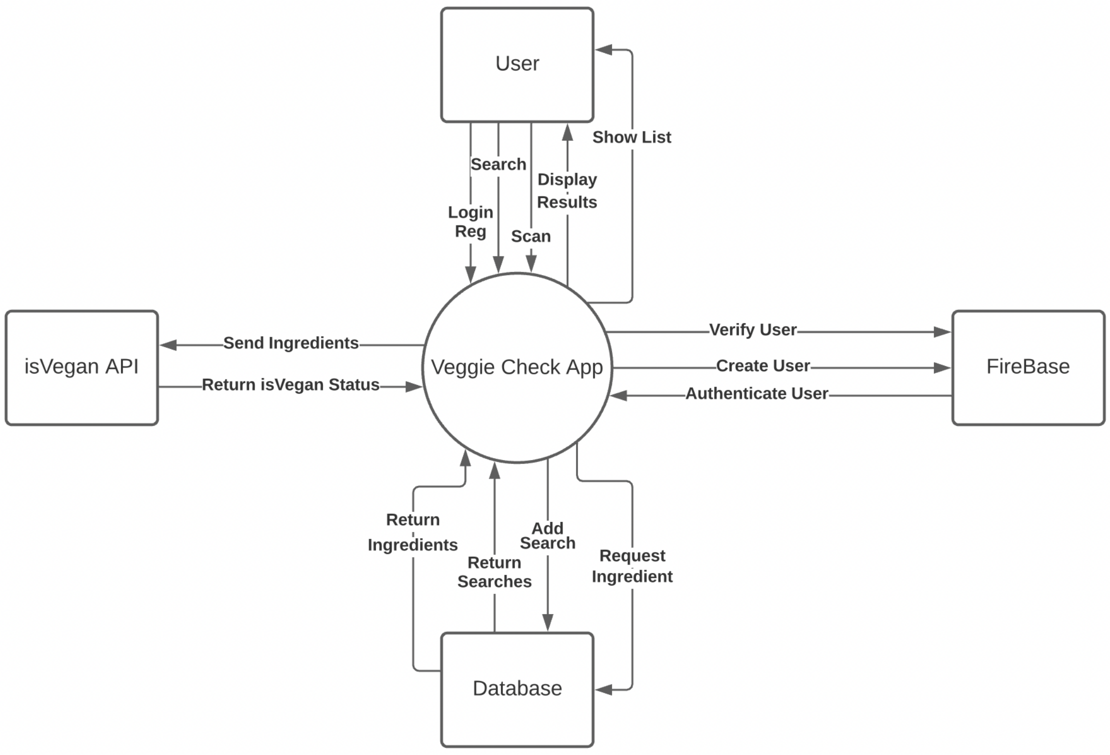
3. High Level Design

3.1 Activity Diagram of Application



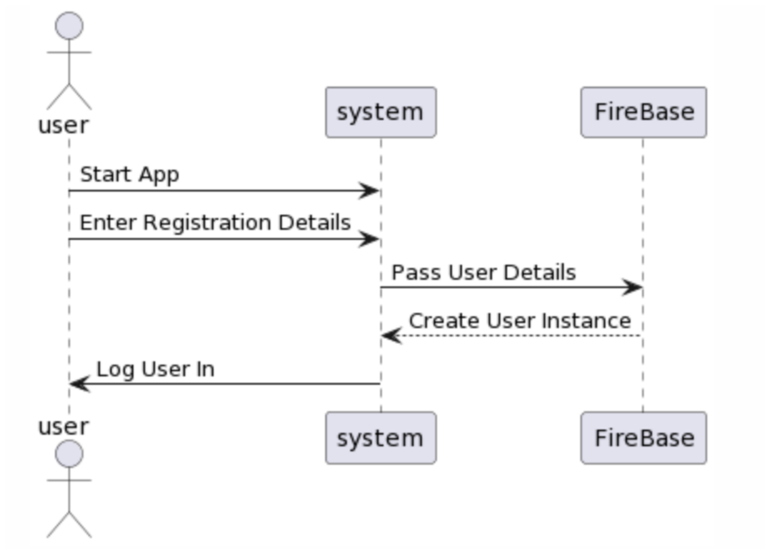
log in
capture image of the product
crop image to show only ingredients
type ingredient into text field
process image with text recognition
yes
internet connection?
no
check ingredients using an API
check ingredients using local database
display results

3.2 Context Diagram

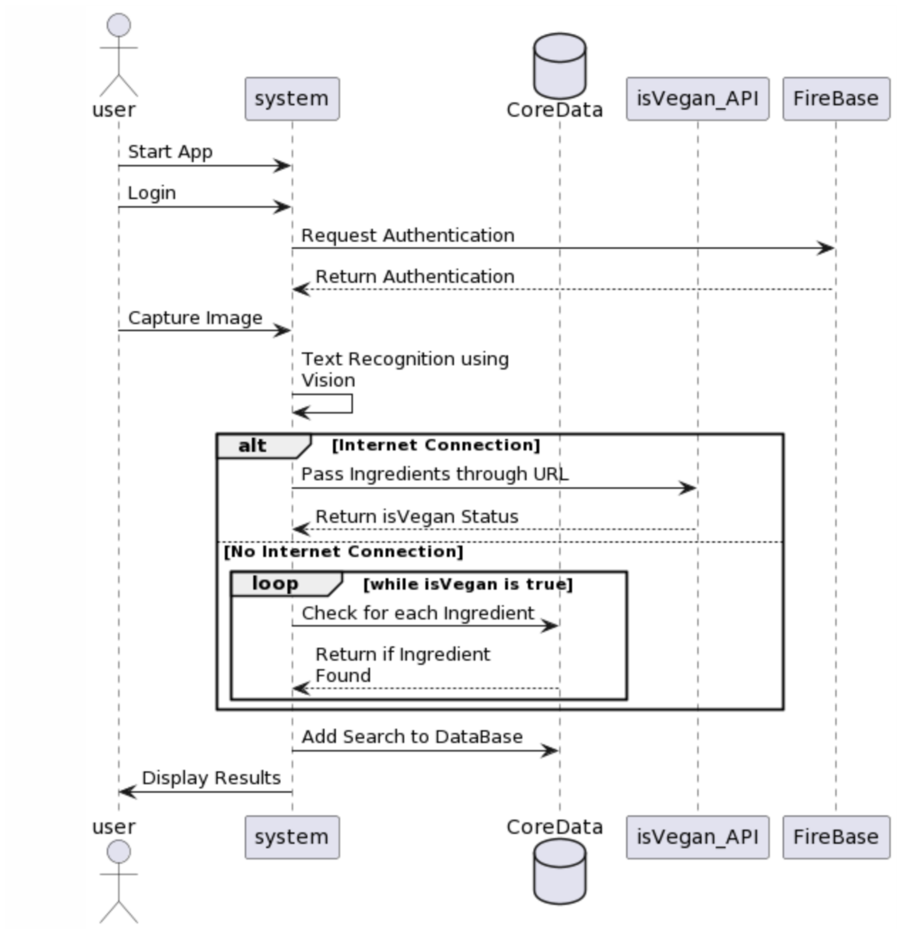


3.3 Sequence Diagrams

3.3.1 User Registration

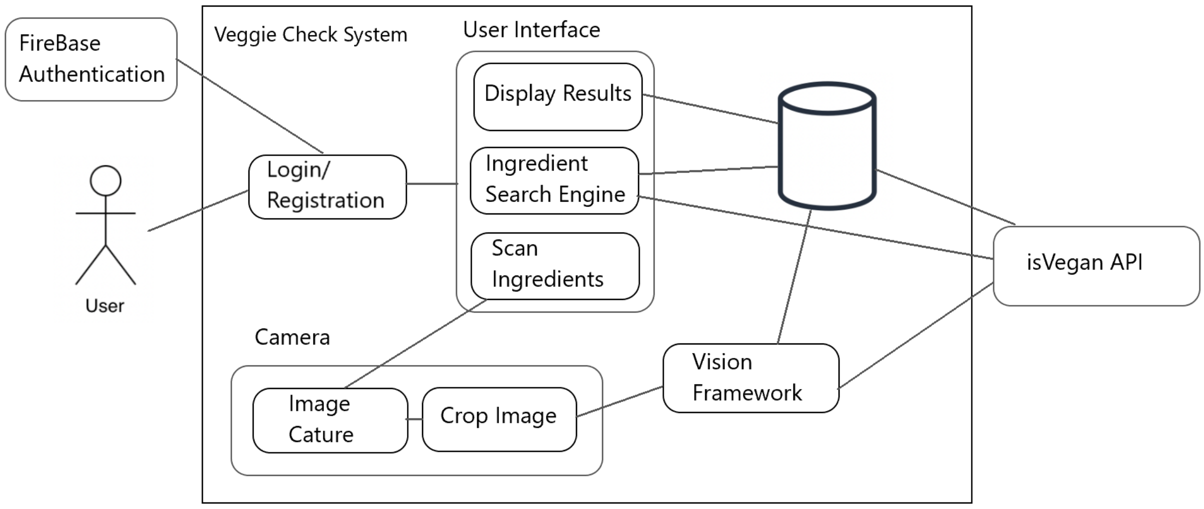


3.3.2 Scanning Ingredients



4. System Architecture

4.1 System Architecture Diagram



4.2 System Architecture Overview

The system architecture is made up of several key components; the user interface, the Vision Framework, the FireBase authentication, the isVegan API, the camera, and the local database.

4.2.1 FireBase Authentication

To use the application, the user must have an account. The user login and registration is handled through the FireBase Authentication service, which is an external component of the system. FireBase also handles other aspects of logging in such as forgotten passwords and other errors that may occur.

4.2.2 User Interface

Once the user has been authenticated, they can view the user interface of the application. It displays the functionalities of the app, divided into equal containers. The user interface has a green theme to reflect the nature of the application.

4.2.3 Camera

The camera is used to scan the food product in order to process the list of ingredients. Once the user has installed the application, they must allow it access to the phone camera. Once the image has been captured, the user must crop the image to focus on the ingredient list.

4.2.4 Vision Framework

The Vision Framework is responsible for the text recognition. Vision processes the image captured and extracts the text in the image. That text is then formatted into a string which can be used to pass the ingredients to the API.

4.2.5 isVegan API

The isVegan API is used in this application to verify if a list of ingredients are vegan friendly or not. The ingredients, separated by commas, are appended to the URL and the API returns the relevant information in JSON format. The information we are interested in is if the API deems the list vegan safe.

4.2.6 Local Database

In addition to the isVegan API, the application also has a local Core Data database to verify if a product is suitable, which is used when there is no internet connection. The persistent database also stores the 20 most recent search results of each user on the device. Data can be fetched from the database using SQL queries.

5. Implementation

We began our project by creating a project using the XCode IDE, which is the Apple IDE for creating iOS and MacOS applications. Before starting on the code, we decided to use an external API instead of a local database to check ingredients, so we did not choose to include Core Data in the original project, which caused issues later on. The first aspect of the application that we implemented was the camera and the Vision framework for the text recognition functionalities.

Once the system could capture an image and process the text in that image, we worked on being able to pass the list of ingredients to the API. To do this we created a function to change the text so that it can be appended to a URL and a function to make the API call. The data returned was not persistent and was deleted if the app was closed which is something that we later corrected when we introduced the database.

We then added user login and registration to the application and created a database. Additional code was written for the database to work as we had not included it in the original set up for the project. We extracted a list of non vegan ingredients from the PETA website, and then loaded the list into the database and created a view so that the user can scroll through the list of ingredients. With the ingredients in the database, additional functionality was added with the ability to use the app offline by querying the database instead of only relying on the API.

Using the user login details, we added an entity to the database of the 20 most recent searches of the user on that device so that the data will persist and will not be deleted whenever the user closes the app.

6. Problems Solved

6.1 XCode and Swift

At the beginning of the project, both members of the group were unfamiliar with the XCode IDE that was necessary for the application and had not written in the Swift programming language, so there was a definite learning curve involved in the project to familiarise ourselves with the different technologies. Also the SwiftUI framework that we used for VeggieCheck was only introduced in 2019, so most of the videos and tutorials online use the previous Storyboard framework for developing apps. We overcame this obstacle by using the Apple Developer documentation to assist us.

6.2 Internet Connection

In our original design, the application would use a local database to check if the ingredients on a product were vegan friendly. We later realised that an external API would provide more coverage in checking ingredients, however, using the API requires internet connection which may not be available to the user when shopping. We overcame this problem by using both the API and database, and to use them depending on the availability of internet. If there is an internet connection, the API is used, and if not the database is used.

6.3 Issues with Database

We encountered some issues with the database used in the application. SwiftUI uses a Core Data database which we were unfamiliar with. We did not select to include Core Data when creating the project in XCode and then had to write additional code for it to function properly. Another issue we had was saving data to the database. We realised you have to be inside a view file to save to the database, so to circumvent this problem, once the user has scanned the product, they must click to view the results of the scan before it can be saved to Core Data.

7. Future Work

Currently, the application displays if a product scanned is vegan friendly or not, but it does not display what the non vegan ingredients in the product are, if some have been identified. This is something that could be added to the app to give more clarity and information to the user about the foods they are consuming.

The app shows if a product is vegan friendly or not. A future development for the application would be to expand on not only checking if a product is vegan, but also checking for other dietary restrictions. These could include a gluten free diet, a vegetarian diet, or a diet catered to someone's medical needs such as products suitable for someone with diabetes or products that do not contain certain allergens.