Lambton College In Toronto

Lambton College Toronto

Capstone Project 1

Mobile Application Development

Implementation Report

IngSpector

Presentation date April 2020

Presented by

Dogukan Karayilanoglu C0755495

Rosette Lopez Pardillo C0768425

Supervisor

Mohammad Kianni

Lambton College | In Toronto

Abstract

This is a technical document that discusses and supports proposed project on how a mobile application is developed. The mobile application covers complex topics on native IOS development intended for MADT 5264 students.

This document will discuss feature specifications of a Food Allergy Detector Application. Details on the following will be further discussed in this document:

- Project Planning
- Tasks Estimation
- Tools & Technologies used
- Expected behavior and actual behavior
- Expected flow and actual flow
- Complexity and Learnings

Lambton College | In Toronto

Table of Contents

Table of Figures

Lambton College | In Toronto

IngSpector

April 20, 2020

Introduction to capstone project

Objective of the document

This document shows how IngSpector is developed and planned in support to the Proposal Document submitted. Detailed implementation information of ***IngSpector*** Mobile Application will be discussed .

Here we will go into details technical information of the following

- Black Box Implementation Design
- Mobile Application Flow
- Implementation Timeline
- Implementation
- Testing
- Tools and Technologies used

Capstone Project

Specification phase

Specification phase discussed the technical details on how the ***IngSpector*** Mobile Application is to be implemented. Our goal is to implement an application which is not only stable but as well us, implemented in a good design and maintenance friendly.

Realization phase

This Realization phase, we will give more detailed information on the implementation information as well as the actual results and timeline in developing the project.

Subject

The context of study

This mobile application project aims to deliver a stable mobile application software that covers important features that are required for a Capstone-Level Project.

This mobile application project is implemented in IOS using Swift.

Aside from its technical complexity, this mobile application project allows implementing developers to learn new knowledge that are not covered in Term 2 of MADT Program, namely, saving data into server, and analysis of image using Optical Character Recognition (OCR)

IngSpector Mobile Application targets users which has food preferences specifically the ones whose body reacts to allergens. With ***IngSpector*** Mobile Application handy and user friendly, the user can detect if food he/she is about to consume has allergen ingredients he/she is not allowed to intake.

This is not only useful to those who has allergies but helpful for hosts/event planners who wants to have a smooth party by preparing and serving food to guests without having to worry on allergy attacks.

Defining and analyzing the problem

Allergy is a person's body damaging reaction when its immune system is hypersensitive to food or harmless substances in the environment. Common allergic reactions are swelling of body, sneezing, redness, and appearance of rashes.

On allergies caused by food, allergic people tend to have difficulties ordering food from a new restaurant, and accepting unfamiliar food offered by their friends. In school, children are not even allowed to bring food that contains common allergens because children might do sharing, affecting allergic kids to have allergy attacks inside the school premises.

If only there is a faster way to know if a food contains allergen ingredients, it would be helpful to lessen such allergy attacks.

The proposal of a solution

The Idea

Prevention is better than cure. The idea is to help people detect earlier if a food in front contains allergen ingredients before consumption. Thus, a person should be able to know if he/she allowed to eat the food served on his/her table before it is too late!

Method followed to solve the problem

We are currently technology age where most tasks can be done by using a handheld device, the mobile phone. With this, most people have at least one mobile phone for entertainement, work, and digital assistance.

To prevent the Food Allergy issued as addressed, using the advantage of mobile applications as a preventive aid.

Presenting **IngSpector Mobile Application**. This application does not cure ones allergies but it helps users to prevent allergy attacks. How? By letting them know if a food the user is about to eat contains ingredients he/she is allergic to.

App Features

In this section, you must explain what features you will build for your app.

1. **User Register & Log In/Log Out:** User will create accounts and will be able to Log In/Out from different devices

2. **User Profile:** User will be able to view his/her profile.
   The user profile contains:
   - User Name
   - Weight
   - Height
   - List of allergens user is allergic to
   - List of allergic foods

   In the future, we can add social networking. Friends will be able to see publicly saved entries of their friends' entries so they may know which food their friends are allergic into. This will help how friends will prepare food in events like parties, game nights, and other gatherings.

3. Analyzing Food: The allergens of the food inputted can be done by:
   - **Analyzing Allergens by Food Label**
     User will take a photo from the food label from the device camera. Optical Character Recognition (OCR) will be applied in order to support this feature. The ingredients will be processed and will be compared to the list of allergens the user is allergic to

   - **Analyzing Allergens by Food Name**
     The application does ingredient search from the internet. If user is not connected into the internet, a warning message will be displayed that this cannot be supported offline.

     If the user is not allergic to the food, he/she will be able to see the following features:
     - User will be prompt for an option to display nutritional value after a successful search of food name
     - If user opt for nutrition value, he/she will be prompted for the food quantity then a nutritional value will be displayed e.g. "*This 85g of grilled beef will meet %25 of your daily calorie needs*"

   After the food analyzation is process, the result will be displayed on screen. The result is dependent on the allergens the user inputted in his profile.

   If the result shows the user is allergic to the input food, the name of the food and the allergic ingredient will be saved into the user list of food.

4. List of Food the User is Allergic to: Users will be able to retrieve the list of food he/she has already searched which he/she is allergic to

5. Deleting Entries: User can opt to delete a food allergen from his/her list

6. Server Data: Saved entries will also be saved into a Web Server where the user can retrieve his/her data when trying to log in from a different device

Technical Information About the App

Diagram of Application

The figure below shows a diagram on the connectivity of the application modules where the data flows.

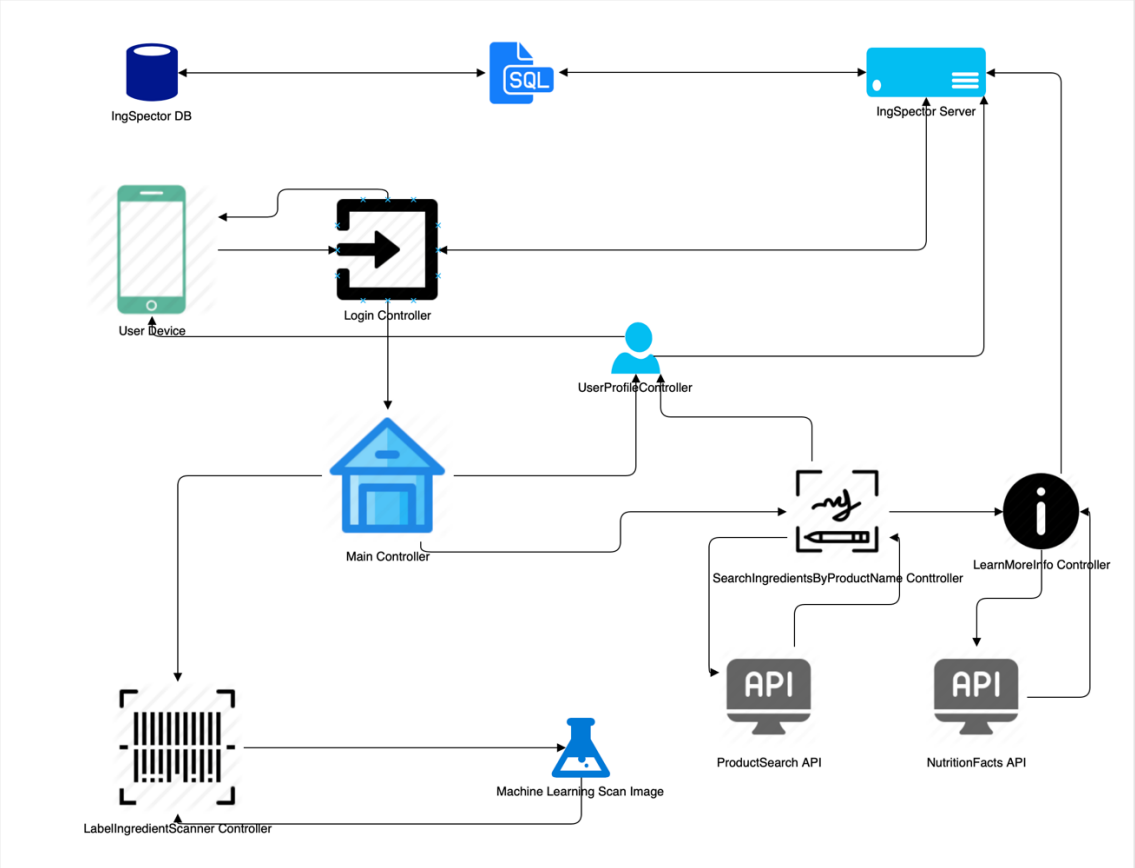


Figure 1 Application Diagram

Database

Information of user will be hold at external database. MySQL will be the database of the application and application will communicate with the database via custom server which will build by us.

Primary key will be the e-mail address of the user, CRUD operations will be operated according to e-mail address.

Attributes of Database

- E-mail Address
- Password
- Name
- Height
- Weight
- Allergens
- Allergic Foods

Software Platforms

Application will be developed via using Apple's XCode by using Swift language. Server will build as Spring Application via Spring Tool Suite and Java will be the main language for server, furthermore Xml language is necessary for settings of Pom file of Spring application.

- Native iOS Application
  - XCode
  - Swift 5
- Spring Application for Server
  - Spring Tool Suite
  - Java 8
  - Xml

Additional Software and Hardware

- Apache-Tomcat
  Container of the server will be Tomcat.
- Adobe Photoshop
  Visuals will be prepared by Photoshop.
- Postman
  Post and Get requests of the server will be tested via Postman
- Pods
  A. Alamofire
  B. SwiftyJson
  C. SVProgressHud
  D. Reachability
- API
  A. RecipePuppy Api (Ingredient Search of Any Product)
  B. Edamam Api (Nutritional Facts)
- Desktop Computer
  Device will serve as a server.

Why need for 3rd party APIs

Our application is designed to meet user's needs, so to fulfill that demand our application requires a huge database which contains a lot of information related to food and its ingredients. We do not have that information, so we have to use 3rd party APIs to gather required information.

Mockup

Mock-up Flow

Below figure is a mock up on how the application is designed to flow

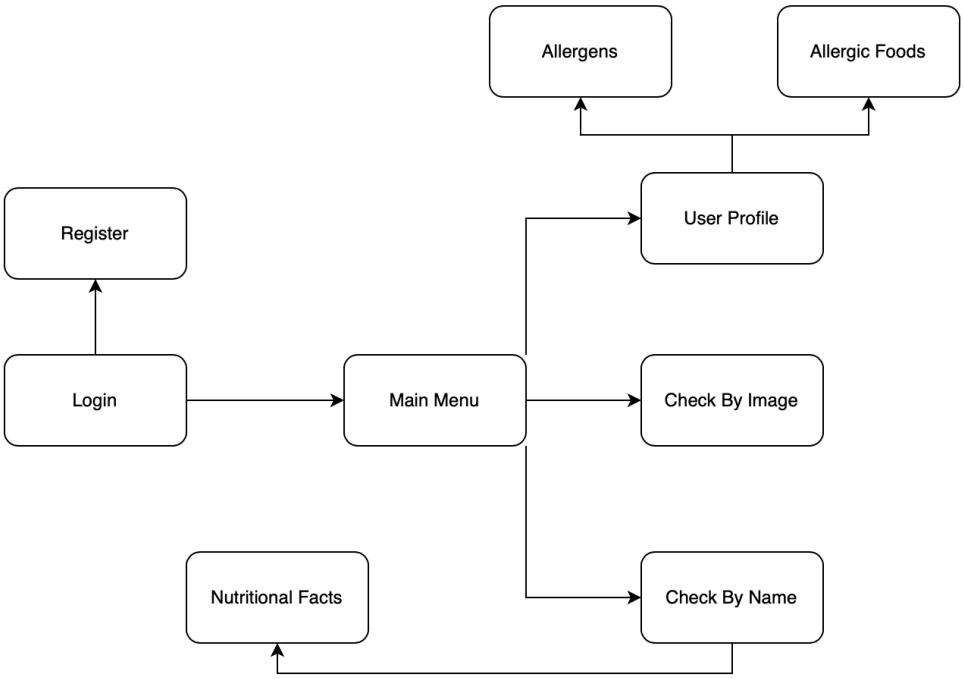


Figure 2 Mock Up Flow

Mock-up Design

Below figure is a mock up on the initial layout plan for the application, the colors may not be still final. If logged in, the user initial view will be the Home page.

In *Home Page*, the user can see buttons to navigate to other pages, the user profile, the list page. In this page, the user can do food analyzing by pressing the big plus button in the center to and will be prompted to if he/she will be analyzing food by taking a photo of food label or by inputting food name.

In *List Page*, the user expects to see all the saved food in his account.

In *Profile Page*, the user can see his profile information and option to log out.

Other pages to develop are the Results Page, Log In Page & Register Page.

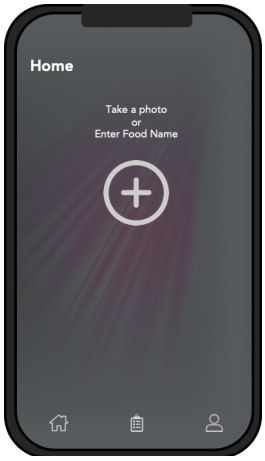


Figure 3 Home Page Mock Up

The figure below shows the design layout plan of the Log In screen ran in an Iphone 8.

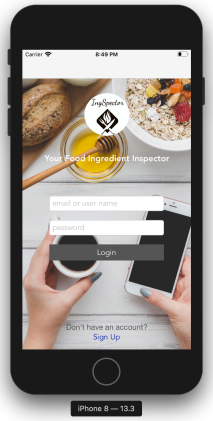


Figure 4 Log In Page in Iphone 8

The User Interface/User Flow

- Login
  - User opens application
  - If not registered
    - Push register button
    - Fill required information
    - Save information
    - Wait to be validated at server
  - Fill required information to login
  - User pushes login button

- User Profile
  - User pushes profile button in main controller
  - User fills all mandatory fields to be able to use application
  - User pushes save/update button to create its profile
  - User waits while application connect with database and save all.

- Control Allergens by Label Image
  - User pushes Control Allergens by Using Camera button
  - If button not works, user should check its profile for missing information
  - User takes the picture of the label to scan image
  - User waits while image is processing
  - User sees the alert about whether it's allergic or not

- Control Allergens by Food Name
  - User pushes Control Allergens by Food Name button
  - If button not works, user should check its profile for missing information
  - User enters the name of the product
  - User pushes the check allergens button
  - User waits while gathering required information from ingredient API
  - If ingredients of product matched with any allergens of user, user sees warning, and wait until the product added to allergic foods list of users at database.
  - If ingredients are not allergic to user, application asks to learn more about searched product.
    - If user says yes, application asks for the weight of the product and connects the nutrition API.
    - User waits while gathering required information

User sees the percentage of products calorie according to daily calorie needs which calculated by using the information of user.

Project Plan

Tasks

This application will be implemented in Swift and will initially support IOS devices.

There are two major modules for this development, the server and the mobile. The next figure lists the breakdown of tasks with the respective target sprint when it will be worked on and the estimate of work hours it requires. The sprint details will be discussed further in the next section.

	TASK	SPRINT	ESTIMATION (*hours*)
	SPECIFICATION ANAYLYSIS		
1	Implementation Design	1	3.0
	IMPLMENTATION		
	Server Side		
1	Setup Server Environment	1	4.0
2	Main Process Implementation & Database	1	12.0
3	Process Data Read	1	6.0
4	Process Data Write	1	6.0
5	Mobile Applicationn Access Function	2	12.0
6	User Credentials Support - Saving	3	4.0
7	User Credentials Support - Reading	3	4.0
8	Sending Data For Logged In User	3	4.0
	Mobile Side		
1	Development Environment Setup	1	2.0
2	Mobile Splash Screen	1	1.0
3	Log In Page	1	2.0
4	Register Page	1	2.0
5	Profile Page	1	2.0
6	Information Page	1	2.0
7	Taking Photo Handling	1	6.0
8	OCR - Image to Text	1	6.0
9	OCR - Interpreting Text Label	1	4.0
10	Analyzing Allergens from Photo Label	2	6.0
11	List Page for saved foods	2	2.0
12	Taking Food Label Photo Error Handling	2	3.0
13	Setting Up Food Ingredient API	2	6.0
14	Processing of Food Ingredient API results	2	6.0
15	Process and Show Results	2	6.0
16	Function in sending new registered user to server	2	4.0
17	Function in sending logged in user to server	2	4.0
18	Function sending logged out to server	2	4.0
19	Add food search by name	3	3.0
20	Setting Up Food Nutrition Value API	3	6.0
21	Process Food Nutrion API results	3	8.0
22	Process and show nutritional value	3	4.0
23	Function in sending food data into server	3	4.0
24	Function in retrieving food data from server	3	4.0
25	Data Persistence	3	6.0
	SERVER & MOBILE INTEGRATION		
1	Integration #1	2	8.0
2	Integration #2	3	8.0
	TEST & DEBUG		
1	Sprint 1 Deliverables Testing & Debugging	1	6.0
2	Sprint 2 Deliverables Testing & Debugging	2	6.0
3	Sprint 3 Deliverables Testing & Debugging	3	6.0
	TOTAL HOURS		**192.0**

Figure 5 Task Estimation

Both mobile and server project codes will be centralized using GIT and will be stored in GitHUB for version control. Here is the link for the repository : GitHub

Timeline

IngSpector will be implemented apllying the process of Agile Scrum. Since the implmentation phase is limited for 12 business days, development is divided into three sprints, giving 4 business days for each sprint. *See figure below for Sprint Timeline.*

DATE	SPRINT		
	1	2	3
7-Apr-20			
8-Apr-20			
9-Apr-20			
10-Apr-20			
11-Apr-20			
12-Apr-20			
13-Apr-20			
14-Apr-20			
15-Apr-20			
16-Apr-20			
17-Apr-20			
18-Apr-20			
19-Apr-20			
20-Apr-20			
21-Apr-20			
22-Apr-20			
23-Apr-20			

Figure 6 Sprint Timeline

The following Gantt chart below is the date estimates on when a specific task is planned to be focused and implmeneted.

	TASK	SPRINT	07-Apr-20	08-Apr-20	09-Apr-20				13-Apr-20	14-Apr-20	15-Apr-20	16-Apr-20	17-Apr-20			20-Apr-20	21-Apr-20	22-Apr-20	23-Apr-20
	SPECIFICATION ANAYLYSIS																		
1	Implementation Design	1																	
	IMPLMENTATION																		
	Server Side																		
1	Setup Server Environment	1																	
2	Main Process Implementation & Database	1																	
3	Process Data Read	1																	
4	Process Data Write	1																	
5	Mobile Applicationn Access Function	2																	
6	User Credentials Support - Saving	3																	
7	User Credentials Support - Reading	3																	
8	Sending Data For Logged In User	3																	
	Mobile Side																		
1	Development Environment Setup	1																	
2	Mobile Splash Screen	1																	
3	Log In Page	1																	
4	Register Page	1																	
5	Profile Page	1																	
6	Information Page	1																	
7	Taking Photo Handling	1																	
8	OCR - Image to Text	1																	
9	OCR - Interpreting Text Label	1																	
10	Analyzing Allergens from Photo Label	2																	
11	List Page for saved foods	2																	
12	Taking Food Label Photo Error Handling	2																	
13	Setting Up Food Ingredient API	2																	
14	Processing of Food Ingredient API results	2																	
15	Process and Show Results	2																	
16	Function in sending new registered user to server	2																	
17	Function in sending logged in user to server	2																	
18	Function sending logged out to server	2																	
19	Add food search by name	3																	
20	Setting Up Food Nutrition Value API	3																	
21	Process Food Nutrion API results	3																	
22	Process and show nutritional value	3																	
23	Function in sending food data into server	3																	
24	Function in retrieving food data from server	3																	
25	Data Persistence	3																	
	SERVER & MOBILE INTEGRATION																		
1	Integration #1	2																	
2	Integration #2	3																	
	TEST & DEBUG																		
1	Sprint 1 Deliverables Testing & Debugging	1																	
2	Sprint 2 Deliverables Testing & Debugging	2																	
3	Sprint 3 Deliverables Testing & Debugging	3																	

Figure 7 Gantt Chart

From the Gantt chart, the following major tasks is ecpected to be completed at the end of each sprint.

SPRINT 1

- ✓ Setup Server & Implementation on Main process
- ✓ Setup Development Environment
- ✓ Start Mobile Application Layout & Navigation
- ✓ Layout for Register, Log In/LogOut

SPRINT 2

- ✓ Delivery of Minimum Valuable Product (MVP)
- ✓ Mobile-Server Connection
- ✓ Support for Optical Character Recognition (OCR)
- ✓ Initialization in accessing Food APIs

SPRINT 3

- ✓ Full Feature Functionality
- ✓ Data Persistence
- ✓ Data is saved and retrieved to/from server

Figure 8 Sprint Deliveries

Acceptance Procedure

The following are the features and functions expected from ***IngSpector*** to define its completion

- ☐ Install IngSpector Mobile Application in any IOS device successfully
- ☐ Successful user Register, Log In & Out
- ☐ Successful image interpretation from food label photo capture with results
- ☐ Successful food ingredient search and presentation of matching allergen results
- ☐ Successful retrieving of user data from server whe user logs in from the same or different device
- ☐ IngSpector performs all of its functionalities without crashing

Implementation Phase

Technical topics from schedlue to code development of the Implementation phase will be discussed starting in this section.

Sprint and Deliverables

Achievements and realization on planned Sprint deliverables is tracked and recorded every Sprint. As planned, the project contains three sprints, each with different goals.

Tasks are managed by using a scrum board from GIT-SCRUM shareable link:
https://gitscrum.com/bp/5e9686ef2bbc1

Sprint 1

Scheduled April 7 – April 13

All goals expected to be completed for Sprint 1 is achieved

- ✓ Setup Server & Implementation on Main process
- ✓ Setup Development Environment
- ✓ Start Mobile Application Layout & Navigation
- ✓ Layout for Register, Log In/LogOut

What went well

- All team members are cooperating and focused on completing ones goals
- No delayed tasks, project development is on track

What did not went well

- Encountered issues in merging branches, took the members 2 hours
- Encountered errors in SVProgressHUD. This was immediately resolved when fixed are added to AppDelegate file

Learnings

- Learned PODS specifically Alamofire, SwiftyJSON, SVProgressHUD

Sprint 2

Scheduled April 14 – April 17

- ✓ Delivery of Minimum Valuable Product (MVP)
- ✓ Mobile-Server Connection
- ✓ Support for Optical Character Recognition (OCR)
- ✓ Initialization in accessing Food APIs

What went well

- All team members are cooperating and focused on completing ones goals
- No delayed tasks, project development is on track

What did not went well

- Errors with Alamofire crashing when installing the application to mobile. This was resolved when updating XCode. This issue occurs in Xcode 13.3.1.
- XCode 13.4 has issues with the views in Navigation. Wasted time on this.
- Git conflicts. Took time resolving the conflicts.
- Started on documentation. Documentation is not fun

Learnings

- Learned OCR by Tesseract
- Learned more information in Alamofire
- Learned Recipe Puppy API

Sprint 3

Scheduled April 20 – April 23

This sprint is scheduled in a worst case development scenario as the project is due on the April 20th with a grace period of 3 days. Since this is the proposal schedule so this is left to be this way. Despite the schedule, developers are able to complete the following without any.

- ✓ Full Feature Functionality
- ✓ Data Persistence
- ✓ Data is saved and retrieved to/from server

What went well

- Project is completed despite of schedule

What did not went well

- Unequal distribution of days in Sprints
- Documentation is not included in Sprint Task Estimation
- Buffer time (weekends) used to meet deadline

Learnings

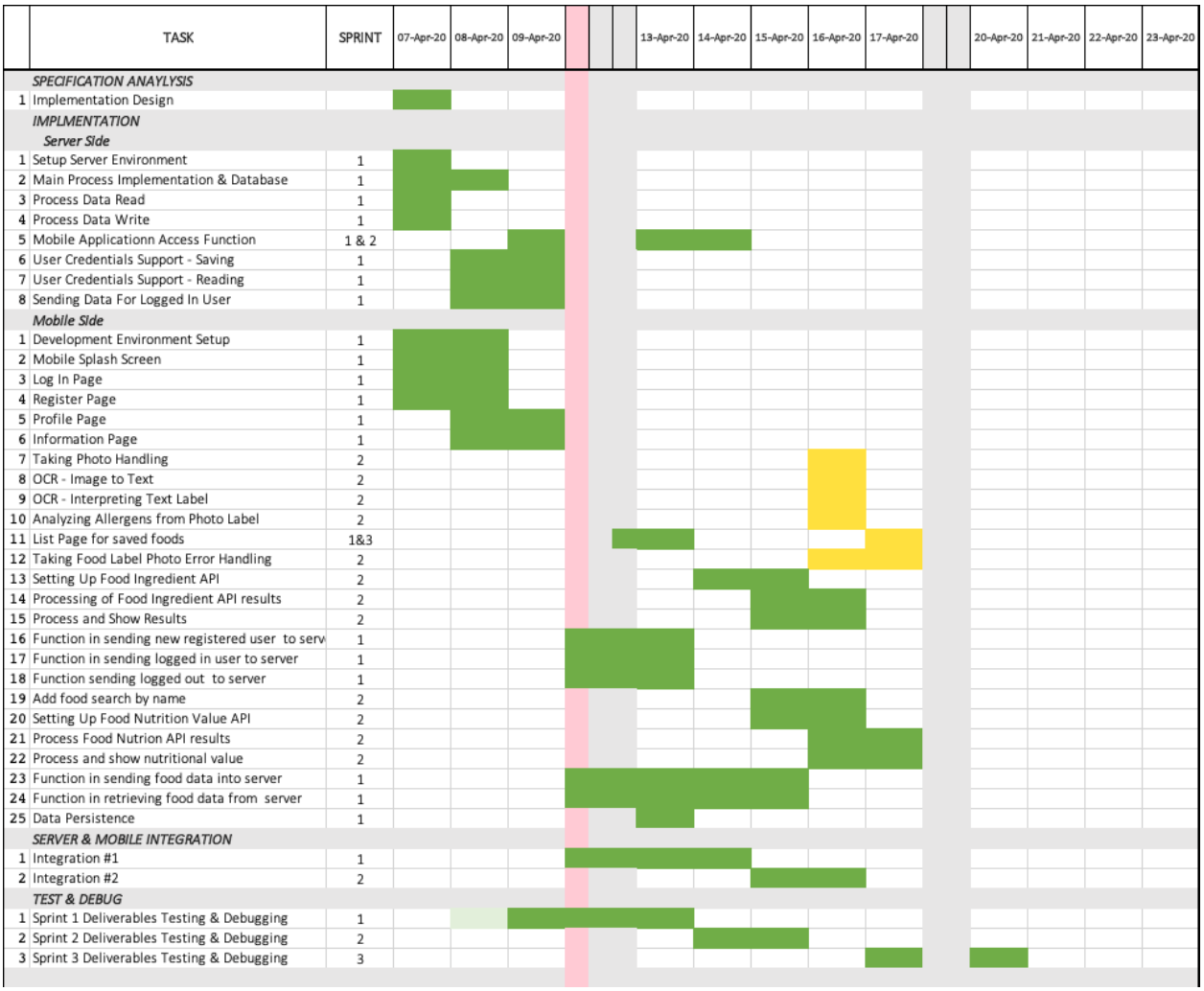
- Time management

Since tasks are completed earlier, new features are added into the project to make *IngSpector* so that users have more *IngSpectacular* experience.

- Food List Search
- Food List Swipe Down Update

Actual Task Timeline

Below figure is the Gantt chart with timeline on which Sprint and actual dates when a specific task is worked on. Despite the ***IngSpector*** being completed earlier, there are tasks that are worked on dates pass its expected timeline. For comparison, refer to *Figure 7 Gantt Chart* or check Gantt in Sharepoint.



	TASK	SPRINT	07-Apr-20	08-Apr-20	09-Apr-20			13-Apr-20	14-Apr-20	15-Apr-20	16-Apr-20	17-Apr-20			20-Apr-20	21-Apr-20	22-Apr-20	23-Apr-20
	SPECIFICATION ANAYLYSIS																	
1	Implementation Design																	
	IMPLMENTATION																	
	Server Side																	
1	Setup Server Environment	1																
2	Main Process Implementation & Database	1																
3	Process Data Read	1																
4	Process Data Write	1																
5	Mobile Applicationn Access Function	1 & 2																
6	User Credentials Support - Saving	1																
7	User Credentials Support - Reading	1																
8	Sending Data For Logged In User	1																
	Mobile Side																	
1	Development Environment Setup	1																
2	Mobile Splash Screen	1																
3	Log In Page	1																
4	Register Page	1																
5	Profile Page	1																
6	Information Page	1																
7	Taking Photo Handling	2																
8	OCR - Image to Text	2																
9	OCR - Interpreting Text Label	2																
10	Analyzing Allergens from Photo Label	2																
11	List Page for saved foods	1&3																
12	Taking Food Label Photo Error Handling	2																
13	Setting Up Food Ingredient API	2																
14	Processing of Food Ingredient API results	2																
15	Process and Show Results	2																
16	Function in sending new registered user to serv	1																
17	Function in sending logged in user to server	1																
18	Function sending logged out to server	1																
19	Add food search by name	2																
20	Setting Up Food Nutrition Value API	2																
21	Process Food Nutrion API results	2																
22	Process and show nutritional value	2																
23	Function in sending food data into server	1																
24	Function in retrieving food data from server	1																
25	Data Persistence	1																
	SERVER & MOBILE INTEGRATION																	
1	Integration #1	1																
2	Integration #2	2																
	TEST & DEBUG																	
1	Sprint 1 Deliverables Testing & Debugging	1																
2	Sprint 2 Deliverables Testing & Debugging	2																
3	Sprint 3 Deliverables Testing & Debugging	3																

Figure 9 Gantt Chart Actual Timeline

Legend:

Business days
Holiday
Weekend
Expected Schedule
Ahead/On time
Past scheduled

Document tasks are not considered in creating the project. This should also be included in the Gantt as it is consuming Developer's time.

Actual Task Hours

Below figure compares expected implementation hours with the actual hours used. As recorded, the project took lesser hours than expected as Developers are making sure that the project must be completed by April 20th which is 3 days earlier of Sprint 3 end schedule.

	TASK	ESTIMATION (*hours*)	
		Actual	Projected
	SPECIFICATION ANAYLYSIS		
1	Implementation Design	3.0	3.0
	IMPLMENTATION		
	Server Side		
1	Setup Server Environment	3.0	4.0
2	Main Process Implementation & Database	6.0	12.0
3	Process Data Read	4.0	6.0
4	Process Data Write	4.0	6.0
5	Mobile Applicationn Access Function	8.0	12.0
6	User Credentials Support - Saving	4.0	4.0
7	User Credentials Support - Reading	4.0	4.0
8	Sending Data For Logged In User	4.0	4.0
	Mobile Side		
1	Development Environment Setup	2.0	2.0
2	Mobile Splash Screen	1.0	1.0
3	Log In Page	2.0	2.0
4	Register Page	2.0	2.0
5	Profile Page	2.0	2.0
6	Information Page	2.0	2.0
7	Taking Photo Handling	1.0	6.0
8	OCR - Image to Text	1.0	6.0
9	OCR - Interpreting Text Label	1.0	4.0
10	Analyzing Allergens from Photo Label	1.0	6.0
11	List Page for saved foods	4.0	2.0
12	Taking Food Label Photo Error Handling	2.0	3.0
13	Setting Up Food Ingredient API	6.0	6.0
14	Processing of Food Ingredient API results	6.0	6.0
15	Process and Show Results	6.0	6.0
16	Function in sending new registered user to serv	4.0	4.0
17	Function in sending logged in user to server	4.0	4.0
18	Function sending logged out to server	4.0	4.0
19	Add food search by name	3.0	3.0
20	Setting Up Food Nutrition Value API	6.0	6.0
21	Process Food Nutrion API results	3.0	8.0
22	Process and show nutritional value	3.0	4.0
23	Function in sending food data into server	5.0	4.0
24	Function in retrieving food data from server	5.0	4.0
25	Data Persistence	2.0	6.0
	SERVER & MOBILE INTEGRATION		
1	Integration #1	8.0	8.0
2	Integration #2	6.0	8.0
	TEST & DEBUG		
1	Sprint 1 Deliverables Testing & Debugging	15.0	6.0
2	Sprint 2 Deliverables Testing & Debugging	15.0	6.0
3	Sprint 3 Deliverables Testing & Debugging	6.0	6.0
	TOTAL HOURS	168.0	192.0

Figure 10 Actual Task Hours

Legend:

0.0	Hours as expected
0.0	Hours shorter as expected
0.0	Hours took more time

Deliverables Realization

The Sprint is scheduled in relation to MADT 5264 schedule which is 23rd of April instead of checking the actual Project due date schedule which is the 20th of April.
Developers are able to accomplish the project completeness despite of schedule due to buffer estimation. The number of hours my be no calculated in a worst case for a three-point estimation but the estimation done where focused only on business days, 8 hour per day.

Developers understand that this should not happen in real business project as developers should not be asked to work on holidays, weekends and over times in order to maintain work-life balance.

Development

This project is divided into two major modules: Server and Mobile. It takes an internet connection for the Mobile Application, ***IngSpector***, to be able to access data to and from server.

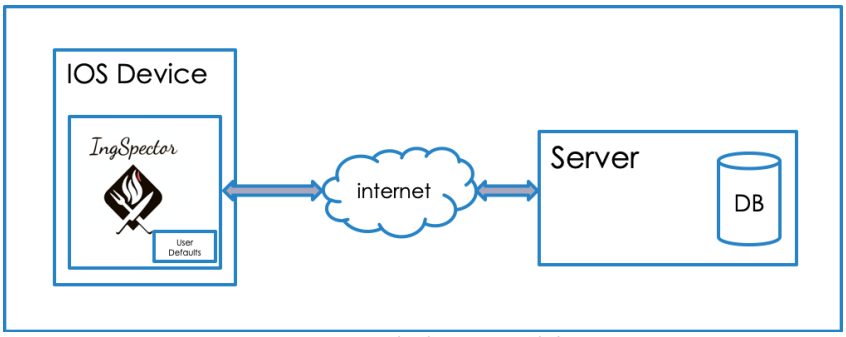


Figure 11 Black Box Module

Server

The server makes it possible for a user to view his/her latest data even when logged into different device. This is completed by storing all IngSpector user data into a database, specifically using MySQL.

The attributes of a user on server are the following:

- email : *String*
- password : *String*
- name : *String*
- height : *String*
- weight : *String*
- allergens : *String Array*
- allergicFoods : *String Array*

In order for the client to connect to server, there are instructions the server is providing.

These are the services and its corresponding url format in order for a client, in this case, the mobile application, to be able to request and send information to the server.

This is the base URL to access the server :
http://72.137.45.112:8080/ingSpectorMobileServices/ingspector/

	Instruction	URL Format	Return
1	Get all email addresses	Base URL + *getallemails*	JSON value
2	Get user's list of allergens	Base URL + *getallallergens*/<user-email>/*get*	JSON value
3	Get user's list of allergic food	Base URL + *getallallergicfoodlist*/<user-email>/*get*	JSON value
4	Add user	Base URL + *adduser*/<user-email>/<password>/<name>/<height>/<weight>/<allergens> allergens list must be in String sperated by a comma: e.g. allergen1,allergen2,allergen3	Boolean value *true* if SUCCESS *false* otherwise
5	Add allergic food	Base URL + *addallergicfood*/<user-email>/<allergicfood> Note: single allergic food per request	none
6	Add allergens	Base URL + *addallergens*/<user-email</<allallergens> allergens list must be in String sperated by a comma: e.g. allergen1,allergen2,allergen3	none
7	Remove allergen	Base URL + *removeallergen*/<user-email>/<allergen> Note: single allergen per request	none
8	Update user	Base URL + *updateuser*/<user-email>/<allergen>/<height>/<weight> allergens list must be in String sperated by a comma: e.g. allergen1,allergen2,allergen3	none
9	User Log In	Base URL + *userlogin*/<user-email>/<password>	Boolean value *true* if SUCCESS *false* otherwise
10	Get all user data	Base URL + *userinfo*/<user-email>/get	JSON value

Figure 12 Server Instructions

Mobile

The Mobile Application code is implemented in XCode using Swift. The Xcode Poject consists of different controllers that corresponds to different pages of the mobile application.

Log In Page

The Log In Page is the first page the user will see after installation. The user has two options either to log in his/her credentials or to Sign up as a new user. Similar with other social media applications, once the user is logged in, the user information is automatically saved to UserDefaults of the device. Thus, the user does not need to log in again if he/she opens IngSpector again. The user can opt to log out if he/she wishes to log in as a different user from the User Profile page.

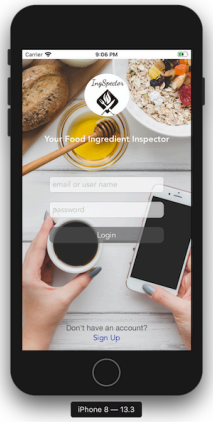


Figure 13 Actual Log In Page

This page is also tested to return information to errors to users when there are errors logging in. Fields will be highlighted when user opts to log in without completing the fields and the user will also be notified if the credentials logged is not existing.

In the XCode Project, the implementation of Log In Page is implemented in a UIViewController and is set to be the initial view of the story board.

Register Page

The user can register for a new account in the Register Page. It is just easy, he/she just need to complete the required fields in the list.

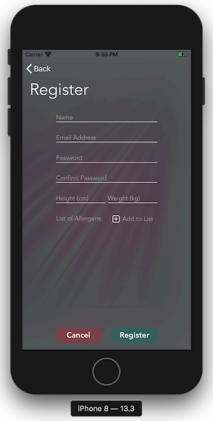


Figure 14 Actual Register Page

Incomplete fields and repeatition of an existing email will result to unsuccessful Registration. Empty fields will be highlighted to let user know which field has not been filled. Upon tapping register button, inputted details will be sent to server. Successful registration will send view back to Log In Page, as well as when user opt to press the Cancel button.

In XCode Project, the implementation of Register Page is implemented in a UIViewController.

Home Page

The Home Page sus the first view the user sees after he/she is logged in. If the user has not logged out, he/she will not be prompted to log in again and will be diverted directly into this page.

Figure 15 Actual Home Page

In this page the user can perform IngSpector awesome functionalities, analyzing food by taking photo of food label or analyzing food by searching food ingredients by food name.

The user can traverse from different pages, User Profile Page and Food List Page, from this page. This page is implemented in a UIViewController in Xcode Project.

Analyzing Food By Taking Photo of Food Label. Upon camera button press, the user willl be diverted to camera to take photo. If the photo is taken, the image will be analyzed using and text will be extracted using Optical Character Recognition (OCR) by Tesseract. Once the text of the images are extracted, these will be analyzed and matched if it contains words matching the allergen the user has registered into his/her account. If it does, the user will be notified, the user will be asked for the food name and the food name will be automatically listed into the users account under Food List. This data will also be uploaded to the server.

Analyzing Food By Entering Food Name. Upon search button press, the user will be asked to input the food name he/she wants to be analyzed. Using Recipe Labs API, IngSpector will be able to retrieve the list of ingredients the inputted food has in a form of JSON file. This JSON returned file is parsed and is compared to the allergen list the user registered to his/her account. If there are matching ingredients or allergens, the user will be notified that he/she is allergic to the inputted food and this food will automatically listed into the users account under Food List. This data will also be uploaded to the server. If the food does not contain allergens the user is allergic to, nutritional facts will be given for the user as a bonus information if the user will take the food. The nutritional fact is made possible by using the Ednama API.

Food List Page

The Food List Page is always updated from the server data to make sure the user is viewing the latest details. In this page, the user will see his/her list of Food he/she is allergic to.

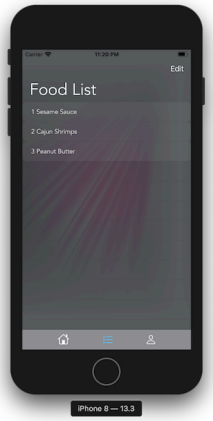


Figure 16 Actual Food List Page

Modifying the list will only allow the user to reorder the arrangement of the foodlist.

The user can traverse from different pages, Home Page and Food List Page, from this page. This page is implemented in a UITableViewController in Xcode Project.

User Profile Page

In this page, the user can see his/her profile details. Fields Weight, Height and List of Allergens are editable but fields Email and Name cannot be changed.

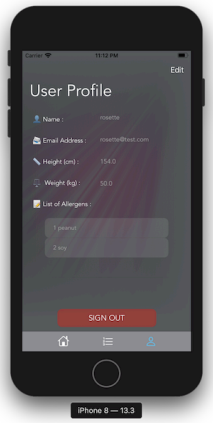


Figure 17 Actual User Profile Page

The user can traverse from different pages, Home Page and Food List Page, from this page. If user tries to traverse while on editing mode, the user will be prompted if he/she wish to discard his/her changes.

This page is implemented in a UIViewController in Xcode Project.

Repository

All Development codes are centralized and GIT in GitHUB is used to version control. The repository is IngSpector with two sub folders for mobile and for server.

The code is committed in a private GIT Repository in GitHUB:
https://github.com/otetLopez/IngSpector.git

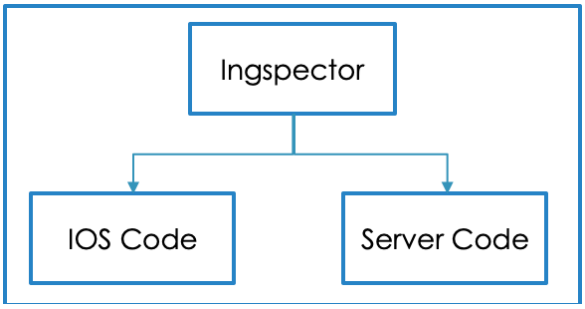


Figure 18 Repository Composition

Project Results

Analysis on Specification Phase vs Realization Phase

IngSpector Mobile Application is developed as expected as planned in Specification Phase. Although there were changes in schedule, project is submitted on time, tested, and with quality.

Since the project is completed ahead of SPRINT schedule, ***IngSpector*** Developers are adding more features to give users ***IngSpectacular*** experience.

- Food List Search
- Food List Slide Down Update

Testing Results

The test cases are created basing from Acceptance Procedure presented during the Specification Phase.

- Install IngSpector Mobile Application in any IOS device successfully

	TEST CASE	EXPECTED RESULT	RESULT	COMMENT
1	Install in IPhone XS Max	Installs successfully and user is directed to Log In Page	PASSED	
2	Install in IPhone X	Installs successfully and user is directed to Log In Page	PASSED	

Figure 19 Test Cases 1: Installation

- Successful user Log In

	TEST CASE	EXPECTED RESULT	RESULT	COMMENT
1	Press Log In without filling fields	Page will not traverse to Home Page. Email and Password fields are highlighted	PASSED	
2	Press Log In without filling Password field	Page will not traverse to Home Page. Password field is highlighted	PASSED	
3	Press Log In without filling Email field	Page will not traverse to Home Page. Email field is highlighted. Password characters are secured	PASSED	
4	Log In with incorrect details	Page will not traverse to Home Page. Email and Password fields are highlighted. User will be notified of incorrect credentials	PASSED	
5	Log In with correct details	Page will log in successfully and will traverse to Home Page	PASSED	

Figure 20 Test Cases 2: Log In

- Successful user Log Out

	TEST CASE	EXPECTED RESULT	RESULT	COMMENT
1	Press Sign Out from User	User will be prompt for Sign Out	PASSED	
2	Sign Out No	Page will not traverse to Log In Page	PASSED	
3	Sign Out Yes	Page will traverse to Log In Page User is successfully logged out	PASSED	

Figure 21 Test Cases 3: Sign Out

- Successful user Register

	TEST CASE	EXPECTED RESULT	RESULT	COMMENT
1	Press Register without filling fields	Page will not traverse to LogIn Page. Missing fields are highlighted	PASSED	
2	Press Register with Password length less than 6 characters	Page will not traverse to Log In Page. Register will not be successful. Password is secured.	PASSED	
3	Press Register with passwords not matching	Page will not traverse to Log In Page. Register will not be successful. Password is secured	PASSED	
4	Cancel Registration	User will be prompted for cancellation	PASSED	
5	Cancel Registration Yes	Page will traverse to Log In Page	PASSED	
6	Cancel Registration No	Page will remain to Register Page Fields will not be cleared	PASSED	
7	Successful Registration	User will be informed of a successul Registration	PASSED	

Figure 22 Test Cases 4: Register

- Successful image interpretation from food label photo capture with results

	TEST CASE	EXPECTED RESULT	RESULT	COMMENT
1	Take Photo of Food Label	Prompts user that food is safe if no allergen matches the user profile	PASSED	
2	Take Photo of Food Label	Prompts user that food is not safe because it contains allergen that matches the user profile	PASSED	

Figure 23 Test Cases 5: Image To Text

- Successful food ingredient search and presentation of matching allergen results

	TEST CASE	EXPECTED RESULT	RESULT	COMMENT
1	Search Food Name	Prompts user that food is safe if no allergen matches the user profile	PASSED	
2	Search Food Name	Prompts user that food is not safe because it contains allergen that matches the user profile	PASSED	

Figure 24 Test Cases 6: Ingredient Search

- Successful retrieving of user data from server whe user logs in from the same or different device

	TEST CASE	EXPECTED RESULT	RESULT	COMMENT
1	Log In of Existing User	User Profile displays correct user details and food list	PASSED	

Figure 25 Test Cases 7: Retrieving Server Data

- IngSpector performs all of its functionalities without crashing

	TEST CASE	EXPECTED RESULT	RESULT	COMMENT
1	Log In Page	Performs without crashing	PASSED	
2	Register Page	Performs without crashing	PASSED	
3	Home Page	Performs without crashing	PASSED	
4	User Profile Page	Performs without crashing	PASSED	
5	Food List Page	Performs without crashing	PASSED	
6	Analyze Food By Name	Performs without crashing	PASSED	
7	Analyze Food By Taking Photo	Performs without crashing	PASSED	

Figure 26 Test Cases 8: Crash Test

Learnings

IngSpector Developers not only enhanced mobile development skills in Native IOS but also enhaced other development skills.

Server Skills transmitting and receiving data from mobile application,

Time Management developing the project on time, following sprint deliverables as a guide, as well as, preparing the required documents and presentation.

Agile Scrum Learning Agile Scrum development cycle by doing development in sprints and using scrum board, specifically GIT-SCRUM.

Team Work Working on project with good team work cooperation, sharing responsibilities, focused in one goal which is completing IngSpector on time, tested, and of quality in both performace and code.

Conclusion

IngSpector is a fun project. Developers have enhanced their development skills in building it.

The use of Tesseract OCR is interesting but it has limitations. It cannot detect handwriting and is limited to specific fonts only. Images should also be aligned properly so the text on the images can be recognized. With this, it is more reliable to do the search and input the food name conpared to taking picture of the food label.

This application is very useful and if given the chance to be presented to clients, this project has a big possibilty that will be invested in or be partnered with existing mobile applications. Yuka, a mobile application that detects chemicals in Food is quite popular in France but this that does not detect allergens.

As awesome as it already is, there is still a lot features that can be added to improvement *IngSpector*. We can add food images in the food list, improve message flight between server and mobile clients, user profile picture, sharing of food list among friends, and many more.

Teamwork

Two developers will be working on the development of *IngSpector* Mobile Application. Both developers are expected to contribute in documentation, presentation, testing and integration aside from coding development.

The Plan. This project will be implemented in Agile Scrum and all broken down tasks will be placed in Scrum board. Free developer will pick a task which he/she wants to accomplish for an expected amount of time. Any delays and extenstions needed due to blockers will be discussed within the team for immediate resolution. The team aims to develop this project in a fast phased development at the same time delivering a stable, full-fucntionaing, user friendly project.

Actual Task Distribution. Developers are able to achieve the Agile Scrum approach were broken down tasks are defined and free developer picks a tasks he/she wants to work on. But, one developer has the computer that was used as the server so, one developer did not pick and contribute server development.

Here is the link to the team's scrum board : GitScrum-IngSpector

Bibliography

a. Recipe Labs. (2020). *Recipe Puppy*. **Recipe Labs**. http://www.recipepuppy.com
b. Edamam. (2020). *Nutrition Analysis API*. Edamam. https://developer.edamam.com/edamam-nutrition-api
c. LindSey Scott. (May 20 2019). *Tesseract OCR Tutorial for IOS*. **Raywenderlich.com**. https://www.raywenderlich.com/2010498-tesseract-ocr-tutorial-for-ios