



# **Software Engineering Group Project: Nutritics**

## **Development Report**

**Client: Damian O'Kelly**

**Group 15**

Rory Flynn, Cormac O'Higgins, Leong Kai Ler, James Healy, Owen Duffy, Luan Williams,

# **1 Introduction**

- Background and Problem Statement
- Technical Approach (how the work was done)

# **2 Requirements**

- Functional and non functional requirements for the Project
- User Interaction Scenarios including Use Case diagrams

# **3 Design**

- Design used including
- Architecture diagram
- UML Class and Sequence diagrams

# **4 Implementation**

- Tools/Libraries/Platforms used
- User interface
- Implementation of algorithms described

# **5 Conclusions**

- Problems encountered during design/implementation
- Self Evaluation of how well the project objectives were reached
- Self evaluation of working as a group
- Description of algorithms and suggestions for enhancements

# 1 Introduction

- Background and Problem Statement

Nutritics provide softwares that manage recipes, analyse diets and activities, thus creating meal plans to their clients, such as dietician, sport nutritionist and food manufacturer sectors. To improve their efficiency in collecting information, they have decided to implement an API system with Optical Character Recognition (OCR) feature is to interpret nutrition labels and organise them into trustworthy and accurate information stored in Nutritics database.

This project comprises of a new feature offered by Nutritics as an approach to address the problem of disjunction between software developers and nutritional professionals. This new feature is required to turn photos of food products' nutrition panels into text form and thus, adding newly gathered information into Nutritics database.

- Technical Approach (how the work was done)

To design the OCR feature, we have implemented Google Vision Optical Character Recognition to extract information from images passed in. Then, multiple checks have to be run on the JSON results from Google Vision OCR to ensure only the requisite nutrients and their corresponding amount information is extracted.

The checks are backed up by a set of functions that can tolerate a percentage of errors of texts in the label. Essentially, flexible checks are implemented to accommodate all kinds of situations, such as distorted images or different types of listing values method on labels. Then, all the information collected will be put into a JSON file and to be stored in Nutritics database. All the codes are written in PHP.

## 2 Requirements

- Functional and non functional requirements for the Project

### Functional Requirements

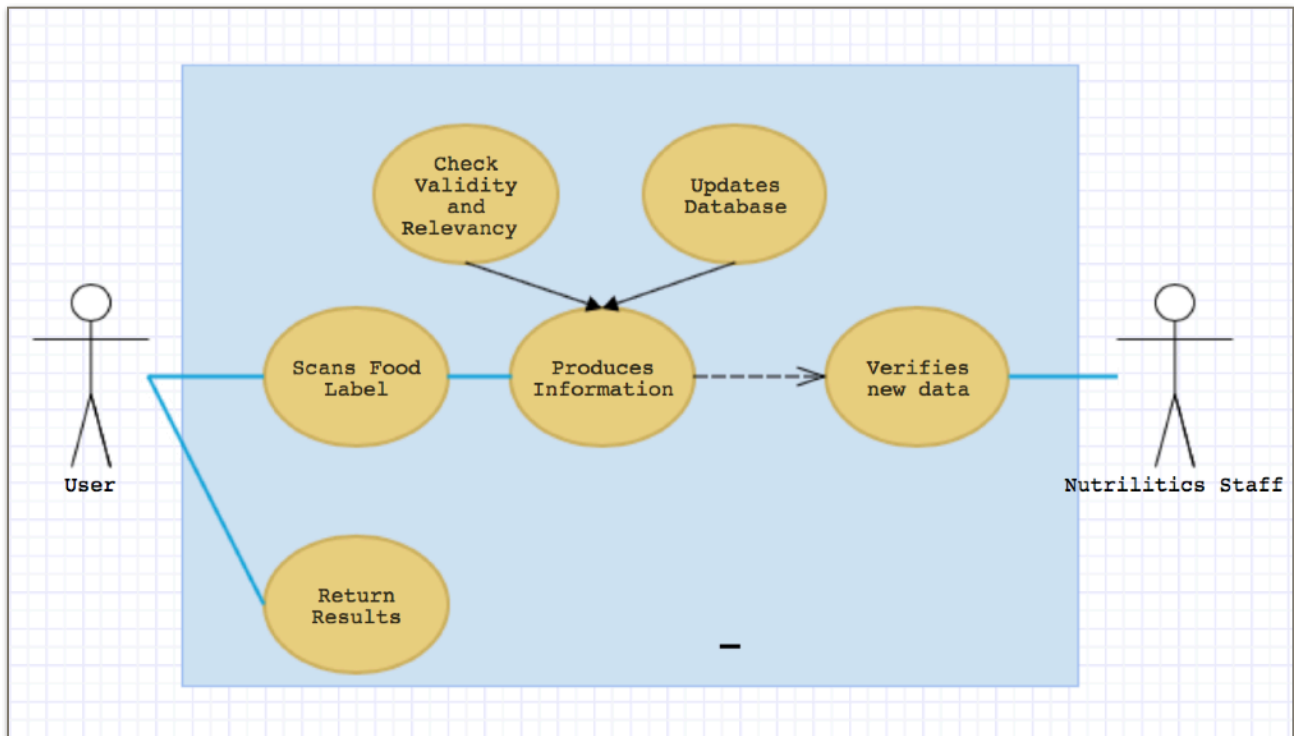
The designed API must be able to

- at least scan labels of food products designated by the European Food Safety Authority (EFSA).
- read relevant information on the labels and interpret the important parts as nutritional facts of the product.
- return a text form result that contains types of nutrients and its nutritional content per 100g.
- determine the validity and trustworthiness of the information collected.
- Semi-Automated (verification is partly done by Nutrilitics staffs)
- updates the information on the server database.
- incorporates machine learning onto its functionality, so that it will be able to learn from verification mistakes and failures. (This is for further enhancement)

### Non-Functional Requirements

- Provides reliable and clear scan on the food label
- Flexible and works in different structures of food labels
- Able to gather correct and relevant details from the scan
- Securely updates the newly added information onto the database
- Responsive to mobile devices mainly
- Capacity for addition functionalities (eg. machine learning for error detections)

- User Interaction Scenarios including Use Case diagrams



## UML Use Case Scenario Descriptions

1.

Name:	Scans Food Label
Participating Actor:	User
Entry Condition:	User scans a label on a food product
Exit Condition:	Scanning was successful.
Normal Scenario:	User scans the image of the label by accessing the camera of the mobile device with the app.
Error Scenario:	Scanning failed.

2.

Name:	Verifies new data
Participating Actor:	Nutrilitics staff
Entry Condition:	A food product (can also not be a food) with a food label or nutritional values per serving unrecognised or unseen by the OCR API
Exit Condition:	The new food product is verified by Nutrilitics staff.
Normal Scenario:	Nutrilitics staff checks if the label scanned is actually a food. Nutrilitics calculates the nutritional values per serving of the food listed on the label to determine if the values are true, valid and relevant.
Error Scenario:	Nutrilitics staff determined errors or fallacy on the label and thus verification of the new data fails.

3.

Name:	Return results
Participating Actor:	User
Entry Condition:	Verification of a new data scanned a user is done.
Exit Condition:	The user receives a JSON text format of the contents verified and its results.
Normal Scenario:	The OCR API creates a JSON text format of the food label scanned. The OCR API sends the text and results to the user.
Error Scenario:	The user does not receive the relevant or correct results and text.

4.

Name:	Produces new Information
Participating Actor:	-(managed by the OCR API)
Entry Condition:	The OCR API receives a new image scanned by the user.
Exit Condition:	The new image is interpreted successfully and relevant information on it is extracted to form a new piece of nutritional information of a food product.
Normal Scenario:	The OCR API attempts to read and interpret relevant information on the food label image it receives. The API checks the validity and relevancy of the information and updates it on the database.
Error Scenario:	The newly scanned image cannot be recognised by the API to create a new information and is sent to the Nutrilitics staff to be verified.

5.

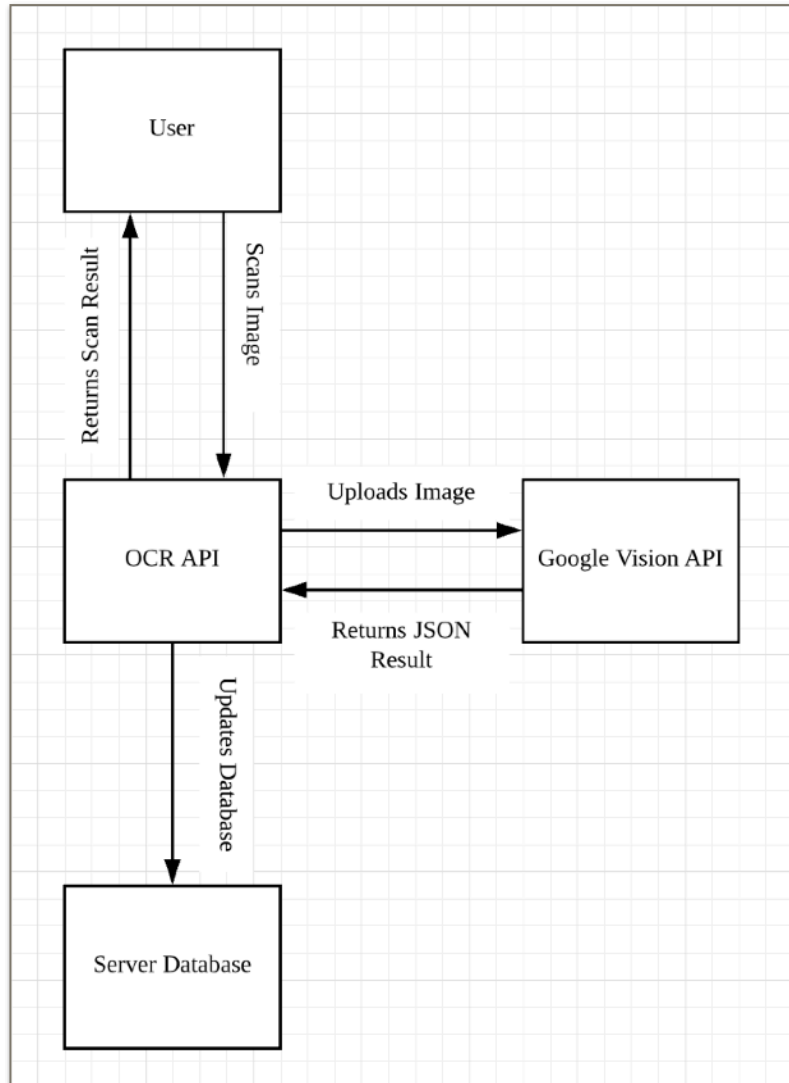
Name:	Checks Validity and Relevancy
Participating Actor:	-(managed by the OCR API)
Entry Condition:	A new information is processed by the API and needs to be examined.
Exit Condition:	The API approves the new information.
Normal Scenario:	API search for related words on the image structure. API works out if each nutritional values corresponds to their types logically.
Error Scenario:	The API is unable to determine the validity and relevancy of the information and it is sent to the Nutrilitics staff to handle it.

6.

Name:	Updates databases
Participating Actor:	-(managed by the OCR API)
Entry Condition:	A newly produced information is to be stored in the database for future reference.
Exit Condition:	The information is successfully stored.
Normal Scenario:	The API stores the information in the database by organising the correct content values of each nutrient at the right category.
Error Scenario:	The database does not accept the new information, thus update on database fails.

### 3 Design

- Design used including



The proposed project is to add a new API feature onto the current system to enhance its functionality to scan labels from food products and directly updating them on the server database. This allows the system to not only deal with food products in general, but also narrow it down to which brand of a certain type of food products is suitable to the client. Subsequently, it also saves the time of the user to fill in the form in the app as shown above.

This new API is written in PHP as it is most compatible with the system structure of Nutritics. Open sources chosen for OCR is Google Vision OCR API.

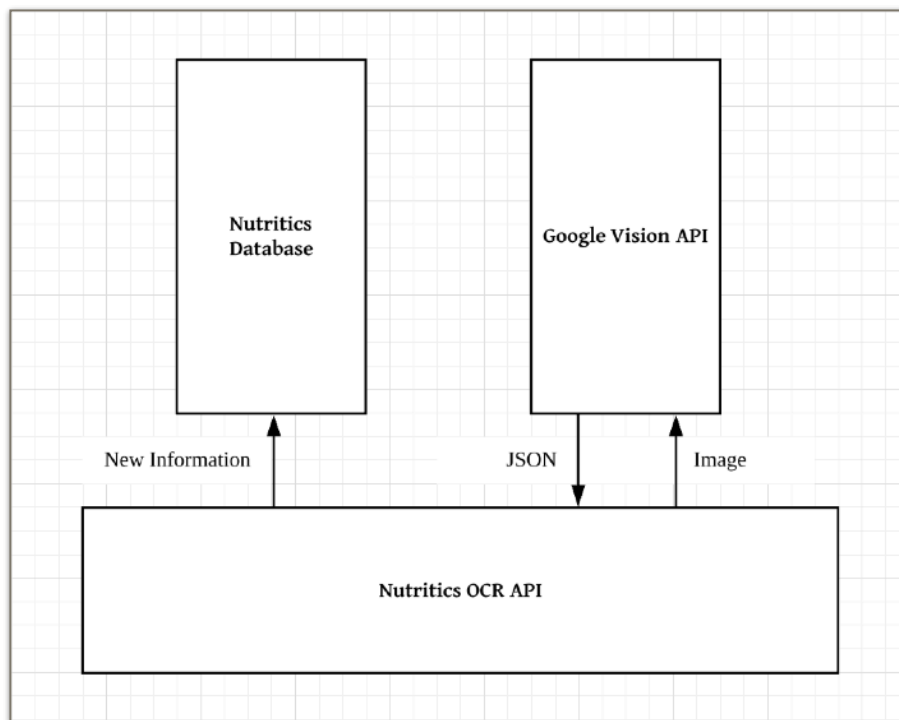
The System consists of the user scanning an image of food label using the OCR API by Nutritics.

The API then sends the image to the Google Vision API which will then returns its interpretation in the format of JSON. Then, the Nutritics OCR will perform a search on the returned result to extract relevant results. The search is assisted with a few functions to accommodate the possibility of typo errors and inconsistency in position of values on the labels, thus correcting them before incorporating them in the JSON file. All operations are run in Docker environment, as it is a versatile container platform.

The API then updates the data onto Nutritics database to be stored and returns the result to the user to verify the scan successful.



- Architecture diagram

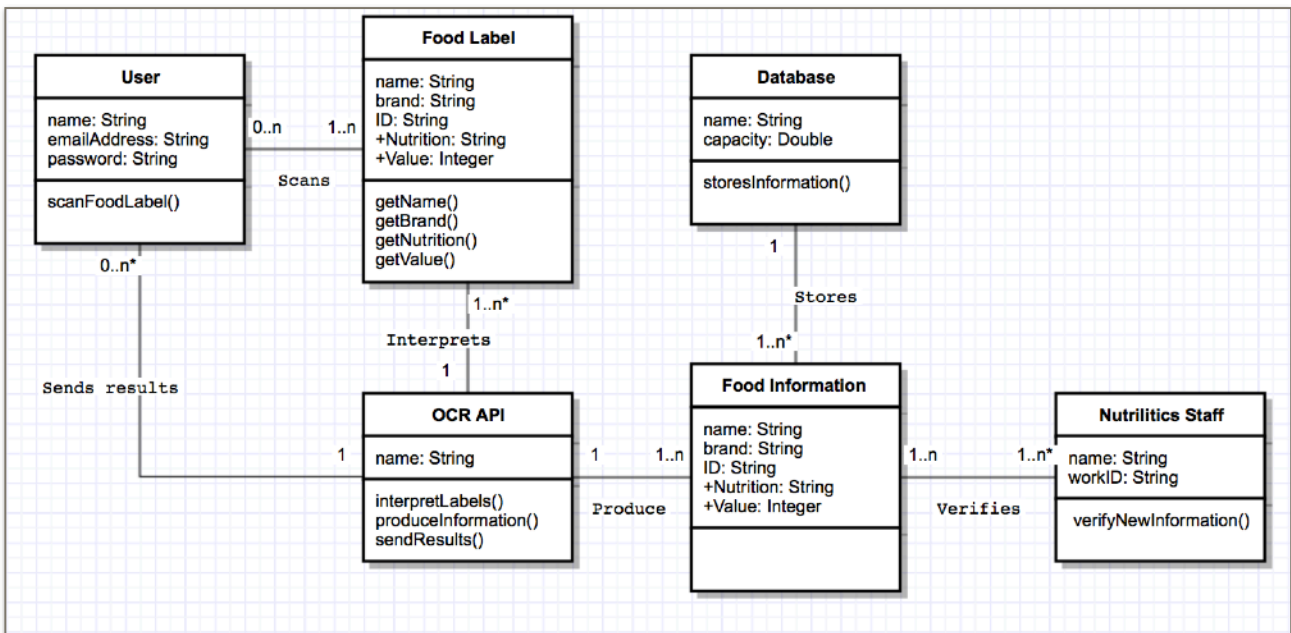


### Overview

The architecture concept employed in the system incorporates the use of Google Vision API as it can easily **detect broad sets of objects** in images commonly found within images. Moreover, it offers an excellent Optical Character Recognition (OCR) to **detect text** within images, along with **automatic language identification**. Vision API supports a broad set of languages.

In general, the Nutritics OCR API is only responsible in sending the uploaded image to Google Vision API to extract required information and updates them in the Nutritics Database. The OCR API should be able to accept various image formats such as PNG, JPG, PDF and etc. After receiving the JSON result from Google Vision API, it should be able to store the data accordingly in the server database.

- UML Class and Sequence diagrams



## Overview

The class diagram provides a representation the interaction and relationships between all elements in the operation of Nutrilitics new API feature.

Firstly, users are to scan image of food labels using services provided by Nutrilitics. The image will then be uploaded to the Google OCR vision API to scan for relevant information. Then, the Google API will return a result of requisite data in the format of JSON which will then be stored in the database. At the same time, the results have to verified by operating Nutrilitics staff to make sure the information are valid and accurate. Lastly, the JSON formatted result will be send back to the user by the OCR API in text format as verification of a successful scan.

## 4 Implementation

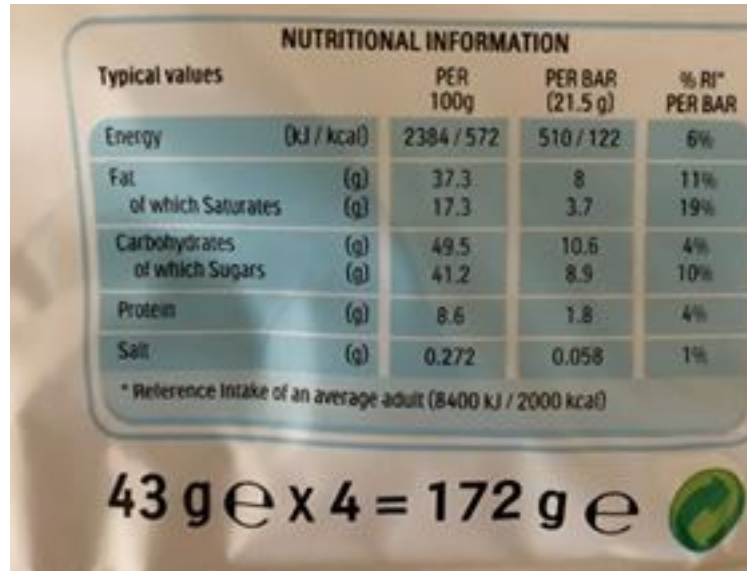
- Tools/Libraries/Platforms used

Tool / Framework	Usage
Google CLOUD VISION API	Google Cloud Vision API enables developers to understand the content of an image by encapsulating powerful machine learning models in an easy to use REST API. It quickly classifies images into thousands of categories, detects individual objects and faces within images, and finds and reads printed words contained within images. Users can build metadata on their image catalog, moderate offensive content, or enable new marketing scenarios through image sentiment analysis. They can also analyze images uploaded in the request or integrate with their image storage on Google Cloud Storage.
Git	Github is used as a version control software, so that changes made by group members won't overlap with each other and can be traced.
PhpStorm	PhpStorm provides the best code completion, refactorings, on-the-fly error prevention, and more for PHP coding. It incorporates all the features of WebStorm and offers full-fledged support for PHP and Databases/SQL support
Docker	The Docker platform is the only container platform to build, secure and manage the widest array of applications from development to production both on premises and in the cloud. Docker delivers operations at scale by addressing a diverse set of applications and infrastructure for both developers and IT.

PHP is used to develop an API that helps the client scan and uploads image to be processed by the Google Cloud Vision API and store the return JSON formatted result in the client database, which is also programmed in PHP.

- User interface

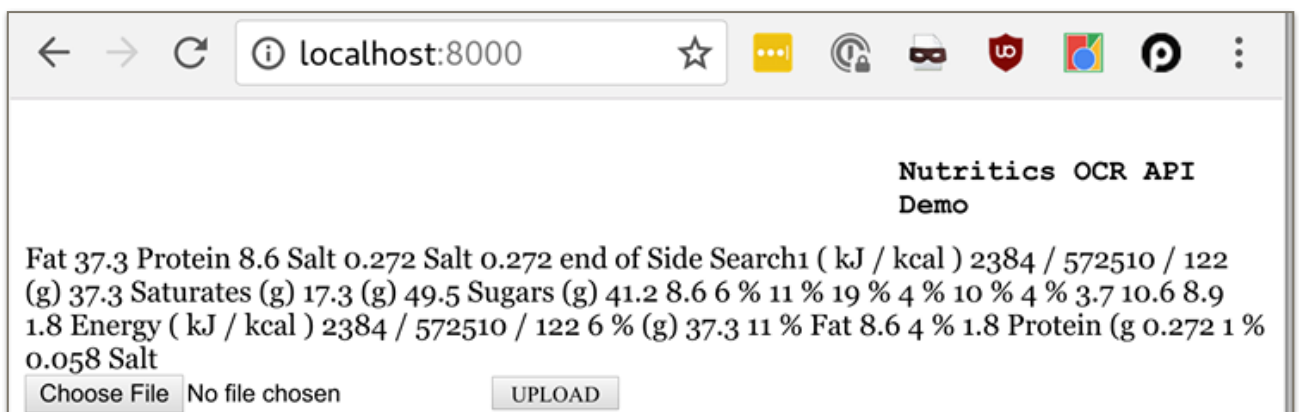
Note: The design of the project does not have any front end user interface, as it is supposed to be done by the client. As a result, only the backend idea and operation is presented here. This is an example of the JSON result to be stored in the database.



NUTRITIONAL INFORMATION				
Typical values		PER 100g	PER BAR (21.5 g)	% RI* PER BAR
Energy	(kJ / kcal)	2384 / 572	510 / 122	6%
Fat	(g)	37.3	8	11%
of which Saturates	(g)	17.3	3.7	19%
Carbohydrates	(g)	49.5	10.6	4%
of which Sugars	(g)	41.2	8.9	10%
Protein	(g)	8.6	1.8	4%
Salt	(g)	0.272	0.058	1%

\* Reference Intake of an average adult (8400 kJ / 2000 kcal)

43 g e x 4 = 172 g e



- Implementation of algorithms described

The project did not involve a lot of coding in terms of scanning for texts in the image, as most of it is done by the Google OCR API. On the other hand, a significant amount of searching and checking codes have to be implemented to get and store the correct details in the JSON result to be stored in the Nutritics database.

For example to ensure, that the right information is found at the right place, 3 different tests had to be run on three directions of 100g box, so that no overlap results or the wrong values are taken. However, despite three of the sophisticated tests implemented, it is still not able to cope with every situation due to the variety of food labels available.

## 5 Conclusions

- Problems encountered during design/implementation

Overall, the main problem faced is using new tools such as Google OCR and Docker to develop the project design. A significant amount of time has been spent just to peruse through the use the available APIs and tools usage, setting up new developing environment and learning new coding languages, in this case PHP. Moreover, the use of Github as a version control unit for the codes was also a problem in the beginning, as some of the members accidentally committed the wrong code on the master branch. Consequently, some efforts has to be done to salvage the situation. More time had to be put into making the code as sophisticated and clean as possible.

- Self Evaluation of how well the project objectives were reached

Overall, the team have achieved more than it set out for in some aspects. In the beginning, we were able to convinced the client to use Google OCR due to its advantages over other available APIs even though it costs money. Meetings were organised once every week, some times twice depending on the objectives and workload, this ensured we are always ahead and on time. By establishing an agenda in the meetings, team members have a clear insight of their tasks and were able to share and fix problems they faced while doing their part.

- Self evaluation of working as a group

The team was lucky to be under the guidance of Rory, who was very helpful in checking and correcting the errors in the code and explaining new tools, which were used in the project to the second years. Every one has worked hard to deliver their work on time and did not hesitate to request for help. This allows every one in the team to be updated with the situation early before the problem spiral out of control and were able to solve it first. As the project goes on, the teamwork of between members gradually improved as well.

- Description of algorithms and suggestions for enhancements

There are several attributes of the design that can be improved. Firstly, more in detailed and sophisticated can be implemented to accommodate more irregularities of the uploaded images. Secondly, a function that can rotate the images in case they are not in the right position before being pushed to Google OCR API to extract information. Thirdly, incorporating machine learning in the

design, so that it can deal with more variety of informations. This would require more time to develop but still it is very encouraged by the client. Lastly, most of the members have all agreed that using PHP to develop the project turned out to be a less suitable choice, as it causes a lot of problems and mistakes due to its syntax and operations.