

Cuisine 2.0

An AI-powered cuisine for smart consumption with enhanced nutrition



Team: AYB Creations

*Diploma in Information Technology*

*in*

*Big Data Analytics*

**First Review**

**Design**

**Supervisor:** Mr Zoubeir Aungnoo

**AYB Team Members:**

* BABOOLULL Yougesh
* BHAGOOLI Divesh Singh
* SUBRON Jhivesh Kumar

**Date: 07 April 2021**

# Abstract

Designing is an interesting section in the project management cycle, where developers anticipate upcoming challenges and difficulties. It is where user requirements are transformed into a functional prototype.

Our Cuisine 2.0 project has 3 main parts:

* Cuisine Vision
* Cuisine Android app
* Cuisine 2.0 Website

Each part has been linked to a main server, each accessing the same Database using our custom written PHP APIs.

//More intro to be review @Divesh @Yougesh

**Table of Contents**

[Abstract 2](#_Toc73043943)

[Potential Users 5](#_Toc73043944)

[System Architecture 7](#_Toc73043945)

[Database 7](#_Toc73043946)

[Database-Relationship Diagram 7](#_Toc73043947)

[Description of Tables 7](#_Toc73043948)

[API 9](#_Toc73043949)

[Object Detection 9](#_Toc73043950)

[Model 9](#_Toc73043951)

[Algorithm 9](#_Toc73043952)

[Android App 11](#_Toc73043953)

[Class Diagrams of Activities 11](#_Toc73043954)

[Class Diagrams of Fragments 13](#_Toc73043955)

[Website 16](#_Toc73043956)

[Cuisine Vision 20](#_Toc73043957)

[All equipment used for Cuisine Vision 21](#_Toc73043958)

[Raspberry Pi 4 Model B Overview 22](#_Toc73043959)

[Server 23](#_Toc73043960)

[User Interface 24](#_Toc73043961)

[Android App 24](#_Toc73043962)

[Raspberry Pi 29](#_Toc73043963)

[Website 29](#_Toc73043964)

**Table of figures**

No table of figures entries found.

# Potential Users

Our project may attract many people around almost all social classes. Nevertheless, the potential users of our system have specific needs that made them choose this system. The personas of our potential users:

* Busy workers

Many people work extreme work hours which leaves them very little time left to do anything else. One of the issues they face is preparing their grocery list. To prepare their grocery list, they need to be at home, check their inventory and then prepare their grocery list. This process is quite hectic for many of us as it is time-consuming especially if you work far from home. Moreover, not planning the grocery list properly, can cause overbuying or underbuying which will indeed cause issues such as food wastage and lack of food ingredients.

Cuisine 2.0 allows its users to have a virtual inventory which allows the latter to have 24/7 access of their inventory. The Cuisine 2.0 App also allows its users to manage multiple grocery lists in form of checklist to make it easier for them while shopping and also review their inventory.

* Health-conscious people

Many health-conscious people usually check all their food ingredients before planning their meal which sometimes is a bit hectic. Furthermore, some people like to plan their meal for the week.

The App has a feature that can extract details of different food ingredients which is helpful for health-conscious people. The App has a meal planner which can help them plan their week meals.

* Housewives

Housewives often face this problem of “not knowing what to cook despite having all the ingredients”. The Cuisine App have the recipe suggestion feature which suggests the user available recipes to help them decide their meal.

* Average income families

In the end, everything comes down to the cost of such a system. New Smart refrigerators such as Samsung’s family hub come at a cost of $1,889.00 to $4,599.00 which is really expensive.

Our system is different. It converts your old working refrigerator into a smart one. Now, purchasing a cheaper refrigerator is no longer an issue as this system can be integrated with any refrigerator you have. Our system has an estimated maximum cost of as low as $200 with all the components.

# System Architecture

## Database

Our project uses MySQL databases to store user credentials. The database cuisineUserAcc has 2 tables: UserAuth, User\_Secret\_Keys.

### Database-Relationship Diagram

UserAuth

User\_Secret\_Keys

u511941246\_cuisineUserAcc

Cuisine 2.0Android App

Cuisine Vision

Cuisine 2.0 Website

Via PHP APIs

## Description of Tables

The fields in the UserAuth table are: UserId, First Name, Last Name, Email Address, Verified.

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Datatype** | **Description** |
| UserId | Integer | It is a unique id assigned to each user account created to be able to identify each user correctly. |
| First Name | String | It is the First Name of the User. |
| Last Name | String | It is the Last Name of the User. |
| Email Address | String | It is the email of the User |
| Verified | Boolean | It acts as a flag to check if a user has verified his email address or not. |

The fields in the User\_Secret\_Keys table are: UserId, Password.

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Datatype** | **Description** |
| UserId | Integer | It is a unique id assigned to each user account created to be able to identify each user correctly. |
| Password | String (256) | It stores the hashed password of the user which is encrypted using PHP SHA-256 cryptographic hashing method. There is no possible way of reversing a cryptographic hashed password as it uses a one-way function which makes the user account most secured. |

The Database Inventory has a table for each user to reduce the Create, Update and Delete (CUD) Anomalies concerning Table Normalization. Each user table name is in the following format for uniqueness: User\_<unique userID>.

Each table has the following fields:

* ID: Unique Auto-increment primary key
* Item: Food Item name
* Quantity: Quantity of the food item
* Expiry: Expiry date of the food item

## API

All the APIs are written in PHP which provides data to the different system components. The API work on POST methods only to ensure security on transmission of data.

The MYSQLi procedural package was used to connect and interact with the database.

[Note: The API codes were not included in the report because there are too many APIs. But they are available in our github repository on request.]

## Object Detection

Object Detection is a Computer Vision and Image Processing technique that identify and locate objects in an image or in a real-time video feed by drawing a bounding box around the object and determine the class of the object discovered. The ultimate importance of object detection in Cuisine Vision is to offer the users the facility to automatically record the items in their inventory without the need of doing it manually, thus saving time.

**List of popular applications which make use of object detection is:**

1. Self-driving cars: Computer Vision and Image Processing are used to determine the distance between the car and the moving objects to create alerts and guides to self-driving cars.
2. Surveillance Industry: To build systems like Traffic Tracking, Activity Recognition (for example: whether the person is sitting or standing) and Sentiment Analysis are all possible through Object Detection now.

### What is the technology behind Object Detection?

* Artificial Intelligence (AI) is the basic principles that drives object detection.

The idea is simple: The data is collected through Computer Vision and then sent to a model and use a Machine Learning algorithm or Deep Learning algorithm. The camera collects the images from different angles and then sends it the Image Processing unit. Now based on how the model was trained, what the training data was, the results are laid out to the system (self-driving cars, for example) in the form of alerts and instructions like “yes/no”.

### 

Object Detection can be practiced in various ways:

1. Feature-Based Object Detection
2. Viola Jones Object Detection
3. SVM Classifications with HOG Features
4. Deep Learning Object Detection [**SELECTED**]

### TensorFlow framework for Object Detection

For our project we have decided to implement object detection via TensorFlow.

* TensorFlow is an open-source library for deep learning applications.
* For example:
  + - Text Recognition (Handwritten Text Recognition Systems)
    - Image Recognition (DeepFace, Facebook’s Image Recognition System)
    - Voice Recognition (Apple’s Siri)
* The TensorFlow object detection API is the framework for creating a deep learning network that makes it easy to construct, train and deploy custom object detection models.

### Why TensorFlow Object Detection API?

* It has a whole bunch of additional capabilities that make it a whole heap easier to be able to build object detection models. A real-time object detection architecture or model has a lot of stuff that is needed to build in order to get it up and running; there are preprocessing steps, postprocessing steps, visualization utilities, image augmentation, model builder and other important components. Building this from scratch takes a lot of effort and time, therefore, using TensorFlow Object Detection API speeds it up.
* A list of pre-trained models is available in TensorFlow Model Zoo which has different architecture and the accuracies of each model is differed from each other in terms of speed of execution and the accuracy in placing bounding boxes around the object.

### Model overview

When it comes to training a model using TensorFlow Object Detection there is a bunch of steps that is needed to go through and a bunch of dependencies that is needed. Now because TensorFlow Object Detection is being used, one of the core dependencies is TensorFlow. In order to leverage a **custom** TensorFlow Object Detection model we need to fine-tune, in other words, it means to train a custom computer vision model.

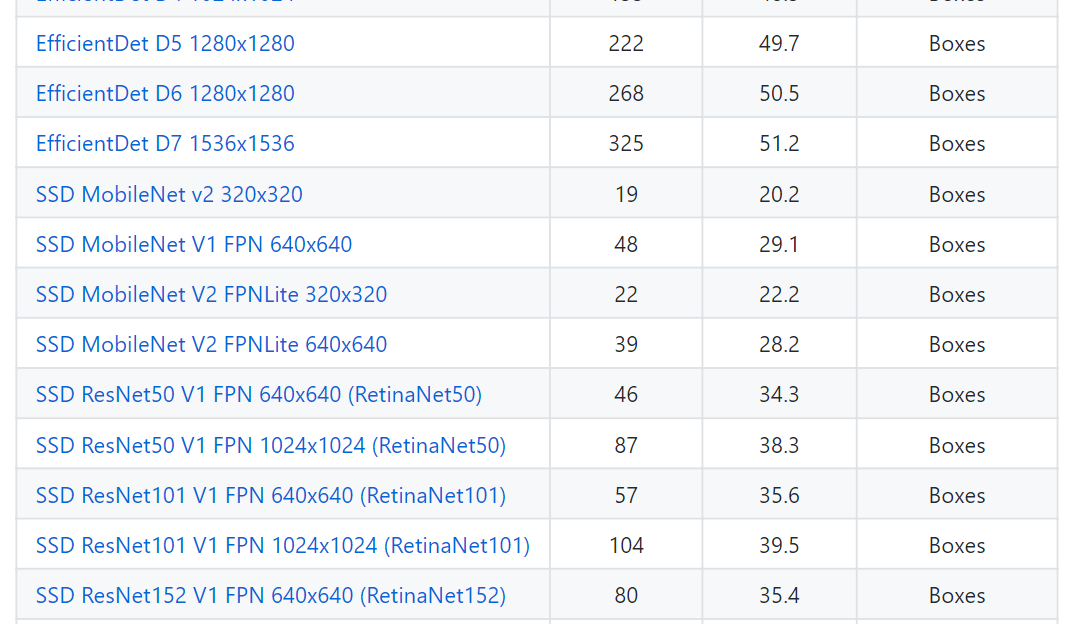
### Choosing a Model

* There are different types of models that we can make use of to train available in [TensorFlow 2 Model Zoo](https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md). What normally have to be done is balance off a trade-off; so, we should balance faster detections to ideally be able to detect objects quicky versus lower accuracy, so the ones that tend to be faster, tends to be less accurate, most of the time. The ones that tend to be highly accurate tend to be slower; the more sophisticated models, the bigger the architectures, the slower they are.

FASTER DETECTIONS = LOWER ACCURACY

HIGHER ACCURACY = SLOWER DETECTIONS

### TensorFlow Model Zoo



In our case, where we want to deploy our model on a Raspberry Pi, we want to ideally optimize for speed. The model that we have chosen for real-time detection is **SSD MobileNet V2 FPNLite 320x320,** considering that its **Speed (ms)** is 22 milliseconds per frame (the lower, the better) and it has a mean average precision (**mAP**) of 22.2, which is reasonably balanced.

**NOTE:** To compensate for the low accuracy, we might add more images to our dataset.

### Data Collection and Labelling

Before building our model, we collected our own apple, orange and yogurt datasets and labelled then. We used our webcam and mobile phone camera and label them using a software called “labelimg”.

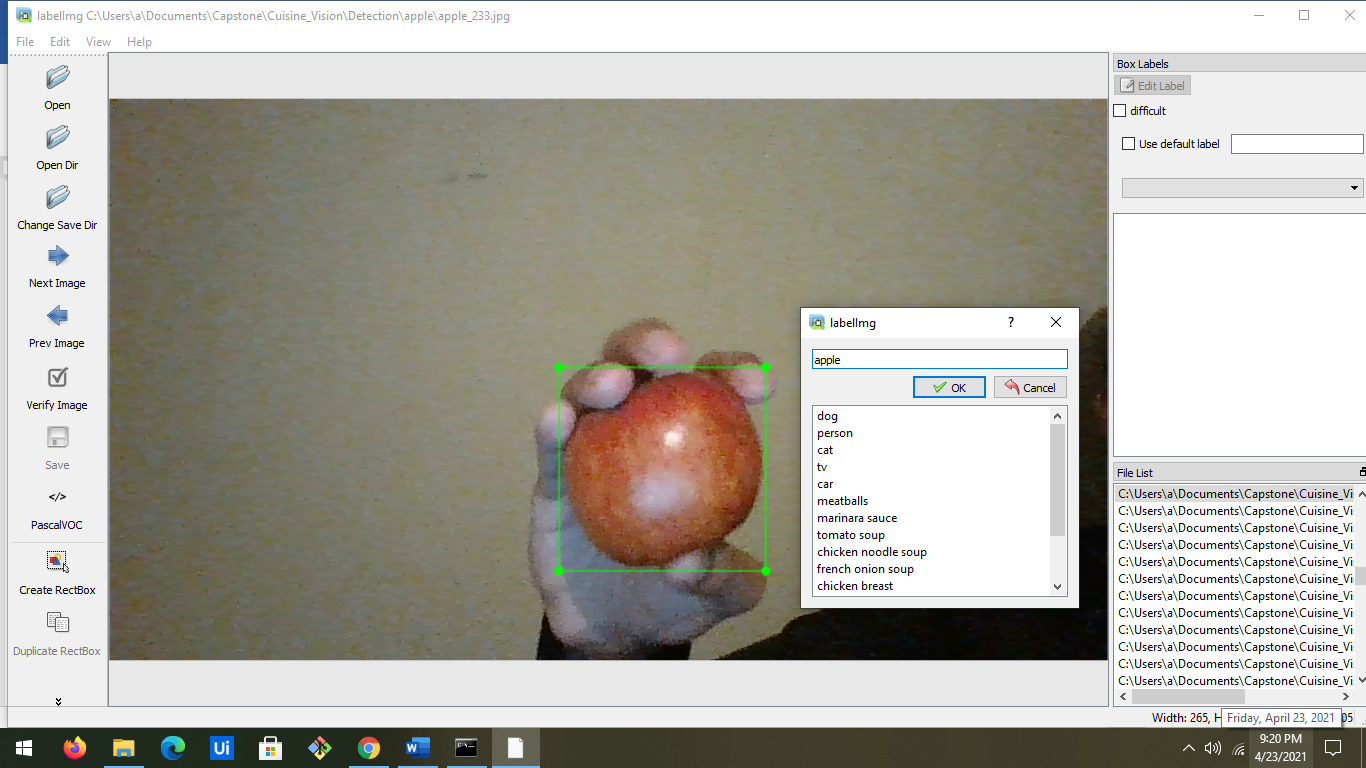
### What is Labelling?

When building an Object Detection Model, we should be able to label or identify the objects that are going to be detected. For example: We are building a detection model consisting of apple, orange and yogurt. We might take webcam or mobile phone camera to take pictures of the objects and then draw label around the objects. Then we will take those images that have been labelled and leverage them in a trained object detection model to be able to detect apple and yogurt.

### Best practices we followed when it comes to labelling:

* We kept the labels as ‘tight’ as possible to allow the model to detect specifically that object, as shown in figure \_\_ below. This helps a lot in boosting performance of the model.

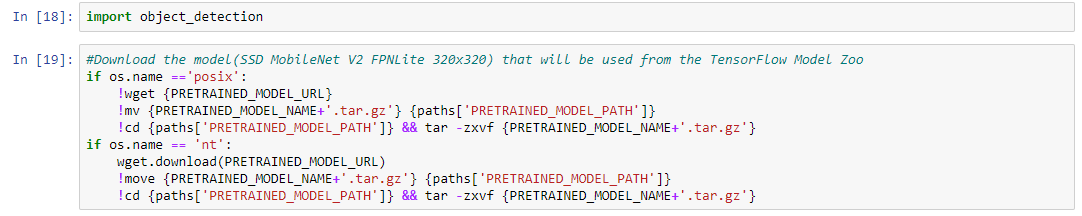
We included pictures of objects at different angles, under different lighting conditions and of different qualities and dimensions (For e.g.: the camera quality of an iPhone/Android can be slightly better than that of a PC’s webcam; both picture qualities are equally important to obtain an accurate model)

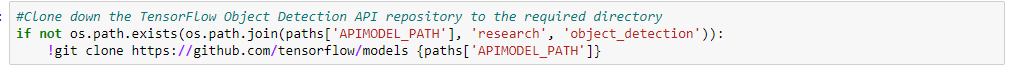


### Configuration steps prior to model training

**Step 1:**

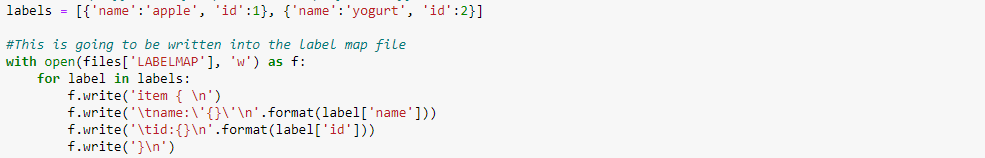
To download the **SSD MobileNet V2 FPNLite 320x320** pre-trained model that we chose to utilize from the TensorFlow Model Zoo and to install TensorFlow Object Detection API.



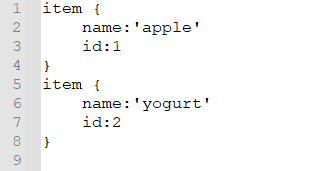


**Step 2:**

* In this step, the label map is created. A label effectively represents a map of all different labels (apple and yogurt). That is, the label maps every class name to an integer.



* The label map can be seen below from the label\_map.pbtxt file:



**Step 3:**

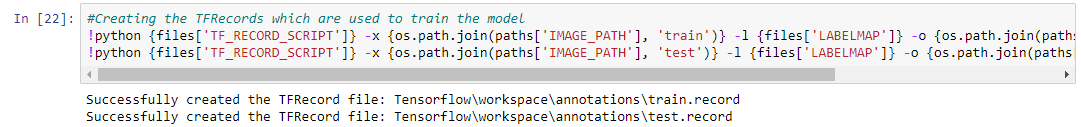
Before generating TFRecords, we moved 80 % our images into Training and 20 % into Testing partition into their respective folders with both containing a variety of each class.

****

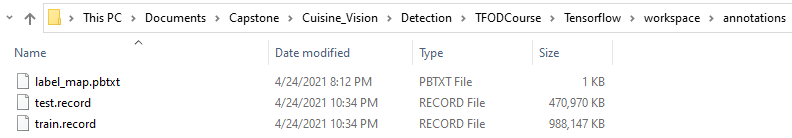
**Step 4:**

Next step, TFRecords is generated. TFRecords are a binary file format for storing data. Using a TFRecord helps speed up training our custom object detection model.





* Both test.record and train.record are binary files which contain information about the encoded jpg and bounding box annotation information for the corresponding train/test set.



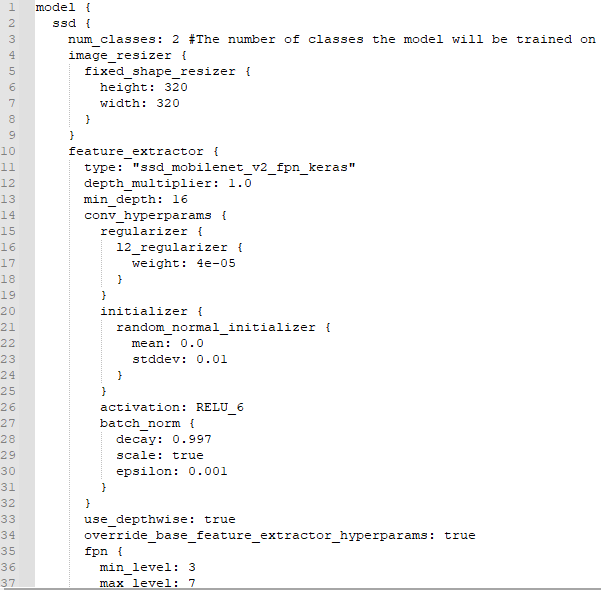
## Algorithm (Model Architecture)

### Purpose of the file pipeline.config

The pipeline.config file is what controls what the model actually looks like. This is what defines the architecture of the model. The default pipeline.config file gives us a baseline and we can use it to customize our own model accordingly.

The config file is split into 5 parts:

1. **Model** configuration: This defines on which model is trained (like post processing, box predictor).
2. **train\_config**: This decides what parameters is used to train model (like number of steps to be trained on, input preprocessing and feature extractor initialization values).
3. **train\_input\_reader**: This defines what dataset the model should be trained on.
4. **eval\_config**: This determines what set of metrics will be reported for evaluation.
5. **eval\_input\_reader**: This defines what dataset the model will be evaluated on.



## Kick-off model training

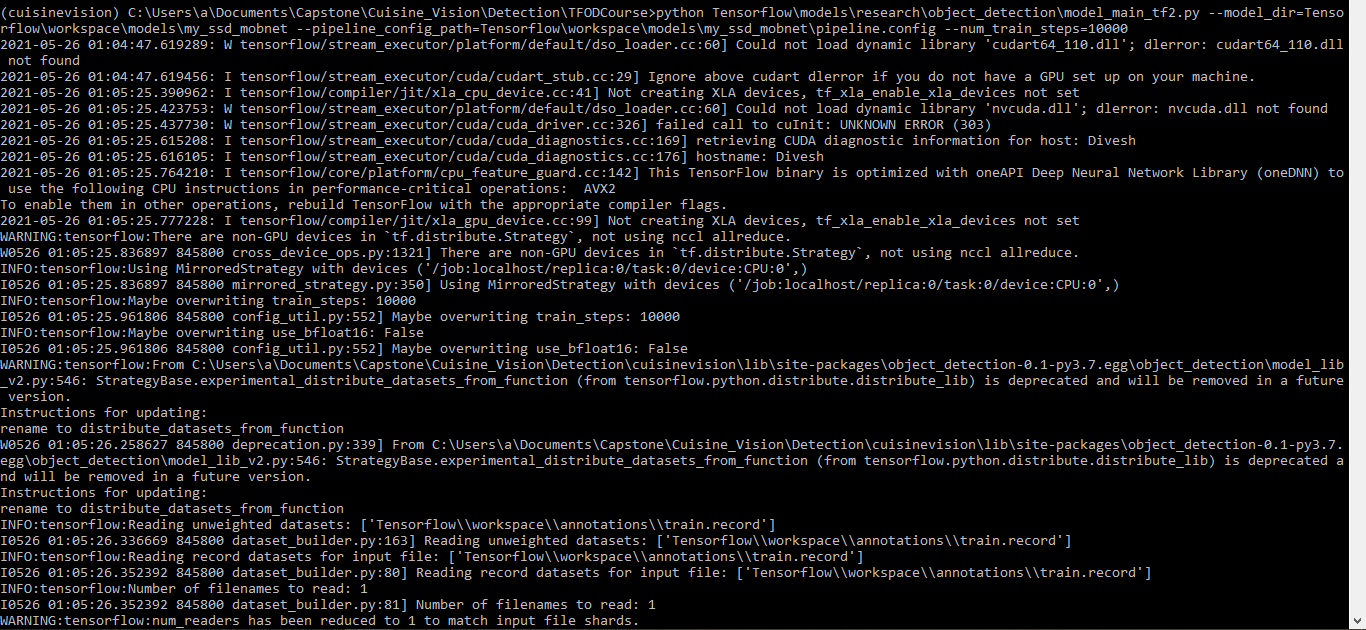
Finally, after following the configuration steps and modifying the pipeline.config, the command below is executed to start our custom model training:



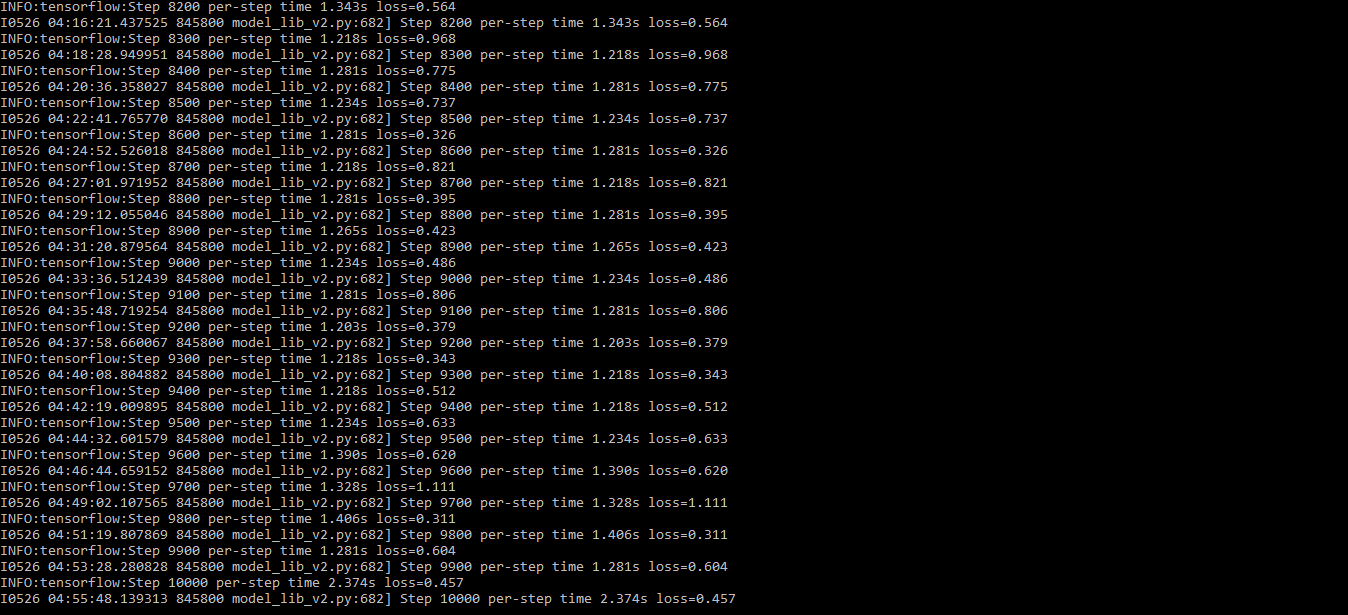
* **The component:** python Tensorflow\models\research\object\_detection\model\_main\_tf2.py
  + It is going into the directory (Tensorflow\models\research\object\_detection\model) and it is running the script **model\_main\_tf2.py**
* **First argument:** model\_dir=Tensorflow\workspace\models\my\_ssd\_mobnet
  + It is passing through the customized model directory (Tensorflow\workspace\models\my\_ssd\_mobnet). This is where the customized pipeline.config file is.
* **Second argument:** pipeline\_config\_path=Tensorflow\workspace\models\my\_ssd\_mobnet\pipeline.config
* It is passing through the customized pipeline.config file
* **Third argument:** num\_train\_steps=10000

This is the number of training steps that is going to be run through. Higher number of steps improve the performance and hence, increase the accuracy of the model.

* Beginning of training:



* Model training has successfully completed:



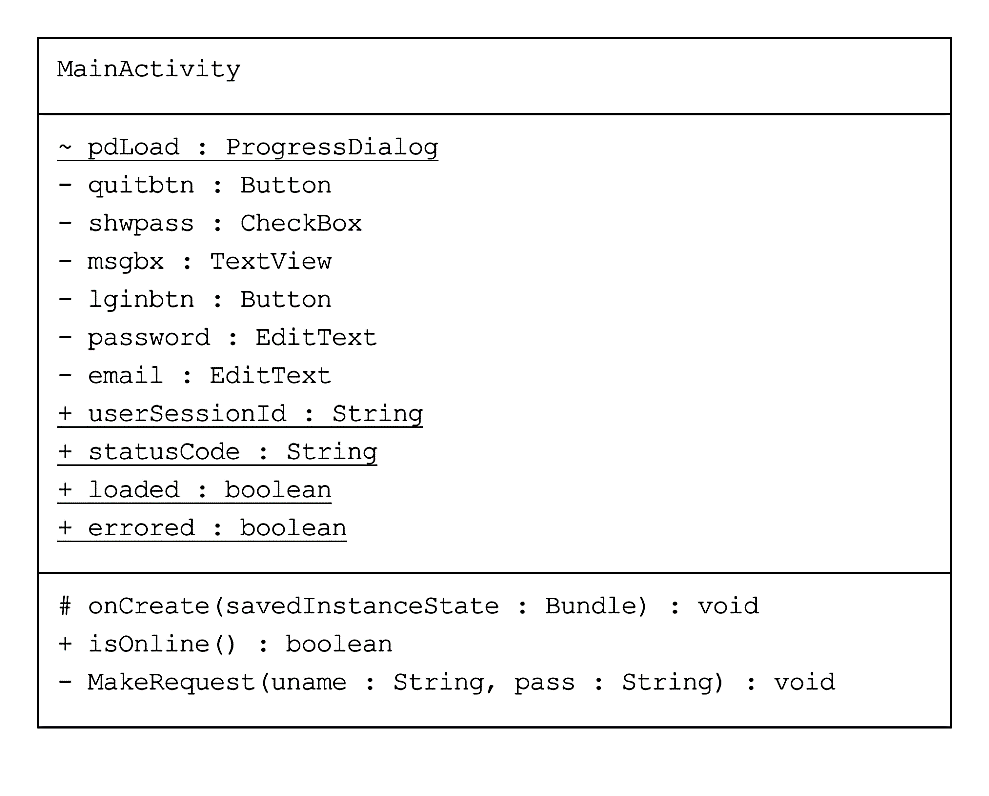
## Android App

The Android App makes use of a combination of activities and fragments to make user navigation easier.

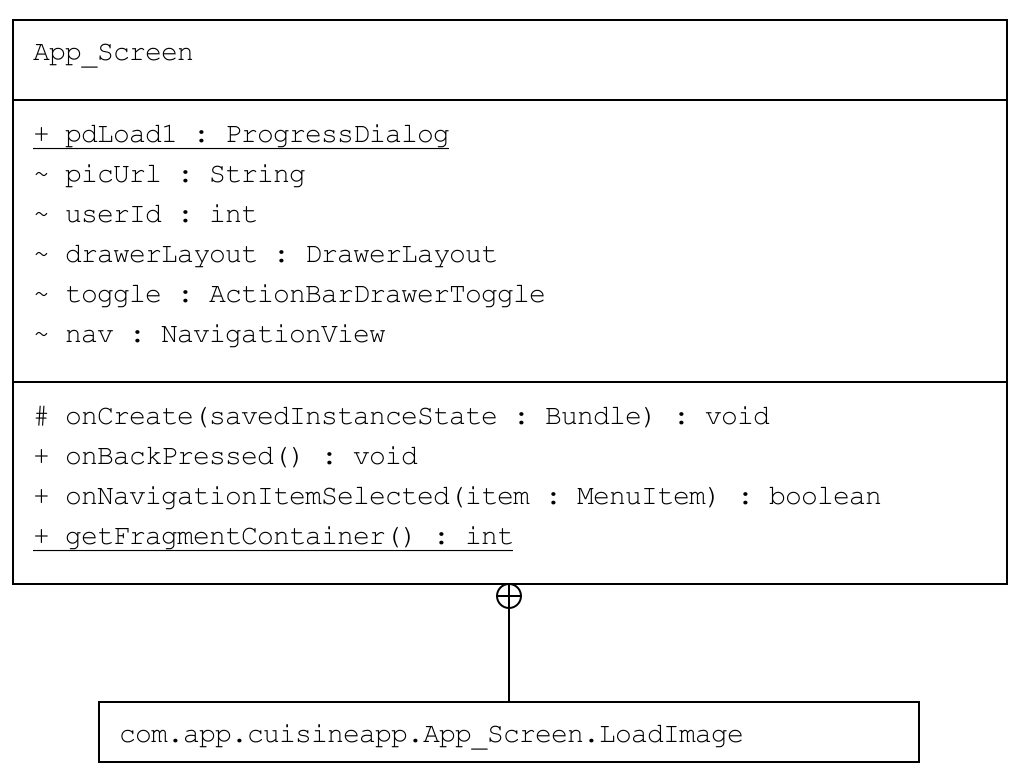
### Class Diagrams of Activities

* **MainActivity Activity**

This activity handles the login page of the app. It makes php requests to the Server using Google’s Volley library. Volley is a standard HTTP library that makes networking for android apps easier and faster.

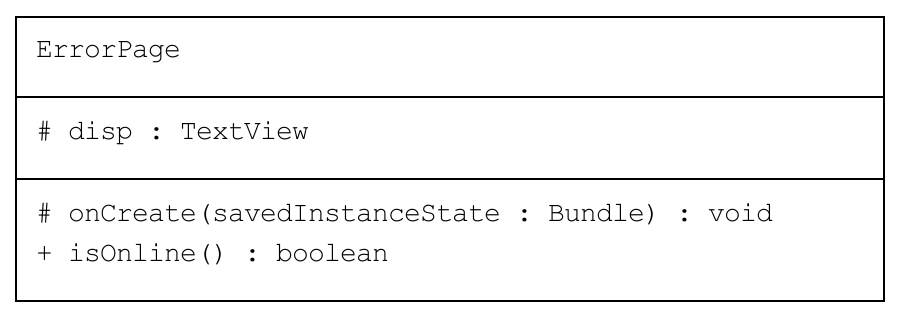


* **App\_Screen Activity**

This Activity extends the navigation view which is a left slider menu. It also handles the fragment views by swapping the fragment container with the selected item layout.

* **Error Page Activity**

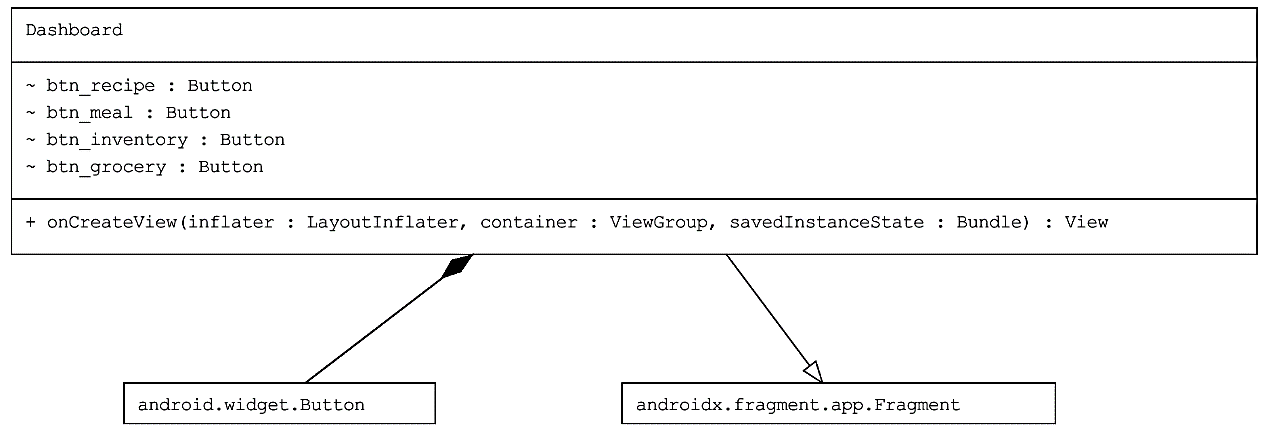
This activity is shown when the app faces a network issue or loss of internet connection. It has a timer that checks for internet connection every 5 seconds. If stable internet connection is found, it redirects the user to the page he was on



### Class Diagrams of Fragments

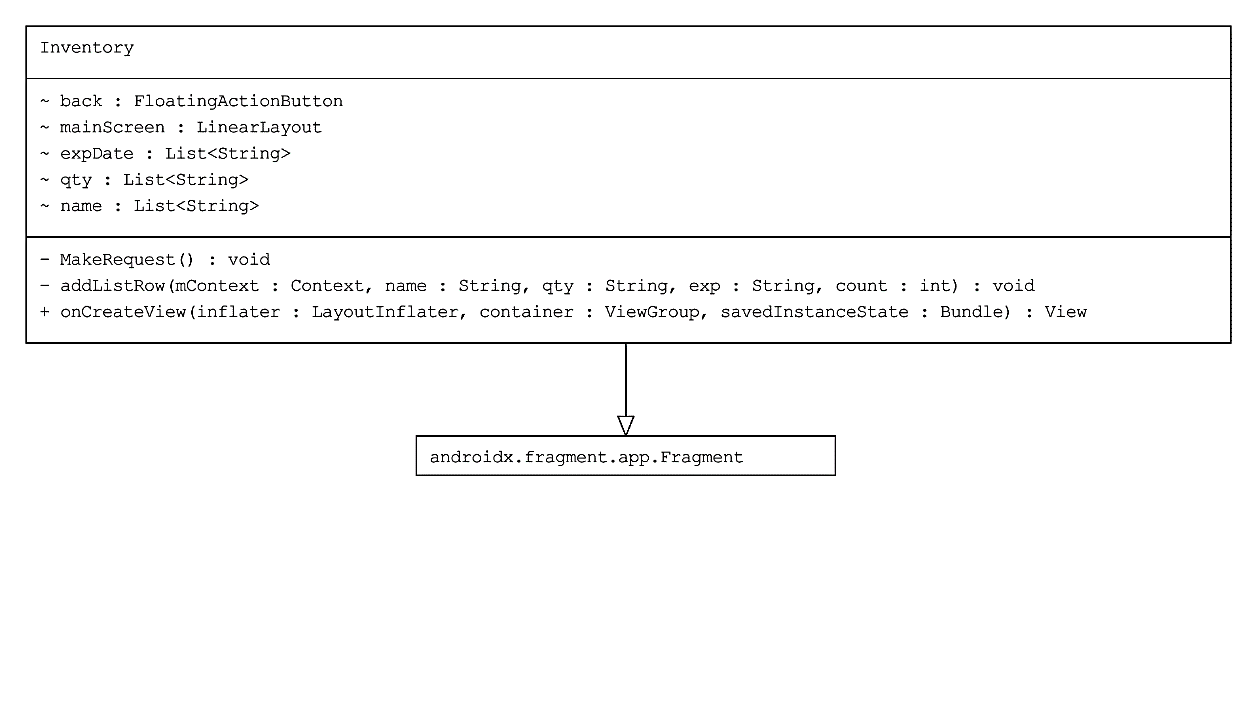
* Dashboard

This is the primary page after a user login. It contains useful tips and information for the user and well as quick links for the user to navigate through the app.



* Inventory

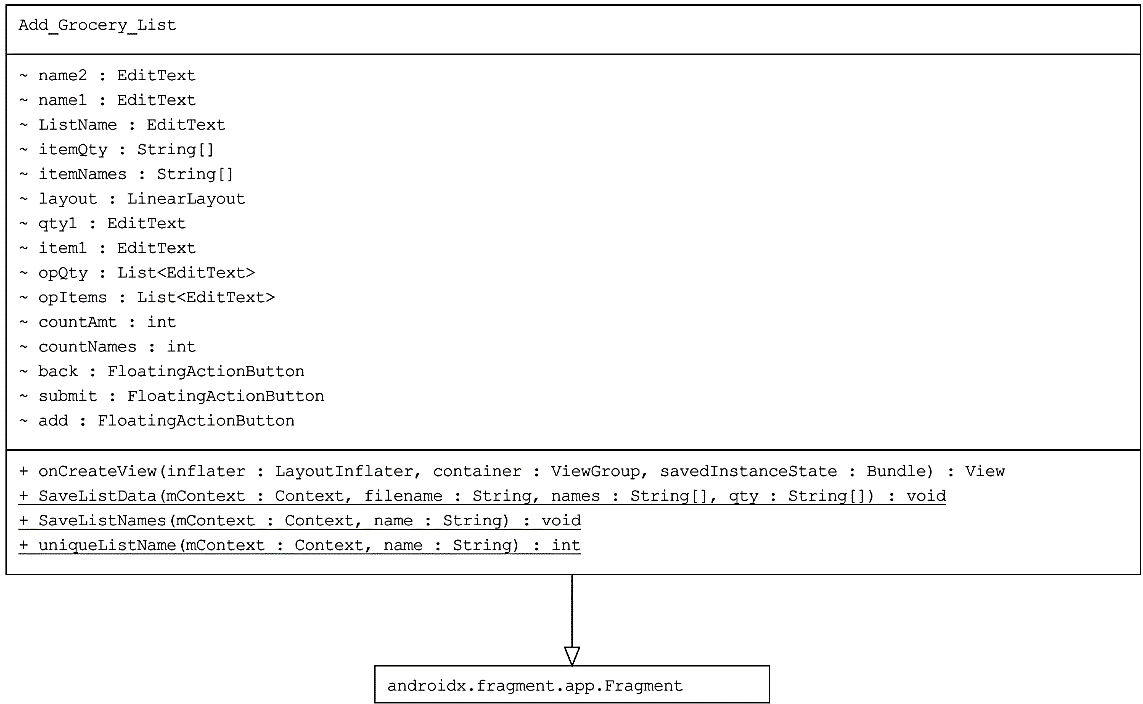
This fragment handles the inventory part of the user. It allows the user to manage his food inventory, that is, to add, delete or edit food items. The inventory items are stored online in MySQL database.



* Grocery\_List

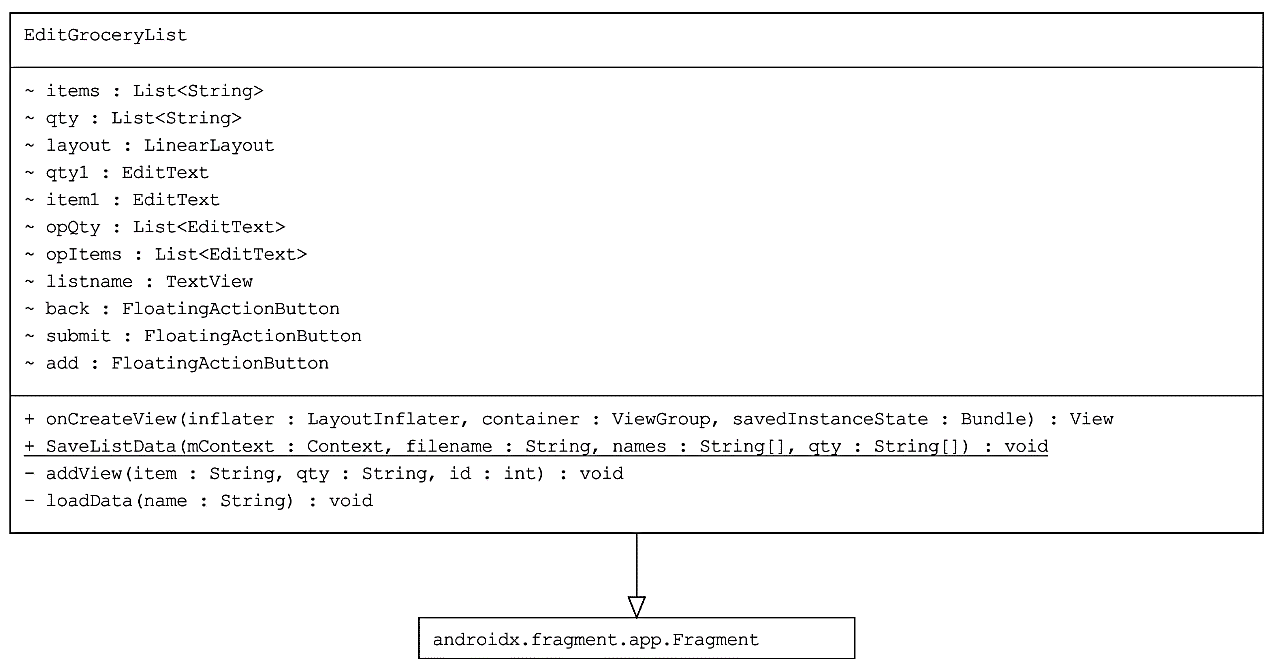
This fragment handles the grocery list for the user. It displays existing grocery lists and also allows the user to create a new grocery list.

* Add\_Grocery\_List

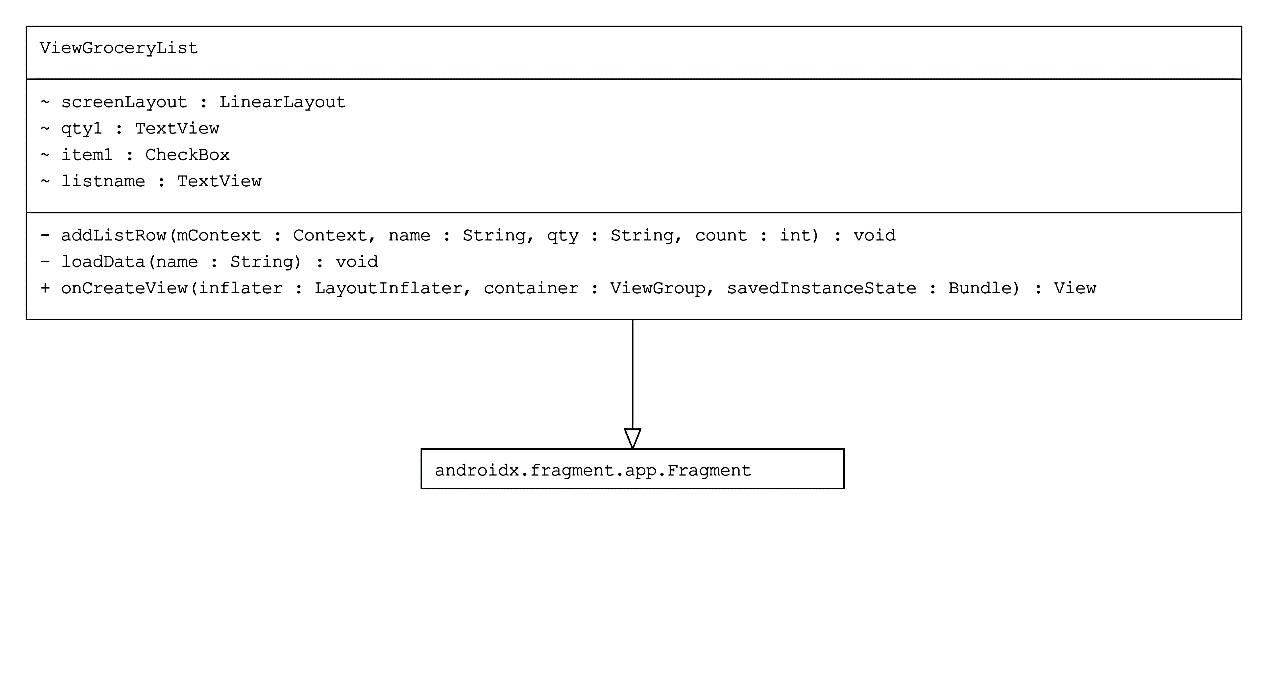
This is a sub fragment which provides an interface for the user to create a new grocery list.

* EditGroceryList

This sub-fragment provides an interface for the user to edit a grocery list.



* View\_Grocery\_List



* Meal\_Planning

This activity will allow the user to plan their meal for the week.

|  |
| --- |
| Meal\_Planner |
| - meal\_day: mealplan  - meal\_name |
| -load\_data : void  -create\_plan : void  -add\_meal : void  -editPlan : void |

* Recipe\_Suggestion

This activity will suggest a recipe to the user. It will make a call to an online API.

## Cuisine Vision

Raspberry Pi is basically a single-board computer. It has all the same components of a desktop PC but in a much smaller factor. Raspberry Pi 4 is an upgrade of the previous generation Pi adding onboard dual monitor support and the option to have up to 8 gigabytes of RAM. It is a practical and useful tool for prototyping of our project, Cuisine Vision.

To implement the features of Cuisine Vision and eventually realize our Cuisine 2.0 project, a Raspberry Pi 4 along with a touchscreen display enclosed in the SmartCase Pi Touch 2 came into handy. One of the advantages of Cuisine Vision is to be able to attach and detach it to and from the refrigerator door. This will enable the ability to displace Cuisine Vision anytime and anywhere the user wants to with little effort. The Raspberry Pi and the touchscreen display tick all the requirements to successfully implement Cuisine Vision and shape Cuisine 2.0 into a reality.

### All equipment used for Cuisine Vision



Raspberry Pi Camera

SmartiPi Touch 2 Enclosure

Raspberry Pi 4 Model B

Jumper Wires

Adapter Board

USB-C Power Connector

Raspberry Pi 7” Touchscreen Display

DSI (Display Serial Interface) Ribbon cable

### Raspberry Pi 4 Model B Overview

2-Lane MIPI DSI Display Port

Micro SD Card Slot

* Bluetooth 5.0
* 802.11b/g/n/ac wireless
* 2.4/5 GHz Wi-Fi

64-bit quad-core Cortex-A72 processor

2GB LPDDR4 RAM

PoE (Power over Ethernet) Header pins

Gigabit Ethernet port

2 USB 3.0 ports

2 USB 2.0 ports

Audio & Video Composite 3.5mm jack

2-Lane MIPI CSI Camera port

2 micro HDMI ports

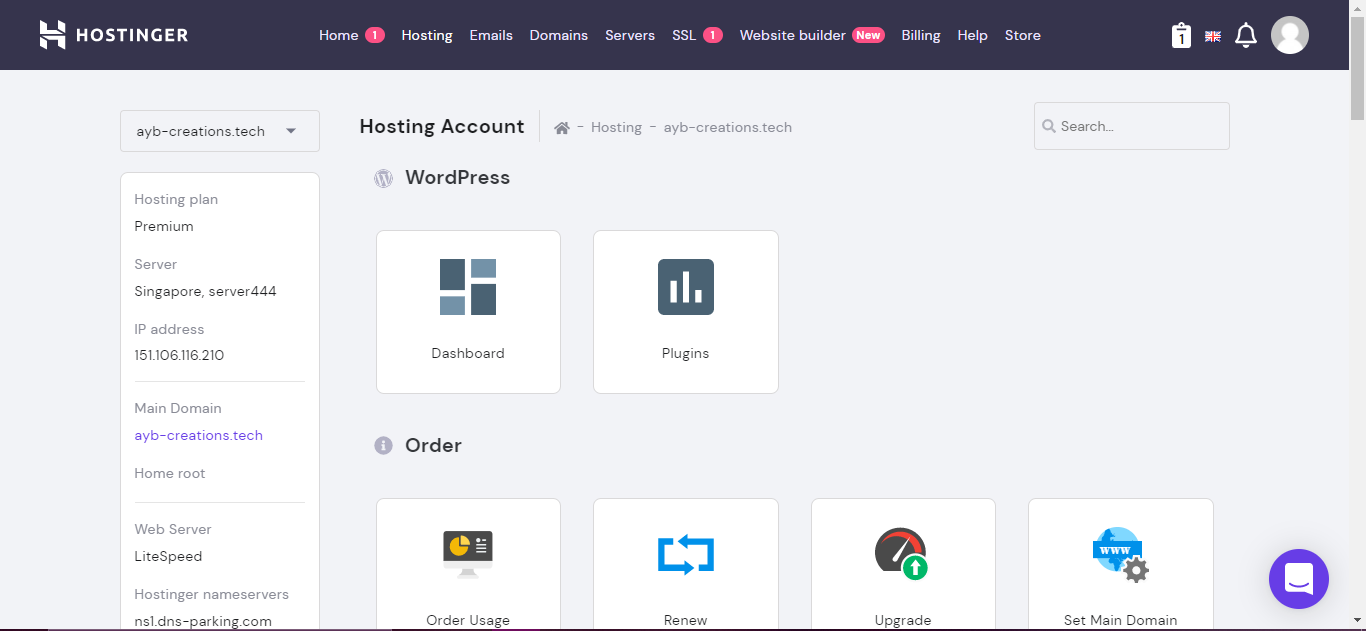
USB-C power port 5V/3A



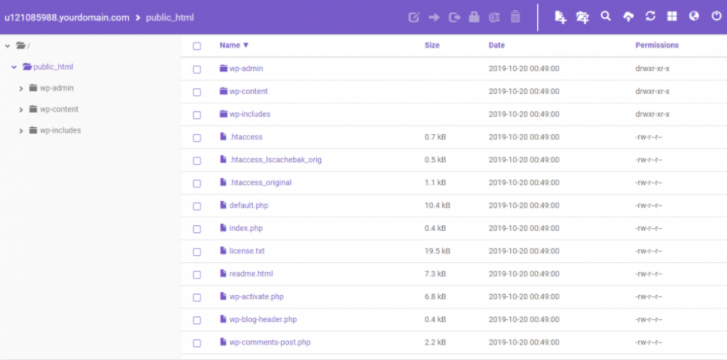
# Server

The server we used to host our website and PHP APIs is a premium Linux shared hosting virtual server. The hosting company is Hostinger. The server is optimized for WordPress website and provides high performance with SSD storage, Litespeed caching, and GZIP compression (form of file compression used to increase speed).

* HPanel, which is the control panel of the server is very user-friendly.



* It also provides an online file manager to view, edit and delete scripts and remote ftp access.



## Website

The link to our website is: <https://cuisine.ayb-creations.tech>. The website of Cuisine 2.0 has been designed and implemented using WordPress. WordPress is a free and open-source content management system written in PHP and MySQL or MariaDB Database

The theme used to construct the website is Astra. Astra is one of the most lightweight and reliable website themes providing full customization. We chose Elementor as our website builder tool as it allows many plugins to integrate with itself and has all our required tools.

Plugins used for the website:

|  |  |
| --- | --- |
| **Plugin Name** | **Description** |
| Elementor - Header, Footer & Blocks | Used to create custom header and footer for website |
| Essential Addons for Elementor | Provides additional widgets for use |
| Fluent Forms | Provides Contact for interface |
| FluentSMTP | SMTP service for the fluent form which sends the administrator an email after someone has contacted them. |
| HT Mega - Absolute Addons for Elementor Page Builder | Elementor Addons Package providing additional widgets. |
| LiteSpeed Cache | Cache Webserver which makes loading of the resources faster |
| Premium Addons for Elementor | Elementor Addons Package with provides additional widgets for different functions |
| Simple Custom CSS and JS | Allows us to post our custom CSS and JavaScript codes |
| Wordfence Security | It is an anti-virus, firewall and malware scanner. It also protects the website from login bots. |
| WP Super Cache | A caching service to load high-definition images really fast. |

***Note:*** *All plugins are open-source plugins and are free.*

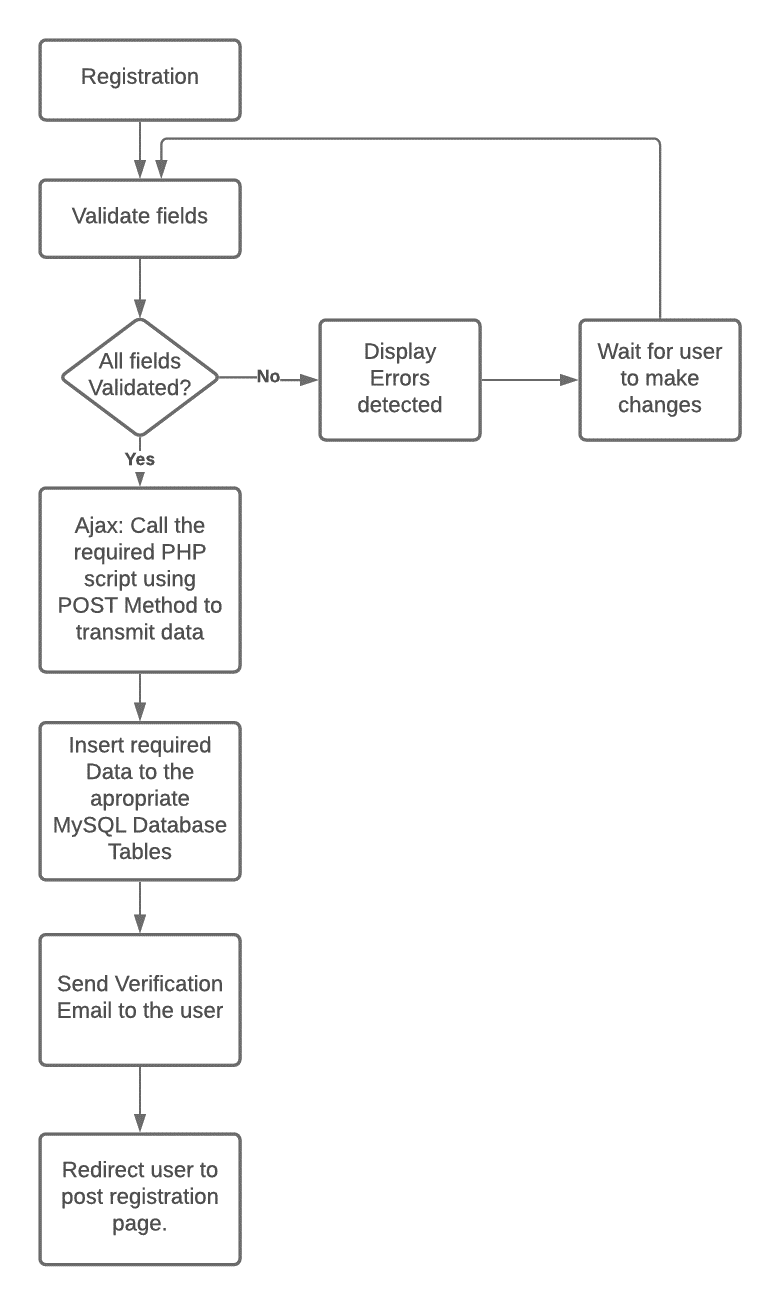
Our Website follows a different approach using a vertical slider. The website has a single page which slides multiple templates on scroll.

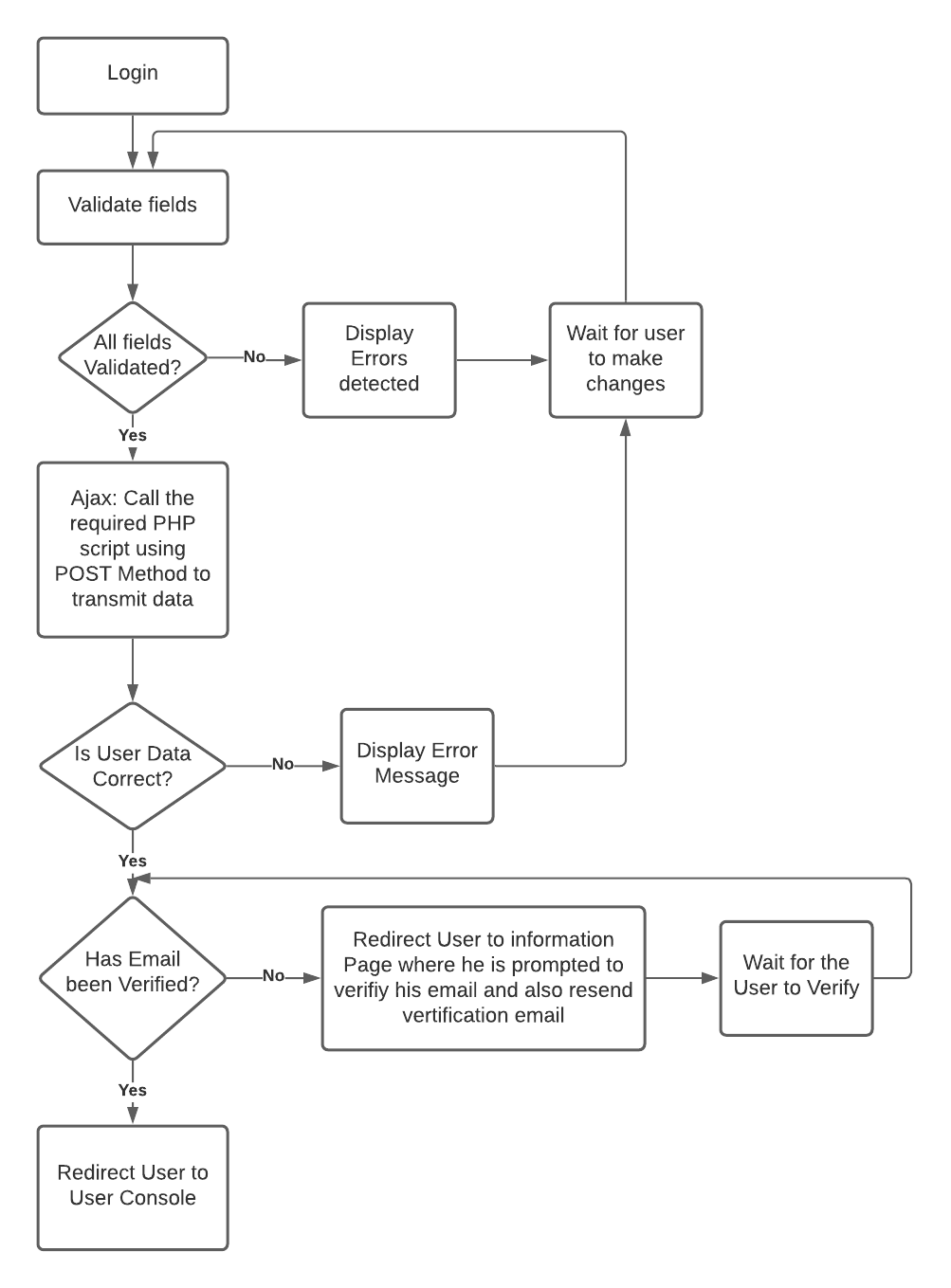
The Homepage has options for the user to login or create a new user account. All the user data are stored and queried from a MySQL database via our custom written PHP APIs

The Login and Registration are displayed as a JavaScript overlay window which is written in HTML, CSS and JavaScript. When the user submits the form, the form connects to the PHP script using JavaScript Ajax.

We used Ajax to make calls to the PHP codes because:

1. WordPress doesn’t allow custom PHP to be included in the codes because WordPress is itself made of PHP which will create conflicts between internal codes and plugins.
2. Ajax makes asynchronous calls to be webserver which improves performance and speed.

**Flowchart demonstrating the user registration process:**

**Flowchart demonstrating user login:**

# User Interface

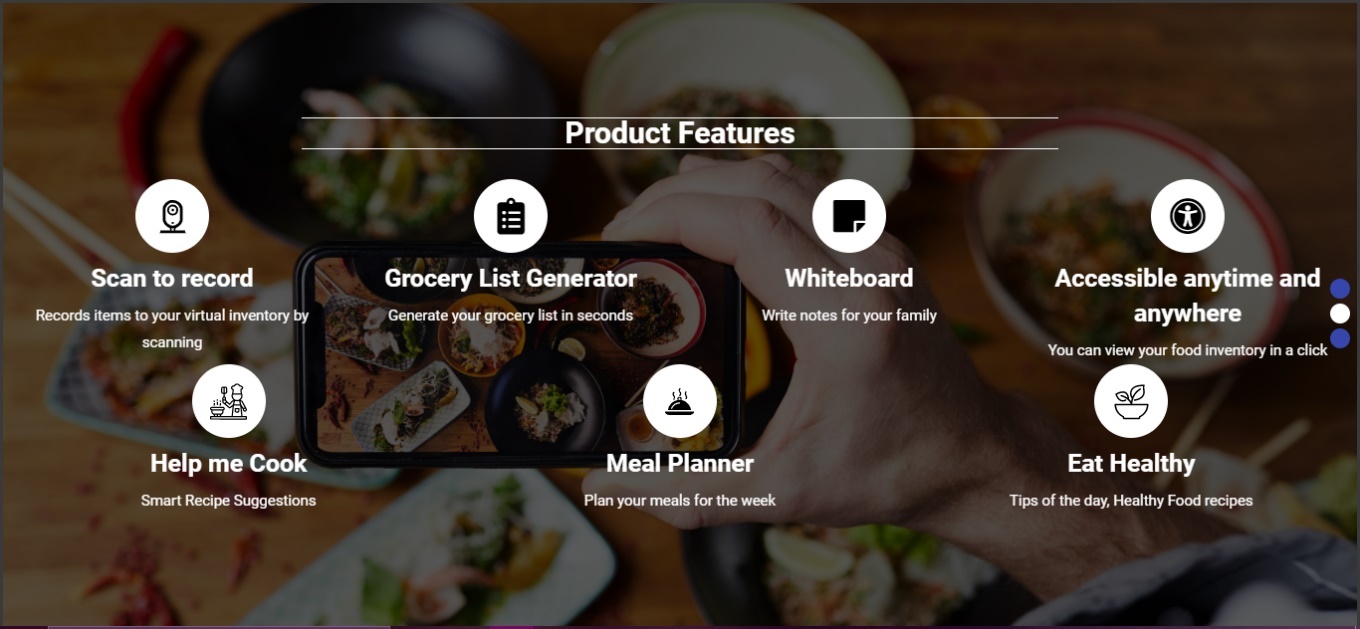
## Android App

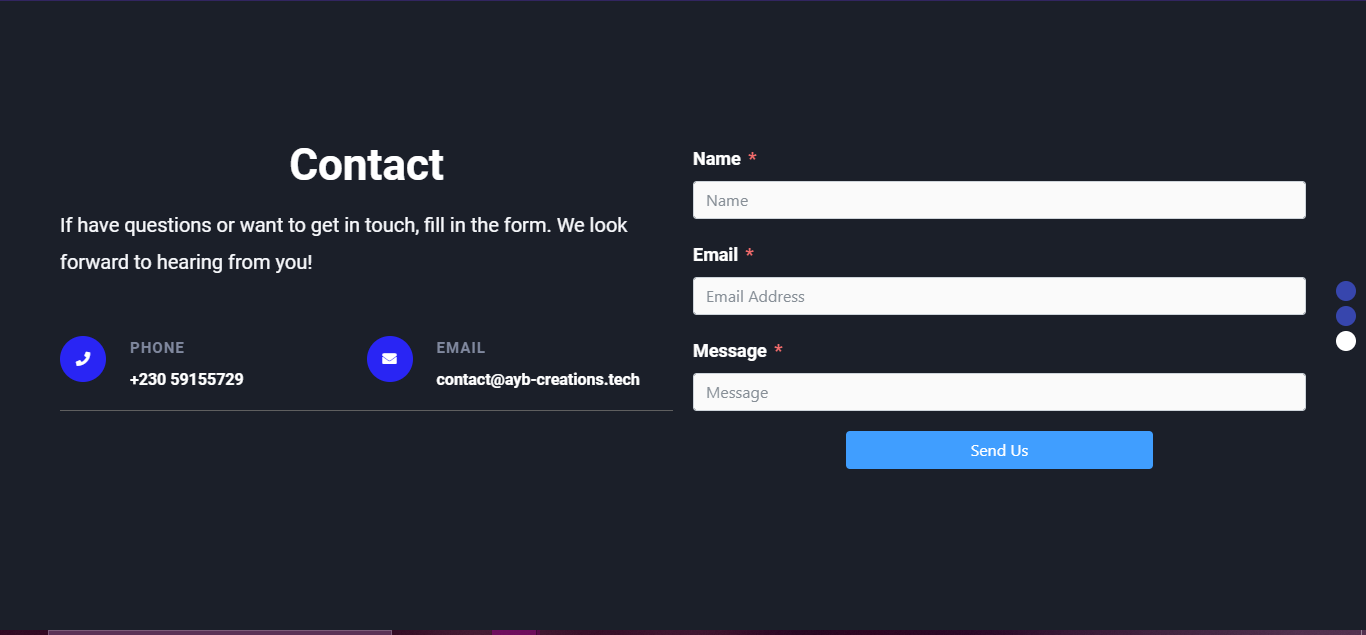
|  |  |  |
| --- | --- | --- |
| Login Page | Navigation Left Slider | Error Page |
|  |  |  |

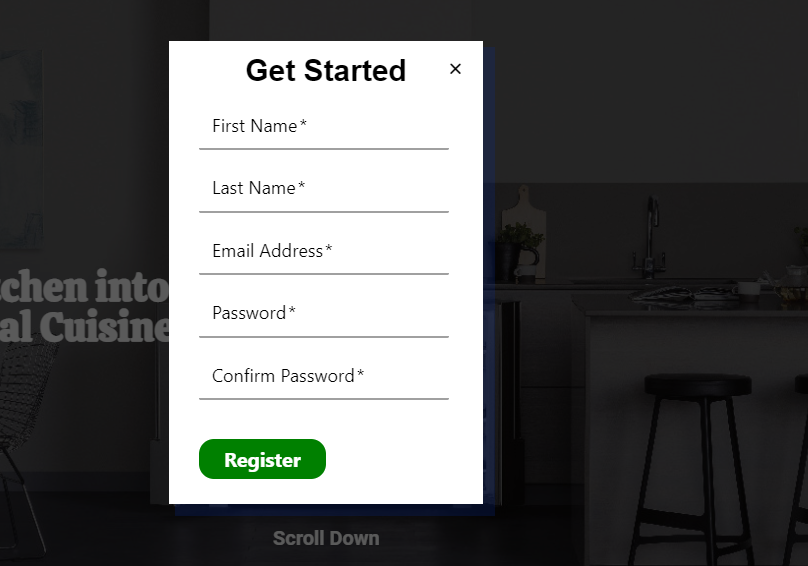
|  |  |  |
| --- | --- | --- |
| Dashboard Fragment | Grocery List Fragment | Adding new Grocery List |
|  |  |  |
| Editing Grocery List | Viewing Grocery List | Inventory Fragment |
|  |  |  |

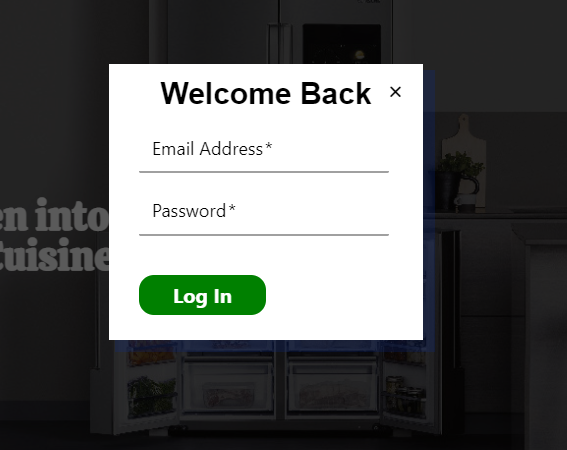
## Website

Main Pages:









Error and Verification Pages:

