



# MustEatNow

## Smart Management of Household Perishables Inventory

### Members:

Goh Kum Whye Leslie	A0229982M
Lim Jingcheng, Konchok	A0229969A
Low Ruo Qi	A0229977E
Low Yim Tong	A0229983L

---

Project Report for the Intelligent Reasoning Systems course at Institute of Systems Science,  
National University of Singapore

# Table of Contents

---

EXECUTIVE SUMMARY	4
BUSINESS PROBLEM / BACKGROUND	4
MARKET RESEARCH	6
PROJECT OBJECTIVES	11
PROJECT SOLUTION (DESIGN AND IMPLEMENTATION)	11
Overall System Design Architecture	11
Database Management System	14
Smart OCR System	17
Image Reader Subsystem	17
Text Parser Subsystem Description	18
Text Parser Rule-Based Inference Model	19
Text Parser Observation State Estimation	23
Data Representation	24
User Interface Integration	25
Grocery Stock Level	27
Smart Purchase Recommender System	30
PROJECT PERFORMANCE AND VALIDATION	32
Semi-automated record logging	32

Smart curated grocery list	33
Consumption reminders	34
FINDINGS AND RECOMMENDATIONS	35
Improving model accuracy with automated diagnostics	35
Improving model generalization	35
Improving user input flexibility	36
CONCLUSIONS	36

## EXECUTIVE SUMMARY

Food wastage has been an increasingly serious and long persisting global problem. While entities and corporations have put in place measures to tackle this issue from a supply chain perspective, consumers can also do their part in reducing their domestic food wastage. This project aims to implement a system with smart features to support consumers in making smarter decisions when managing their household perishables inventory.

The project has targeted the problem from three key focus areas: (1) fast and convenient inventory recording, (2) timely consumption reminders, and (3) smart purchase recommendations to encourage mindful purchasing. In order to achieve these objectives, the project maintains a knowledge base which contains the historical data from the user, and a knowledge base of food grocery items. These knowledge bases were used by the different reasoning models, and the frontend interface system to make logical deductions and recommendations to the user.

The project has implemented a smart OCR system which takes in receipt image input to generate purchase records for convenient inventory logging. We have also implemented a purchase recommendation system based on projected inventory levels and past consumption data. Lastly, there is also a consumption reminder system to remind consumers of food expiry dates. All the above features are integrated and presented to the user through an intuitive and interactive web application. With the above, the implemented system functionalities were able to achieve the targeted project objectives.

## BUSINESS PROBLEM / BACKGROUND

Domestic food waste is now a large contributor to the global food-waste problem. In Singapore, the amount of food waste generated has grown by around 20% over the last 10 years<sup>1</sup>. In 2019, Singapore generated around 744 million kg of food waste - equivalent to 2 bowls of rice per person per day, or around 51,000 double decker buses. Each household in Singapore generates an average of 1.5kg of waste per day, with food waste comprising half of that. Within that, it is

---

<sup>1</sup>National Environmental Agency, <https://www.nea.gov.sg/our-services/waste-management/3r-programmes-and-resources/food-waste-management>

estimated that more than half of household food waste can be prevented - “avoidable” food waste being items that could have been consumed if better managed, including leftovers, expired food, stale food, blemished fruits and vegetables.

Grocery retail and foodservice industries, which have been traditionally blamed for large quantities of unsold food being wasted, have made headway to reduce, recycle and/or redistribute food waste in the supply chain. This includes price mark-downs and discounts on perishable products at later day-parts, on-site food waste treatment systems and long-term partnerships with charities to donate unsold products<sup>2</sup>. This is in addition to robust procurement inventory management systems, which businesses utilise to prevent over-ordering from suppliers.

However, the problem of food wasted at home remains persistent. We explored the potential overlaps in the solutions used at a business that can be economically applied at the level of a multi-consumer household. According to a 2014 survey by the National Environmental Agency, the key driver for consumer food wastage was food going out of date (**Figure 1**). This stemmed from a need to ensure that household members had more than enough to eat, resulting in over-purchasing and leftovers from home prepared meals. We identified three gap areas to address, namely (i) recipe and meal planners to help consumers more accurately estimate and match grocery purchases with intended meals, (ii) inventory tracking to ensure consumers can monitor what is in their fridge and purchase only what is needed at any point in time, and (iii) adopting smart food storage and preparation practices and products.

---

<sup>2</sup> The Straits Times, <https://www.straitstimes.com/singapore/how-supermarkets-fight-food-waste-in-singapore>



Figure 1: NEA consumer survey on food wastage

Source: <https://www.towardszerowaste.gov.sg/resources/infographics/food-waste>

## MARKET RESEARCH

Based on the gaps identified, we focused our market research on how intelligent systems have been applied in the market and how to design a product that best addresses the needs of our

target focus group of multi-consumer households.

Our researched revealed three main groups of solutions currently in the market - (i) applications for fridge and pantry management (**Table 1**), (ii) applications providing shopping list management and grocery price comparison (**Table 2**), and (iii) applications and devices focused on helping reducing food waste through redistribution and monitoring (**Table 3**). The third group of solutions were primarily focused on F&B and hospitality businesses - except for OLIO, which functions as an online food bank, allowing individuals (in addition to companies) to list surplus food items and for others to procure.

### Addressing the gaps

Narrowing our research on the first two groups of solutions, we found that *recipe and meal planning* was covered by applications such as Epicurious, where a large database of recipes and efficient search function allows it to reach out to consumers across different countries. A growth lever would be to grow the database and tailor recipes for local markets and synchronise naming conventions for items entered into the application. We found that the *adoption of smart food storage and preparation practices* were best addressed through consumer education and awareness (i.e. FoodKeeper in the US), and by appliances (e.g. smart fridge, dustbins) and houseware (e.g. storage containers, seals, tags). In order to have a product that could penetrate the consumer market and scale up efficiently, and be updated continuously as it is being rolled out, we focused our project on web and/or mobile based solutions. User feedback gathered from our research on existing web and/or mobile applications were (i) the ability for users to import/export .csv files to/from an application was important for customers who had existing data which they wanted to bring across when switching applications, and (ii) easy-to-use and an accessible user interface was key to user adoption.

We concluded there was a good addressable opportunity in *inventory tracking*, as the applications available locally (**Table 2**) focused on shared shopping lists, cost aggregation and price comparisons. Data input to these applications were manual or via barcode scanners (which has a mixed user feedback). We positioned our product to be differentiated in data input (via receipt digitisation), providing expiry and stock-out reminders with effective visual representations.

Fridge Pal (**Table 1**) had a similar value proposition but was not available for download in this region. Our product would also generate a prioritised shopping list based on inventory stock levels and projected consumption rates, serving more as a smart recommender rather than simply a digital replacement of a pen/paper grocery list.

Table 1: Applications for Fridge and pantry management




Application	Description
<b>Fridge Pal</b> 	Utilises a barcode scanner to quickly add items to a user's list. Provides reminders nearing food expiry, shares shopping lists with other family members, recipe search and meal calendar to support planning ahead. Not available for download in Singapore.
<b>FoodKeeper</b> 	Users can easily search through a wide database for storage times of food items to ensure consumption at peak quality and reduction of waste. The times will vary depending on the growing conditions, harvesting techniques, manufacturing processes, transportation and distribution conditions, nature of the food, and storage temperatures. Developed by the USDA's Food Safety and Inspection Service.
<b>Epicurious</b> 	Database of over 30,000 recipes compiled from Bon Appetit, Gourmet, and top chefs from across the globe. By entering an item into the app (these can be items close to going bad or what is available at that point in time), users can access a list of recipes which can utilise the available item and step-by-step instructions for each recipe.



Table 2: Applications providing Shopping list management and price comparison


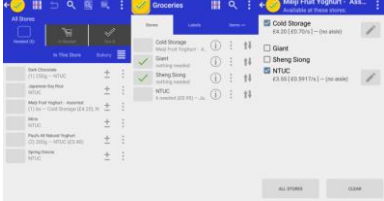
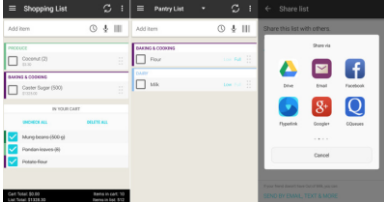





<p><b>Bring! Shopping List</b></p> 	<p>Shared shopping list which syncs instantly, so members within a group get real-time perspectives of what is needed and what has been bought. Added items show up immediately, and a "List updated" button lets the shopper know that the list has been finalised. Claims to have reached 10 million households and expanding into mobile advertising for FMCG and retail partners.</p>
<p><b>rShopping List</b></p> 	<p>Input lists, stores, items, and labels. Location search (powered by Google Maps) notifies the user each time they are near a store and have a pending item on the grocery list. Last price entered at each store is remembered and a warning is sent if the user is purchasing an item that was less expensive at another store; predicts cost of shopping basket over time. Mixed reviews on barcode scanning function.</p>
<p><b>Out of Milk</b></p> 	<p>Items (price, quantity) can be added to a Shopping List by typing or speaking to it (voice recognition) or based on what was bought before. A separate Pantry List lets users enter items and indicate Low or Full, to prioritise essentials (Shopping List) from the good-to-have's (Pantry List), depending on whether the user had a car, etc.</p>
<p><b>Price Kaki</b></p> 	<p>Crowdsourcing mobile application allowing consumers to compare prices of in-store groceries, household items, and hawker food through a single platform.</p>

Table 3: Applications and devices focused on reducing food waste through redistribution and monitoring in Singapore

<p><b>Treatsure</b></p> 	<p>The app connects customers with hotel buffet restaurants and F&amp;B merchants with excess food. Users scan in a QR code at the restaurant of their choice to purchase leftovers (typically from S\$10+), payment is made directly to the establishment for a hassle-free process. Users can also order ‘ugly’ produce and premium dairy products from surplus grocery suppliers via the app.</p>
<p><b>OLIO</b></p> 	<p>OLIO allows users to put up and peruse listings of food and non-food items (bags, clothes, etc.) before procuring them for no additional fee. The British app was founded in 2015 as a platform to share surplus goods after realising that more than half of all food waste in Western countries takes place at home. The app now sees more than a million users worldwide and over two thousand in Singapore.</p>
<p><b>Makan Rescue</b></p> 	<p>Makan Rescue notifies users about F&amp;B businesses in their vicinity that offer surplus food for free. The startup’s users currently cover staff and students from NUS, SMU, and NTU as that is where the leftovers are from and is planning to expand in the future.</p>
<p><b>Lumitics</b></p> 	<p>Food-waste tracker developed in Singapore which uses sensors and proprietary image recognition technology to weigh and identify what restaurants and kitchens throw away. Launched in 2019, the device integrates with standard rubbish bins used in kitchens, generating data to help hospitality and F&amp;B businesses reduce their food waste and reduce costs.</p>

## PROJECT OBJECTIVES

In order to tackle the problem of domestic food wastage, the project has chosen three focus areas to support and encourage consumer action. These three areas are namely in terms of inventory recording, food consumption and grocery purchasing. The project aims to implement an effective home perishables inventory management system to achieve the following key objectives and success outcomes:

1. Provide a simple and convenient method for inventory update. The project aims to implement a smart OCR system to reduce the time and effort taken to log the records into the system.
2. Reduce food wastage with timely reminders. The project aims to reduce the amount of food being disposed of, by making expiry reminders with effective visual representations.
3. Simplify the users' decision-making process during grocery shopping. The project aims to encourage mindful purchase habits, and simplify grocery shopping with smart recommendations for the users' shopping list based on inventory stock levels and projected consumption rates.

## PROJECT SOLUTION (DESIGN AND IMPLEMENTATION)

### Overall System Design Architecture

To achieve the key project objectives, the project was broken down into the following sub-systems and model for implementation.

1. Frontend user interface for overall system integration, data visualisation, and to receive user input for updating the knowledge base in support of continuous model training.
2. Smart OCR system to parse the receipt images as input data and to extract key information on items and quantities purchased for fast and convenient data entry into the inventory logs.

3. Smart purchase recommendations system to recommend a curated grocery list based on past consumption patterns.
4. Database system to manage the knowledge base of historical grocery items data and user records data.

The overall system architecture diagram is shown in **figure 2a** below. In this system architecture implementation, the frontend user interface system acts as the core interface and integration mechanism between the end user and the various subsystems within the system (see **figure 2b** for web application design). It has been designed to be intuitive and simple to use, in order to provide a seamless user experience (see **figure 3** for web application login details). For the database, it was maintained in an SQL database, storing the knowledge base for the reasoning models. The user interface would receive user feedback on the output of the models and this feedback would be sent back to update the database with the new information, providing a closed-loop training feedback to the knowledge base.

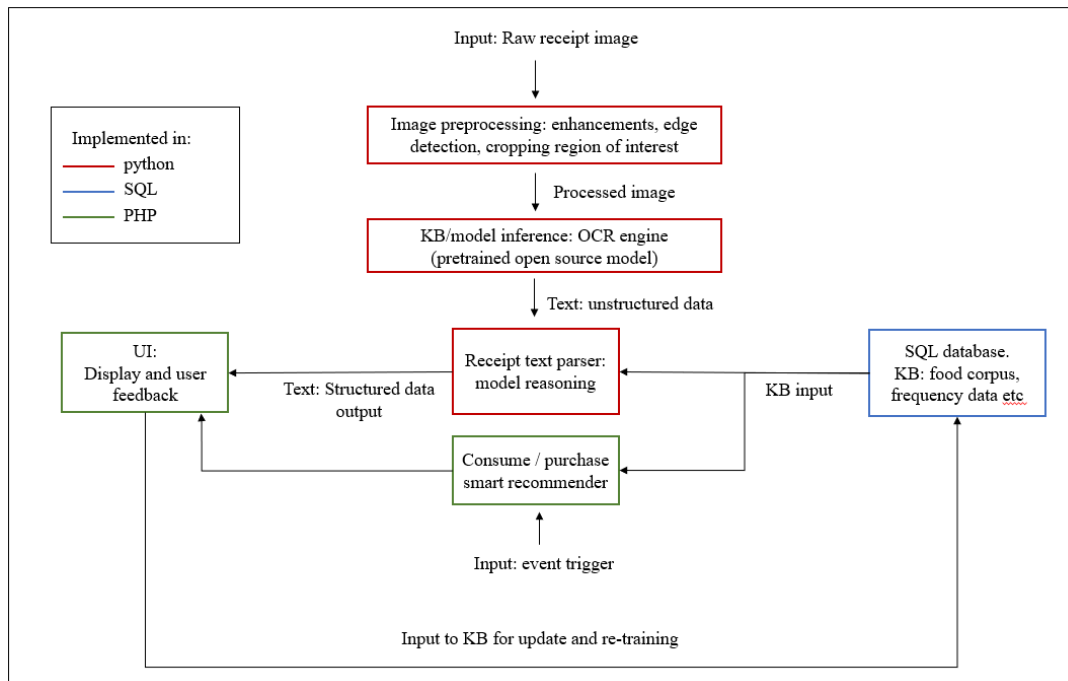


Figure 2a: Overall System Architecture

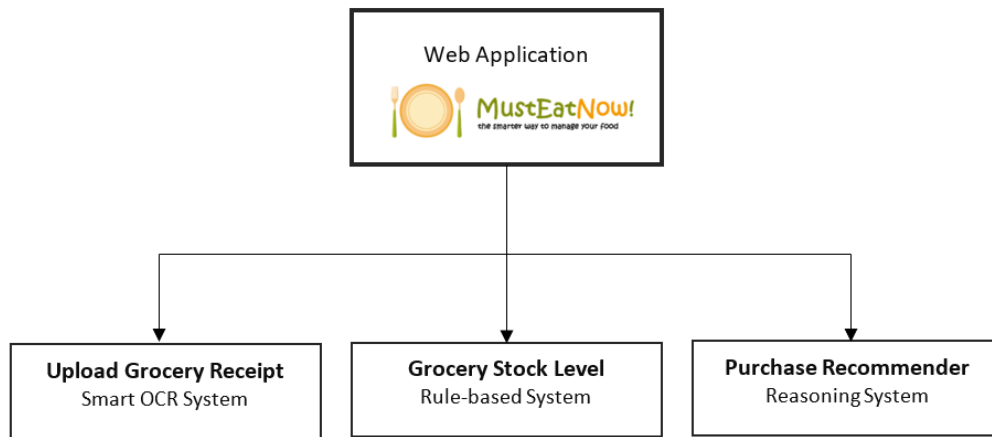
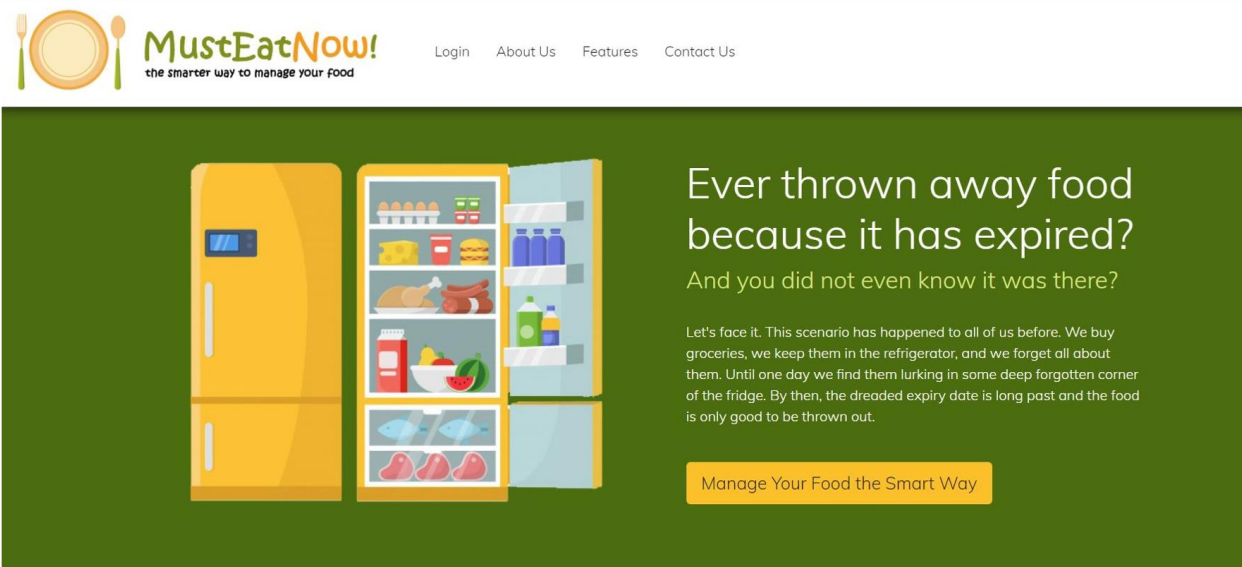


Figure 2b: Overall Web Application Design

[www.musteatnow.com](http://www.musteatnow.com)



**Login:** [my\\_family@example.com](mailto:my_family@example.com)

**Password:** irs2021

Figure 3: Web Application Main Page and Login Details

## **Database Management System**

### **Database application selection**

The application required a form of persistent storage which would support multiple users and allow the knowledge base to grow over time. Due to the transactional nature of the system, the storage should also be ACID compliant (Atomicity, Consistency, Isolation, Durability), and offers low-latency response (<3 seconds) to application requests. As we are in the prototyping phase of the solution, horizontal storage scalability is a bonus, but not integral at this stage. Availability of the knowledge required to further enhance and maintain the storage is also a consideration, as a scarcity of this knowledge in the talent market would mean higher running costs for supporting this solution in future.

As such, we chose MySQL Server as it satisfies these requirements. We also considered Hadoop Distributed File System (HDFS). HDFS is scalable, and with Apache Spark to compute the data, may be able to achieve a latency response which users can accept, however it lacks ACID compliance by default, which makes it unsuitable to support a transactional system.

### **Knowledge base**

The existing knowledge base in the system consists of a corpus of food/grocery items, past purchases of each user, and frequency of consuming food & grocery items. The initial acquisition of this knowledge base was conducted manually by looking through various past receipts and inserting common grocery items with their expected expiry dates into the database.

During this knowledge acquisition process, the project had considered to deploy existing word corpus such as the “food” synsets from WordNet. However, the categorical segregation in WordNet was insufficient for us to effectively and efficiently populate sufficiently accurate expected expiry dates for all items in the WordNet corpus. In addition, the target focus of the project was on perishable items only and the WordNet would be overly extensive for our current use case. As such, the project has instead chosen to implement a training feedback loop to overcome the shortfalls of our relatively limited knowledge base of food corpus.

## **Design**

The system was designed to offload most of the data computations in the database, where the data resides, and returns the application specifically what it needs. As such, the use of logical views(virtual tables) were used, where data transformation knowledge is stored as code, and processed on-the-fly with the current tables on every request. The benefits of this approach contrasts the transfer of large tables across the network where overheads would be significantly higher, affecting the scalability of the system, and application responsiveness would degrade as a result.

## **Optimisations**

The use of data compression improves the storage scalability of the solution, and increases disk performance. Together with physical indexing of tables, this approach significantly reduces the latency between database and applications.

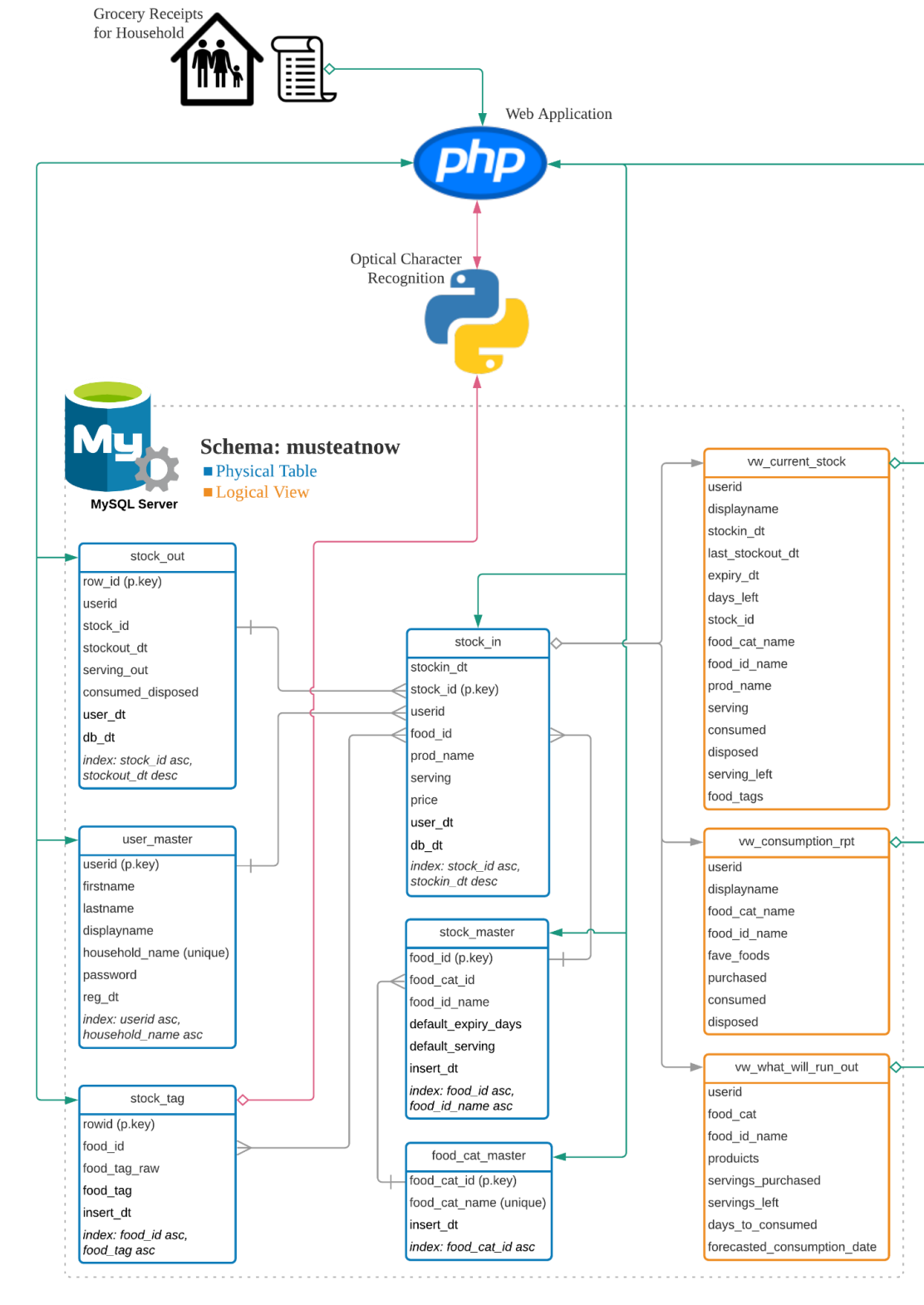


Figure 4: Database schema design diagram



## **Smart OCR System**

The Smart OCR system has been designed with the main functionality of taking in an image input and to output with a structured text format with the relevant information. It has been designed with two key subsystems with the following functional breakdowns:

1. Image Reader subsystem: to read in an input image and output the detected lines of text. This was implemented with an open-source conventional OCR system to take advantage of their accuracy and robustness.
2. Text Parser subsystem: to take in the unstructured text data from the Image Reader, and to extract the relevant information for output as a structured text data format.

### **Image Reader Subsystem**

In deploying the open-source Tesseract OCR engine, the project was able to integrate and make use of the accuracy of the pretrained model to achieve the required functionality of extracting text data from image data. However, the use of an out-of-the-box OCR engine also introduces some limitations and disadvantages.

Firstly, preprocessing the image would be an important step in the system as the accuracy of the output would be greatly improved if the images could be preprocessed in a similar way to the training data as much as possible. Unlike the training data with mostly clean background, receipt images typically contain some background, and this would affect the accuracy of the output. The Image Reader subsystem implements preprocessing steps such as thresholding and edge detection, in order to attempt to identify the receipt region, remove and crop the background out of the image as much as possible (see **figure 5**).

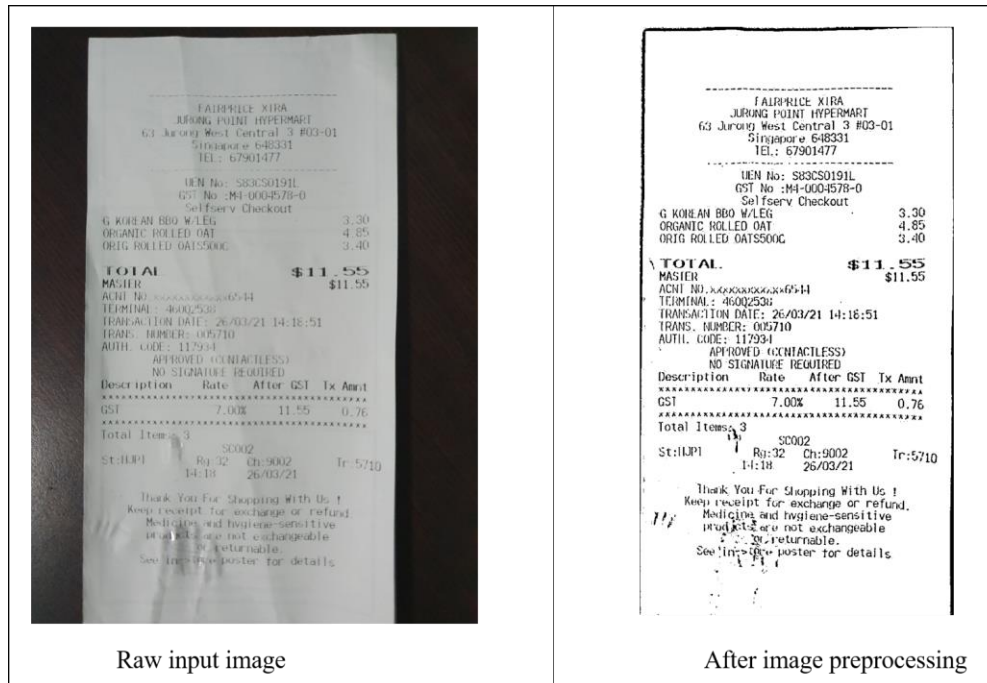


Figure 5: result of image preprocessing

In addition, on inspecting the training data of the Tesseract OCR engine, it was observed that data such as bigrams, unigrams and their frequency data has been used in the training. However, typical receipt images contain many short-forms, acronyms and special characters, which would likely differ from the standard bigram/unigram frequency distribution of general text documents with standard English words. As the expected input data differs from the standard training data, this might introduce some inaccuracies into the output of the OCR engine.

Nonetheless, the efficiency and time-savings from deploying a pre-trained model outweighs the disadvantages. In order to manage these challenges from the Image Reader subsystem, the project has implemented additional checks in the Text Parser subsystem to handle these expected uncertainties arising from the output of the OCR engine.

## Text Parser Subsystem Description

The Text Parser subsystem implements a rule-based inference model in order to parse through the lines of text and extract relevant information such as the items purchased, quantity and price. The key considerations taken in designing this subsystem was the lack of labelled training data, the expected standard format of the receipt text input, and the specific task set the system was

designed for. As the system is designed for the specific task of reading unstructured receipt text data, it is relatively simple for a human to analyse the receipts format and derive the rules for text parsing. In comparison, gathering and labelling large amounts of data for other forms of supervised training models would be much more time-consuming and may not be necessary for this task.

### **Text Parser Rule-Based Inference Model**

The rules for the system were derived through a manual extraction process by analyzing the format of information printed on the receipts. As a first prototype, the rule set was developed only based on the format of NTUC Fairprice Supermarket receipts. Two main patterns were identified and implemented in the system: (see **figure 6** for details)

1. Pattern 1: the item description and price are printed on a single line and the quantity purchased is one.
2. Pattern 2: the item description is printed on the first line, and the quantity and price is printed on the next line.



Figure 6: Patterns that the rule set was developed for (NTUC)

Each new line of text is first treated as a single observation, with a Boolean state each for item and price identification. The rules were designed to match the observations with either pattern 1 or 2. In cases of pattern 2 match, both lines of text would then be grouped as one observation.

Before the rules are matched, simple preprocessing steps such as case normalisation, special characters removal and blank line removals were also implemented. In addition, as part of preprocessing, the system also attempts to crop off the text to keep only the region with the lines containing items purchased. As the rule sets were developed for this specific task, general text lines like the headers would not fit well into the rules and result in rubbish output.

The model inference flowchart for the system implementation is shown in **figure 7** below. In the current implementation, these rules have priority and the system attempts to match pattern 1 before pattern 2. In cases of partial match with either pattern 1 or pattern 2, the system matches it as pattern 1, with missing information to be filled in with default values. Such an implementation

was enforced as the assumption of pattern 1 match would be safer and more reliable. This is because a wrong assumption of pattern 2 match would cause the two lines of text to be grouped as one observation, and could lead to rolling errors that greatly affect the output of the model. These rules can be simply summarized as below:

1. IF the state of item is true, AND the state of price is true, THEN the observation matches with pattern 1.
2. IF the state of item is true AND the state of price is false AND the state of next line's item is false AND the state of next line's price is true, THEN the observation matches with pattern 2.

**Text Parsing inference flow chart:  
unstructured to structured text data**

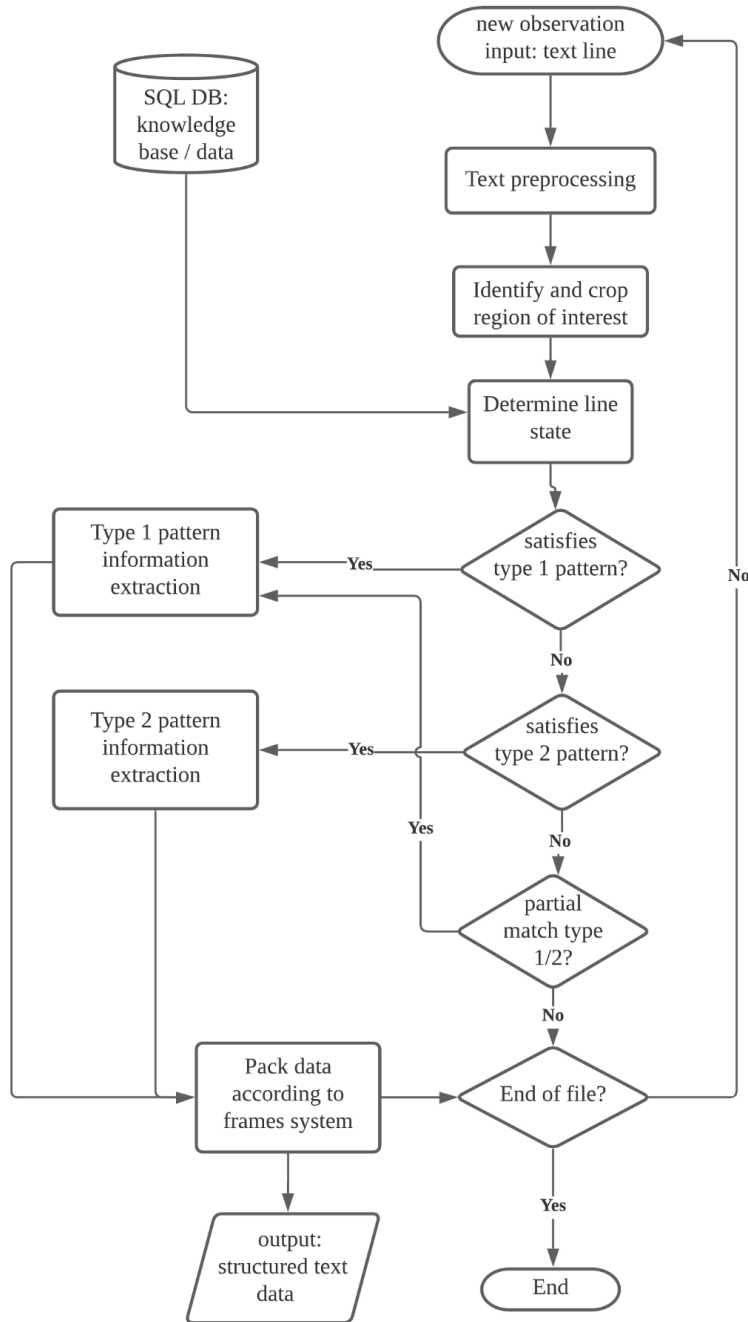


Figure 7: model inference flowchart

## Text Parser Observation State Estimation

In order to determine the Boolean price and item state of each text line, rules are built into the state estimation inference function. Similarly, these rules are derived from a manual extraction process by analyzing and comparing the output with the actual data, and tuning the model rules to improve the output. In addition, these rules take into consideration the expected inaccuracies from the OCR input and the relatively limited SQL knowledge base. The inference flowchart for this observation state estimation function is shown in **figure 8**.

Determination of price state was implemented through a simple regex string matching based on the standard price pattern. During the tuning of the rule sets, it was observed that the OCR engine commonly misidentified “,” and “.”. Therefore, the regex matching for price state compensates for these inaccuracies by allowing such cases as a correct match.

For the determination of item state, it was implemented with a word matching against the database food corpus to identify any food items in the text line. As it would be expected that there would be new items that do not exist in the database, additional rules were thus added to infer the possibility of having some unknown items in the text line. These rules include alpha characters percentage count, basic keywords filtering and running a spell check though the most probable word. In addition, the spellcheck is built with a custom dictionary corresponding to the same grocery food corpus database in order to improve the relevance of the output from the spellcheck. Coupled with the training feedback loop for the grocery food corpus, the performance of the spellcheck is expected to improve over time. With the above rules, the state estimation function attempts to correctly estimate the Boolean state of a text line, taking into consideration the unknown factors and inaccuracies of the input data.

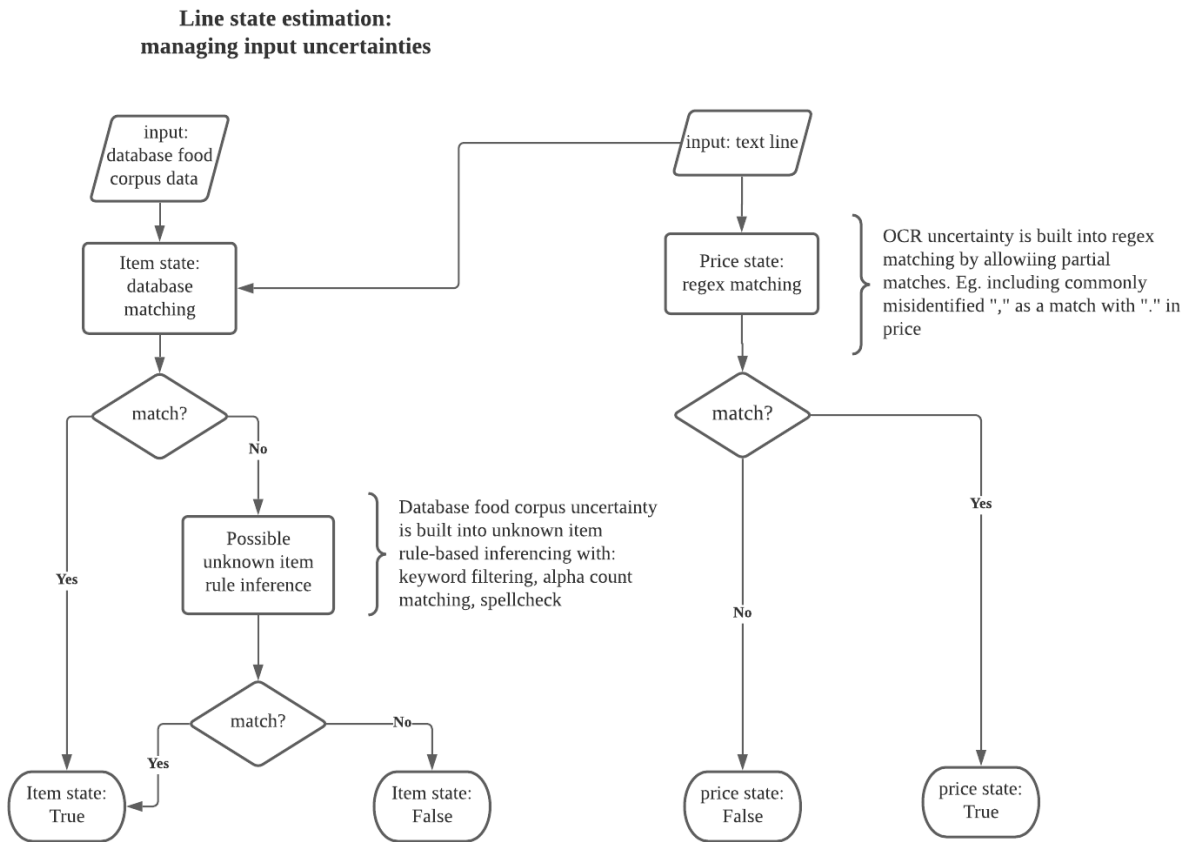


Figure 8: observation state estimation flowchart

## Data Representation

These information were represented with a frames system corresponding to each observation (see **Table 4** for an example of a single frame for an observation). As the smart OCR reasoning model takes input from the database system and sends output data to the frontend system, the frames representation would support the data transfer to these systems which were implemented in different programming languages. As such, the use of the frames system ensured the proper integration of the full system with a standard data representation format.

Table 4: Example of a single frame for a single observation

Slots	Filter
Text String	Kf baby spinach 200g




Food Tag	Spinach
Food ID	211
Price	1.10
Quantity	1

### User Interface Integration

In order to ensure the system functionalities were easily accessible to the users, the frontend user interface plays an integral role in the overall integration. The frontend system acts as the key data transfer channel between the various subsystems. Acting as the main entry point for data input, the user interface prompts the user to upload the receipt image, which would then be passed down the pipeline for model inferencing. Also, not all data received from the model are relevant to the user. As such, the frontend system ensures that only relevant information would be presented to the user while the rest of the data is sent to the database for storage and processing.

In addition, the user interface supports the continuous model training process. It compensates for the inaccuracies of the model by building in various flexible UI features which allows users to provide the relevant inputs into the system. Unlike standard English words, text printed on the receipt tends to contain many short-forms and variations. For example, yoghurt can be printed as yog, yogh, yoghurt, yogurt, or even s/yog. Therefore, it was imperative for the system to be able to recognise these variations and label these words to the correct item in the database. As such, the frontend system has built in features (see **figure 9**) which allow the user to tag these short forms to the correct category. This information is then updated into the database grocery food corpus to support the continuous model training process. With the integration of the frontend system, the overall system was thus able to achieve its functionality of reading image input and extracting and presenting the relevant information to its users.

User selection for items  
unidentified by model



Unidentified Products


Identifying Tag	Description	Price	Quantity	Classification
promo.h	fresh fruits promo.h	\$7.90	2	<div style="border: 1px solid #ccc; padding: 2px;">Fruit - Apples</div>
toppingn.komi40	mr. toppingn.komi40	\$4.90	1	<div style="border: 1px solid #ccc; padding: 2px;">Do Not Store This Item</div>
topping-s.fune40	mr. topping-s.fune40	\$4.90	1	<div style="border: 1px solid #ccc; padding: 2px;">Do Not Store This Item</div>

Add Items

Figure 9: UI features to integrate model training functionality

## Grocery Stock Level

The Grocery Stock Level system is the second subsystem designed and developed for the MustEatNow! application. In a nutshell, the Grocery Stock Level page shows the existing food stock amount for the current user. The entire list is broken into two sections: expired and fresh food items.



App ▾ About Us Features Contact Us

Grocery Stock Level					
<b>Must Throw Now!</b> (expired, eat at your own risk)					
Category	Food Name	Description	Qty Left	Days Expired	
Dairy	butter	scs salted butter	1	37	✗
Beverage	yakult	yakult	5	28	✗
Dairy	yogurt	greek yogurt	2	27	✗
Dairy	yogurt	plain yogurt	2	21	✗
Beverage	yakult	yakult	4	20	✗
Dairy	butter	bordier butter	1	20	✗
Meat	pork	pork loin	1	19	✗
Fruit	apples	busan apple	1	18	✗
<b>The Fresh List</b> (can eat now)					
Category	Food Name	Description	Qty Left	Days Left	
Dairy	yogurt	greek yogurt	1	1	📝
Fruit	apples	busan apple	1	4	📝
Dairy	yogurt	greek yogurt	1	5	📝
Meat	chicken	chicken breast	4	7	📝
Grain	bread	4 sh pkt bread/bun 350	4	7	📝
Grain	bread	4 sh pkt bread/bun 350	4	7	📝
Fruit	apples	busan apple	1	8	📝

Figure 10: Grocery Stock Level UI

The key functionality of the Grocery Stock Level system is to make use of certain fixed rules in the database knowledge (namely, the food expiration dataset) to make meaningful representations and visualizations of the user's food and grocery data in an easily interpretable and relevant format.

Food that have crossed their expiry date rule set are deemed to have expired and only fit for disposal. Food that are still within their “fresh” state can be either consumed or disposed as the user sees fit. The different colors used in the user interface are also based on a simple rule set to provide a stronger visualisation pattern to the user.

A sample of the Food Expiry data rules is given in the screenshot below. The rules are stored as a knowledge base in the database and referenced in the backend code when matching

purchased food items to their corresponding expiry dates.

	food_id	food_cat_id	food_id_name	default_expiry_days	default_serving	insert_dt
	10	2	Crab	8	1	2021-04-28 09:14:16
	11	2	Processed Se...	30	1	2021-04-28 09:14:16
	12	3	Fresh Vegeta...	8	1	2021-04-28 09:14:16
	13	3	garlic	10	1	2021-04-28 09:14:16
	14	3	onion	10	1	2021-04-28 09:14:16
	15	3	parsley	10	1	2021-04-28 09:14:16
	16	3	cilantro	10	1	2021-04-28 09:14:16
	17	3	potato	10	1	2021-04-28 09:14:16
	18	3	spinach	10	1	2021-04-28 09:14:16
	19	3	chive	10	1	2021-04-28 09:14:16
	20	3	ginger	10	1	2021-04-28 09:14:16
	21	3	chilli	10	1	2021-04-28 09:14:16
	22	4	Apples	10	1	2021-04-28 09:14:16
	23	4	Bananas	10	10	2021-04-28 09:14:16
	24	4	pineapple	10	1	2021-04-28 09:14:16
	25	4	Pear	10	1	2021-04-28 09:14:16
	26	4	mango	10	1	2021-04-28 09:14:16
	27	4	strawberries	10	1	2021-04-28 09:14:16

Figure 11: Food expiry data rules

An SQL SELECT query is then used to match and return food items that have expired. The resultant dataset is then displayed to the user in the frontend interface.

```
<?php
$query = "SELECT * FROM vw_current_stock WHERE userid = ? AND serving_left > 0 AND days_left <= 0 ORDER BY days_left ASC";
$result = PDO_PreparedSelect_Array($conn, $query, array($userID));
if($result)
{
    echo '<table class="table table-hover" id="dataTable">';
```

Figure 12: SQL implementation for food expiry data

A color rule set is then used to provide the visuals to the user. For example, all expired foods are highlighted in red and the user can only dispose of them.

The Fresh List contains foods that have not exceeded its expiry date. In the Fresh List, there is another set of rules that determine the color of the row in order to provide a clearer visual representation to the user. The rules are:

- expiry date within 0 – 3 days :: dark yellow
- expiry date within 4 – 8 days :: light yellow
- expiry date 9 days and beyond :: white

The Fresh List (can eat now)					
Category	Food Name	Description	Qty Left	Days Left	
Beverage	yakult	yakult	1	1	
Dairy	yogurt	greek yogurt	1	2	
Meat	pork	pork loin	1	4	
Fruit	apples	busan apple	1	5	
Dairy	yogurt	greek yogurt	2	6	
Vegetable	fresh vegetable	cold storage	1	7	
Meat	chicken	chicken breast	6	8	
Grain	bread	4 sh pkt bread/bun 350	4	8	
Grain	bread	4 sh pkt bread/bun 350	4	8	
Meat	pork	nat.shabu shabu250g	1	9	
Meat	ham	sfp hny b/ham200g 7 ee	1	9	
Meat	ham	sfp hny b/ham200g 7 ee	1	9	
Dairy	butter	bordier butter	1	9	

Figure 13: UI implementation of food expiry visual representation

### **Smart Purchase Recommender System**

The goal of this feature is to aid the user in making grocery shopping decisions. Specifically, it tells the user which of the items in their inventory is projected to be consumed fully. On the frontend, the user has the option to ask the feature which food items would be consumed from current day up to a week.

The data hierarchy for food items are as follows:

1. Food\_cat\_id: Food category(dairy, grain, meat)
2. Food\_id: Food type(milk, rice, chicken)
3. Stock\_id: product (HL milk, golden pineapple rice, chicken leg)

It first looks at the products that have been consumed fully in the past(including disposed products), and derives the number of days to finish it. Secondly, at a product level, it takes the maximum consumption rate value from this group, and aggregates it to the food level. This will give us the consumption rate at each food type level, and it is then multiplied by the amount of remaining servings left to inform the user, in how many days this food product would be consumed.

This method is useful as we can apply the consumption rate to new products with no prior consumption history, and make a fairly reasonable estimation as it shares the same food type. For example, it takes a household 4 days to consume a carton of “Meiji milk”, the user decides to try a new product “Hokkaido milk”, and we can estimate that it would approximately take the same time to consume “Hokkaido milk” fully.

For products that are the first in their category(no user history of consuming this product), we can aggregate the consumption rates for known food item, and apply it to the food category level. For example, a family has consumed products “Meiji milk 300ml”, “greek yogurt”, and “farm fresh eggs” at 1 serving a day. These 3 products fall under food types “milk”, “yogurt”, and “eggs”, and share the food category “dairy”.

The user decides to try a new product “Häagen-Dazs ice-cream” which will be tagged by the OCR subsystem as “ice-cream”. Assuming there is no consumption history of the “ice-cream” food type for this user, the corpus will identify the food type “ice-cream” as “dairy” in food

category, and apply the aggregate the known consumption rates of “milk”, “yogurt”, and “eggs, and apply it to “ice-cream. The final output to the user is the projected consumption date of “ice-cream”, given the total current stock available for this food type.

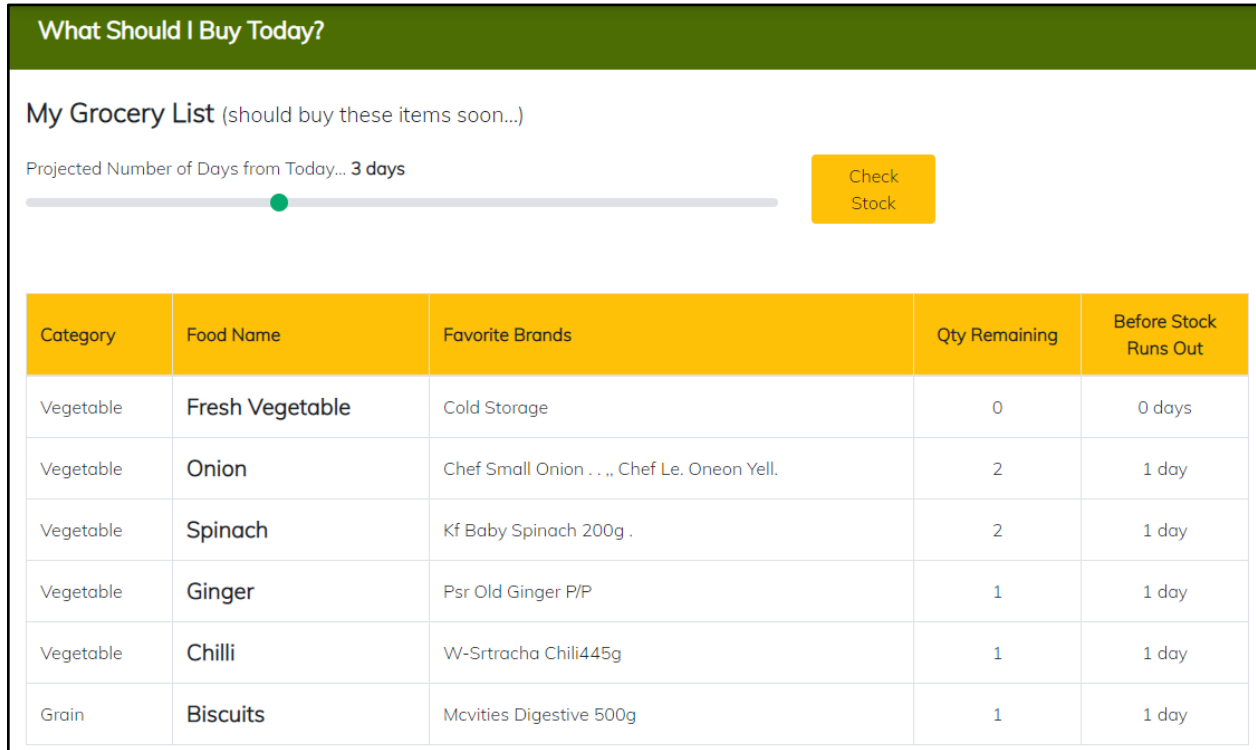


Figure 14: output of smart purchase recommendation

## **System Deployment Architecture**

The System was implemented as a web-based application residing on a cloud server architecture. It was deployed on the DigitalOcean Cloud Platform. This was essential in ensuring the easy accessibility of the system functionalities to the end users. The following technological components were involved in the deployment of the application.

1. A server running the Linux Ubuntu (v 20.04) operating system and serviced by the Apache HTTP Server system was deployed to host the core application.
2. A MySQL (community edition) database was deployed on a separate dedicated server and serves as the knowledge base and data repository for the system.
3. The web application was created using HTML, CSS and JavaScript for its frontend interface and the PHP language for its backend logic control.
4. A Python (v.3) environment was deployed on the web server for the Smart OCR component to interface with the PHP backend.
5. A web domain (musteatnow.com) was purchased and set up to point to the IP Address of the web server which allows users to use a normal browser to access the application.

## **PROJECT PERFORMANCE AND VALIDATION**

The three key objectives that were identified for the project were: (1) to reduce time taken for inventory log updates, (2) to simplify the decision making process in grocery shopping, (3) to reduce food wastage due to disposal of expired food. The project performance would be validated against these three objectives.

### **Semi-automated record logging**

To achieve a fast and convenient inventory update process, the implemented system was able to quickly and effectively populate the inventory records based on receipt identification of the grocery items purchased (see **figure 15**).



FAIRPRICE XTRA NEX HYPERMART 23 Serangoon Central #03-42 Singapore 556083 TEL: 6634 4947/4948		
UEN No: S83CS0191L GST No :M4-0004578-0		
A PACKHAM PEAR 800G	6.75	
COWHEAD CHEESE 3x250	7.95	
Normal Price 9.85		
D T/C YOG SBERY150G		
2X\$2.75/Unit	5.50	
2 FOR \$3.95	-1.55	
F GREEK STRAWBERRY90		
3X\$1.20/Unit	3.60	
3 FOR \$2.85	-0.75	
FP HNY B/HAM200G	2.95	
FRESH BLUEBERRIES		
3X\$3.50/Unit	10.50	
3 FOR \$6.95	-3.55	
FRESH FRUITS PROMO H		
2X\$3.95/Unit	7.90	
KF BABY SPINACH 200G	1.10	
KORU NZ/US APPLE800G	5.95	
Normal Price 7.90		
M R.TOPPING-N.KOMI40	4.90	
M R.TOPPING-S.FUMI40	4.90	
NAT.SHABU SHABU250G	7.00	
P CHSE SMKY CHDR125G	5.55	
S G/C TB-REG/50FT 6S	11.85	
SH PKT BREAD/BUN 350		
2X\$3.50/Unit	7.00	
USA R'BERRY(A/FLOWN)		
2X\$6.95/Unit	13.90	
LINK CARD 8105000001824397		
<b>TOTAL</b>	<b>\$101.45</b>	
<b>MASTER</b>	<b>\$101.45</b>	

My Grocery List (identified products)

Product Name	Product Tag	Description	Price	Quantity
Pear	pear	a packham pear 800g	\$6.75	1
Cheese	cheese	cowhead cheese 3x250	\$7.95	1
Strawberries	strawberry	f greek strawberry90 .	\$3.60	3
Ham	ham	sfp hny b/ham200g 7 ee	\$2.95	1
Blueberries	blueberries	fresh blueberries 4	\$10.50	3
Spinach	spinach	kf baby spinach 200g .	\$1.10	1
Apples	apple	koru nz/us apple800g 595	\$7.90	4
Pork	nat shabu	nat.shabu shabu250g	\$7.00	1
Bread	bun	4 sh pkt bread/bun 350	\$7.00	2

Figure 15: End-to-end results for semi-automated record logging

In implementing this solution, the project has achieved significant time savings for the task of inventory records update. As compared to manual record logging which might take about 30 seconds to input 10 records, the implemented system would be able to populate the records in about 5 seconds with an additional few more seconds for the user to make minor amendments due to model inaccuracies. This presents a time savings of approximately 100% for 10 records. Additionally, with an increased number of records, the amount of time required for manual logging increases linearly. In comparison, for our implemented solution, the amount of time required would remain largely the same regardless of the number of records. Hence, the project objective in this aspect has been successfully implemented and achieved.

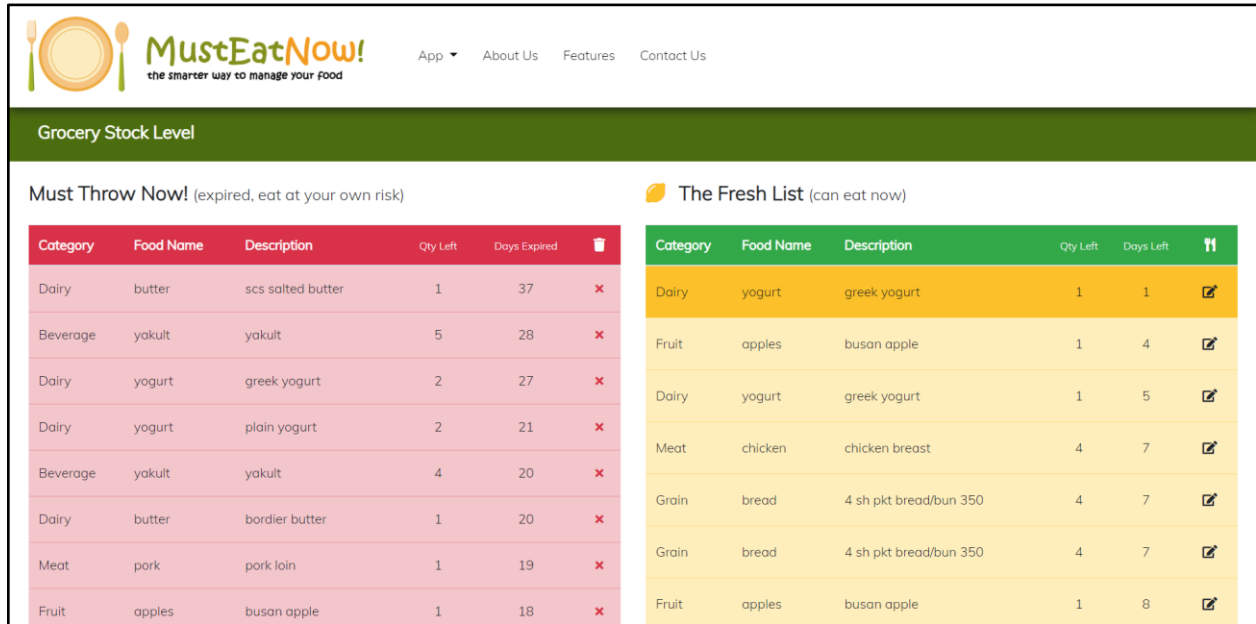
### Smart curated grocery list

Next, to simplify the decision making process in grocery shopping, we have implemented a purchase recommendation system which provides the user with a curated shopping list. It is a rule based system making use of current inventory stock level and their past consumption habits. This streamlines the process of users having to manually check their stock level and decide what to purchase, which is usually based on their personal intuition. We have thus implemented a data-based approach to help users curate their shopping list. This results in time and effort savings.

Hence, the project objective in this aspect has been successfully implemented and achieved.

### Consumption reminders

Lastly, to reduce food wastage, the implemented system makes rule-based, visual representations of expiry reminders to ensure that users are reminded to consume the food before the expiry (see **figure 16**).



The screenshot shows the MustEatNow! app interface. At the top is the logo and navigation links. Below is a green header for 'Grocery Stock Level'. The main content area is divided into two sections: 'Must Throw Now! (expired, eat at your own risk)' and 'The Fresh List (can eat now)'. Each section contains a table with food items, their quantities, and expiration dates.

Must Throw Now! (expired, eat at your own risk)						The Fresh List (can eat now)					
Category	Food Name	Description	Qty Left	Days Expired		Category	Food Name	Description	Qty Left	Days Left	
Dairy	butter	scs salted butter	1	37	✗	Dairy	yogurt	greek yogurt	1	1	✍
Beverage	yakult	yakult	5	28	✗	Fruit	apples	busan apple	1	4	✍
Dairy	yogurt	greek yogurt	2	27	✗	Dairy	yogurt	greek yogurt	1	5	✍
Dairy	yogurt	plain yogurt	2	21	✗	Meat	chicken	chicken breast	4	7	✍
Beverage	yakult	yakult	4	20	✗	Grain	bread	4 sh pkt bread/bun 350	4	7	✍
Dairy	butter	bordier butter	1	20	✗	Grain	bread	4 sh pkt bread/bun 350	4	7	✍
Meat	pork	pork loin	1	19	✗	Fruit	apples	busan apple	1	8	✍
Fruit	apples	busan apple	1	18	✗						

Figure 16: Rule-based visual representations of expiry reminders

In addition, the project hopes that this in the long run also changes the consumption and purchase habits of the users, to be more mindful and less wasteful by tracking their historical data. Due to the short timeline of the project, there was insufficient time for data collection post-deployment to verify quantitatively the results of the outcome. The amount of data available within the short timeframe post-deployment could be simply a part of the usual variance in consumption patterns, and likely to be statistically insignificant. There is insufficient evidence for the project to quantitatively conclude that there has been meaningful reduction in food wastage. Nonetheless, the project has implemented the relevant features, and would be able to verify the performance in this area in future.

## **FINDINGS AND RECOMMENDATIONS**

In general, the implemented system was able to achieve its key functionalities and objectives that it had been designed for. Nonetheless, there are some limitations and areas of improvements that could be further made to expand the functionalities and business use case of the system. These include expanding the smart OCR model reasoning task scope to achieve better generalization, and expanding the UI features for improved useability and flexibility.

### **Improving model accuracy with automated diagnostics**

One of the key limitations of the smart OCR system was its reliance on a reasonably accurate output from the OCR engine, in order for the text parsing rule set to achieve a good output. The image preprocessing pipeline thus becomes a core part of the system in ensuring this. Hence, for further development, more steps and optimization can be put in place in the image preprocessing pipeline. These could include subtasks such as perspective dewarping and text alignment to try and align the image more accurately. In addition, the image reader subsystem could also implement some form of independent automated diagnostics to assess the quality of the OCR engine output, before sending the output to the text parsing model. These improvements would likely help in improving the accuracy of the output of the overall system.

### **Improving model generalization**

In addition, due to the short timeline, the implementation only took into consideration receipt printing formats by NTUC Fairprice Supermarket. These rules would not generalize well to other receipts that do not follow the same format of information printing. While the system might still be able to extract some information such as the item purchased, other information such as the price and quantity would not be properly extracted. For future development, it would be advantageous to expand the problem scope to include receipts from other supermarkets. It could be implemented as either one single generalized model or various different models custom for a particular supermarket. These would allow the system to achieve better generalization and functionality overall.

**Improving user input flexibility**

In this project, we have implemented the core UI features necessary for the system functionalities to be achieved. Nonetheless, there are some feature improvements which could be made to further improve the flexibility of the system. For example, the current implementation allows the users to tag unknown food items to an existing category in the database. In future. This could be expanded to allow users to add in new categories for more flexibility and to scale up the scope of the database coverage. Besides that, users could also be allowed to amend the quantity and description. This could make it more user friendly as users may have their own preference for the item description in the database. These feature improvements could make the overall user experience more positive and intuitive.

**CONCLUSIONS**

In conclusion, the project has successfully implemented the MustEatNow system with the aim of supporting the users to make smarter management of their household perishables inventory. With the use of a smart OCR system, the project has implemented a semi-automated records logging feature to reduce the time and effort required for the task. The project also implemented a purchase recommendation system to provide a curated grocery shopping list based on the user's past consumption and inventory data. Lastly, the project implemented a consumption reminder system to make meaningful representations of inventory data to the user, to remind and encourage timely consumption of food items.