

WEB-BASED COFFEE SHOP WITH CHATBOT INTEGRATED

CHEAH WEN HUEY

**A project report submitted in partial fulfilment of the
requirements for the award of Bachelor of Science
(Honours) Software Engineering**

**Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman**

October 2023

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : *Cheah Wen Huey*

Name : Cheah Wen Huey

ID No. : 18UEB04877

Date : 06/10/2023

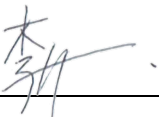
APPROVAL FOR SUBMISSION

I certify that this project report entitled “**WEB-BASED COFFEE SHOP WITH CHATBOT INTEGRATED**” was prepared by **CHEAH WEN HUEY** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Science (Honours) Software Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature

:



Supervisor

:

Lee Ming Jie

Date

:

06/10/2023

Signature

:

Co-Supervisor

:

Date

:

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2023, Cheah Wen Huey. All right reserved.

ACKNOWLEDGEMENTS

Firstly, I would like to extend my heartfelt gratitude to my supervisor, Dr. Lee Ming Jie, for his guidance throughout my journey. I am sincerely thankful for his patience, especially when dealing with my last-minute submissions.

I also want to express my deep appreciation to my mother, whose unwavering support and encouragement have been my anchor in navigating life's challenges and triumphs.

Last but not least, I am grateful for my friends, who have always been there to lend a helping hand whenever I needed it. I am genuinely thankful for all the wonderful people who have touched my life and made it richer.

ABSTRACT

In a world where coffee stands as an enduring favorite, it has become a daily ritual for many. The global demand for coffee has spurred the growth of tools and an expanding variety of coffee beans. However, in the midst of the evolving coffee culture and the proliferation of trendy cafes, modern coffee websites often lack a truly comprehensive product range. Introducing a web-based coffee shop that caters to coffee enthusiasts' every need. This platform offers an extensive selection of coffee products, a diverse range of beverages, and high-quality home brewing tools and equipment. It goes beyond mere merchandise by providing valuable insights into products, brewing techniques, and various coffee-related topics. Recognizing the importance of customer support in the face of such a vast selection, the platform seamlessly integrates a chatbot feature. This intelligent assistant offers not only personalized product recommendations but also around-the-clock support. Customers can find answers to common queries or obtain assistance with orders, day or night, ensuring a seamless and stress-free shopping experience. In summary, this project reimagines the online coffee shopping experience by combining a comprehensive product selection with AI-powered support, ultimately enhancing customer satisfaction.

TABLE OF CONTENTS

TABLE OF CONTENTS		i
LIST OF TABLES		v
LIST OF FIGURES		viii
LIST OF SYMBOLS / ABBREVIATIONS		xviii
LIST OF APPENDICES		xix
CHAPTER		
1	INTRODUCTION	20
1.1	General Introduction	20
1.2	Problem Statement	21
1.3	Project Objectives	23
1.4	Proposed Solution	24
1.5	Proposed Approach	25
1.6	Project Scope	26
1.6.1	Targeted Users	26
1.6.2	Modules Covered	27
2	LITERATURE REVIEW	30
2.1	Introduction	30
2.2	Similar Existing Web Applications	30
2.2.1	Noc Coffee Company	30
2.2.2	Starbucks Coffee Company	35
2.2.3	Seattle Coffee Gear	45
2.2.4	The Coffee Bean & Tea Leaf	54
2.2.5	Comparison Between Existing Similar Web Applications	58
2.3	Programming Framework	59
2.3.1	Laravel	60
2.3.2	Reactjs	62
2.3.3	Node.js	64

	2.3.4 Comparison Between Programming Frameworks	66
2.4	Software Development Methodology	67
	2.4.1 Waterfall Methodology	67
	2.4.2 Prototype Methodology	69
	2.4.3 Rapid Application Development (RAD) Methodology	71
	2.4.4 Feature Driven Development (FDD) Methodology	73
	2.4.5 Comparison Between Software Development Methodologies	75
3	METHODOLOGY AND WORK PLAN	77
	3.1 Introduction	77
	3.2 Software Development Methodology	77
	3.2.1 Requirement Analysis	78
	3.2.2 Prototyping Iteration	79
	3.2.3 Develop	82
	3.2.4 Test	83
	3.2.5 Release	83
	3.3 Project Plan	84
	3.3.1 Work Breakdown Structure (WBS)	84
	3.3.2 Gantt Chart	88
	3.4 Project Development Tools	89
	3.4.1 Laravel	89
	3.4.2 Bootstrap	89
	3.4.3 Visual Studio Code	89
	3.4.4 OpenAI	90
	3.4.5 PhpMyAdmin	90
4	PROJECT INTIAL SPECIFICATION	91
	4.1 Introduction	91
	4.2 Facts Finding	91
	4.3 Functional Requirements	98
	4.4 Non-Functional Requirements	100
	4.5 Use Case Diagram	101

4.6	Use Case Description	102
4.7	User Interface (UI) Prototype	124
5	SYSTEM DESIGN	129
5.1	Introduction	129
5.2	System Architecture Design	129
5.3	Chatbot Data Embedding Interaction with OpenAI	131
5.4	Chatbot Question and Answering Interaction with OpenAI	133
5.5	Database Architecture Design	134
5.6	Data Flow Diagram	136
	5.6.1 Context Diagram	136
	5.6.2 Level 1 Data Flow Diagram	137
	5.6.3 Activity Diagram	138
5.7	UI Design	155
	5.7.1 Introduction	155
	5.7.2 Register Page	155
	5.7.3 Login Page	156
	5.7.4 Home Page	157
	5.7.5 Beverage List Page	158
	5.7.6 Homebrew Product List Page	159
	5.7.7 Product Detail Page	160
	5.7.8 Write Review Modal	163
	5.7.9 Shopping Cart	164
	5.7.10 Admin Panel	166
	5.7.11 Order List Page	166
	5.7.12 Recipe Guiding Page	167
	5.7.13 Brewing Guide Page	169
	5.7.14 Chatbot Pop Up Window	171
6	SYSTEM IMPLEMENTATION	173
6.1	Authentication	173
6.2	Product List with Search and Filter features	175
6.3	Product Detail	178
6.4	Beverage Customization	179

6.5	Product Recommendation	180
6.6	Product Rating	181
6.7	Chatbot Interaction with OpenAI	183
	6.7.1 OpenAI Data Embedding	183
	6.7.2 Question and Answering	185
6.8	Data Embedding Preparation	188
6.9	Shopping Cart	190
	6.9.1 Session and Database Cart Implementation	190
	6.9.2 Cart CRUD Implementation	199
6.10	Stripe Payment Gateway	210
6.11	Order List	214
6.12	Admin Panel	215
	6.12.1 Manage Products	215
	6.12.2 Manage Guiding Materials	218
	6.12.3 Manage Data Embedding File	220
7	SYSTEM TESTING	223
	7.1 Unit Testing	223
	7.2 Integration Testing	240
	7.3 User Acceptance Testing (UAT)	243
	7.4 System Usability Testing	248
8	CONCLUSION AND FUTURE WORKS	253
	8.1 Project Objectives Milestone	253
	8.2 Limitations and Recommendations	253

LIST OF TABLES

Table 2.2.1: Comparison Between Existing Applications	58
Table 2.3.1: Key Features of Laravel	61
Table 2.3.2: Advantages and Disadvantages of Laravel	61
Table 2.3.3: Key Features of Reactjs	62
Table 2.3.4: Advantages and Disadvantages of Reactjs	63
Table 2.3.5: Key Features of Node.js	64
Table 2.3.6: Advantages and Disadvantages of Node.js	65
Table 2.3.7: Comparison Between Programming Frameworks	66
Table 2.4.1: Advantages and Disadvantages of Waterfall Model	68
Table 2.4.2: Advantages and Disadvantages of Prototype Model	70
Table 2.4.3: Advantages and Disadvantages of RAD Model	72
Table 2.4.4: Advantages and Disadvantages of FDD Model	74
Table 2.4.5: Comparison Between Software Development Methodologies	75
Table 4.2.1: Results of Participants' Prioritization	97
Table 4.6.1: Use Case Description – Register Account	102
Table 4.6.2: Use Case Description – Login Account	103
Table 4.6.3: User Description – View Orders	104
Table 4.6.4: Use Case Description – Rate and Review Product	105
Table 4.6.5: Use Case Description – View Product Detail	107
Table 4.6.6: Use Case Description – View Guiding Materials	108
Table 4.6.7: Use Case Description – Purchase Product	109
Table 4.6.8: Use Case Description – Make Payment	110
Table 4.6.9: Use Case Description – Create Order	111

Table 4.6.10: Use Case Description – Search and Filter Product	112
Table 4.6.11: Use Case Description – Manage Cart	114
Table 4.6.12: Use Case Description – Initiate chat	115
Table 4.6.13: Use Case Description – Answer Question	116
Table 4.6.14: Use Case Description – Manage Product	117
Table 4.6.15: Use Case Description – Manage Guiding Materials	119
Table 4.6.16: Use Case Description – Manage Data Embedding File	121
Table 5.5.1: Database Architecture Design Table	135
Table 7.1.1: Unit Test Case – User Register	224
Table 7.1.2: Unit Test Case – User Login	225
Table 7.1.3: Unit Test Case – View Product List	227
Table 7.1.4: Unit Test Case – Search and Filter Product	227
Table 7.1.5: Unit Test Case – Manage Cart	229
Table 7.1.6: Unit Test Case – Write a Review	230
Table 7.1.7: Unit Test Case – View Order List	231
Table 7.1.8: Unit Test Case – Interact with Chatbot	232
Table 7.1.9: Unit Test Case – View Payment Page	233
Table 7.1.10: Unit Test Case – User Payment Process	233
Table 7.1.11: Unit Test Case – Manage Product	234
Table 7.1.12: Unit Test Case = Manage Data Embedding File	235
Table 7.1.13: Unit Test Case – Manage Guiding Material	237
Table 7.1.14: Unit Test Case – Admin Login	238
Table 7.2.1: Integration Test Case – Product and Image	240
Table 7.2.2: Integration Test Case - Guiding Material and Image	241
Table 7.2.3: Integration Test Case – Data Embedding File and OpenAI	

Table 7.2.4:	Integration Test Case – Payment Process and Create Order	242
Table 7.2.5:	Integration Test Case – Payment Process and Delete Order	242
Table 7.2.6:	Integration Test Case – Payment Process and Update Order	242
Table 7.2.7:	Integration Test Case – Create Order and Create Order Detail	242
Table 7.2.8:	Payment Process and Delete Cart Items	243
Table 7.2.9:	Login and Transfer Cart Listener	243
Table 7.3.1:	User Acceptance Testing Template for Customer	243
Table 7.3.2:	User Acceptance Testing Template for Admin	246
Table 7.4.1:	SUS Score for Web-based Coffee Shop	249
Table 7.4.2:	SUS Score for Admin Panel	250
Table 7.4.3:	Test Scenario for Customer	250
Table 7.4.4:	Test Scenario for Admin	251
Table 8.2.1:	Limitations and Recommendations Table	253

LIST OF FIGURES

Figure 1.4.1: System Architecture	25
Figure 1.5.1: Prototyping Methodology	25
Figure 2.2.1: Currency and Language Options	31
Figure 2.2.2: A sample of Simplified Chinese applied to the web page	32
Figure 2.2.3: Product Filtering with Gift Card filter applied	32
Figure 2.2.4: Shopping Cart page of Noc	33
Figure 2.2.5: Coffee capsule recommendations when users view on a coffee capsule product	34
Figure 2.2.6: Product Information Feature of Noc	34
Figure 2.2.7: Internal Roastery Process where (a) shows the philosophy of coffee bean and (b) shows the roasting equipment used for roasting process	35
Figure 2.2.8: Product filtering page with Whole Bean filter applied	37
Figure 2.2.9: Product purchasing page with third party websites provided	38
Figure 2.2.10: Product information page of Starbucks' purchasing website	38
Figure 2.2.11: K-Cup® Pods product recommendations when users view on K-Cup® Pods coffee	39
Figure 2.2.12: Product review sorted with most recent filter	40
Figure 2.2.13: Review writing page of the purchasing website	41
Figure 2.2.14: Product searching in the purchasing website	42
Figure 2.2.15: Recipe of Caramel Macchiato provided as a guiding materials in the purchasing website	42
Figure 2.2.16: Product filtering page with Hot Coffee filter applied.	43
Figure 2.2.17: Shopping cart page of Starbucks	44
Figure 2.2.18: Product information page of Starbucks' ordering website	44

Figure 2.2.19: Beverage customization of Starbucks' ordering website	45
Figure 2.2.20: Product filtering based on various product types	46
Figure 2.2.21: Product filtering page with price range between \$1000 to \$2000 filter applied	46
Figure 2.2.22: Shopping cart page of Seattle Coffee Gear	47
Figure 2.2.23: Product information of a coffee bean product	48
Figure 2.2.24: Product Information of Coffee Equipment	48
Figure 2.2.25: Comparison table between similar coffee equipment	49
Figure 2.2.26: Product Searching Page of Seattle Coffee Gear	50
Figure 2.2.27: Real Time Service page of Seattle Coffee Gear where (a) showing agents away and (b) showing agents online	51
Figure 2.2.28: Product review page with Most Recent and Star Ratings filters applied	52
Figure 2.2.29: Review writing page of Seattle Coffee Gear	53
Figure 2.2.30: Learning and guiding materials provided by Seattle Coffee Gear	54
Figure 2.2.31: Product filtering page with Coffee filter applied (The Coffee Bean and Tea Leaf® Malaysia)	55
Figure 2.2.32: Shopping Cart Page	55
Figure 2.2.33: Searching Result with the "caramel" keyword entered	56
Figure 2.2.34: Product Recommendation Page of CBTL	56
Figure 2.2.35: Product information of Brazil Cerrado, a coffee bean	57
Figure 2.2.36: A sample of the review form of CBTL's website	57
Figure 2.3.1: MVC pattern (Ighodaro, 2018)	60
Figure 2.4.1: Waterfall Model (Rault, 2022)	67
Figure 2.4.2: Prototyping Model (Despa, 2014)	69
Figure 2.4.3: Rapid Application Development (RAD) model (Despa, 2014)	71

Figure 2.4.4: Feature Driven Development (FDD) Model (Despa, 2014)	73
Figure 3.2.1: Prototype Methodology (GeeksForGeeks, 2021)	77
Figure 3.3.1: Gantt Chart	88
Figure 4.2.1: Distribution of age among the participants	91
Figure 4.2.2: Distribution of gender among the participants	92
Figure 4.2.3: Distribution of occupations among the participants	92
Figure 4.2.4: Distribution of feature demands on the coffee shop website	93
Figure 4.2.5: Distribution of respondents' opinion on customizable coffee drink option	93
Figure 4.2.6: Distributions of preferable beverage customization options	94
Figure 4.2.7: Distribution in having difficulty to determine the suitability of the coffee products	94
Figure 4.2.8: Distribution of whether customer support in product selection would be helpful	95
Figure 4.2.9: Questions on Preferred Customer Support Assist for Product Selection	95
Figure 4.2.10: Distribution of preferred coffee guiding and educational materials	96
Figure 4.2.11: Distribution of Participants' opinion on the features provided on the coffee shop website	96
Figure 4.5.1: Use Case Diagram	101
Figure 4.7.1: Prototype of Login Page UI	124
Figure 4.7.2: Prototype of Register Page UI	124
Figure 4.7.3: Prototype of Home Page with Search Function Provided UI	125
Figure 4.7.4: Prototype of chatbot integrated UI	125
Figure 4.7.5: Prototype of Shopping cart UI	125

Figure 4.7.6: Prototype of Payment Module – Credit Card UI	126
Figure 4.7.7: Prototype of the Product List – Barista Tools UI	126
Figure 4.7.8: Prototype of Product Information – Barista Tools UI with product recommendations and product ratings and reiews included	127
Figure 4.7.9: Prototype of Guiding Materials – Brewing Guides UI	127
Figure 4.7.10: Prototype of Prooduct Management by Admin UI	128
Figure 5.2.1: System Architecture Design	130
Figure 5.3.1: Chatbot Data Embedding Interaction Architecture	132
Figure 5.4.1: Chatbot Question and Answering Interaction Architecture	133
Figure 5.5.1: Database Architecture Design	134
Figure 5.6.1: Context Diagram	136
Figure 5.6.2: Level 1 Data Flow Diagram	137
Figure 5.6.3: Activity Diagram – Register Account	138
Figure 5.6.4: Activity Diagram – Login Account	139
Figure 5.6.5: Activity Diagram – View Order	140
Figure 5.6.6: Activity Diagram – Make Payment	141
Figure 5.6.7: Activity Diagram – Search and Filter Product	142
Figure 5.6.8: Activity Diagram – Create Product	143
Figure 5.6.9: Activity Diagram – Edit Product	144
Figure 5.6.10: Activity Diagram – Initiate Chat	145
Figure 5.6.11: Activity Diagram – Add New Embedding File	146
Figure 5.6.12: Activity Diagram – Edit Embedding File	147
Figure 5.6.13: Activity Diagram – Create Guiding	148
Figure 5.6.14: Activity Diagram – Edit Guiding	149
Figure 5.6.15: Activity Diagram – Add To Cart	150

Figure 5.6.16: Activity Diagram – View Cart	151
Figure 5.6.17: Activity Diagram – Update Cart	152
Figure 5.6.18: Activity Diagram – Delete Cart	153
Figure 5.6.19: Activity Diagram – Write Rating and Review	154
Figure 5.7.1: Register Page	155
Figure 5.7.2: Responsive Register Page	156
Figure 5.7.3: Login Page	156
Figure 5.7.4: Responsive Login Page	157
Figure 5.7.5: Home Page	157
Figure 5.7.6: Responsive Home Page	158
Figure 5.7.7: Beverage List Page	158
Figure 5.7.8: Responsive Beverage List Page	159
Figure 5.7.9: Homebrew Product List Page	159
Figure 5.7.10: Responsive Homebrew Product List Page	160
Figure 5.7.11: Beverage Product Detail Page (Part 1)	160
Figure 5.7.12: Beverage Product Detail Page (Part 2)	161
Figure 5.7.13: Beverage Product Detail Page (Part 3)	161
Figure 5.7.14: Responsive Beverage Product Detail Page (Part 1)	162
Figure 5.7.15: Responsive Beverage Product Detail Page (Part 2)	162
Figure 5.7.16: Responsive Beverage Product Detail Page (Part 3)	163
Figure 5.7.17: Write Review Modal	163
Figure 5.7.18: Responsive Write Review Modal	164
Figure 5.7.19: Shopping Cart Modal	164
Figure 5.7.20: Shopping Cart Page	165
Figure 5.7.21: Responsive Shopping Cart Page	165

Figure 5.7.22: Admin Panel Page	166
Figure 5.7.23: User Order List Page	166
Figure 5.7.24: Responsive User Order List Page	167
Figure 5.7.25: Recipe Guiding List Page	167
Figure 5.7.26: Responsive Recipe Guiding List Page	168
Figure 5.7.27: Recipe Guiding Detail Page	168
Figure 5.7.28: Responsive Recipe Guiding Detail Page	169
Figure 5.7.29: Brewing Guide List Page	169
Figure 5.7.30: Responsive Brewing Guide List Page	170
Figure 5.7.31: Brewing Guide Detail Page	170
Figure 5.7.32: Responsive Brewing Guide Detail Page	171
Figure 5.7.33: Chatbot Pop Up Window	171
Figure 5.7.34: Responsive Chatbot Pop up Window	172
Figure 6.1.1: Defining Admin Guards	173
Figure 6.1.2: Filament Guards Configuration	174
Figure 6.1.3: Attempting Login with Admin Credentials on Normal User Login Form	174
Figure 6.1.4: Attempting Login with User Credentials on Admin Login Form	175
Figure 6.1.5: Attempting Register with Existing User Credentials on User Register Form	175
Figure 6.2.1: Product List with Price Range Filter Applied	176
Figure 6.2.2: Product List with Category Filter Add On	176
Figure 6.2.3: Product List with Search Filter using keywords of “caramel” Add On	176
Figure 6.2.4: Code Implementation of the Filters	177
Figure 6.2.5: Add To Cart Button When Hover on the Price	178

Figure 6.3.1: Product Information Provided in Product Detail Page	178
Figure 6.4.1: List of Beverage Customization Options	179
Figure 6.4.2: Customization Choices with Clearly Stated Prices	179
Figure 6.4.3: Display of Beverage Cart: Total Price and Customization Overview	180
Figure 6.5.1: Sample of General Product Recommendations	180
Figure 6.6.1: Product Rating and Review Code Implementation	181
Figure 6.6.2: Sample of Product Rating and Review List	182
Figure 6.6.3: Sample View of Authenticated User's Product Rating and Review List	182
Figure 6.6.4: Product Rating and Review Form Modal	183
Figure 6.7.1: Data Embedding File Code Implementation (Part 1)	183
Figure 6.7.2: Data Embedding File Code Implementation (Part 2)	184
Figure 6.7.3: Data Embedding File Code Implementation (Part 3)	185
Figure 6.7.4: Chatbot Question and Answering Code Implementation	185
Figure 6.7.5: Chatbot Question and Answering Code Implementation	186
Figure 6.7.6: Sample Chat of Asking Product Recommendations Question	187
Figure 6.7.7: Sample Chat of Asking Customer Support Question	187
Figure 6.7.8: Sample Chat of Asking Unrelated Question	188
Figure 6.8.1: Sample of Product Information Data in Excel File	188
Figure 6.8.2: FAQ from The Coffee Bean & Tea Leaf	189
Figure 6.8.3: FAQ from ZUS Coffee	189
Figure 6.8.4: Sample of FAQ data for Embedding File Preparation	189
Figure 6.9.1: Product Cart	190
Figure 6.9.2: Beverage Cart	190
Figure 6.9.3: Beverage Session Cart Code Implementation	191

Figure 6.9.4: Product Session Cart Code Implementation	192
Figure 6.9.5: Sample of Beverage Cart Session Storage	193
Figure 6.9.6: Sample of Product Cart Session Storage	194
Figure 6.9.7: Beverage Database Cart Code Implementation	195
Figure 6.9.8: Product Database Cart Code Implementation	195
Figure 6.9.9: Prepare Beverage Cart For Transfer Code Implementation	196
Figure 6.9.10: Prepare Product Cart For Transfer Code Implementation	197
Figure 6.9.11: Transfer Beverage Session Cart to Database Code Implementation	198
Figure 6.9.12: Transfer Product Session Cart to Database Code Implementation	199
Figure 6.9.13: Beverage Cart Modal	200
Figure 6.9.14: Product Cart Modal	200
Figure 6.9.15: Beverage Cart Page	201
Figure 6.9.16: Product Cart Page	201
Figure 6.9.17: Add to Cart button in Beverage Product Detail Page	202
Figure 6.9.18: Add to Cart button in the Homebrew Product List Page	202
Figure 6.9.19: Add to Cart button in the Homebrew Product Detail Page	202
Figure 6.9.20: Add to Beverage Cart Code Implementation for Unauthenticated User	203
Figure 6.9.21: Add to Product Cart Code Implementation for Unauthenticated User	204
Figure 6.9.22: Add to Beverage Cart Code Implementation for Authenticated User	205
Figure 6.9.23: Add to Product Cart Code Implementation for Authenticated User	205
Figure 6.9.24: Sample of Increase in Quantity of one Cart Item in Product Cart	206

Figure 6.9.25: AJAX code Implementation to connect to Laravel Controller	207
Figure 6.9.26: Sample Code Implementation for Both Unauthenticated and Authenticated User to Update Cart	208
Figure 6.9.27: Sample of Product Cart with Delete Item Icon	209
Figure 6.9.28: Delete Item from Beverage Cart Code Implementation	209
Figure 6.9.29: Delete Item from Product Cart Code Implementation	210
Figure 6.10.1: Stripe Payment Gateway Code Implementation (Part 1)	211
Figure 6.10.2: Stripe Payment Gateway Code Implementation (Part 2)	211
Figure 6.10.3: Stripe Payment Gateway Code Implementation (Part 3)	212
Figure 6.10.4: Stripe Payment Gateway Code Implementation (Part 4)	212
Figure 6.10.5: Sample of Stripe Payment Page	213
Figure 6.10.6: Payment Failure Page	213
Figure 6.10.7: Payment Success Page	214
Figure 6.10.8: Order Confirmation Email	214
Figure 6.11.1: Order History List	215
Figure 6.12.1: Sample of Creating New Beverage Form	216
Figure 6.12.2: Sample of Beverage List	216
Figure 6.12.3: Sample of Edit Beverage Form	217
Figure 6.12.4: Alert Message for Delete Confirmation	217
Figure 6.12.5: Sample of Basic Information in the Brewing Guide Form	218
Figure 6.12.6: Sample of “More” button to increase the inputs row	218
Figure 6.12.7: Sampil of Brewing Guide List	219
Figure 6.12.8: Sample of Edit Brewing Guide Form	219
Figure 6.12.9: Alert Pop up to Confirm Delete Action	220
Figure 6.12.10: Notification telling admin to delete existing file before adding a new one	221

Figure 6.12.11: Code Snippet of After Successfully Uploaded the File	221
Figure 6.12.12: Alert Message to Reconfirm Delete Action	222
Figure 7.4.1: SUS Standard Template (Brooke, 1995)	248
Figure 7.4.2: SUS Grading Table (Sasmito, Zulfiqar and Nishom, 2019)	249

LIST OF SYMBOLS / ABBREVIATIONS

API	Application Programming Interface
CBTL	The Coffee Bean & Tea Leaf
CRUD	Create, Read, Update and Delete
CSS	Cascading Style Sheets
FDD	Feature Driven Development
FYP	Final Year Project
HTML	Hypertext Markup Language
MVC	Model View Controller
PHP	Hypertet Preprocessor
RAD	Rapid Application Development
SQL	Structured Query Language
UI	User Interface
WBS	Work Breakdown Structure
SUS	System Usability Scal
UAT	User Acceptance Testing
SKU	Stock Keeping Unit

LIST OF APPENDICES

Appendix A: Questionnaire	262
---------------------------	-----

CHAPTER 1

INTRODUCTION

1.1 General Introduction

Coffee has a long and rich history, beginning as a direct trade commodity where farmers sold their beans to buyers or roasters. As coffee consumption grew, coffee shops emerged, which transformed the coffee industry by introducing middlemen who purchased beans from farmers and sold them to buyers or roasters. Today, coffee is one of the most popular beverages worldwide, with an estimated 1.6 billion cups consumed daily. Studies have also shown that coffee consumption may have various health benefits, such as reducing the risk of chronic diseases like type 2 diabetes, Parkinson's disease, coronary artery disease, and stroke (Dirks-Naylor, 2015). With such widespread consumption and potential health benefits, coffee's impact on human life is undeniable.

Furthermore, the rise of Industry 4.0 has led to the rising of a new trend in buying and selling, namely e-commerce (Murdiana and Hajaoui, 2020). This trend gained momentum during the COVID-19 pandemic in Malaysia, when the Movement Control Order (MCO) was implemented. Nonessential businesses and stores were ordered to suspend operations, and strict rules for social distancing were put in place. The resulting changes in people's lifestyles and spending habits led to a surge in online shopping, Malaysians became more willing to purchase items through e-commerce platforms. Research studies has also shown that there is a drastic increase in the number of inactive and new users on the usage of online retail web applications as well as the mobile applications. This has indicates the demands and new opportunities for e-business (Lee et al., 2020).

To meet this demand, coffee shop websites were introduced as platforms for selling coffee products. As mentioned earlier, coffee plays a significant role in people's lives, and with the implementation of the MCO, physical coffee shops were closed, leading to a surge in online ordering. However, issues arise when users purchase items that do not meet their expectations, and when they try to reach out to the responsible party online, they

often fail to do so as most of the coffee shop websites does not offer a proper customer support. Additionally, users may also struggle with decision making when faced with a wide variety of products available on different coffee websites.

Thus, in this project, a user-friendly web-based platform has been developed to function as a coffee shop. The platform offers a wide range of products, including beverages, coffee products, and barista tools. To further enhance customer support, a chatbot feature has been integrated into the website. The chatbot is designed to answer users' questions and provide personalized product recommendations based on their interests.

1.2 Problem Statement

In this section, 4 problem statements is defined based on a review of the existing web application, which includes limited customer support feature for users, absence of personalized recommendations, limited product-related information provided for users, and lack of all-inclusive coffee-related rproducts website.

i. Lack of Customer Support Feature for Users

Customer support is a crucial success factor for social commerce, as it directly impacts users' purchase decisions, behavioral intentions, and company performance. Consumers often hope of gaining the maximum benefits such as more extensive knowledge, higher reputations, and social and economic returns in a minimum effort or cost needed (Molinillo, Anaya-Sánchez and Liébana-Cabanillas, 2020). Therefore, providing customer support is essential for creating a positive user experience and driving customer loyalty. However, a comparison of the four reviewed websites reveals that most of them lack complete customer support features. For instance, only Seattle Coffee Gear offers live chat support during working hours only, leaving users without assistance after business hours. Others fail to provide users with an avenue to seek help when they have doubts or concerns. The absence of customer support could lead to worse user experiences and lower satisfaction levels, which, in the long run, may damage the trust relationship necessary for building customer loyalty. Therefore, incorporating comprehensive chatbot customer support

features on social commerce websites is critical to enhancing the user experience, fostering customer loyalty, and driving business success.

ii. The Absence of Personalized Product Recommendations Makes Finding the Perfect Cup of Coffee Difficult for Users

Personalization is a key strategy that could provide significant benefits for businesses, including the ability to deliver precise and timely information to users, leading to increased sales. Among various methods of personalization, product recommendations are widely regarded as the most effective (Senecal and Nantel, 2004). As such, implementing a personalized product recommendation feature on a website has become increasingly important. However, most product recommendation features, as reviewed in chapter 2, are limited to only suggesting a few related products when users view the details of a specific item. This approach could lead to users being presented with a wide range of options that may not be tailored to their individual needs and interests, resulting in a negative user experience and potential reputational damage as well as drop of sales for the website over the time. Therefore, a more sophisticated approach to product recommendations that truly enhances the customer experience and drives sales growth while maintaining the website's reputation.

iii. Limited Product-Related Information Provided for Users

Providing comprehensive product-related information on a website is vital to enable users to make informed purchasing decisions, reduce their search time, and ultimately enhance their satisfaction when shopping online. In addition, such information could help to attract and engage customers, building a stronger connection between them and the company's products or services (Salehi et al., 2012). Therefore, it is critical for websites to prioritize providing adequate and detailed product information to improve the user experience, drive customer loyalty, and boost business success. However, a comparison of the 4 reviewed websites reveals that some websites lack essential information and customer support. For example, CBTL does not provide any guides for users, while Noc only offers its own product roastery guides. In contrast, Starbucks and Seattle Coffee Gear offer a wide range of coffee knowledge. The absence of

comprehensive product information may lead to user doubts and hesitation when making a purchase, which could result in lower profits for the website in the long run. To address this issue, the coffee shop website will focus on providing sufficient product information to improve the user experience, with the extent of building customer loyalty and driving business success.

iv. Lack of all-inclusive coffee-related products website

Product variety is a measure of the number of different products offered by a company to its customers, which may differ in terms of characteristics, taste, formulation, and other factors. The availability of a range of products is often seen as a greater competitiveness, as it relates with a company's strategies of fulfilling customer expectations and boosting sales results (Santos, Sampaio and Alliprandini, 2020). By offering a diverse range of products, companies could appeal to a wider audience and cater to different customer needs and preferences, ultimately resulting in increased customer satisfaction and loyalty. However, by referring to the 4 reviewed coffee websites, it is seen that there is lack of all-inclusive coffee-related products in the current society. For example, all of the websites only includes either beverage and coffee products or coffee products and barista tools, there are no website that included all beverages, coffee product, and barista tools in one website. This could result in decreased customer satisfaction and loyalty, as customers may become frustrated with the fragmented shopping experience due to less product varieties. Therefore, developing a web-based coffee shop that offers a wide range of coffee-related products, including beverages, coffee products, and barista tools, could provide a competitive advantage by catering to the needs and preferences of a broader audience.

1.3 Project Objectives

1. To identify the features and functions to be implemented into the website as well as the chatbot.
2. To develop a web-based coffee shop that allows users to make purchase of coffee-related products, tools, and beverages.

3. To implement a chatbot into the proposed website to enhance the user experience by providing a personalized product recommendation and customer support.

1.4 Proposed Solution

By referring to the problem statements, a web-based coffee shop is developed to offers customers a comprehensive selection of coffee products, beverages, and barista tools for making coffee at home. The website will also feature comprehensive product information, including details on the taste, characteristics, brewing methods, as well as helpful information on coffee-related topics. Additionally, a chatbot feature is implemented to provide personalized recommendations for individual customers and serve as a 24-hour customer support system for addressing any questions or concerns users may have.

The system architecture of the web-based coffee shop is depicted in the figure below. The user interface is created using HTML, CSS, JavaScript, and Bootstrap. One of the key features of the interface is a chatbot, which is powered by the OpenAI API. The chatbot processes and responds to user input messages or questions. When a message or question is received, the API generates a response based on the user input, and the front-end forwards the request to the backend framework based on the user's interactions. To ensure a seamless user experience and continuity, all conversations between the chatbot and the user are stored in a database. These conversations could be easily retrieved when a user reloads the page, allowing them to pick up where they left off.

Laravel is utilized as the backend framework in this project. It is one of the most widely used PHP frameworks that provides pre-existing codes for the overall structure, making it easier to modify and customize the features based on specific project requirements (Yadav, Rajpoot and Dhakad, 2019). When a request is forwarded to Laravel, the type of request will be identified based on the Create, Read, Update, and Delete (CRUD) operations, and the corresponding request will be passed to the database.

The database used in this project is MySQL, which is a high-performance relational database management system that provides a wide range

of unique features such as speed, reliability, scalability, and usability (Vaswani, 2009). To run the database, PhpMyAdmin will be utilized in conjunction with WampServer. The database will receives and handles the request and then return the response back to Laravel with a message indicating whether the operation was completed successfully or not. Finally, the message will be passed back to the front-end to notify the user of the success or failure of their operation.

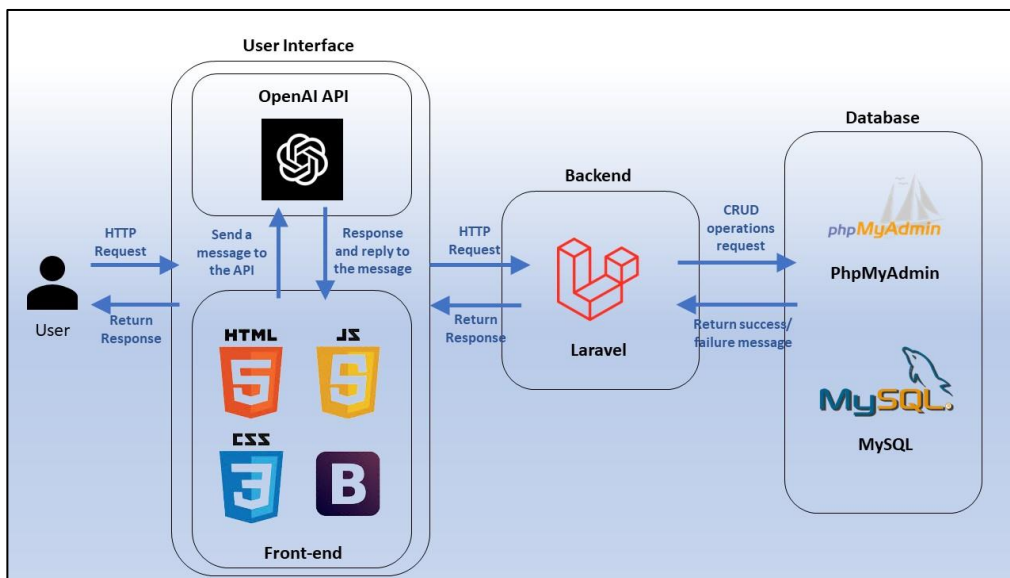


Figure 1.4.1: System Architecture

1.5 Proposed Approach

In this project, the prototyping methodology is implemented as the development methodology to develop the web-based coffee shop with chatbot integrated. The figure below shows each phase included in the prototyping methodology.

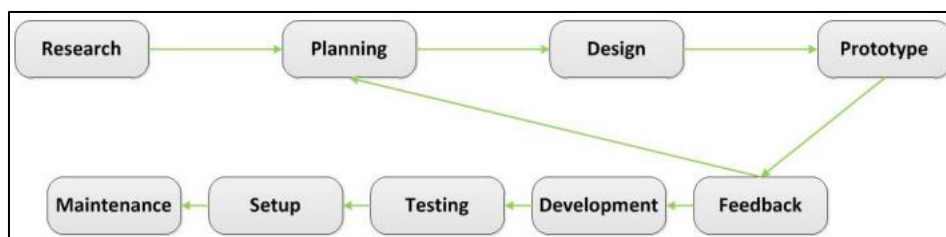


Figure 1.5.1: Prototyping Methodology

Prototyping methodology is an iterative approach to software development that emphasizes building small, functional prototypes of a product

or system early in the development process, and then refining it through multiple iterations. In the planning and design phases of the methodology, only essential functions and features necessary for a web-based coffee shop is implemented to produce the first prototype. Once the initial prototype is built, it is given to the customer to gather feedback and comments. Based on these feedback, the second iteration is initiated to improve the prototype further. The iterations is repeated until customers are satisfied with the subsequent prototype. The project then enters the development phase based on the final prototype. The prototyping methodology allows for early feedback and promotes collaboration between the development team and customers, leading to improved customer satisfaction and a better end product.

1.6 Project Scope

This section defines the project scope, including the targeted end users (customers and staffs) and the modules to be covered. The modules are categorized as main and basic. The main modules include product filtering, product purchasing, product information, product recommendation, customer reviews, product searching, guidance and educational materials, and chatbot while basic modules would be login module and register module. The project duration is 6 months.

1.6.1 Targeted Users

i. Customers

One of the targeted users for this project is the customers of young adults aged between 18 to 40 years old in Malaysia, specifically university students and working professionals. These demographic rely heavily on coffee to stay alert and focused throughout the day, as they often have tight schedules and demanding responsibilities. University students may require coffee to stay awake during lecture classes and study sessions, while working professionals may need it to start their day and stay productive during long working hours.

ii. Staffs

Another targeted user for this project is the staff, who is an administrators for the coffee shop website. The admin staff is responsible in managing the website in terms of product sales, changing the product information, updating stock inventory, and other backend operations.

1.6.2 Modules Covered

1.6.2.1 Main Modules (Customers)

i. Product Purchasing

The product purchasing module allows users to purchase a wide range of products which includes a wide range of coffee-related products, such as beverages, barista tools, and other coffee products through online. Users could also add their desired products into their shopping cart.

ii. Product Information

The product information module displays detailed information about each product, such as the description, price, ratings and reviews and others.

iii. Product Recommendation

The product recommendation module suggests related products to users while they're viewing a product's information page. By clicking on the suggestions, users could easily navigate to the recommended products' information pages.

iv. Product ratings and reviews

The product ratings and reviews module displays the product's overall ratings as rated by other users. Users could also read and write reviews on the product, providing valuable feedback and insights for other potential buyers.

v. Product Searching

The product searching module allows users to easily search for products by typing in relevant keywords in the searching box provided. The website will then display a list of products related to the keywords entered, helping users quickly find the products they're looking for.

vi. Product Filtering

The product filtering module enhances user experience by offering two distinct filtering options on the website. Users can refine their search through price filtering, specifying minimum and maximum price ranges, or opt for product categories filtering. Additionally, the seamless removal of filters is facilitated with the convenient "Reset" button.

vii. Guidance and Educational Materials

The guidance and educational materials module provides users with valuable information about coffee-related topics, including brewing guides, beverage recipes, and more. Users could easily access this information on the website and relate it to the products they purchase, enhancing their overall coffee experience.

viii. Chatbot Integrated

The chatbot module offers users the ability to ask questions and get answers when they have doubts or need assistance. Additionally, the chatbot provides personalized recommendations based on the user's interested products and criteria. Users could list out their desired products with specific criteria and the chatbot will generate a list of product recommendations tailored to their preferences.

1.6.2.2 Main Modules (Staffs)

i. Managing Products

The staff of the coffee shop website has the privilege to manage their products efficiently using the Create, Read, Update, Delete (CRUD) operations. This feature allows the staff to perform various tasks such as adding new products, viewing the list of available products, managing the inventory of the products, updating the latest information of the products, and removing the products that are no longer needed. By utilizing the CRUD operations, the coffee shop staff could streamline their product management process and ensure that their inventory is always up-to-date.

ii. Managing Guides

The staff of the coffee shop website will have full privileges to perform essential CRUD operations on the guides shared on the website. As administrators, they can easily manage the content by adding new recipes or brewing guides with the ability to view a comprehensive list of guides that ensures efficient management, make modifications to the guides if there are any errors found, and lastly remove outdated or obsolete guides, keeping the website's content fresh.

iii. Managing Data Embedding File

The staff of the website will have the needs and rights to manage on the data embedding file that serves as the chatbot context purpose, thus, it is required for the admin to ensure that the data in the file is always up to date in order to let guarantee customers' user experience when approaching to the chatbot.

1.6.2.3 Basic Modules

i. Login Module

The login module enables both admins and normal users to store their credentials, including delivery address, phone number, and email address. Additionally, it also allows users to preserve their chatbot conversations.

ii. Register Module

The register module only allows first time users to register their accounts to enjoy some extra features of owning an account. Users are required to enter some basic credentials such as delivery address, phone numbers, email address, password and more. Admins are not allowed to register for a new account.

1.6.2.4 Out of Scope

In addition, any delivery-related functions are not included in this web-based coffee shop. This means that after users made their payment, they will not be provided with any delivery tracking information through the website.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In this chapter, a literature review is conducted to gather information and resources on similar existing applications, programming frameworks, and software development methodologies. These reviews aim to identify the most effective approach for developing a coffee shop website that meets the needs of customers. By analyzing these existing literatures, insights into best practices, potential challenges, and opportunities for innovation could be gained. Ultimately, this helped to make better decisions and a high-quality website that provides value to users could be developed.

2.2 Similar Existing Web Applications

In order to develop a web-based coffee shop that meets the society's requirements and expectations, a good understanding of the required domain is essential. Therefore, 4 similar existing websites is reviewed and studied based on their features, functions and services provided in these coffee shop websites in this section. The selected websites that is reviewed in this section are Noc Coffee Company, Starbucks Coffee Comapny, Coffee Seattle Gear, and The Coffee Bean & Tea Leaf. Each of these websites consist of its unique functionalities analysed so that a clearer image could be seen and applied.

2.2.1 Noc Coffee Company

Noc Coffee Company is an artisanal coffee company that curates coffee from the beginning to the end, from firstly sourcing the best beans from all around the world to roasting them and then curating the perfect cup of coffee. With 6 physical retail cafes located specifically in Hong Kong and Bangkok, Thailand. Noc is being recognised with its unique and overall great tasting coffee. On its steps to further expand the brand internationally, Noc also has a well curated website and e-shop which not only provides detailed information of their product but also serves as a platform where international consumers are able to

purchase coffee products from their brand. In this web application, the primary focus of Noc is to offer its services to regular customers, and cater exclusively for this type of user. The web application could be accessed at <https://noccoffee.com/en/>. The following provides an overview of some of the key features of Noc’s website, accompanied by explanations which includes:

- i. Currency and Language Options
- ii. Product Filtering
- iii. Anonymous Online Purchasing
- iv. Product Recommendations
- v. Product Information
- vi. Internal Roasting Process

i. Currency and Language Options

Noc offers the features of currency and language options, allowing users to switch to their preferred currency and language for a better user experience. By referring to Figure 2.3.1, it is seen that Noc website has provided the language options between English, Simplified Chinese, Japanese and also Korean on the left path for users to choose while on the right path, the currency options are provided between Hong Kong Dollars(HKD\$), Korean Won(KRW), and Japanese Yen(JPY). The available options are very limited, and based on Figure 2.3.2, it shows that certain sections of the website may still be displayed in English even after selecting preferable language.

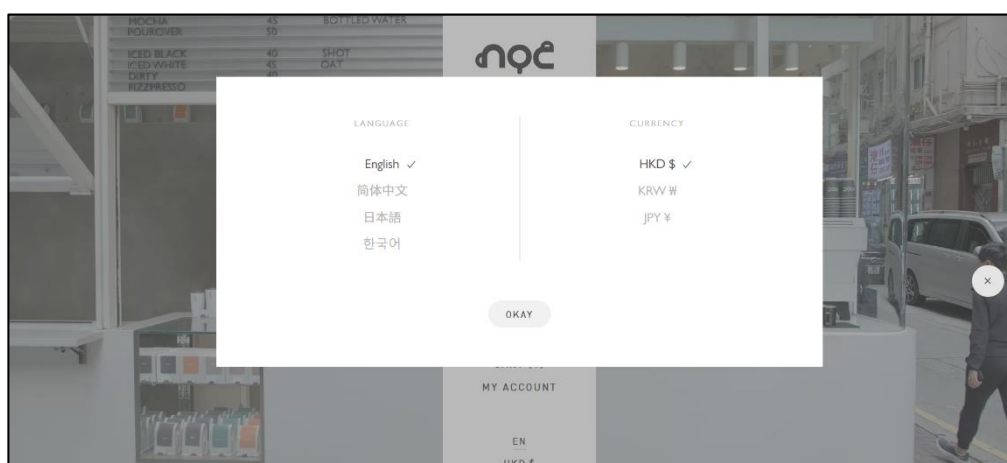


Figure 2.2.1: Currency and Language Options



Figure 2.2.2: A sample of Simplified Chinese applied to the web page

ii. Product Filtering

Noc also provides its users with the ability to filter and sort the product list to help them easily find the products they are interested in. This feature allows users to view the product list based on various product and coffee types. Figure below shows the available filter options between specialty instant coffee, espresso roast, filter roast, coffee capsules, drip bag, gift card, and merchandise.

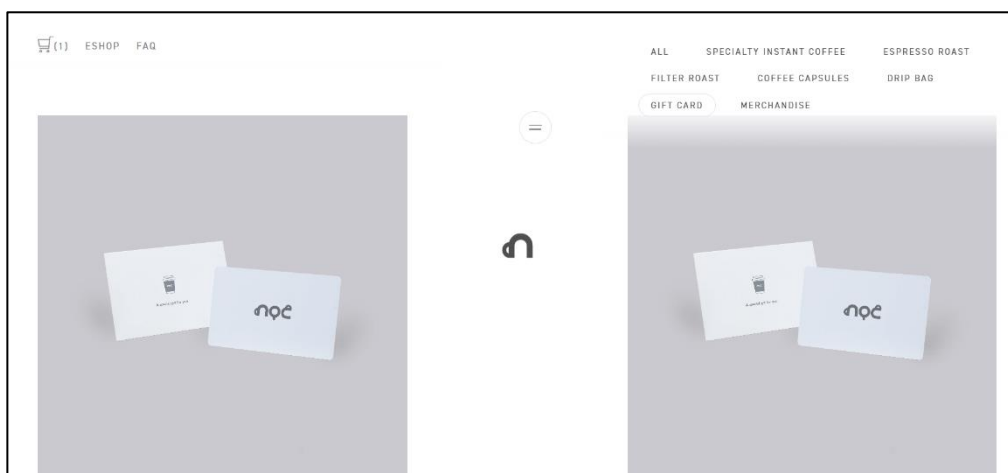


Figure 2.2.3: Product Filtering with Gift Card filter applied

iii. Product Purchasing

Noc's website provides a convenient online purchasing feature, allowing users to purchase coffee without having to visit a physical store. Additionally, Noc

has also incorporated a user-friendly anonymous shopping cart feature that allows users to effortlessly add products to their cart and complete payments without having to register or log in to their account. The contents of the cart will persist even if the user refreshes the website or remains unauthenticated. Figure below shows the sample of the shopping cart of Noc's website.

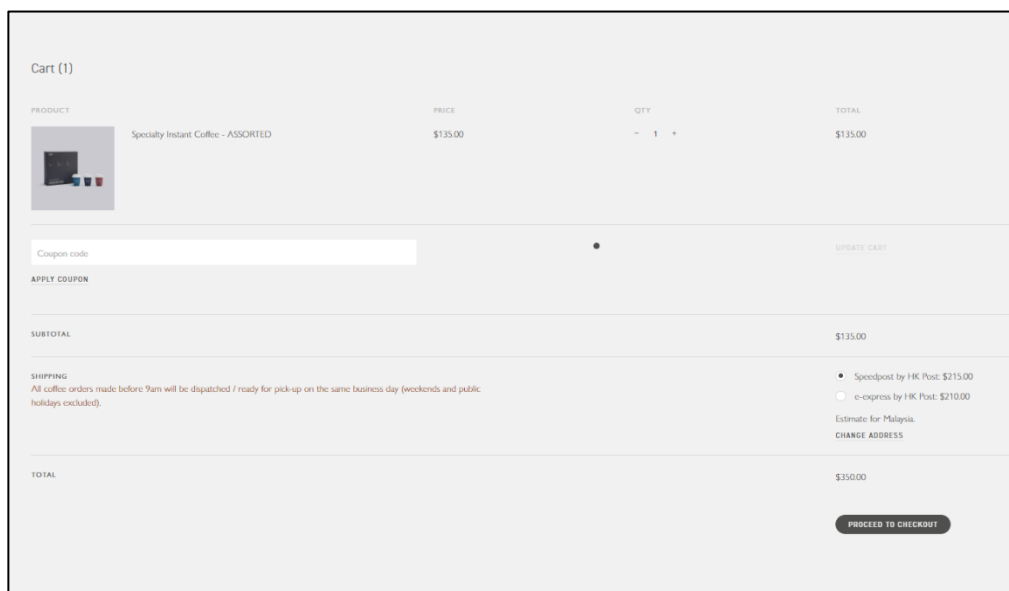


Figure 2.2.4: Shopping Cart page of Noc

iv. Product Recommendations

Noc has implemented a product recommendation feature on their website, which suggests related products to users when they view a specific product's detailed information. For example, when a user views a coffee capsule product, the website will prompt several recommendations on related coffee capsule products, as shown in the figure below.

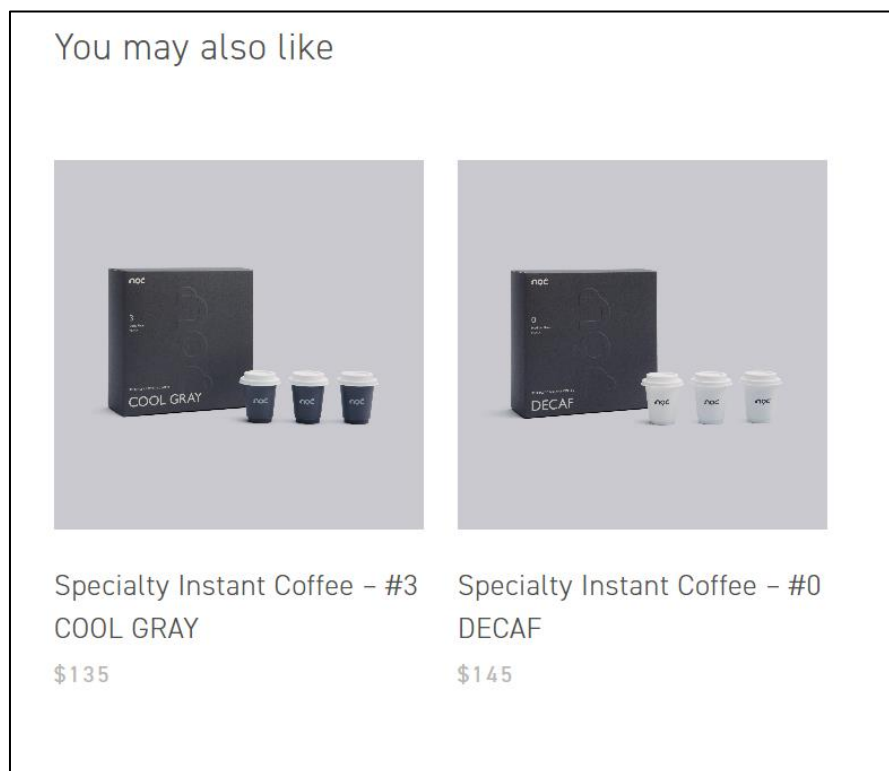


Figure 2.2.5: Coffee capsule recommendations when users view on a coffee capsule product

v. Product Information

Noc provides comprehensive product information in its web application, so that users could familiarize themselves with the products before making a purchase. Figure below shows one of the product information which includes the ideal usage scenarios, different variations based on flavor intensity, and the unique qualities of the product.

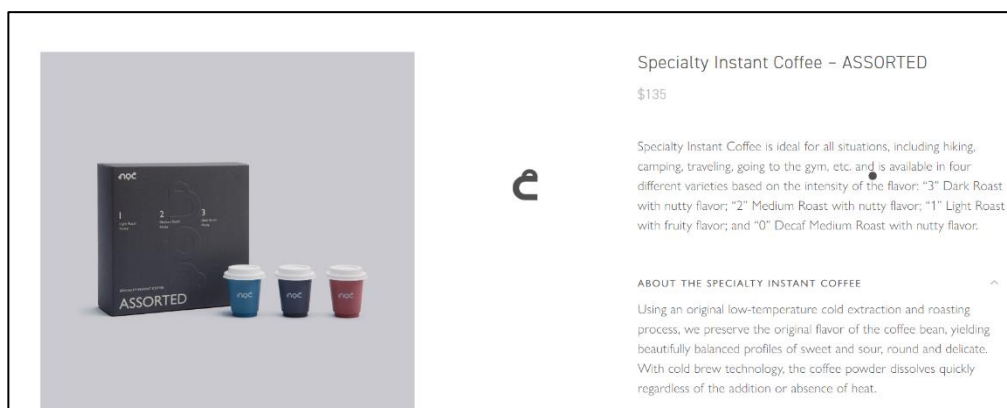
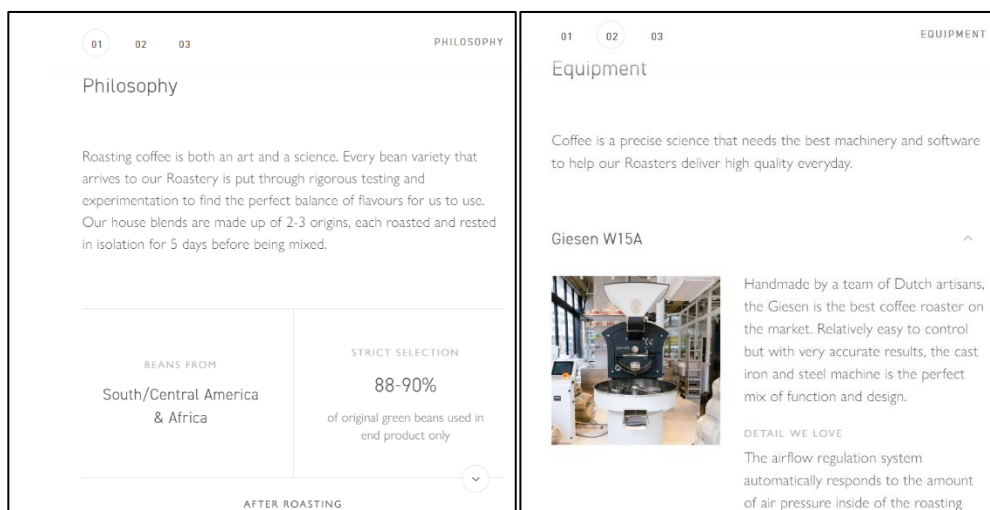


Figure 2.2.6: Product Information Feature of Noc

vi. Internal Roasting Process

Noc also offers additional insights into their coffee roasting process, including their philosophy for selecting coffee beans, the post-roasting procedures, and the specialized equipments used to roast their coffees to perfection as shown in figures below.



(a)

(b)

Figure 2.2.7: Internal Roastery Process where (a) shows the philosophy of coffee bean and (b) shows the roasting equipment used for roasting process

2.2.2 Starbucks Coffee Company

Starbucks is founded in Washington in the year of 1971, by Jerry Baldwin, Zev Siegl, and Gordon Browker. It is a globally known roaster and retailer of specialty coffee. Over the years, Starbucks has solidified its position as one of the leading coffee companies, operating more than 32,000 physical locations across dozen countries (Petruzzello and Bondarenko, 2023). In addition to its brick-and-mortar locations, Starbucks has also established an online presence through its website, which is dedicated to promoting their products to the customers worldwide. It could be accessed at two different URLs, <https://athome.starbucks.com/> (purchasing website) and <https://www.starbucks.com/> (ordering website).

2.2.2.1 Purchasing Website

Starbucks' purchasing website is a platform that primarily focuses on providing a platform for users to purchase coffee products such as coffee beans and creamer to enable users could make their cup of coffee at home. It also serves as a hub for coffee enthusiasts to learn more about coffee materials and brewing guides. Below listed some of the key features of the purchasing website with brief explanations provided:

- i. Product Filtering
- ii. Product Purchasing
- iii. Product Information
- iv. Product Recommendations
- v. Product Review
- vi. Product Searching
- vii. Learning and Guiding Materials

i. Product Filtering

The Starbucks purchasing website offers an advanced product filtering feature that allows users to sort the products with multiple criteria. As shown in the figure below, products are categorized into coffee and non-coffee items at first and could be further filtered based on different format, roast, and origin. In addition to this feature, users could easily clear the applied filters with a single click.

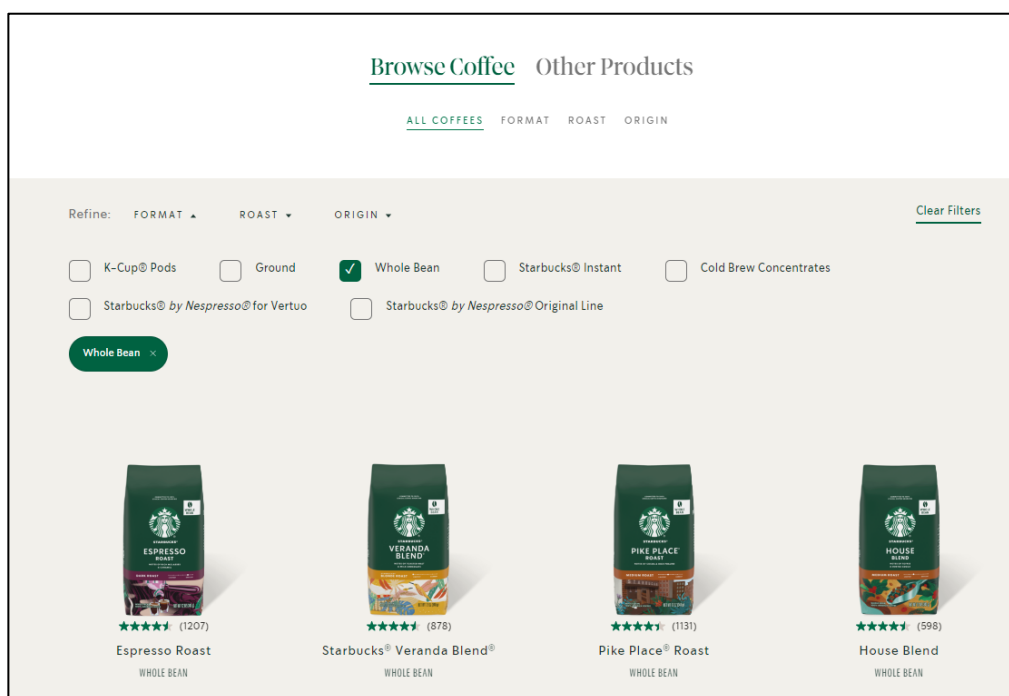


Figure 2.2.8: Product filtering page with Whole Bean filter applied

ii. Product Purchasing

Starbucks' purchasing website provides an indirect sales feature, where it primarily directs users to third-party websites, such as Amazon, Walmart, and Stop&Shop as shown in the figure below, to complete their transactions on the products displayed in the website. Additionally, it also provides a location navigation where users could also purchase physically in local stores.

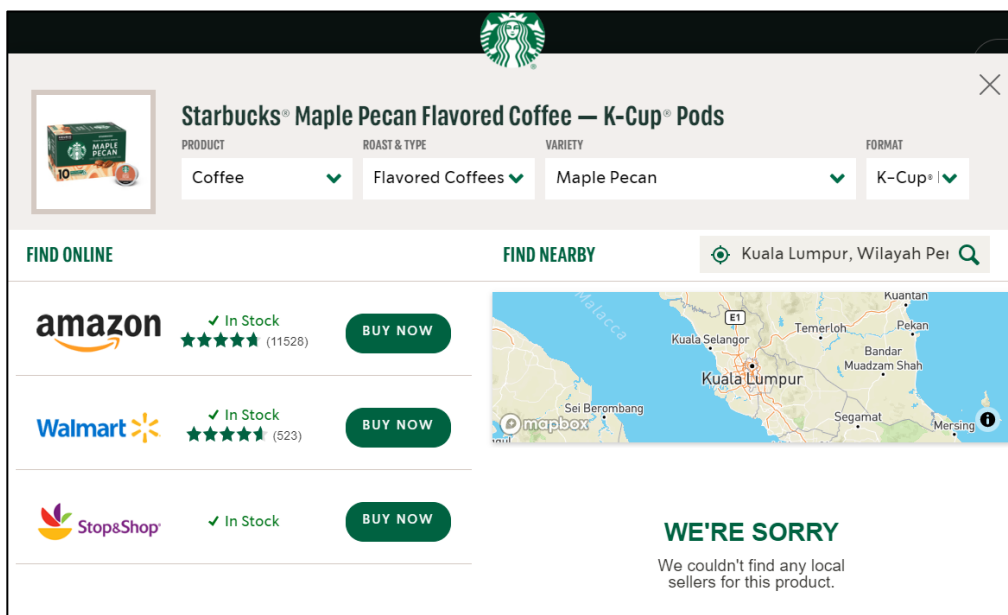


Figure 2.2.9: Product purchasing page with third party websites provided

iii. Product Information

Starbucks' purchasing website offers comprehensive information about each of its products, serving as a valuable reference and resource for users. As shown in the figure below, detailed information such as ingredients are provided for each product, allowing users to make informed decisions about their purchases.



Figure 2.2.10: Product information page of Starbucks' purchasing website

iv. Product Recommendations

On Starbucks' purchasing website, users could find product recommendations when viewing a product's detailed information. The website presents related products for users to consider, as illustrated in the figure below where the Maple Pecan Flavored Coffee prompts a few related products for recommendations.

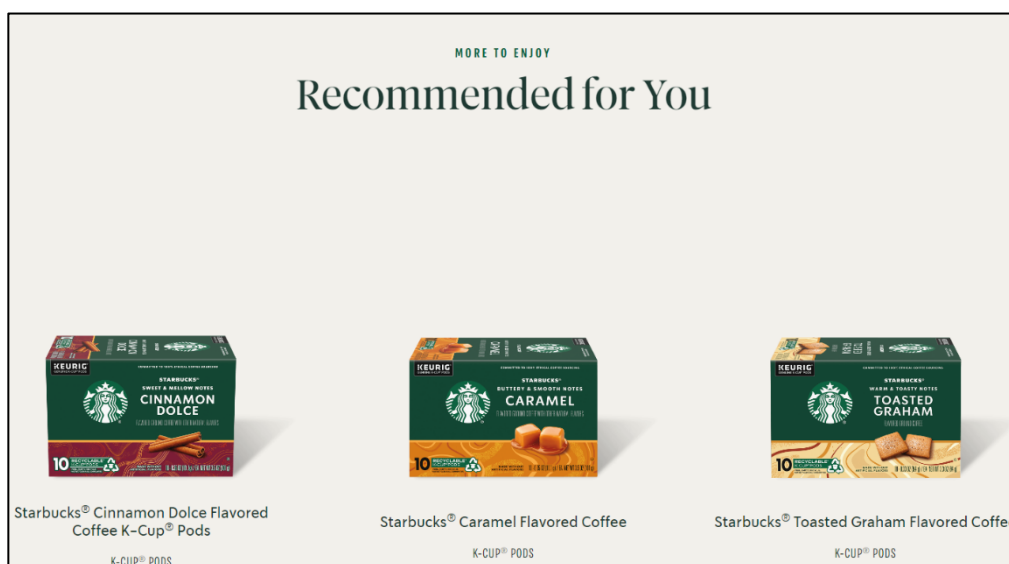



Figure 2.2.11: K-Cup® Pods product recommendations when users view on K-Cup® Pods coffee

v. Product Review

Starbucks has integrated a product review section in its purchasing website, where users could view and share their thoughts and experiences with specific products. As shown in the figures below, the review section includes a rating system, with the ability to filter reviews based on the number of stars given. Additionally, an overall rating for each product is also provided, giving users an idea of the product's performance.

Product review


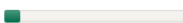
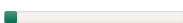

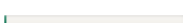
 Authentic Reviews

Reviews


[WRITE A REVIEW](#)

RATING SNAPSHOT


Select a row below to filter reviews.

5 ★		225
4 ★		25
3 ★		19
2 ★		6
1 ★		3

AVERAGE CUSTOMER RATINGS

Overall  4.7

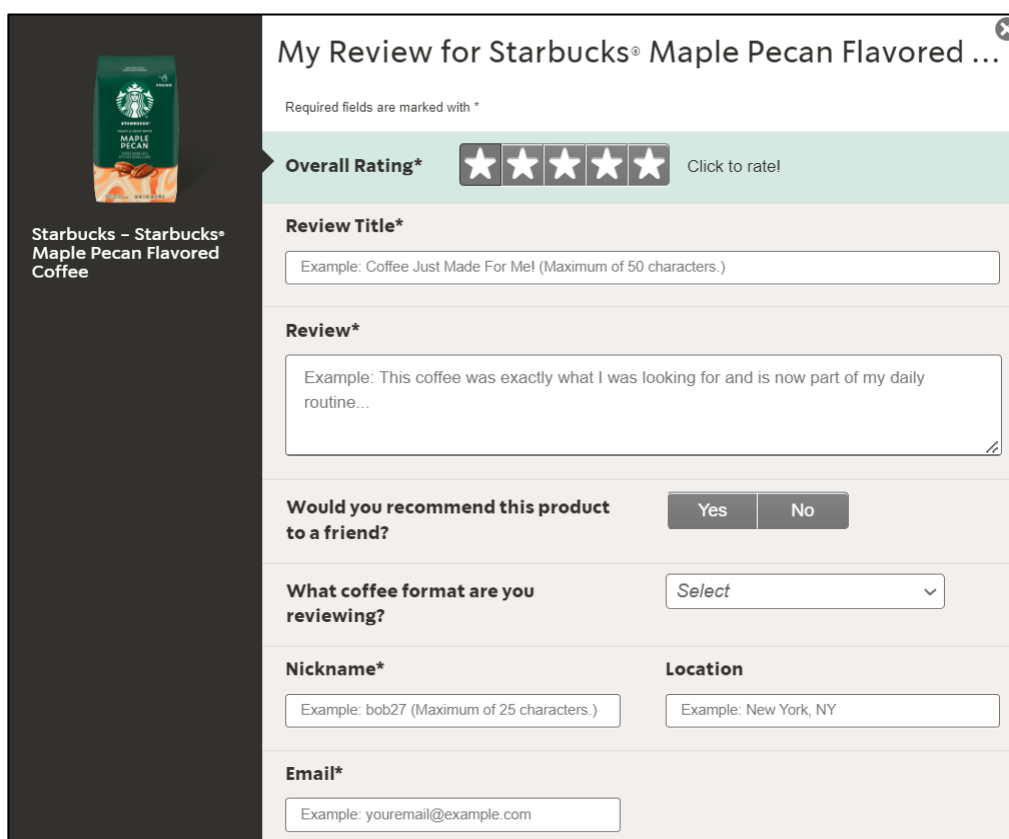
1-8 of 278 Reviews Sort by: Most Recent

 JannieB · 4 months ago

Rich coffee but no maple pecan flavor

Usually drink Green Mountain maple pecan that is full of flavor, so was disappointed this coffee had none.

Figure 2.2.12: Product review sorted with most recent filter



Starbucks - Starbucks® Maple Pecan Flavored Coffee

My Review for Starbucks® Maple Pecan Flavored ...

Required fields are marked with *

Overall Rating* Click to rate!

Review Title*
Example: Coffee Just Made For Me! (Maximum of 50 characters.)

Review*
Example: This coffee was exactly what I was looking for and is now part of my daily routine...

Would you recommend this product to a friend?

What coffee format are you reviewing?

Nickname* **Location**

Email*

Figure 2.2.13: Review writing page of the purchasing website

vi. Product Searching

Starbucks has integrated a product search feature into its purchasing website, allowing users to easily find the products they are looking for. By simply entering relevant keywords, the website will display a list of all related products. For example, when the keyword "Caramel" is entered, all related products are displayed, as shown in the figure below.

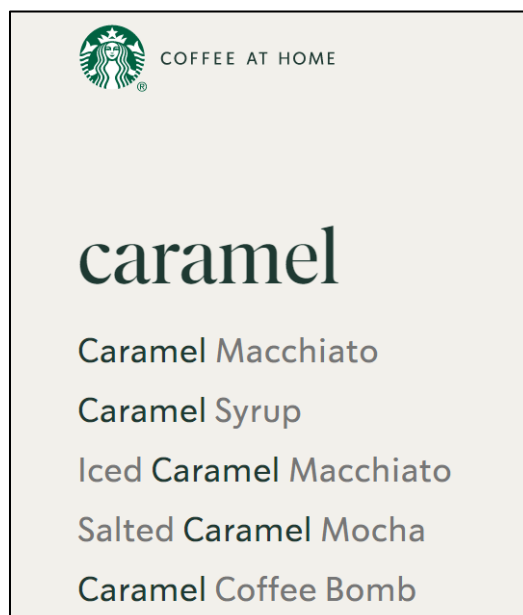


Figure 2.2.14: Product searching in the purchasing website

vii. Learning and Guiding Materials

The Starbucks purchasing website offers a comprehensive and informative hub for coffee lovers, with access to an extensive collection of coffee-related knowledge. For instance, by referring to the figure below, it provides a recipe of one of their bestselling beverages, Caramel Macchiato, in the website.

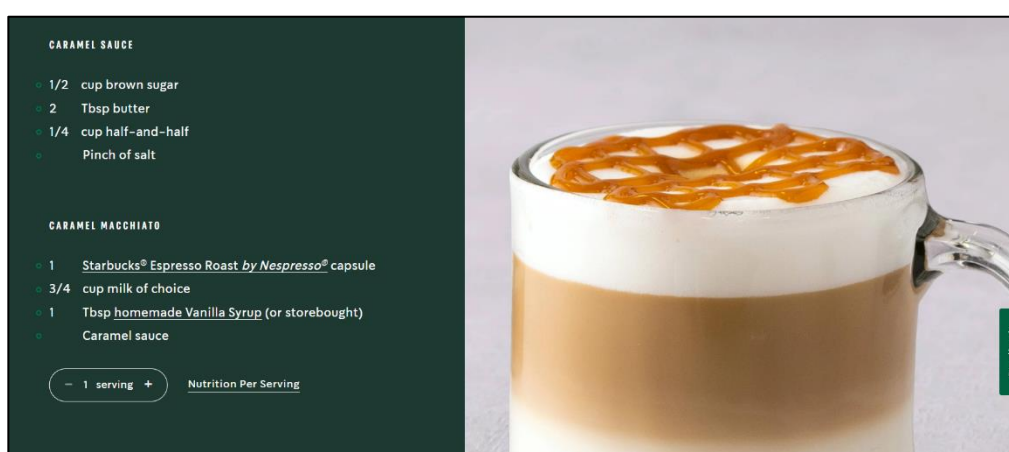


Figure 2.2.15: Recipe of Caramel Macchiato provided as a guiding materials in the purchasing website

2.2.2.2 Ordering Website

The ordering website will focus on providing an online platform for users to easily view and order their desired beverages and food from the nearest franchise. The main features of the website are as below, along with brief explanations:

- i. Product Filtering
- ii. Product Purchasing
- iii. Product Information
- iv. Beverage Customization

i. Product Filtering

The product filtering feature provided on the ordering website is simple yet effective. The website categorizes all products by different types, as shown in the figure below. Users could easily filter products by selecting the desired product type, such as drinks, food, At Home Coffee, merchandise, and gift cards.

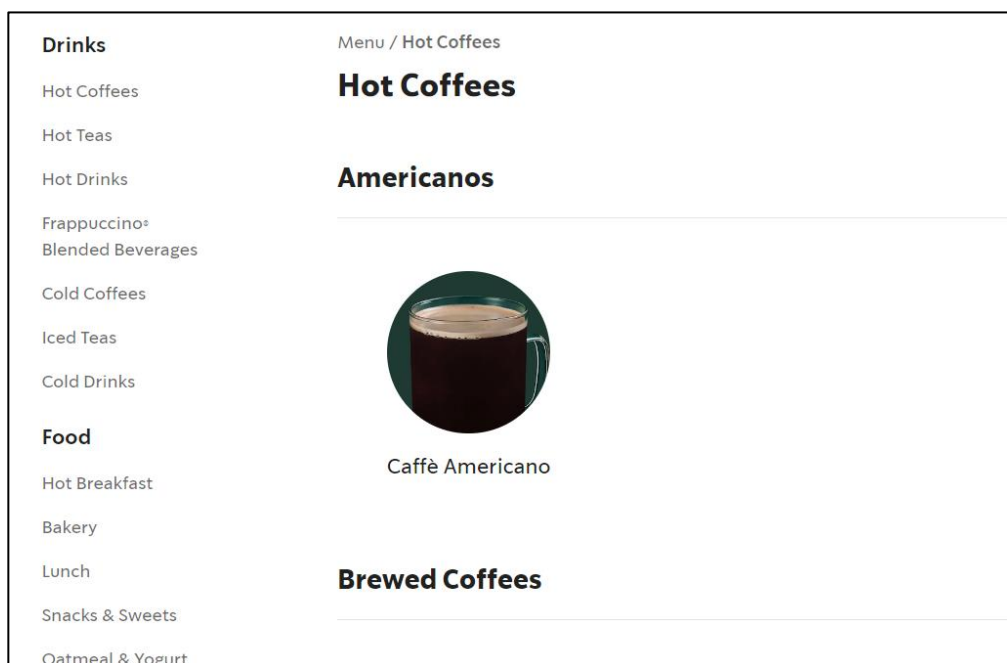


Figure 2.2.16: Product filtering page with Hot Coffee filter applied.

ii. Product Purchasing

Starbucks' ordering website also offers the feature of purchasing their products, which mainly include beverages, food, merchandise, and gift cards. While the website does sell coffee beans, but the range of selections is very limited. Users

could add products to their shopping cart, but they are required to log in and select a pickup store before proceeding to payment.

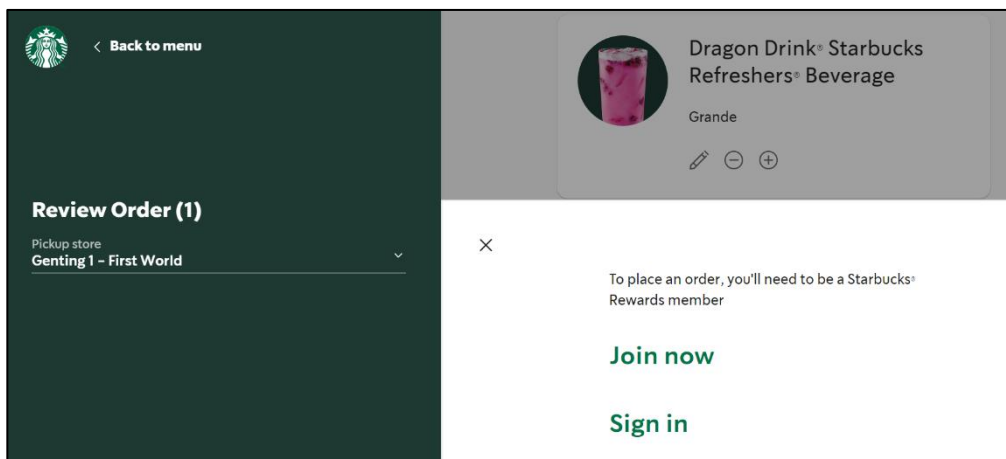


Figure 2.2.17: Shopping cart page of Starbucks

iii. Product Information

The ordering website also provides detailed information for each product for users to refer to. As shown in the figure below, information such as ingredients, nutrition, and allergens are provided for the Caffe Americouldo beverage product.

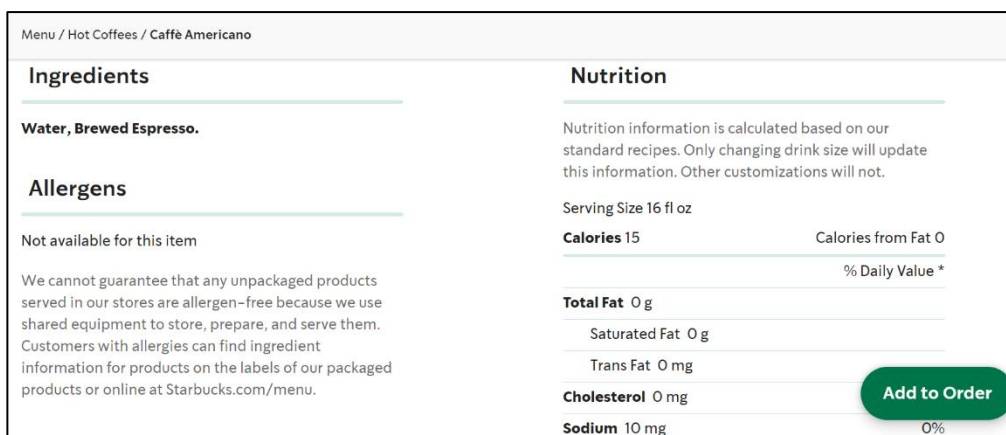


Figure 2.2.18: Product information page of Starbucks' ordering website

iv. Beverage Customization

Starbucks offers a unique feature on its ordering website: the ability to customize beverages to personal taste. By referring to the figure below, users

could tailor their drinks with options such as flavor, toppings, size, tea, sweeteners, and more.

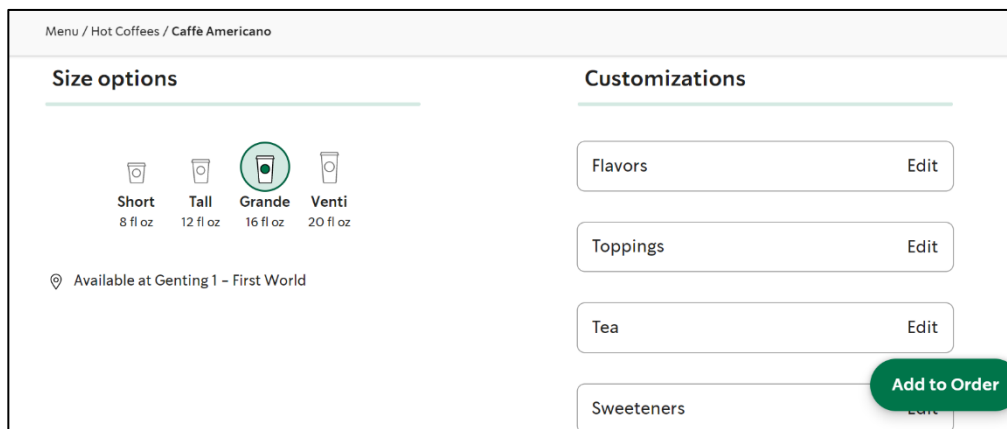


Figure 2.2.19: Beverage customization of Starbucks' ordering website

2.2.3 Seattle Coffee Gear

Seattle Coffee Gear, founded in the year of 2000 in United States, is a leading retailer of coffee equipment and accessories, catering to both personal and commercial users. Adhering to its slogan, "Helping you make coffee you love", the company strives to offer high-quality coffee equipment and expertise to ensure their customers could make the perfect cup of coffee. To expand its reach, the company has launched up its own website, which is designed to showcase their coffee equipment and also their freshly introduced coffee beans to the customers globally. The website could be accessed at <https://www.seattlecoffeegear.com/>. Below listed some of the key features of Seattle Coffee Gear's website along with a simple explanation which includes:

- i. Product Filtering
- ii. Product Information
- iii. Product Purchasing
- iv. Product Searching
- v. Real Time Chat Service
- vi. Product Review
- vii. Learning and Guiding Materials

i. Product Filtering

The Seattle Coffee Gear website offers a comprehensive product filtering feature that enables users to easily search and find products that meet their specific needs. With the ability to sort by product types, price range, bestsellers, and other criteria shown in the figures below, the website provides a user-friendly and efficient way to navigate the wide selection of coffee equipment and accessories.

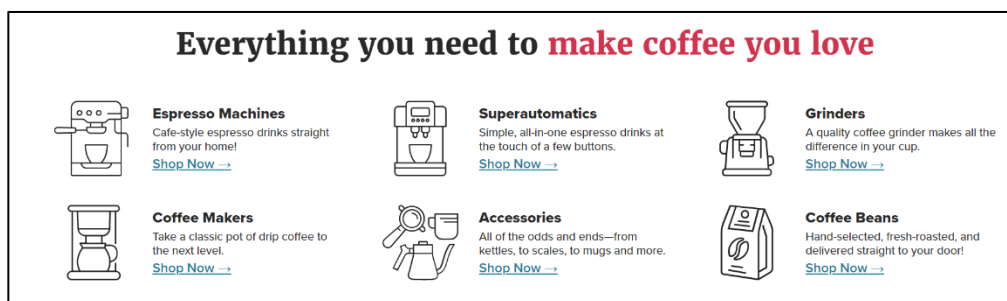


Figure 2.2.20: Product filtering based on various product types

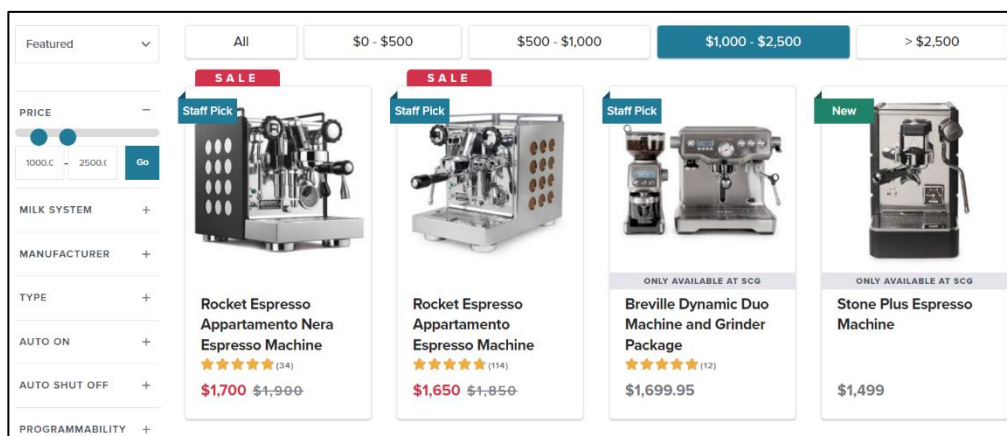


Figure 2.2.21: Product filtering page with price range between \$1000 to \$2000 filter applied

ii. Product Purchasing

The Seattle Coffee Gear website provides a convenient and easy-to-use platform for customers to purchase a diverse range of coffee equipment, accessories, and coffee beans. With a user-friendly interface, customers could effortlessly browse products and add items to their cart before completing their purchase through a secure payment process, either by utilizing a trusted third-party

payment method or the website's own payment options. A sample of a shopping cart in the website are as illustrated in the figure below.

Checkout [Sign In](#) [Edit Cart](#)

Shipping Address
Express Checkout

amazon pay >>
USE YOUR AMAZON ACCOUNT
Are you an Amazon customer? Pay now with address and payment details stored in your Amazon account.

or continue with Standard Checkout
Email Address
jollncwh15@gmail.com
You can create an account after checkout.

Order Summary


	Solis Scala Zero Coffee Grinder QTY: 1 \$0
Cart Subtotal	\$0
Shipping	Not yet calculated
Order Total	\$0

Figure 2.2.22: Shopping cart page of Seattle Coffee Gear


iii. Product Information

The Seattle Coffee Gear website provides an in-depth product description in their website. By referring to Figure 2.3.23, which is a coffee bean product, the detailed information provided includes a brief introduction, roastery level, approachability, region, and more. On the other hand, based on Figure 2.3.24 and 2.3.25, which is the product information of a coffee equipment, the information provided includes an overview, skill level, comparison of similar equipment, and more. With this, it is easier for customers to make informed purchasing decisions and learn how to use their coffee equipment to its full potential.


About Lavazza Super Crema Espresso – Whole Bean – 2.2 lb


Lavazza Super Crema whole bean espresso roast combines washed and unwashed Arabica and washed Robusta coffee beans that originate from Brazil, Central America, and Indonesia. Super Crema is a perennial favorite with a light-to-medium body, delicious hazelnut aromas, and sweet, fruity notes with just a touch of smokiness.

Lavazza Super Crema can transport your taste buds to a plaza in Florence, Italy where you sip an espresso at a chic sidewalk cafe. Pleasant aroma, smooth not bitter, fantastic taste...everything a coffee should be combined into one steaming cup.





ROAST LEVEL
Medium






APPROACHABILITY
Crowd Pleaser





REGION
Multi-Region (Varies)

[Learn about coffee origins →](#)



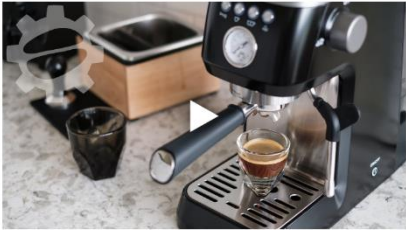
PROCESSING
Washed


[Learn about coffee processing →](#)

Figure 2.2.23: Product information of a coffee bean product


Overview

Perfect your morning coffee routine with the Solis Barista Perfetta Espresso Machine. Built for the home espresso journey, the Perfetta offers an excellent starting point for new baristas without sacrificing the tools more experienced enthusiasts need. This careful design means you get labor-saving niceties like programmable shot times, auto-purge solenoids, and optional pressurized baskets for pre-ground coffee or pods—alongside features like a front-and-center pressure gauge, single hole manual steam wand, and easy access to hot water for tea or Americanos. Even the Perfetta's included accessories carry this attention to detail with just-right sizing and high quality finishes.







SKILL LEVEL
Beginner



PROGRAMMING
Moderate



SIZE
Compact



TEMP CONTROL
Thermoblock

Figure 2.2.24: Product Information of Coffee Equipment




COMPARE SIMILAR MACHINES			
			
	Rocket Espresso Giotto Timer Type V Espresso Machine	Rancilio Silvia PID Espresso Machine	Breville Dual Boiler Espresso Machine
Customer Rating	★★★★★ (10)	★★★★★ (30)	★★★★☆ (31)
Price	\$2,350	As low as \$1,195	As low as \$1,599.95
Auto On	No	No	Yes
Auto Shut Off	No	No	Yes
Boiler Design	Heat Exchanger	Single Boiler	Double Boiler
Boiler Material	Brass/Copper	Brass	Stainless Steel
Boiler Volume	1.8 L	12 ounces	Espresso Boiler: 0.3L (10 ounces) Steam Boiler: 0.95L (32 ounces)
Cup Clearance	3.5 inches	3 inches	4 inches
Cup Tray	Yes	Yes	Yes
Depth	16.5 inches	11.0 inches	15.00 inches
Height	Adjustable 14.75 to 16.75 inches	13.5 inches	15.00 inches
Portafilter Size	58mm	58mm	58mm
Programmability	Temperature	PID, Pre-infusion, Shot Time	PID, Pre-infusion, Shot Time
Reservoir Size	2.5 L	67 ounces	84 ounces
Solenoid Valve	Yes	Yes	Yes
Warm Up: Brew Time	20 minutes	3:30	18 minutes
Water Sources	Reservoir/Internal Tank	Reservoir/Internal Tank	Reservoir/Internal Tank
Width	13.2 inches	9.5 inches	16.25 inches

Figure 2.2.25: Comparison table between similar coffee equipment

iv. Product Searching

The Seattle Coffee Gear website provides a user-friendly product searching feature, allowing users to find the products they are looking for by entering relevant keywords in a shorter time and simpler way. The website then generates a list of all products that match the entered keywords, making it easy for users to find what they need. An example is provided as shown in the figure below, when the keyword of “Caramel” is entered, related products will be prompted.

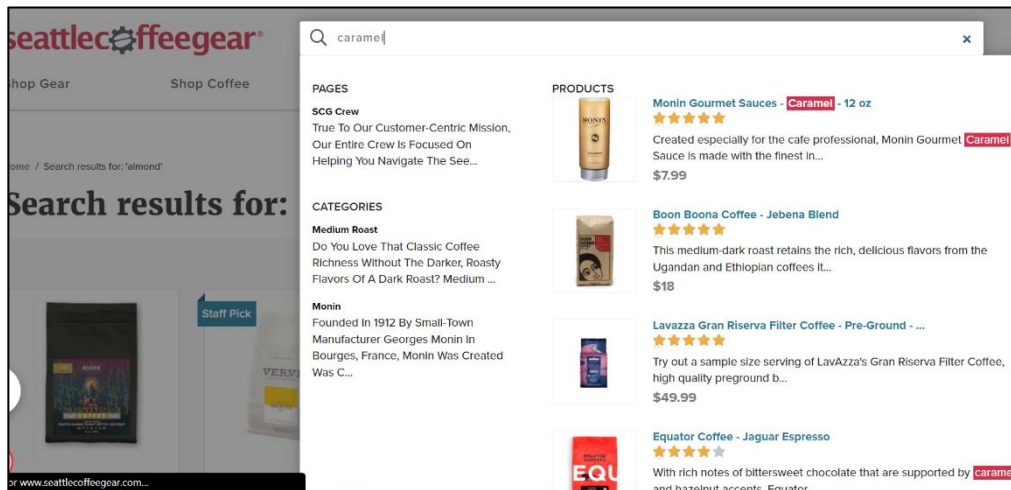


Figure 2.2.26: Product Searching Page of Seattle Coffee Gear

v. Real Time Chat Service

Seattle Coffee Gear has implemented a real-time chat service on its website, accessible through a prominent chat icon. As shown in the figures below, the service is powered by LiveChat, enabling the company's customer support team to offer immediate assistance and support to customers in real-time during working hours. After working hours, users could leave a message, and the team will respond when they are back.

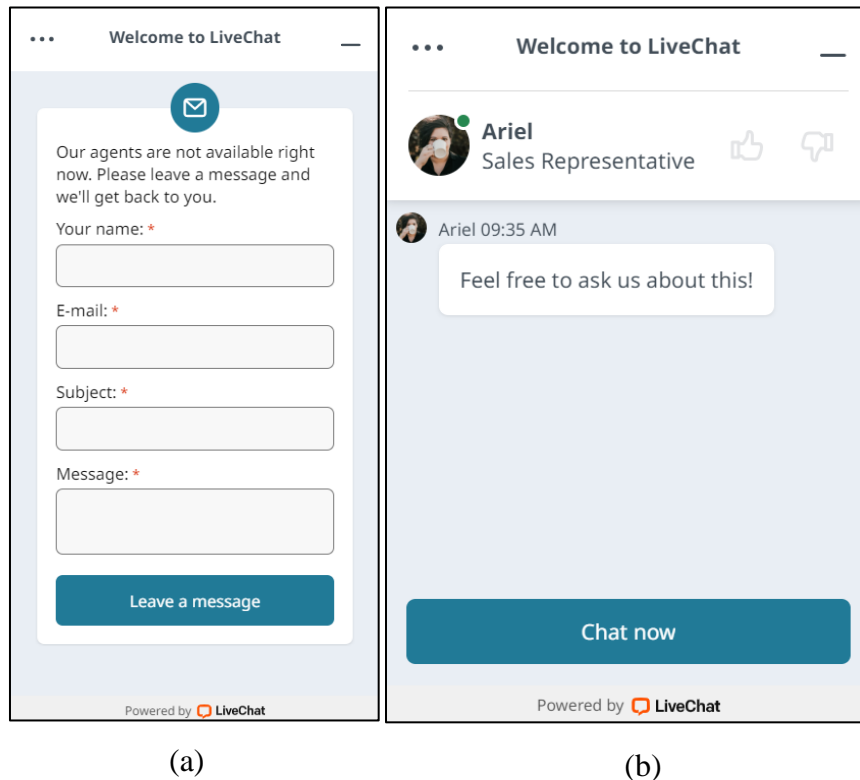


Figure 2.2.27: Real Time Service page of Seattle Coffee Gear where (a) showing agents away and (b) showing agents online

vi. Product Review

The Seattle Coffee Gear website has implemented a comprehensive product review section, providing customers with a platform to share their experiences and provide valuable feedback on the products they have purchased. As shown in the two figures below, users are able to write, sort and view reviews based on star ratings, recency, helpfulness, and more, this section helps to provide useful information to customers looking to make future purchases and helps Seattle Coffee Gear improve its offerings.

REVIEWS

114 Ratings

4.7 out of 5 stars

★★★★★

WRITE A REVIEW

114 Ratings

★ 5	85%
★ 4	9%
★ 3	2%
★ 2	2%
★ 1	3%

1-10 of 114 Reviews

SORT Most Recent

Q Search reviews...

FILTER Star Ratings

★★★★★

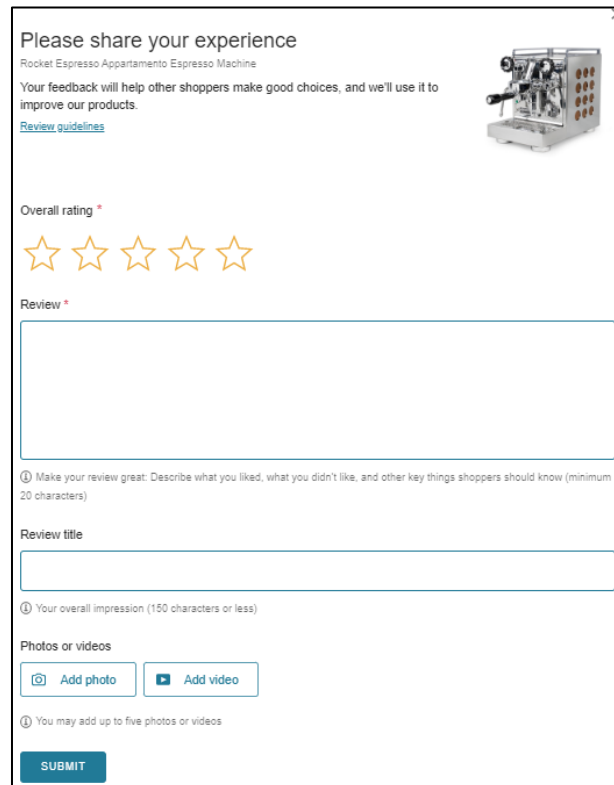
Excellent machine!

Purchased on Dec 19, 2022

The Appartamento is hands down the best espresso machine I have ever owned. My previous daily use machine was a LaPavoni Europiccola purchased around 2001. The LaPavoni had a steep learning curve and I owe most of what I know about grind and pressure to that machine. However, the Appartamento was a breath of fresh air. Right out of the box, I was "pulling" better shots than I ever achieved from the LaPavoni. This thing is built like a Maserati! It is imperative that the machine reach... [Read More](#)

Frederick W. **VERIFIED PURCHASER**

Figure 2.2.28: Product review page with Most Recent and Star Ratings filters applied



Please share your experience

Rocket Espresso Appartamento Espresso Machine

Your feedback will help other shoppers make good choices, and we'll use it to improve our products.

[Review guidelines](#)

Overall rating *

☆☆☆☆☆



Review *

ⓘ Make your review great. Describe what you liked, what you didn't like, and other key things shoppers should know (minimum 20 characters)

Review title

ⓘ Your overall impression (150 characters or less)

Photos or videos

 Add photo  Add video

ⓘ You may add up to five photos or videos

SUBMIT

Figure 2.2.29: Review writing page of Seattle Coffee Gear

viii. Learning and Guiding Materials

The Seattle Coffee Gear website features a comprehensive coffee resource center, offering a wide range of information to help users find their perfect cup of coffee. The resource center is divided into different categories, including recipes, coffee types, coffee culture, education, coffee history, news, reviews, and the company's own podcast. This information provides a step-by-step guide to coffee enthusiasts, giving them the knowledge they need to make an informed decision and enjoy the pleasure. For example, the figure below shows a recipe for Chocolate Covered Strawberry Latte provided by the website.

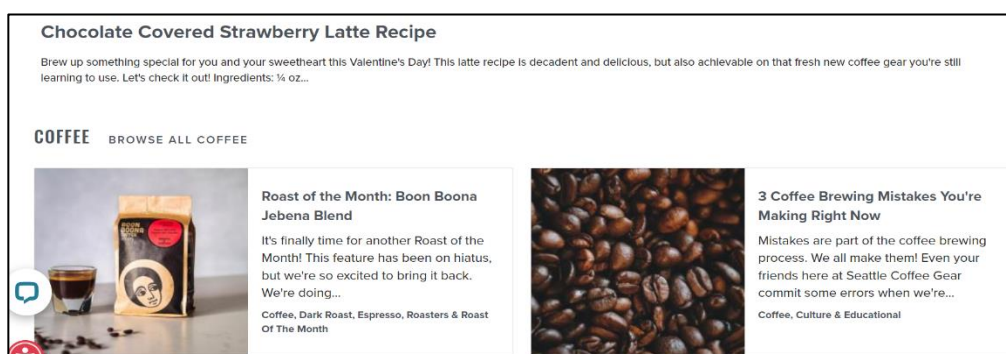


Figure 2.2.30: Learning and guiding materials provided by Seattle Coffee Gear

2.2.4 The Coffee Bean & Tea Leaf

The Coffee Bean & Tea Leaf, commonly known as CBTL, was founded by Herb Hyman in Southern California, United States in 1963. The company's invention of Ice Blended Drinks played a significant role in its success. Over the years, CBTL has expanded to over 25 countries, with nearly 1000 company-owned and franchise stores worldwide, including in Malaysia. In order to serve customers globally, the company has launched its own website in different countries, including Malaysia. This review will be focusing on the website of Malaysia branch, which could be accessed at <https://www.coffeebean.com.my/>. The following are some of the key features of the website, along with explanations:

- i. Product filtering
- ii. Product purchasing
- iii. Product searching
- iv. Product recommendation
- v. Product information
- vi. Product review

i. Product Filtering

CBTL's website offers a product filtering feature that allows users to sort and filter products based on various criteria. For instance, users could filter the product list based on specific categories, such as coffee, as shown in the figure

below. When users choose to filter by coffee, only coffee products are displayed on the website.



Figure 2.2.31: Product filtering page with Coffee filter applied (The Coffee Bean and Tea Leaf® Malaysia)

ii. Product Purchasing

CBTL has implemented an anonymous shopping cart feature on its website, allowing users to add products to their cart and make purchases without having to create an account or log in. As shown in the figure below, the website's shopping cart displays the products added by the user along with their prices, and provides options to update the quantity of items.

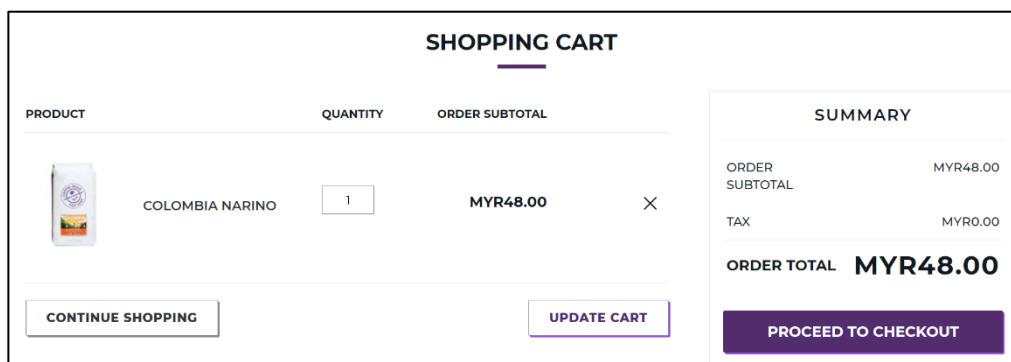


Figure 2.2.32: Shopping Cart Page

iii. Product Searching

CBTL's website also includes a search feature that allows users to find products by entering relevant keywords. As shown in the figure below, users could input keywords related to the product they are looking for, and the website will display a list of related products.

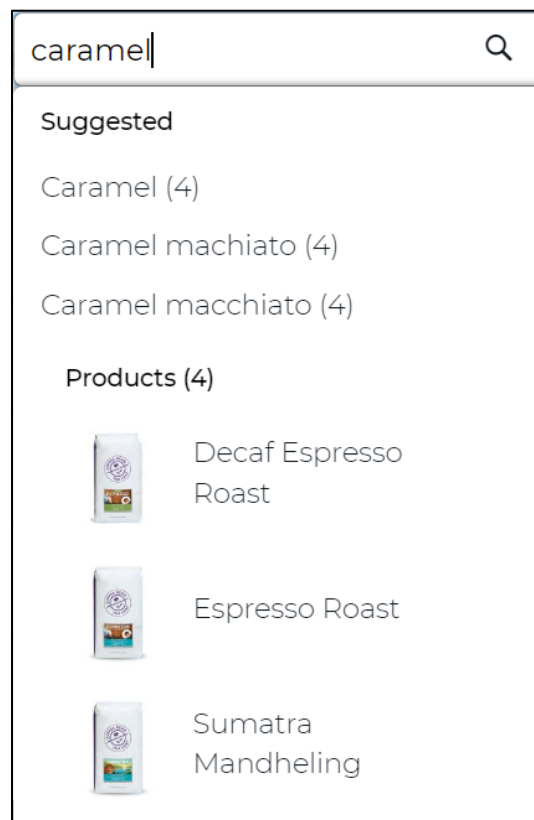


Figure 2.2.33: Searching Result with the “caramel” keyword entered

iv. Product Recommendations

CBTL has also implemented a product recommendation feature on its website. When a user views a detailed information of a specific product, related and non-related products will be recommended for their consideration. For instance, based on the figure below, when a coffee bean product is selected, both related and non-related products will be recommended on the website.

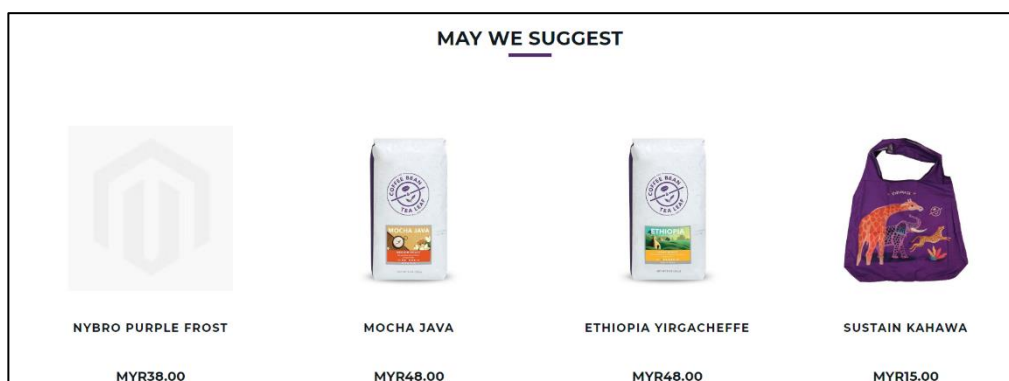


Figure 2.2.34: Product Recommendation Page of CBTL

v. Product Information

CBTL's website provides information on each product, but the product descriptions are brief. For example, as shown in the figure below, the product background information provided is short and simple.

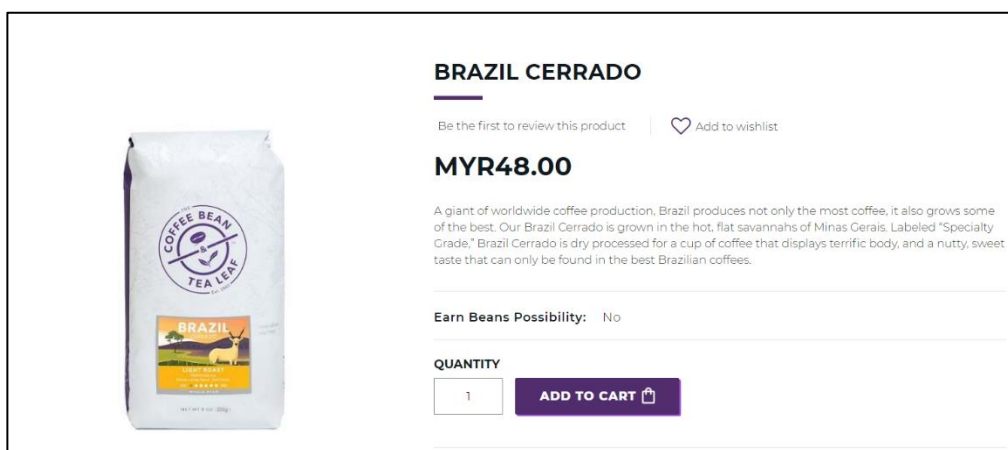


Figure 2.2.35: Product information of Brazil Cerrado, a coffee bean

vi. Product Review

CBTL's website allows users to write reviews and feedback on their products. However, unlike some other websites, CBTL does not provide a feature for users to view reviews written by other customers.

Figure 2.2.36: A sample of the review form of CBTL's website

2.2.5 Comparison Between Existing Similar Web Applications

Table 2.2.1: Comparison Between Existing Applications

<i>Website Name</i> Features	Noc Coffee Company	Starbucks Coffee Company	Seattle Coffee Gear	The Coffee Bean & Tea Leaf
Currency and Language Options	Yes	No	No	No
Product Filtering	Yes (1 criteria)	Yes (multiple criterias)	Yes (multiple criterias)	Yes (multiple criterias)
Product Purchasing	Yes	Yes	Yes	Yes
Product Information	Yes	Yes	Yes	Yes
Product Recommendation	Yes	Yes	No	Yes
Product Searching	No	Yes	Yes	Yes
Product Review	No	Yes (read & write)	Yes (read & write)	Yes (write only)
Guiding and Learning Materials	No	Yes	Yes	No
Internal Roasting Process	Yes	No	No	No
Customization of Beverages	No	Yes	No	No
Real Time Chat Service	No	No	Yes	No

In conclusion, each of the web applications reviewed above has provided unique and valuable features that set them apart from their competitors.

Some features are worth considering for implementation in the development of this project. These include:

- i. Product Purchasing
- ii. Product Recommendations
- iii. Product Information
- iv. Customer Reviews and Ratings
- v. Product Searching
- vi. Guiding and Educational Materials

By incorporating these features, the web-based coffee shop could enhance its user experience and provide a more comprehensive and convenient platform for customers to browse, purchase, and learn about coffee products. In addition, basic features such as a login module and register module will also be included in the project development. Other than that, some of the other features that relates with communication such as the real time chat service implemented in the Seattle Coffee Gear website will be eased and combined using the chatbot features.

Overall, by combining these features, the web-based coffee shop will provide a seamless and enjoyable user experience, as well as a wealth of information and resources to guide customers in their coffee journey.

2.3 Programming Framework

A programming framework is a useful tool for developers to promote code reusable, build software applications faster, and ensures a better quality for the end product. However, with so many options available, choosing the right framework could be a challenge. If the wrong framework is chosen, it could cause time wastage and errors when switching to a different framework (Salas-Zárate et al., 2015). Thus, in this section, there will be 3 programming frameworks to be reviewed, which include Laravel, Reactjs, and Node.js to determine the most suitable programming framework in this project.

2.3.1 Laravel

Laravel is a highly popular, free and open-source PHP-based web application framework designed to simplify and accelerate the development of robust and scalable web applications. It was created by Taylor Otwell and utilizes the widely adopted Model View Controller (MVC) architectural pattern to arrange application logic. It could be said that the emergence of Laravel has had a significant impact on the PHP-based framework landscape (Radchenko and Pyatkin, 2019).

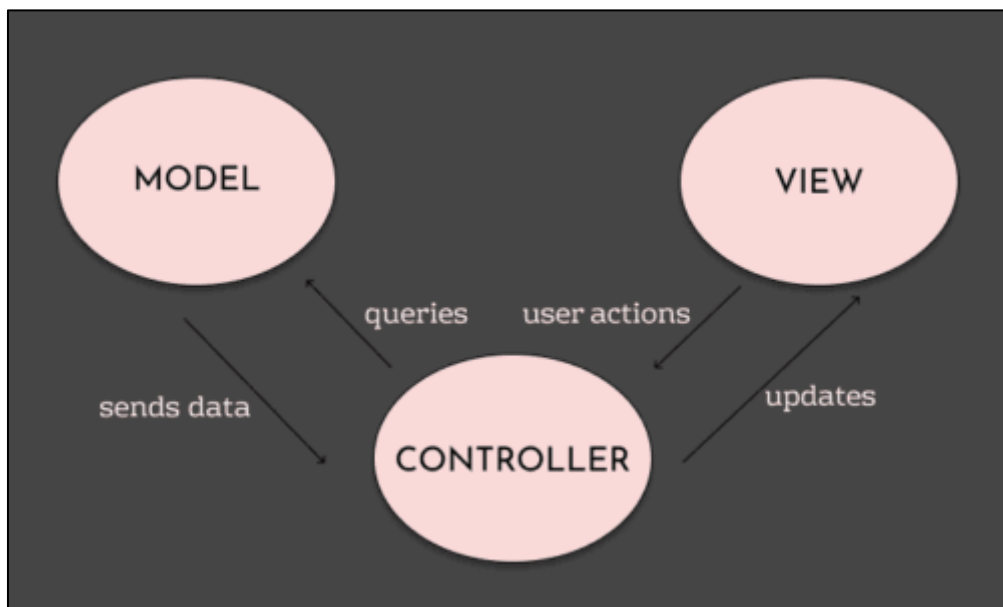


Figure 2.3.1: MVC pattern (Ighodaro, 2018)

The figure above illustrates the architecture of the Model-View-Controller (MVC) pattern, which is made up of three significant components: the Model, View, and Controller. The Model primarily deals with the data-related logic and acts as an intermediary between the View and Controller, facilitating the transfer of data between them. The View, on the other hand, is responsible for managing the components in the user interface of the application and presenting the data to the user in a comprehensible manner while the Controller handles all the incoming requests and operations, acting as a mediator between the Model and View (Yadav, Rajpoot and Dhakad, 2019).

In addition to its adherence to the MVC pattern, Laravel provides a range of features that developers could use. The table below outlines some of the features provided by Laravel.

Table 2.3.1: Key Features of Laravel

Features	Description
Composer package manager	Composer package manager offers a simple and efficient way to install and manage packages and dependencies I Laravel (Radchenko and Pyatkin, 2019).
Eloquent Object Relational Mapping (ORM)	Eloquent ORM allows developers to establish relationships between objects and databases and provides a range of methods for querying and manipulating data (Radchenko and Pyatkin, 2019).
Blade template engine	Blade template engine offers a simple, yet powerful syntax for creating templates that define the user interface of web applications (Radchenko and Pyatkin, 2019).
Artisan command	Laravel provides a built-in command-line interface called Artisan, which offers a range of commands to execute common development tasks (Radchenko and Pyatkin, 2019).

Despite with the features provided, Laravel also consist of its benefits and drawbacks. The table below shows the advantages and disadvantages of Laravel framework.

Table 2.3.2: Advantages and Disadvantages of Laravel

Advantages	Disadvantages
Provide adequeate understandable documentations	The learning stage of Laravel could be difficult
Have large and powerful community that helps in continuous developing the tools of Laravel	Problems evolved due to frequent updates

Have built-in authentication and authorization that ease a lot of coding works and ensure security levels	Consist of complex features that are hard to understand
The integration of API is simply and straightforward	Hard to find a skilled Laravel developer
Provides easy way to handle errors and exceptions	Most of the documentations are only in English language

2.3.2 Reactjs

React is a widely used front-end web framework that employs a JavaScript library and was originally created by a software engineer, Jordan Walke, who was working at Facebook, in 2011. It is currently being developed by Facebook and a community of individual developers. Facebook itself uses React extensively in its own projects, including its top social media applications such as Facebook and Instagram (Saks, 2019). React has become highly sought-after in the industry due to its straightforwardness, adaptability, and efficient development process for producing reusable UI components (Rawat and Mahajan, 2020). With React, developers could easily create complex UI components and manage the application state using a declarative approach.

React has provided a wide range of features, the following table outlines some of the key features provided by React.

Table 2.3.3: Key Features of Reactjs

Features	Description
Lightweight Document Object Model (DOM)	React uses the lightweight DOM, namely virtual DOM to reduce direct manipulation of the DOM tree. This results in faster and more efficient execution of the application and minimizes the unnecessary updates or reload of pages, leading to a better performance (Bhalla, Garg and Singh, 2020).
JSX syntax	JSX is a tool that extends the syntax of JavaScript and enables programmers to write code that looks like HTML right inside their JavaScript code. It simplifies

	the process of building user interfaces in React-based applications by allowing developers to define UI components in a way that closely resembles the final HTML output. This makes it easier to understand and improve the code, resulting in simpler and more efficient development (Bhalla, Garg and Singh, 2020).
One Way Data Flow	ReactJS utilizes a unidirectional data flow where data is passed down from parent components to child components, creating a single direction of data flow between the different layers and components of the application. This approach provides better control over the application's state and reduces information inconsistencies (Bhalla, Garg and Singh, 2020).
Third party ecosystem	Reactjs makes use of third-party packages available through the Node Package Manager (NPM) to expand its capabilities and access additional features, such as React Router. These packages are part of the larger ReactJS ecosystem and allow developers to build more robust and complex applications (Rawat and Mahajan, 2020).

Despite to its key features, React also consist of its pros and cons. The table below shows the advantages and disadvantages of using React.

Table 2.3.4: Advantages and Disadvantages of Reactjs

Advantages	Disadvantages
Component-based architecture where codes are loosely coupled with each other.	Rapid changes and updates in the system which forces user to change their way of code
High performance as it uses the concept of virtual DOM	Lack of convention in which there are multiple ways of code, which might lead to difficulty in picking up

Learning stage of React is easy	Not a full-featured framework in which extra tools or framework needed for the server side
Support multiplatform such as web or mobile applications (KnowledgeHut, 2022)	Poor documentation due to rapid and high development of React (KnowledgeHut, 2022)

2.3.3 Node.js

Node.js is a JavaScript runtime that operates on the server-side. It utilizes the V8 JavaScript engine from Google and is designed to provide high performance and low memory consumption for web applications (Tilkov and Vinoski, 2010). Created by Ryan Dahl in 2009, Node.js has quickly gained popularity as a powerful tool for building scalable web applications. As an open-source, cross-platform framework, it allows developers to write server-side code in JavaScript, enabling seamless integration between the front-end and back-end of web applications (Dalbard and Isacson, 2021). Notable web industry companies, such as Yahoo, have helped in the process of developing Node.js. Meanwhile, Microsoft have gone a step further by incorporating Node.js support into their cloud infrastructure (Chitra and Satapathy, 2017). The following table highlights some of the key features of Node.js that distinguish it from other frameworks, making it a popular choice for web applications development.

Table 2.3.5: Key Features of Node.js

Features	Description
Event-based architecture	Event-based architecture collaborates with asynchronous I/O to handle a huge number of requests at one time in a single thread to execute code, resulting in lower memory consumption, higher scalability and higher performance (Chitra and Satapathy, 2017).
Asynchronous Input/Output (I/O)	By using non-blocking I/O, Node.js is able to handle multiple complex input and output requests simultaneously, without blocking the event loop. This allows for faster and more efficient processing of

	requests, and makes Node.js well-suited for building high-performance, scalable applications (Chitra and Satapathy, 2017).
NPM Package Manager	The libraries of Node.js could be increased by installing additional packages using NPM Package Manager. These packages may help to simplify complex task and fasten the development process (Chitra and Satapathy, 2017).
Runtime Environment	Node.js provides a runtime environment that manages programs running on the event thread, offering benefits such as high performance, scalability, and efficient handling of concurrent requests (Chitra and Satapathy, 2017).

In addition to its key features, Node.js also has its pros and cons as well. The following table outlines the advantages and disadvantages of Node.js.

Table 2.3.6: Advantages and Disadvantages of Node.js

Advantages	Disadvantages
Full stack development in which it could be used as both front-end and backend framework	Poor performance with heavy computations tasks
Fast speed in processing a web task or request	May caused nested callbacks due to its heavily reliable on asynchronous IO operations
Rich ecosystem in which additional packages could be installed through NPM Package Manager	Rely on third party libraries that are not included in Node.js core library which may causes in difficult to find relevant information and documentations
Learning stage of Node.js is easy (Tondon, 2022)	Extra steps needed to connect between relational database (Tondon, 2022)

2.3.4 Comparison Between Programming Frameworks

Table 2.3.7: Comparison Between Programming Frameworks

Frameworks Factors	Laravel	Reactjs	Node.js
Language	PHP	JavaScript	JavaScript
Architecture	MVC	Component-based	Event-based
Performance	High	High	High
Ease of Use	Difficult	Easy	Easy
Community Support	Large	Large	Large
Integration	Optional to integrate with front-end frameworks	Must integrate with a server-side framework	Optional to integrate with backend frameworks

To summarize, each programming framework has unique features that contribute to its popularity and distinguish it from others. For this project, Laravel has been selected as the preferred framework because it simplifies complex tasks and accelerates the development process. One example is Laravel's ability to seamlessly connect relational databases with objects. Given the importance of database interaction in ecommerce websites, having a simple way to configure and manage the database is critical. Another factor in favor of Laravel is its robust built-in security mechanism. It provides authentication and authorization features that could be easily implemented with a single command, reducing the development time and increasing security. Moreover, Laravel could function as a standalone framework without the need for additional front-end frameworks, which simplifies the development process and minimizes code conflicts. Overall, Laravel's intuitive syntax, powerful tools, and active developer community make it a solid choice for web development projects of various sizes and complexities.

2.4 Software Development Methodology

Software development methodology is a structured approach that helps developers in designing, implementing, and testing new software. By following a methodology, developers could reap the benefits and guidance of a professional and structured sequence of steps at each stage of development (Indeed Editorial Team, 2022). Thus, selecting a correct software development methodology plays a critical role in the success of a software development project and satisfaction of a client (Mitchell, 2022). In this section, an in-depth review of 4 methodologies, which are, Waterfall, Rapid Application Development (RAD), Prototype, and Feature Driven Development (FDD) will be conducted to determine the most suitable one for this project.

2.4.1 Waterfall Methodology

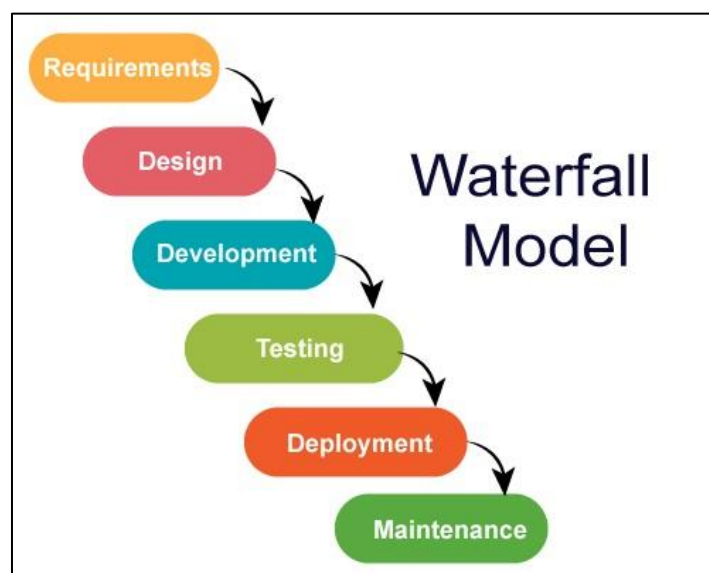


Figure 2.4.1: Waterfall Model (Rault, 2022)

The Waterfall model is the first and oldest methodology that was initially presented by Winston W. Royce in 1970 as a sequential project management approach. This model emphasizes the importance of well-defined and structured phases in the development process. Each phase must be thoroughly defined and well documented before the next phase could begin (Sinha and Das, 2021). This helps to minimize the occurrence of overlapping issues as reversing a phase in

the project is extremely difficult in this model, which could lead to problems if changes are needed later on (Balaji and Murugaiyan, 2012).

Each phase will have its own goals and deliverables with a specified time of period that primarily rely on the previous phase to achieve. For instance, a flawed design may not be identified until the development phase is reached, which could cause problems as the development phase builds upon the design phase. In such cases, a poor design may lead to development failure. Moreover, in the practice of this methodology, testing is only involved after the development phase as shown in the figure above, meaning that defects will be found very late, resulting in higher costs and a longer overall project timeline (Balaji and Murugaiyan, 2012).

The Waterfall model is ideal for a small-scale project as requirements are clearly defined before the project starts and thus, making planning to be easily documented (Despa, 2014). However, it is not suitable for large-scale projects that are complex and consist of critical missions from time to time, making it challenging to define requirements completely before the project commences (Chandra, 2015).

The following table illustrates the advantages and disadvantages of the Waterfall model.

Table 2.4.1: Advantages and Disadvantages of Waterfall Model

Advantages	Disadvantages
Requirements are clear at project outset	Inflexible to changes occur
Each phase is given a specified time duration to complete	Extremely hard to reverse to previous phase and unsupportive of iterations
Proper documentation is implemented into each phase for quality control	Testing involved in the late stage of phase resulting in higher costings and time-consuming
Easy to be implemented due to its sequential development approach (Balaji and Murugaiyan, 2012)	High risk if problems are not fully solved in each phase. (Balaji and Murugaiyan, 2012)

More manageable due to minimum overlapping issues	Not suitable for complex projects
Precise goals set due to the well-defined requirements and detailed planning (Rault, 2022)	Excluded user feedbacks in which final product might varies to the user requirements (Nikolaieva, 2023)

2.4.2 Prototype Methodology

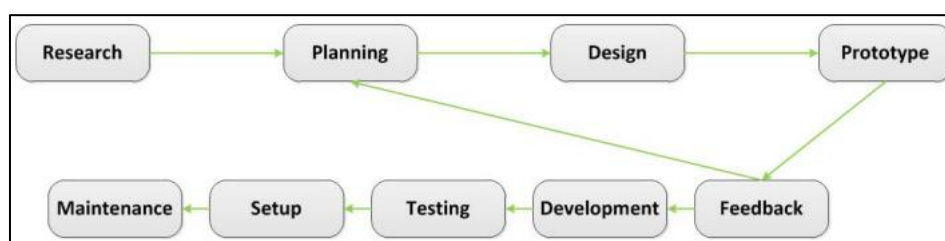


Figure 2.4.2: Prototyping Model (Despa, 2014)

The Prototype methodology is invented based on the concept of hardware prototyping in the early 1980s (Fern and Donaldson, 1989). It was developed to address the need for more effective specification definition that could accommodate uncertainties such as frequent requirements change. For instance, a request for changes in the design phase is accepted and it should be treated as a part of the development process in which the changes could be reacted into the development process in a short period of time (Burns and Dennis, 1985).

Prototype methodology primarily focuses on developing the actual product than defining and planning requirements which involves creating a preliminary version of a software product that incorporates key functionality, rather than a complete product. During the initial phase of the methodology, only critical functions and features are implemented and will be improved by modifying and adding new features through subsequent iterations based on user feedback. Moreover, this approach often results in high user satisfaction since users are given the opportunity to be involved in the development process by examining prototypes, indirectly preventing miscommunications and misunderstandings (Sabale and Dani, 2012).

As depicted in the figure above, after receiving user feedback, if there are more feature to be implemented, a new iteration of the prototype

methodology will be carried out to produce a new prototype, otherwise, the project will proceed to the next phase. Despa (2014) reported that project owners have provided feedback suggesting that the Prototype methodology is best suited for innovative projects with no similar examples to reference. However, this methodology may not be suitable for large scale project due to undefined requirements at the project's outset, which may require more iterations to achieve the desired outcome, resulting in increased project timelines and costs (Chandra, 2015).

The table below describes the advantages and disadvantages of applying Prototype methodology.

Table 2.4.2: Advantages and Disadvantages of Prototype Model

Advantages	Disadvantages
Flexible in requirement changes	Lack of analysis and design as it focus more on developing a prototype
Higher users satisfaction	User confusion as users may confuse the prototype with the final product
Active users involvement in which users could participate actively into the development	Time consuming when multiple prototypes are developed
Quick feedback from users due to high user involvement (Tram, 2022)	Lack of quality control as prioritize more on completing the prototype faster (Tram, 2022)
Perform well in innovative projects with no similar examples to reference	Not suitable for large scale project

2.4.3 Rapid Application Development (RAD) Methodology

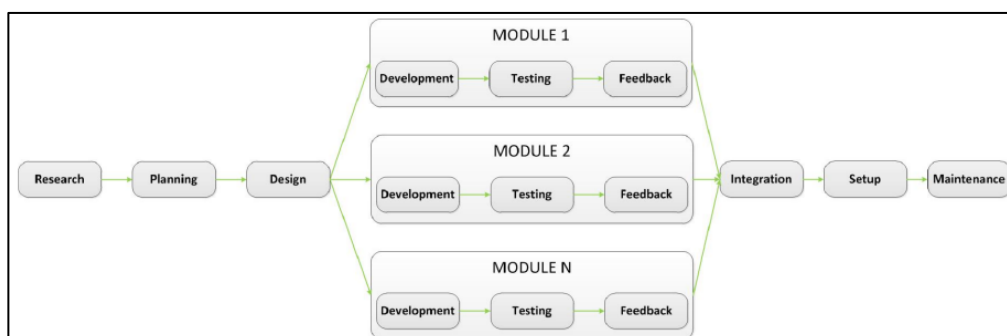


Figure 2.4.3: Rapid Application Development (RAD) model (Despa, 2014)

The Rapid Application Development (RAD) methodology was initially introduced by James Martin in the year of 1991 in his book "Rapid Application Development". Martin aimed to overcome the limitations of the traditional development lifecycle by prioritizing the speed and quality with a lower cost in the development process (Daud, Bakar and Rusli, 2010). The RAD model mainly focuses on prototyping and user involvement, compressing all phases into a short, iterative development cycle (Berger, Beynon-Davies and Cleary, 2004). Specifically, according to Geambaşu et al. (2011), requirements specification and design phase only accounts for up to 30% of the total effort in RAD.

The RAD methodology is designed to streamline the software development process by breaking it down into smaller modules, enabling the simultaneous development of multiple cycles as depicted in the figure above. It also employs time-boxing, where delivery deadlines are defined and must not be exceeded. If necessary, features and requirements may be adjusted to fit within the allotted time frame, ensuring that the project stays on track and within budget (Geambaşu et al., 2011). Another key aspect of RAD is the use of a prototype. The idea of this is to create a prototype and make improvement in terms of modifying or adding new features or functionality before it is acknowledged as a complete product (Gall and Richard, 2005). The prototype undergoes a cycle of inspection, discussion, and amendment for at least 3 iterations, to refine and improve it until it meets user satisfaction (Beynon-

Davies and Holmes, 2002). To ensure user satisfaction, effective communication and involvement of users is highly emphasized in the RAD methodology. This helps to prevent misunderstandings and miscommunications between users and developers, ensuring that the final product meets the user's requirements and expectations (Gall and Richard, 2005).

This methodology is generally best suited for projects with a smaller scope and developed by a small team (Rosyad et al., 2019). However, it may not be as suitable for large-scale projects that are more complex in nature (Chandra, 2015).

The table below illustrates the advantages and disadvantages of applying the RAD methodology.

Table 2.4.3: Advantages and Disadvantages of RAD Model

Advantages	Disadvantages
Time saving as everything will be developed in a rapid way	Projects are frequently rushed to meet deadlines
Requirement changes could be implemented quicker than traditional SDLC approaches (Chandra, 2015)	Lack of attention to details as it emphasizes more on speed and efficiency
Active user involvement in defining user requirements	Requires an experienced team to work on the system and learn new skills to adapt to changing requirements
Stakeholders will feel a strong sense of ownership over the project's success due to active participation	Possibility of high risk due to lack of attentions to details (Chandra, 2015)
Works well for small teams and projects with limited scope (Rosyad et al., 2019)	May not perform well in large and complex projects. (Rosyad et al., 2019)

2.4.4 Feature Driven Development (FDD) Methodology



Figure 2.4.4: Feature Driven Development (FDD) Model (Despa, 2014)

The Feature Driven Development (FDD) methodology was introduced by Jeff De Luca and Peter Coad in 1997 after working on a project in Singapore. FDD, which evolved from The Coad Method proposed by Peter Coad, is an agile software development methodology that aims to prevent confusion and minimize the likelihood of costly rework. This methodology involves breaking down development activities into two stages, defining a feature list to be implemented and conducting iterations for feature implementation. (Chowdhury and Huda, 2011).

Unlike other agile methodologies, the first stage in FDD involves creating an overall project model. This model serves as the basis for identifying and creating a list of features that are needed to be developed (Tirumala, Ali and Babu, 2016). This step is crucial, as the quality and work done here directly impact the accuracy of project monitoring, the ability to maintain and extend the project code (Chowdhury and Huda, 2011). To ensure the accuracy of the feature list, it is essential for customers and stakeholders to fully participate in this process to define and understand the features to be implemented. To ensure clarity, the features should be written in a language that is understandable by both customers and developers. These features will be presented as a business value to the clients, as they represent the functionality that will be available in the project (ISMAIL, QADRI and FAHAD, 2015).

The second stage in FDD involves the implementation of each feature defined in the feature list into the project. This stage typically follows a cycle of iterations that include planning, designing, and building, with each iteration taking no more than two weeks to complete (Nikolaieva, 2023). During the planning phase, each feature is assigned to an individual developer who will be responsible for its creation and maintenance. The planning phase also includes determining a start time and an estimated duration for each feature, with the

focus on ensuring that the process remains efficient and on schedule. The designing and building phase mainly focuses on outlining and designing the features in compliance with the assigned individual developer. They will organize a list of activities leading to the implementation of features, which will be reflected in the source code of the project (Rychlý and Tichá, 2008).

According to Despa (2014), the FDD model is suitable for all scales of projects with constraint of constant interactions between developer teams and customers throughout the entire project.

The following table listed the advantages and disadvantages of applying FDD model.

Table 2.4.4: Advantages and Disadvantages of FDD Model

Advantages	Disadvantages
High flexibility in requirement changes	More complex compared to other agile methodologies due to its detailed and structured approach for implementing features
Focused scope as development only focus on a small, incremental features one at a time	Limited visibility of overall progress of project as development is bits by bits
Early and frequent delivery as it adapt to implementation of feature by feature	High cohesion relation between features
Actively participation and communication by customers (Stanke, 2022)	Limited control to the final product as it is implemented feature by feature (Stanke, 2022)

2.4.5 Comparison Between Software Development Methodologies

Table 2.4.5: Comparison Between Software Development Methodologies

<i>Model</i> Characteristic	Waterfall	Prototype	RAD	FDD
Flexibility in Requirement Changes	Very low due to sequential approach	High due to iterative approach	High due to iterative approach	High due to iterative approach
User Involvement	Little (involve only in the beginning of the project)	High (involve throughout entire project)	High (involve throughout entire project)	High (involve throughout entire project)
Experts Required	No	Little	More	More
Development Cost	Low	High	High	High
Project Team	Large Team	Small Team	Small Team	Multiple teams work simultaneously
Risk factor	High	Low	Medium	Low
Project Scale	Small and medium project (requirements must be clearly defined before start)	Small project (best suited for innovative projects)	Small and medium project (best suited for projects with small scopes)	All sizes but complex project

In order to successfully implement a project, choosing the right methodology is critical. Each methodology has its own set of advantages and disadvantages, and it is important to carefully evaluate which methodology is best suited for a specific project to maximize the potential benefits.

For this project, the chosen methodology is the prototype methodology. This methodology is well-suited for this project for several reasons. Firstly, by referring to the comparison table above, Prototype methodology allows for high flexibility in terms of requirement changes compared to the Waterfall methodology. This means that requirements could be improved or modified under several iterative approaches. Secondly, it involves high user involvement, which ensures that clients are catered to as part of the project and requirements are defined based on their needs.

Moreover, compared to other methodologies such as the RAD model and the FDD model, the development cost needed for prototype methodology is relatively low. The RAD model requires hiring experts, while FDD needs to work in multiple teams simultaneously. The prototype methodology, on the other hand, does not require these additional costs. Therefore, the prototype methodology seems to be the most ideal methodology to choose for this project when compared to others.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

In this chapter, a comprehensive explanation of the selected methodology, which is the prototype methodology is provided. In addition, work breakdown structure and Gantt chart is provided as a reference and tracking for the whole development process. The chapter will also specify the development tools that is used in this project.

3.2 Software Development Methodology

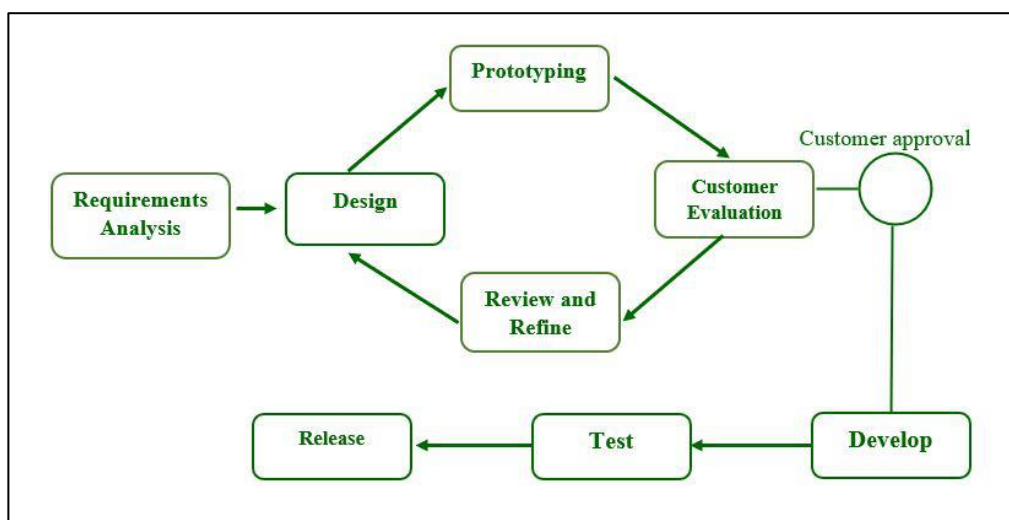


Figure 3.2.1: Prototype Methodology (GeeksForGeeks, 2021)

By referring to the figure above, it could be seen that the prototype methodology consists of 5 primary project phases: requirement analysis, prototyping iteration, development, testing, and deployment. The prototyping iteration phase comprises 4 iterative phases, including design, prototyping, customer evaluation, and review and refine. These iterative processes are repeated until the prototype satisfies the customer's requirements and expectations. A detailed explanation of each phase is provided to ensure a clear understanding of how to carry out each step in the development process.

3.2.1 Requirement Analysis

Requirement analysis is the first stage of the prototype methodology. It focuses on collecting all relevant information related to project requirements. In the case of this project, various methods are used to gather accurate and informative data. These methods include conducting a thorough literature review and performing quantitative analysis through the use of questionnaires. By utilizing these approaches, it could ensure that all necessary information is collected and analyzed, allowing to develop a website that meets the needs and expectations of its users.

3.2.1.1 Literature Review

Literature review plays a crucial role in broadening knowledge related to the domain of the project, as well as identifying the best tools and approaches for successful project development. In this project, 3 literature reviews are conducted.

The first literature review will analyze 4 similar existing web applications, including Noc Coffee Company, Starbucks Coffee Company, Seattle Coffee Gear, and The Coffee Bean & Tea Leaf. This review will identify the features provided by each web application and compare them. This will help identify the relatively complete features needed by a coffee website and analyze the weaknesses of competitors.

The second literature review will analyze 3 programming frameworks, including Laravel, Reactjs, and Node.js. This review will identify the most suitable tools for the development process to simplify tasks and streamline the development process.

Lastly, the third literature review will analyze 4 software development methodologies, including the Waterfall methodology, Prototype methodology, RAD methodology, and FDD methodology. This review will help identify the most suitable methodology for developing a project that meets high-quality standards and customer satisfaction.

By conducting these literature reviews, it helps to make informed decisions and develop a high-quality coffee shop website that meets the needs and expectations of the customers.

3.2.1.2 Questionnaire

Questionnaire is a form of quantitative analysis that could help to understand behavior. The questionnaire is distributed to 50-70 respondents which will comprise of 12 questions within one month. The first section will consist of general questions aimed at gathering basic personal information from respondents. The second section will consist of demonstrative questions, which will seek the opinions of respondents on the best way to implement features into the website. This section will provide valuable insights into the preferences and needs of the target audience.

3.2.2 Prototyping Iteration

There are 3 iterations in total for the prototyping methodology, which includes the phases of design, prototyping, customer evaluation, and review and refine. During each iteration, new features are added or improvements are made to the existing functions based on the feedback received during the customer evaluation phase. As there are no customer roles in the project, the users will be the primary focus of feedback and review providers during the customer evaluation phase. These feedback are used to make necessary changes and improvements to the project before moving on to the next iteration.

3.2.2.1 First Iteration

The first iteration will primarily focus on designing the user interface for the webpages and creating the necessary database for the web application. This will include the login and register page with a corresponding user database, a product page with a product database, a shopping cart page, and any other relevant pages. In addition to the user interface, the routing between each webpage is well navigated to ensure a smooth user experience. Furthermore, the relationships between the various databases are well-defined and optimized.

i. Design

Since the first iteration will focus primarily on designing a user-friendly interface and creating the necessary relational databases, the design layout for

each webpage and the mapping of the routes between them are drafted. In addition, the necessary entities for each database and the relationships between them are also well-defined and drafted. This will help to provide a clearer understanding of the specific tasks that need to be accomplished on each webpage and streamline the development process by having well-planned relational databases.

ii. Prototyping

A basic preliminary prototype is created using the Bootstrap and Laravel frameworks, showcasing the designed user interface for each webpage, which will enable users to navigate between them. On the other hand, the database is created with sample data in it using the PhpMyAdmin that is powered by WampServer. The development of this prototype is expected to take approximately 2 weeks.

iii. Customer Evaluation

Once the first prototype has been completed, the fully designed user interface and well-connected relational databases are showcased to the users for review and feedback. This will provide some valuable insights into the usability of the application and help to refine the design based on user needs and preferences.

iv. Review and Refine

The review and feedback provided by customers during the evaluation phase is carefully evaluated and internally defined. Based on this feedback, the prototype is refined to ensure that it satisfies users before moving on to the next iteration.

3.2.2.2 Second Iteration

The second iteration focused on implementing the defined features into the website. These features are divided into two sections: core features and extra features. The core features consist of the basic CRUD (Create, Read, Update, Delete) operations, while the extra features include additional functionality such

as product recommendations, product information, product search, customer reviews, and more.

i. Design

During the design phase of the second iteration, all features needed are listed and drafted with the data flow implemented based on the database created in the first iteration. Moreover, the overall architecture of the website are drafted using relative tools such as draw.io to ensure that the design is on the right path.

ii. Prototyping

The first approved prototype is served as the basis for further implementation in the second iteration. During this phase, all defined features are implemented, ensuring that the functionality of each feature works as expected and is integrated with the database. The development of the second prototype is expected to take approximately 3 weeks.

iii. Customer Evaluation

The second prototype with well implemented features are proposed to the customers as what have done in the first iteration. Customers will review and test on the functionality for each feature and provide feedback based on this prototype evaluation. The feedback are recorded to further refine the second prototype.

iv. Review and Refine

Similar with the first iteration, the feedback provided during the customer evaluation phase are used to further refine on the second prototype.

3.2.2.3 Third Iteration

The third and final iteration primarily focused on training and implementing chatbot features into the website. The chatbot will serve as a customer support function, as well as providing customized product recommendations.

i. Design

In the design phase of the third iteration, the chatbot conversations and scenarios are designed and drafted to provide comprehensive solutions for customer inquiries. The data flow between the chatbot conversation and its relational database is carefully planned and drafted to ensure that these conversations could be saved and retrieved.

ii. Prototyping

The chatbot is trained with all available training data to ensure that it could effectively fulfill the features that are supposed to be offered. OpenAI will be the chatbot platform used for this purpose. Once the chatbot has been trained, it is integrated into the second prototype, ensuring that it did not cause any errors on any other part of the website. The chatbot is tested to ensure that it functions as expected. The time duration to complete this prototype is approximately 2 weeks.

iii. Customer Evaluation

As mentioned previously, once development of the third prototype is completed, it is presented to the customers for review and feedback. During this phase, the functionality of the chatbot is tested by the customers, and any feedback received is recorded for further refinement.

iv. Review and Refine

Similar to the previous iterations, any feedback received from customers during the customer evaluation phase is used to refine the prototype further and improve overall user satisfaction.

3.2.3 Develop

Once the final prototype is approved and met customer satisfactions, the project moved on to the development phase. During this phase, the finalized prototype is used as a reference to develop the real web application. A more detailed approach to development is implemented to ensure the overall quality of the project. This includes actions such as making code reusable, ensuring functions

are loosely coupled between each other, and incorporating best practices for security and performance. By taking these steps, it could ensure that the final product is reliable, scalable, and efficient.

3.2.4 Test

In the testing phase, several types of testing is conducted to ensure the functionality and reliability of the web application. These include internal testing such as unit testing, integration testing, system testing, and external testing such as user acceptance testing. Internal testing is carried out in the developer environment to identify any issues or errors in the code. On the other hand, in the absence of a customer for this project, 10 users is selected to conduct the user acceptance testing to ensure that the web application has met general needs and preferences. Any bugs or issues found during testing is promptly addressed and fixed, and regression testing is conducted to ensure that the bug fix does not negatively impact other parts of the code.

3.2.5 Release

At this stage, the web application is ready for deployment. Users could use the website to conduct business operations in the real world, such as browsing and purchasing products or services. A complete documentation related with the requirements and the features of the website are acknowledged and agreed by the customers.

3.3 Project Plan

3.3.1 Work Breakdown Structure (WBS)

0.0 Web-based Coffee Shop with Chatbot Integrated

1.0 Project Planning and Requirements Gathering

1.1. Outline Introduction

1.2. Identify problem statement

1.3. Identify project objective

1.4. Determine project solution

1.5. Propose project approach

1.6. Specify project scope

1.6.1. Identify target users

1.6.2. Define main module covered for each target user

1.6.3. Out of scope

1.7. Conduct literature review

1.7.1. Review on similar existing web applications

1.7.1.1. Study features of Noc Coffee Company

1.7.1.2. Study features of Starbucks Coffee Company

1.7.1.3. Study features of Seattle Coffee Gear

1.7.1.4. Study features of The Coffee Bean & Tea Leaf

1.7.1.5. Comparison between the similar existing web applications

1.7.2. Research on programming frameworks

1.7.2.1. Study on the features, pros, and cons of Laravel

1.7.2.2. Study on the features, pros, and cons of Reactjs

1.7.2.3. Study on the features, pros, and cons of Node.js

1.7.2.4. Comparison between the programming frameworks

1.7.3. Research on software development methodologies

1.7.3.1. Specify the characteristics, pros, and cons of Waterfall Model

1.7.3.2. Specify the characteristics, pros and cons of Prototype Methodology

- 1.7.3.3. Specify the characteristics, pros, and cons of Rapid Application Development Methodology
- 1.7.3.4. Specify the characteristics, pros, and cons of Feature Driven Development Methodology
- 1.7.3.5. Comparison between software development Methodologies
- 1.8. Methodology and work plan
 - 1.8.1. Determine the most applicable methodology
 - 1.8.1.1. Define each phase of the methodology
 - 1.8.2. Create Work Breakdown Structure (WBS)
 - 1.8.2.1. Identify tasks and activities
 - 1.8.3. Create Gantt Chart
 - 1.8.3.1. List down the tasks and activities
 - 1.8.3.2. Set start-date and end-date
- 1.9. Conduct fact finding analysis
 - 1.9.1. Construct Questionnaire
 - 1.9.2. Distribute Questionnaire in Google forms
 - 1.9.3. Perform data analysis for questionnaire
- 1.10. Identify project development tools
- 1.11. Requirement Specifications
 - 1.11.1. Determine the functional requirements
 - 1.11.2. Determine the non-functional requirements
- 2.0 System Design
 - 2.1. Use Case Modeling
 - 2.1.1. Illustrate use case diagram
 - 2.1.1.1. Determine use cases
 - 2.1.2. Compile use case description table
 - 2.2. Design Modeling
 - 2.2.1. Illustrate context diagram
 - 2.2.2. Illustrate Level 1 data flow diagram
 - 2.2.3. Illustrate activity diagrams
 - 2.3. System Architecture Design
 - 2.3.1. Illustrate Architecture Design

- 2.3.2. Architecture Design Explanation
- 2.4. Chatbot Architecture Design
 - 2.4.1. Illustrate chatbot embedding architecture design
 - 2.4.2. Chatbot embedding architecture explanation
 - 2.4.3. Illustrate chatbot question answering architecture design
 - 2.4.4. Chatbot question answering architecture explanation
- 2.5. Rough Prototyping
 - 2.5.1. UI design
 - 2.5.2. Define routing between UI
- 3.0 Prototype Implementation
 - 3.1. First Iteration
 - 3.1.1. Design UI prototype details of each page
 - 3.1.2. Get User Feedback
 - 3.1.3. Improve prototype based on user feedback
 - 3.2. Second Iteration
 - 3.2.1. Construct prototype with main functions
 - 3.2.1.1. Authentication Function
 - 3.2.1.2. Product Function
 - 3.2.1.3. Brewing Guide Function
 - 3.2.1.4. Shopping Cart Function
 - 3.2.1.5. Payment Gateway Integration Function
 - 3.2.1.6. Implement Admin Panel
 - 3.2.2. Get User Feedback
 - 3.2.3. Improve prototype based on user feedback
 - 3.3. Third Iteration
 - 3.3.1. Construct prototype with chatbot development
 - 3.3.1.1. Prepare Embedding Data into a File
 - 3.3.1.2. Integration with OpenAI
 - 3.3.1.3. Compute OpenAI Interaction Function
 - 3.3.2. Get User Feedback
 - 3.3.3. Improve prototype based on user feedback
- 4.0 Development
 - 4.1. Develop all module and functions along with its routings

- 4.2. Develop the authentication function for both users and admins
 - 4.3. Develop Admin Panel CRUD functions
 - 4.4. Develop chatbot functions
 - 4.5. Integrate Chatbot into the website
 - 4.6. Build Database For All Functions
 - 4.7. Refine and reduce code duplication
 - 4.8. Improve overall UI design
- 5.0 Testing
- 5.1. Conduct Unit Testing
 - 5.1.1. Create unit test cases
 - 5.2. Conduct Integration Testing
 - 5.2.1. Create integration test cases
 - 5.3. Conduct System Usability Testing
 - 5.3.1. Create usability test scenario for customer
 - 5.3.2. Create usability test scenario for admin
 - 5.3.3. Calculate and document the results
 - 5.4. Conduct User Acceptance Testing
 - 5.4.1. Create user acceptance test template for customer
 - 5.4.2. Create user acceptance test template for admin
 - 5.5. Rearrange Documentation
 - 5.6. Prepare Presentation slides

3.3.2 Gantt Chart

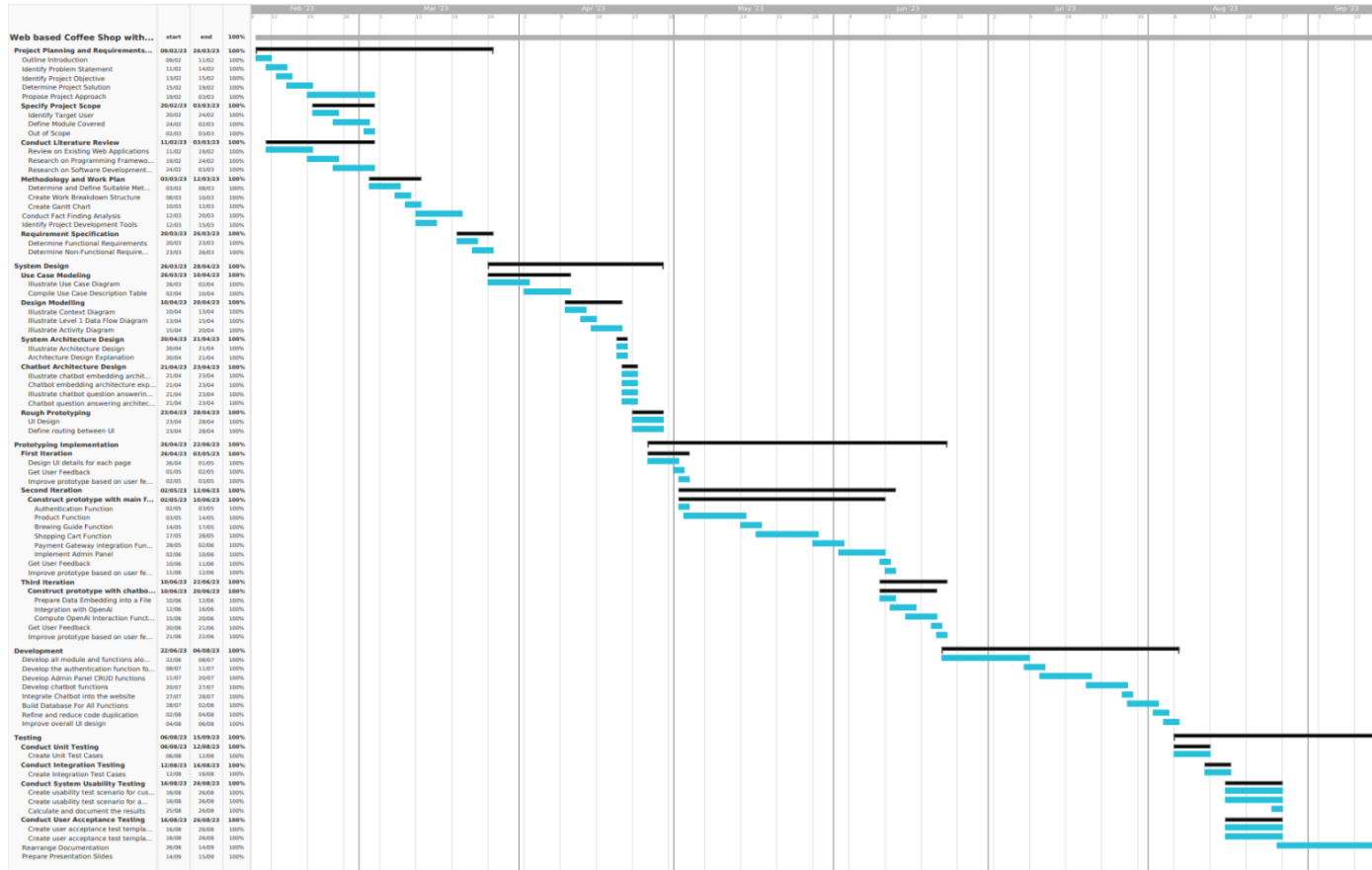


Figure 3.3.1: Gantt Chart

3.4 Project Development Tools

The development of the coffee shop website is facilitated by several tools, including Laravel, Bootstrap, Visual Studio Code, OpenAI API, and PhpMyAdmin. These tools are selected to streamline the development process and optimize the performance of the web application.

3.4.1 Laravel

Laravel is a popular web application framework that utilizes PHP as its programming language. It offers a wide range of useful and unique features, including authentication and authorization. Laravel is used in this project as a backend framework that mainly deals with the CRUD operations with databases along with the extra features it offered. Moreover, the Artisan CLI provided by Laravel has also helps in hosting the website. More information on Laravel could be found in Chapter 2.

3.4.2 Bootstrap

Bootstrap is a widely used framework that helps create responsive and user-friendly websites by utilizing basic web programming languages such as HTML, CSS, and JavaScript. It provides a range of pre-designed components, including tables, forms, buttons, and modals, that could be easily customized using HTML and CSS. Additionally, JavaScript could also be seamlessly integrated into Bootstrap to enhance its functionality and create interactive features (Gaikwad and Adkar, 2019). Therefore, in this project, Bootstrap is utilized as the front-end framework to design the user interface.

3.4.3 Visual Studio Code

Visual Studio Code, or VS Code, is a widely used and free code editor that is open-source and developed by Microsoft. It offers robust support for a diverse range of programming languages such as PHP, JavaScript and more. It also provides users with an extensive collection of extensions to enhance their coding experience. Therefore, in this project, VS Code is utilized as the code editor tool.

3.4.4 OpenAI

OpenAI is a well-known research laboratory focused on developing advanced artificial intelligence systems that could positively impact society. Its primary objective is to create safe and friendly AI that could coexist with humans. One of its major accomplishments is the development of GPT-3, one of the most advanced and largest language models available today. The ChatGPT system was built using GPT-3 as a basis. Thus, it is used in the project as an integrated API to support the chatbot feature.

3.4.5 PhpMyAdmin

PhpMyAdmin is a free and open source that mainly works on the databases, including MySQL and MariaDB. It allows users to perform CRUD operations and run SQL queries on their databases. In this project, the MySQL and phpMyAdmin are used for managing the databases.

CHAPTER 4

PROJECT INTIAL SPECIFICATION

4.1 Introduction

This section will involve conducting a fact-finding analysis based on a questionnaire distributed through Google Form to 50 respondents. The functional and non-functional requirements, illustrations of the use case diagram along with descriptions for each defined use case are outlined as well.

4.2 Facts Finding

In this section, an overview analysis is performed based on the survey conducted using a questionnaire. A total of 50 respondents participated in this survey, mainly between the ages of 18 and 40, comprising of different occupations, which is consistent with the targeted user demographic.

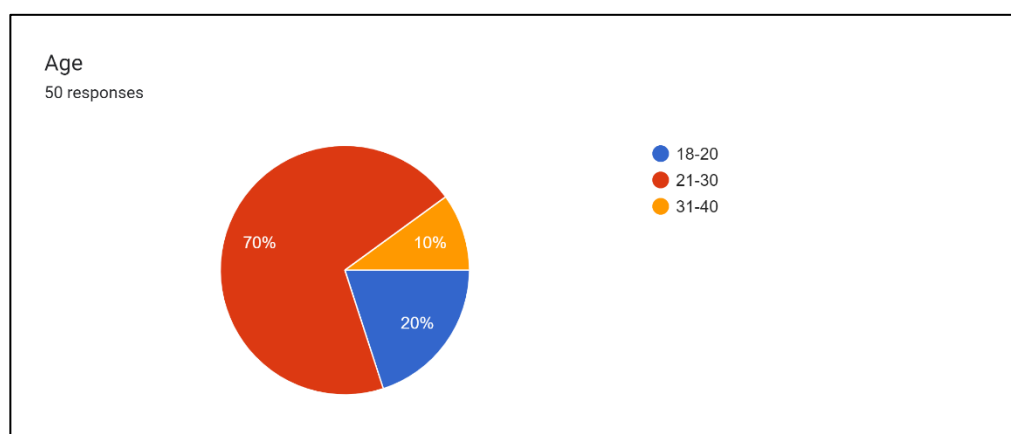


Figure 4.2.1: Distribution of age among the participants

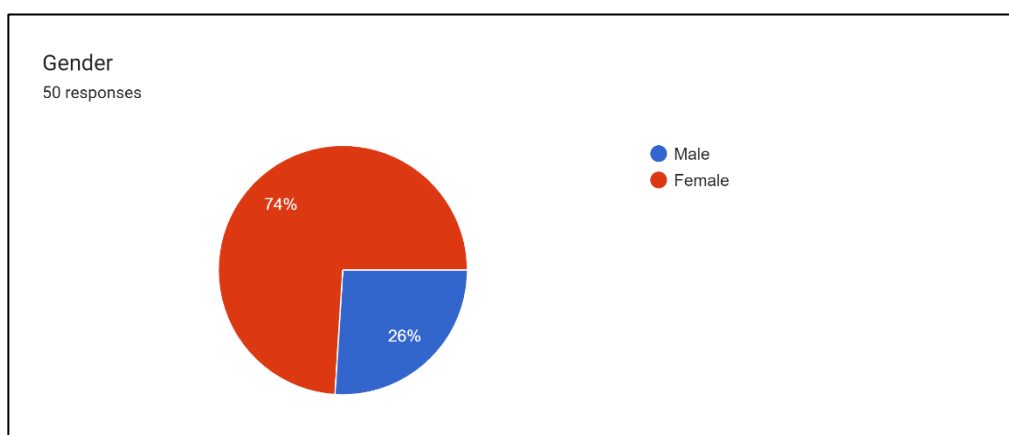


Figure 4.2.2: Distribution of gender among the participants

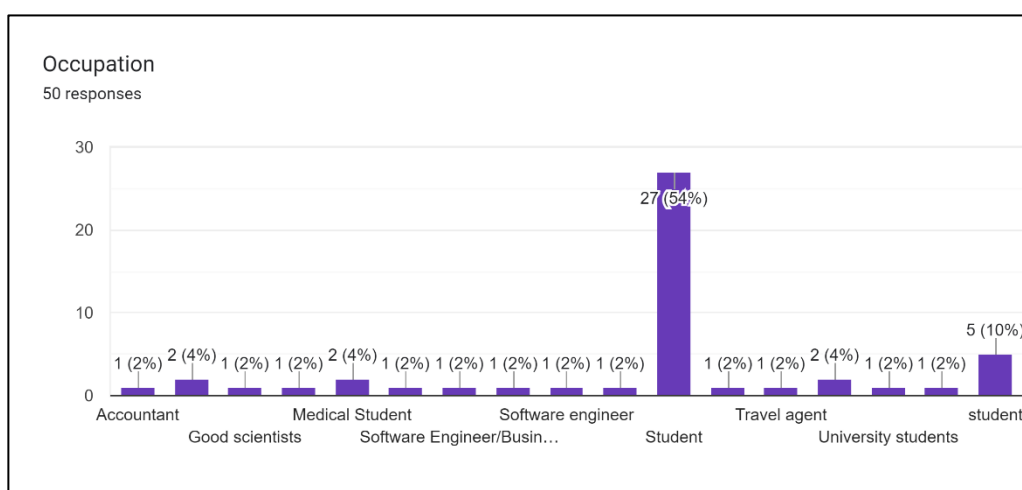


Figure 4.2.3: Distribution of occupations among the participants

In the first 3 questions, the questionnaire mainly focused on collecting demographic information from the targeted users. According to Figure 4.2.1, the age range of the 50 respondents was between 18 to 40 years old, with 70% being between 21 to 30 years old, 20% between the age of 18 to 20 years old, and 10% between 31 to 40 years old. Among these participants, 74% were female and 26% were male by referring to the Figure 4.2.2. There was a diverse range of occupations among the respondents, as seen in Figure 4.2.3. Most of them were students, while others were employed as software developers, accountants, food scientists, and more.

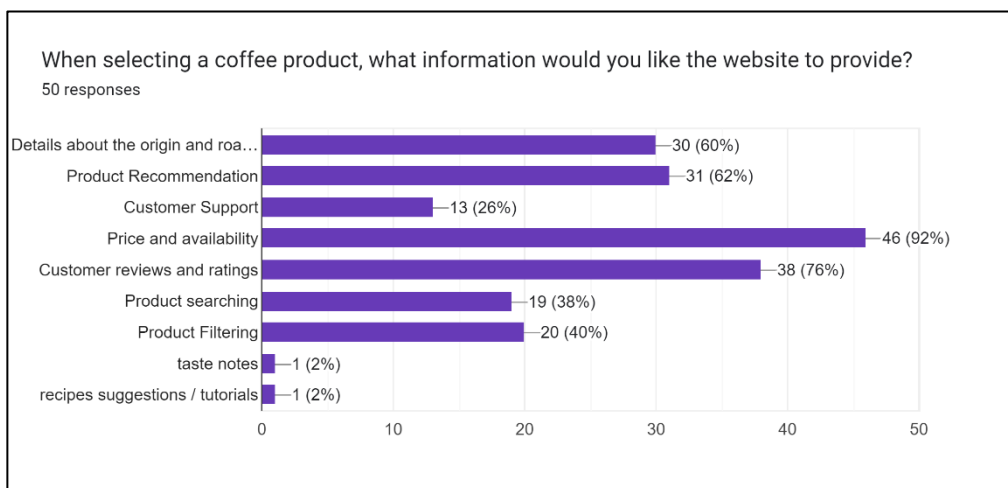


Figure 4.2.4: Distribution of feature demands on the coffee shop website

The fourth question of the survey mainly aimed to gather information on the most in-demand features for the coffee shop website. Based on the results shown in Figure 4.2.4, it is evident that the highest demand feature among the 50 respondents is the availability and pricing of the products, followed by features such as customer reviews and ratings, product recommendations, information on the origin and roasting process, product filtering, product searching, customer support, and lastly taste notes and recipe suggestions, respectively.

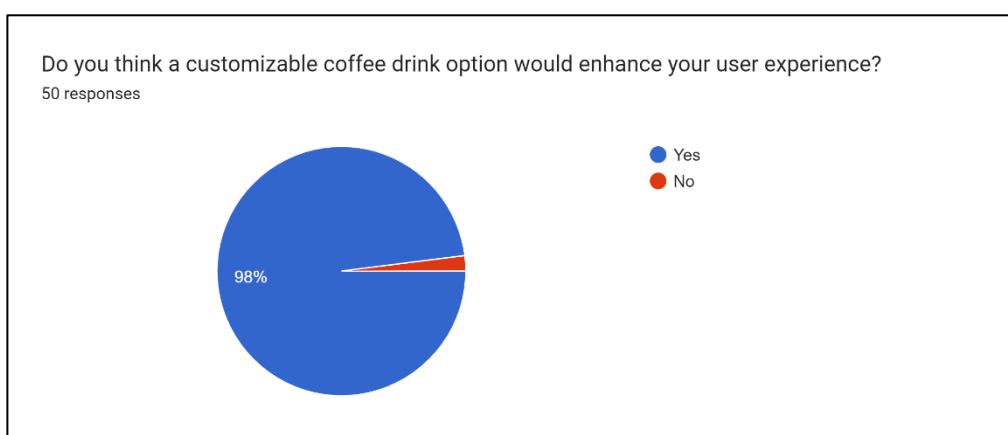


Figure 4.2.5: Distribution of respondents' opinion on customizable coffee drink option

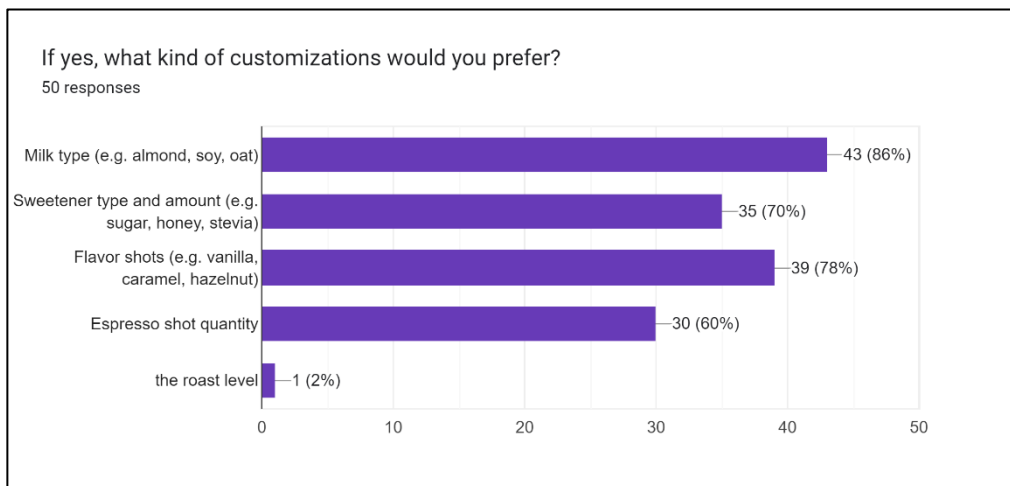


Figure 4.2.6: Distributions of preferable beverage customization options

The following questions are mainly focused on the gathering the information of the demand for beverage customization and the most preferable customization options among the 50 respondents. Based on Figure 4.2.5 and Figure 4.2.6, it could be observed that 98% of the respondents consider beverage customization as a necessary feature, while only 2% believe it is not essential. Moreover, according to the result shown above, it could also be seen that the milk type is the most in-demand option for customization, followed by flavour shot, sweetener type and amount, espresso shot quantity and lastly roast level.

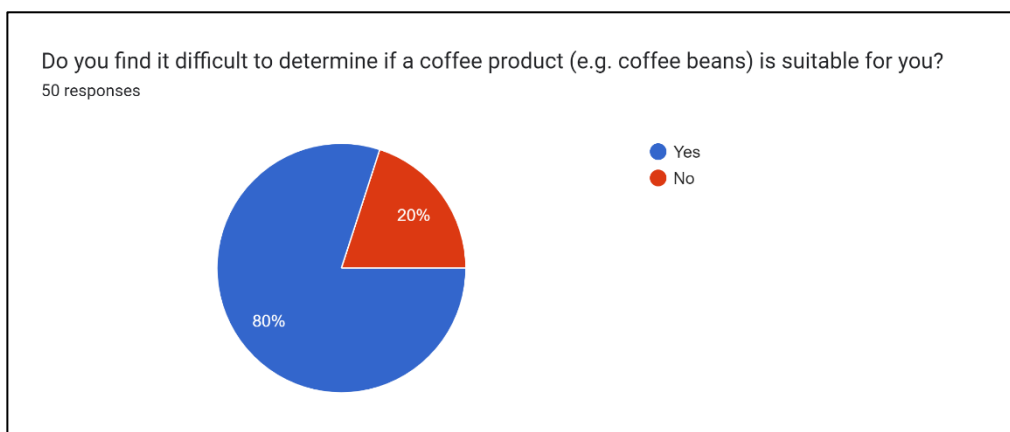


Figure 4.2.7: Distribution in having difficulty to determine the suitability of the coffee products

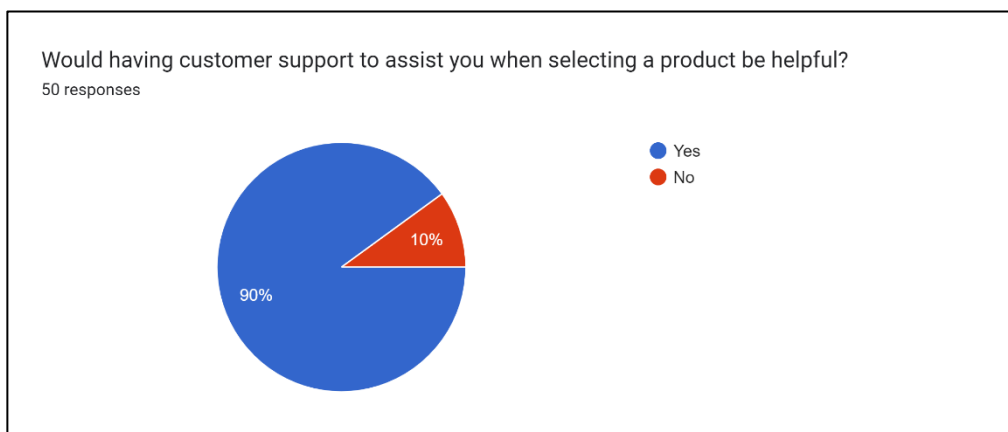


Figure 4.2.8: Distribution of whether customer support in product selection would be helpful

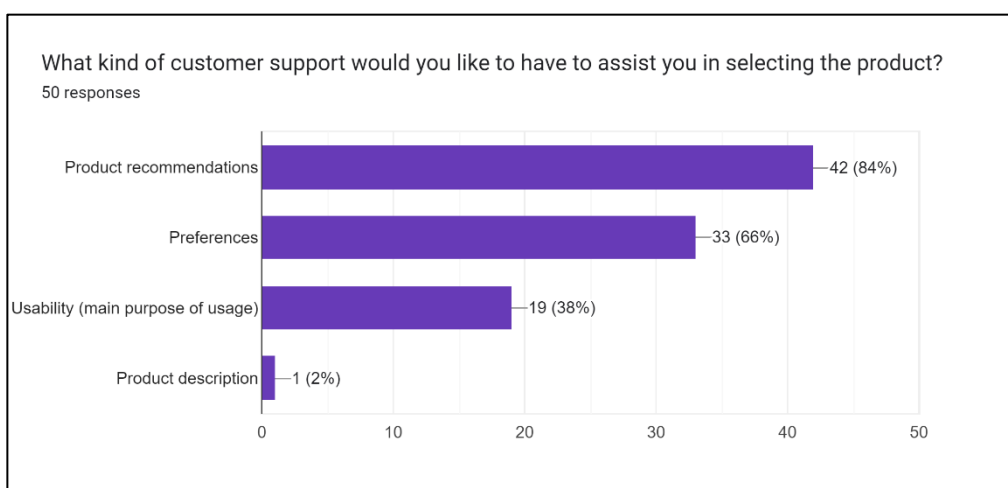


Figure 4.2.9: Questions on Preferred Customer Support Assist for Product Selection

The following 3 questions aimed to gather information regarding the importance of customer service on the coffee shop website, as well as the preferable types of assistance that respondents would like to receive from customer support. By referring to Figure 4.2.7, 80% of the respondents find difficulty to determine whether the products is suitable for them while 20% do not face this issue. 90% of the respondents think that a customer support assistant would greatly help them in selecting a product that suits them while 10% think it is not essential based on the result in Figure 4.2.8. Moreover, based on Figure 4.2.9, it could also be seen that product recommendation is the most

in-demand assistance to be provided by the customer service, followed by preferences, usability and lastly product description.

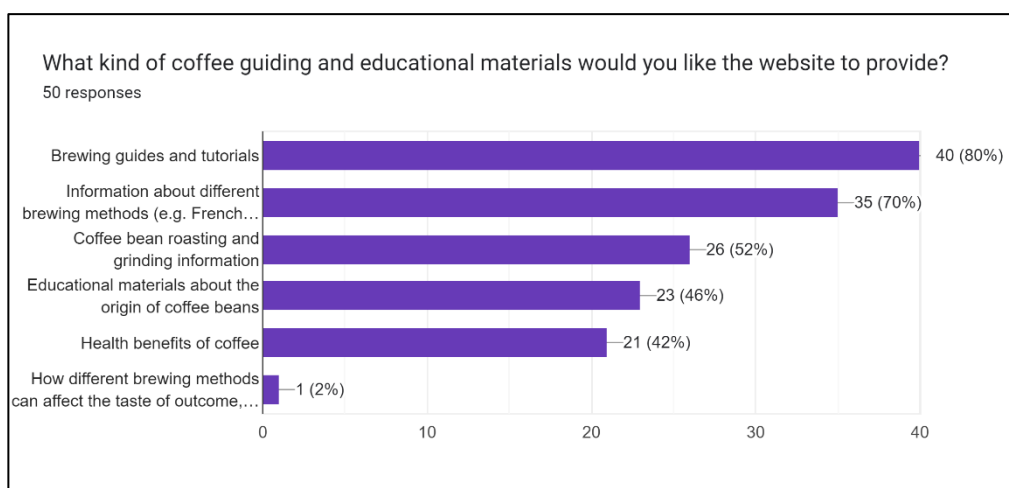


Figure 4.2.10: Distribution of preferred coffee guiding and educational materials

The next question aimed to determine the preferred coffee guiding and educational materials among the 50 respondents. Based on the results presented in Figure 4.2.10, the majority of the respondents expressed their interest in having brewing guides and tutorials available on the website, followed by information about different brewing methods, coffee bean roasting and grinding methods, educational materials on the origin of coffee beans, health benefits of coffee, and lastly, how different brewing methods could affect the taste of coffee, which was suggested by one of the respondents.

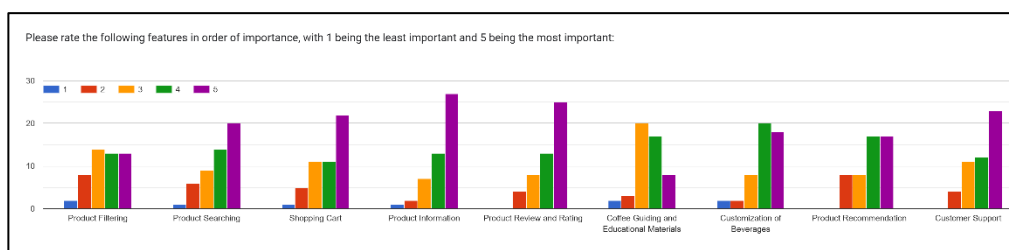


Figure 4.2.11: Distribution of Participants' opinion on the features provided on the coffee shop website

The last question in this survey aimed to determine the respondents' priorities regarding the features provided by the coffee shop website. The collected information is summarized in the table below, based on the number of respondents' preferences.

Table 4.2.1: Results of Participants' Prioritization

Features	Not Important	Less Important	Reasonably Important	Important	Very Important
Product Filtering	2	8	14	13	13
Product Searching	1	6	9	14	20
Shopping Cart	1	5	11	11	22
Product Information	1	2	7	13	27
Product Review and Rating	0	4	8	13	25
Coffee Guiding and Educational Materials	2	3	20	17	8
Beverage Customization	2	2	8	20	18
Product Recommend	0	8	8	17	17
Customer Support	0	4	11	12	23

4.3 Functional Requirements

The functional requirements for existing or new users:

1. The system shall allow existing users to login with their email and password.
2. The system shall allow new user to register a new user account.
3. The system shall allow both existing users and guests to view the product detail with the product recommendation and feedback and ratings provided on the product detail page.
4. The system shall allow both existing users and guests to make beverage customization.
5. The system shall allow only existing users with their account logged in to write a review and rate a product.
6. The system shall allow both existing users and guests to purchase the products on the website.
7. The system shall offer two types of shopping carts: one for beverages while another for homebrew products and machinery products.
8. The system shall allow both existing users and guests to manage the products in their shopping carts.
 - a. Users shall be able to view the items in their shopping carts.
 - b. Users shall be able to add the desired products into their shopping carts.
 - c. Users shall be able to update the quantity of the items in their shopping carts.
 - d. Users shall be able to delete the unwanted items from their shopping carts.
9. The system shall allow existing users to select payment method and make payment on the products or beverages they have selected.
10. The system shall allow existing users to view the order list on the products that payment has made.
11. The system shall allow existing users to make payments for orders where payment has not yet been completed.
12. The system shall send an order confirmation email to both existing users and guests once the payment has been completed.

13. The system shall allow both existing users and guests to search or filter the product list based on categories and price range provided.
14. The system shall allow both existing users and guests to view guiding materials.
15. The system shall allow both existing users and guests to initiate chat and ask questions to the chatbot.

The functional requirements for admin:

1. The system shall allow admins to login to their account.
2. The system shall allow admins to manage the products.
 - a. Admins shall be able to add new products.
 - b. Admins shall be able to view on the product lists
 - c. Admins shall be able to update the existing products
 - d. Admins shall be able to delete the unwanted products.
3. The system shall allow admins to manage the guiding materials.
 - a. Admins shall be able to add new guiding materials
 - b. Admins shall be able to view on the guiding material lists.
 - c. Admins shall be able to update on an existing guiding material.
 - d. Admins shall be able to delete an existing guiding material.
4. The system shall allow admins to manage data embedding file.
 - a. Admins shall be able to view on the data embedding file.
 - b. Admins shall be able to add new data embedding file.
 - c. Admins shall be able to update on the existing data embedding file.
 - d. Admins shall be able to delete the unwanted data embedding file.
5. The system shall limit the number of data embedding file to one file at any given time.
6. The system shall display a notification prompting administrators to delete the existing file when attempting to add another file while a file is already present.

The functional requirements for chatbot:

1. The chatbot shall serve as a customer support answering the questions based on what users asked.
2. The chatbot shall offer personalized product recommendations based on the criteria mentioned by the customers.
3. The chatbot shall be restricted from answering questions that are not in related field.

4.4 Non-Functional Requirements

The non-functional requirements of the system are outlined as below:

1. Performance
 - The loading time of the system should be quick and the response time upon user request should be no more than 3 second.
2. Usability
 - The website shall have a user-friendly interface and intuitive navigation, allowing users to easily move between pages.
3. Reliability
 - The website and chatbot shall be available and accessible to users for 24 hours a day and 7 days a week.
4. Security
 - The system shall protect the user credentials from unauthorized access.

4.5 Use Case Diagram

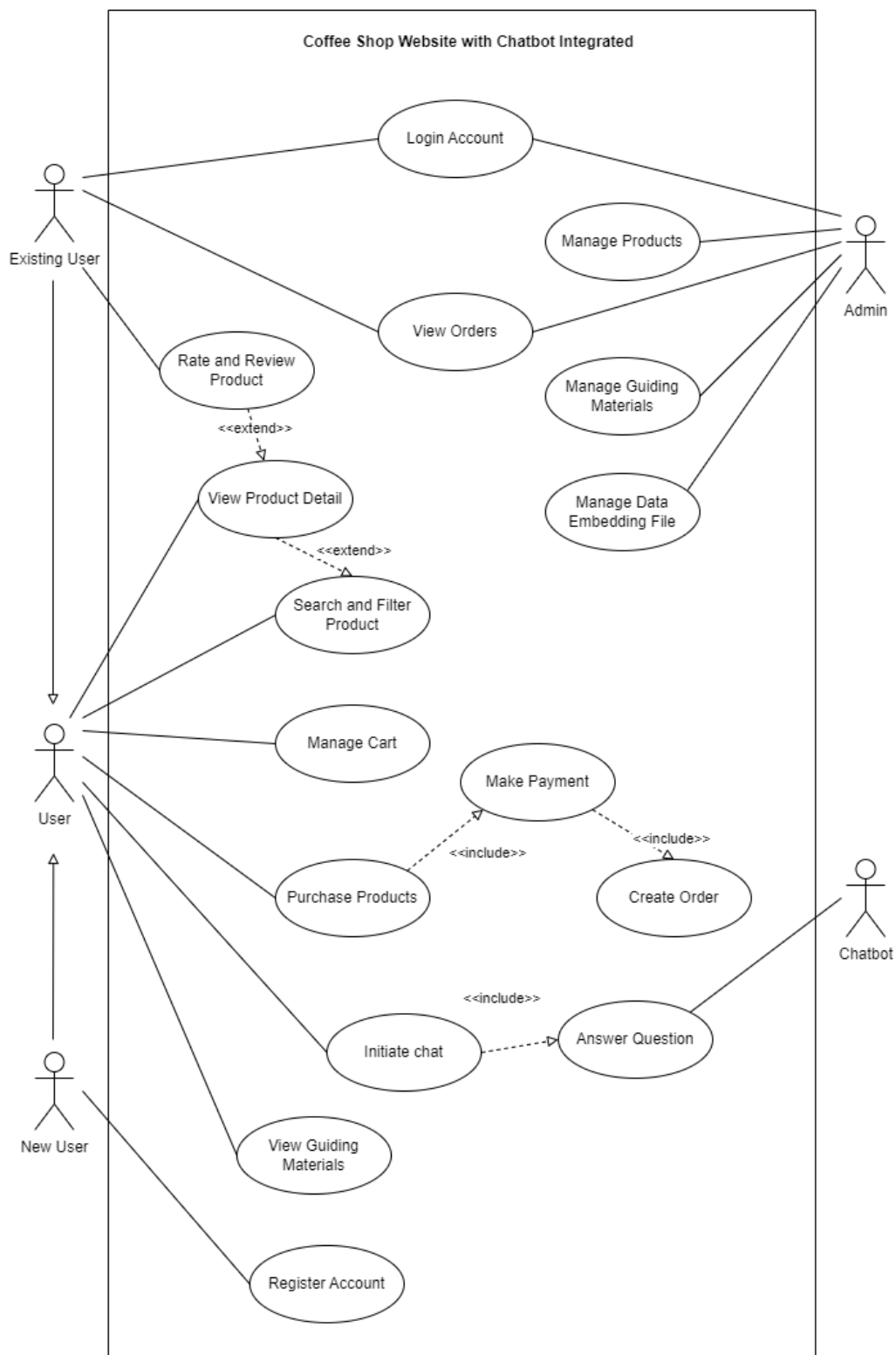


Figure 4.5.1: Use Case Diagram

4.6 Use Case Description

Table 4.6.1: Use Case Description – Register Account

Use Case Name: Register Account	ID: 1	Importance Level: High
Primary Actor: New user	Use Case Type: Detailed, Essential	
<p>Stakeholders and Interests:</p> <p>New user – An unregistered customer that intends to register for a new account on the coffee shop website.</p>		
<p>Brief Description: This use case describes how a new user create a new account on the coffee shop website.</p>		
<p>Trigger: When a new user wants to register for a new user account to access to the coffee shop website.</p>		
<p>Relationships:</p> <p>Association: New user</p> <p>Include: -</p> <p>Extend: -</p> <p>Generalization: -</p>		
<p>Normal Flow of Events:</p> <ol style="list-style-type: none"> 1. A new user clicks on the “Register” button displayed on the side bar of the home page of the website. 2. The system redirects the new user to the register page of the website. 3. The new user enters the required information such as username, email, and password stated on the registration form on the register page and submits it. 4. The system validates on the user inputs. 5. The system creates the user account based on the user input information provided. 		

6. The system redirects the newly registered user to the home page, with their account logged in.
Sub-flows: None
Alternative/Exceptional Flows: A-1: User inputs are invalid or existed. <ul style="list-style-type: none"> i. The system displays an error message indicating that the information either has already existed or the input is invalid. ii. Proceed Normal Flow 3 to re-enter the information required.

Table 4.6.2: Use Case Description – Login Account

Use Case Name: Login Account	ID: 2	Importance Level: High
Primary Actor: Existing user, Admin	Use Case Type: Detailed, Essential	
Stakeholders and Interests: Existing user – A registered customer that intends to login to his/her account on the coffee shop website to perform actions that require authentication. Admin – Staff from the coffee shop website that intends to login to his/her account to manage the website's operations.		
Brief Description: This use case describes how an existing user or admin login to their account on the coffee shop website.		
Trigger: When an existing user or admin wants to access to his/her own account on the coffee shop website.		

<p>Relationships:</p> <p>Association: Existing user, admin</p> <p>Include: -</p> <p>Extend: -</p> <p>Generalization: -</p>
<p>Normal Flow of Events:</p> <ol style="list-style-type: none"> 1. The system redirects the existing user or admin to the login page of the website. 2. The existing user or admin enters the required information, email and password stated on the login form on the login page and submits it. 3. The system validates on the user inputs. 4. The system redirects the existing user or admin to their respective home page after successful login.
<p>Sub-flows:</p> <p>None</p>
<p>Alternative/Exceptional Flows:</p> <p>A-1: User inputs are invalid or not exist.</p> <ol style="list-style-type: none"> i. The system displays an error message indicating that the information either does not exist or the input is invalid. ii. Proceed Normal Flow 3 to re-enter the information required.

Table 4.6.3: User Description – View Orders

Use Case Name: View Orders	ID: 3	Importance Level: High
Primary Actor: Existing user, Admin	Use Case Type: Detailed, Essential	
<p>Stakeholders and Interests:</p> <p>Existing user – A registered customer that intends to view on his/her own order list with detailed order items information.</p>		

Admin – Staff from the coffee shop website that intends to view on all order list with detailed order items information.
Brief Description: This use case describes how existing users and admins respectively view the order list.
Trigger: When an existing user or admin wants to check their order list respectively.
Relationships: Association: Existing user, admin Include: - Extend: - Generalization: -
Normal Flow of Events: <ol style="list-style-type: none"> 1. The system redirects to the order list page. 2. The system displays the order list respectively.
Sub-flows: S-1: Admin view order list <ol style="list-style-type: none"> 1. The system displays all order list. S-2: Existing user view order list <ol style="list-style-type: none"> 1. The system display the existing user’s order list.
Alternative/Exceptional Flows: None

Table 4.6.4: Use Case Description – Rate and Review Product

Use Case Name: Rate and Review Product	ID: 4	Importance Level: High
Primary Actor: Existing user	Use Case Type: Detailed, Essential	
Stakeholders and Interests:		

Existing user – A registered customer has logged in and intends to rate and write a review on a product.
Brief Description: This use case describes how an existing user rate and review on a product in the coffee shop website.
Trigger: When an existing user wants to provide feedback on a product to share with others in the coffee shop website.
Relationships: Association: Existing user Include: - Extend: - Generalization: -
Normal Flow of Events: <ol style="list-style-type: none"> 1. The system display the rate and review modal of the website. 2. The existing user writes his/her feedback and rates the product based on a 5 stars system and submits it. 3. The system validates on the user inputs and authenticity of the user. 4. The system redirects the existing user back to the product detail page of the website.
Sub-flows: None
Alternative/Exceptional Flows: A-1: User inputs are invalid. <ol style="list-style-type: none"> i. The system displays error message indicating the input is invalid. ii. Proceed Normal Flow 3 to re-enter the information required. A-2: User is not logged in. <ol style="list-style-type: none"> i. The system redirects the user to login page. A-3: User has rated on the product before <ol style="list-style-type: none"> i. The system displays message indicating each user can only write review on a product once.

Table 4.6.5: Use Case Description – View Product Detail

Use Case Name: View Product Detail	ID: 5	Importance Level: High
Primary Actor: User	Use Case Type: Detailed, Essential	
Stakeholders and Interests: User – New user or existing user that intends to view on the detailed information of a product.		
Brief Description: This use case describes how a user view the product detail information on the coffee shop website.		
Trigger: When a user wants to have a detailed understanding on a product in the coffee shop website.		
Relationships: Association: User Include: - Extend: Rate and review product Generalization: -		
Normal Flow of Events: <ol style="list-style-type: none"> 1. A user clicks on a product from the product list to view more information. 2. The system redirects the user to the product detail page of the website. 3. The system displays the detailed information of the selected product. 		
Sub-flows: S-1: Display Product Detail <ol style="list-style-type: none"> 1. The system displays the ratings and reviews of the selected product. 2. The system displays 3 recommended products from the same category. 		
Alternative/Exceptional Flows: A-1: User clicks on the “Write a Review” button.		

<p>i. Perform Rate and Review Product use case.</p> <p>A-2: User clicks on the “Add To Cart” button</p> <p>i. The system adds the product to the shopping cart.</p> <p>ii. The system displays a message indicating that the product has been successfully added to the shopping cart.</p>
--

Table 4.6.6: Use Case Description – View Guiding Materials

Use Case Name: View Guiding Materials	ID: 6	Importance Level: High
Primary Actor: User	Use Case Type: Detailed, Essential	
<p>Stakeholders and Interests:</p> <p>User – New user or existing user intends to view the guiding materials on the coffee shop website.</p>		
<p>Brief Description: This use case describes how a user view the guiding materials.</p>		
<p>Trigger: When a user seeks for guidance on the product they pruchased from the coffee shop website.</p>		
<p>Relationships:</p> <p>Association: User</p> <p>Include: -</p> <p>Extend: -</p> <p>Generalization: -</p>		
<p>Normal Flow of Events:</p> <ol style="list-style-type: none"> 1. A user clicks on the “Guides” button on the navigation bar of the website. 2. The system redirects the user to the guiding materials page. 3. The system displays a list of guiding materials related to the products selling in the website. 		

4. If the user clicks on a guiding material to view its detail, sub-flow S-1 will be performed.
Sub-flows: S-1: View Guiding Material Detail <ol style="list-style-type: none"> 1. The system redirects the user to the selected guiding material's detail page. 2. The system retrieves the information of the selected guiding material. 3. The system displays the information on the detail page.
Alternative/Exceptional Flows: None

Table 4.6.7: Use Case Description – Purchase Product

Use Case Name: Purchase product	ID: 7	Importance Level: High
Primary Actor: User	Use Case Type: Detailed, Essential	
Stakeholders and Interests: User – New user or existing user intends to purchase a product from the coffee shop website.		
Brief Description: This use case describes how a user purchase a product from the coffee shop website.		
Trigger: When a user wants to purchase a product from the coffee shop website.		
Relationships: Association: User Include: Make Payment Extend: - Generalization: -		

<p>Normal Flow of Events:</p> <ol style="list-style-type: none"> 1. A user clicks on the “View Complete Cart” button on the shopping cart modal. 2. The system redirects the user to his/her shopping cart. 3. The existing user clicks on the “Proceed To Checkout” button on the shopping cart page. 4. Perform Make Payment use case.
<p>Sub-flows:</p> <p>None</p>
<p>Alternative/Exceptional Flows:</p> <p>None</p>

Table 4.6.8: Use Case Description – Make Payment

Use Case Name: Make Payment	ID: 8	Importance Level: High
Primary Actor: User	Use Case Type: Detailed, Essential	
<p>Stakeholders and Interests:</p> <p>User – New user or existing user intends to make payment transaction to purchase products on the website.</p>		
<p>Brief Description: This use case describes how a user make payment for a product on the coffee shop website.</p>		
<p>Trigger: When a user wants to make payment for a product from the coffee shop website.</p>		
<p>Relationships:</p> <p>Association: User</p> <p>Include: Create Order</p> <p>Extend: -</p> <p>Generalization: -</p>		

<p>Normal Flow of Events:</p> <ol style="list-style-type: none"> 1. The system redirects user to the payment page. 2. User selects the payment method. 3. The existing user enters the payment information and clicks on the “Pay” button. 4. The system validates the inputs and payment method. 5. The system redirects user to payment successful page.
<p>Sub-flows:</p> <p>S-1: Redirects to payment page</p> <ol style="list-style-type: none"> i. The system integrates with Stripe payment gateway using secret key. ii. Perform Create Order use case.
<p>Alternative/Exceptional Flows:</p> <p>A-1: User inputs are invalid</p> <ol style="list-style-type: none"> i. The system displays an error message indicating that the input is invalid. ii. Proceed Normal Flow 2 to re-enter information required. <p>A-2: Payment information is cancelled</p> <ol style="list-style-type: none"> i. The system redirects user to payment failure page.

Table 4.6.9: Use Case Description – Create Order

Use Case Name: Create Order	ID: 9	Importance Level: High
Primary Actor: System	Use Case Type: Detailed, Essential	
<p>Stakeholders and Interests:</p> <p>System – System will create a new order when user is redirected to Stripe payment page.</p>		
<p>Brief Description: This use case describes how an order is created after the user is redirected to Stripe payment page.</p>		

<p>Trigger: When a user is redirected to the Stripe payment page.</p>
<p>Relationships:</p> <p style="padding-left: 40px;">Association: System</p> <p style="padding-left: 40px;">Include: -</p> <p style="padding-left: 40px;">Extend: -</p> <p style="padding-left: 40px;">Generalization: -</p>
<p>Normal Flow of Events:</p> <ol style="list-style-type: none"> 1. The system create an order database upon user redirecting to the Stripe payment page. 2. The system stores the order information into the order database.
<p>Sub-flows:</p> <p>S-1: Payment Successful</p> <ol style="list-style-type: none"> i. The system updates the order status to paid.
<p>Alternative/Exceptional Flows:</p> <p>A-1: Payment Unsuccessful or Cancel</p> <ol style="list-style-type: none"> i. The system will delete the order from database.

Table 4.6.10: Use Case Description – Search and Filter Product

Use Case Name: Search and Filter Product	ID: 10	Importance Level: High
Primary Actor: User	Use Case Type: Detailed, Essential	
<p>Stakeholders and Interests:</p> <p>User – New or existing user wants to filter and search for a product on the coffee shop website.</p>		
<p>Brief Description: This use case describes how a new or existing user filter and search for a product on the coffee shop website.</p>		

Trigger: When the website has too many products and user feels difficult to find the desired product.

Relationships:

Association: User

Include: -

Extend: View Product Detail

Generalization: -

Normal Flow of Events:

1. User inputs on the search textbox on the product list page.
2. User filters the product list based on price range and product categories.
3. The system validates the inputs provided.
4. The system displays a list of relevant products based on the search and filter conditions.

Sub-flows:

S-1: Filters by searching relevant keywords

1. User inputs the keywords in the search textbox and clicks on the “Apply” button.

S-2: Filters using price range

1. User selects the minimum and maximum price range and clicks on the “Apply” button.

S-3: Filters using product categories

1. User selects one category and clicks on the “Apply” button.

S-4: User submits empty search filter textbox

1. The system will remove the search filter.

S-5: User clicks on the “Reset” button on the price range filter

1. The system will remove the price range filter.

S-6: User selects “No Filter” option on the category filter

1. The system will remove the category filter.

Alternative/Exceptional Flows:

A-1: No related information found.

- i. The system displays a message indicating that no relevant product found.

A-2: User clicks on the product image.

- i. Perform View Product Detail use case.

Table 4.6.11: Use Case Description – Manage Cart

Use Case Name: Manage Cart	ID: 11	Importance Level: High
Primary Actor: User	Use Case Type: Detailed, Essential	
Stakeholders and Interests: User – New or existing user intends to manage his/her shopping carts.		
Brief Description: This use case describes how a new or existing user manage his/her shopping carts.		
Trigger: When user wants to perform changes on his/her shopping cart.		
Relationships: Association: User Include: - Extend: - Generalization: -		
Normal Flow of Events: <ol style="list-style-type: none"> 1. User wishes to add item to their product. <ol style="list-style-type: none"> a. User clicks on the “Add To Cart” button either on the product list page or the product detail page. b. The system store the items into relevant carts. c. The system updates the shopping cart modal and icon. 2. Admin wishes to update the item quantity in the relevant cart. <ol style="list-style-type: none"> a. The admin clicks on the “+” or “-” icon button for the selected item to make changes on the item quantity. 		

<ul style="list-style-type: none"> b. The system updates on the relevant prices of the item. c. The system updates on the shopping cart modal. <p>3. Admin wishes to delete an item from the shopping cart.</p> <ul style="list-style-type: none"> a. The admin clicks on the “x” button for the selected item on the shopping cart page. b. The system deletes the selected item in relevant database. c. The system updates on the shopping cart modal and icon.
<p>Sub-flows:</p> <p>None</p>
<p>Alternative/Exceptional Flows:</p> <p>None</p>

Table 4.6.12: Use Case Description – Initiate chat

Use Case Name: Initiate chat	ID: 12	Importance Level: High
Primary Actor: User	Use Case Type: Detailed, Essential	
<p>Stakeholders and Interests:</p> <p>User – New or existing user intends to initiate a chat with the chatbot on the coffee shop website</p>		
<p>Brief Description: This use case describes how a new or existing user initiate a chat with the chatbot on the coffee shop website.</p>		
<p>Trigger: When user has enquires and would like to seek for help from the coffee shop website.</p>		
<p>Relationships:</p> <p>Association: User</p> <p>Include: Answer Question</p> <p>Extend: -</p> <p>Generalization: -</p>		

<p>Normal Flow of Events:</p> <ol style="list-style-type: none"> 1. A user clicks on the chatbot icon on the home page of the website. 2. The system displays a chatbot floating modal. 3. The user enters the enquiries in the textbox provided. 4. The user click on the send icon in the textbox to initiate a chat with the chatbot. 5. The system validates the user input. 6. Perform Answer Question use case.
<p>Sub-flows:</p> <p>None</p>
<p>Alternative/Exceptional Flows:</p> <p>A-1: User input is empty.</p> <ol style="list-style-type: none"> i. The system displays an error message indicating that the textbox is required to fill in. ii. Proceed Normal Flow 4 to re-enter information required.

Table 4.6.13: Use Case Description – Answer Question

Use Case Name: Answer Question	ID: 13	Importance Level: High
Primary Actor: Chatbot	Use Case Type: Detailed, Essential	
<p>Stakeholders and Interests:</p> <p>Chatbot – serves as a customer support providing solutions to user’s enquiries on the coffee shop website.</p>		
<p>Brief Description: This use case describes how a chatbot provide solutions to user’s questions or enquiries on the coffee shop website.</p>		
<p>Trigger: When user sent enquires to seek for help from the chatbot on the coffee shop website.</p>		

<p>Relationships:</p> <p>Association: User</p> <p>Include: -</p> <p>Extend: -</p> <p>Generalization: -</p>
<p>Normal Flow of Events:</p> <ol style="list-style-type: none"> 1. The chatbot receives the enquiries from the user. 2. The chatbot processes the enquiries from the user. 3. The chatbot return answers related to the question.
<p>Sub-flows:</p> <p>S-1: Questions related with customer support</p> <ol style="list-style-type: none"> i. The chatbot responds and provides information based on the enquiries submitted. <p>S-2: Questions related with customized product recommendation.</p> <ol style="list-style-type: none"> 1. The chatbot responds and asks for a more detailed specification on the user preference. 2. The chatbot provides a best matched recommended products for the user based on the criterias from the user. <p>S-3: User asks unrelated questions</p> <ol style="list-style-type: none"> 1. The chatbot responds states that it does not know the answer.
<p>Alternative/Exceptional Flows:</p> <p>None</p>

Table 4.6.14: Use Case Description – Manage Product

Use Case Name: Manage Product	ID: 14	Importance Level: High
Primary Actor: Admin	Use Case Type: Detailed, Essential	
Stakeholders and Interests:		

Admin – Staff from the coffee shop website intends to manage the selling products.
Brief Description: This use case describes how an admin manage the products on the coffee shop website.
Trigger: When there are changes needed in the product.
Relationships: Association: Admin Include: - Extend: - Generalization: -
Normal Flow of Events: <ol style="list-style-type: none"> 1. The system redirects admin to manage product page. 2. Admin wishes to add new product. <ol style="list-style-type: none"> a. Admin clicks on the “Add New Product” button on the manage product page. b. The system redirects admin to the add product page. c. Admin enters the required product information and submits it. d. The system validates on the user input e. The system redirects the admin back to the manage product page. f. The system displays a message indicating that the product has added successfully. 3. Admin wishes to edit an existing product. <ol style="list-style-type: none"> a. Admin clicks on the “Edit” button for the selected product on the manage product page. b. The system redirects the admin to the edit product page. c. The admin updates on the product information and submits it. d. The system validates on the user input. e. The system redirects the admin back to the mange product page. f. The system displays a message indicating that the product has updated successfully.

<p>4. Admin wished to delete an existing product.</p> <p>a. Admin clicks on the “Delete” button for the selected product on the manage product page.</p> <p>b. The system deletes the selected product from the database. The system displays a message indicating that the product has deleted successfully.</p>
<p>Sub-flows: None</p>
<p>Alternative/Exceptional Flows: A-1: User inputs are invalid or empty.</p> <p>i. The system displays an error message indicating that the input is either invalid or the required field is empty.</p>

Table 4.6.15: Use Case Description – Manage Guiding Materials

Use Case Name: Manage Guiding Materials	ID: 15	Importance Level: High
Primary Actor: Admin	Use Case Type: Detailed, Essential	
<p>Stakeholders and Interests: Admin – Staff from the coffee shop website intends to manage the guiding materials on the website.</p>		
<p>Brief Description: This use case describes how an admin manage the guiding materials on the coffee shop website.</p>		
<p>Trigger: When changes needed in the guiding materials.</p>		

Relationships:

Association: Admin

Include: -

Extend: -

Generalization: -

Normal Flow of Events:

1. The system redirects admin to the manage guiding material page.
2. Admin wishes to add new guiding material.
 - a. Admin clicks on the “Add New Guiding Material” button on the manage guiding material page.
 - b. The system redirects admin to the add guiding material page.
 - c. Admin enters the required guiding material information and submits it.
 - d. The system validates on the user input.
 - e. The system redirects admin back to the manage guiding material page.
 - f. The system displays a message indicating that the guiding material has added successfully.
3. Admin wishes to update an existing guiding material.
 - a. The admin clicks on the “Edit” button for the selected guiding material on the manage guiding material page.
 - b. The system redirects the admin to the update guiding material page.
 - c. Admin updates on the guiding material information and submits it.
 - d. The system validates on the user input.
 - e. The system redirects admin back to the manage guiding material page.
 - f. The system displays a message indicating that the guiding material has updated successfully.
4. Admin wishes to delete an existing guiding material.

<ul style="list-style-type: none"> a. The admin clicks on the “Delete” button for the selected guiding material on the manage guiding material page. b. The system deletes the selected guiding material from the database. c. The system displays a message indicating that the guiding material has deleted successfully.
<p>Sub-flows:</p> <p>None</p>
<p>Alternative/Exceptional Flows:</p> <p>A-1: User inputs are invalid or empty.</p> <ul style="list-style-type: none"> i. The system display the error messages indicating that the input is either invalid or the required field is empty.

Table 4.6.16: Use Case Description – Manage Data Embedding File

Use Case Name: Manage Data Embedding File	ID: 16	Importance Level: High
Primary Actor: Admin	Use Case Type: Detailed, Essential	
<p>Stakeholders and Interests:</p> <p>Admin – Staff from the coffee shop website intends to manage the data embedding file for chatbot embedding purpose.</p>		
<p>Brief Description: This use case describes how an admin manage the data embedding file for chatbot embedding purpose.</p>		
<p>Trigger: When there are changes or updates on the information in data embedding file.</p>		

Relationships:

Association: Admin

Include: -

Extend: -

Generalization: -

Normal Flow of Events:

4. The system redirects admin to the manage data embedding file page.
5. Admin wishes to add new guiding material.
 - a. Admin clicks on the “Add New Embedding File” button on the manage data embedding file page.
 - b. The system redirects admin to the add embedding file page.
 - c. Admin upload the data embedding file.
 - d. The system redirects admin back to the manage data embedding file page.
 - e. The system displays a message indicating that the data embedding file has added successfully.
6. Admin wishes to update an existing data embedding file.
 - a. The admin clicks on the “Edit” button for the selected data embedding file on the manage guiding material page.
 - b. The system redirects the admin to the update data embedding file page.
 - c. Admin removes the existing data embedding file and uploads the new data embedding file.
 - d. The system displays a message indicating that the guiding material has updated successfully.
7. Admin wishes to delete an existing data embedding file.
 - a. The admin clicks on the “Delete” button for the selected data embedding file on the manage data embedding file page.
 - b. The system deletes the selected data embedding file from the database.
 - c. The system displays a message indicating that the data embedding file has deleted successfully.

Sub-flows:

None

Alternative/Exceptional Flows:

A-1: Invalid file type.

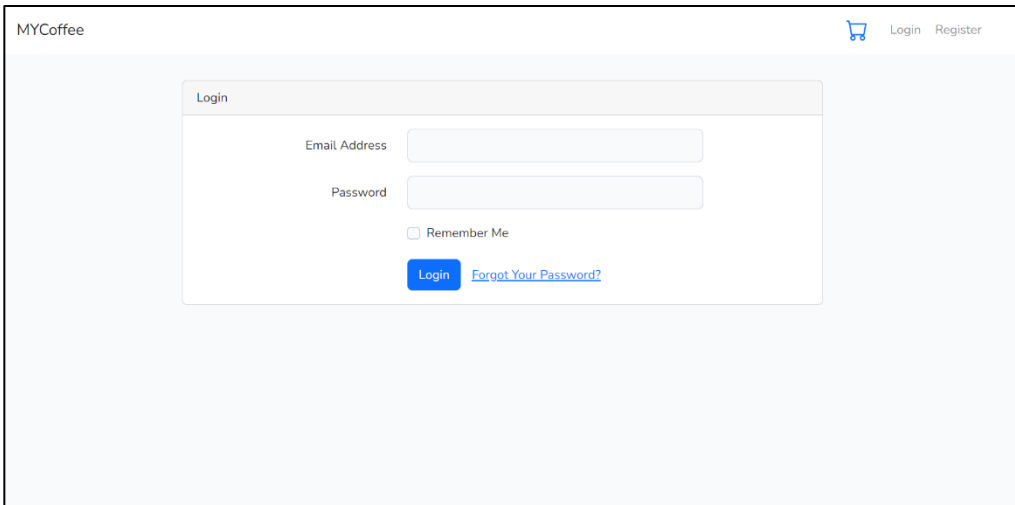
- i. The system display the error messages indicating that the input is either invalid or the required field is empty.

A-1: Attempt to add new data embedding file when there is existing file

- ii. The system displays an error message asking admin to delete existing file first before adding a new file.

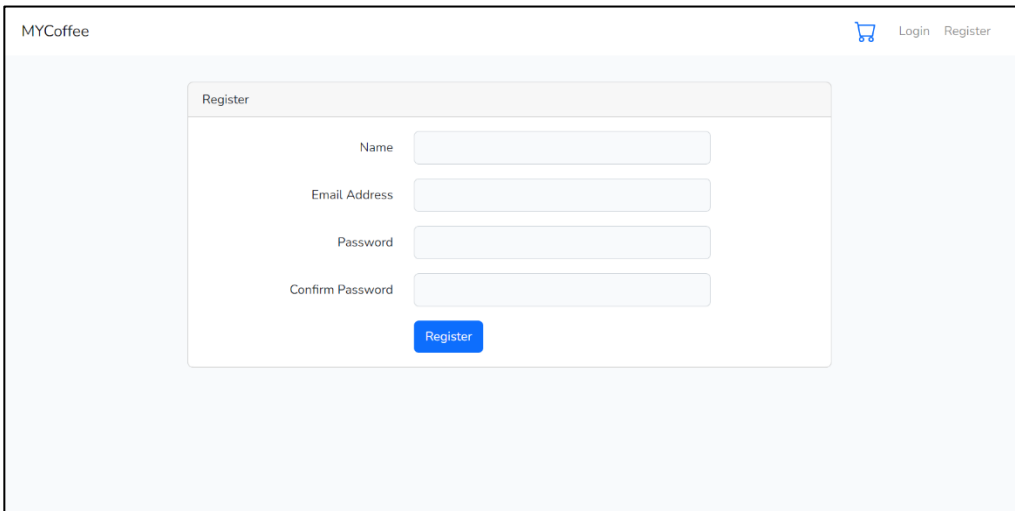
4.7 User Interface (UI) Prototype

The figures below are the UI design of the main modules, note that that this is the first prototype, and while all the features presented in each figure are included, their functionality has not been fully implemented yet. The primary focus of this prototype is to design the user interface and navigate between the user interfaces. The subsequent iterations worked on implementing the features and their functionality. Additionally, there are multiple types of products that fall under the same module, such as the Products module, which includes beverages, coffee products, and barista tools. In these cases, a summary of the prototype will be provided, which shows only one example of the product type UI from that module.



The image shows a web browser window with the URL 'MYCoffee'. In the top right corner, there is a shopping cart icon, a 'Login' link, and a 'Register' link. The main content area features a 'Login' form with a title bar. The form contains two input fields: 'Email Address' and 'Password'. Below these fields is a checkbox labeled 'Remember Me'. At the bottom of the form, there is a blue 'Login' button and a blue link labeled 'Forgot Your Password?'.

Figure 4.7.1: Prototype of Login Page UI



The image shows a web browser window with the URL 'MYCoffee'. In the top right corner, there is a shopping cart icon, a 'Login' link, and a 'Register' link. The main content area features a 'Register' form with a title bar. The form contains four input fields: 'Name', 'Email Address', 'Password', and 'Confirm Password'. At the bottom of the form, there is a blue 'Register' button.

Figure 4.7.2: Prototype of Register Page UI

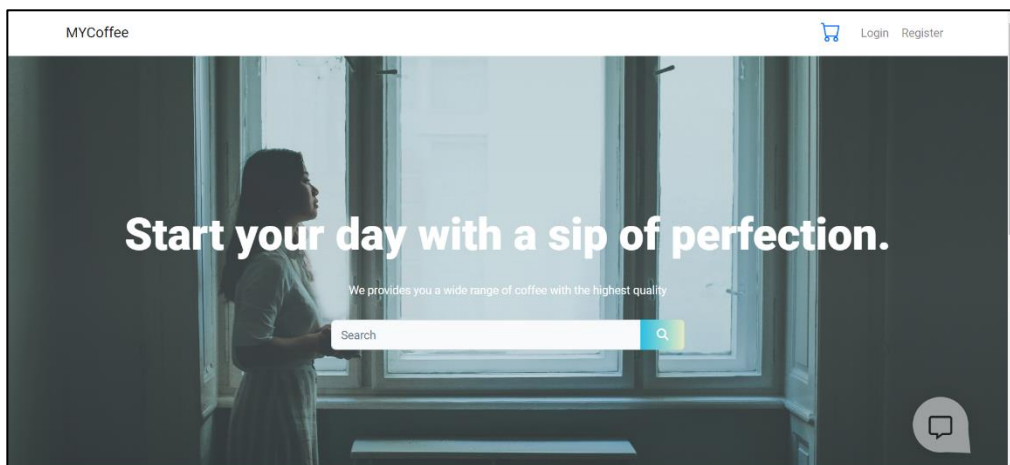


Figure 4.7.3: Prototype of Home Page with Search Function Provided UI

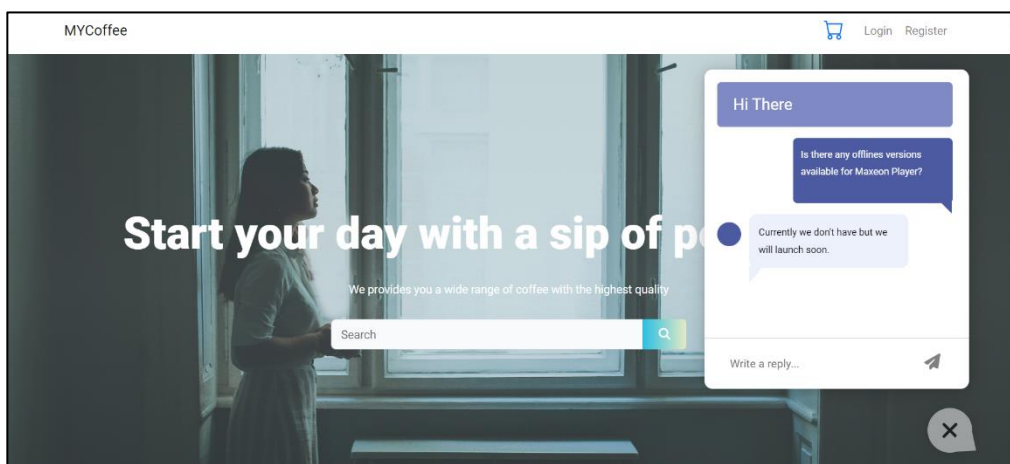


Figure 4.7.4: Prototype of chatbot integrated UI

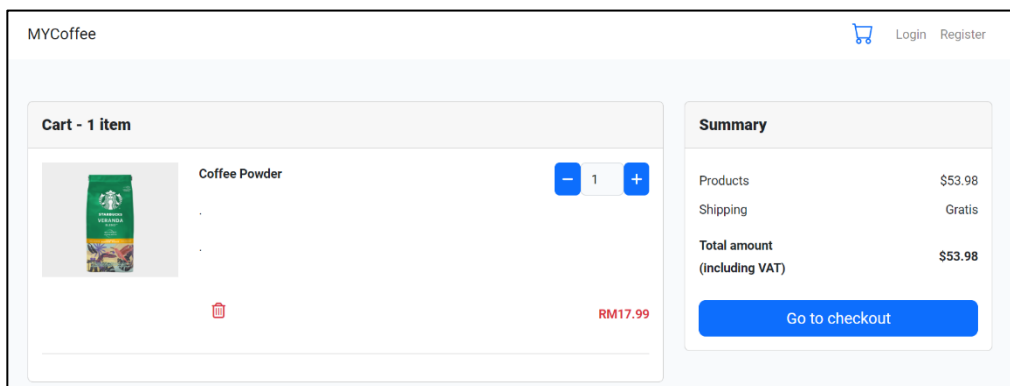


Figure 4.7.5: Prototype of Shopping cart UI

Payment

Credit Card [Paypal](#)

Cardholder Name

Card Number

Expiration Date CVV

[Submit Payment](#)

Figure 4.7.6: Prototype of Payment Module – Credit Card UI

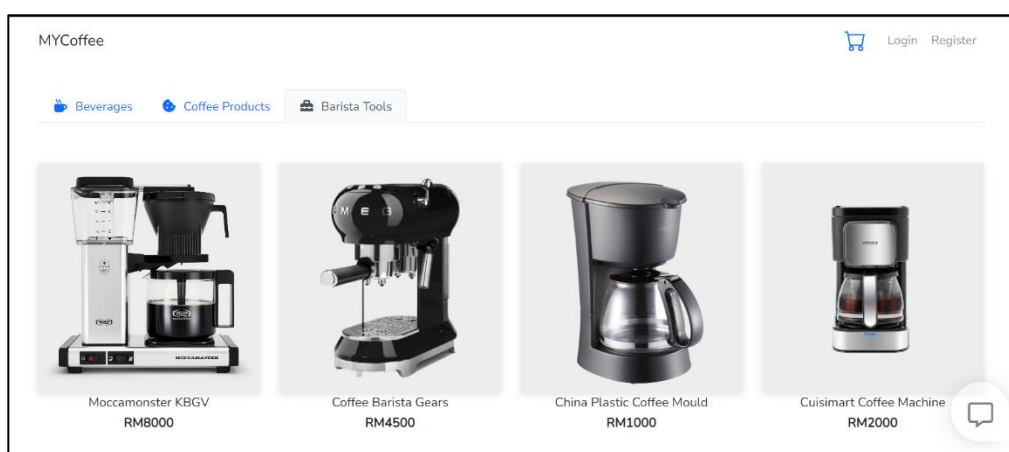


Figure 4.7.7: Prototype of the Product List – Barista Tools UI

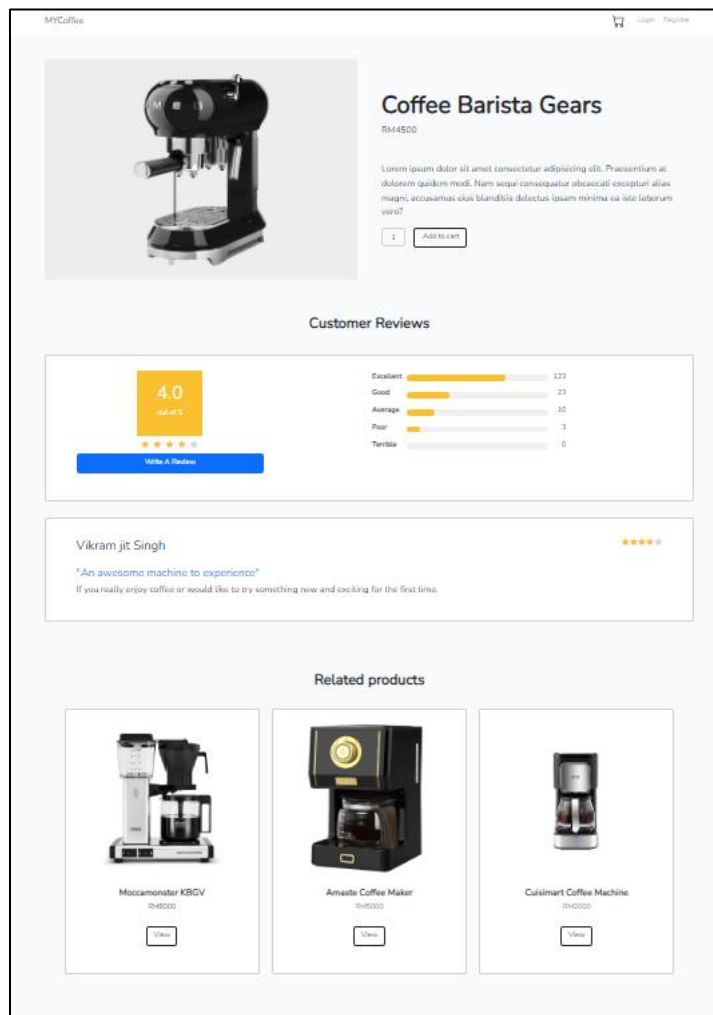


Figure 4.7.8: Prototype of Product Information – Barista Tools UI with product recommendations and product ratings and reiews included

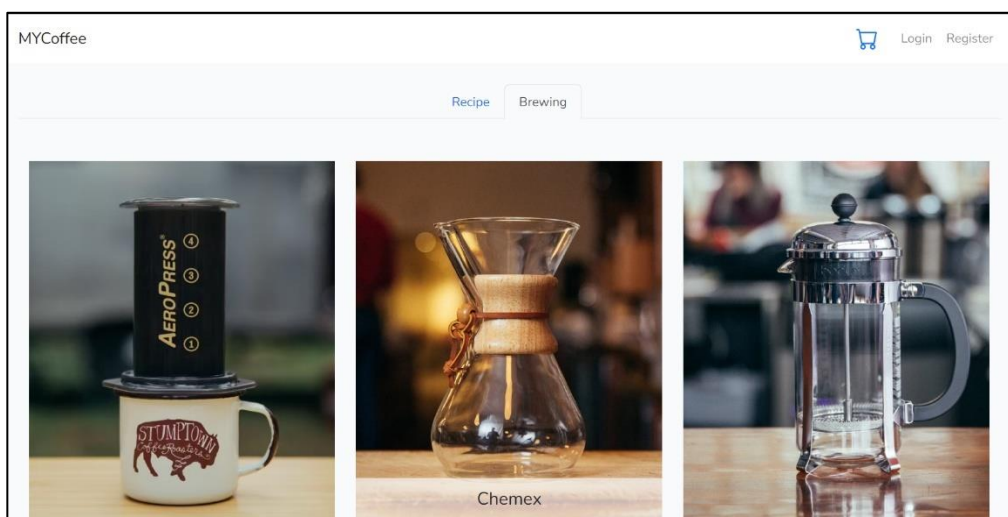


Figure 4.7.9: Prototype of Guiding Materials – Brewing Guides UI

MYCoffee Login Register

[Add New Product](#)

ID	Name	Stock	Price(RM)	Operations
1	Coffee Powder	78	17.99	View Edit Delete
2	Coffee Barista Gears	30	1799	View Edit Delete

Figure 4.7.10: Prototype of Product Management by Admin UI

CHAPTER 5

SYSTEM DESIGN

5.1 Introduction

In this chapter, the system design will be explained, encompassing the architecture layer design, data flow diagram, database schema, and overall UI design of each module.

5.2 System Architecture Design

In this project, a widely adopted three-tier architecture is utilized. The architecture comprises three primary layers: the presentation tier, responsible for user interface communication and interaction, collecting user inputs, and forwarding operations to the application tier. The application tier serves as the core of the application, processing operations from the presentation tier through business logic and directing changes to the data tier. Once the data tier responds, the application tier relays the response back to the presentation tier, effectively notifying users of their requests. Lastly, the data tier is accountable for storing and managing data, executing queries from the application tier to perform data updates. The three-tier architecture offers significant advantages in terms of flexibility and scalability as it isolates changes to one tier from affecting others. It also streamlines modifications and maintenance, thereby enhancing the system's long-term viability and adaptability. Furthermore, it ensures robust security management, allowing different tiers to require authorization from various user levels (Tie, Jin and Wang, 2011). By adopting this architecture, it could help the project to lay a solid foundation for the web-based coffee shop's efficient development and seamless operation.

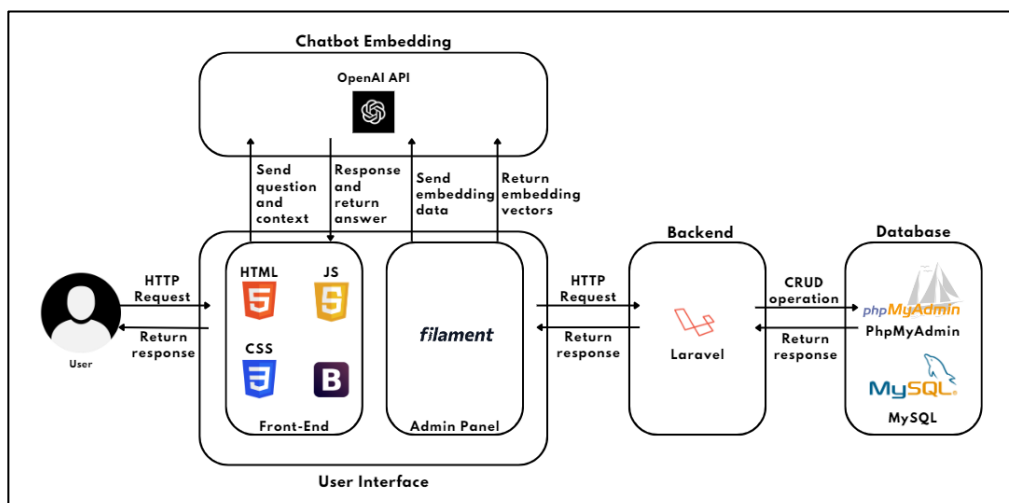


Figure 5.2.1: System Architecture Design

The overall system architecture, depicted in the figure above, consists of several key components within the presentation tier. These components are designed to enhance the user experience and streamline interactions with customers:

i. User Interface

The primary objective of the user interface is to create a user-friendly environment that ensures a seamless interaction with customers. It is meticulously crafted using a combination of web development tools, including HTML, CSS, JavaScript, and Bootstrap.

ii. Admin Panel

The admin panel is constructed using Filament, a Laravel package renowned for its simplicity and efficiency. Filament offers a robust administrative interface that fulfills essential requirements while maintaining ease of use. Notably, it simplifies the process of interacting between the admin panel and the CRUD operations, seamlessly integrating them with the underlying database tables.

iii. Chatbot Integration

The chatbot is a pivotal component which comprises two distinct modules. The first module involves the interaction between the admin panel and OpenAI by transmitting the embedding data file to generate context vectors. The second

module focuses on question answering. Within the user interface, customers can submit questions in the chatbot pop up window, initiating a dynamic interaction with the OpenAI API to return answers based on the context from the data embedding file.

Next, the application tier, primarily uses Laravel framework, serves as the intermediary layer that handles incoming user interface requests. It conducts CRUD operations on the database as per the requests passed. Once these operations are completed, the application tier promptly communicates the outcome, indicating either success or failure, back to the user interface.

Lastly, the data tier, primarily driven by MySQL as the database system and managed through PhpMyAdmin, plays a central role in the project by managing and executing data modifications within the database. This tier is dedicated to ensuring data integrity, handling data changes, and supporting the overall data management aspects of the project.

5.3 Chatbot Data Embedding Interaction with OpenAI

When talking about OpenAI, ChatGPT will always be there as it was the most popular precise search engine right now. ChatGPT brings forth a wide array of advantages, including a substantial enhancement in efficiency. For instance, in the business domain, it excels at promptly and precisely generating answers to user queries, ultimately contributing to an optimal user experience (Deng and Lin, 2022). However, it's important to acknowledge that ChatGPT does come with certain limitations. One notable limitation is its inability to inquire or interact with proprietary or personalized data. Hence, OpenAI has introduced the Embedding API, primarily designed for generating numerical vectors that represent text. These vectors serve as a numerical format that can be easily understood by the computer. Leveraging these numerical representations as a context, when there is a question related to the context, the system can identify the most similar data, which subsequently forms the basis for providing precise answers to user queries.

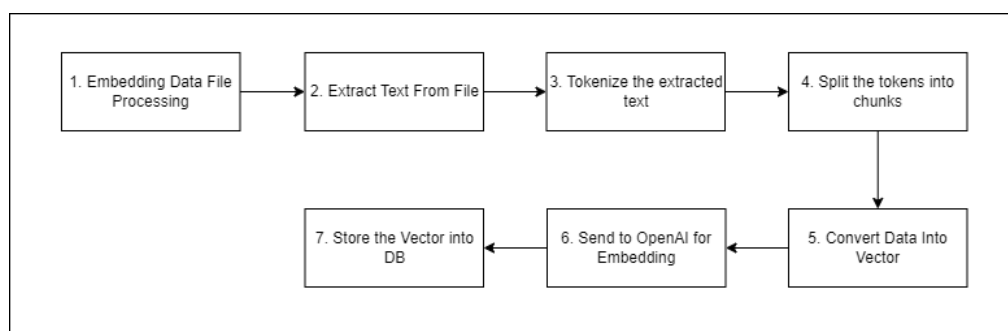


Figure 5.3.1: Chatbot Data Embedding Interaction Architecture

The figure above provides an overarching architectural overview of the process for preparing embedding data used by the chatbot. When an admin uploads the embedding data file, a series of data processing steps is initiated. Firstly, the system extracts the text content from the uploaded file. Subsequently, it proceeds to perform tokenization and normalization on the extracted text. During tokenization, the text is divided into individual tokens or units of meaning, facilitating further analysis. Simultaneously, normalization processes like lowercasing and punctuation removal are applied to standardize the text. To facilitate interaction with the OpenAI Embedding API, the tokens need to be segmented into multiple chunks to ensure they don't exceed the API's token limit, which is set at 8000 tokens. Going beyond this limit would trigger an error. Once the chunking process is completed, the individual chunks are sent one by one to the OpenAI Embedding API, which generates numerical vectors corresponding to each chunk. These vectors represent the contextual information needed for the chatbot's responses. Finally, the generated vectors are saved and stored in the project website's database. These vectors serve as context for the chatbot, enabling it to provide accurate and context-aware responses to user inquiries.

5.4 Chatbot Question and Answering Interaction with OpenAI

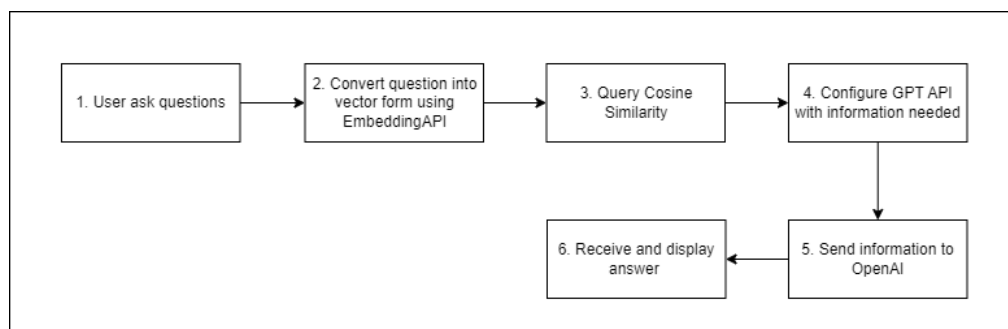


Figure 5.4.1: Chatbot Question and Answering Interaction Architecture

Once users upload or update the data embedding file, they gain the ability to ask questions through the website's chatbot pop-up window. The flow of this question-and-answer process is illustrated in the diagram above. When a user submits a question, the system engages in a series of steps to provide a meaningful response.

Firstly, the system reaches out to the Embedding API to transform the user's question text into a vector format. Following this, the system computes the cosine similarity score, which helps identify the most relevant context within the available data for answering the question effectively.

Next, the system configures the information needed for the OpenAI GPT API. This configuration includes specifying the GPT model to use, constructing an appropriate system prompt, providing the user's question, and context data. This meticulous setup ensures that the GPT model generates a response that is contextually accurate and mimics human-like behavior.

The system then communicates with the OpenAI GPT API, forwarding the prepared information for response generation. Once the response is generated, it is displayed within the chatbot conversation. This process ensures that users receive precise and contextually relevant answers to their queries, enhancing their interaction with the chatbot on the website and facilitating a more engaging and informative user experience.

5.5 Database Architecture Design

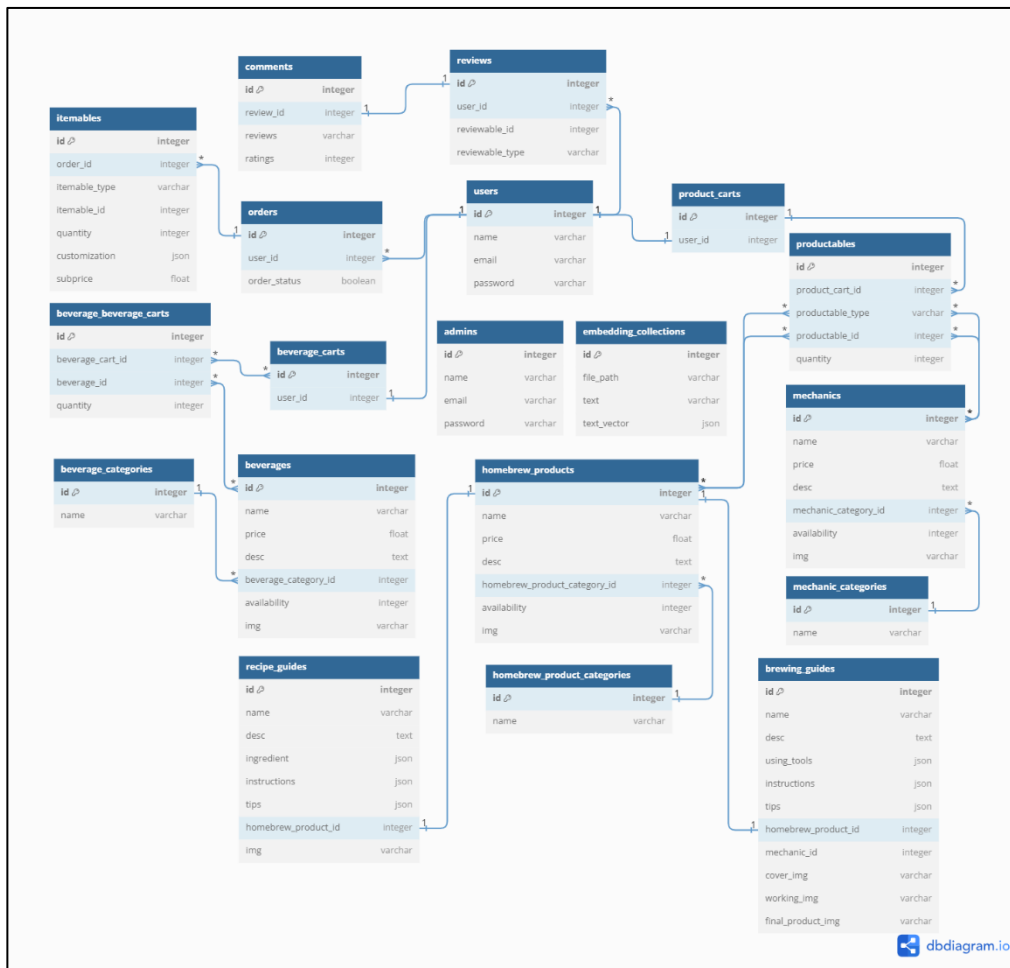


Figure 5.5.1: Database Architecture Design

Table Name	Description
Users	Stores user details such as name, email, and password.
Admins	Stores admin details such as name, email, and password.
Embedding_collections	Stores the data for chatbot embedding, which includes the data in text form, the file path, and the data in vector form.
Beverages	Stores beverage product details, including the name, image, price, description, etc.

Homebrew_products	Stores homebrew product details, including the name, image, price, description, etc.
Mechanics	Stores machines and tools product details, including the name, image, price, description, etc.
Beverage_categories	Stores the beverage category details.
Homebrew_product_categories	Stores the homebrew product category details.
Mechanic_categories	Stores the machines and tools category details.
Brewing_guides	Stores the information of the brewing guide details.
Recipe_guides	Store the information of the recipe guide details.
Product_carts	Stores the primary key of each authenticated user.
Productables	Stores the information of the product cart items which includes the homebrew products and other machines or tools products.
Beverage_carts	Stores the primary key of each authenticated user.
Beverage_beverage_carts	Stores the information of the beverage items
Orders	Stores the order details such as order type, total price, etc.
Itemables	Stores the information of the order items details.
Reviews	Stores the related primary and foreign keys of the review.
Comments	Stores the information of comments and ratings for a product.

Table 5.5.1: Database Architecture Design Table

5.6 Data Flow Diagram

5.6.1 Context Diagram

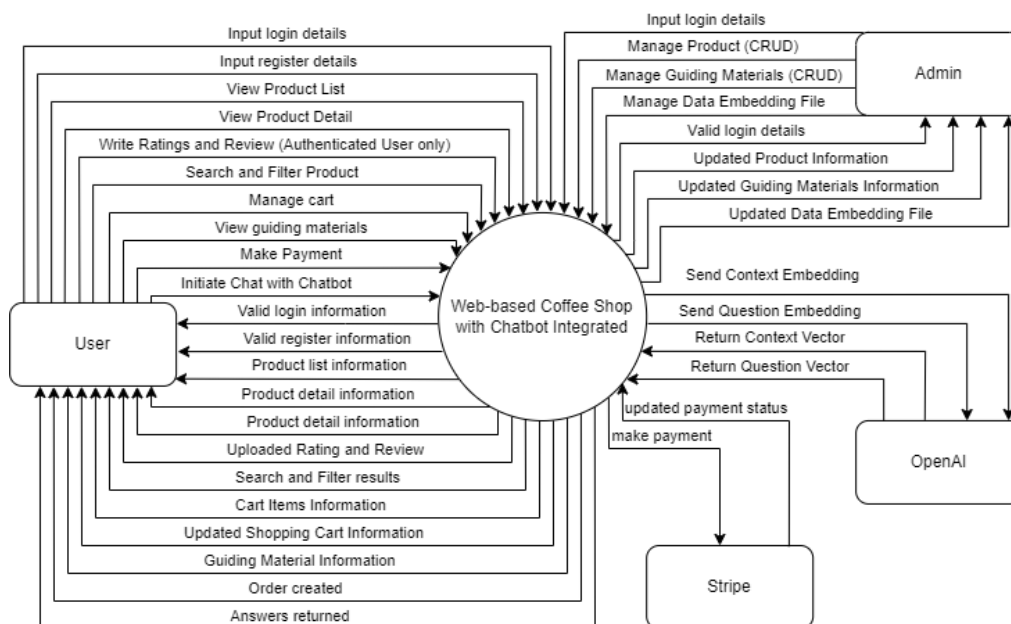


Figure 5.6.1: Context Diagram

5.6.2 Level 1 Data Flow Diagram

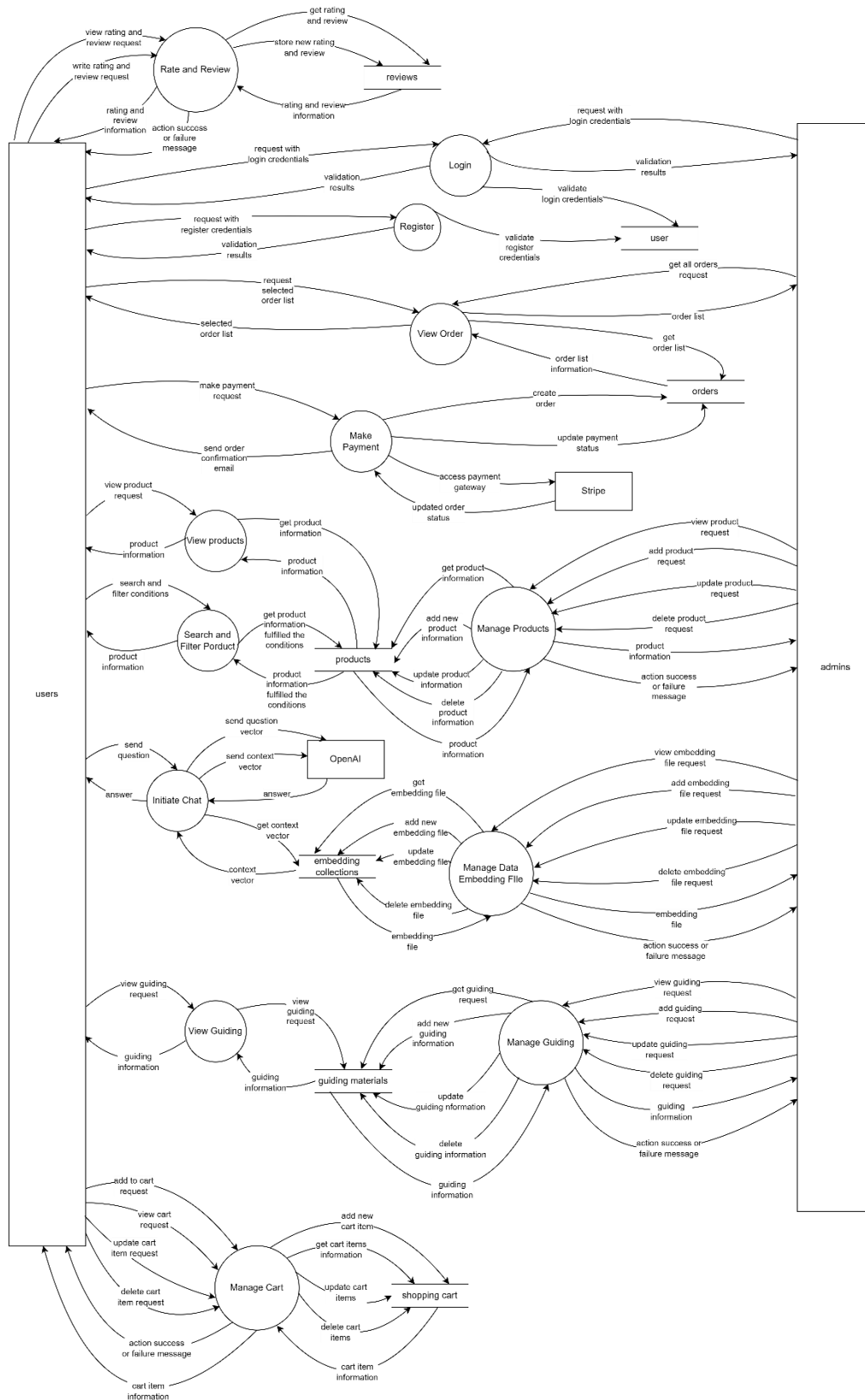


Figure 5.6.2: Level 1 Data Flow Diagram

5.6.3 Activity Diagram

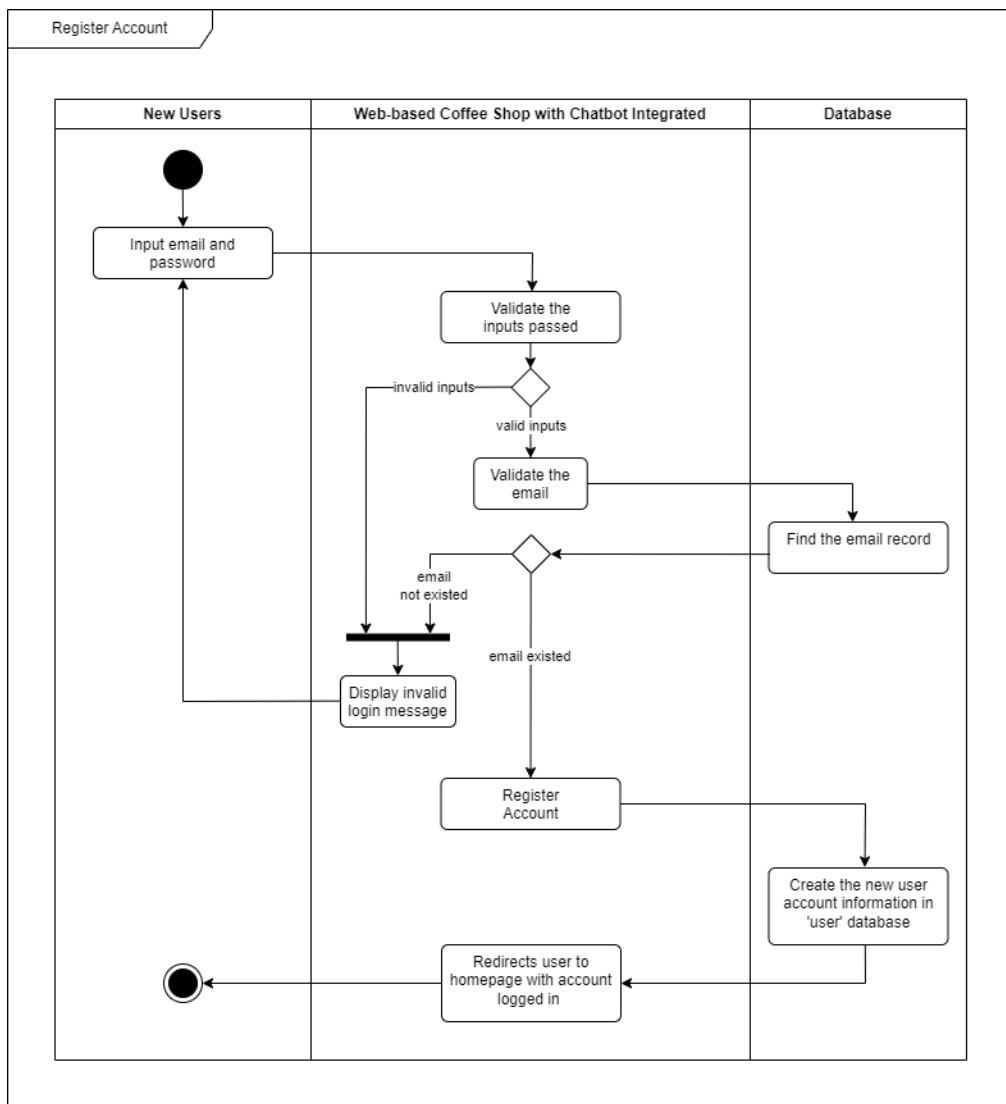


Figure 5.6.3: Activity Diagram – Register Account

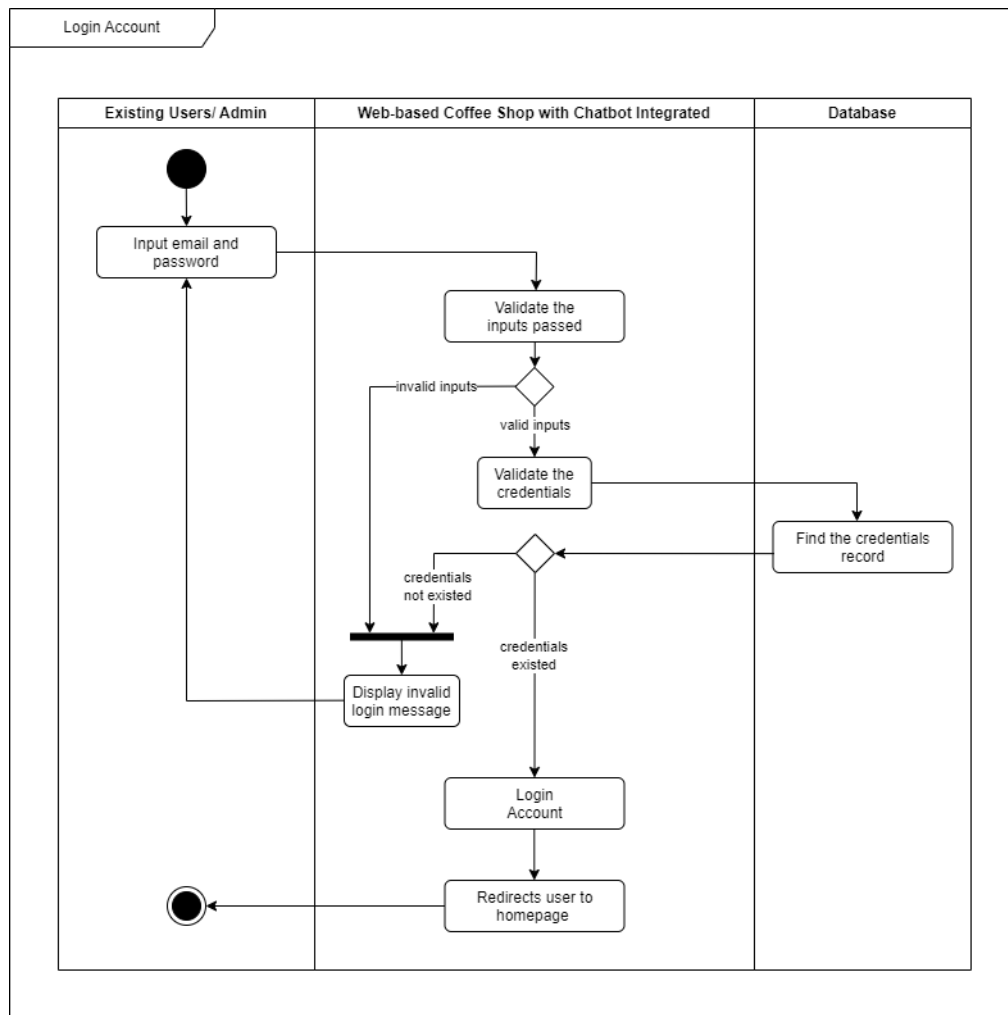


Figure 5.6.4: Activity Diagram – Login Account

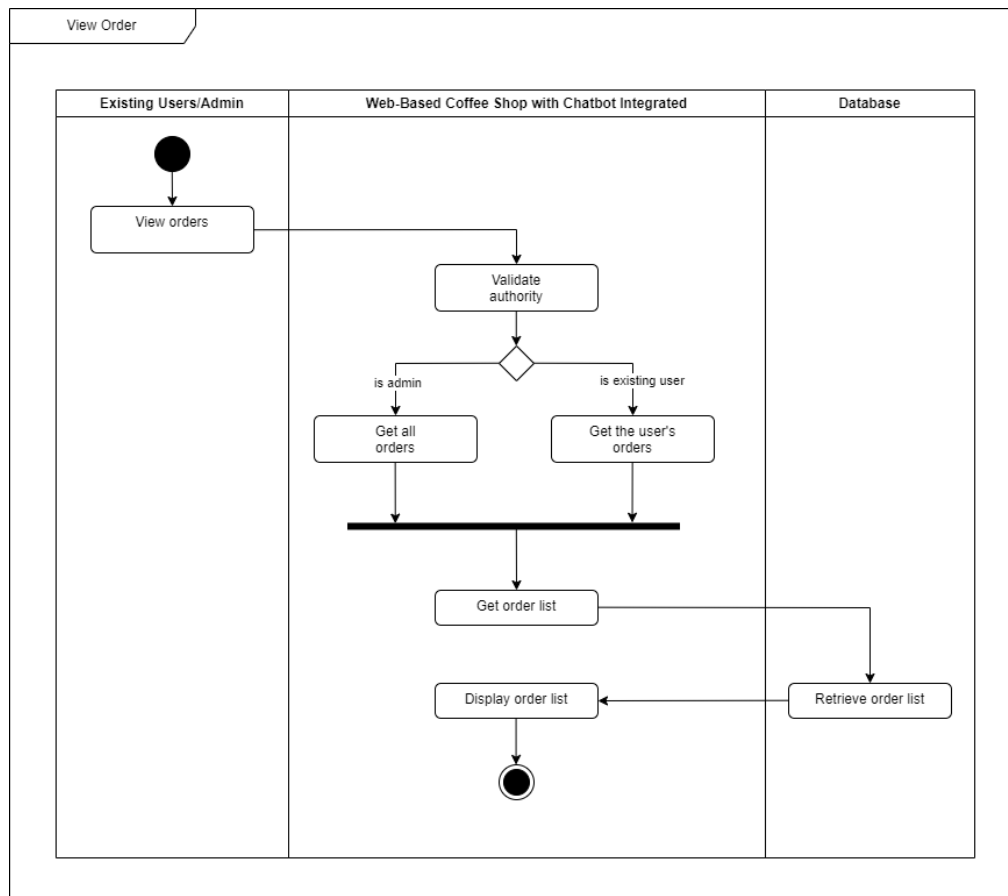


Figure 5.6.5: Activity Diagram – View Order

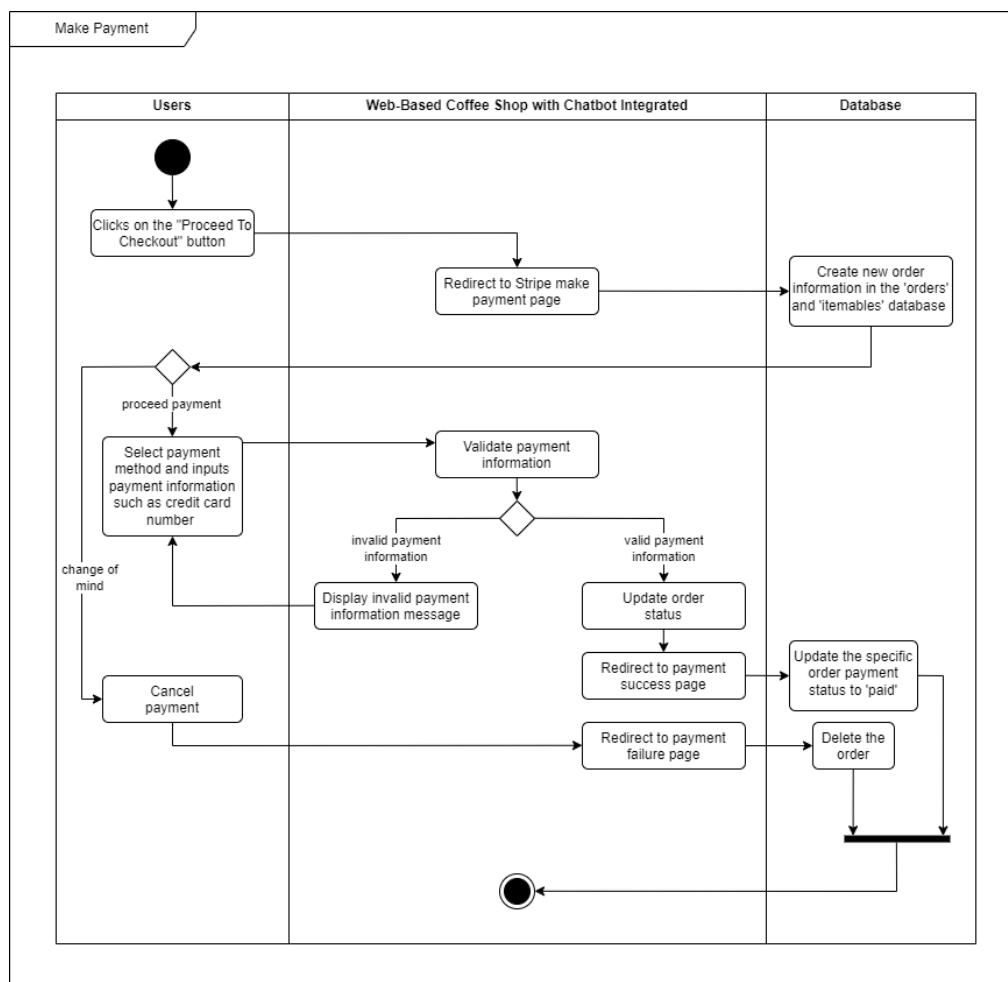


Figure 5.6.6: Activity Diagram – Make Payment

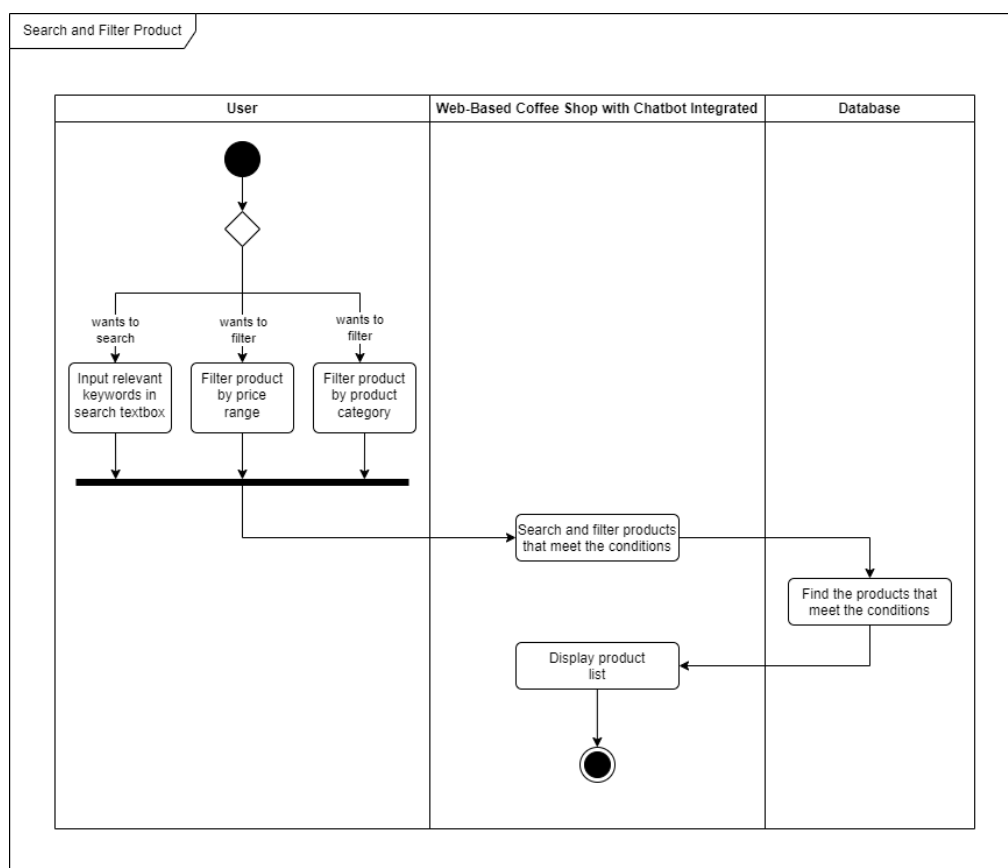


Figure 5.6.7: Activity Diagram – Search and Filter Product

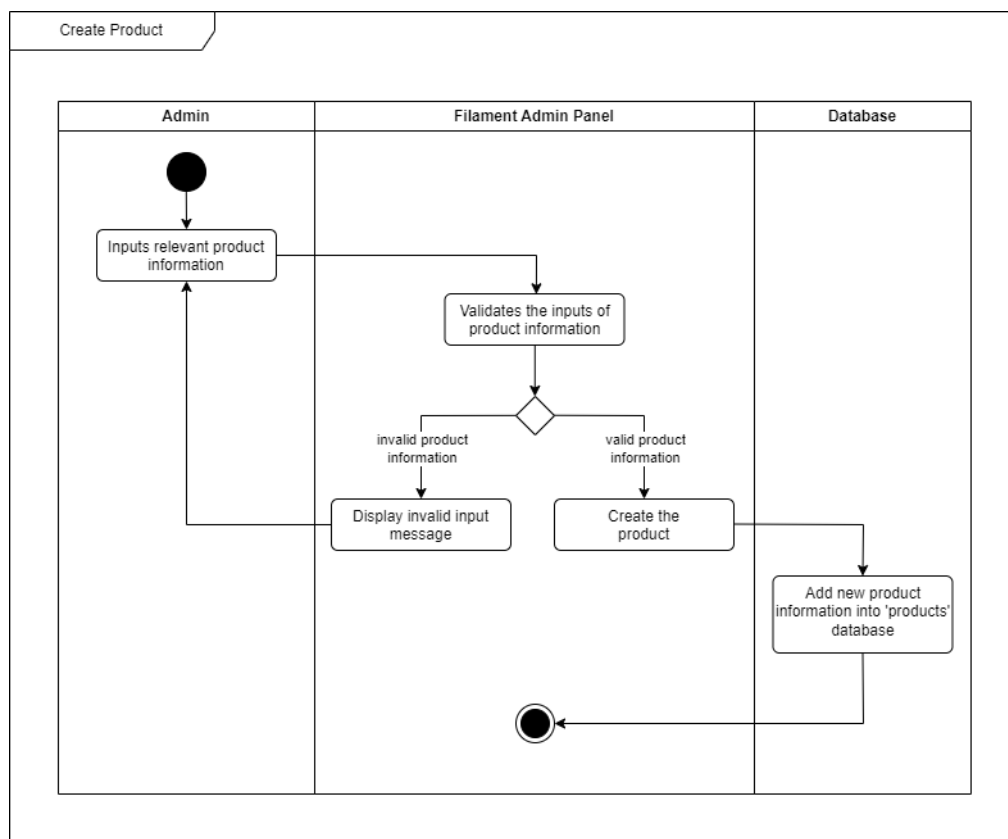


Figure 5.6.8: Activity Diagram – Create Product

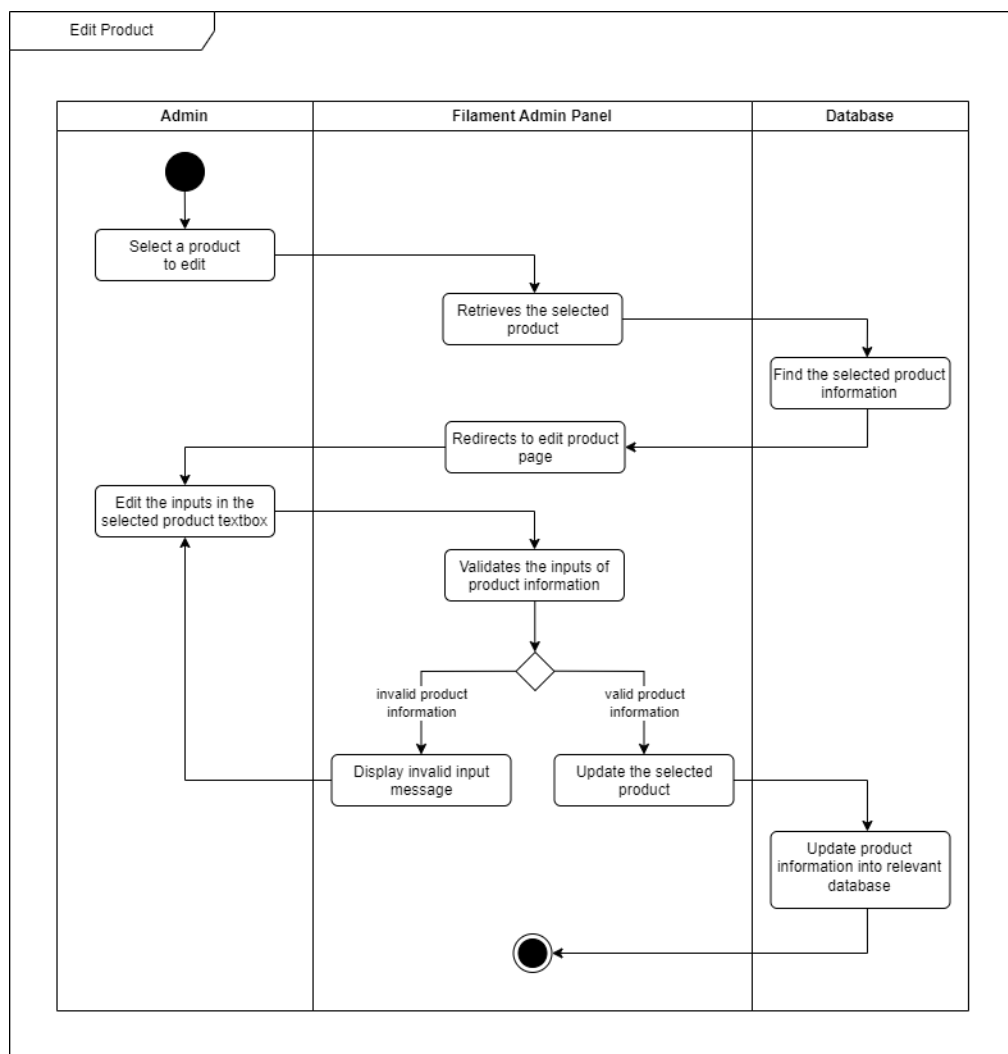


Figure 5.6.9: Activity Diagram – Edit Product

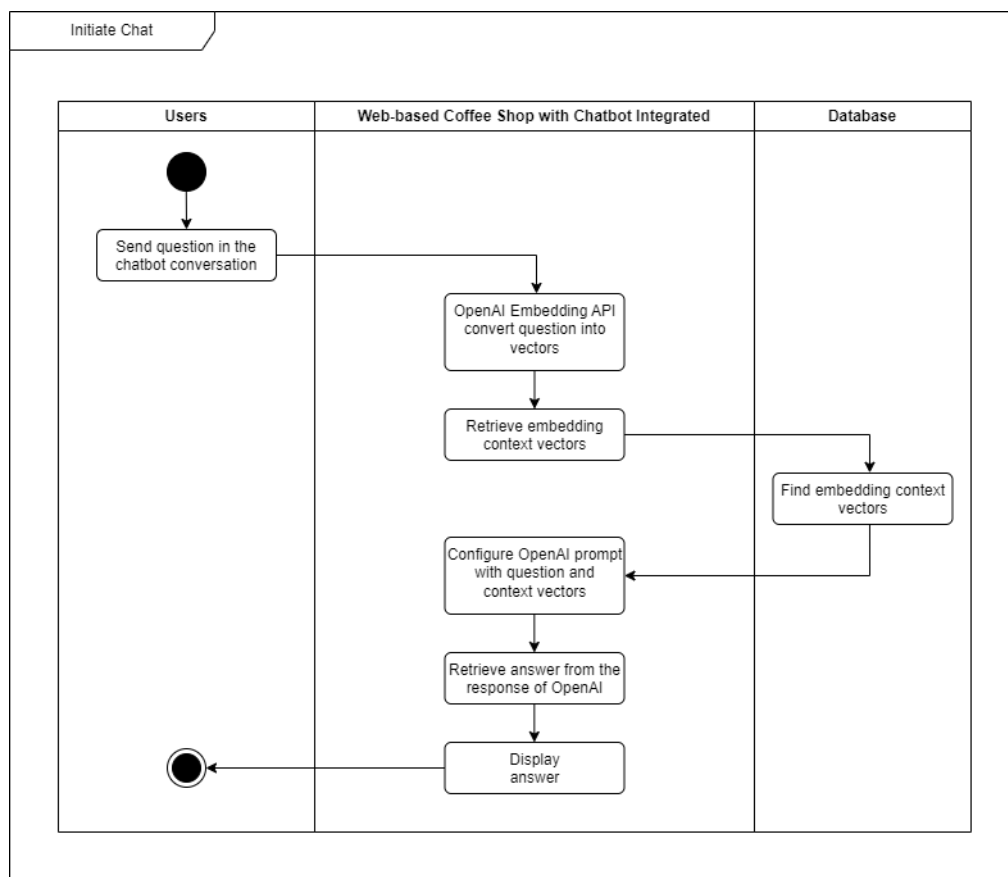


Figure 5.6.10: Activity Diagram – Initiate Chat

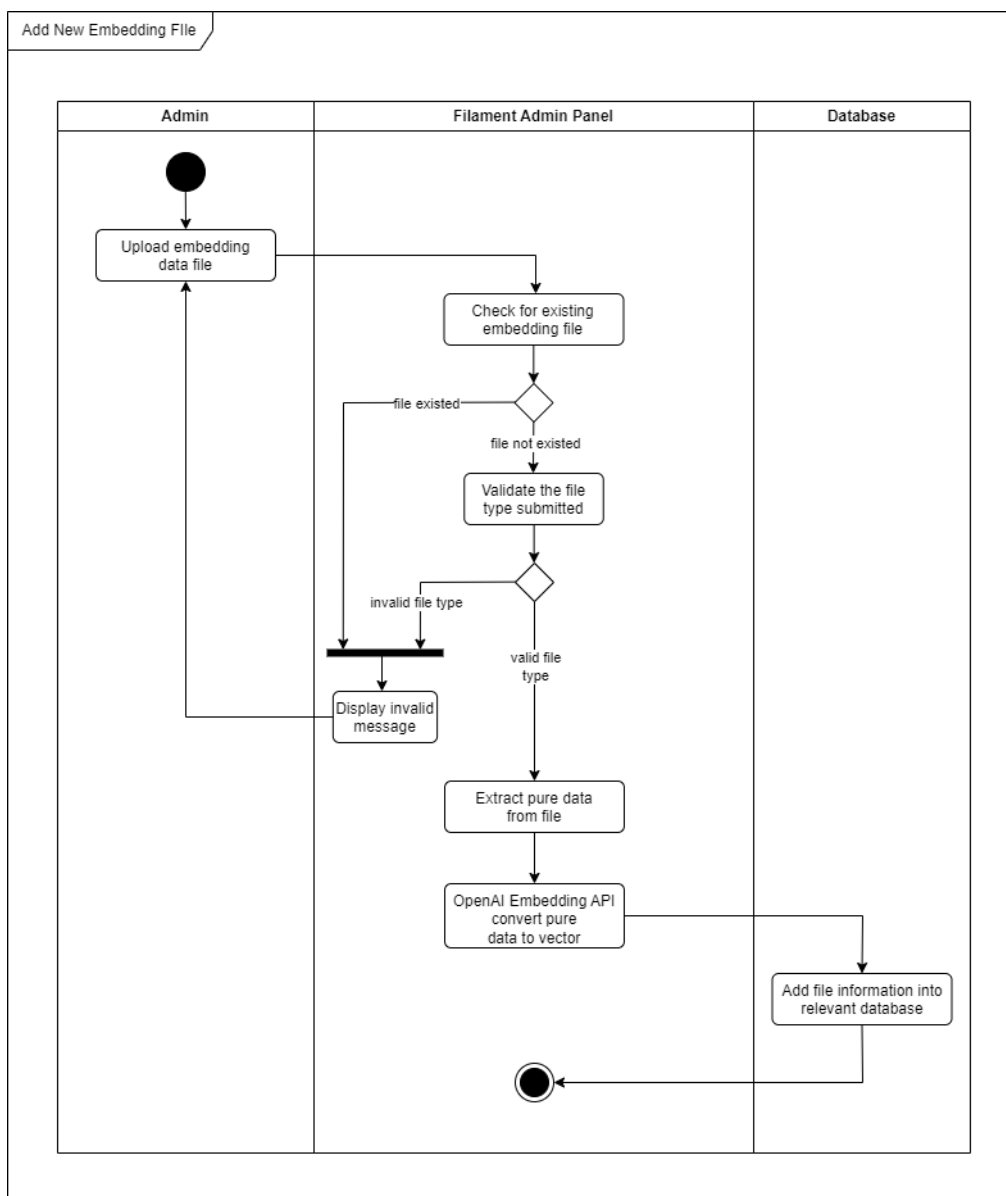


Figure 5.6.11: Activity Diagram – Add New Embedding File

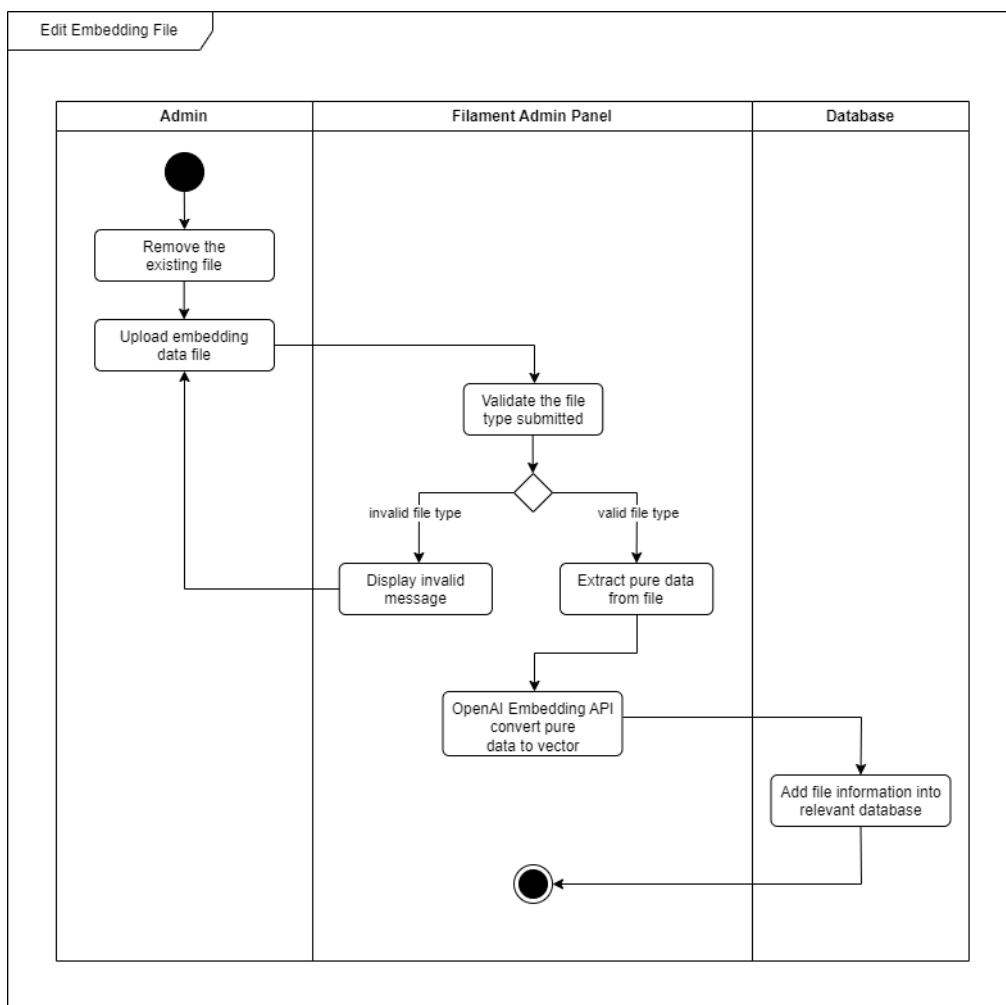


Figure 5.6.12: Activity Diagram – Edit Embedding File

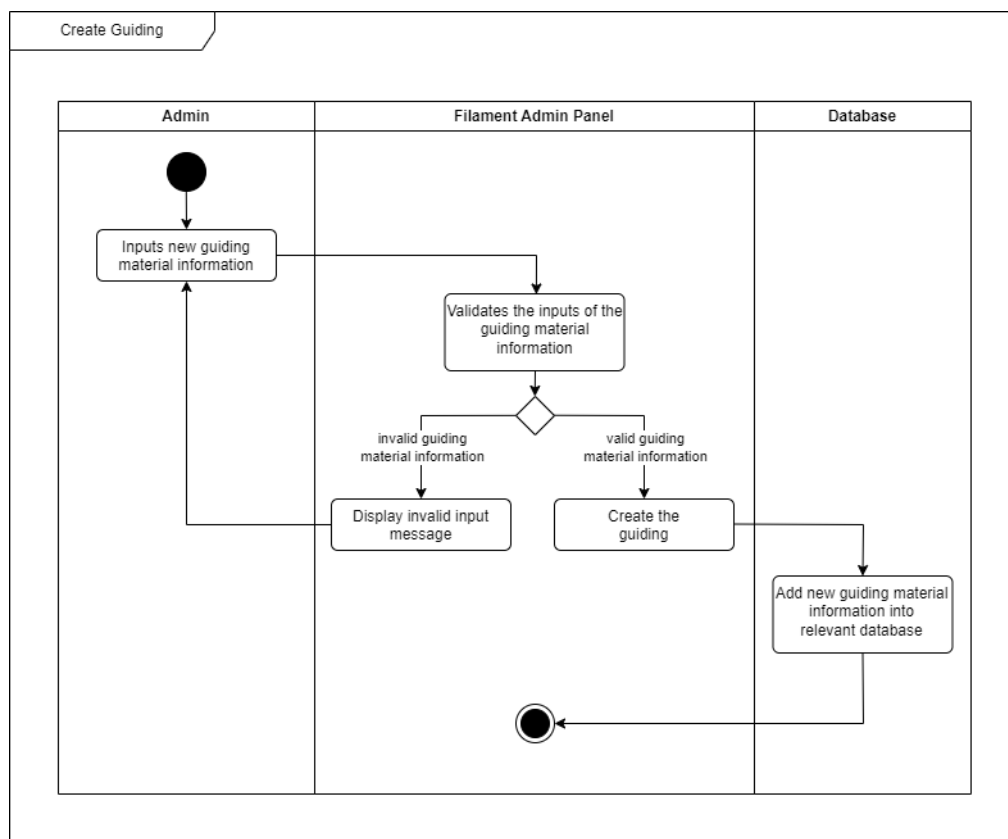


Figure 5.6.13: Activity Diagram – Create Guiding

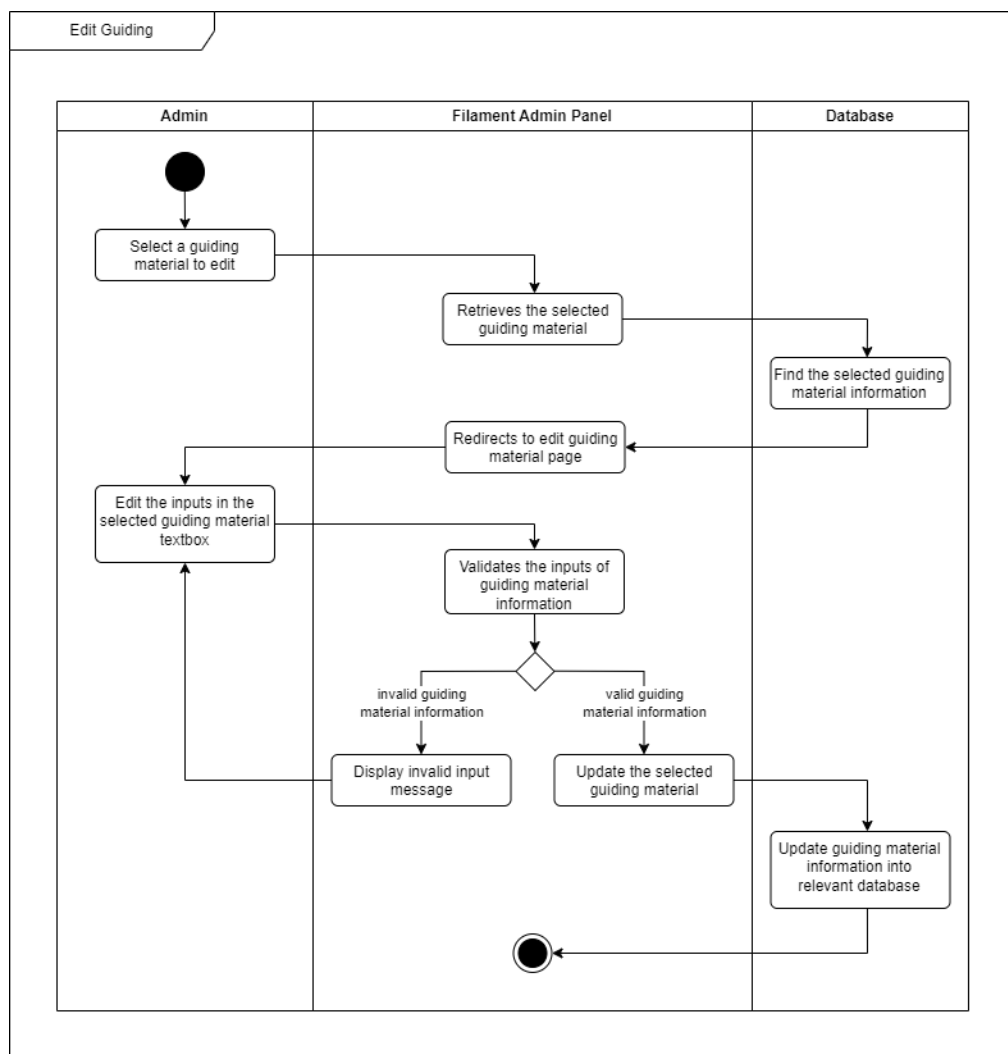


Figure 5.6.14: Activity Diagram – Edit Guiding

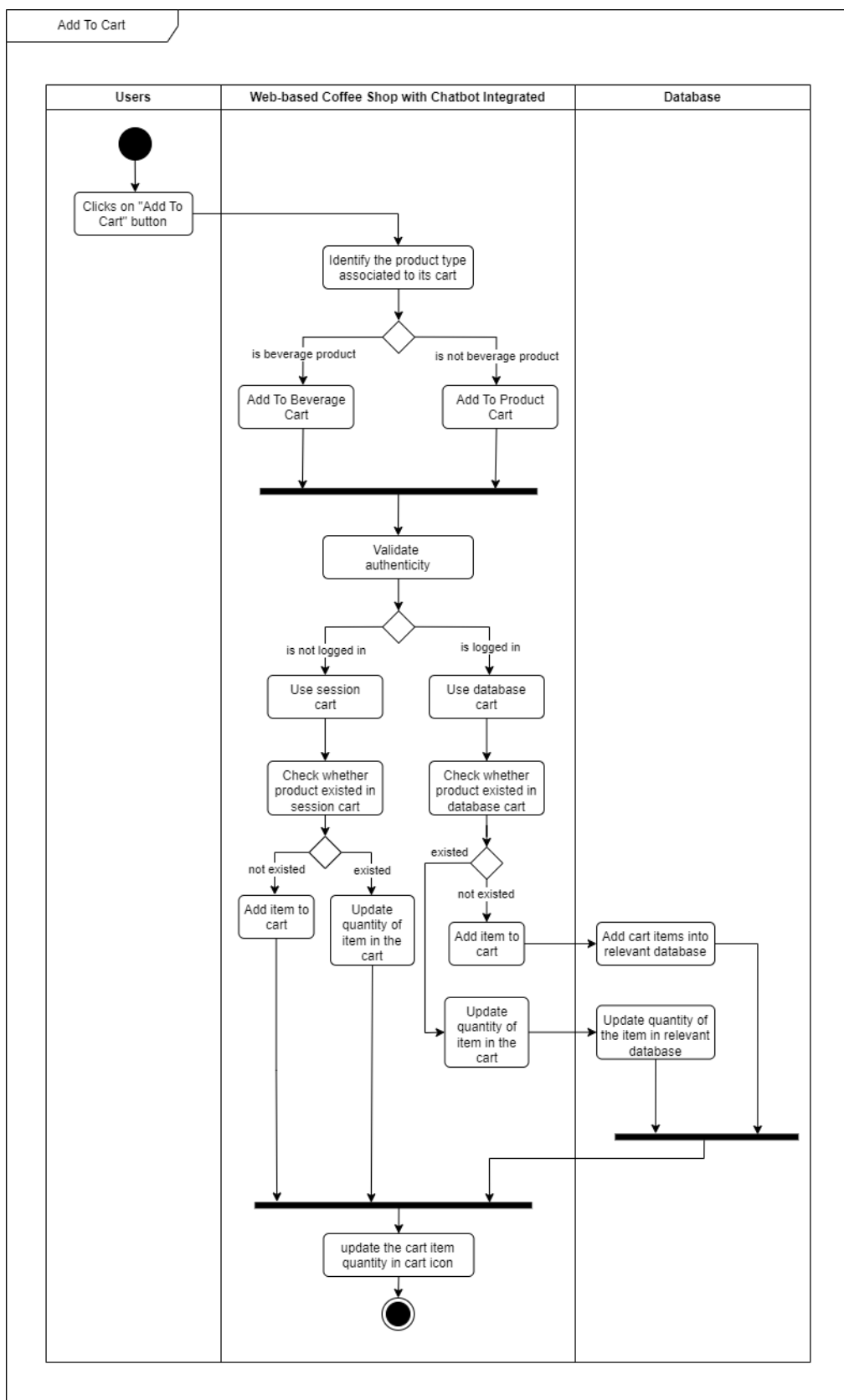


Figure 5.6.15: Activity Diagram – Add To Cart

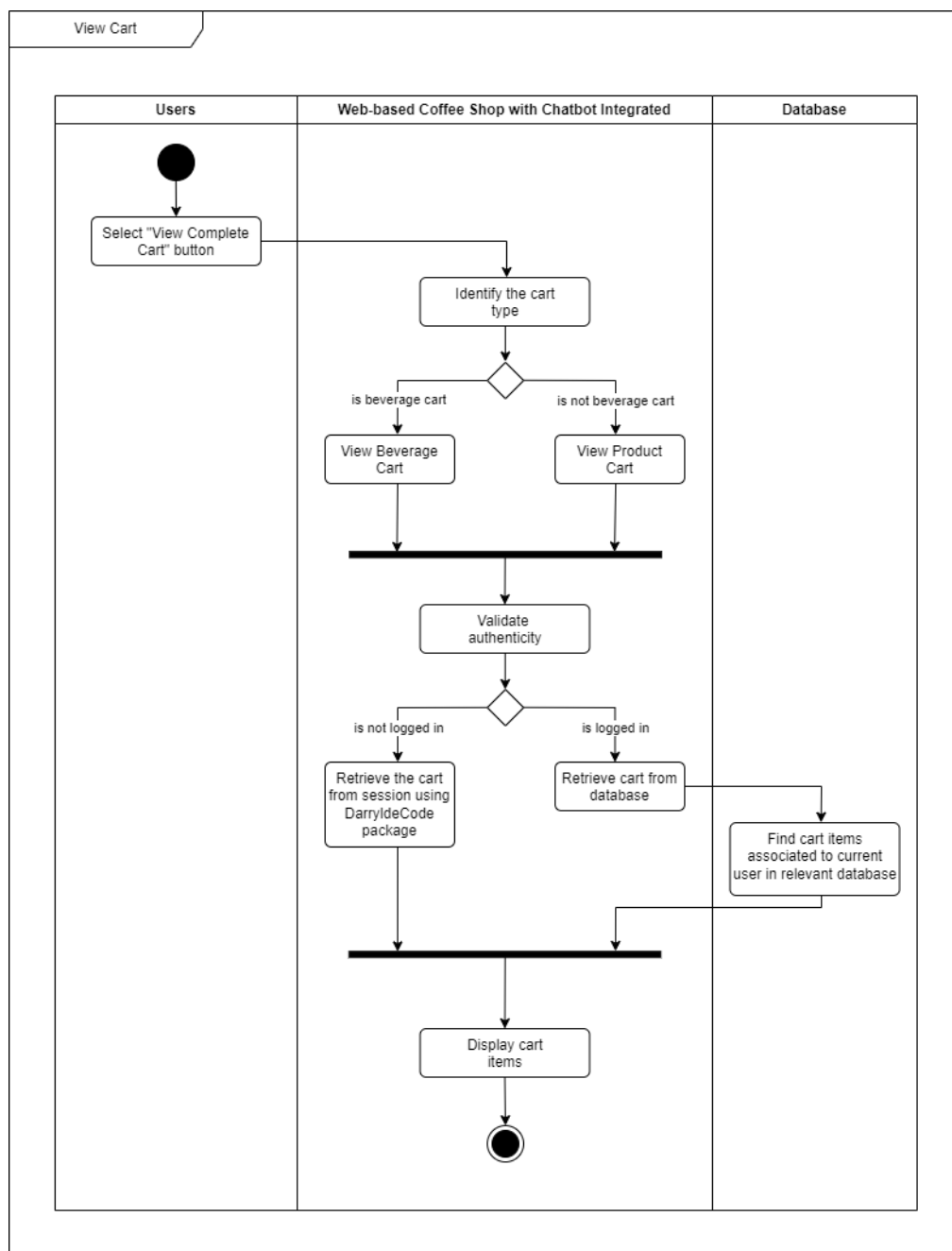


Figure 5.6.16: Activity Diagram – View Cart

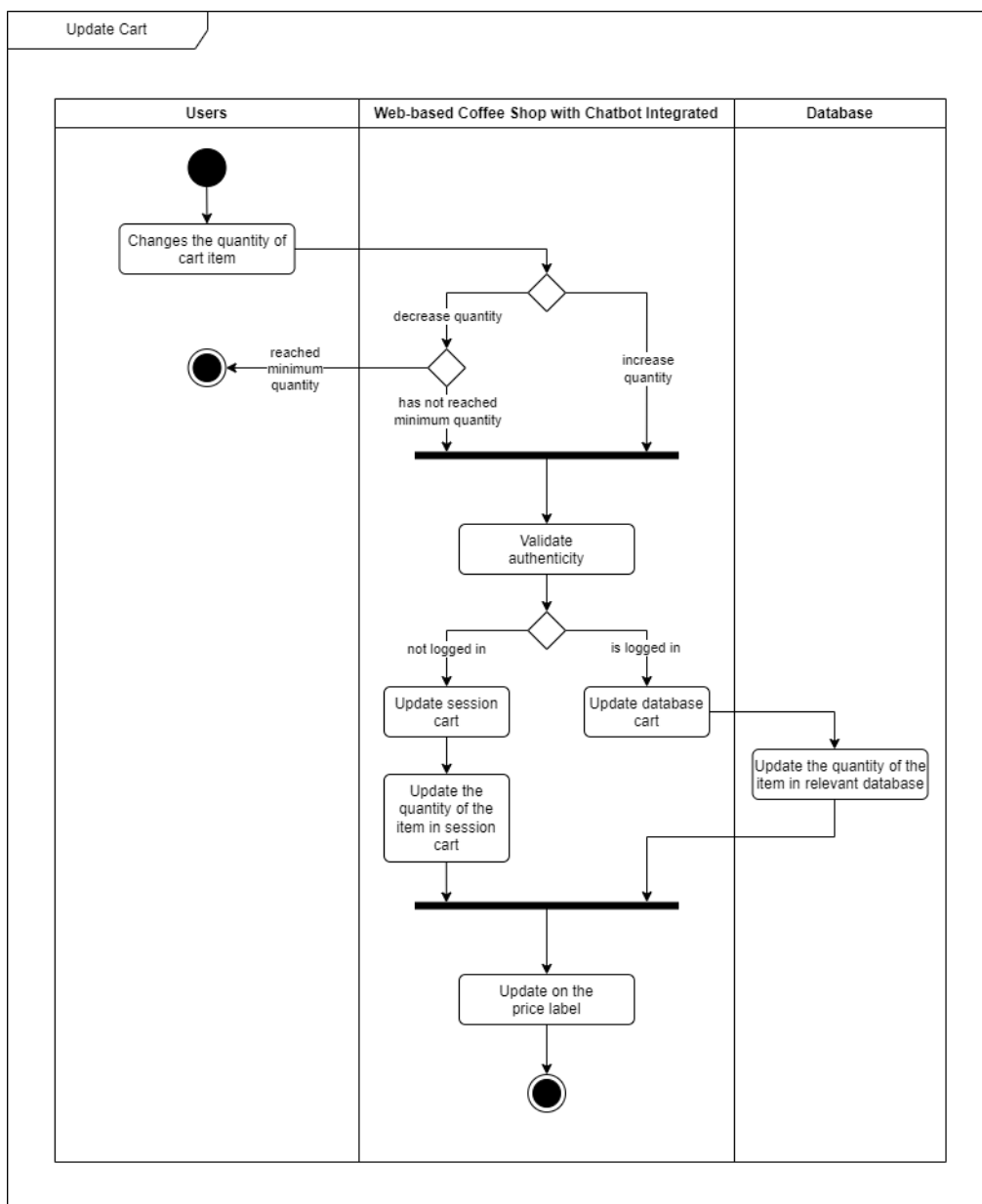


Figure 5.6.17: Activity Diagram – Update Cart

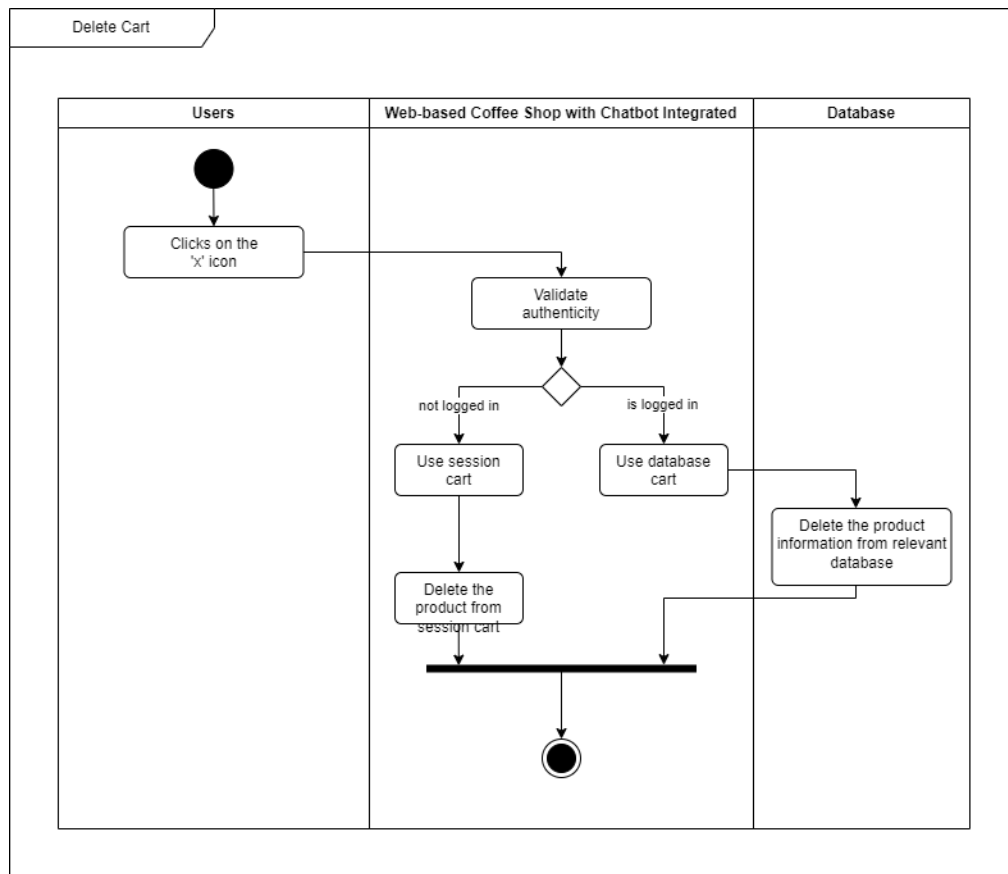


Figure 5.6.18: Activity Diagram – Delete Cart

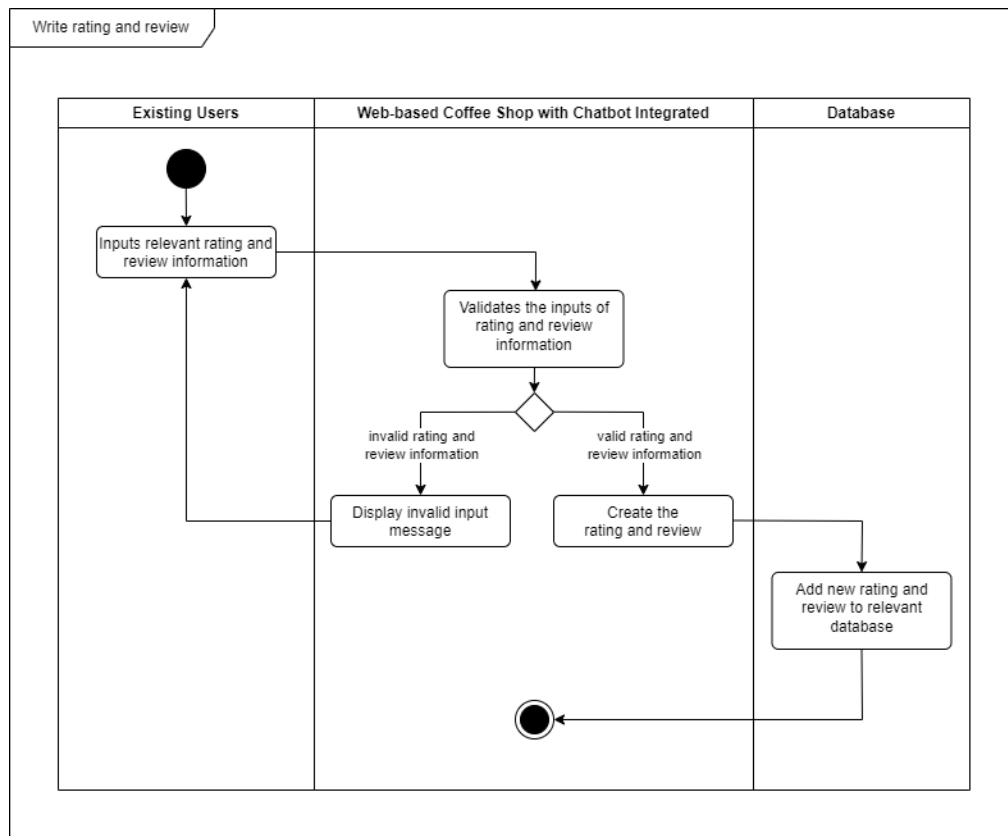


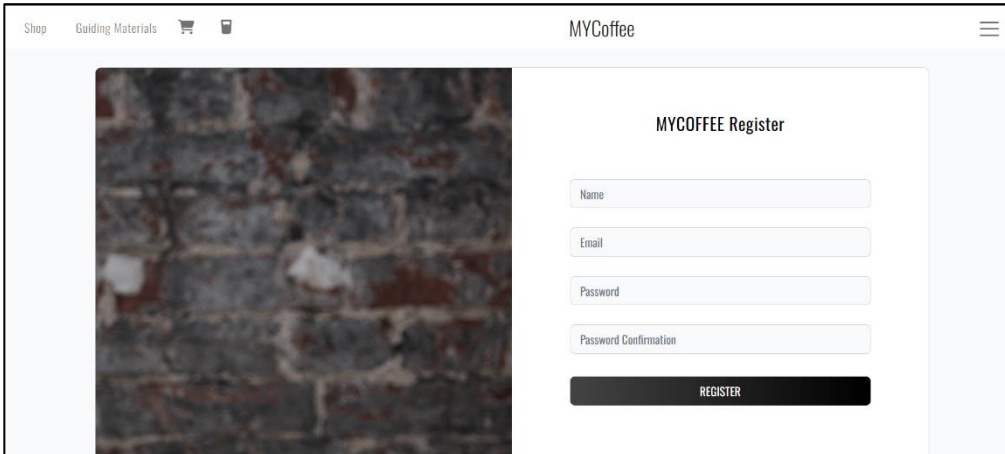
Figure 5.6.19: Activity Diagram – Write Rating and Review

5.7 UI Design

5.7.1 Introduction

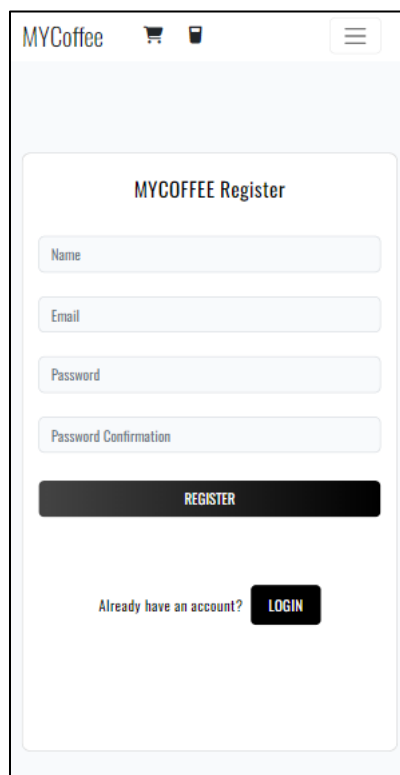
In this subsection, comprehensive UI designs for each feature are presented. These UI designs have been meticulously crafted with responsiveness in mind, ensuring a seamless user experience on both web and mobile platforms.

5.7.2 Register Page



The screenshot displays the MYCOFFEE Register page. The top navigation bar includes 'Shop', 'Guiding Materials', a shopping cart icon, a trash icon, and the 'MYCoffee' logo. A hamburger menu icon is located on the right. The main content area is split: the left side shows a close-up image of coffee beans, and the right side contains a registration form titled 'MYCOFFEE Register'. The form includes input fields for 'Name', 'Email', 'Password', and 'Password Confirmation', followed by a black 'REGISTER' button.

Figure 5.7.1: Register Page



MYCoffee

MYCOFFEE Register

Name

Email

Password

Password Confirmation

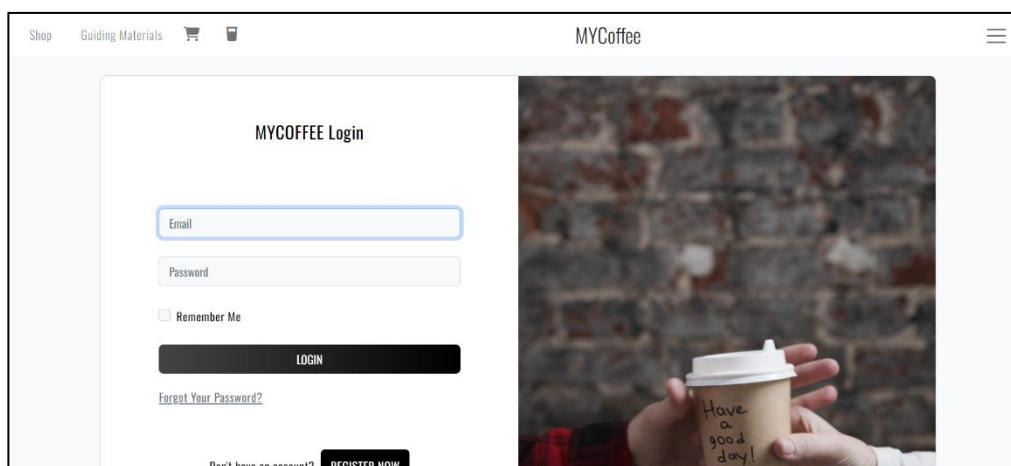
REGISTER

Already have an account? LOGIN

The image shows a mobile application interface for the 'MYCoffee' app. At the top, there is a header with the app name 'MYCoffee', a shopping cart icon, a trash can icon, and a hamburger menu icon. The main content area is titled 'MYCOFFEE Register'. It contains four text input fields: 'Name', 'Email', 'Password', and 'Password Confirmation'. Below these fields is a large black button with the text 'REGISTER' in white. At the bottom of the form, there is a link 'Already have an account?' followed by a black button with the text 'LOGIN' in white.

Figure 5.7.2: Responsive Register Page

5.7.3 Login Page



Shop Guiding Materials

MYCoffee

MYCOFFEE Login

Email

Password

Remember Me

LOGIN

[Forgot Your Password?](#)

Don't have an account? REGISTER NOW

The image shows a desktop application interface for the 'MYCoffee' app. The header includes 'Shop', 'Guiding Materials', a shopping cart icon, a trash can icon, the app name 'MYCoffee', and a hamburger menu icon. The main content area is titled 'MYCOFFEE Login'. It features two text input fields: 'Email' and 'Password'. Below these is a checkbox labeled 'Remember Me'. A large black button with the text 'LOGIN' in white is positioned below the checkbox. A link 'Forgot Your Password?' is located below the 'LOGIN' button. At the bottom of the login form, there is a link 'Don't have an account?' followed by a black button with the text 'REGISTER NOW' in white. To the right of the login form is a large image of a person's hands holding a coffee cup with the text 'Have a good day!' written on it.

Figure 5.7.3: Login Page

MYCoffee

MYCOFFEE Login

Email

Password

Remember Me

LOGIN

[Forgot Your Password?](#)

Don't have an account? [REGISTER NOW](#)

Figure 5.7.4: Responsive Login Page

5.7.4 Home Page

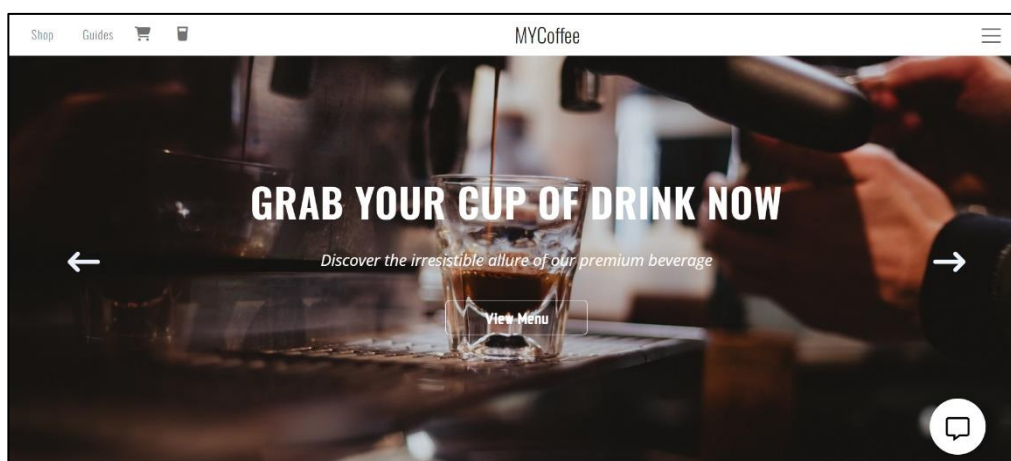


Figure 5.7.5: Home Page

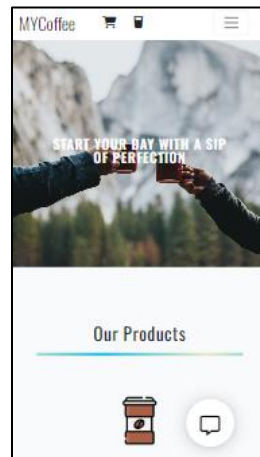


Figure 5.7.6: Responsive Home Page

5.7.5 Beverage List Page

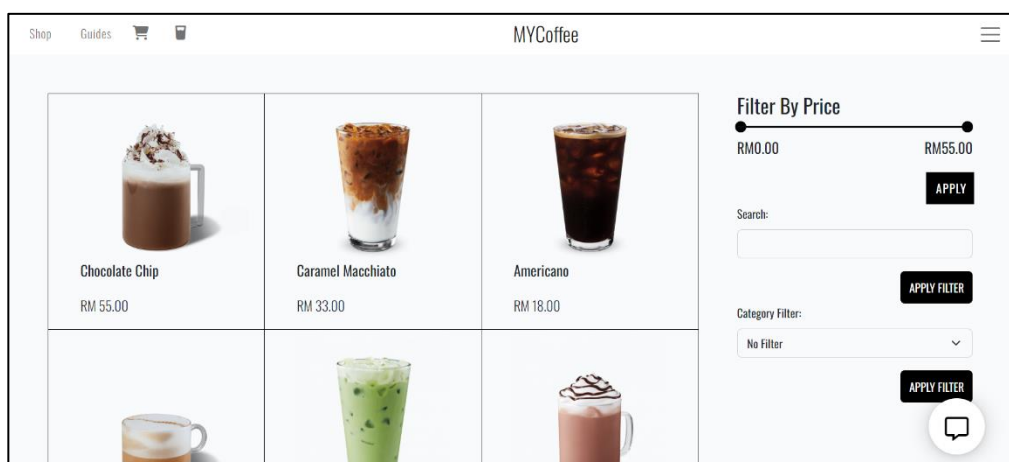


Figure 5.7.7: Beverage List Page

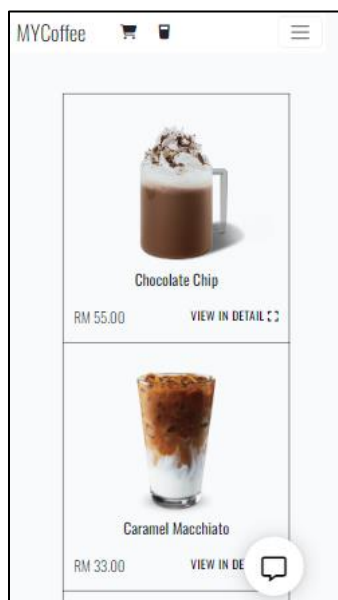


Figure 5.7.8: Responsive Beverage List Page

5.7.6 Homebrew Product List Page

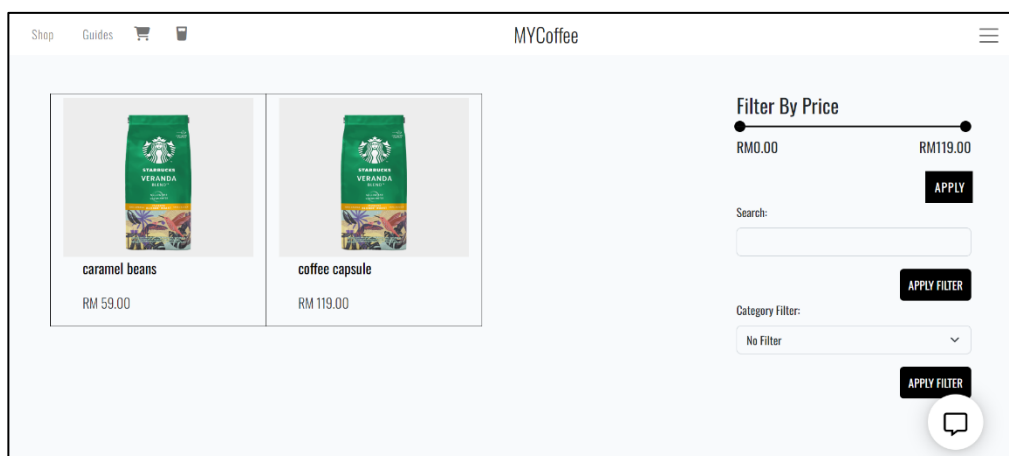


Figure 5.7.9: Homebrew Product List Page

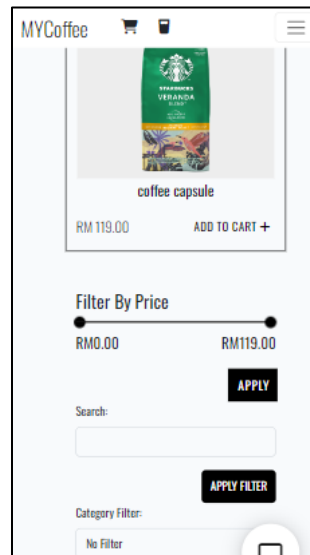


Figure 5.7.10: Responsive Homebrew Product List Page

5.7.7 Product Detail Page

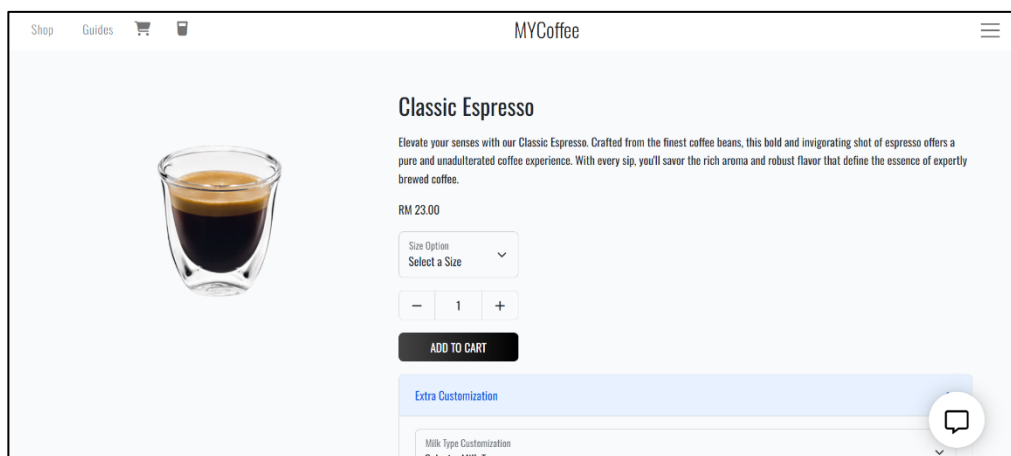


Figure 5.7.11: Beverage Product Detail Page (Part 1)

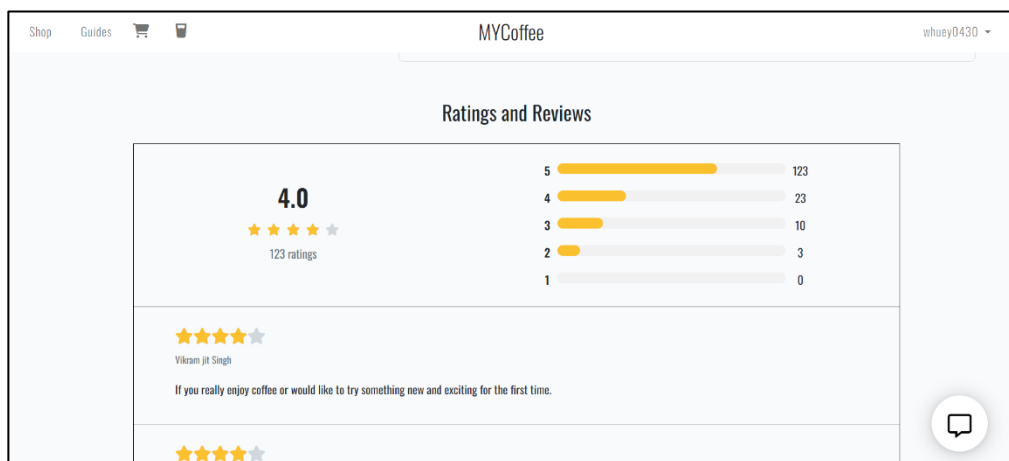


Figure 5.7.12: Beverage Product Detail Page (Part 2)

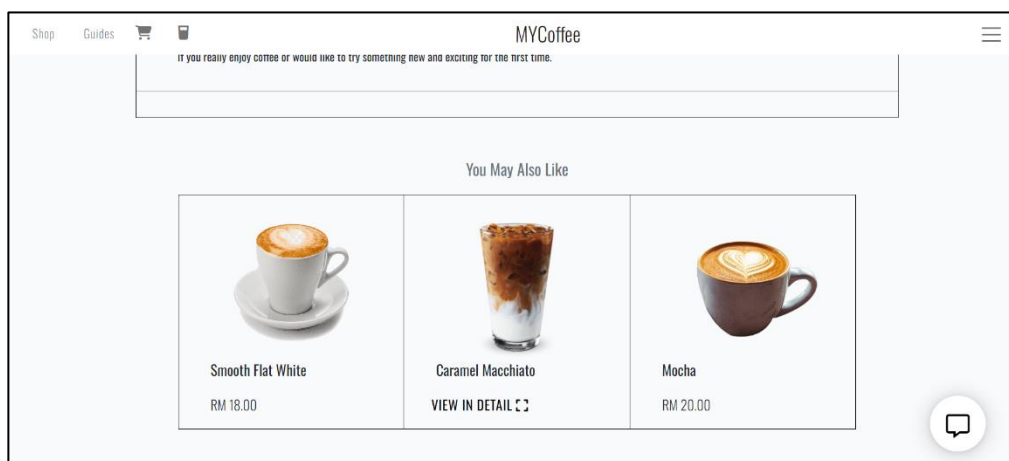


Figure 5.7.13: Beverage Product Detail Page (Part 3)



Figure 5.7.14: Responsive Beverage Product Detail Page (Part 1)

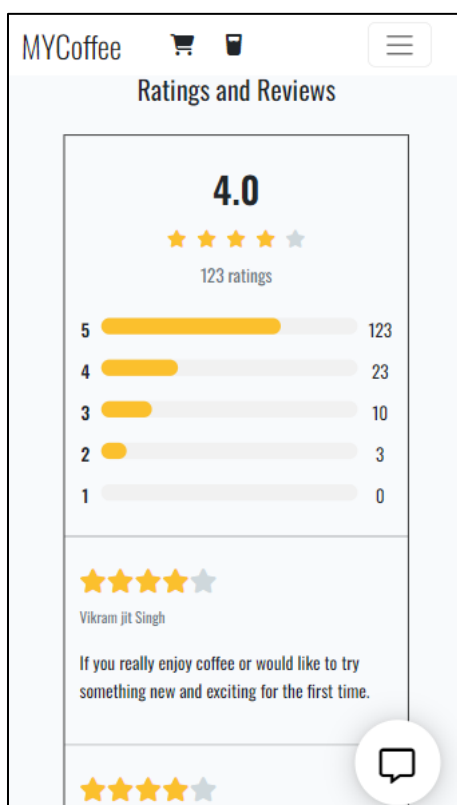


Figure 5.7.15: Responsive Beverage Product Detail Page (Part 2)

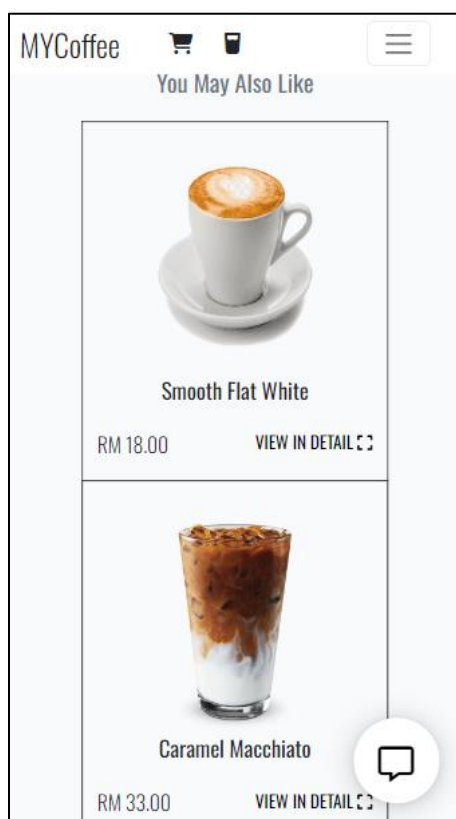


Figure 5.7.16: Responsive Beverage Product Detail Page (Part 3)

5.7.8 Write Review Modal

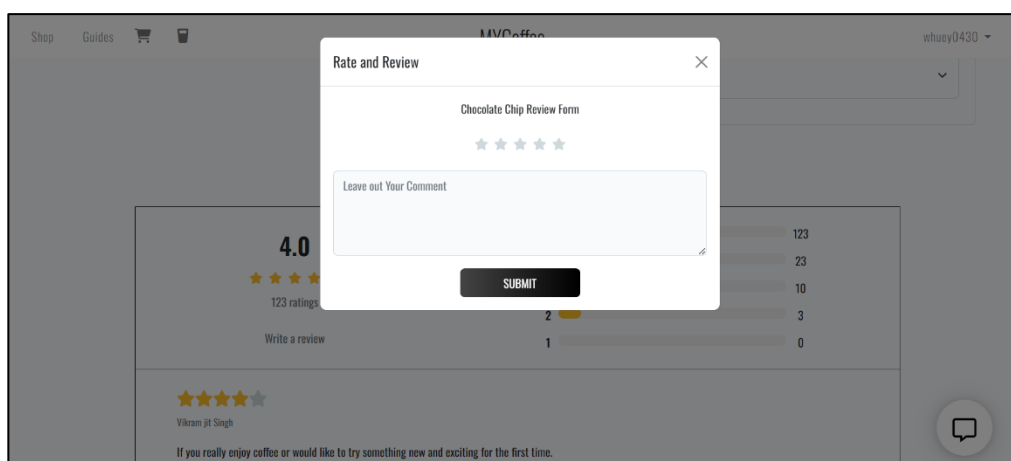


Figure 5.7.17: Write Review Modal

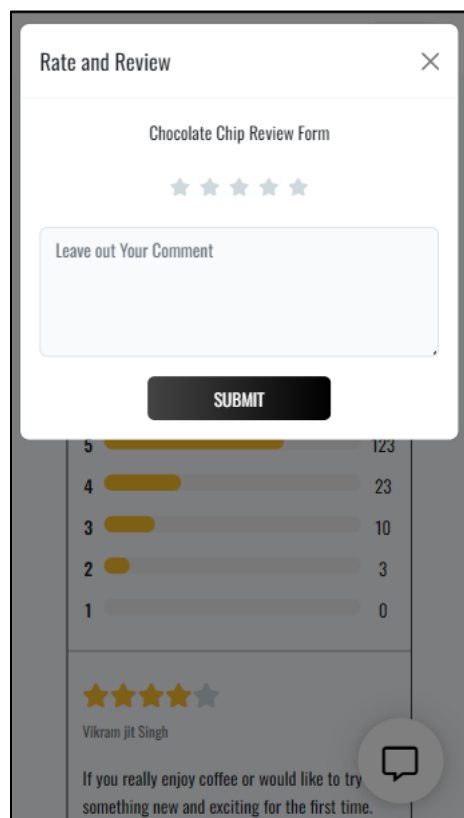


Figure 5.7.18: Responsive Write Review Modal

5.7.9 Shopping Cart

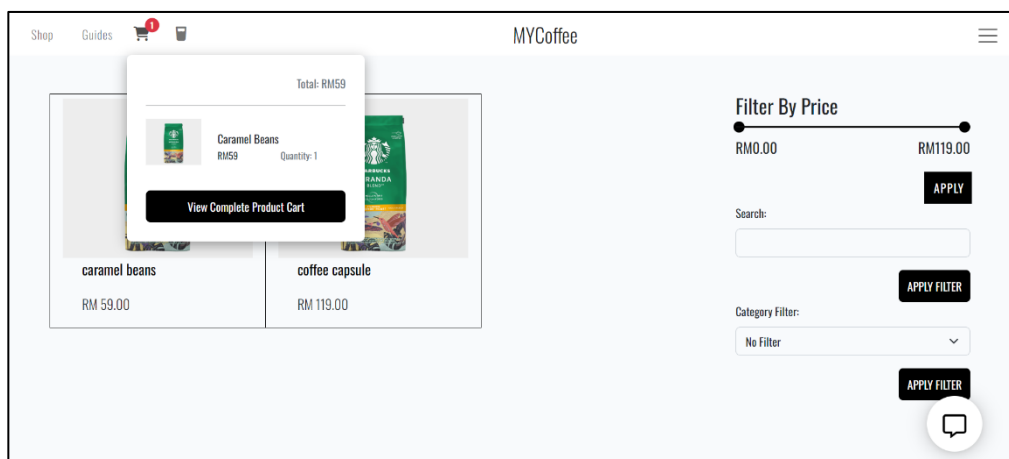


Figure 5.7.19: Shopping Cart Modal

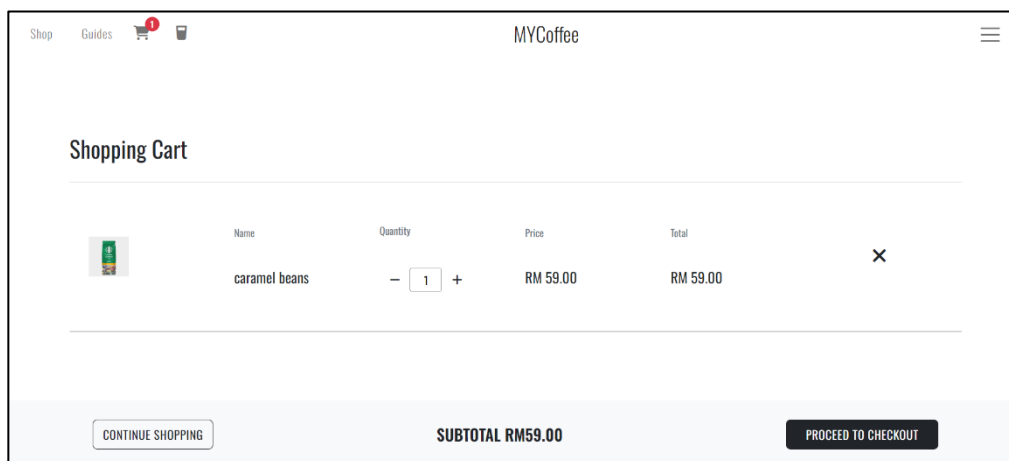


Figure 5.7.20: Shopping Cart Page



Figure 5.7.21: Responsive Shopping Cart Page

5.7.10 Admin Panel

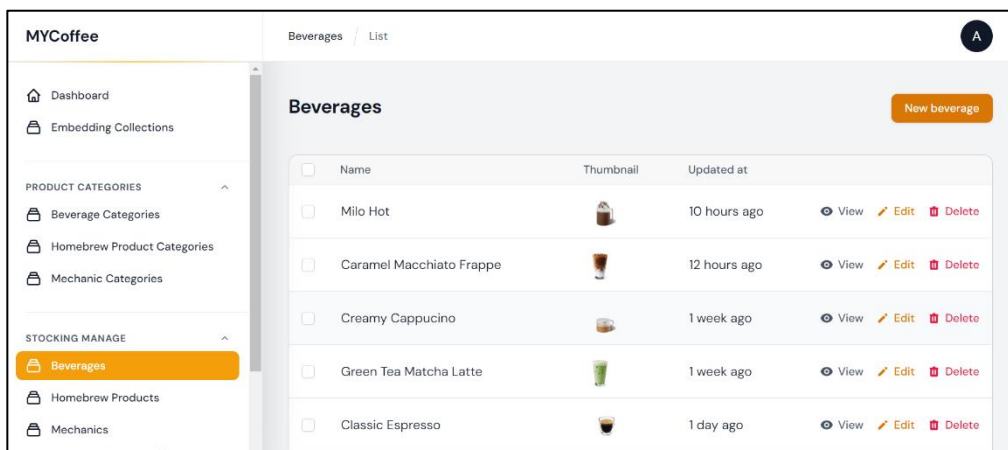


Figure 5.7.22: Admin Panel Page

5.7.11 Order List Page

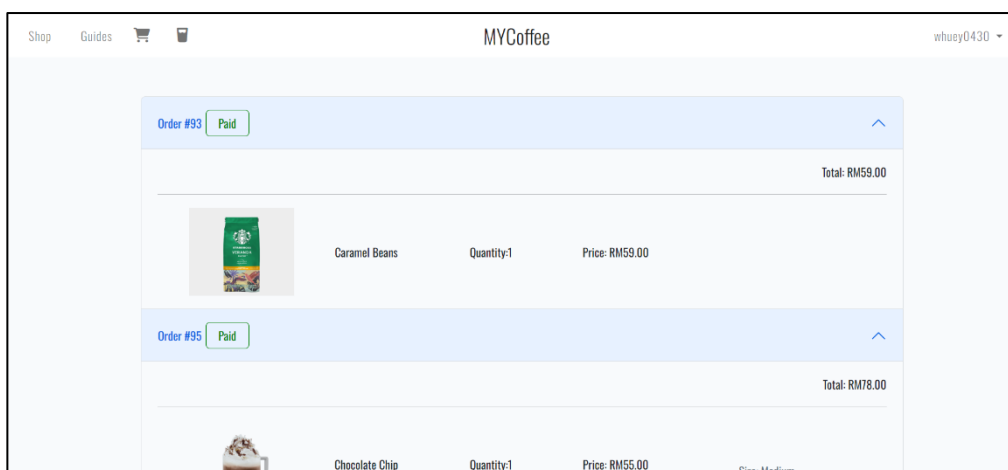


Figure 5.7.23: User Order List Page

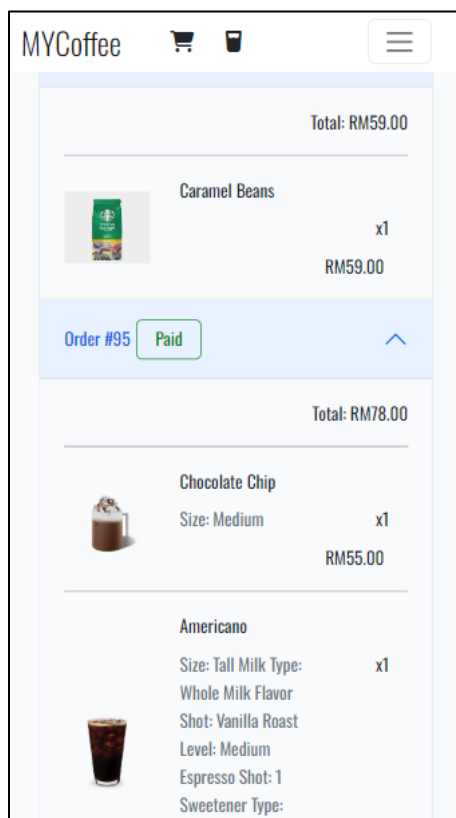


Figure 5.7.24: Responsive User Order List Page

5.7.12 Recipe Guiding Page

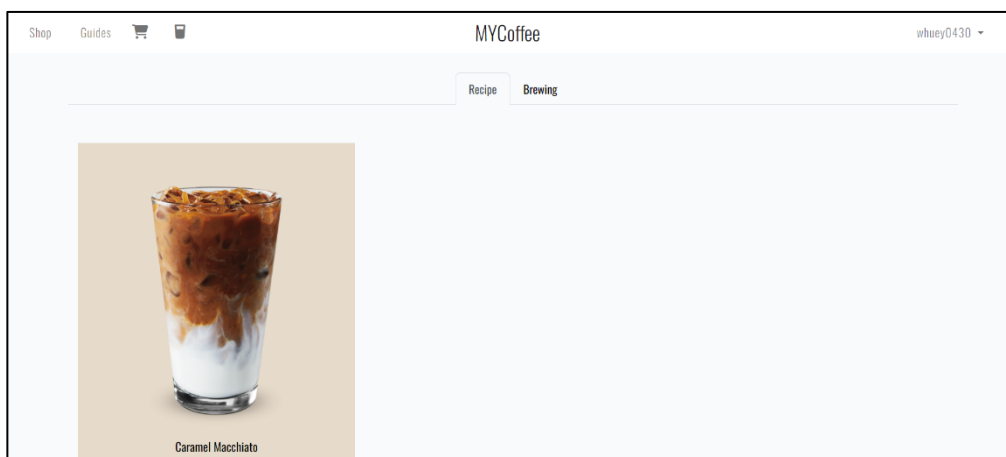


Figure 5.7.25: Recipe Guiding List Page

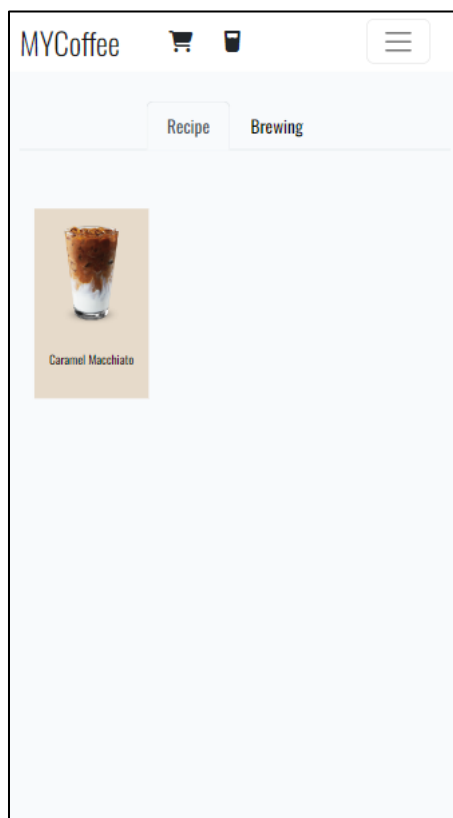


Figure 5.7.26: Responsive Recipe Guiding List Page

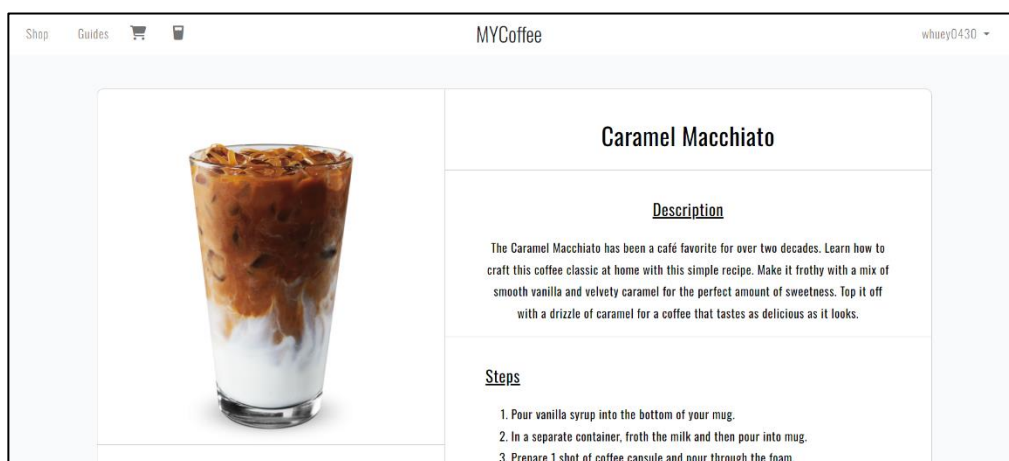


Figure 5.7.27: Recipe Guiding Detail Page



Figure 5.7.28: Responsive Recipe Guiding Detail Page

5.7.13 Brewing Guide Page

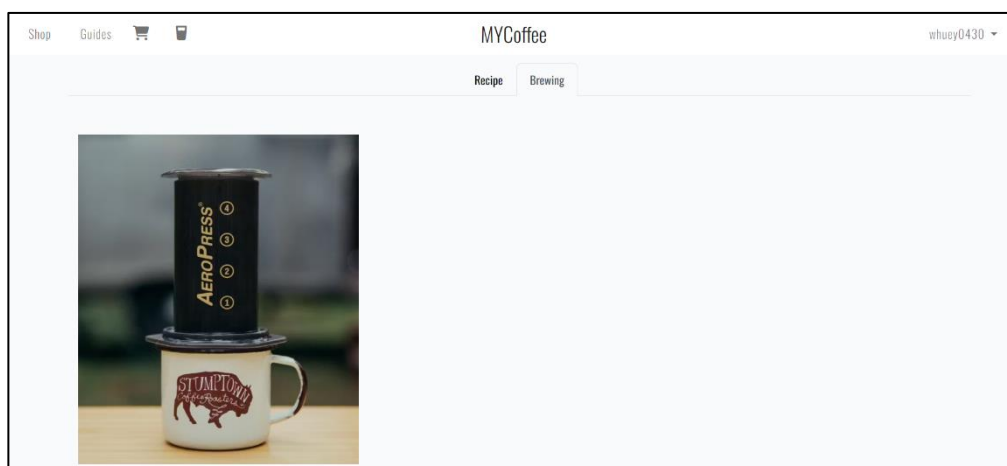


Figure 5.7.29: Brewing Guide List Page

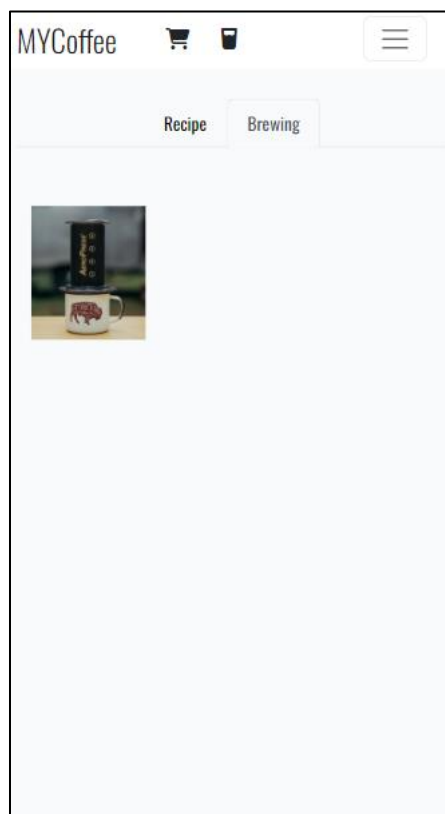


Figure 5.7.30: Responsive Brewing Guide List Page

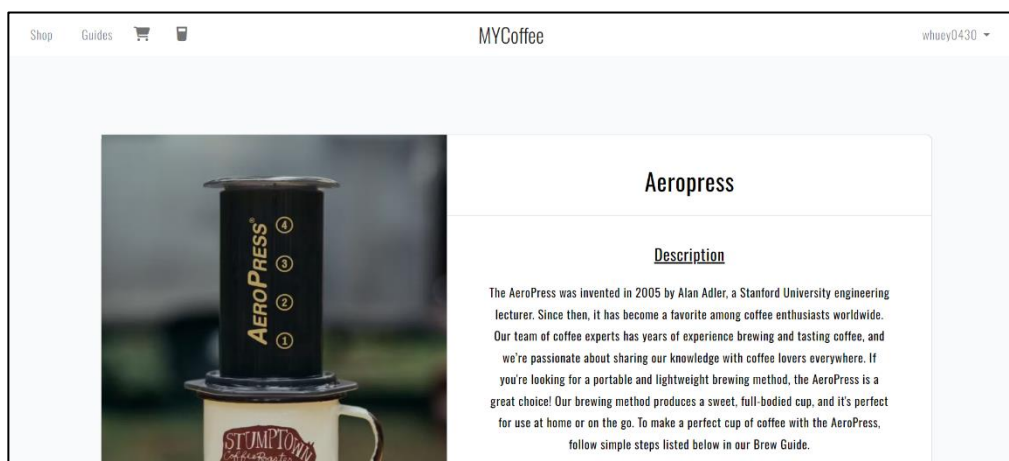


Figure 5.7.31: Brewing Guide Detail Page



Figure 5.7.32: Responsive Brewing Guide Detail Page

5.7.14 Chatbot Pop Up Window

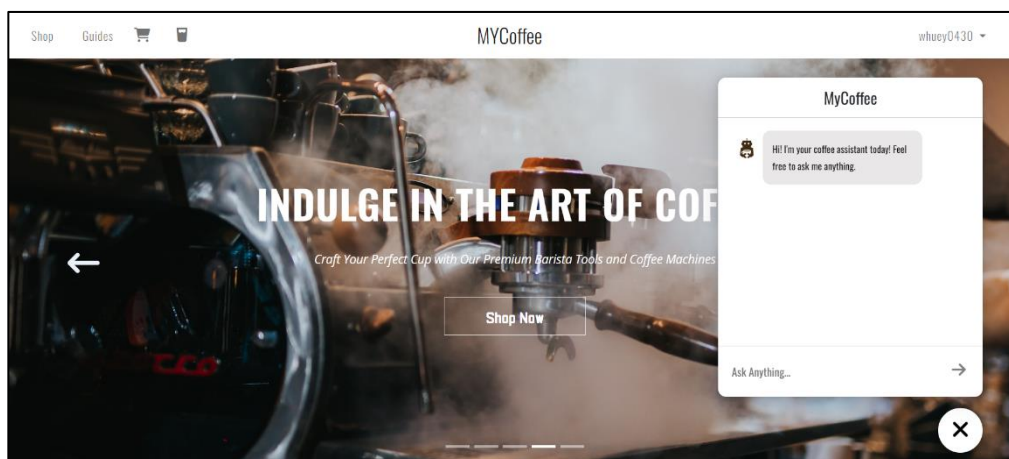


Figure 5.7.33: Chatbot Pop Up Window

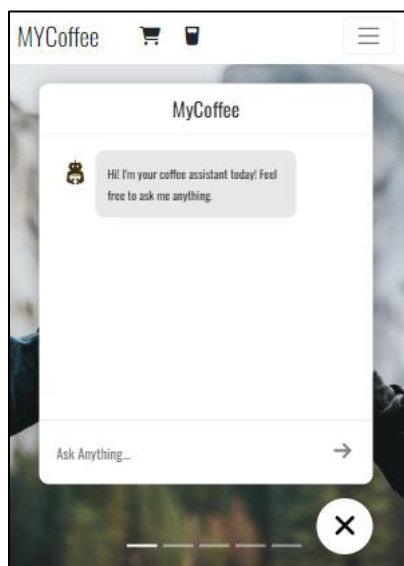


Figure 5.7.34: Responsive Chatbot Pop up Window

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 Authentication

In this project, the Laravel default authentication package will be used to facilitate user authentication. The Laravel authentication package seamlessly integrates with the project using just a few commands. It's worth noting that this project implements multi-authentication, catering to both regular users and admins. Nevertheless, it's essential to emphasize that admins are barred from registering new accounts as a crucial measure to enhance security and maintain control.

```
'guards' => [  
    'web' => [  
        'driver' => 'session',  
        'provider' => 'users',  
    ],  
  
    'admin' => [  
        'driver' => 'session',  
        'provider' => 'admins',  
    ],  
],
```

Figure 6.1.1: Defining Admin Guards

Laravel provides the concept of "guards," which allows the system to handle multiple types of user authentication simultaneously. By default, Laravel comes with the "web" guard, which handles regular user authentication. However, in this project, there's a need for two types of authentication, so a new guard called "admin" is defined. This "admin" guard will operate using the "admins" database tables.

```
// env('FILAMENT_AUTH_GUARD', 'web')
'auth' => [
  'guard' => 'admin',
  'pages' => [
    'login' => \Filament\Http\Livewire\Auth\Login::class,
  ],
],
```

Figure 6.1.2: Filament Guards Configuration

Once the "admin" guard is registered, it needs to be configured in Filament. Configuring the "admin" guard in Filament enables the project to achieve multi-authentication, accommodating both regular users and administrators. Below are some examples of authentication validation.

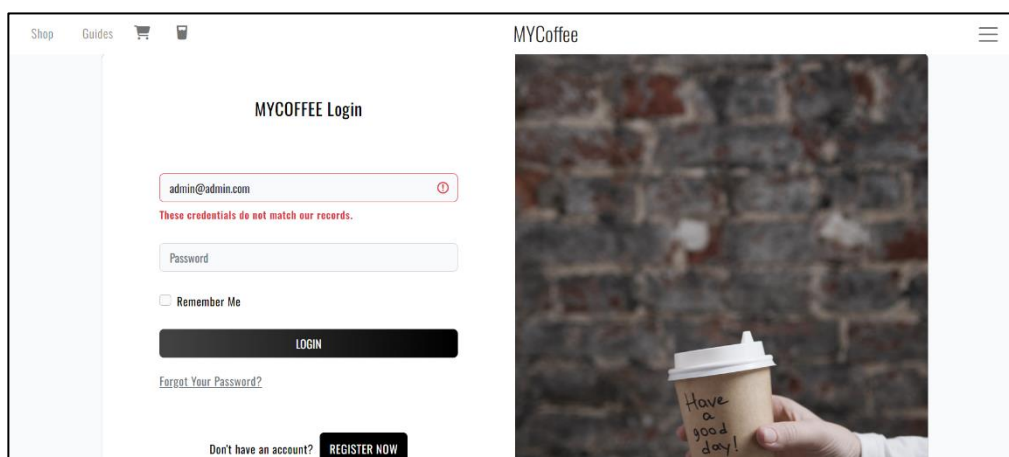


Figure 6.1.3: Attempting Login with Admin Credentials on Normal User Login Form

Figure 6.1.4: Attempting Login with User Credentials on Admin Login Form

Figure 6.1.5: Attempting Register with Existing User Credentials on User Register Form

6.2 Product List with Search and Filter features

In this project, three types of products will be available: beverages, homebrew products, and machinery products. Each product type will have its own dedicated list page, along with specific supporting features such as filtering by price range, filtering by category, and product search. These features are designed to work seamlessly together, allowing users to apply multiple filters successively. The figures below illustrate some examples of applying filters to the product list and conducting searches from the list.

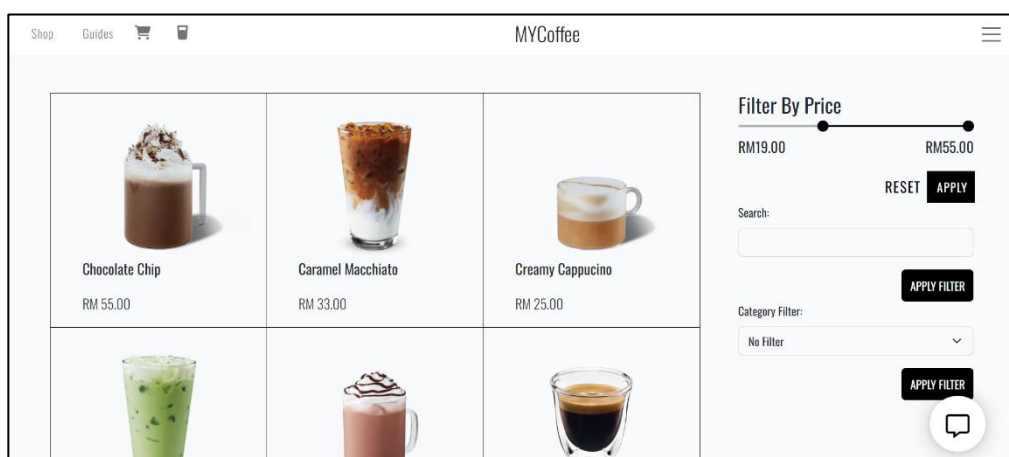


Figure 6.2.1: Product List with Price Range Filter Applied

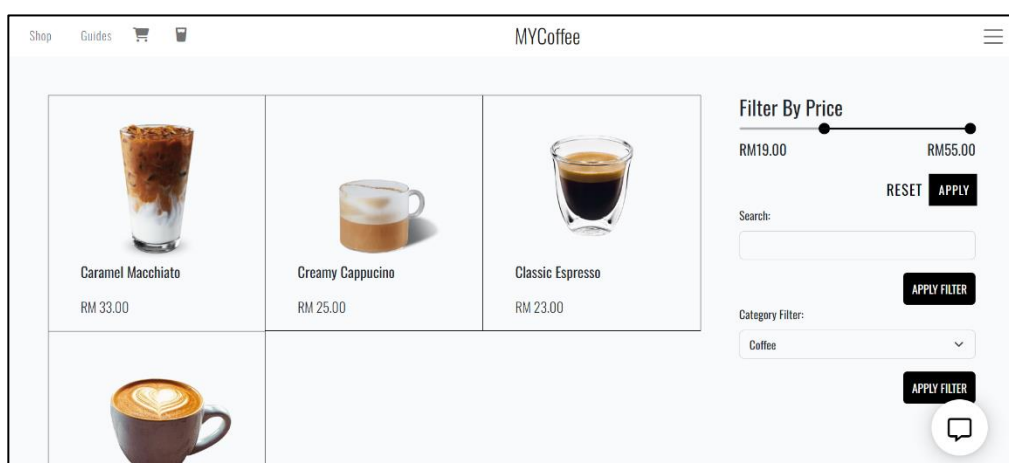


Figure 6.2.2: Product List with Category Filter Add On

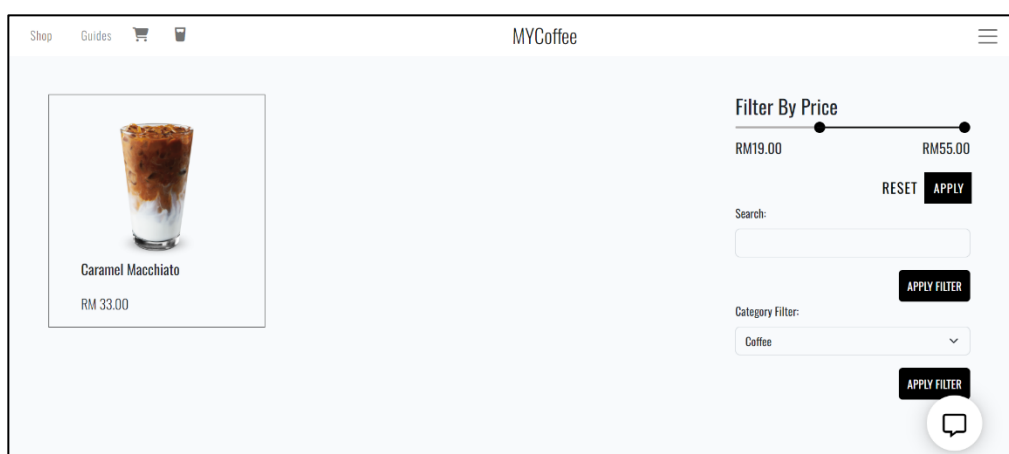


Figure 6.2.3: Product List with Search Filter using keywords of “caramel”
Add On

Furthermore, users will have the ability to effortlessly remove applied filters. For instance, when a price range filter has been applied, users can simply click on the "Reset" button, which will promptly revert the price range to its original state. In the case of category filters, users can easily select the "No Filter" option, causing the filter to be removed. Lastly, when it comes to search filters, users can achieve filter removal by simply submitting a blank input, effectively nullifying the search filter.

```

public function index(Request $req)
{
    $categories = BeverageCategory::all();
    $minPrice = $req->min_price ?? 0;
    $maxPrice = $req->max_price ?? Beverage::max('price');

    $rangeMaxPrice = Beverage::max('price');
    $minPriceValue = preg_replace('/^[^0-9]/', '', $minPrice);
    $maxPriceValue = preg_replace('/^[^0-9]/', '', $maxPrice);
    if ($req) {
        $query = Beverage::query();

        if ($req->has('search')) {
            $query->where('name', 'like', '%' . $req->input('search') . '%');
        }

        if ($req->has('min_price') && $req->has('max_price')) {
            $query->whereBetween('price', [$minPriceValue, $maxPriceValue]);
        }

        if ($req->has('category')) {
            if ($req->category) {
                $query->where('beverage_category_id', $req->input('category'));
            }
        }
    }

    $filteredBeverages = $query->get();

    return view('beverageList.index', ['beverages' => $filteredBeverages, 'categories' => $categories, 'minPrice' => $minPriceValue, 'maxPri
}

```

Figure 6.2.4: Code Implementation of the Filters

In addition to the filtering feature, users have the option to add items to their cart, but this feature is specifically available for homebrew products and machinery products. Beverage products, on the other hand, do not support this functionality due to specific customization requirements, which will be discussed in detail later on. When users click on the "Add To Cart" button provided, the selected item will be placed into their respective shopping carts.

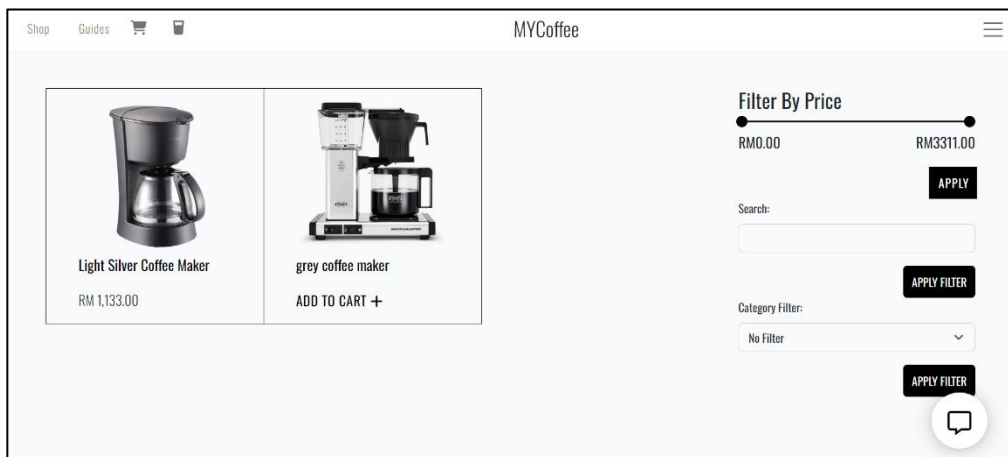


Figure 6.2.5: Add To Cart Button When Hover on the Price

6.3 Product Detail

Users will discover in-depth details about the product, encompassing comprehensive descriptions and pricing information. They have the flexibility to fine-tune the quantity they wish to add to the beverage cart. Moreover, the product detail page features ratings and reviews, providing valuable insights, along with relevant recommendations for related products which will be discussed in later sub-sections.

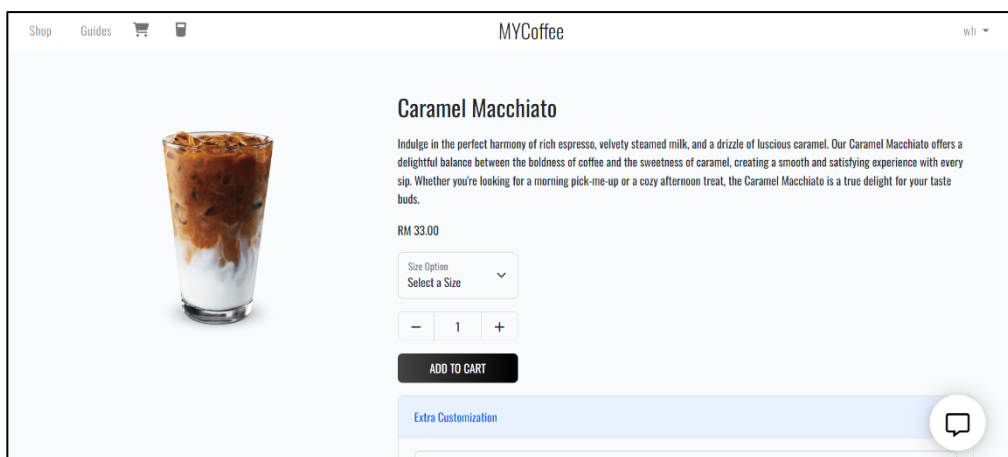


Figure 6.3.1: Product Information Provided in Product Detail Page

6.4 Beverage Customization

The website also provides an exclusive features of beverage customization. Each beverage will have the customization selection which are based on the questionnaire survey conducted previously. The options include the milk type, sweetener type, flavor shot, espresso shot and also the roast level. Users are required to select the size of the beverage while the customization is optional, but if there will additional charges for selected customization and users could view on the total price and the customizations added upon adding the beverage into their cart.

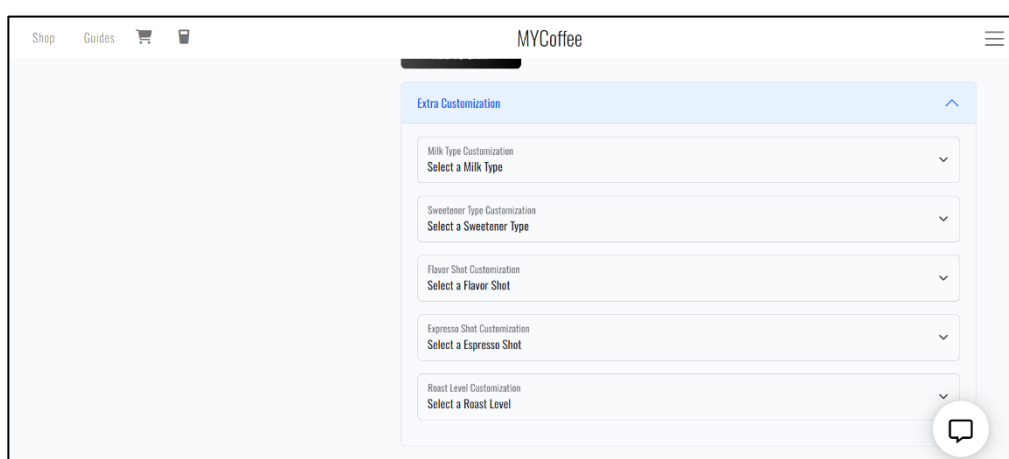


Figure 6.4.1: List of Beverage Customization Options

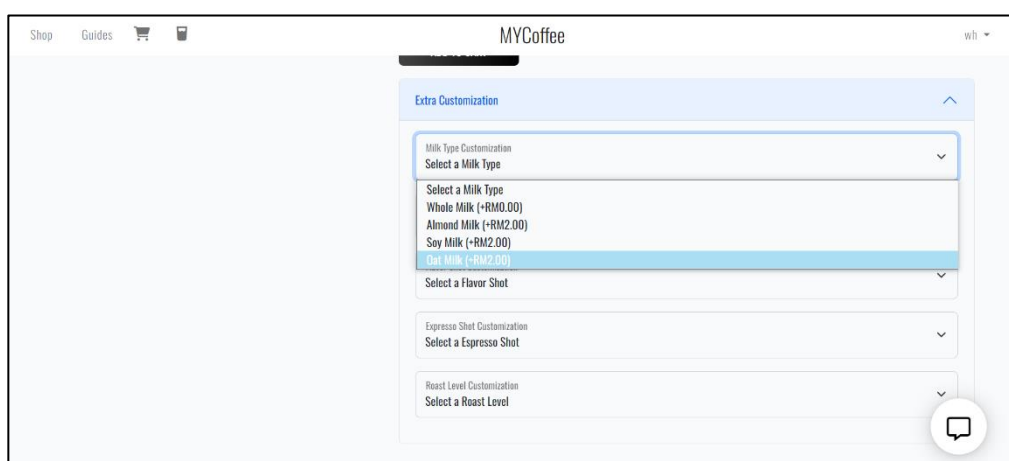


Figure 6.4.2: Customization Choices with Clearly Stated Prices

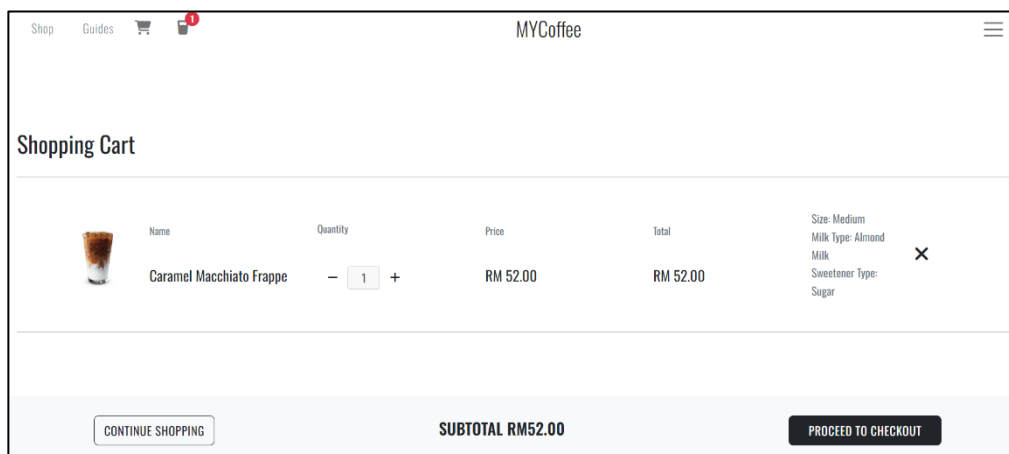


Figure 6.4.3: Display of Beverage Cart: Total Price and Customization Overview

6.5 Product Recommendation

A general product recommendation is also provided in the product detail page in which the product recommendation is recommended randomly with same category as current product in the product detail page. For example, in the illustration below, within the product detail page showcasing a caramel macchiato categorized under coffee, three randomly selected coffee beverages are presented as recommendations. Users can effortlessly explore the recommended products by navigating to their respective pages if they find them of interest.

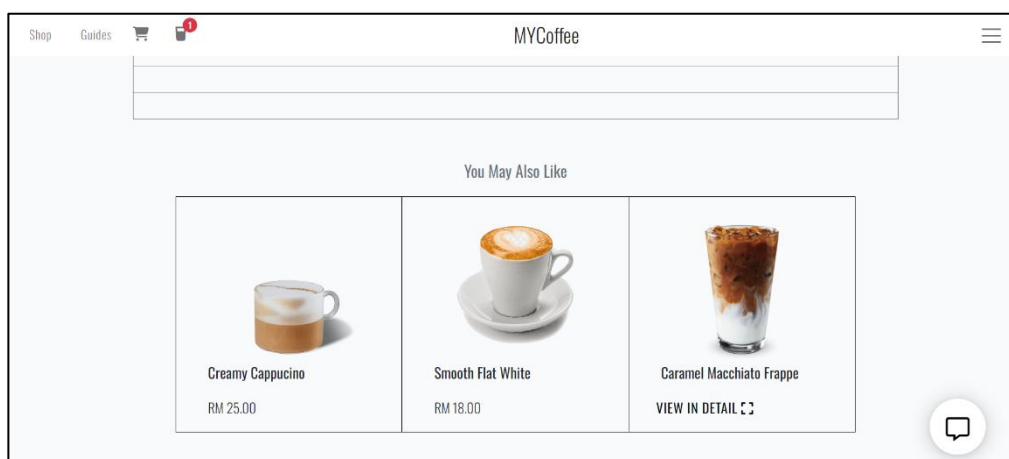


Figure 6.5.1: Sample of General Product Recommendations

6.6 Product Rating

The website offers a robust feature for rating and reviewing products, allowing users to peruse existing feedback on the product detail page. It's worth noting that only authenticated users have the privilege to both rate and compose new reviews for a particular product. Each authenticated user is restricted to submitting one rating and review per product, with no option for subsequent edits or deletions after submission.

```
public function store(Request $req)
{
    $rating = $req->rating;
    $comment = $req->comment;
    $modelType = $req->modelType;
    $id = $req->product_id;
    $user = auth()->user();
    $review = new RateReview();

    if ($modelType === "Beverage") {
        $item = Beverage::find($id);
    } elseif ($modelType === "HomebrewProduct") {
        $item = HomebrewProduct::findOrFail($id);
    } elseif ($modelType === "Mechanic") {
        $item = Mechanic::findOrFail($id);
    }

    $exist = RateReview::join('reviewables', 'rate_reviews.id', '=', 'reviewables.rate_review_id')
        ->where('rate_reviews.user_id', $user->id)
        ->where('reviewables.reviewable_id', $item->id)
        ->where('reviewables.reviewable_type', get_class($item))
        ->first();

    if (!$exist) {
        $user->rateReview()->save($review);
        $reviewable = $review->related()->make();
        $reviewable->reviewable()->associate($item);
        $reviewable->comment = $comment;
        $reviewable->rating = intval($rating);
        $reviewable->save();

        return redirect()->back()->with('msg', 'Successful write');
    } else {
        return redirect()->back()->with('msg', 'You have written review');
    }
}
```

Figure 6.6.1: Product Rating and Review Code Implementation

Based on the implemented code snippet above, a validation process is in place to determine if the user has already submitted a review for the specific product. If a review exists, the user is redirected to the previous page with a notification signaling that a review has already been written for that particular product. Conversely, if no prior review exists, the information from the submitted rating and review form is saved into the database. The user is then redirected back to the previous page, where the review list is updated,

accompanied by a notification confirming the successful submission of the new review. The figures below showcase samples of product ratings and reviews.



Figure 6.6.2: Sample of Product Rating and Review List

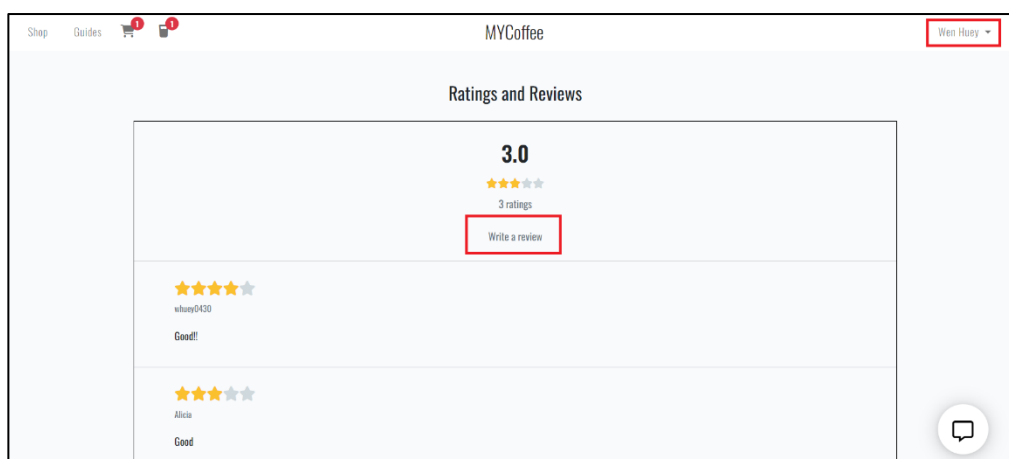


Figure 6.6.3: Sample View of Authenticated User's Product Rating and Review List

Observing Figure 6.6.2 and Figure 6.6.3, a distinct "Write a review" button is discernible exclusively for authenticated users. Upon clicking this button, an engaging user experience unfolds, revealing a modal that hosts the rate and review form as shown in the figure below.

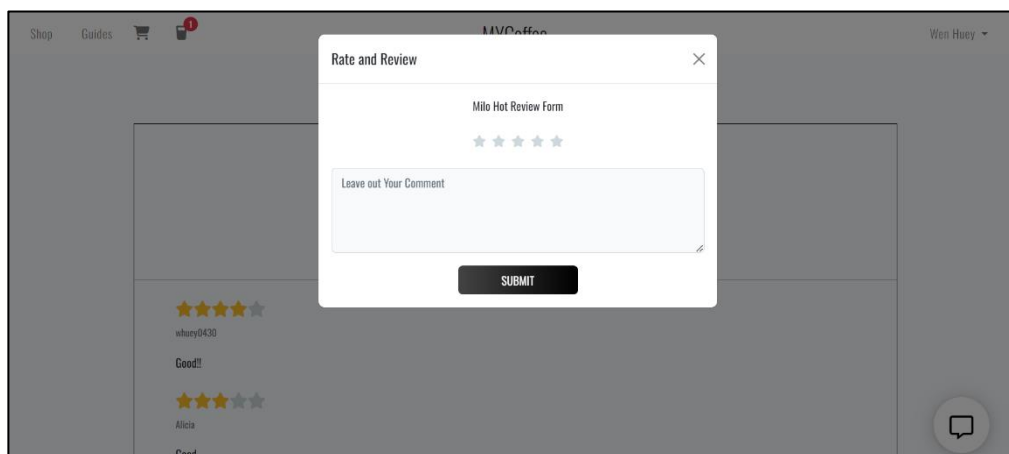


Figure 6.6.4: Product Rating and Review Form Modal

6.7 Chatbot Interaction with OpenAI

6.7.1 OpenAI Data Embedding

```

1 reference | 0 overrides
public function extractDataFromCSV($filePath)
{
    $plainText = '';
    $sentences = '';
    $import = new EmbeddingCollectionImport;
    Excel::import($import, $filePath);

    foreach ($import->importedData as $index => $item) {
        if ($index == 0) {
            continue; // skip the first row
        }
        $name = $item[0];
        $content = $item[1];
        $sentences .= $item[0] . " where contents are " . $item[1] . " ";
    }

    //tokenize the data
    $tokens = tokenize($sentences);
    $normalizedTokens = normalize_tokens($tokens);
    $cleanedText = implode(' ', $normalizedTokens);

    $wordsPerChunk = 1000; // number of words per chunk
    $overlapWords = 200; // number of overlapping words between chunks
    $pattern = '/\s+' . '{1,' . ($wordsPerChunk + $overlapWords) . '\}(?=\s)/'; // regex pattern to split by words with overlap

    $chunks = preg_split($pattern, $cleanedText, -1, PREG_SPLIT_NO_EMPTY);
    $chunkCount = count($chunks);
}

```

Figure 6.7.1: Data Embedding File Code Implementation (Part 1)

```

$wordsPerChunk = 1000; // number of words per chunk
$overlapWords = 200; // number of overlapping words between chunks
$pattern = '/\s+' . '\{1,' . ($wordsPerChunk + $overlapWords) . '\}(?=\s)/'; // regex pattern to split by words with overlap

$chunks = preg_split($pattern, $cleanedText, -1, PREG_SPLIT_NO_EMPTY);
$chunkCount = count($chunks);

// Loop through the chunks and store each one as a vector
foreach ($chunks as $a => $chunk) {
    $context = '';
    if ($a > 0) {
        $prevChunk = $chunks[$a - 1];
        $context = implode(' ', array_slice(explode(' ', $prevChunk), -$overlapWords));
    }
    if ($a < $chunkCount - 1) {
        $nextChunk = $chunks[$a + 1];
        $context .= ' ' . implode(' ', array_slice(explode(' ', $nextChunk), 0, $overlapWords));
    }

    $chunkWithContext = $context . ' ' . $chunk;

    $clearEmbeddingDatabase = Embedding::truncate();
    // Call the function to generate the embeddings for the context
    // if statement to ensure get the latest data from the file
    if ($clearEmbeddingDatabase) {
        $vector = $this->generateEmbedding($chunkWithContext);
        // Store the chunk to the database
        Embedding::create([
            'text' => $chunkWithContext,
            'text_vector' => json_encode($vector),
        ]);
    }
}

```

Figure 6.7.2: Data Embedding File Code Implementation (Part 2)

As detailed in Chapter 5's architecture, the initial step involves extracting data from the uploaded data embedding file. This task is managed by the 'VectorService,' a service primarily focused on handling actions related to converted vectors. The first function within the service extracts data from the uploaded file, following the code implementation outlined in the diagrams above. To seamlessly extract data from the file, an Excel Laravel package is employed. Subsequently, the extracted data is concatenated to resemble a coherent sentence, facilitating better understanding of the context by OpenAI in subsequent stages. Following concatenation, the data undergoes tokenization and normalization. It is then split into multiple chunks to prevent exceeding the token limit per request. Afterward, the database is cleared to maintain only the latest data consistently. Finally, the vector generation process is executed, as demonstrated in the figure below. The response is then saved into the database.

```

2 references | 0 overrides
public function generateEmbedding($embeddingData): array
{
    $vector = OpenAI::embeddings()->create([
        'model' => 'text-embedding-ada-002',
        'input' => $embeddingData,
    ]);

    if (count($vector['data']) == 0) {
        throw new Exception("Failed to generate embedding!");
    }

    return $vector['data'][0]['embedding'];
}

```

Figure 6.7.3: Data Embedding File Code Implementation (Part 3)

6.7.2 Question and Answering

```

$question = $req->question;

try {
    $vectorService = new VectorService;
    $questionVector = $vectorService->generateEmbedding($question);
    $relevantChunks = $vectorService->getMostSimilarVectors($questionVector);
    $similarTexts = $vectorService->getTextsFromIds(array_column($relevantChunks, 'id'));
    $context = implode(' ', $similarTexts);

    $responses = $this->askQuestionStreamed($context, $question);
    $resultText = "";

    foreach ($responses as $response) {
        $text = $response->choices[0]->delta->content;
        $resultText .= $text;
        if (connection_aborted()) {
            break;
        }
    }

    return response()->json([
        'answer' => $resultText
    ]);
} catch (\Throwable $th) {
    dd($th->getMessage());
}

```

Figure 6.7.4: Chatbot Question and Answering Code Implementation
(Part 1)

Upon a question is submitted, the system promptly engages the 'VectorService' to generate question vectors. These vectors are then used to search for similar vectors within the database. The array of text data associated with the most similar vector is converted into a context. Subsequently, the system invokes the

'askQuestionStreamed' function, which handles the configuration of the system prompt and related tasks as shown below.

```
public function askQuestionStreamed($context, $question)
{
    $prompt =
        "You are a helpful ecommerce coffee website assistant.
        Respond to the Q based on the context information of our coffee products given : " . $context .
        "\n Q: " . $question .

        "\n If the answer to the question is not in the context, just say 'sorry I do not know'";

    return OpenAI::chat()->createStreamed([
        'model' => 'gpt-3.5-turbo',
        'temperature' => 0.8,
        'messages' => [
            ['role' => 'system', 'content' => $prompt],
            ['role' => 'user', 'content' => $question],
        ],
    ]);
}
```

Figure 6.7.5: Chatbot Question and Answering Code Implementation
(Part 2)

The prompt serves the purpose of informing OpenAI about its role, emphasizing that it should provide responses solely based on the context to prevent unrelated answers. Following this, the GPT-3.5 model is specifically chosen, and the messages are conveyed in two distinct roles: one as the system and the other as the user. Figures below show the sample of question and answering process.

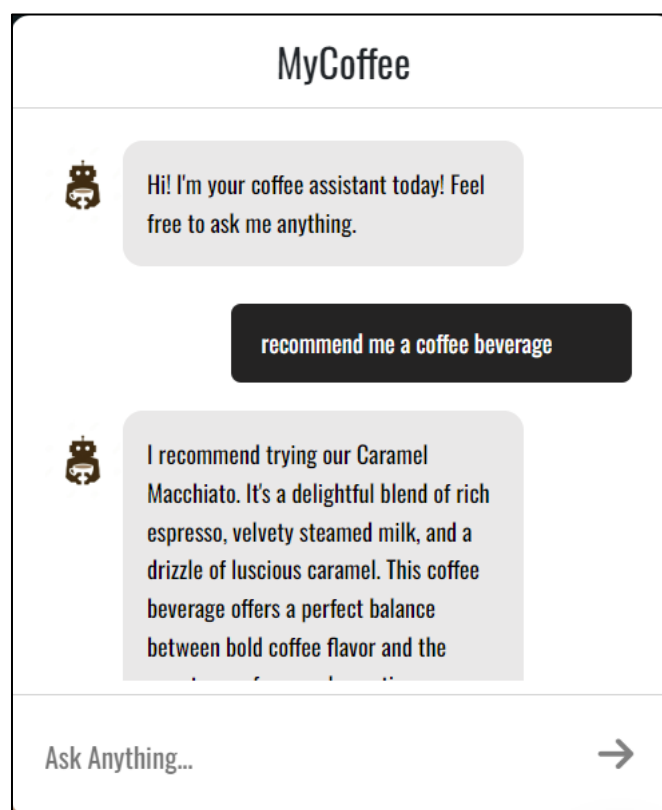


Figure 6.7.6: Sample Chat of Asking Product Recommendations Question

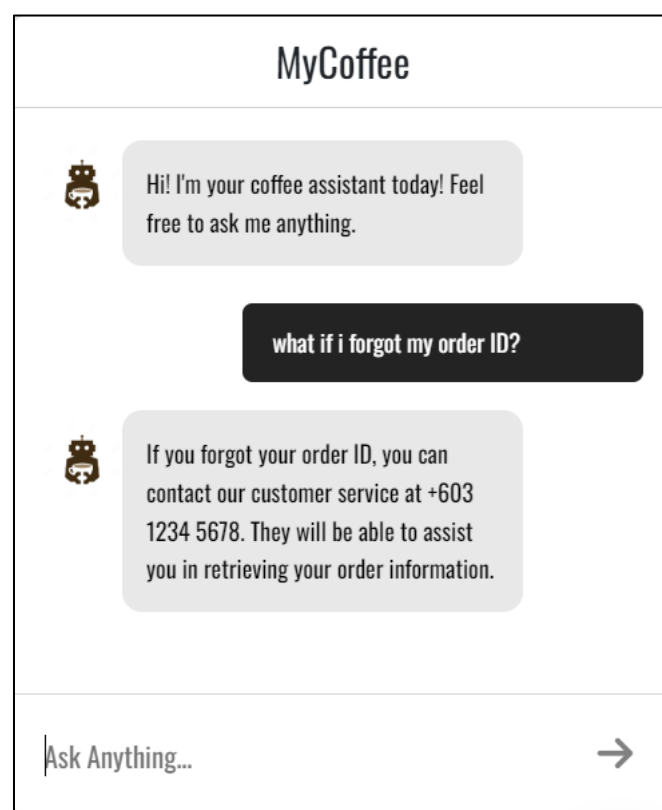


Figure 6.7.7: Sampel Chat of Asking Customer Support Question

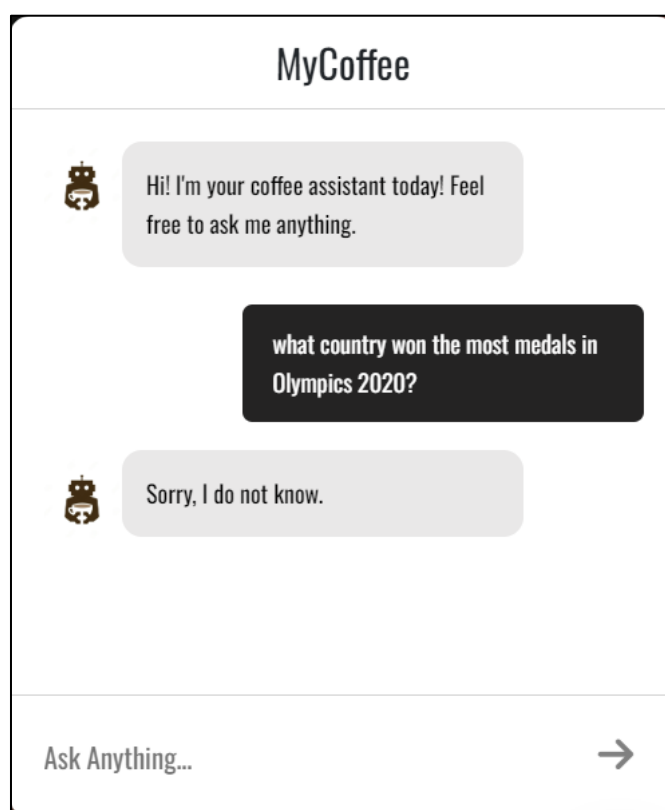


Figure 6.7.8: Sample Chat of Asking Unrelated Question

6.8 Data Embedding Preparation

In the previous subtopic, the process of embedding data into OpenAI was thoroughly discussed, thus, in this section, the preparation for the data embedding will be deep dived in. The data can be categorized into two segments: product recommendations and customer support. In the case of product recommendations, the product information available on our website will be utilized. This information encompasses details such as product names, prices, descriptions, and more. A representation of this data is illustrated in Figure 6.6.1, showcasing a sample stored in an Excel file.

Name	Content
Caramel Macchiato	Indulge in the perfect harmony of rich espresso, velvety steamed milk, and a drizzle of luscious caramel. Our Caramel Macchiato offers a delightful balance between the boldness of coffee and the sweetness of caramel, creating a smooth and satisfying experience with every sip. Whether you're looking for a morning pick-me-up or a cozy afternoon treat, the Caramel Macchiato is a true delight for your taste buds. Price: RM33.00 Product Type: Beverage Product Category: Coffee

Figure 6.8.1: Sample of Product Information Data in Excel File

Moving on to the customer support section, the dataset is inherently expansive and general. To address this, a thorough review of Frequently Asked Questions (FAQs) from various sources has been conducted, including the websites discussed in Chapter 2. The project incorporates the FAQs collected from these external websites as the foundation for our customer support data embedding. This approach is preferred as it ensures that the customer support information is both valuable and reliable, drawing upon established and trusted sources rather than generating data independently.

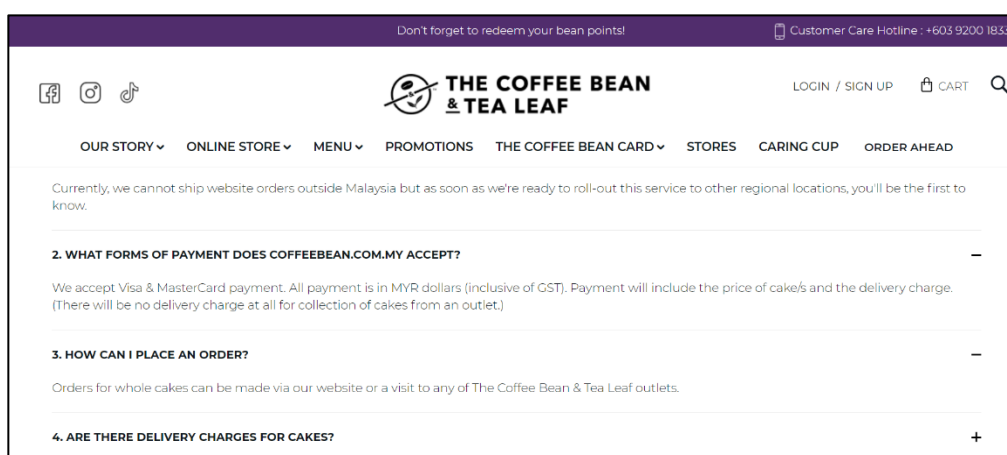


Figure 6.8.2: FAQ from The Coffee Bean & Tea Leaf

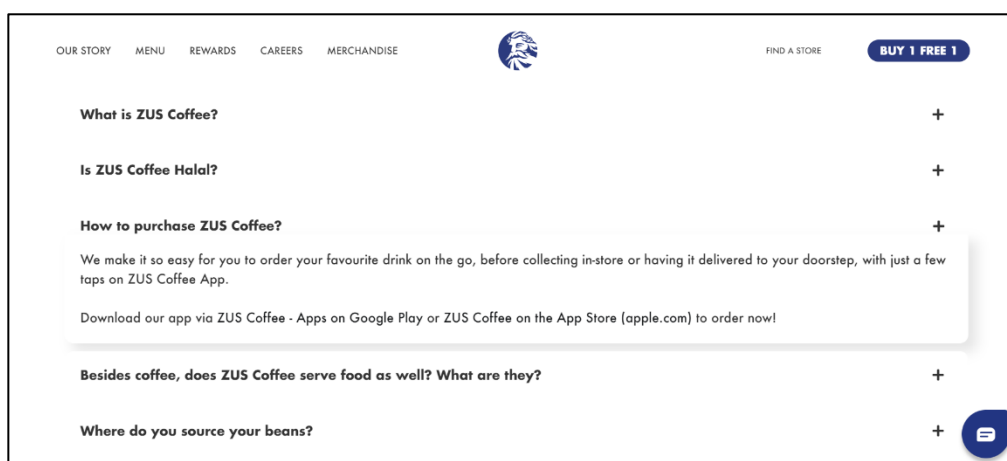


Figure 6.8.3: FAQ from ZUS Coffee

Can I place an international	Currently, we cannot ship website orders outside Malaysia but as soon as we're ready to roll-out this service to other regional locations, you'll be the first to know.								
How can I place an order?	Orders be made via our website								
What if I forget my order n	Please contact Customer Service at +603 1234 5678								
HOW IS MY ORDER DELIVERE	Orders will be delivered by our transportation partner or registered mail for small non-bulky items								

Figure 6.8.4: Sample of FAQ data for Embedding File Preparation

6.9 Shopping Cart

6.9.1 Session and Database Cart Implementation

i. Types of Shopping Cart

To accommodate the diverse range of products available on this website, two distinct shopping carts have been implemented. This decision is driven by the varying requirements for the products. For instance, beverages demand immediate delivery, ensuring they reach customers within one hour. In contrast, items like homebrew products or machinery may require a few days to reach their destination. Figure 6.3.1 and 6.3.2 shows the sample of the two shopping carts.

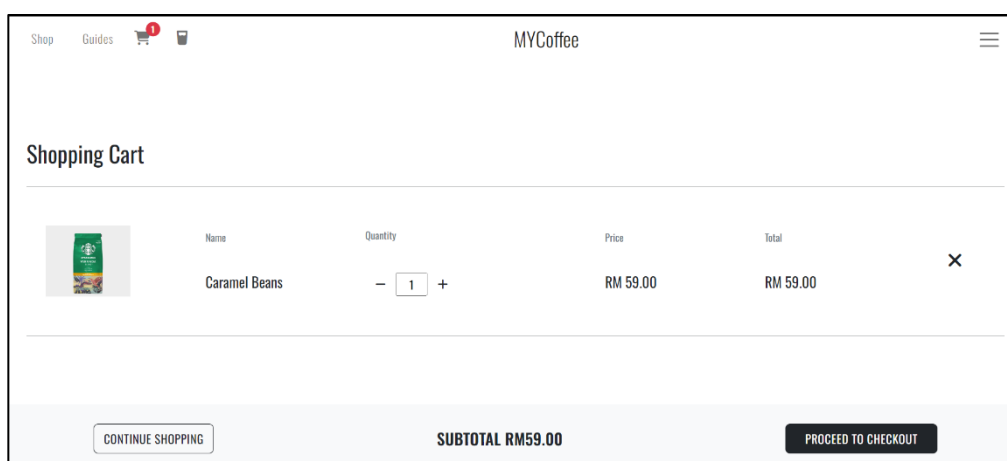


Figure 6.9.1: Product Cart

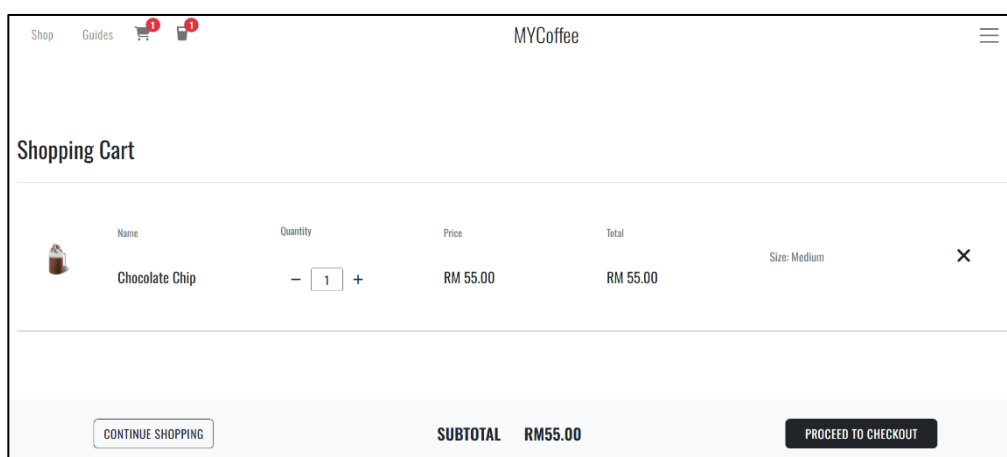


Figure 6.9.2: Beverage Cart

ii. Session and Database Shopping Cart

Moreover, customers have the flexibility to make payments, whether they are authenticated or unauthenticated. As a result, two distinct cart storage methods are employed for different types of users to perform changes on their carts.

For the case of unauthenticated customers, When they add an item to their cart, the session cart is utilized. This stores the cart information temporarily within the session. This approach is chosen because the session ID inherently ensures uniqueness, mitigating the risk of data duplication, as well as helps in minimizing data usage due to its temporary nature of storage. To streamline this process, the project incorporates the DarryldeCode package in Laravel. This package simplifies the task of storing items in the session, offering a range of features that allow for seamless item storage by invoking its functions.

```
if (!auth()->check()) {
    // Guest User
    $guestIdentifier = session()->getId() . "-beverage";
    $beverage = Beverage::find($beverageId);

    if ($beverage) {
        foreach ($attributes as $key => $value) {
            $beverageId .= "-" . $value;
        }
    }

    Cart::session($guestIdentifier)->add([
        'id' => $beverageId,
        'name' => $beverage->name,
        'price' => $beverage->price + $additionalPrice,
        'quantity' => $quantity,
        'associatedModel' => $beverage,
        'attributes' => $attributes,
    ]);
}
```

Figure 6.9.3: Beverage Session Cart Code Implementation

```

if (!auth()->check()) {
    $guestIdentifier = session()->getId() . "-products";

    if ($modelType === "HomebrewProduct") {
        $id = $req->homebrewProduct_id;
        $productItem = HomebrewProduct::find($id);
    } elseif ($modelType === "Mechanic") {
        $id = $req->mechanic_id;
        $productItem = Mechanic::find($id);
    } else {
        return redirect()->back()->with('msg', 'error code given');
    }

    if ($productItem) {
        Cart::session($guestIdentifier)->add([
            'id' => $id . "-" . $modelType,
            'name' => $productItem->name,
            'price' => $productItem->price,
            'quantity' => $quantity,
            'associatedModel' => $productItem,
            'attributes' => array(
                'modelType' => $modelType
            ),
        ]);
    }
}

```

Figure 6.9.4: Product Session Cart Code Implementation

Referring to the Figure 6.9.3 and Figure 6.9.4 above, the system's initial step involves checking the authentication status of the current user. In the event that the customer is not authenticated, the system proceeds to fetch the user's existing session ID. Given the presence of two distinct cart types, a subtle but important modification is made to the session ID. Specifically, a descriptor representing the cart type is concatenated to the session ID to ensure its uniqueness. Additionally, if necessary, a similar customization can be applied to the product ID to differentiate between various types of products within the cart. Subsequent to these customizations, the cart information is efficiently stored using the provided DarryldeCode function. This is accomplished by invoking the "add()" function, which inserts the relevant data under the modified session ID. This process guarantees a robust and organized system for unauthenticated users to manage their shopping carts, with each session cart uniquely identified based on cart type and product variations, if applicable.

```

Darryldecode\Cart\CartCollection {#2075 ▾ // app\Http\Controllers\BeverageCartController.php:182
  #items: array:1 [▶
    "3-Medium-AlmondMilk" => Darrylde...\ItemCollection {#1618 ▶
      #items: array:7 [▶
        "id" => "3-Medium-AlmondMilk"
        "name" => "Milo Hot"
        "price" => "57.0"
        "quantity" => "1"
        "attributes" => Darrylde...\ItemAttributeCollection {#1619 ▶
          #items: array:2 [▶
            "size" => "Medium"
            "milkType" => "AlmondMilk"
          ]
          #escapeWhenCastingToString: false
        }
      ]
      "conditions" => []
      "associatedModel" => App\Model...\Beverage {#1620 ▶
        #connection: "mysql"
        #table: "beverages"
        #primaryKey: "id"
        #keyType: "int"
        +incrementing: true
        #with: []
        #withCount: []
        +preventsLazyLoading: false
        #perPage: 15
        +exists: true
        +wasRecentlyCreated: false
        #escapeWhenCastingToString: false
        #attributes: array:9 [▶
          "id" => 3
          "name" => "Milo Hot"
          "price" => "55.0"
          "desc" => "<p>Dive into a world of chocolaty euphoria with our Chocolate Chip. This deligh
ous sprinkling of chocolate chips. With each sip, you'll experience the luxurious combination of rich
seeking a refreshing indulgence on a sunny day or a pick-me-up after a long day, the Chocolate Chip is
Chip. This delightful beverage is a creamy concoction of smooth chocolate ice cream, swirls of cho ▶"
          "beverage_category_id" => 2
          "availability" => 100
          "img" => "yniiGyRvRmJ28S6hjt9mWAOrmYbzmy-metabW9jaGEuanBn-.webp"
          "created_at" => "2023-09-01 08:00:09"
          "updated_at" => "2023-09-14 15:33:53"
        ]
      ]
    ]
  ]
}

```

Figure 6.9.5: Sample of Beverage Cart Session Storage

```

Darryldecode\Cart\CartCollection {#2078 ▾ // app\Http\Controllers\ProductCartController.php:167
#items: array:1 [▶
  "2-HomebrewProduct" => Darrylde...\ItemCollection {#1623 ▶
    #items: array:7 [▶
      "id" => "2-HomebrewProduct"
      "name" => "coffee capsule"
      "price" => 119.0
      "quantity" => "1"
      "attributes" => Darrylde...\ItemAttributeCollection {#1624 ▶
        #items: array:1 [▶
          "modelType" => "HomebrewProduct"
        ]
        #escapeWhenCastingToString: false
      }
      "conditions" => []
    "associatedModel" => App\Model\HomebrewProduct {#1625 ▶
      #connection: "mysql"
      #table: "homebrew_products"
      #primaryKey: "id"
      #keyType: "int"
      +incrementing: true
      #with: []
      #withCount: []
      +preventsLazyLoading: false
      #perPage: 15
      +exists: true
      +wasRecentlyCreated: false
      #escapeWhenCastingToString: false
      #attributes: array:9 [▶
        "id" => 2
        "name" => "coffee capsule"
        "price" => 119.0
        "desc" => "<p>ertyui</p>"
        "homebrew_product_category_id" => 2
        "availability" => 100
        "img" => "YYg88S0xsgLHqvp1nHdBHAbAxq8mp1-metacG93ZGVyLnBuZw==.png"
        "created_at" => "2023-07-25 16:23:00"
        "updated_at" => "2023-07-25 16:23:00"
      ]
    ]
  ]
]

```

Figure 6.9.6: Sample of Product Cart Session Storage

Analyzing the content of Figure 6.9.5 and Figure 6.9.6, which provide a glimpse into the session storage of beverage and product carts, a systematic approach to storing essential cart item information is evident. Key details such as ID, name, price, and quantity are methodically recorded, while additional or exclusive data—such as customizations for the beverage cart and model types for the product cart—are appropriately stored in the attributes section. The nomenclature for the ID in the session storage aligns with Stock Keeping Unit (SKU) principles. This practice ensures a unique identifier for each product variant, accounting for specific attributes like function, color, style, and more. In this context, the SKU serves as the customization for the beverage cart and the product types (homebrew product and mechanic) for the product cart, contributing to the overall uniqueness of each ID.

On the other hand, when authenticated customers wish to perform changes on their carts, the changes will be saved into the database. Unlike the session cart, database cart could save the cart information all the time unless is

being deleted. The figures below show the code implementation for database carts.

```

$user = auth()->user();
$beverageCart = $user->beverageCart;
$beverage = Beverage::find($beverageId);

if ($beverage) {

    $existingCartItem = $beverageCart->beverage()
        ->where('beverage_id', $beverageId)
        ->whereRaw('JSON_CONTAINS(customization, ?)', [json_encode($attributes)])
        ->first();

    if ($existingCartItem) {
        // Item with the same beverage exists in the cart, update the quantity
        $newQuantity = $existingCartItem->pivot->quantity + $quantity;

        $beverageCart->beverage()->newPivotStatement()
            ->where('beverage_id', $beverageId)
            ->whereRaw('JSON_CONTAINS(customization, ?)', [json_encode($attributes)])
            ->update(['quantity' => $newQuantity]);
    } else {
        $subPrice = $beverage->price + $additionalPrice;

        // Item with the same beverage does not exist, add it as a new item
        $beverageCart->beverage()->attach($beverage->id, [
            'quantity' => $quantity,
            'customization' => json_encode($attributes),
            'sub_price' => $subPrice,
        ]);
    }
}

```

Figure 6.9.7: Beverage Database Cart Code Implementation

```

$user = auth()->user();
$productCart = $user->productCart;

if ($modelType === "HomebrewProduct") {
    $id = $req->homebrewProduct_id;
    $productItem = HomebrewProduct::find($id);
    $productType = "App\Models\HomebrewProduct";
} elseif ($modelType === "Mechanic") {
    $id = $req->mechanic_id;
    $productItem = Mechanic::find($id);
    $productType = "App\Models\Mechanic";
} else {
    return redirect()->back()->with('err', 'Code error');
}

if ($productItem) {
    $exist = $productCart->related->where('productable_id', $productItem->id)->where('productable_type', $productType)->first();
    if ($exist) {
        $initialQuantity = $exist->quantity;
        $newQuantity = $initialQuantity + $quantity;
        $exist->quantity = $newQuantity;

        $exist->quantity = $newQuantity;
        $exist->save();

        return redirect()->back()->with('msg', 'Product has been added to cart');
    } else {
        $productable = $productCart->related()->make();
        $productable->productable()->associate($productItem);
        $productable->quantity = $quantity;
        $productable->save();
    }
}

```

Figure 6.9.8: Product Database Cart Code Implementation

Similar to the session cart, upon verifying the user's authentication status, the system initiates a sequence of meticulous checks tailored to the specific cart type in use. Once these checks are completed, the system proceeds to verify the presence of the item within the specific cart database. In the event that the item is found to already exist within the cart database, the system adds up the quantity of that item. Conversely, if the item is not found in the cart, the system proceeds to save the item as a new entry in the related cart database tables. SKU is not applied in the database cart because of the polymorphic relationship applied between the products and the cart where each product ID and its model type will be saved to guarantee for the uniqueness and at the same time avoid any unnecessary confusions.

iii. Special Case : Transitioning from Guest to Authenticated User

It's common to come across situations where a guest user initially adds items to their cart and then decides to log in. In such cases, the system seamlessly transfers the cart items from the session cart to the database cart.

```
0 references | 0 overrides
public function handle(Attempting $event): void
{
    $guestIdentifier = session()->getId() . "-beverage";
    $guestBeverageCart = Cart::session($guestIdentifier)->getContent();

    if (\Auth::guest()) {
        session()->flash('guest_beverage_cart', [
            'session' => $guestIdentifier,
            'data' => $guestBeverageCart,
        ]);
    }
}
```

Figure 6.9.9: Prepare Beverage Cart For Transfer Code Implementation

```
public function handle(Attempting $event): void
{
    $guestIdentifier = session()->getId() . "-products";
    $guestProductCart = Cart::session($guestIdentifier)->getContent();

    if (\Auth::guest()) {
        session()->flash('guest_product_cart', [
            'session' => $guestIdentifier,
            'data' => $guestProductCart,
        ]);
    }
}
```

Figure 6.9.10: Prepare Product Cart For Transfer Code Implementation

Two listeners, "PrepareBeverageCartTransfer" and "PrepareProductCartTransfer," have been created to listen for the "Attempting" event, which triggers when a user attempts to log in or register. Upon receiving this event, these listeners are responsible for saving the session carts into another flash session. This step is essential due to a unique feature provided by Laravel: whenever a user performs login, register, or logout actions, the current session data gets wiped and replaced with a fresh session. This mechanism is in place to enhance security. However, it became a problem in this case because of the need to retrieve the session data after login. Therefore, saving the cart data into a session flash is crucial to ensure that it remains accessible even after the user completes login or registration actions.

```

0 references | 0 overrides
public function handle(Login $event): void
{
    $guard = $event->guard;
    $guestBeverageCart = session("guest_beverage_cart.data");

    if ($guard == 'web') {
        $user = $event->user;
        if ($user->beverageCart) {
            $beverageCart = $user->beverageCart;
        } else {
            $beverageCart = new BeverageCart();
            $user->beverageCart()->save($beverageCart);
        }
    }

    if ($guestBeverageCart !== null && count($guestBeverageCart) > 0) {
        foreach ($guestBeverageCart as $item) {
            $beverageId = $item->associatedModel->id;
            $attributes = $item->attributes;
            $quantity = $item->quantity;

            $existingDatabaseItem = $beverageCart->beverage()
                ->where('beverage_id', $beverageId)
                ->whereRaw('JSON_CONTAINS(customization, ?)', [json_encode($attributes)])
                ->first();

            if ($existingDatabaseItem) {
                $initialQuantity = $existingDatabaseItem->pivot->quantity;
                $newQuantity = $initialQuantity + $quantity;

                $beverageCart->beverage()->newPivotStatement()
                    ->where('beverage_id', $beverageId)
                    ->whereRaw('JSON_CONTAINS(customization, ?)', [json_encode($attributes)])
                    ->update(['quantity' => $newQuantity]);
            } else {
                $beverageCart->beverage()->attach($beverageId, [
                    'quantity' => $quantity,
                    'customization' => json_encode($attributes),
                    'sub_price' => $item->price,
                ]);
            }
        }
    }
}

```

Figure 6.9.11: Transfer Beverage Session Cart to Database Code Implementation

```

0 references | 0 overrides
public function handle(Login $event): void
{
    $guard = $event->guard;
    $guestProductCart = session("guest_product_cart.data");

    if ($guard == 'web') {
        $user = $event->user;
        if ($user->productCart) {
            $productCart = $user->productCart;
        } else {
            $productCart = new ProductCart();
            $user->productCart()->save($productCart);
        }

        if ($guestProductCart != null) {
            if (count($guestProductCart)) {
                foreach ($guestProductCart as $item) {
                    $productId = $item->associatedModel->id;
                    $modelType = $item->attributes->modelType;
                    $productType = "App\Models\\" . $modelType;
                    $quantity = $item->quantity;

                    $databaseItem = $productCart->related->where('productable_id', $productId)->where('productable_type', $productType)->first();
                    if ($databaseItem) {
                        $initialQuantity = $databaseItem->quantity;
                        $newQuantity = $initialQuantity + $quantity;
                        $databaseItem->quantity = $newQuantity;
                        $databaseItem->save();
                    } else {
                        $productable = $productCart->related()->make();
                        $productable->productable()->associate($item->associatedModel);
                        $productable->quantity = $quantity;
                        $productable->save();
                    }
                }
            }
        }
    }
}

```

Figure 6.9.12: Transfer Product Session Cart to Database Code Implementation

Following this, an additional pair of listeners, "TransferProductCartToUser" and "TransferBeverageCartToUser," have been implemented to listen for the "Login" event. Once a user successfully logs in, these listeners come into action, retrieving the data that was previously saved in the session flash. Subsequently, they proceed to save this retrieved data into the relevant cart database. This orchestration ensures that the user's cart information seamlessly transitions from the session storage to the relevant cart database.

6.9.2 Cart CRUD Implementation

The beverage and product cart functionality revolves around four primary operations that empower users to manage their cart items effectively:

i. View Cart

Users have the flexibility to access their product and beverage carts through two distinct interfaces: cart modals and cart pages. Within the cart modal, users can obtain a preview of their entire cart, displaying crucial details such as the total price, individual product prices, product names, and accompanying images. This preview is easily accessible by seamlessly hovering over the respective cart

icons in the navigation bar. For users seeking additional operations on their cart items, a convenient "View Complete Cart" button is available, directing users to their dedicated beverage or product cart. The figures below provide visual representations of both the cart modals and cart pages.

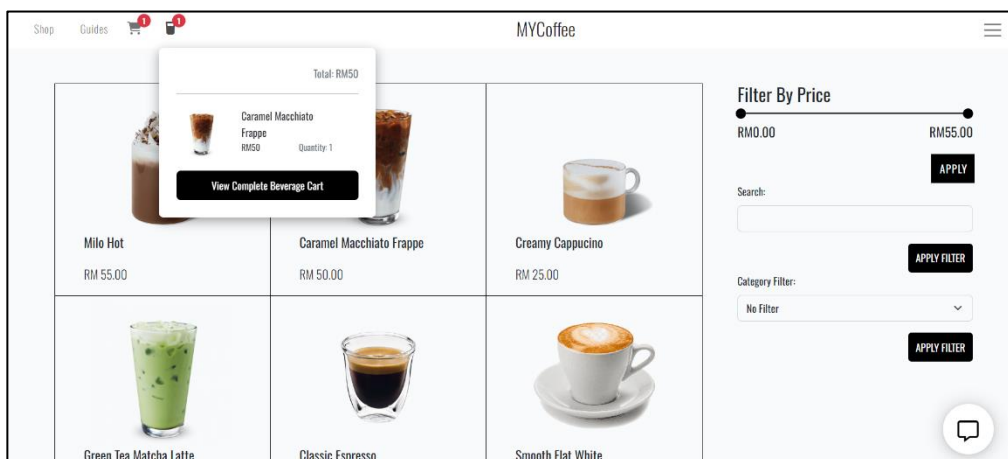


Figure 6.9.13: Beverage Cart Modal

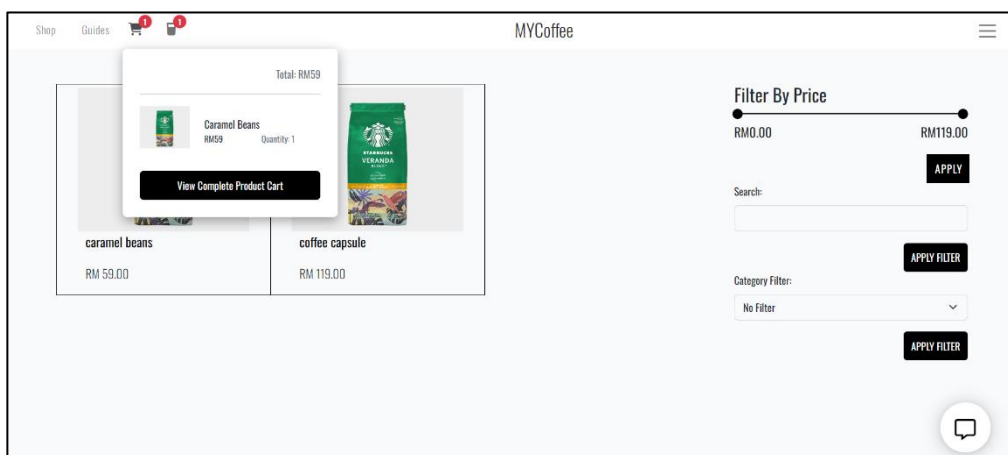


Figure 6.9.14: Product Cart Modal

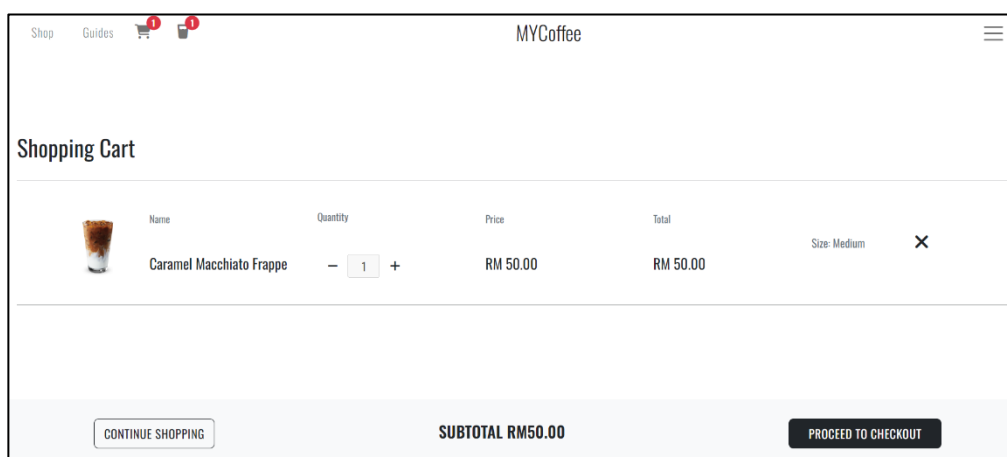


Figure 6.9.15: Beverage Cart Page

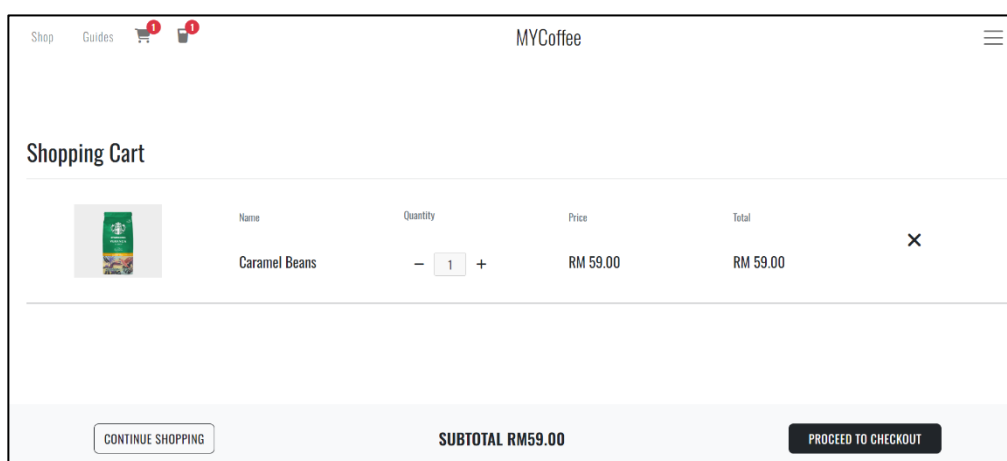


Figure 6.9.16: Product Cart Page

ii. Add to Cart

To empower users to conveniently add items to their carts at their discretion, the website strategically places "Add to Cart" buttons across various pages. In the context of beverages, users can add them to their carts exclusively from the product detail page, as the selection of beverage size is a requisite. Conversely, for other product types, users enjoy the flexibility of adding items to their respective carts directly from both the product list page and the product detail page. The figures below showcase examples of the strategic placement of the "Add to Cart" button.

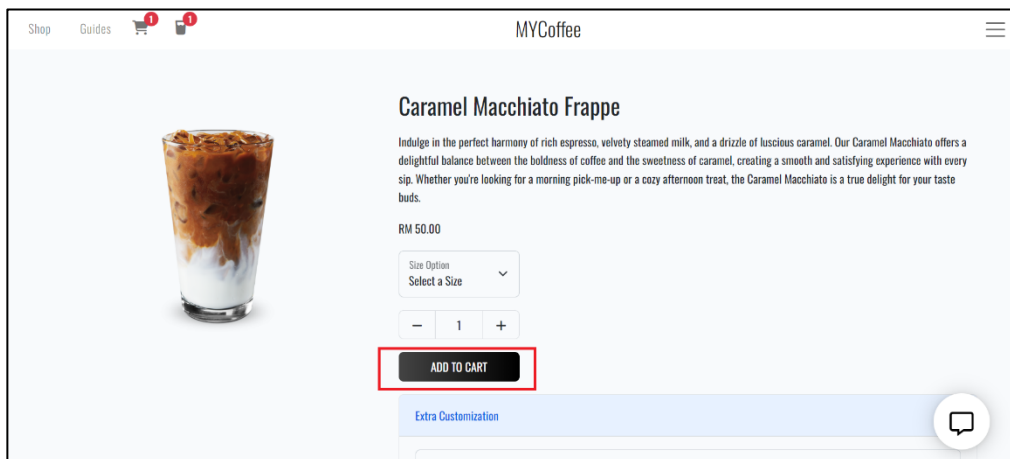


Figure 6.9.17: Add to Cart button in Beverage Product Detail Page

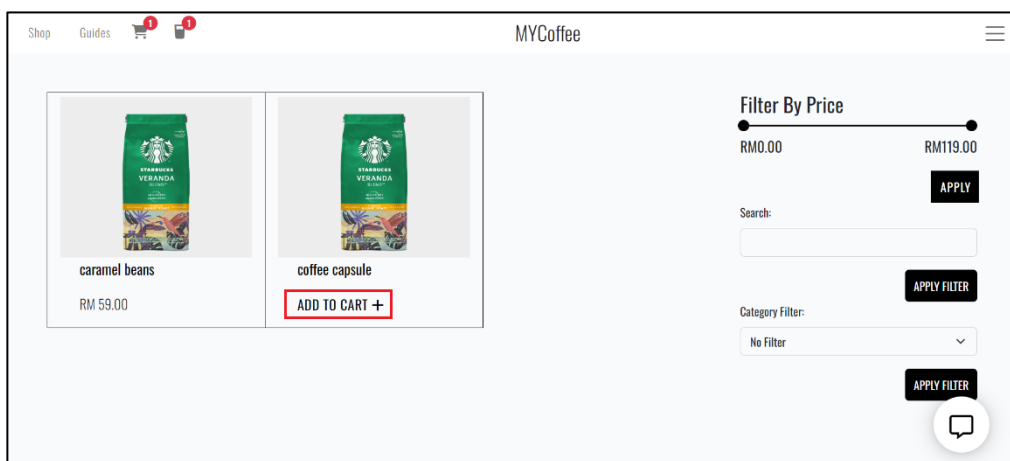


Figure 6.9.18: Add to Cart button in the Homebrew Product List Page

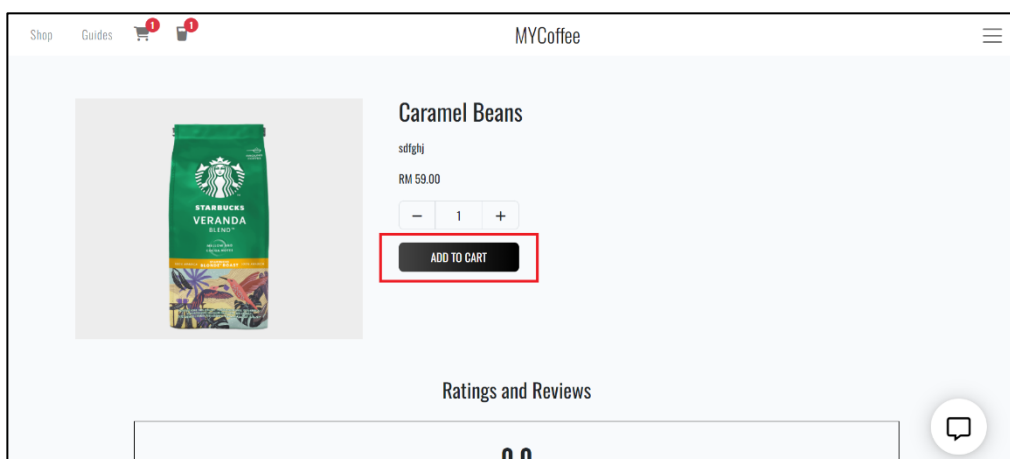


Figure 6.9.19: Add to Cart button in the Homebrew Product Detail Page

Moreover, a common scenario arises when users click on the "Add to Cart" button even when there are already existing items in their respective carts. A validation mechanism is in place where it checks whether the item already exists in the cart. If it does, the system increments the quantity of the specific item. On the other hand, if the item is not present, the system adds the product as a new cart item to the respective cart. This ensures a seamless and intuitive experience for users interacting with the shopping cart functionality.

```
if (!auth()->check()) {  
    // Guest User  
    $guestIdentifier = session()->getId() . "-beverage";  
    $beverage = Beverage::find($beverageId);  
  
    if ($beverage) {  
        foreach ($attributes as $key => $value) {  
            $beverageId .= "-" . $value;  
        }  
  
        Cart::session($guestIdentifier)->add([  
            'id' => $beverageId,  
            'name' => $beverage->name,  
            'price' => $beverage->price + $additionalPrice,  
            'quantity' => $quantity,  
            'associatedModel' => $beverage,  
            'attributes' => $attributes,  
        ]);  
  
        return redirect()->back()->with('msg', 'Product has been added to the cart');  
    }  
  
    return redirect()->back()->with('err', 'Failed to Add, Product Not Found');  
}
```

Figure 6.9.20: Add to Beverage Cart Code Implementation for Unauthenticated User

```

$guestIdentifier = session()->getId() . "-products";

if ($modelType === "HomebrewProduct") {
    $id = $req->homebrewProduct_id;
    $productItem = HomebrewProduct::find($id);
} elseif ($modelType === "Mechanic") {
    $id = $req->mechanic_id;
    $productItem = Mechanic::find($id);
} else {
    return redirect()->back()->with('msg', 'error code given');
}

if ($productItem) {
    Cart::session($guestIdentifier)->add([
        'id' => $id . "-" . $modelType,
        'name' => $productItem->name,
        'price' => $productItem->price,
        'quantity' => $quantity,
        'associatedModel' => $productItem,
        'attributes' => array(
            'modelType' => $modelType
        ),
    ]);
}

return redirect()->back()->with('msg', 'Product is added into product cart');

```

Figure 6.9.21: Add to Product Cart Code Implementation for Unauthenticated User

For unauthenticated users utilizing the DarryldeCode package for session carts, the implementation is designed to effortlessly handle scenarios where the product is being added. If the product is not already present in the cart, the package seamlessly adds it as a new cart item. On the other hand, if the product does exist in the cart, the package intuitively increments the quantity without requiring any additional code implementation.

```

$user = auth()->user();
$beverageCart = $user->beverageCart;
$beverage = Beverage::find($beverageId);

if ($beverage) {

    $existingCartItem = $beverageCart->beverage()
        ->where('beverage_id', $beverageId)
        ->whereRaw('JSON_CONTAINS(customization, ?)', [json_encode($attributes)])
        ->first();

    if ($existingCartItem) {
        // Item with the same beverage exists in the cart, update the quantity
        $newQuantity = $existingCartItem->pivot->quantity + $quantity;

        $beverageCart->beverage()->newPivotStatement()
            ->where('beverage_id', $beverageId)
            ->whereRaw('JSON_CONTAINS(customization, ?)', [json_encode($attributes)])
            ->update(['quantity' => $newQuantity]);
    } else {
        $subPrice = $beverage->price + $additionalPrice;

        // Item with the same beverage does not exist, add it as a new item
        $beverageCart->beverage()->attach($beverage->id, [
            'quantity' => $quantity,
            'customization' => json_encode($attributes),
            'sub_price' => $subPrice,
        ]);
    }

    return redirect()->back()->with('msg', 'Product has been added into the cart');
} else {
    return redirect()->back()->with('msg', 'Product Not Found');
}

```

Figure 6.9.22: Add to Beverage Cart Code Implementation for Authenticated User

```

if ($productItem) {
    $exist = $productCart->related->where('productable_id', $productItem->id)->where('productable_type', $productType)->first();
    if ($exist) {
        $initialQuantity = $exist->quantity;
        $newQuantity = $initialQuantity + $quantity;
        $exist->quantity = $newQuantity;

        $exist->quantity = $newQuantity;
        $exist->save();

        return redirect()->back()->with('msg', 'Product has been added to cart');
    } else {
        $productable = $productCart->related()->make();
        $productable->productable()->associate($productItem);
        $productable->quantity = $quantity;
        $productable->save();

        return redirect()->back()->with('msg', 'Product has been added to cart');
    }
}

```

Figure 6.9.23: Add to Product Cart Code Implementation for Authenticated User

In contrast, for the authenticated user, the validation mechanism implemented in such a way that if the product existed in the respective cart, then it will increase the quantity of the specific cart item, while if the product does

not exist in the cart, then it will add the product as a new cart item into the database cart respectively.

iii. Update the Cart

Other than that, the cart facilitates the operation of dynamically updating the quantity of cart items, allowing users to seamlessly increase or decrease the quantity as needed. This process occurs in real-time through AJAX, ensuring that the page remains responsive without the need for a complete reload. As users modify the quantity, both the total price and subtotal for each cart item are updated instantaneously, providing a fluid and uninterrupted shopping experience.

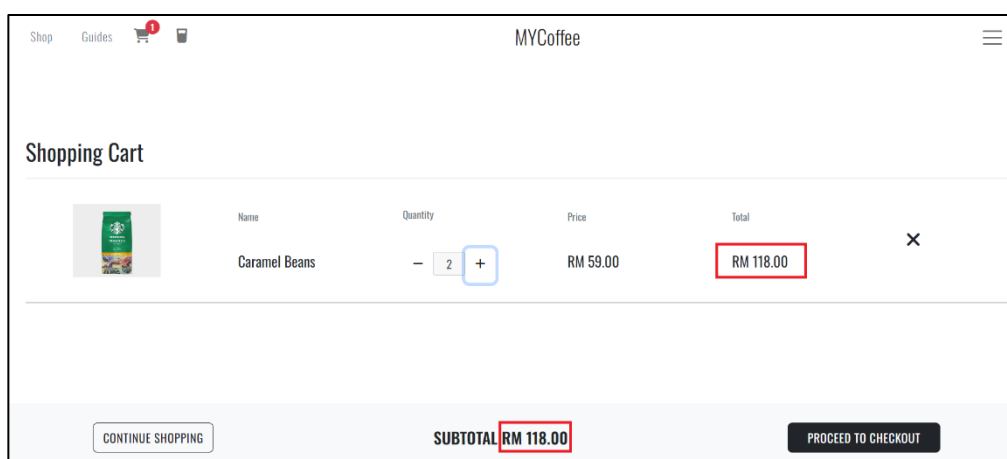


Figure 6.9.24: Sample of Increase in Quantity of one Cart Item in Product Cart

```
$.ajax({
  url: url,
  type: "POST",
  data: {
    _token: csrfToken,
    id: index,
    quantity: quantity,
  },
  success: function (response) {
    var totalPrice = document.getElementById("total-price-" + index);
    var subTotal = document.getElementById("subtotal");
    var modalSubTotal = document.getElementById(modalTotal);

    totalPrice.textContent = "RM " + response.price.toFixed(2);
    subTotal.textContent = "RM " + response.subTotal.toFixed(2);
    modalSubTotal.textContent = "RM " + response.subTotal.toFixed(2);
  },
  error: function (xhr, status, error) {
    // Handle error
    console.log(error);
  },
});
```

Figure 6.9.25: AJAX code Implementation to connect to Laravel Controller

```

if (!auth()->check()) {
    $id = $req->id;
    $guestIdentifier = session()->getId() . "-products";
    Cart::session($guestIdentifier)->update($id, [
        'quantity' => $quantity,
    ]);

    $price = Cart::session($guestIdentifier)->get($id)->getPriceSum();
    $subTotal = Cart::session($guestIdentifier)->getSubTotal();

    return response()->json([
        'price' => $price,
        'subTotal' => $subTotal,
    ]);
} else {
    $user = auth()->user();
    $productCart = $user->productCart;

    $id = $req->id;

    $updateProduct = $productCart->related->find($id);
    if ($updateProduct) {
        $updateProduct->quantity = $quantity;
        $updateProduct->save();

        $itemSubTotal = $updateProduct->productable->price * $quantity;
        $updatedProductCart = $user->fresh()->productCart;
        $productable = $updatedProductCart->related;

        $orderSubTotal = $productable->sum(function ($item) {
            return $item->productable->price * $item->quantity;
        });

        return response()->json([
            'itemSubTotal' => $itemSubTotal,
            'subTotal' => $orderSubTotal,
        ]);
    }
}

```

Figure 6.9.26: Sample Code Implementation for Both Unauthenticated and Authenticated User to Update Cart

Upon reviewing Figure 6.9.25 and Figure 6.9.26, it's evident that the AJAX functionality operates by initiating a POST request to the specified URL, directing it to the designated "update cart" method in the Laravel route. Upon successful transmission, the quantity undergoes updating, contingent upon the user's authentication status. Subsequently, the system calculates both the total price and subtotal price for the specific cart item, sending these computed values back as a response to the initial request. Upon successfully fetching this response, the prices are dynamically updated in real-time, providing users with an instantaneous and seamless experience during quantity adjustments in the cart.

iv. Delete Cart

Users have the convenient option to remove items from their respective carts by simply clicking on the "x" button within the cart page. This removal process operates similarly for both unauthenticated and authenticated users, ensuring a consistent and intuitive experience across different user scenarios.

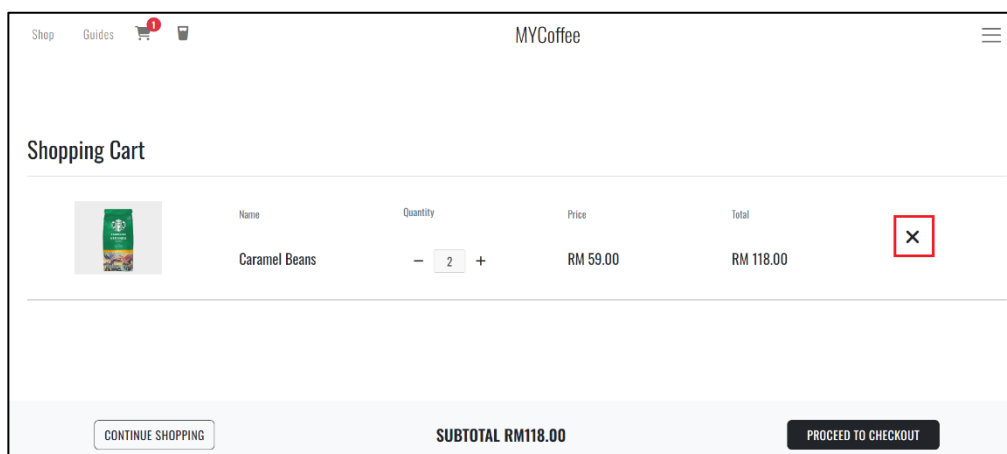


Figure 6.9.27: Sample of Product Cart with Delete Item Icon

```

1 reference | 0 overrides
public function deleteBeverageCart($id)
{
    if (!auth()->check()) {
        $guestIdentifier = session()->getId() . "-beverage";

        Cart::session($guestIdentifier)->remove($id);
        return redirect()->back()->with('msg', 'Successfully Removed from Beverage Cart');
    } else {
        $user = auth()->user();
        $beverageCart = $user->beverageCart;
        $deleteBeverage = $beverageCart->beverage()->wherePivot('id', $id)->first();
        if ($deleteBeverage) {
            $beverageCart->beverage()->wherePivot('id', $id)->detach();
            return redirect()->back()->with('msg', 'Product has been deleted');
        } else {
            return redirect()->back()->with('msg', 'Product Not Found');
        }
    }
}
}

```

Figure 6.9.28: Delete Item from Beverage Cart Code Implementation

```

1 reference | 0 overrides
public function deleteProductCart($id)
{
    if (!auth()->check()) {
        $guestIdentifier = session()->getId() . "-products";

        Cart::session($guestIdentifier)->remove($id);

        return redirect()->back()->with('msg', 'Successfully Removed from Product Cart');
    } else {
        $user = auth()->user();
        $productCart = $user->productCart;
        $deleteProduct = $productCart->related->find($id);

        if ($deleteProduct) {
            $deleteProduct->delete();

            return redirect()->back()->with('msg', 'Successfully deleted from Product Cart');
        } else {
            return redirect()->back()->with('msg', 'Product Not Found');
        }
    }
}

```

Figure 6.9.29: Delete Item from Product Cart Code Implementation

6.10 Stripe Payment Gateway

In this project, Stripe serves as the chosen payment gateway, facilitating a wide array of payment processes. To align with the Malaysian market, this project incorporates payment methods such as GrabPay and online banking. Stripe offers an assortment of payment methods and features, including a user-friendly payment UI, simplifying the development process. Furthermore, Stripe provides a Laravel package, streamlining the implementation of Stripe as the payment gateway. Figures below illustrates the implementation of Stripe as the project's payment gateway.


```

foreach ($cartItems as $item) {
    if ($type == "beverage") {
        $customizations = auth()->check() ? json_decode($item->pivot->customization) : $item->attributes;
        foreach ($customizations as $key => $value) {
            $descriptions .= ucwords(implode(' ', preg_split('/(?=[A-Z])/', $key))) . ": " . ucwords(implode(' ', preg_split('/(?=[A-Z])/', $value)));
        }

        $lineItems[] = [
            'price_data' => [
                'currency' => 'myr',
                'product_data' => [
                    'name' => ucwords($item->name),
                    'description' => $descriptions,
                    'metadata' => [
                        'order_item_id' => auth()->check() ? $item->id : $item->associatedModel->id,
                        'modelType' => "App\Models\\Beverage",
                        'customization' => auth()->check() ? $item->pivot->customization : json_encode($item->attributes),
                    ],
                ],
            ],
            'unit_amount' => (auth()->check() ? $item->pivot->sub_price : $item->price) * 100,
            'quantity' => auth()->check() ? $item->pivot->quantity : $item->quantity,
        ];
    } elseif ($type == "product") {
        $lineItems[] = [
            'price_data' => [
                'currency' => 'myr',
                'product_data' => [
                    'name' => auth()->check() ? ucwords($item->productable->name) : $item->name,
                    'metadata' => [
                        'order_item_id' => auth()->check() ? $item->productable->id : $item->associatedModel->id,
                        'modelType' => auth()->check() ? $item->productable->type : "App\Models\\" . $item->attributes->modelType,
                    ],
                ],
            ],
            'unit_amount' => (auth()->check() ? $item->productable->price : $item->price) * 100,
        ];
    }
}

```

Figure 6.10.1: Stripe Payment Gateway Code Implementation (Part 1)

```

        'quantity' => $item->quantity,
    ];
}
}

$checkout_session = $stripe->checkout->sessions->create([
    'line_items' => $lineItems,
    'mode' => 'payment',
    'success_url' => route('checkout.success', [], true) . "?session_id={CHECKOUT_SESSION_ID}",
    'cancel_url' => route('checkout.cancel', [], true) . "?session_id={CHECKOUT_SESSION_ID}",
    'customer_creation' => 'always',
]);

$order = new Order();
$order->total_price = $totalPrice;
$order->status = 'unpaid';
$order->type = $type;
$order->session_id = $checkout_session->id;
$order->save();

foreach ($lineItems as $item) {
    $orderItem = $order->related()->make();
    $orderItem->itemable_id = $item['price_data']['product_data']['metadata']['order_item_id'];
    $orderItem->itemable_type = $item['price_data']['product_data']['metadata']['modelType'];
    $orderItem->quantity = $item['quantity'];
    $orderItem->customization = $item['price_data']['product_data']['metadata']['customization'] ?? null;
    $orderItem->sub_price = intval($item['price_data']['unit_amount']) / 100;
    $orderItem->save();
}

return redirect($checkout_session->url);

```

Figure 6.10.2: Stripe Payment Gateway Code Implementation (Part 2)

The Stripe checkout session is initiated once all the necessary product information has been accurately assigned. Subsequently, it will seamlessly redirect users to the payment page, which is fully integrated with Stripe's secure payment processing system. Concurrently, in the background, the database will

create a new order entry, leveraging the product information to populate the order details.

```
public function success(Request $req)
{
    $stripe = new \Stripe\StripeClient(env('STRIPE_SECRET_KEY'));
    $sessionId = $req->get('session_id');

    $session = $stripe->checkout->sessions->retrieve($sessionId);
    if (!$session) {
        return view('payment.failure', ['message' => 'Invalid Session ID Provided']);
    }
    $customer = $stripe->customers->retrieve($session->customer);

    $order = Order::where(['session_id' => $session->id])->whereIn('status', ['unpaid', 'paid'])->first();
    if (!$order) {
        return view('payment.failure', ['message' => 'Order does not exist']);
    }

    if ($order->status == 'unpaid') {
        $this->updateOrderSuccess($order, $customer);
    }
}
```

Figure 6.10.3: Stripe Payment Gateway Code Implementation (Part 3)

```
public function cancel(Request $req)
{
    $stripe = new \Stripe\StripeClient(env('STRIPE_SECRET_KEY'));
    $sessionId = $req->get('session_id');

    try {
        $session = $stripe->checkout->sessions->retrieve($sessionId);

        if (!$session) {
            return view('payment.failure', ['message' => 'Invalid Session ID Provided']);
        }

        $order = Order::where(['session_id' => $session->id, 'status' => 'unpaid'])->first();

        if (!$order) {
            return view('payment.failure', ['message' => 'Order does not exist']);
        }

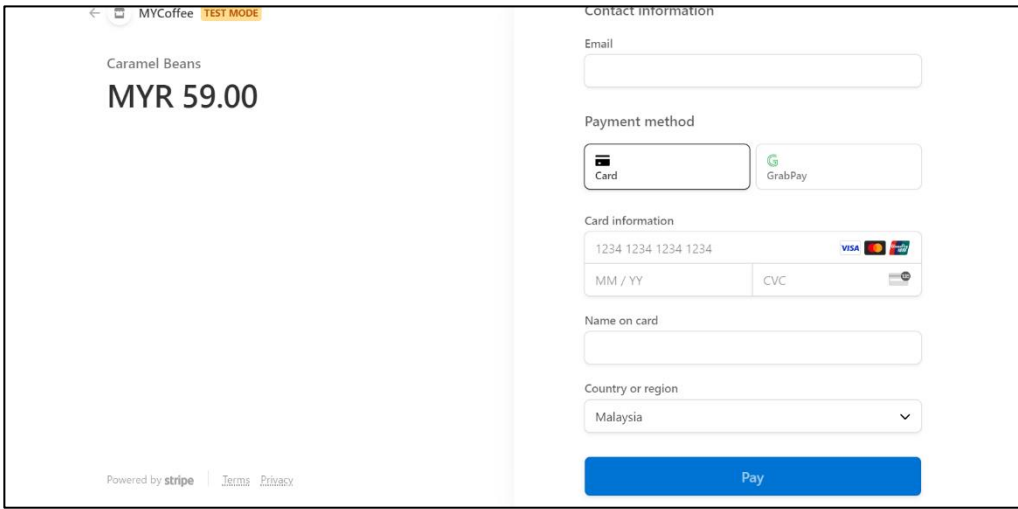
        $order->delete();

        return view('payment.failure', ['message' => 'Payment has been cancelled!']);
    } catch (Exception $e) {
        return view('payment.failure', ['message' => $e->getMessage()]);
    }
}
```

Figure 6.10.4: Stripe Payment Gateway Code Implementation (Part 4)

Upon submission of the payment form, the system distinguishes between two possible outcomes. If the payment is successfully processed, the user is redirected to a payment success page. Simultaneously, on the backend,

the system updates the payment status of the corresponding order to "paid" and clearing the cart informations. Furthermore, an order confirmation email is dispatched to the customer. Conversely, in cases of payment failure or user-initiated payment cancellation, the system redirects the user to a payment failure page, signaling that the payment process was not finalized. Behind the scenes, the system takes prompt action by promptly removing the specific order record associated with the unsuccessful or canceled payment. The figures displayed below showcase several screenshots of the related payment pages.



MYCoffee TEST MODE

Caramel Beans
MYR 59.00

Contact information
Email

Payment method
 Card GrabPay

Card information
1234 1234 1234 1234 VISA
MM / YY CVC

Name on card

Country or region
Malaysia

Powered by stripe | [Terms](#) | [Privacy](#)

Pay

Figure 6.10.5: Sample of Stripe Payment Page



Figure 6.10.6: Payment Failure Page

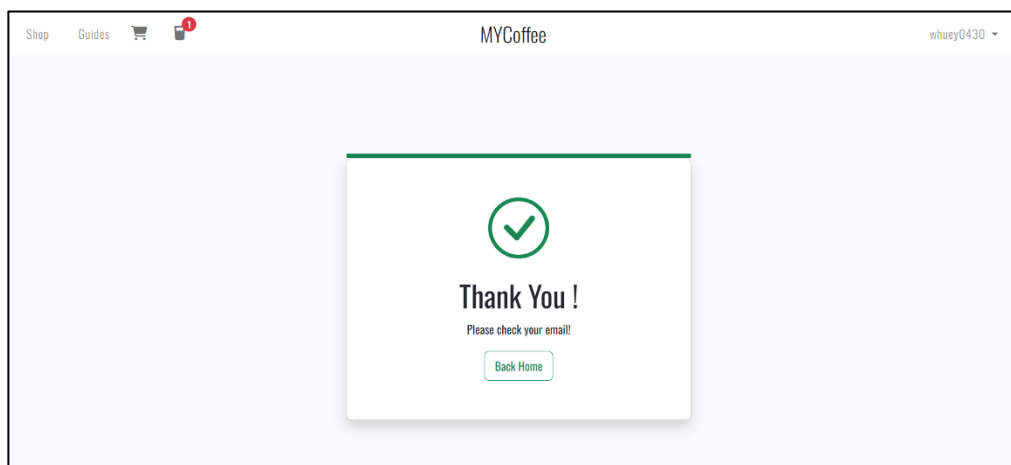


Figure 6.10.7: Payment Success Page

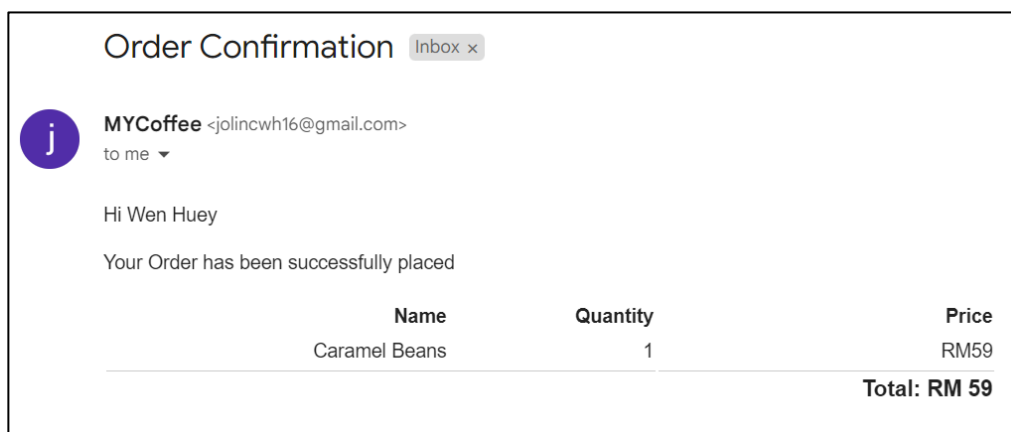


Figure 6.10.8: Order Confirmation Email

6.11 Order List

The website extends a valuable feature allowing users to access their order history, providing a comprehensive list of past purchases alongside detailed order items. It's worth noting that users have the privilege to view orders exclusively, with the processes of order creation and deletion detailed in the earlier section on the Stripe Payment Gateway. The figure below show the sample of the order list.

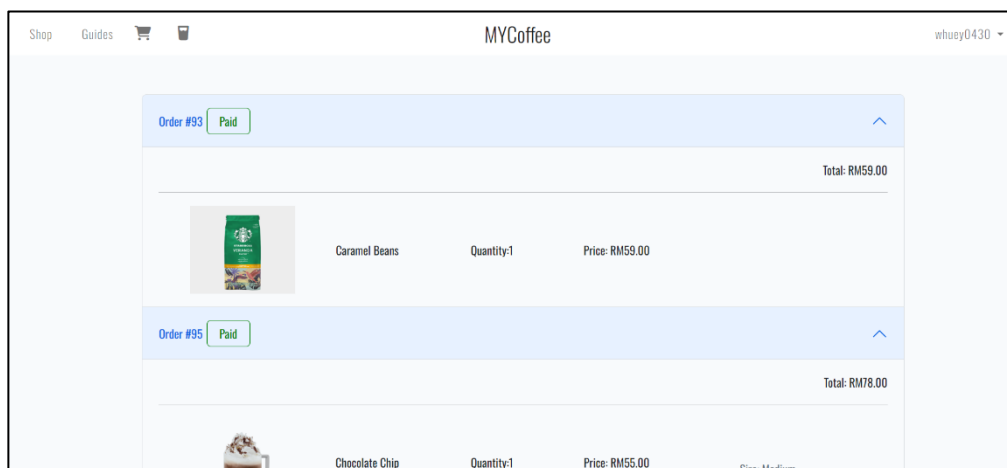


Figure 6.11.1: Order History List

6.12 Admin Panel

The admin panel in this project is constructed using the Laravel package known as Filament. Filament offers a streamlined integration between the admin panel and the database by enabling the creation of Filament resources through simple commands. In the context of this project, the primary responsibilities of the admin panel include product management, oversight of guiding materials, and management of data embedding files.

6.12.1 Manage Products

i. Add New Product

Admin has the capability to seamlessly add new products through the admin panel. This functionality encompasses a diverse range of products, including beverages, homebrew products, and mechanics. For each product, administrators can input essential information such as the name, price, image, and other relevant details. Since the form and process of creating a new product are the same for the three products, thus, only one example will be shown throughout the CRUD process.

The screenshot shows the 'Create Beverage' form in the MYCoffee application. The form is titled 'Create Beverage' and is located in the 'Beverages / Create' section. The form includes the following fields and controls:

- Name**: A text input field.
- Price(RM)**: A text input field with a validation message: "Please fill out this field."
- Description**: A rich text editor with options for Bold (B), Italic (I), Underline (U), Strikethrough (ABC), and text alignment (Left, Center, Right). There are also undo and redo buttons.
- Beverage Category**: A dropdown menu with the text "Select an option".
- Availability**: A text input field.
- Thumbnail**: A file upload area with the text "Drag & Drop your files or Browse".

The sidebar on the left contains the following navigation items:

- Dashboard
- Embedding Collections
- Orders
- PRODUCT CATEGORIES
 - Beverage Categories
 - Homebrew Product Categories
 - Mechanic Categories
- PRODUCT MANAGEMENT
 - Beverages** (highlighted)
 - Homebrew Products
 - Mechanics

Figure 6.12.1: Sample of Creating New Beverage Form

ii. View Product List

Furthermore, admin could also view the product list and perform operations such as view in detail, edit and delete products either one by one or in bulk. The figure below shows the sample of beverage list.

The screenshot shows the 'Beverages List' page in the MYCoffee application. The page is titled 'Beverages' and is located in the 'Beverages / List' section. The page includes a 'New beverage' button in the top right corner. The main content is a table of beverages with the following columns: Name, Thumbnail, Updated at, and actions (View, Edit, Delete). The table contains the following data:

Name	Thumbnail	Updated at	View	Edit	Delete
<input type="checkbox"/> Milo Hot		3 weeks ago			
<input type="checkbox"/> Caramel Macchiato Frappe		3 weeks ago			
<input type="checkbox"/> Creamy Cappuccino		1 month ago			
<input type="checkbox"/> Green Tea Matcha Latte		1 month ago			
<input type="checkbox"/> Classic Espresso		3 weeks ago			
<input type="checkbox"/> Smooth Flat White		3 weeks ago			

The sidebar on the left contains the following navigation items:

- Dashboard
- Embedding Collections
- Orders
- PRODUCT CATEGORIES
 - Beverage Categories
 - Homebrew Product Categories
 - Mechanic Categories
- PRODUCT MANAGEMENT
 - Beverages** (highlighted)
 - Homebrew Products
 - Mechanics

Figure 6.12.2: Sample of Beverage List

iii. Edit Existing Product Information

Other than that, admin possesses the capability to edit existing product information, and the "Edit" button is strategically placed for easy access across various locations, including the product detail page and the product list page. This thoughtful placement enhances the efficiency for admin, ensuring they can swiftly locate and modify product details whenever necessary. Upon submission of the edit form, the latest information seamlessly updated to the database.

The screenshot shows the 'Edit Beverage' form in the MYCoffee application. The form is titled 'Edit Beverage' and has a 'View' button and a red 'Delete' button in the top right corner. The form contains the following fields:

- Name:** Milo Hot
- Price(RM):** 55.00
- Description:** A rich text editor containing the text: "Dive into a world of chocolaty euphoria with our Chocolate Chip. This delightful beverage is a creamy concoction of smooth chocolate ice cream, swirls of chocolate syrup, and a generous sprinkling of chocolate chips. With each sip, you'll experience the luxurious combination of rich flavors and delightful textures, making it a heavenly treat for all chocolate enthusiasts. Whether you're seeking a refreshing indulgence on a sunny day or a pick-me-up after a long day, the Chocolate Chip is the ultimate chocolate-lover's dream."
- Beverage Category:** chocolate
- Availability:** 100

Figure 6.12.3: Sample of Edit Beverage Form

iv. Delete Existing Product

Similarly to the "Edit" functionality, the strategically placed "Delete" button is easily accessible on both the product list and product detail pages. Clicking on the "Delete" button triggers an alert confirmation message, providing a necessary step for administrators to reconfirm their intent to delete the product. Upon the administrator's reconfirmation, the product undergoes deletion from the database.

The screenshot shows the 'Edit Beverage' form with a modal alert message overlay. The alert is titled 'Delete Beverage' and contains the text: "Are you sure you would like to do this?". There are two buttons: a white 'Cancel' button and a red 'Confirm' button. The background form is dimmed.

Figure 6.12.4: Alert Message for Delete Confirmation

6.12.2 Manage Guiding Materials

i. Add New Guiding Materials

The guiding materials encompasses the recipe and brewing guides where admin could add new guides for both. Each guiding material will have information required for admin to fill in such as the name, description, images and others. In addition to that, there will be tips or tools inputs where if it is not enough, admin could clicks on the “More” button to increase the column. Upon submitting the form, the information will then be saved into the database.

Figure 6.12.5: Sample of Basic Information in the Brewing Guide Form

Figure 6.12.6: Sample of “More” button to increase the inputs row

ii. View Guiding Material List

Admin could also view the guiding material list where operations such as view in detail, edit and delete in either one or in bulk could be done in the guiding material list as well.

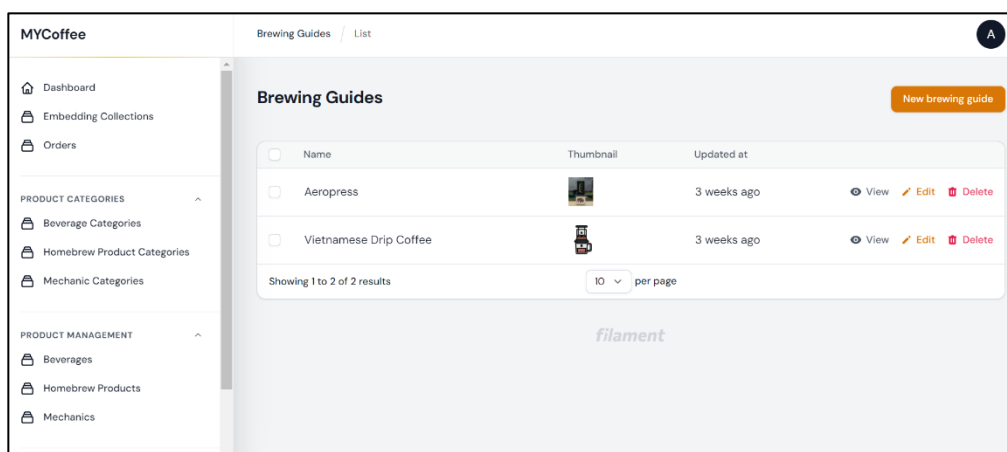


Figure 6.12.7: Sampel of Brewing Guide List

iii. Edit Existing Guiding Material

Users could also perform edit actions by clicking on the “Edit” button which can be seen in both guiding material list and guiding material detail pages. Upon submitting the edit form, the latest information will then be updated to the database.

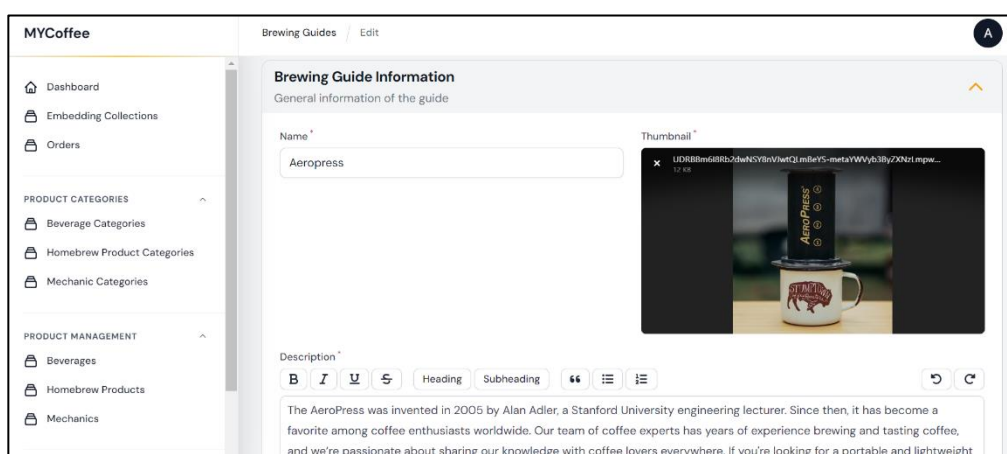


Figure 6.12.8: Sample of Edit Brewing Guide Form

iv. Delete Existing Guiding Material

Lastly, users could also delete existing guiding materials in which the “Delete” button can be seen in both guiding material list and guiding material detail pages. An alert message will pop up when users click on the “Delete” button confirming to proceed the action. Upon submitting the delete request, the specific guiding material will be deleted from database.

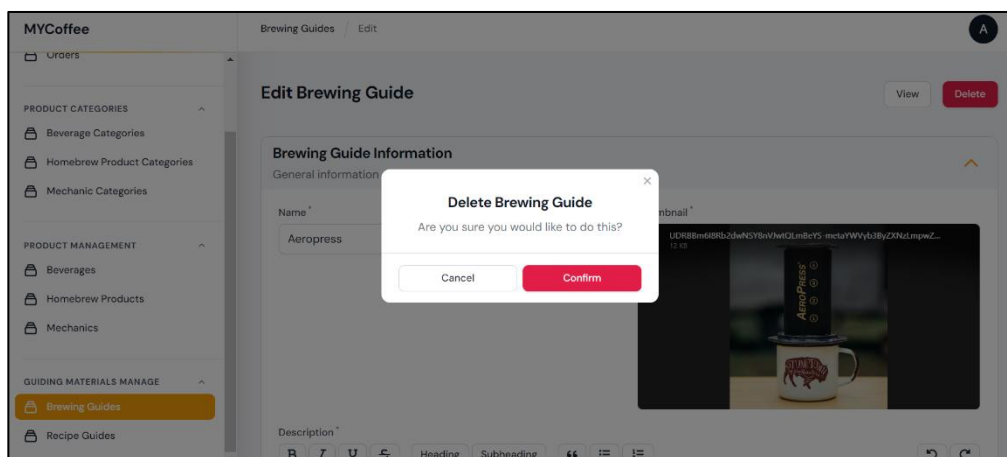


Figure 6.12.9: Alert Pop up to Confirm Delete Action

6.12.3 Manage Data Embedding File

i. Add New Data Embedding File

Admin have the capability to upload new data embedding files, initiating a process to transmit the file to OpenAI for the generation of embedding vectors. However, a unique validation is in place to ensure a smooth workflow. Before adding a new file, the system checks for the existence of any prior data embedding file. If one is found, a notification prompts the admin to delete the existing file before proceeding with the addition of a new one. Examining the provided code snippet in Figure 6.12.11, the uploading process seamlessly integrates with the OpenAI Embedding functionality. Upon file upload, the system promptly interacts with OpenAI to generate embedding vectors, which are subsequently stored in the database.

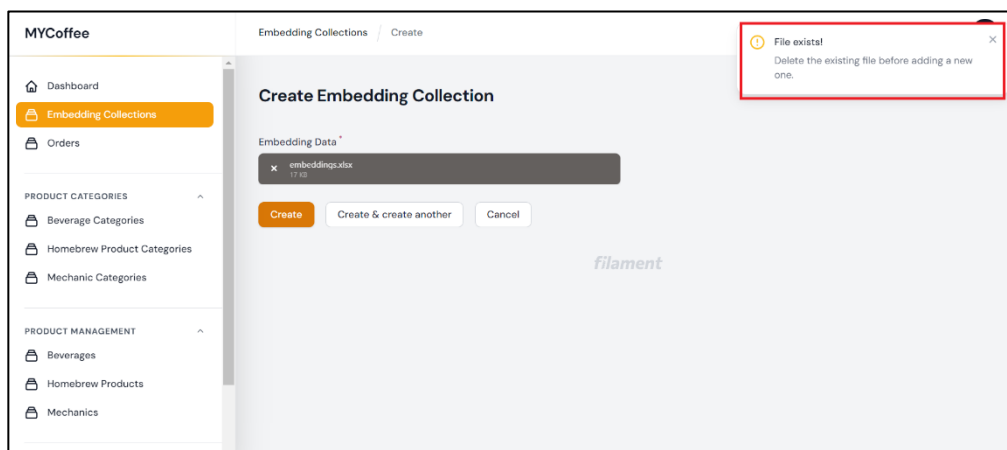


Figure 6.12.10: Notification telling admin to delete existing file before adding a new one

```

0 references | 0 overrides
protected function afterCreate()
{
    $newCollection = EmbeddingCollection::first();
    $vectorService = new VectorService;
    $filePath = storage_path() . "/app/public/" . $newCollection->file_path;
    $vectorService->extractDataFromCSV($filePath);
}

```

Figure 6.12.11: Code Snippet of After Successfully Uploaded the File

ii. Delete Existing Data Embedding File

In addition to file uploads, admin could effortlessly manage existing data embedding files by utilizing the strategically placed "Delete" button on the product list page. This streamlined interface offers admin the option to delete an existing file, especially when they intend to upload a newer version. Upon clicking the "Delete" button, an alert message intuitively pops up, seeking confirmation to proceed with the deletion action. This two-step confirmation process ensures that admin can make informed decisions, preventing accidental deletions and maintaining control over the data embedding file management process.

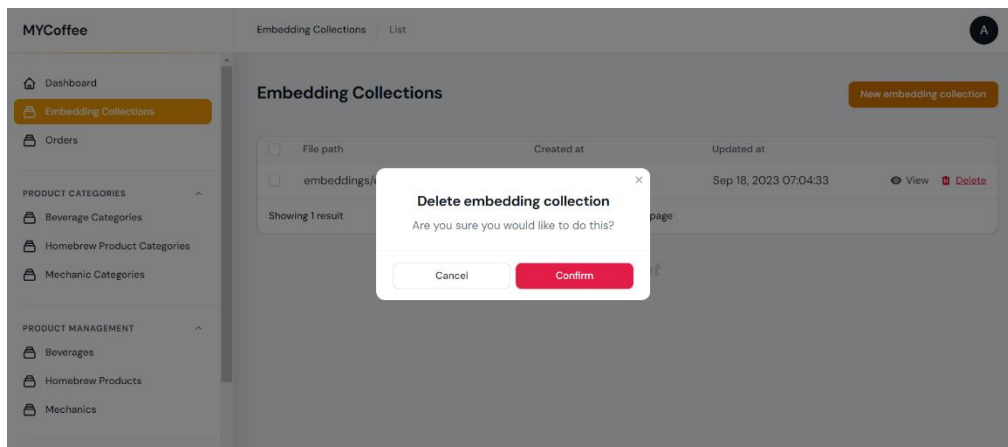


Figure 6.12.12: Alert Message to Reconfirm Delete Action

CHAPTER 7

SYSTEM TESTING

7.1 Unit Testing

Unit testing is a critical aspect of software development, focusing on examining the smallest individual components or units of a project's codebase. Its importance becomes evident when dealing with projects of relatively high complexity, as it plays a pivotal role in uncovering errors that may lurk within the intricate interplay of these components. By subjecting each unit to rigorous testing in isolation, developers can ensure that they function as intended and catch any deviations or defects early in the development process. Tables below show the unit testing for each module.

Table 7.1.1: Unit Test Case – User Register

Test Case ID	WBCS-001	Module Name	Register Module	
Test Title	Unit Test Case for Register Account			
Pre-condition	-			
Test Case Description	Execution Steps	Expected Result	Actual Result	Status
Input new email and valid password criteria	<ol style="list-style-type: none"> 1. Input a valid name. 2. Input a valid email. 3. Input a valid password. 4. Input a valid password confirmation. 5. Clicks the Register button. 	The system redirects to the logged in home page.	The system redirects to the logged in home page.	Pass
Input existing email	<ol style="list-style-type: none"> 1. Input a valid name 2. Input an existing email. 3. Input valid password. 4. Input valid password confirmation. 5. Clicks the Register button. 	The system displays error message of invalid email.	The system displays error message of invalid email.	Pass

Password does not meet criteria	<ol style="list-style-type: none"> 1. Input a valid name. 2. Input a valid email. 3. Input a valid password. 4. Input invalid password confirmation 5. Clicks the Register button. 	The system displays error message of invalid password criteria.	The system displays error message of invalid password criteria.	Pass
Empty inputs	<ol style="list-style-type: none"> 1. Clicks the Register button. 	The system displays alert message of invalid inputs.	The system displays alert message of invalid inputs.	Pass
Input invalid format of email	<ol style="list-style-type: none"> 1. Input valid name. 2. Input invalid email format. 3. Input valid password. 4. Input valid password confirmation. 5. Clicks the Register button. 	The system displays error message of invalid email format.	The system displays error message of invalid email format.	Pass

Table 7.1.2: Unit Test Case – User Login

Test Case ID	WBCS-002	Module Name	Login Module
Test Title	Unit Test Case for User Login Account		

Pre-condition	User has a registered account			
Test Case Description	Execution Steps	Expected Result	Actual Result	Status
Input registered email and correct password	<ol style="list-style-type: none"> 1. Input a valid email. 2. Input a valid password. 3. Clicks the Login button. 	The system redirects to the logged in home page.	The system redirects to the logged in home page.	Pass
Input a unregistered email and correct password	<ol style="list-style-type: none"> 1. Input an invalid email. 2. Input valid password. 3. Clicks the Login button. 	The system displays error message of unregistered email.	The system displays error message of unregistered email.	Pass
Input incorrect password	<ol style="list-style-type: none"> 1. Input a valid email. 2. Input a incorrect password. 3. Clicks the Login button. 	The system displays error message of incorrect password.	The system displays error message of incorrect password.	Pass
Empty inputs	<ol style="list-style-type: none"> 1. Clicks the Login button. 	The system displays alert message of invalid inputs.	The system displays alert message of invalid inputs.	Pass
Input invalid format of email	<ol style="list-style-type: none"> 1. Input invalid email format. 2. Input valid password. 3. Clicks the Login button 	The system displays error message of invalid email format.	The system displays error message of invalid email format.	Pass

Table 7.1.3: Unit Test Case – View Product List

Test Case ID	WBCS-003	Module Name	Product Module		
Test Title	Unit Test Case for User view product list				
Pre-condition	-				
Test Case Description	Execution Steps	Expected Result	Actual Result	Status	
View beverage list	1. Clicks the “Beverages” button in the “Shop” dropdown.	The system redirects to the beverage list page.	The system redirects to the beverage list page.	Pass	
View homebrew product list	1. Clicks the “Homebrew Products” button in the “Shop” dropdown.	The system redirects to the homebrew product list page.	The system redirects to the homebrew product list page.	Pass	
Input incorrect password	1. Clicks the “Machines & Barista Tools” button in the “Shop” dropdown.	The system redirects to the mechanics list page.	The system redirects to the mechanics list page.	Pass	

Table 7.1.4: Unit Test Case – Search and Filter Product

Test Case ID	WBCS-004	Module Name	Product Module		
Test Title	Unit Test Case for User search and filter product				

Pre-condition	-			
Test Case Description	Execution Steps	Expected Result	Actual Result	Status
Input keywords for product searching	<ol style="list-style-type: none"> 1. Inputs a keyword in the search textbox. 2. Clicks the Apply button. 	The system displays the keyword search filtered product list.	The system redirects to the beverage list page.	Pass
Empty input for product searching	<ol style="list-style-type: none"> 1. Clicks the Apply button. 	The system displays the product list with no search filter	The system displays the product list with no search filter	Pass
Filter by price range	<ol style="list-style-type: none"> 1. Select to the minimum and maximum price range 2. Click the Apply button 	The system displays the price range filtered product list.	The system displays the price range filtered product list.	Pass
Remove filter of price range	<ol style="list-style-type: none"> 1. Clicks the Reset button 	The system displays product list with no price filter.	The system displays product list with no price filter.	Pass
Filter by product categories	<ol style="list-style-type: none"> 1. Select product category option 2. Clicks the Apply button 	The system displays the product category filtered product list.	The system displays the product category filtered product list.	Pass

Remove filter of product categories	<ol style="list-style-type: none"> 1. Select the No Filter option 2. Clicks the Apply button 	The system displays product list with no product category filter.	The system displays product list with no product category filter.	Pass
-------------------------------------	--	---	---	------

Table 7.1.5: Unit Test Case – Manage Cart

Test Case ID	WBCS-005	Module Name	Cart Module	
Test Title	Unit Test Case for User Manage Cart			
Pre-condition	-			
Test Case Description	Execution Steps	Expected Result	Actual Result	Status
Add item to cart	<ol style="list-style-type: none"> 1. Clicks on the Add To Cart button. 	Item Added to relevant cart.	Item Added to relevant cart.	Pass
Add existing cart item to cart	<ol style="list-style-type: none"> 1. Clicks on the Add To Cart button for an existing cart item 	Item quantity increased	Item quantity increased	Pass
Add item to beverage cart with missing size	<ol style="list-style-type: none"> 1. Clicks on the Add To Cart button for a beverage product with no size selection. 	The system displays alert message of required field	The system displays alert message of required field	Pass
Perform login with cart items in guest carts	<ol style="list-style-type: none"> 1. Clicks on the Add To Cart button for any product. 	Cart items will be transferred to database	Cart items will be transferred to database	Pass

	2. Clicks the Login button.	under the authenticated user	under the authenticated userPass	
Increase item quantity in carts	1. Clicks on the “+” icon in the shopping cart page	The system increases the item quantity and relevant prices	The system increases the item quantity and relevant prices	Pass
Decrease item quantity in carts	1. Clicks on the “-” icon in the shopping cart page	The system decreases the item quantity and relevant prices	The system decreases the item quantity and relevant prices	Pass
Delete item from carts	1. Clicks on the “x” icon in the shopping cart page	Item is being removed from the cart	Item is being removed from the cart	Pass

Table 7.1.6: Unit Test Case – Write a Review

Test Case ID	WBCS-006	Module Name	Customer Module	
Test Title	Unit Test Case for User Write a Review			
Pre-condition	User has logged into his/her account			
Test Case Description	Execution Steps	Expected Result	Actual Result	Status

Valid ratings and feedback inputs	1. Select valid rating. 2. Inputs valid feedback. 3. Clicks on the Submit button.	The system displays the message of review successfully submitted	The system displays the message of review successfully submitted	Pass
Empty feedback inputs	1. Select valid rating. 2. Inputs invalid feedback. 3. Clicks on the Submit button.	The system displays alert message of required field	The system displays alert message of required field	Pass
No ratings	1. Inputs valid feedback. 2. Clicks on the Submit button.	The system displays alert message of required field	The system displays alert message of required field	Pass

Table 7.1.7: Unit Test Case – View Order List

Test Case ID	WBCS-007	Module Name	Customer Module	
Test Title	Unit Test Case for User view his/her own order list			
Pre-condition	User has logged into his/her account			
Test Case Description	Execution Steps	Expected Result	Actual Result	Status
View order list	1. Clicks on the View My Order button.	The system displays the user's order list	The system displays the user's order list	Pass

Test Case ID	WBCS-008	Module Name	Guiding Material Module		
Test Title	Unit Test Case for User view guiding materials				
Pre-condition	-				
Test Case Description	Execution Steps	Expected Result	Actual Result	Status	
View recipe guide list	<ol style="list-style-type: none"> 1. Clicks on the Guides button. 2. Clicks on the Recipes tab. 	The system displays the recipe guide list	The system displays the recipe guide list	Pass	
View brewing guide list	<ol style="list-style-type: none"> 1. Clicks on the Guides button. 2. Clicks on the Brewing tab. 	The system displays the brewing guide list	The system displays the brewing guide list	Pass	

Table 7.1.8: Unit Test Case – Interact with Chatbot

Test Case ID	WBCS-009	Module Name	Chatbot Module		
Test Title	Unit Test Case for User interact with the chatbot				
Pre-condition	-				
Test Case Description	Execution Steps	Expected Result	Actual Result	Status	
Input valid inquiries	<ol style="list-style-type: none"> 1. Inputs valid question. 2. Clicks on the arrow icon to send 	The systems displays the response from OpenAI	The systems displays the response from OpenAI	Pass	

Empty inputs	1. Clicks on the arrow icon to send	The system displays alert message of the invalid inputs.	The system displays alert message of the invalid inputs.	Pass

Table 7.1.9: Unit Test Case – View Payment Page

Test Case ID	WBCS-010	Module Name	Payment Module	
Test Title	Unit Test Case for User view payment page			
Pre-condition	-			
Test Case Description	Execution Steps	Expected Result	Actual Result	Status
View Payment Page	1. Clicks Proceed To Checkout button in the shopping cart page	The systems displays Stripe payment page	The systems displays Stripe payment page	Pass

Table 7.1.10: Unit Test Case – User Payment Process

Test Case ID	WBCS-011	Module Name	Payment Module	
Test Title	Unit Test Case for User payment process			

Pre-condition	User is redirected to Stripe payment page			
Test Case Description	Execution Steps	Expected Result	Actual Result	Status
Inputs valid payment information	<ol style="list-style-type: none"> Inputs valid payment information Clicks on the Pay button 	The systems redirects to payment success page	The systems redirects to payment success page	Pass
Inputs invalid payment information	<ol style="list-style-type: none"> Inputs invalid payment information Clicks on the Pay button 	The system displays error message of invalid payment information	The system displays error message of invalid payment information	Pass
Empty inputs	<ol style="list-style-type: none"> Clicks on the Pay button 	The system displays error message of empty inputs	The system displays error message of empty inputs	Pass
Cancel Payment	<ol style="list-style-type: none"> Clicks on the Back button 	The system redirects to payment failure page	The system redirects to payment failure page	Pass

Table 7.1.11: Unit Test Case – Manage Product

Test Case ID	AP-001	Module Name	Manage Product Module	
Test Title	Unit Test Case for Admin to manage product			
Pre-condition	Admin has logged into the account			
Test Case Description	Execution Steps	Expected Result	Actual Result	Status

View product list page	1. Select the Product list resource	The systems redirects to product list page	The systems redirects to product list page	Pass
Add new product with valid inputs	1. Inputs valid product information 2. Clicks on the Save button	The system displays notification of created	The system displays notification of created	Pass
Add new product with invalid or empty inputs	1. Inputs with invalid product information or empty inputs 2. Clicks on the Save button	The system displays error message of the inputs	The system displays error message of the inputs	Pass
Update a product with valid inputs	1. Inputs valid product information 2. Clicks on the Save Changes button	The system displays notification of updated	The system displays notification of updated	Pass
Update a product with invalid or empty inputs	3. Input with invalid product information or empty inputs 4. Clicks on the Save Changes button	The system displays error message of the inputs	The system displays error message of the inputs	Pass
Delete a product	5. Clicks on the Delete button	The system displays notification of deleted	The system displays notification of deleted	Pass

Table 7.1.12: Unit Test Case = Manage Data Embedding File

Test Case ID	AP-002	Module Name	Manage Data Embedding File Module
---------------------	--------	--------------------	-----------------------------------

Test Title	Unit Test Case for Admin to manage data embedding file			
Pre-condition	Admin has logged into the account			
Test Case Description	Execution Steps	Expected Result	Actual Result	Status
Add new file with no file existed	<ol style="list-style-type: none"> 1. Upload new file with valid file type 2. Clicks the Save button 	The systems displays notification of created	The systems displays notification of created	Pass
Upload invalid file type	<ol style="list-style-type: none"> 1. Upload new file with invalid file type 2. Clicks on the Save button 	The system displays error message of invalid file type	The system displays error message of invalid file type	Pass
Add new file with file existed	<ol style="list-style-type: none"> 1. Upload new file with valid file type 2. Clicks on the Save button 	The system displays error message to delete current file before adding new file	The system displays error message to delete current file before adding new file	Pass
Update on the existing file	<ol style="list-style-type: none"> 6. Remove existing file and upload a new file 7. Clicks on the Save Changes button 	The system displays notification of updated	The system displays notification of updated	Pass
Delete a file	<ol style="list-style-type: none"> 8. Clicks on the Delete button 	The system displays notification of deleted	The system displays notification of deleted	Pass

Table 7.1.13: Unit Test Case – Manage Guiding Material

Test Case ID	AP-003	Module Name	Manage Guiding Material Module		
Test Title	Unit Test Case for Admin to manage guiding material				
Pre-condition	Admin has logged into the account				
Test Case Description	Execution Steps	Expected Result	Actual Result	Status	
View guiding material list page	1. Select the guiding material list resource	The systems redirects to guiding material list page	The systems redirects to guiding material list page	Pass	
Add new guiding material with valid inputs	3. Inputs valid guiding material information 4. Clicks on the Save button	The system displays notification of created	The system displays notification of created	Pass	
Add new guiding material with invalid or empty inputs	3. Inputs with invalid guiding material information or empty inputs 4. Clicks on the Save button	The system displays error message of the inputs	The system displays error message of the inputs	Pass	
Update a guiding material with valid inputs	9. Inputs valid guiding material information 10. Clicks on the Save Changes button	The system displays notification of updated	The system displays notification of updated	Pass	

Update a guiding material with invalid or empty inputs	11. Input with invalid guiding material information or empty inputs 12. Clicks on the Save Changes button	The system displays error message of the inputs	The system displays error message of the inputs	Pass
Delete a guiding material	13. Clicks on the Delete button	The system displays notification of deleted	The system displays notification of deleted	Pass

Table 7.1.14: Unit Test Case – Admin Login

Test Case ID	AP-004	Module Name	Login Module	
Test Title	Unit Test Case for Admin Login Account			
Pre-condition	Admin has a registered account			
Test Case Description	Execution Steps	Expected Result	Actual Result	Status
Input registered email and correct password	4. Input a valid email. 5. Input a valid password. 6. Clicks the Login button.	The system redirects to the logged in home page.	The system redirects to the logged in home page.	Pass
Input a unregistered email and correct password	4. Input an invalid email. 5. Input valid password. 6. Clicks the Login button.	The system displays error message of unregistered email.	The system displays error message of unregistered email.	Pass

Input incorrect password	<ol style="list-style-type: none"> 4. Input a valid email. 5. Input a incorrect password. 6. Clicks the Login button. 	The system displays error message of incorrect password.	The system displays error message of incorrect password.	Pass
Empty inputs	<ol style="list-style-type: none"> 2. Clicks the Login button. 	The system displays alert message of invalid inputs.	The system displays alert message of invalid inputs.	Pass
Input invalid format of email	<ol style="list-style-type: none"> 4. Input invalid email format. 5. Input valid password. 6. Clicks the Login button 	The system displays error message of invalid email format.	The system displays error message of invalid email format.	Pass

7.2 Integration Testing

Integration testing is a crucial phase in the software testing process that evaluates the interactions and relationships between individual components or modules of a software system. It aims to uncover potential issues that may arise when these components are combined and function together as a unified whole. Thus, in this project, there will be several integration testing conducted as the tables below.

Table 7.2.1: Integration Test Case – Product and Image

Integration Test Case		Add and Update Product and Image		
ID	Execution Steps	Expected Result	Actual Result	Status
1	1. Clicks on the New Beverage button 2. Input information related and upload beverage image 3. Clicks on the Save button	The image will be stored in the /storage directory with a random name assigned by Filament	The image will be stored in the /storage directory with a random name assigned by Filament	Pass
2	1. Clicks on the New Homebrew Product button 2. Input information related and upload homebrew product image 3. Clicks on the Save button	The image will be stored in the /storage directory with a random name assigned by Filament	The image will be stored in the /storage directory with a random name assigned by Filament	Pass
3	1. Clicks on the New Mechanic button 2. Input information related and upload mechanic image 3. Clicks on the Save button	The image will be stored in the /storage directory with a random name assigned by Filament	The image will be stored in the /storage directory with a random name assigned by Filament	Pass

Table 7.2.2: Integration Test Case - Guiding Material and Image

Integration Test Case		Add and Update Guiding Material and Image		
ID	Execution Steps	Expected Result	Actual Result	Status
1	<ol style="list-style-type: none"> 1. Clicks on the New Recipe Guide button 2. Input information related and upload recipe guide image 3. Clicks on the Save button 	The image will be stored in the /storage directory with a random name assigned by Filament	The image will be stored in the /storage directory with a random name assigned by Filament	Pass
2	<ol style="list-style-type: none"> 1. Clicks on the New Brewing Guide button 2. Input information related and upload brewing guide image 3. Clicks on the Save button 	The image will be stored in the /storage directory with a random name assigned by Filament	The image will be stored in the /storage directory with a random name assigned by Filament	Pass

Table 7.2.3: Integration Test Case – Data Embedding File and OpenAI

Integration Test Case		Add and Update Data Embedding File and Interaction with OpenAI		
ID	Execution Steps	Expected Result	Actual Result	Status
1	<ol style="list-style-type: none"> 1. Clicks on the New Data Embedding File button 2. Upload latest file 3. Clicks on the Save button 	Data extracted and sent for OpenAI interaction with response and storage on the data vector	Data extracted and sent for OpenAI interaction with response and storage on the data vector	Pass
2	<ol style="list-style-type: none"> 1. Clicks on the Edit button 2. Remove current file 3. Upload latest file 4. Clicks on the Save button 	Data extracted and sent for OpenAI interaction with response and storage on the data vector	Data extracted and sent for OpenAI interaction with response and storage on the data vector	Pass

Table 7.2.4: Integration Test Case – Payment Process and Create Order

Integration Test Case		Stripe payment page and Create Order		
ID	Execution Steps	Expected Result	Actual Result	Status
1	1. Clicks on the Proceed to Checkout Button	Redirects to Stripe payment page and the order will be created	Redirects to Stripe payment page and the order will be created	Pass

Table 7.2.5: Integration Test Case – Payment Process and Delete Order

Integration Test Case		Stripe payment page and Delete Order		
ID	Execution Steps	Expected Result	Actual Result	Status
1	1. Clicks on the Back Button	Redirects to payment failure and the order will be deleted	Redirects to payment failure and the order will be deleted	Pass

Table 7.2.6: Integration Test Case – Payment Process and Update Order

Integration Test Case		Stripe payment page and Update Order		
ID	Execution Steps	Expected Result	Actual Result	Status
1	1. Input valid payment information 2. Clicks on the Pay button	Redirects to payment success and the order status will be updated to paid	Redirects to payment success and the order status will be updated to paid	Pass

Table 7.2.7: Integration Test Case – Create Order and Create Order Detail

Integration Test Case		Create Order and Create Order Detail		
ID	Execution Steps	Expected Result	Actual Result	Status
1	1. Clicks on the Proceed To Checkout button	Order and order detail created	Order and order detail created	Pass

Table 7.2.8: Payment Process and Delete Cart Items

Integration Test Case		Payment Success Page and Delete Cart Items		
ID	Execution Steps	Expected Result	Actual Result	Status
1	1. Clicks on the Pay button	Redirects to the payment success page and cart items will be deleted	Redirects to the payment success page and cart items will be deleted	Pass

Table 7.2.9: Login and Transfer Cart Listener

Integration Test Case		Login and Transfer Cart Listener		
ID	Execution Steps	Expected Result	Actual Result	Status
1	1. Clicks on the Login button	Redirects to home page with guest cart items added	Redirects to home page with guest cart items added	Pass

7.3 User Acceptance Testing (UAT)

A total of 8 users have actively participated in the User Acceptance Testing (UAT) phase of the project. These participants are divided into two groups: 4 users are actively engaged in the Customer UAT, interacting with the web-based coffee shop, while the remaining 4 are actively involved in the Admin UAT focusing on the admin panel. Below are the templates for documenting the UAT results for both customer and admin testing. The UAT sessions have been conducted remotely using the Zoom platform. Detailed UAT results are available in the appendices.

Table 7.3.1: User Acceptance Testing Template for Customer

Web-based Coffee Shop with Chatbot Integrated				
Test Module	Test Case ID	Test Scenario	Status	Comment
Register	CUAT-001	1. Able to register a new account		

Login	CUAT-002	1. Able to login to the newly registered account		
Product	CUAT-003	1. Able to view beverage list page		
	CUAT-004	2. Able to view homebrew product list page		
	CUAT-005	3. Able to view mechanic list page		
	CUAT-006	4. Able to apply search filter on product list		
	CUAT-007	5. Able to remove search filter on product list		
	CUAT-008	6. Able to apply price range filter on product list		
	CUAT-009	7. Able to remove price range filter on product list		
	CUAT-010	8. Able to apply product category filter on the product list		
	CUAT-011	9. Able to remove product category filter on product list		
	CUAT-012	10. Able to view product detail page		
	CUAT-013	11. Able to view rating and reviews of the product		
	CUAT-014	12. Able to view the product recommendations		
	CUAT-015	13. Able to select beverage customization options		
Cart	CUAT-016	1. Able to view product cart page		

	CUAT-017	2. Able to view beverage cart page		
	CUAT-018	3. Able to add an item into cart		
	CUAT-019	4. Able to increase the item quantity in cart		
	CUAT-020	5. Able to decrease the item quantity in cart		
	CUAT-021	6. Able to delete an item from cart		
	CUAT-022	7. Able to view guest beverage cart items after login		
	CUAT-023	8. Able view guest product cart items after login		
Guiding Materials	CUAT-024	1. Able to view recipe guiding list page		
	CUAT-025	2. Able to view brewing guide list page		
	CUAT-026	3. Able to view recipe guide detail page		
	CUAT-027	4. Able to view brewing guide detail page		
Chatbot	CUAT-028	1. Able to view chatbot pop up window		
	CUAT-029	2. Able to initiate chat with the chatbot		
	CUAT-030	3. Able to get reply from chatbot		
Customer	CUAT-031	1. Able to view own order list		

	CUAT-032	2. Able to write review on a product		
Payment	CUAT-034	1. Able to view payment page		
	CUAT-035	2. Able to make payment		
	CUAT-036	3. Able to select payment method		
	CUAT-036	4. Able to view payment successful page		
	CUAT-037	5. Able to cancel payment		
	CUAT-038	6. Able to view payment failure page		

Table 7.3.2: User Acceptance Testing Template for Admin

MYCoffee Admin Panel				
Test Module	Test Case ID	Test Scenario	Status	Comment
Login	AUAT-001	1. Able to login to the admin account		
Manage Product	AUAT-002	1. Able to view product list		
	AUAT-003	2. Able to view product category list		
	AUAT-004	3. Able to add new product information		
	AUAT-005	4. Able to add new product category		
	AUAT-006	5. Able to update on a product information		

	AUAT-007	6. Able to update on a product category		
	AUAT-008	7. Able to delete a product		
	AUAT-009	8. Able to delete product category		
Manage guiding materials	AUAT-010	1. Able to view recipe guide list		
	AUAT-011	2. Able to view brewing guide list		
	AUAT-012	3. Able to add new recipe guide		
	AUAT-013	4. Able to add new brewing guide		
	AUAT-014	5. Able to update on a recipe guide		
	AUAT-015	6. Able to update on a brewing guide		
	AUAT-015	7. Able to delete a recipe guide		
	AUAT-016	8. Able to delete a brewing guide		
Manage Data Embedding File	AUAT-017	1. Able to add new data embedding file		
	AUAT-018	2. Able to update a data embedding file		
	AUAT-019	3. Able to delete a data embedding file		
View Order	AUAT-020	1. Able to view complete order list		

7.4 System Usability Testing

System usability testing will be conducted concurrently with the UAT, with both assessments involving the same user group. These tests will take place on the Zoom platform, providing a practical remote testing environment. The primary goal of this combined approach is to comprehensively evaluate the system's usability, encompassing not only functional requirements but also aspects related to user experience and interface design. The results from this evaluation will directly influence customer acceptance and the final product approval. To measure system usability, the selected evaluation metric is the System Usability Scale (SUS), which is renowned for its effectiveness in quantifying user satisfaction and usability. Below, you can see the standard SUS template, which will be used to gather feedback from the user group, guiding decisions to enhance the product's overall user experience.

System Usability Scale

© Digital Equipment Corporation, 1986.

	Strongly disagree	Strongly agree
1. I think that I would like to use this system frequently	1	5
2. I found the system unnecessarily complex	1	5
3. I thought the system was easy to use	1	5
4. I think that I would need the support of a technical person to be able to use this system	1	5
5. I found the various functions in this system were well integrated	1	5
6. I thought there was too much inconsistency in this system	1	5
7. I would imagine that most people would learn to use this system very quickly	1	5
8. I found the system very cumbersome to use	1	5
9. I felt very confident using the system	1	5
10. I needed to learn a lot of things before I could get going with this system	1	5

Figure 7.4.1: SUS Standard Template (Brooke, 1995)

Going further into the intricacies of the template depicted above, we observe that each question therein is assigned a rating scale ranging from 1 to 5. To compute the SUS score effectively, it necessitates the segmentation of questions into two distinct groups: one comprising odd-numbered questions and the other encompassing even-numbered questions. The calculation method diverges for these two segments.

For the odd-numbered questions, the computation entails subtracting 1 from the assigned ratings. Conversely, for the even-numbered questions, it involves subtracting the ratings from 5. Once these individual scores have been determined, they are aggregated to obtain the total SUS score.

To ascertain the final SUS score, the total score is then multiplied by 2.5. The resulting SUS score falls within a range spanning from 0 to 100. This scoring system not only allows for a nuanced evaluation of system usability but also provides a standardized measure that aids in comprehending user satisfaction and perception which shown in the figure below.

SUS Score	Grade	Adjective Rating
>80.3	A	Excellent
68-80.2	B	Good
67	C	Okay
51-66	D	Poor
<51	F	Awful

Figure 7.4.2: SUS Grading Table (Sasmito, Zulfiqar and Nishom, 2019)

The final SUS score will be utilized to apply the rating system, as depicted in the figure above, in order to assess the quality and usability of the website. The following tables present the SUS scores, corresponding grades, and adjective ratings for both the website and the admin panel. The SUS survey form can be refer in the appendice.

Table 7.4.1: SUS Score for Web-based Coffee Shop

Participants	Scores for each Question										Total	Percentage
	1	2	3	4	5	6	7	8	9	10		

1	5	1	5	1	5	1	5	1	5	1	40	100%
2	5	1	5	1	4	1	5	1	5	1	38	95%
3	5	1	5	1	5	1	5	5	5	1	36	90%
4	5	1	5	1	5	1	5	1	5	1	40	100%
5	5	1	5	1	5	1	5	1	5	1	40	100%
Average SUS Score											97%	
Grade	A		Adjective Rating					Excellent				

Table 7.4.2: SUS Score for Admin Panel

Participants	Scores for each Question										Total	Percentage
	1	2	3	4	5	6	7	8	9	10		
1	3	1	4	2	4	4	5	2	3	1	29	72.5%
2	5	1	5	1	5	1	5	1	5	1	40	100%
3	4	1	5	1	5	2	5	1	5	1	38	95%
4	5	1	5	1	5	1	5	1	5	1	40	100%
5	4	2	4	3	4	2	4	2	5	2	30	75%
Average SUS Score											88.5%	
Grade	A		Adjective Rating					Excellent				

Furthermore, to ensure clarity and precision in the usability testing process, comprehensive test scenarios will be meticulously crafted for both customers and administrators. These test scenarios will serve as structured guidelines to evaluate the website's performance, functionality, and user-friendliness. Each scenario will include all the relevant information required to simulate real-world user interactions and experiences

Table 7.4.3: Test Scenario for Customer

ID	Test Scenario	Scenario Description
1	Register	You are a guest and you wished to create an account for the website. How would you create an account in the website?

2	Login	After performing scenario 1, you wish to login to your own account. How would you do to login to your account?
3	Search Filter, Price Range Filter, Category Filter	Upon browsing the product list, you find that there are too many products, you wish to performing filtering. How would you perform the filtering?
	Remove filter	After applying the filter, you realised that you would want to look back on the full product list for more options How would you remove the filterings applied?
	View Product Detail, Add to cart	When browsing the product detail, you would like to add the product into the cart. How would you do to add into cart?
	Make Payment	You decided to purchase the item in the cart. How would you do to make payment? <u>Card Information:</u> Card Number: 4242 4242 4242 4242 Expiry Date: 12/23 CVC: 123
	Write Review	You wish to write a review on a product. How would you do to write review and rate on a product?
	Ask Chatbot	You wish to have a clearer information before making the decision to purchase a product. How would you ask the chatbot?

Table 7.4.4: Test Scenario for Admin

ID	Test Scenario	Scenario Description
1	Login	You are an admin to the website, you would like to login to your account. How would you do?

		Email: admin@admin.com Password:password
2	Add New Beverage Category	You would like to add a new beverage category. How would you do?
3	Add New Beverage	You would like to add new beverage product. How would you do?
	Add New Brewing Guide	You would like to add new brewing guide. How would you do?
	Add Data Embedding File	You would like to add a new data embedding file to update on the chatbot context. How would you do?

CHAPTER 8

CONCLUSION AND FUTURE WORKS

8.1 Project Objectives Milestone

Despite facing a tight schedule, this project has successfully accomplished its three primary objectives.

The first objective involved identifying the essential features and functions for both the website and the chatbot. This was achieved through an extensive process that included comprehensive literature reviews, the use of questionnaires, and other research methodologies.

The second objective aimed to develop a comprehensive product lineup for the web-based coffee shop. This objective was also met, with the coffee shop offering a diverse range of products, including beverages, homebrew items, and machinery products. Additionally, the website provides valuable guiding materials such as recipes and brewing guides to enhance the user experience.

The third objective centered on implementing a chatbot into the website, serving dual roles as a customer support tool and a personalized product recommendation system. This objective, too, has been successfully achieved, as the chatbot effectively fulfills both functions, providing valuable support and tailored product recommendations to customers.

8.2 Limitations and Recommendations

Despite successfully achieving the project's objectives, it's important to acknowledge that there are several limitations to this endeavor. The table below highlights these limitations alongside the recommendations.

Table 8.2.1: Limitations and Recommendations Table

Limitations	Recommendations
No delivery system integrated	Integrating a comprehensive delivery system, whether developed in-house or sourced from a third-party provider, is a pivotal recommendation with far-reaching benefits. This integration

	<p>facilitates real-time order tracking for customers, fostering transparency and satisfaction, while simultaneously empowering administrators with efficient order management tools. The result is a reduction in customer inquiries, an elevated customer experience, and the ability to make data-driven decisions to optimize delivery processes. Moreover, it ensures scalability for future growth and provides flexibility in tailoring the integration to specific project needs, ultimately enhancing competitiveness and customer satisfaction.</p>
<p>No stocking management</p>	<p>The absence of a stocking management system within this project poses potential challenges, particularly regarding order generation. Without such a system, administrators may remain unaware of depleting stock levels, potentially leading to scenarios where customers can attempt to purchase items that are no longer available. To mitigate this issue, the implementation of a robust stocking management system is recommended. Such a system would offer clear, real-time information to administrators, enabling them to promptly hide sold-out products or update stock availability. This proactive approach not only enhances operational efficiency but also ensures a smoother, more satisfying shopping experience for customers.</p>
<p>No real time update data embedding</p>	<p>Within this project, a notable limitation is the absence of real-time updates for data embedding. Currently, administrators are required to manually update the data embedding whenever there are changes in the FAQ or product information. This manual process introduces complexity and the potential for human error, a common occurrence in</p>

	<p>such tasks. Implementing a real-time, on-the-spot update mechanism for data embedding would address this issue effectively. This enhancement would ensure that the chatbot consistently serves its intended purpose, ultimately leading to an improved user experience.</p>
--	--

REFERENCES

- Balaji, S. and Murugaiyan, M.S., 2012. Waterfall vs. V-Model vs. Agile: A comparative study on SDLC. *International Journal of Information Technology and Business Management*, 2(1), pp.26–30.
- Berger, H., Beynon-Davies, P. and Cleary, P., 2004. The utility of a rapid application development (RAD) approach for a large complex information Systems development. *ECIS 2004 Proceedings*, p.7.
- Beynon-Davies, P. and Holmes, S., 2002. Design breakdowns, scenarios and rapid application development. *Information and Software Technology*, [online] 44(10), pp.579–592. [https://doi.org/10.1016/S0950-5849\(02\)00078-2](https://doi.org/10.1016/S0950-5849(02)00078-2).
- Bhalla, A., Garg, S. and Singh, P., 2020. Present day web-development using reactjs. *International Research Journal of Engineering and Technology*, 7(05).
- Brooke, J., 1995. SUS: A quick and dirty usability scale. *Usability Eval. Ind.*, 189.
- Burns, R.N. and Dennis, A.R., 1985. Selecting the appropriate application development methodology. *ACM SIGMIS Database: the DATABASE for Advances in Information Systems*, 17(1), pp.19–23.
- Chandra, V., 2015. Comparison between various software development methodologies. *International Journal of Computer Applications*, 131(9), pp.7–10.
- Chitra, L.P. and Satapathy, R., 2017. Performance comparison and evaluation of Node.js and traditional web server (IIS). In: *2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET)*. pp.1–4. <https://doi.org/10.1109/ICAMMAET.2017.8186633>.
- Chowdhury, A.F. and Huda, M.N., 2011. Comparison between Adaptive Software Development and Feature Driven Development. In: *Proceedings of 2011 International Conference on Computer Science and Network Technology*. pp.363–367. <https://doi.org/10.1109/ICCSNT.2011.6181977>.
- Dalbard, A. and Isacson, J., 2021. *Comparative study on performance between ASP. NET and Node. js Express for web-based calculation tools*.

- Daud, N.M.N., Bakar, N.A.A.A. and Rusli, H.M., 2010. Implementing rapid application development (RAD) methodology in developing practical training application system. In: *2010 International Symposium on Information Technology*. pp.1664–1667. <https://doi.org/10.1109/ITSIM.2010.5561634>.
- Deng, J. and Lin, Y., 2022. The benefits and challenges of ChatGPT: An overview. *Frontiers in Computing and Intelligent Systems*, 2(2), pp.81–83.
- Despa, M.L., 2014. Comparative study on software development methodologies. *Database Systems Journal*, 5(3), pp.37–56.
- Dirks-Naylor, A.J., 2015. The benefits of coffee on skeletal muscle. *Life Sciences*, [online] 143, pp.182–186. <https://doi.org/https://doi.org/10.1016/j.lfs.2015.11.005>.
- Fern, D.A. and Donaldson, S.E., 1989. Tri-Cycle: a prototype methodology for advanced software development. In: *Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences. Volume II: Software Track*. IEEE Computer Society. pp.377–378.
- Gaikwad, S.S. and Adkar, P., 2019. A review paper on bootstrap framework. *IRE Journals*, 2(10), pp.349–351.
- Gall, D. and Richard, B., 2005. Dynamic Systems Development Method and Rapid Application Development.
- Geambașu, C.V., Jianu, I., Jianu, I. and Gavrilă, A., 2011. Influence factors for the choice of a software development methodology. *Accounting and Management Information Systems*, 10(4), pp.479–494.
- GeeksForGeeks, 2021. *Software Prototyping Model and Phases*. [online] Available at: <<https://www.geeksforgeeks.org/software-prototyping-model-and-phases/>> [Accessed 28 March 2023].
- Ighodaro, N., 2018. *How Laravel implements MVC and how to use it effectively*. [online] Available at: <<https://pusher.com/blog/laravel-mvc-use/#0-introduction>> [Accessed 21 March 2023].
- Indeed Editorial Team, 2022. *What Is a Software Development Methodology? (With 11 Types)*. [online] Available at: <<https://au.indeed.com/career-advice/career-development/software-development-methodology#:~:text=A%20software%20development%20methodology%20is>>

%20a%20process%20by,that%20guide%20professionals%20through%20each%20stage%20of%20development.> [Accessed 26 February 2023].

ISMAIL, U., QADRI, S. and FAHAD, M., 2015. Requirement Elicitation for Open Source Software By using SCRUM and Feature Driven Development. *International Journal of Natural & Engineering Sciences*, 9(1).

KnowledgeHut, 2022. *What are the Pros and Cons of React*. [online] Available at: <<https://www.knowledgehut.com/blog/web-development/pros-and-cons-of-react>> [Accessed 27 March 2023].

Lee, N.F.C., Qi, C.J., Mansur, K. and Mahmud, R., 2020. The impact of coronavirus (COVID-19) on e-business in Malaysia: a review. In: *The 1 st International Conference on Entrepreneurship and Small Business (ICES2020)*. p.212.

Mitchell, B., 2022. *9 Top Software Development Methodologies: Pros and Cons*. [online] Available at: <<https://www.codingdojo.com/blog/software-development-methodologies>> [Accessed 26 February 2023].

Molinillo, S., Anaya-Sánchez, R. and Liébana-Cabanillas, F., 2020. Analyzing the effect of social support and community factors on customer engagement and its impact on loyalty behaviors toward social commerce websites. *Computers in Human Behavior*, [online] 108, p.105980. <https://doi.org/https://doi.org/10.1016/j.chb.2019.04.004>.

Murdiana, R. and Hajaoui, Z., 2020. E-Commerce marketing strategies in industry 4.0. *International Journal of Business Ecosystem & Strategy* (2687-2293), 2(1), pp.32–43.

Nikolaieva, A., 2023. *8 Best Software Development Methodologies*. [online] Available at: <<https://www.uptech.team/blog/software-development-methodologies>> [Accessed 28 February 2023].

Petruzzello, M. and Bondarenko, P., 2023. *Starbucks*. [online] Available at: <<https://www.britannica.com/topic/Starbucks>> [Accessed 24 February 2023].

Radchenko, V. and Pyatkin, D., 2019. Analysis of the Most Common Php Frameworks.

Rault, J., 2022. *Top 7 Software Development Methodologies Every Developer Needs to Know*. [online] Available at: <<https://www.laneways.agency/top-7-software-development-methodologies/>> [Accessed 26 February 2023].

- Rawat, P. and Mahajan, A.N., 2020. ReactJS: A modern web development framework. *International Journal of Innovative Science and Research Technology*, 5(11), pp.698–702.
- Rosyad, R., Syukur, A., Busro and Rahim, R., 2019. Multimedia Prayer Application for Education with Rapid Application Development Method. In: *2019 7th International Conference on Cyber and IT Service Management (CITSM)*. pp.1–4. <https://doi.org/10.1109/CITSM47753.2019.8965379>.
- Rychlý, M. and Tichá, P., 2008. A tool for supporting feature-driven development. In: *Balancing Agility and Formalism in Software Engineering: Second IFIP TC 2 Central and East European Conference on Software Engineering Techniques, CEE-SET 2007, Poznan, Poland, October 10-12, 2007, Revised Selected Papers*. Springer. pp.196–207.
- Sabale, R.G. and Dani, A.R., 2012. Comparative study of prototype model for software engineering with system development life cycle. *IOSR Journal of Engineering*, 2(7), pp.21–24.
- Saks, E., 2019. JavaScript Frameworks: Angular vs React vs Vue.
- Salas-Zárata, M. del P., Alor-Hernández, G., Valencia-García, R., Rodríguez-Mazahua, L., Rodríguez-González, A. and López Cuadrado, J.L., 2015. Analyzing best practices on Web development frameworks: The lift approach. *Science of Computer Programming*, [online] 102, pp.1–19. <https://doi.org/https://doi.org/10.1016/j.scico.2014.12.004>.
- Salehi, F., Abdollahbeigi, B., Langroudi, A.C. and Salehi, F., 2012. The Impact of Website Information Convenience on E-commerce Success of Companies. *Procedia - Social and Behavioral Sciences*, [online] 57, pp.381–387. <https://doi.org/https://doi.org/10.1016/j.sbspro.2012.09.1201>.
- Santos, V., Sampaio, M. and Alliprandini, D.H., 2020. The impact of product variety on fill rate, inventory and sales performance in the consumer goods industry. *Journal of Manufacturing Technology Management*, 31(7), pp.1481–1505.
- Sasmito, G.W., Zulfiqar, L.O.M. and Nishom, M., 2019. Usability Testing based on System Usability Scale and Net Promoter Score. In: *2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*. pp.540–545. <https://doi.org/10.1109/ISRITI48646.2019.9034666>.

- Senecal, S. and Nantel, J., 2004. The influence of online product recommendations on consumers' online choices. *Journal of Retailing*, [online] 80(2), pp.159–169. <https://doi.org/https://doi.org/10.1016/j.jretai.2004.04.001>.
- Sinha, A. and Das, P., 2021. Agile Methodology Vs. Traditional Waterfall SDLC: A case study on Quality Assurance process in Software Industry. In: *2021 5th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech)*. pp.1–4. <https://doi.org/10.1109/IEMENTech53263.2021.9614779>.
- Stanke, B., 2022. *Feature-Driven Development: The Pros, Cons, and How It Compares to Scrum*. [online] Available at: <<https://www.bobstanke.com/blog/feature-driven-development>> [Accessed 7 March 2023].
- Tie, J., Jin, J. and Wang, X., 2011. Study on application model of three-tiered architecture. In: *2011 Second International Conference on Mechanic Automation and Control Engineering*. pp.7715–7718. <https://doi.org/10.1109/MACE.2011.5988838>.
- Tilkov, S. and Vinoski, S., 2010. Node.js: Using JavaScript to Build High-Performance Network Programs. *IEEE Internet Computing*, 14(6), pp.80–83. <https://doi.org/10.1109/MIC.2010.145>.
- Tirumala, S.S., Ali, S. and Babu, A., 2016. A hybrid agile model using SCRUM and feature driven development. *International Journal of Computer Applications*, 156(5), pp.1–5.
- Tondon, S., 2022. *The Pros and Cons of Node.js Web App Development: A Detailed Look*. [online] Available at: <<https://javascript.plainenglish.io/the-pros-and-cons-of-node-js-web-app-development-a-detailed-look-c91a22f013c>> [Accessed 28 March 2023].
- Tram, N., 2022. *All Advantages and Disadvantages of Prototype Model*. [online] Available at: <<https://biplus.com.vn/advantages-and-disadvantages-of-prototype-model/>> [Accessed 28 February 2023].
- Vaswani, V., 2009. *MySQL database usage & administration*. McGraw Hill Professional.
- Yadav, N., Rajpoot, D.S. and Dhakad, S.K., 2019. LARAVEL: A PHP Framework for E-Commerce Website. In: *2019 Fifth International Conference*

on Image Information Processing (ICIIP). pp.503–508.
<https://doi.org/10.1109/ICIIP47207.2019.8985771>.

APPENDICES

Appendix A: Questionnaire

4/21/23, 6:50 AM FYP - Web-Based Coffee Shop Survey

FYP - Web-Based Coffee Shop Survey

DISCLAIMER

This questionnaire is intended as a survey of my final year project only. Your responses and credentials will be kept confidential and will only be used to improve my understanding of the topic. Please note that there is no right or wrong answer, should you have any concerns, please contact me, Wen Huey at 016-3309827 or email me at cwh0430@1utar.my. Thank you for your participation.

* Indicates required question

1. Age *

Mark only one oval.

18-20

21-30

31-40

2. Gender *

Mark only one oval.

Male

Female

3. Occupation *

Feature Section

<https://docs.google.com/forms/d/1GU8E5izbLGDT6ZpWFYH8w3CkH7sjnk0HUJGKlx220C4/edit#settings> 1/4

4/21/23, 6:50 AM FYP - Web-Based Coffee Shop Survey

4. When selecting a coffee product, what information would you like the website to provide? *

Check all that apply.

Details about the origin and roasting process

Product Recommendation

Customer Support

Price and availability

Customer reviews and ratings

Product searching

Product Filtering

Other: _____

5. Do you think a customizable coffee drink option would enhance your user experience? *

Mark only one oval.

Yes

No

6. If yes, what kind of customizations would you prefer? *

Check all that apply.

Milk type (e.g. almond, soy, oat)

Sweetener type and amount (e.g. sugar, honey, stevia)

Flavor shots (e.g. vanilla, caramel, hazelnut)

Espresso shot quantity

Other: _____

<https://docs.google.com/forms/d/1GU8E5izbLGDt6ZpWfYH8w3CkH7sjnk0HUJGKlx220C4/edit#settings> 2/4

4/21/23, 6:50 AM

FYP - Web-Based Coffee Shop Survey

7. Do you find it difficult to determine if a coffee product (e.g. coffee beans) is suitable *
for you?

Mark only one oval.

Yes

No

8. Would having customer support to assist you when selecting a product be helpful? *

Mark only one oval.

Yes

No

9. What kind of customer support would you like to have to assist you in selecting the *
product?

Check all that apply.

Product recommendations

Preferences

Usability (main purpose of usage)

Other: _____

10. What kind of coffee guiding and educational materials would you like the website *
to provide?

Check all that apply.

Brewing guides and tutorials

Information about different brewing methods (e.g. French press, pour-over)

Coffee bean roasting and grinding information

Educational materials about the origin of coffee beans

Health benefits of coffee

Other: _____

4/21/23, 6:50 AM

FYP - Web-Based Coffee Shop Survey

11. Please rate the following features in order of importance, with 1 being the least important and 5 being the most important: *

Mark only one oval per row.

	1	2	3	4	5
Product Filtering	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Product Searching	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Shopping Cart	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Product Information	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Product Review and Rating	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Coffee Guiding and Educational Materials	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Customization of Beverages	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Product Recommendation	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Customer Support	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

This content is neither created nor endorsed by Google.

Google Forms

Appendix B: UAT Results

Participant 1:

Name: Erin Kong

Test Starting Time: 9.15pm

Test Ending Time: 9.40pm

Web-based Coffee Shop with Chatbot Integrated				
Test Module	Test Case ID	Test Scenario	Status	Comment
Register	CUAT-001	1. Able to register a new account	Pass	-
Login	CUAT-002	1. Able to login to the newly registered account	Pass	-
Product	CUAT-003	1. Able to view beverage list page	Pass	Home Page Products Should Make Clickable and Redirectable
	CUAT-004	2. Able to view homebrew product list page	Pass	
	CUAT-005	3. Able to view mechanic list page	Pass	
	CUAT-006	4. Able to apply search filter on product list	Pass	-
	CUAT-007	5. Able to remove search filter on product list	Pass	-
	CUAT-008	6. Able to apply price range filter on product list	Pass	-
	CUAT-009	7. Able to remove price range filter on product list	Pass	-
	CUAT-010	8. Able to apply product category filter on the product list	Pass	-
	CUAT-011	9. Able to remove product category filter on product list	Pass	-

	CUAT-012	10. Able to view product detail page	Pass	-
	CUAT-013	11. Able to view rating and reviews of the product	Pass	-
	CUAT-014	12. Able to view the product recommendations	Pass	-
	CUAT-015	13. Able to select beverage customization options	Pass	-
Cart	CUAT-016	1. Able to view product cart page	Pass	Beverage and Product cart can combine together with different delivery time notification
	CUAT-017	2. Able to view beverage cart page	Pass	
	CUAT-018	3. Able to add an item into cart	Pass	
	CUAT-019	4. Able to increase the item quantity in cart	Pass	-
	CUAT-020	5. Able to decrease the item quantity in cart	Pass	-
	CUAT-021	6. Able to delete an item from cart	Pass	-
	CUAT-022	7. Able to view guest beverage cart items after login	Pass	-
	CUAT-023	8. Able view guest product cart items after login	Pass	-
Guiding Materials	CUAT-024	1. Able to view recipe guiding list page	Pass	-
	CUAT-025	2. Able to view brewing guide list page	Pass	-
	CUAT-026	3. Able to view recipe guide detail page	Pass	-

	CUAT-027	4. Able to view brewing guide detail page	Pass	-
Chatbot	CUAT-028	1. Able to view chatbot pop up window	Pass	-
	CUAT-029	2. Able to initiate chat with the chatbot	Pass	-
	CUAT-030	3. Able to get reply from chatbot	Pass	-
Customer	CUAT-031	1. Able to view own order list	Pass	-
	CUAT-032	2. Able to write review on a product	Pass	-
Payment	CUAT-034	1. Able to view payment page	Pass	-
	CUAT-035	2. Able to make payment	Pass	-
	CUAT-036	3. Able to select payment method	Pass	-
	CUAT-036	4. Able to view payment successful page	Pass	-
	CUAT-037	5. Able to cancel payment	Pass	-
	CUAT-038	6. Able to view payment failure page	Pass	-

Participant 2:

Name: Tan Jiun Wei

Test Starting Time: 8.28pm

Test Ending Time: 9.05pm

Web-based Coffee Shop with Chatbot Integrated				
Test Module	Test Case ID	Test Scenario	Status	Comment
Register	CUAT-001	1. Able to register a new account	Pass	-
Login	CUAT-002	1. Able to login to the newly registered account	Pass	-
Product	CUAT-003	1. Able to view beverage list page	Pass	Home Page Products Should Make Clickable and Redirectable
	CUAT-004	2. Able to view homebrew product list page	Pass	
	CUAT-005	3. Able to view mechanic list page	Pass	
	CUAT-006	4. Able to apply search filter on product list	Pass	Keywords should preserve in input text if search filter is applied
	CUAT-007	5. Able to remove search filter on product list	Pass	
	CUAT-008	6. Able to apply price range filter on product list	Pass	Price range filter add on input type
	CUAT-009	7. Able to remove price range filter on product list	Pass	
	CUAT-010	8. Able to apply product category filter on the product list	Pass	-
	CUAT-011	9. Able to remove product category filter on product list	Pass	-
	CUAT-012	10. Able to view product detail page	Pass	-

	CUAT-013	11. Able to view rating and reviews of the product	Pass	-
	CUAT-014	12. Able to view the product recommendations	Pass	-
	CUAT-015	13. Able to select beverage customization options	Pass	-
Cart	CUAT-016	1. Able to view product cart page	Pass	-
	CUAT-017	2. Able to view beverage cart page	Pass	-
	CUAT-018	3. Able to add an item into cart	Pass	-
	CUAT-019	4. Able to increase the item quantity in cart	Pass	-
	CUAT-020	5. Able to decrease the item quantity in cart	Pass	-
	CUAT-021	6. Able to delete an item from cart	Pass	-
	CUAT-022	7. Able to view guest beverage cart items after login	Pass	-
	CUAT-023	8. Able view guest product cart items after login	Pass	-
Guiding Materials	CUAT-024	1. Able to view recipe guiding list page	Pass	-
	CUAT-025	2. Able to view brewing guide list page	Pass	-
	CUAT-026	3. Able to view recipe guide detail page	Pass	-
	CUAT-027	4. Able to view brewing guide detail page	Pass	-

Chatbot	CUAT-028	1. Able to view chatbot pop up window	Pass	Words are too small and packed
	CUAT-029	2. Able to initiate chat with the chatbot	Pass	
	CUAT-030	3. Able to get reply from chatbot	Pass	
Customer	CUAT-031	1. Able to view own order list	Pass	Change the name to View Purchase History
	CUAT-032	2. Able to write review on a product	Pass	-
Payment	CUAT-034	1. Able to view payment page	Pass	-
	CUAT-035	2. Able to make payment	Pass	-
	CUAT-036	3. Able to select payment method	Pass	-
	CUAT-036	4. Able to view payment successful page	Pass	-
	CUAT-037	5. Able to cancel payment	Pass	-
	CUAT-038	6. Able to view payment failure page	Pass	-

Participant 3:

Name: Ng Winnie

Test Starting Time: 11.07pm

Test Ending Time: 11.25pm

Web-based Coffee Shop with Chatbot Integrated				
Test Module	Test Case ID	Test Scenario	Status	Comment
Register	CUAT-001	1. Able to register a new account	Pass	-
Login	CUAT-002	1. Able to login to the newly registered account	Pass	-
Product	CUAT-003	1. Able to view beverage list page	Pass	-
	CUAT-004	2. Able to view homebrew product list page	Pass	-
	CUAT-005	3. Able to view mechanic list page	Pass	-
	CUAT-006	4. Able to apply search filter on product list	Pass	-
	CUAT-007	5. Able to remove search filter on product list	Pass	-
	CUAT-008	6. Able to apply price range filter on product list	Pass	-
	CUAT-009	7. Able to remove price range filter on product list	Pass	-
	CUAT-010	8. Able to apply product category filter on the product list	Pass	-
	CUAT-011	9. Able to remove product category filter on product list	Pass	-
	CUAT-012	10. Able to view product detail page	Pass	-

	CUAT-013	11. Able to view rating and reviews of the product	Pass	-
	CUAT-014	12. Able to view the product recommendations	Pass	-
	CUAT-015	13. Able to select beverage customization options	Pass	-
Cart	CUAT-016	1. Able to view product cart page	Pass	Product and Beverage Cart a bit confusing
	CUAT-017	2. Able to view beverage cart page	Pass	
	CUAT-018	3. Able to add an item into cart	Pass	-
	CUAT-019	4. Able to increase the item quantity in cart	Pass	-
	CUAT-020	5. Able to decrease the item quantity in cart	Pass	-
	CUAT-021	6. Able to delete an item from cart	Pass	-
	CUAT-022	7. Able to view guest beverage cart items after login	Pass	-
	CUAT-023	8. Able view guest product cart items after login	Pass	-
Guiding Materials	CUAT-024	1. Able to view recipe guiding list page	Pass	-
	CUAT-025	2. Able to view brewing guide list page	Pass	-
	CUAT-026	3. Able to view recipe guide detail page	Pass	-
	CUAT-027	4. Able to view brewing guide detail page	Pass	-

Chatbot	CUAT-028	1. Able to view chatbot pop up window	Pass	-
	CUAT-029	2. Able to initiate chat with the chatbot	Pass	-
	CUAT-030	3. Able to get reply from chatbot	Pass	-
Customer	CUAT-031	1. Able to view own order list	Pass	-
	CUAT-032	2. Able to write review on a product	Pass	-
Payment	CUAT-034	1. Able to view payment page	Pass	-
	CUAT-035	2. Able to make payment	Pass	-
	CUAT-036	3. Able to select payment method	Pass	-
	CUAT-036	4. Able to view payment successful page	Pass	-
	CUAT-037	5. Able to cancel payment	Pass	-
	CUAT-038	6. Able to view payment failure page	Pass	-

Participant 4:

Name: Alicia Kwan Xin Ying

Test Starting Time: 7.32pm

Test Ending Time: 7.56pm

Web-based Coffee Shop with Chatbot Integrated				
Test Module	Test Case ID	Test Scenario	Status	Comment
Register	CUAT-001	1. Able to register a new account	Pass	-
Login	CUAT-002	1. Able to login to the newly registered account	Pass	-
Product	CUAT-003	1. Able to view beverage list page	Pass	-
	CUAT-004	2. Able to view homebrew product list page	Pass	-
	CUAT-005	3. Able to view mechanic list page	Pass	-
	CUAT-006	4. Able to apply search filter on product list	Pass	-
	CUAT-007	5. Able to remove search filter on product list	Pass	-
	CUAT-008	6. Able to apply price range filter on product list	Pass	-
	CUAT-009	7. Able to remove price range filter on product list	Pass	-
	CUAT-010	8. Able to apply product category filter on the product list	Pass	-
	CUAT-011	9. Able to remove product category filter on product list	Pass	-
	CUAT-012	10. Able to view product detail page	Pass	-

	CUAT-013	11. Able to view rating and reviews of the product	Pass	-
	CUAT-014	12. Able to view the product recommendations	Pass	-
	CUAT-015	13. Able to select beverage customization options	Pass	-
Cart	CUAT-016	1. Able to view product cart page	Pass	-
	CUAT-017	2. Able to view beverage cart page	Pass	-
	CUAT-018	3. Able to add an item into cart	Pass	-
	CUAT-019	4. Able to increase the item quantity in cart	Pass	-
	CUAT-020	5. Able to decrease the item quantity in cart	Pass	-
	CUAT-021	6. Able to delete an item from cart	Pass	-
	CUAT-022	7. Able to view guest beverage cart items after login	Pass	-
	CUAT-023	8. Able view guest product cart items after login	Pass	-
Guiding Materials	CUAT-024	1. Able to view recipe guiding list page	Pass	-
	CUAT-025	2. Able to view brewing guide list page	Pass	-
	CUAT-026	3. Able to view recipe guide detail page	Pass	-
	CUAT-027	4. Able to view brewing guide detail page	Pass	-

Chatbot	CUAT-028	1. Able to view chatbot pop up window	Pass	-
	CUAT-029	2. Able to initiate chat with the chatbot	Pass	-
	CUAT-030	3. Able to get reply from chatbot	Pass	-
Customer	CUAT-031	1. Able to view own order list	Pass	-
	CUAT-032	2. Able to write review on a product	Pass	-
Payment	CUAT-034	1. Able to view payment page	Pass	-
	CUAT-035	2. Able to make payment	Pass	-
	CUAT-036	3. Able to select payment method	Pass	-
	CUAT-036	4. Able to view payment successful page	Pass	-
	CUAT-037	5. Able to cancel payment	Pass	-
	CUAT-038	6. Able to view payment failure page	Pass	-

Participant 5:

Name: Liang Yi Wei

Test Starting Time: 8.25pm

Test Ending Time: 8.43pm

Web-based Coffee Shop with Chatbot Integrated				
Test Module	Test Case ID	Test Scenario	Status	Comment
Register	CUAT-001	1. Able to register a new account	Pass	-
Login	CUAT-002	1. Able to login to the newly registered account	Pass	-
Product	CUAT-003	1. Able to view beverage list page	Pass	Whole image of the product is clickable
	CUAT-004	2. Able to view homebrew product list page	Pass	-
	CUAT-005	3. Able to view mechanic list page	Pass	-
	CUAT-006	4. Able to apply search filter on product list	Pass	-
	CUAT-007	5. Able to remove search filter on product list	Pass	-
	CUAT-008	6. Able to apply price range filter on product list	Pass	-
	CUAT-009	7. Able to remove price range filter on product list	Pass	-
	CUAT-010	8. Able to apply product category filter on the product list	Pass	-
	CUAT-011	9. Able to remove product category filter on product list	Pass	-
	CUAT-012	10. Able to view product detail page	Pass	-

	CUAT-013	11. Able to view rating and reviews of the product	Pass	-
	CUAT-014	12. Able to view the product recommendations	Pass	-
	CUAT-015	13. Able to select beverage customization options	Pass	-
Cart	CUAT-016	1. Able to view product cart page	Pass	-
	CUAT-017	2. Able to view beverage cart page	Pass	-
	CUAT-018	3. Able to add an item into cart	Pass	-
	CUAT-019	4. Able to increase the item quantity in cart	Pass	-
	CUAT-020	5. Able to decrease the item quantity in cart	Pass	-
	CUAT-021	6. Able to delete an item from cart	Pass	Should have delete confirmation
	CUAT-022	7. Able to view guest beverage cart items after login	Pass	-
	CUAT-023	8. Able view guest product cart items after login	Pass	-
Guiding Materials	CUAT-024	1. Able to view recipe guiding list page	Pass	-
	CUAT-025	2. Able to view brewing guide list page	Pass	-
	CUAT-026	3. Able to view recipe guide detail page	Pass	-
	CUAT-027	4. Able to view brewing guide detail page	Pass	-

Chatbot	CUAT-028	1. Able to view chatbot pop up window	Pass	-
	CUAT-029	2. Able to initiate chat with the chatbot	Pass	-
	CUAT-030	3. Able to get reply from chatbot	Pass	-
Customer	CUAT-031	1. Able to view own order list	Pass	Change the button name to View Order History
	CUAT-032	2. Able to write review on a product	Pass	-
Payment	CUAT-034	1. Able to view payment page	Pass	-
	CUAT-035	2. Able to make payment	Pass	-
	CUAT-036	3. Able to select payment method	Pass	-
	CUAT-036	4. Able to view payment successful page	Pass	-
	CUAT-037	5. Able to cancel payment	Pass	-
	CUAT-038	6. Able to view payment failure page	Pass	-

Participant 1:

Name: Albert

Test Starting Time: 9.30pm

Test Ending Time: 9.45pm

MYCoffee Admin Panel				
Test Module	Test Case ID	Test Scenario	Status	Comment
Login	AUAT-001	1. Able to login to the admin account	Pass	-
Manage Product	AUAT-002	1. Able to view product list	Pass	-
	AUAT-003	2. Able to view product category list	Pass	-
	AUAT-004	3. Able to add new product information	Pass	-
	AUAT-005	4. Able to add new product category	Pass	-
	AUAT-006	5. Able to update on a product information	Pass	-
	AUAT-007	6. Able to update on a product category	Pass	-
	AUAT-008	7. Able to delete a product	Pass	-
	AUAT-009	8. Able to delete product category	Pass	-
Manage guiding materials	AUAT-010	1. Able to view recipe guide list	Pass	-
	AUAT-011	2. Able to view brewing guide list	Pass	-
	AUAT-012	3. Able to add new recipe guide	Pass	-

	AUAT-013	4. Able to add new brewing guide	Pass	-
	AUAT-014	5. Able to update on a recipe guide	Pass	-
	AUAT-015	6. Able to update on a brewing guide	Pass	-
	AUAT-015	7. Able to delete a recipe guide	Pass	-
	AUAT-016	8. Able to delete a brewing guide	Pass	-
Manage Data Embedding File	AUAT-017	1. Able to add new data embedding file	Pass	-
	AUAT-018	2. Able to update a data embedding file	Pass	-
	AUAT-019	3. Able to delete a data embedding file	Pass	-
View Order	AUAT-020	1. Able to view complete order list	Pass	-

Participant 2:

Name: Lee Tung Ling

Test Starting Time: 11.30pm

Test Ending Time: 11.43pm

MYCoffee Admin Panel				
Test Module	Test Case ID	Test Scenario	Status	Comment
Login	AUAT-001	1. Able to login to the admin account	Pass	-
Manage Product	AUAT-002	1. Able to view product list	Pass	-
	AUAT-003	2. Able to view product category list	Pass	-
	AUAT-004	3. Able to add new product information	Pass	-
	AUAT-005	4. Able to add new product category	Pass	-
	AUAT-006	5. Able to update on a product information	Pass	-
	AUAT-007	6. Able to update on a product category	Pass	-
	AUAT-008	7. Able to delete a product	Pass	-
	AUAT-009	8. Able to delete product category	Pass	-
Manage guiding materials	AUAT-010	1. Able to view recipe guide list	Pass	-
	AUAT-011	2. Able to view brewing guide list	Pass	-
	AUAT-012	3. Able to add new recipe guide	Pass	-

	AUAT-013	4. Able to add new brewing guide	Pass	-
	AUAT-014	5. Able to update on a recipe guide	Pass	-
	AUAT-015	6. Able to update on a brewing guide	Pass	-
	AUAT-015	7. Able to delete a recipe guide	Pass	-
	AUAT-016	8. Able to delete a brewing guide	Pass	-
Manage Data Embedding File	AUAT-017	1. Able to add new data embedding file	Pass	-
	AUAT-018	2. Able to update a data embedding file	Pass	-
	AUAT-019	3. Able to delete a data embedding file	Pass	-
View Order	AUAT-020	1. Able to view complete order list	Pass	-

Participant 3:

Name: Samantha Chan

Test Starting Time: 8.12pm

Test Ending Time: 8.30pm

MYCoffee Admin Panel				
Test Module	Test Case ID	Test Scenario	Status	Comment
Login	AUAT-001	1. Able to login to the admin account	Pass	-
Manage Product	AUAT-002	1. Able to view product list	Pass	-
	AUAT-003	2. Able to view product category list	Pass	-
	AUAT-004	3. Able to add new product information	Pass	-
	AUAT-005	4. Able to add new product category	Pass	-
	AUAT-006	5. Able to update on a product information	Pass	-
	AUAT-007	6. Able to update on a product category	Pass	-
	AUAT-008	7. Able to delete a product	Pass	-
	AUAT-009	8. Able to delete product category	Pass	-
Manage guiding materials	AUAT-010	1. Able to view recipe guide list	Pass	-
	AUAT-011	2. Able to view brewing guide list	Pass	-
	AUAT-012	3. Able to add new recipe guide	Pass	-

	AUAT-013	4. Able to add new brewing guide	Pass	-
	AUAT-014	5. Able to update on a recipe guide	Pass	-
	AUAT-015	6. Able to update on a brewing guide	Pass	-
	AUAT-015	7. Able to delete a recipe guide	Pass	-
	AUAT-016	8. Able to delete a brewing guide	Pass	-
Manage Data Embedding File	AUAT-017	1. Able to add new data embedding file	Pass	-
	AUAT-018	2. Able to update a data embedding file	Pass	-
	AUAT-019	3. Able to delete a data embedding file	Pass	-
View Order	AUAT-020	1. Able to view complete order list	Pass	-

Participant 4:

Name: Wong Kai Lin

Test Starting Time: 10.05pm

Test Ending Time: 10.22pm

MYCoffee Admin Panel				
Test Module	Test Case ID	Test Scenario	Status	Comment
Login	AUAT-001	1. Able to login to the admin account	Pass	-
Manage Product	AUAT-002	1. Able to view product list	Pass	-
	AUAT-003	2. Able to view product category list	Pass	-
	AUAT-004	3. Able to add new product information	Pass	-
	AUAT-005	4. Able to add new product category	Pass	-
	AUAT-006	5. Able to update on a product information	Pass	-
	AUAT-007	6. Able to update on a product category	Pass	-
	AUAT-008	7. Able to delete a product	Pass	-
	AUAT-009	8. Able to delete product category	Pass	-
Manage guiding materials	AUAT-010	1. Able to view recipe guide list	Pass	-
	AUAT-011	2. Able to view brewing guide list	Pass	-
	AUAT-012	3. Able to add new recipe guide	Pass	-

	AUAT-013	4. Able to add new brewing guide	Pass	-
	AUAT-014	5. Able to update on a recipe guide	Pass	-
	AUAT-015	6. Able to update on a brewing guide	Pass	-
	AUAT-015	7. Able to delete a recipe guide	Pass	-
	AUAT-016	8. Able to delete a brewing guide	Pass	-
Manage Data Embedding File	AUAT-017	1. Able to add new data embedding file	Pass	-
	AUAT-018	2. Able to update a data embedding file	Pass	-
	AUAT-019	3. Able to delete a data embedding file	Pass	-
View Order	AUAT-020	1. Able to view complete order list	Pass	-

Participant 1:

Name: Ryan Wong

Test Starting Time: 10.30pm

Test Ending Time: 10.47pm

MYCoffee Admin Panel				
Test Module	Test Case ID	Test Scenario	Status	Comment
Login	AUAT-001	1. Able to login to the admin account	Pass	-
Manage Product	AUAT-002	1. Able to view product list	Pass	-
	AUAT-003	2. Able to view product category list	Pass	-
	AUAT-004	3. Able to add new product information	Pass	-
	AUAT-005	4. Able to add new product category	Pass	-
	AUAT-006	5. Able to update on a product information	Pass	-
	AUAT-007	6. Able to update on a product category	Pass	-
	AUAT-008	7. Able to delete a product	Pass	-
	AUAT-009	8. Able to delete product category	Pass	-
Manage guiding materials	AUAT-010	1. Able to view recipe guide list	Pass	-
	AUAT-011	2. Able to view brewing guide list	Pass	-
	AUAT-012	3. Able to add new recipe guide	Pass	-

	AUAT-013	4. Able to add new brewing guide	Pass	-
	AUAT-014	5. Able to update on a recipe guide	Pass	-
	AUAT-015	6. Able to update on a brewing guide	Pass	-
	AUAT-015	7. Able to delete a recipe guide	Pass	-
	AUAT-016	8. Able to delete a brewing guide	Pass	-
Manage Data Embedding File	AUAT-017	1. Able to add new data embedding file	Pass	-
	AUAT-018	2. Able to update a data embedding file	Pass	-
	AUAT-019	3. Able to delete a data embedding file	Pass	-
View Order	AUAT-020	1. Able to view complete order list	Pass	-

Appendix C: SUS Survey

Website SUS - Participant 1:

Name: Erin Kong

Age:23

Occupation Architect

System Usability Scale

© Digital Equipment Corporation, 1986.

	Strongly disagree					Strongly agree
1. I think that I would like to use this system frequently	1	2	3	4	5	
2. I found the system unnecessarily complex	1	2	3	4	5	
3. I thought the system was easy to use	1	2	3	4	5	
4. I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5	
5. I found the various functions in this system were well integrated	1	2	3	4	5	
6. I thought there was too much inconsistency in this system	1	2	3	4	5	
7. I would imagine that most people would learn to use this system very quickly	1	2	3	4	5	
8. I found the system very cumbersome to use	1	2	3	4	5	
9. I felt very confident using the system	1	2	3	4	5	
10. I needed to learn a lot of things before I could get going with this system	1	2	3	4	5	

Website SUS - Participant 2:

Name: Tan Jiun Wei

Age:23

Occupation: Student

System Usability Scale

© Digital Equipment Corporation, 1986.

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	1	2	3	4	5
2. I found the system unnecessarily complex	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5
3. I thought the system was easy to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5
5. I found the various functions in this system were well integrated	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5
6. I thought there was too much inconsistency in this system	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	1	2	3	4	5
8. I found the system very cumbersome to use	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5
9. I felt very confident using the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5

Website SUS - Participant 3:

Name: Ng Winnie

Age: 32

Occupation: Doctor

System Usability Scale

© Digital Equipment Corporation, 1986.

	Strongly disagree					Strongly agree
1. I think that I would like to use this system frequently	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
2. I found the system unnecessarily complex	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
3. I thought the system was easy to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
4. I think that I would need the support of a technical person to be able to use this system	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
5. I found the various functions in this system were well integrated	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
6. I thought there was too much inconsistency in this system	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
7. I would imagine that most people would learn to use this system very quickly	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
8. I found the system very cumbersome to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
9. I felt very confident using the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
10. I needed to learn a lot of things before I could get going with this system	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Website SUS - Participant 4:

Name: Alicia Kwan Xin Ying

Age:23

Occupation: Student

System Usability Scale

© Digital Equipment Corporation, 1986.

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	1	2	3	4	5
2. I found the system unnecessarily complex	1	2	3	4	5
3. I thought the system was easy to use	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
5. I found the various functions in this system were well integrated	1	2	3	4	5
6. I thought there was too much inconsistency in this system	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	1	2	3	4	5
8. I found the system very cumbersome to use	1	2	3	4	5
9. I felt very confident using the system	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system	1	2	3	4	5

Website SUS - Participant 5:

Name: Liang Yi Wei

Age:23

Occupation: Student

System Usability Scale

© Digital Equipment Corporation, 1986.

	Strongly disagree					Strongly agree
1. I think that I would like to use this system frequently	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
2. I found the system unnecessarily complex	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
3. I thought the system was easy to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
4. I think that I would need the support of a technical person to be able to use this system	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
5. I found the various functions in this system were well integrated	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
6. I thought there was too much inconsistency in this system	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
7. I would imagine that most people would learn to use this system very quickly	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
8. I found the system very cumbersome to use	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
9. I felt very confident using the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
10. I needed to learn a lot of things before I could get going with this system	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Admin Panel – Participant 1:

Name: Albert

Age:26

Occupation: Software Developer

System Usability Scale

© Digital Equipment Corporation, 1986.

	Strongly disagree						Strongly agree
1. I think that I would like to use this system frequently							
	1	2	3	4	5		
2. I found the system unnecessarily complex							
	1	2	3	4	5		
3. I thought the system was easy to use							
	1	2	3	4	5		
4. I think that I would need the support of a technical person to be able to use this system							
	1	2	3	4	5		
5. I found the various functions in this system were well integrated							
	1	2	3	4	5		
6. I thought there was too much inconsistency in this system							
	1	2	3	4	5		
7. I would imagine that most people would learn to use this system very quickly							
	1	2	3	4	5		
8. I found the system very cumbersome to use							
	1	2	3	4	5		
9. I felt very confident using the system							
	1	2	3	4	5		
10. I needed to learn a lot of things before I could get going with this system							
	1	2	3	4	5		

Admin Panel – Participant 2:

Name: Lee Tung Ling

Age:23

Occupation: Coffee Shop Worker

System Usability Scale

© Digital Equipment Corporation, 1986.

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	1	2	3	4	5
2. I found the system unnecessarily complex	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5
3. I thought the system was easy to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5
5. I found the various functions in this system were well integrated	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	1	2	3	4	5
6. I thought there was too much inconsistency in this system	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	1	2	3	4	5
8. I found the system very cumbersome to use	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5
9. I felt very confident using the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5

Admin Panel – Participant 3:

Name: Samantha Chan

Age:23

Occupation: Business Investment

System Usability Scale

© Digital Equipment Corporation, 1986.

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	1	2	3	4	5
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2. I found the system unnecessarily complex	1	2	3	4	5
	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3. I thought the system was easy to use	1	2	3	4	5
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4. I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5. I found the various functions in this system were well integrated	1	2	3	4	5
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
6. I thought there was too much inconsistency in this system	1	2	3	4	5
	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
7. I would imagine that most people would learn to use this system very quickly	1	2	3	4	5
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
8. I found the system very cumbersome to use	1	2	3	4	5
	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
9. I felt very confident using the system	1	2	3	4	5
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
10. I needed to learn a lot of things before I could get going with this system	1	2	3	4	5
	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Admin Panel – Participant 4:

Name: Wong Kai Lin

Age:23

Occupation: Software Developer

System Usability Scale

© Digital Equipment Corporation, 1986.

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	1	2	3	4	5
2. I found the system unnecessarily complex	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5
3. I thought the system was easy to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5
5. I found the various functions in this system were well integrated	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	1	2	3	4	5
6. I thought there was too much inconsistency in this system	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	1	2	3	4	5
8. I found the system very cumbersome to use	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5
9. I felt very confident using the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5

Admin Panel – Participant 5:

Name: Ryan

Age:19

Occupation: Student

System Usability Scale

© Digital Equipment Corporation, 1986.

	Strongly disagree						Strongly agree
1. I think that I would like to use this system frequently	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
2. I found the system unnecessarily complex	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
3. I thought the system was easy to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
4. I think that I would need the support of a technical person to be able to use this system	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
5. I found the various functions in this system were well integrated	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
6. I thought there was too much inconsistency in this system	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
7. I would imagine that most people would learn to use this system very quickly	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
8. I found the system very cumbersome to use	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
9. I felt very confident using the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
10. I needed to learn a lot of things before I could get going with this system	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Appendix D: FYP 1 Grading

Project title:	Web-based Coffee Shop with Chatbot Integrated
Student Name	CHEAH WEN HUEY
Supervisor	Lee Ming Jie
Moderator	Sneha Kanchan

Key Assessment for Project Proposal	Supervisor Comments/Remarks
Project Description - Is the problem or need to be addressed clearly presented? - Is the proposed approach or solution clearly presented and justified?	no comment
Project Scope and Objectives - Is the scope of the project clearly defined? - Are the objectives of the project clearly specified? - Are the project scope and objectives appropriate for a final year project?	suggest to add "To evaluate the proposed system..."
Literature Review / Fact Finding for Benchmarking / Verification of Project - Are sources for literature review / fact finding appropriate? - Is information from literature review / fact finding relevant and adequate? - Is information from literature review / fact finding clearly presented and discussed?	no comment