PALARIS: A MOBILE POINT-OF-SALE SYSTEM SOLUTION FOR INVENTORY MANAGEMENT OF MSMES WITH TIME-SERIES FORECASTING USING SARIMA AND PROPHET

A Project Study Presented to the Faculty of

Electronics Engineering Department

College of Engineering

Technological University of the Philippines

In Partial Fulfillment of the Course Requirements for the Degree of

Bachelor of Science in Electronics Engineering

Submitted by:

Acierto, Tricier Dane T.

Aquino, Seamus Rod C.

Dichoso, Marian Gail C.

Ortega, Jee Anne M.

Vargas, Raiah Sherina L.

Adviser:

Engr. Romeo L. Jorda Jr.

August 2023

APPROVAL SHEET

This project study entitled **"PALARIS: A MOBILE POINT-OF-SALE SYSTEM SOLUTION FOR INVENTORY MANAGEMENT OF MSMEs WITH TIME-SERIES FORECASTING USING SARIMA AND PROPHET"**, has been prepared and submitted by the following proponents:

Acierto, Tricier Dane T.	Ortega, Jee Anne M.
Aquino, Seamus Rod C.	Vargas, Raiah Sherina L.
Dichoso, Marian Gail C.	

In partial fulfillment of the requirements for the degree of **Bachelor of Science in Electronics Engineering** is hereby recommended for approval:

ENGR. ROMEO L. JORDA JR.
Project Adviser

ENGR. JOHN PETER M. RAMOS
Panel Chair

ENGR. CHERRY G. PASCION
Panel Member

ENGR. EDGAR A. GALIDO
Panel Member

ENGR. GILFRED ALLEN M. MADRIGAL
Panel Member

ENGR. LEJAN ALFRED ENRIQUEZ
Panel Member

Accepted and approved in partial fulfillment of the requirements for the **Degree of Bachelor of Science in Electronics Engineering.**

ENGR. TIMOTHY M. AMADO
Head, ECE Department

ENGR. NILO M. ARAGO
Dean, College of Engineering

ABSTRACT

Digitalization of inventory management is common in large industries, while Micro, Small, and Medium Enterprises (MSMEs) often rely on manual processes. This study addresses the difficulties faced by micro-business retailers in the Philippines, focusing on sales tracking, inventory management, and cash flow. A mobile point of sale system called Palaris was developed using Dart and Flutter. Palaris integrates with retail hardware for barcode scanning and generates sales invoices. It incorporates SARIMA for sales forecasting and Prophet for demand forecasting.

Field testing evaluated the system's functionality and performance. The deployed Palaris system achieved a performance accuracy of 76.1%, showcasing the effectiveness of the predictive analytics. Users reported high satisfaction (4.04) with system functionality, meeting, or exceeding expectations. Hardware functionality received a neutral response (2.91) but did not significantly impact overall perception. Users expressed satisfaction (3.67) with the predictive model, providing accurate predictions and valuable insights for their operations.

The results demonstrate Palaris' effectiveness in addressing common challenges faced by micro-business retailers. High satisfaction with system functionality and accurate predictions highlights its potential in enhancing sales tracking, inventory management, and cash flow for small retail businesses in the Philippines.

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to all those who have contributed to the success of our project study. Their support, guidance, and collaboration have been instrumental in our journey, and we are truly grateful for their contributions.

First and foremost, we would like to thank the Almighty God for making this project study successful. We acknowledge the role of divine strength and wisdom in enabling us to complete the study on time.

To our beloved parents, we extend our sincere appreciation for their unwavering support throughout our project study. Their collaboration, love, care, and invaluable advice have been constant sources of inspiration and motivation. We are deeply grateful for their presence and guidance during this journey.

Our heartfelt thanks go to our thesis adviser, Engr. Romeo L. Jorda Jr., for his guidance and belief in our abilities. We are immensely grateful for his mentorship and the time he dedicated to monitoring our progress, ensuring that our project study stayed on track and aligned with its objectives.

We would also like to express our gratitude to the esteemed panel members, Engr. John Peter M. Ramos, Engr. Cherry G. Pascion, Engr. Gilfred Allen M. Madrigal, Engr. Lejan Alfred Enriquez, and Engr. Edgar A. Galido. Your valuable evaluation and feedback have played a crucial role in shaping our project study into an innovative endeavor. We are truly grateful for your contributions.

We are also grateful to Engr. Timothy M. Amado, our department head for his expertise and experience, has played a crucial role in assessing the potential and viability of our study, as well as guiding us in refining our project concept and ensuring its relevance to real-world applications.

We would like to acknowledge Engr. Nilo M. Arago, the dean of the College of Engineering. His unwavering support, guidance, and mentorship have been invaluable throughout our project study. We are deeply grateful for his leadership and vision, which have created a nurturing and conducive learning environment.

A special word of appreciation goes to Mrs. Jesmar Casiding, the owner of the Triple J store, our chosen deployment for the project study. Without her full cooperation and collaboration, our project study would not have been as successful. We are truly grateful for her trust and support throughout our journey. Moreover, we would like to express our gratitude for entrusting us with her sales data, which played a crucial role in training our machine learning algorithm models and ensuring their accuracy and effectiveness.

Furthermore, we would like to extend our thanks to the owners of the sari-sari stores located in Batasan Hills. Their generosity in giving us ample time for interviews during our market research has been invaluable. Their insights and perspectives have enriched our study and provided valuable insights into the target market. We sincerely thank everyone who contributed to our project's success. Their support and collaboration were crucial, and we are truly grateful for them.

Team MicroPOS

TABLE OF CONTENTS

APPROVAL SHEET …….…………………..…………………………….....……..... **i**

ABSTRACT ……………………………..…………………….……………..….. **ii**

ACKNOWLEDGEMENT ………………..………...…………..………………..……..…. **iii**

TABLE OF CONTENTS ………………………………………………….…….…..….. **v**

A. LIST OF FIGURES …………...……………………………………….……...… x

B. LIST OF TABLES ……………………….…….……...…….……………….…... xiii

C. ANNEXES ……………………………………….……………………….……..... xv

Chapter 1: The Problem and Its Setting ………………..………….………………. **1**

1.1 Introduction ………..…………………………………..………………………. 1

1.2 Background of the Study ………………………….…..………………..... 2

1.3 Research Gap ……..…………………………………….……………….... 4

1.4 Research Objectives ………...…………………………….…………….... 4

1.5 Significance of the Study …………………………………….....………. 5

1.6 Scope and Limitations ……………………………………….….…..……. 6

1.7 Definition of Terms …...…………………………………………….………. 7

Chapter 2: Review of Related Literature ………………...….................................. **9**

2.1 Micro – Enterprises in the Philippines …………………...…….………….... 9

2.1.1 Inventory Management Tool of Micro-Enterprises …….….......... 9

2.1.2 Financial Management of Micro-Enterprises ……………..……. 10

2.1.3 Sales and Inventory Management System ……………….….…. 10

2.2 Sari-Sari Store in the Philippines ……………………..………………... 14

2.2.1 Philippine Food Market Retailers ………………...………...… 14

2.2.2 Sari-sari Store Layout in the Philippines 14

2.2.3 Inventory Management System of Sari-sari Store 16

2.2.4 Problems of Sari-sari Store in the Philippines 17

2.2.5 Importance of Sari-sari Store in a Neighborhood in the Philippines 18

2.2.6 Success and Failures of Sari-sari Stores 18

2.3 Point-Of-Sale System 23

2.3.1 Difference of Traditional Point-of-Sale and mPOS 23

2.3.2 Point-of-Sale System in Retail Business 23

2.3.3 Point-of-Sale System in Small Business 24

2.3.4 Point-of-Sale System in a Restaurant 24

2.3.5 Challenges to the Efficient Use of Point-of-Sale System 25

2.4 Sales Forecasting 28

2.4.1 A Sales Forecasting Model for the Consumer Goods with Holiday Effects 28

2.4.2 Sales Forecasting of Goods in Shoe Retail 28

2.4.3 Increasing Sales Productivity Through Sales Forecasting Using ARIMA Models 29

2.4.4 Sales-forecasting of Retail Stores Using Machine Learning Techniques 30

2.4.5 Machine-Learning Models for Sales Time Series Forecasting 30

2.4.6 Inventory Management Using Machine Learning 31

2.4.7 An Advanced Sales Forecasting Using Machine Learning Algorithm………………………………………………………..… 32

2.4.8 How to Apply Machine Learning to Demand Forecasting ……… 33

2.4.9 Comparative Study on Time Series Forecasting Models .………. 35

2.4.10 Critical Factors Affecting Sales in Supermarket Chains: A Critical Review of Literature ………………..………..…………………. 36

2.5 Demand Forecasting ……………………….……………………... 44

2.5.1 Successful Demand Forecasting Modeling Strategies for Increasing Small Retail Medical Supply Profitability ……………………... 44

2.6 Effects of Weather on Retail Sales ………………………………....... 46

2.6.1 It's the Weather: Quantifying the Impact of Weather on Retail Sales……………………………………………...……….. 46

2.6.2 A Study of Weather's Impact on Consumption of Goods ……..... 47

2.7 Inventory Management ……………………………………….…...… 50

2.7.1 Inventory Control Management and Repeat Purchase in Policy Reformation……………………..……………………….………… 50

2.8 Barcode Scanner ………………………..…………………….………… 51

2.8.1 Process of Barcode Scanning …………..……………………… 51

2.8.2 Information Content of Barcode and Its Types ……...………….. 51

Chapter 3: Methodology ……………………………………………….……….... 54

3.1 Research Design ……..……………………………….………..……. 54

3.2 Research Process Flow ………..…………………………..……..…… 55

3.3 Development Of A Mobile Application Using The Dart Language And Flutter Framework …… 56

3.3.1 Key Features of the Mobile Application …… 56

3.3.2 User Interface and Backend Design …… 57

3.3.3 Interfacing with the Cloud Database …… 57

3.3.4 Data Dashboard …… 58

3.4 Development of a Sales and Demand Prediction Model using Machine Learning Algorithms …… 58

3.4.1 Univariate Approach …… 61

3.4.2 Multivariate Approach …… 62

3.4.3 Data …… 63

3.4.3.1 Data Parameters …… 63

3.4.4 Validation …… 63

3.5 Statistical Test …… 64

3.6 Deployment …… 64

3.6.1 Locale of the Project Study …… 64

3.6.2 Project Study Instrument …… 65

3.6.3 Data Gathering Procedure …… 65

3.6.4 Data Analysis …… 66

3.7 Project Work Plan …… 67

Chapter 4: Data and Results …… 68

4.1 Project Technical Description …… 68

4.2 Project Organizational Structure …… 69

4.2.1 User Interface ……………………………………….…...……... 69

4.2.2 Database …………………………………………….......……. 84

4.3 Sales And Demand Forecasting Model's Results And Analysis (Training Phase) …………………………………….……………..…… 88

4.4 Deployment Results and Analysis (Actual Phase) ...…………………..... 98

Chapter 5: Conclusion and Recommendations.…………….………………………..... 106

5.1 Summary of Findings …………………………………….………… 106

5.2 Conclusion ……………………...…………………………...………. 107

5.3 Recommendation ……………………………….……..………...... 108

REFERENCES …...…………………………………………………………….. 110

ANNEX ...……………………………………………………….……………….. 119

A. LIST OF FIGURES

Figure 2.1: Competing Values Framework …………………………………….……….... 34

Figure 3.1: Input-Process-Output ……………………………………….………...…… 54

Figure 3.2: Block Diagram …………………………………………………..…… 55

Figure 3.3: Mobile Application to Cloud Platform …………………………..........… 56

Figure 3.4: Process Flow Diagram for Time Series Modeling ……………………… 59

Figure 3.5: Process Flow Diagram ……………………………….……………........ 60

Figure 4.1: Palaris Welcome Page ……………………………..……………..… 69

Figure 4.2: Palaris User Verifier Page: (a) Palaris Login; (b) Sign-Up Page …………………..……………………………………………..…... 70

Figure 4.3: Palaris Home Page: (a) Home Page with Features; (b) Drawer Menu at Home Page ……….……………………………………………….... 71

Figure 4.4: Palaris Profile Page ………………………………………….…….. 72

Figure 4.5: Thermal Printer Compatibility with Instructions Page: (a) Thermal Printer Compatibility; (b) Attached Handheld POS Printer Compatibility ………………..………………………………….…..……….. 73

Figure 4.6: Barcode Generator with Instructions Page …………………..………...74

Figure 4.7: Add Products Page: (a) User Interface of Add Product; (b) Categories of Products; (c) Units Available …………………………………………… 75

Figure 4.8: Products Page: (a) Categories; b) All Products; c) Sample List of Items in a Category ...……………………………………………………… 76

Figure 4.9: Inventory Page: (a) In Stock Page; (b) Low Stock Page; (c) Expiry Page…..………………………………………………...……..… 77

Figure 4.10: Hot Items Page ………………………………………………….. 78

Figure 4.11: Palaris Point-of-Sale via Add Items Method: (a) Point-of-Sale Page**;** (b) Items Summary Page in Point-of-Sale …….………………………79

Figure 4.12: Palaris Point-of-Sale via Barcode Scanning Method: (a) Product Barcode Scanning**; (**b) Items Summary Page in Point-of-Sale ……………….……80

Figure 4.13: Palaris Payment Page Flow: (a) Payable Amount Page; (b) Receipt Preview Print and Finish Page; (c) Confirmation Page ….……………...………... 81

Figure 4.14: Transactions: (a) Daily Transactions Page; (b) Weekly Transactions Page……………………………………………………………….….. 82

Figure 4.15: Sales Report …………………………………………………….. 83

Figure 4.16: Analytics: (a) Sales Forecast; (b) Specific Category; (c) With Two or More Categories..…………………………………………..….................. 84

Figure 4.17: Palaris Cloud Firestore Database Main Tab ….…………...……….…. 85

Figure 4.18: Palaris Cloud Firestore Database – Users Products Collection …………. 85

Figure 4.19: Palaris Cloud Firestore Database – Summary of Sales Collection……… 86

Figure 4.20: Palaris Cloud Firestore Database Predictions – Demand Data Collection……………………………………..…………………... 86

Figure 4.21: Palaris Cloud Firestore Database Predictions – Sales Data Collection…………………………………………………………….… 87

Figure 4.22: Palaris Cloud Firestore Database – Orders Collection ...………..……… 87

Figure 4.23: Line Plot of Actual vs Predicted Sales for March 2023 ...……………..... 89

Figure 4.24: Scatter Plot of Actual vs Predicted Sales for March 2023 ..…………..… 89

Figure 4.25: Residual Plot of Actual vs Predicted Sales for March 2023 ...…………... 90

Figure 4.26: Line Plot of Actual vs Predicted Sales for April 2023 ...……….......... 92

Figure 4.27: Scatter Plot of Actual vs Predicted Sales for April 2023 ...……………... 92

Figure 4.28: Residual Plot of Actual vs Predicted Sales for April 2023 ...…….......... 93

Figure 4.29: Line Plot of Actual vs Predicted Sales for May 2023 ...……….......... 95

Figure 4.30: Scatter Plot of Actual vs Predicted Sales for May 2023 ...……………..... 95

Figure 4.31: Residual Plot of Actual vs Predicted Sales for May 2023 ………………. 96

Figure 4.32: Questionnaire Survey Results for System Functionality ……...……... 100

Figure 4.33: Questionnaire Survey Results for Hardware Functionality ...……..…. 102

Figure 4.34: Questionnaire Survey Results for Predictive Model …......……......... 103

B. LIST OF TABLES

Table 2.1: Summary of Related Studies on Micro-Enterprises …..………………….… 11

Table 2.2: Summary of Related Studies on Sari-sari Store in the Philippines ……….19

Table 2.3: Summary of Related Studies on Point-of-Sale System ……………..…… 25

Table 2.4: Average Train and Test Performance of Models ……………………...… 35

Table 2.5: Average forecast time per model on small and large datasets respectively………………………………………………………….. 35

Table 2.6: Revenue Table for Vietnam based Retail Stores …………...…………..... 37

Table 2.7: Summary of Related Studies on Sales Forecasting ……….……............ 37

Table 2.8: Summary of Related Studies on Demand Forecasting …………............ 45

Table 2.9: Summary of Related Studies on Effects of Weather on Retail Sales………………………………………..………………..... 48

Table 2.10: Summary of Related Studies on Inventory Management ……….......... 50

Table 2.11: Summary of Related Studies on Barcode Scanner ……………………... 52

Table 3.1: Forecast Metrics Summary Table …………………………...….…….... 64

Table 4.1: Actual and Predicted Values for March 2023 …….…………...………… 88

Table 4.2: Performance Metrics for the Sales Forecasting Model for March 2023…………………………………………………………......... 90

Table 4.3: Actual and Predicted Values for April 2023……….....………………….. 91

Table 4.4: Performance Metrics for the Sales Forecasting Model for April 2023...... 93

Table 4.5: Actual and Predicted Values for May 2023……………..…………….… 94

Table 4.6: Performance Metrics for the Sales Forecasting Model for May 2023….... 97

Table 4.7: Computation for the Mean Absolute Error (MAE) of Actual and Predicted Values of Deployment ………………………….………………….……... 98

Table 4.8: Results for T-test for the Data for Actual and Predicted Values of Deployment ………………………………………….……………….…... 99

Table 4.9: Verbal Interpretation for System Functionality ……………………....… 99

Table 4.10: Verbal Interpretation for Hardware Functionality …..…………………. 101

Table 4.11: Verbal Interpretation for Predictive Model …………………………..…… 103

Table 4.12: Overall Mean Results for Survey Questionnaire ………………………. 104

Table 4.13: Verbal Interpretation for Overall Mean Results for Post-Deployment Survey……………………………………………………………….… 105

C. ANNEXES

ANNEX I BILL OF MATERIALS ……………………………………………….. 119

ANNEX II SOURCE CODES …………………………………………………….. 121

ANNEX III USER MANUAL ……………………………………………………... 204

ANNEX IV PROGRESS DOCUMENTATION …………………………………… 210

ANNEX V DEPLOYMENT ………………………………………………………. 221

ANNEX VI SURVEY FORM ……………………………………………………… 225

ANNEX VII PHOTOS FROM PREVIOUS DEFENSE …………………………….. 229

ANNEX VIII PROPONENT’S INFORMATION …………………………………… 233

Chapter 1

THE PROBLEM AND ITS SETTING

This chapter discusses the basis of the rationale for the development of the study.

1.1 Introduction

A small retail shop can be found in almost every corner of the Philippines. It is called a sari-sari store, and it sells essential goods like instant noodles, canned goods, soft drinks, and small packs of shampoos. In different parts of the world, sari-sari stores are called by different names. In Singapore, its name is Mama Shop, in the UK, it is called a corner shop; meanwhile, in the US, it is called a convenience store [1]. Filipinos preferred buying in sari-sari stores for its cheaper price as these shops sell products per piece [2]. Sari-sari stores fall under Micro, Small, and Medium enterprises and can be either micro or small depending on their total assets [3]. The Philippine Statistics Authority recorded 952,969 MSMEs businesses, and 88.77% of these are micro-enterprises, followed by 10.25% of small enterprises [4]. As the backbone of the consumer economy, sari-sari stores make up an estimated 30-40% of retail sales in the Philippines [5].

One of the reasons why sari-sari store fails is because of the lack of inventory management skills in which it includes daily inventory of goods, tabulation of cash flow, identification of the most profitable product, identification of the most popular product and checklist of items that need to be replenished [6]. With poor inventory management, products will expire without the owner's knowledge and will result in a more significant loss of profit [7]. In some cases, if a product gets out of stock and the owner forgot to replenish the item, the shopper may choose a cheaper product or will not buy at all, causing a loss of revenue [8]. This will significantly impact the owners as they lose 4 percent of their annual sales [9]. Hence. Inventory management is crucial to one's business as it balances the risks of overstocking and shortage of products [10].

A mobile POS system is ensuring proper inventory management in a sari-sari store. A mobile point-of-sale system is a portable on-the-go device that turns a smartphone into a checkout point and is more flexible than a traditional POS system [11]. In an ideal POS system, it should have inventory management that keeps track of all products, and it should also be able to create detailed sales reports based on the profit, gross margin, total retail amount and can provide quick summary and charts on the sales performance of the store [12]. A sales forecast is a type of sales report that can anticipate the number of sales a business could make. Sales forecasts can also be utilized to predict seasonal and potential business-impacting concerns [13].

1.2 Background of the Study

The creation of a mobile point-of-sale (m-POS) system is supposed to produce a POS machine capable of recording sales transactions and can be easily transported to any location where a sale transaction occurs; thus, the overall power consumption is low [14]. The mobile point-of-sale technology acquired moderate acceptability, high operability, and moderately high perceived sustainability from the corner store owners. It was discovered that it was possible to train the owners of corner stores in underserved areas to utilize a mobile application with a simplified point-of-sale system to track the sales [15]. A m-POS system is a scalable, cost-effective, and compact solution for a small business. Because the only thing that is required is m-POS software, which can be used on any mobile device that also functions as a card reader, it simplifies matters for merchants and their customers [16]. The mobile application Peddlr is a portable point-of-sale (POS) that allows the user who belongs in micro, and small business inventory, to manage their sales, accounting, and reports with the use of a mobile device [17].

A way of organizing the inventory of the products to sell in a sari-sari store is typically done by having many shelves of varying heights and many products hung from wires in a sari-sari store [18]. By implementing the Internet of Things (IoT) with the Automated Inventory Management System (AIMS) is significant to a well-optimized supply chain, thus preserving human resources and cutting operational costs, increasing the inventory operation's effectiveness [19].

A system for inventory management is necessary to keep track of the inflow and outflow in a business, whether the business is micro or large -financial literacy (i.e., recordkeeping, financial statement analysis, credit, and budget management [20].

It is beneficial for a particular MSMEs to integrate efficient inventory management into a mobile application. A successful business relies on keeping track of its inventory [21]. Lack of recordkeeping is one of the primary factors that lead to the failure of a sari-sari store to survive the business [22]. Sari-sari stores often lead to over-spending their budget as they do not track their expenditures, resulting in the business failing [23].

Most businesses still make use of pen and paper in their business, which is not only inefficient but also time-consuming [24]. The application of the portable Enterprise Research Planning (ERP) system in Dumaguete City was assessed by answering a questionnaire for the business owners or employees in a microenterprise. The respondents answered positively that they could manage their accounting inventory using portable ERP [25]. A mobile sari-sari store application for the inventory control system of Efren's Sari-sari Store was created, wherein the store has a problem in monitoring and managing stocks.

The actual test of the mobile application passed the execution test having a 100% rating on its first cycle. A weighted mean of 4.058 (very good) was the application's rating rated by its users [26].

1.3 Research Gap

In recent years, related studies show the lack of technological implementation in the Wholesale and Retail Trade industry for MSMEs in the Philippines. Prior studies have generally found benefits in applying technology to have an efficient inventory management system for MSMEs. Also, studies show that the usage of point-of-sale (POS) systems technology of large enterprises can also be integrated to the micro and small businesses. While various mobile applications emerged and tried to solve these problems like Peddlr, etc., these applications are lacking in features such as sales and demand forecasting utilizing predictive analytics and expiration date monitoring.

1.4 Research Objectives

The main goal of this study is to develop a mobile point-of-sale system solution for MSMEs with time-series forecasting using SARIMA and Prophet. Specifically, the study aims:

1. To develop a mobile application for point-of-sale system using the Dart language and Flutter Framework.
2. To develop a time series model using SARIMA for sales forecasting and Prophet for demand forecasting.
3. To establish a compatibility between the interface of thermal printers to generate sales invoice.
4. To test and evaluate the system functionality through field testing.

1.5 Significance of the Study

The generalization of this present study would be a great contribution to the knowledge of the students. This study could highly contribute to the United Nations Sustainable Development Goals (SDG Number 8: Decent Work and Economic Development). It could also be highly significant and beneficial specifically to the microenterprises as it will help them to know what will be the product or item that will be best-selling for the current season and, they can monitor easily their inventory flow in their store.

Additionally, this study contributes to the Competitive Industries under the Industry, Energy, and Emerging Technology sector in the Harmonized National Research and Development Agenda (HNRDA) as the system will be integrated with predictive analytics to perform forecasting about the product that will be bestseller for a current seasonal trend.

This study will have an economic impact of bringing a convenience in calculating their budget in which product should be considered buying for each current seasonal trend. Furthermore, this study would also contribute to the social impact of the country by aiding the objectives of Republic Act No. 10644 which is also known as Go Negosyo Act to foster decent work that can give the people continuous profit as the study includes demand forecasting. Hence, it will help to secure the future sales and revenue of the sari-sari stores.

This study will enhance the existing knowledge about the advanced technology that is continuously arising today on the mobile Point-of-Sales (POS) and inventory forecasting. With the aid of these innovations in the mobile Point-of-Sales, future research can be able to develop or formulate more advanced technology with different neural networks and technology integrated that can enhance the predictive analytics and inventory forecasting.

1.6 Scope and Limitations

This study aims to develop a mobile application for point-of-sales of MSMEs and inventory management system to monitor products that need to be replenished.

Additionally, the software specifications are designed to address the main problems and issues of the sari-sari store owners such as difficulty in monitoring the flow of daily consumptions of certain products in their store and developed a predictive model to easily forecast the future sales and demand of the sari-sari store to help them to develop strategic plans in managing the sari-sari store. The mobile application is limited to the following: (1) it can be operated only in android operating system, (2) it can be available in online-access only using Wi-Fi, (3) offers only cash-based payments and (4) it only uses historical data of the chosen deployment mini grocery store, Triple J store which is located in 2096 Rosal Street, Batasan Hills, Quezon City to train the sales and demand forecasting model.

It is also programmed with a database to store its product information and labels. Barcode generation is also integrated to the mobile application to issue barcode to those items that do not have initially barcode. The software is designed to analyze collected data to perform predictive analytics that can identify which product will be the bestseller for each season and to monitor the consumption of items in the mini grocery. This study covers the following: (1) Development of mobile application for the point-of-sales of the MSMEs for the inventory management, (2) Use of SARIMA and Prophet in programming the sales and demand forecasting, (3) Using only time-series forecasting for its predictive analysis capability, and (4) Predict the sales and demand of the chosen mini grocery store for the deployment, Triple J store due to the limitation of the gathered data.

This study focuses on the mini grocery sari-sari store. Mini grocery is a type of sari-sari store in which it is an over-the-counter bigger than the household sari-sari store. In mini grocery stores, customers buy at high volumes of products and because of this, the sellers encounter problems in over-stacking [27].

1.7 Definition of Terms

Demand Forecast Accuracy. Refers to the accuracy of product demand prediction. A Key Performance Indicator of Inventory Management Systems.

Enterprise Research Planning (ERP). In terms of accounting, supply chain operations, risk management, and human resources, the software helps organize and manage the day-to-day activities of a business.

Flutter. A programming tool that uses Dart language created by Google that is used for developing a mobile application.

Inventory Management. It involves the order, storage, use, and sale of a company's inventory, as well as the management of the process.

Inventory Turnover. Refers to the measure of stocks replaced when sold out. A Key Performance Indicator for Inventory Management Systems

Mobile Point-of-Sale. A portable cash register that operates a stand-alone device. Programmed like the traditional POS machines.

Neural Network. It consists of a series of algorithms that will allow the computer to recognize patterns to function as a human brain.

Order Cycle Time. Refers to the average time to fulfill orders.

Out of Stocks. Refers to the frequency of inventory requests that occur without stock being available.

Chapter 2

REVIEW OF RELATED LITERATURE

This chapter contains the various literature and studies subdivided with their respective topics used as reference for this development study.

2.1 Micro-Enterprises in the Philippines

This part tackles the microenterprises in the Philippines in which the sari-sari store belongs to and how these micro-enterprises manage their business in terms of inventory and finances.

2.1.1 Inventory Management Tool of Micro-Enterprises

Based on the findings for the local micro-businesses in Brgy. 3 Em’s Barrio East and Brgy. 8 Bagumbayan in Legazpi City, Philippines, regarding their inventory management, it revealed that the notion of inventory is used by several local micro enterprises in Legazpi City without their knowledge. They are familiar with the fundamental inventory. As a result, it will be simpler to integrate these tools into their inventory although there is a complicated recordkeeping approach employing a point-of-sale system. Micro enterprises do not require technology such as barcodes or a cash register; instead, they might benefit from various inventory management systems. This comprises inventory management for just-in-time, safety stock, FIFO, batch monitoring, and perpetual inventory, demand forecasting and management [28].

In Los Baños, Laguna, a survey was performed for the public market small and micro enterprises having 120 respondents. A weighted average of 1.8 (Agree) was the result of the survey for those who believe that record keeping is important in their business for public agribusiness. For them, proper bookkeeping should be practiced in catering a business. It concluded that, in general, with regards to bookkeeping, the respondents have a positive attitude towards it [29].

2.1.2 Financial Management of Micro-Enterprises

A 100 random microentrepreneurs were assessed in Davao City, Philippines. These micro-entrepreneur’s own sari-sari stores, beauty salons, fish vendors, eateries, and auto repair shops. A 5-point Likert survey (Always, Often, Sometimes, Seldom, and Never) was provided for the respondents. In Table 3, Summary in the Level of Financial Literacy of Micro-Entrepreneurs perceived that the microentrepreneurs in Davao City are moderately (mean = 3.10) engaging themselves in record keeping. Most respondents still record business transactions in notebooks and on paper. Utilizing the right book of accounts, including diaries and ledgers, was not always ensured while collecting data. Small business owners have the possibility to retain records, but not formally [30].

2.1.3 Sales and Inventory Management System

The study proposed a Sale and Inventory Management System to an old retail store to manage the goods in the store more efficiently. The phase development prototype is chosen in designing the system. The study also carried out the development stage in accordance with the modules highlighted in the project's scope of work. Thus, it would be necessary for the system to be developed version by version before it is ready for use. There is an outcome of this system, and that is the creation of a user interface that anybody can use, even if they have yet to gain experience in information technology. Furthermore, the system is performing its functions and assists Rahmath Store in saving inventory management time and paperwork [31].

Table 2.1: Summary of Related Studies on Micro-Enterprises

Title	Author/s	Date	Key Features
Inventory Management Tools for Local Micro Businesses [28]	Cordial, J.	2020	This paper tackles the inventory management of the local micro-businesses in Legazpi City. In the day-to-day operation of micro-business, inventory management in their business helps improve their sales by constantly keeping track of which products to sell and which products should be replaced or removed. A clear understanding of implementing the appropriate inventory management will have an efficient return in their business operations. Data was gathered by surveying the local micro-business owners.

Determinants and Impact of Recordkeeping to Agribusiness Growth: A study on Public Marketsmall and Micro Enterprises in Los Baños, Laguna [29]	Cuevas, K.S.M.	2017	In the survey on the attitudes toward recordkeeping, the result shows that most of the respondents agreed that when it comes to the budget decision, recordkeeping helps them decide. For the responders, recordkeeping is fruitful and necessary. However, it is unnecessary for small businesses such as vegetable vendors to track their income since it is small. In addition, owners manage their recordkeeping at the daybook level (composition notebook, pocket notebook, blue record book, and the like).
Determinants of Financial Literacy of Micro Enterprises in Davao City [30]	Sucuahi, W.	2013	This paper is about financial literacy (record keeping, budgeting, personal finance, and savings) based on two (2) variables – gender and educational attainment. Most of the responders are female, and the educational attainment of most

			entrepreneurs is at the college level with 39%; however, the result for this is close, with college graduates having 32% responder frequency. In the findings of this study with the use of regression analysis, a factor in increasing financial superiority is education and not the entrepreneur’s gender.
Sales and Inventory Management System [31]	R. Bee and A. Aleem	2013	This study was all about how the products can be managed in a more efficient way in the old retail stores. The development of this study was based on a window-based or offline system. Data was gathered in the form of interviews and in-depth search on the internet about the sales and inventory system. Microsoft Visual Basic, Microsoft Access, Online Project Management, Diagram Tools, and Microsoft Project Professional Office were used

			as the tools for this study. The outcome of this study was the system can be used by anyone even with no IT experience. Data mining was also defined to help to study the patterns of sales to formulate marketing strategies

2.2 Sari-sari Store in the Philippines

This part discusses the necessary information in a sari- sari store that is divided into specific parts.

2.2.1 Philippine Food Market Retailers

There are four (4) key food retailers in the Philippines in October 2017, namely: Supermarket, Neighborhood Supermarkets, Hypermarket/ Warehouse, and Convenience Stores. The Sari-sari-Stores (a.k.a. Mom & Pop Stores) fall under the traditional retailers. Averaging the share sales for 2012-2017, it dominates the country's retail market having 59.5% share sales according to the Philippines' food retail sales by channels [32].

2.2.2 Sari-sari Store Layout in the Philippines

Sari-sari stores differ from one another with their concept and design. The following are the layouts for a sari-sari store that is found in the Philippines [27].

- *Store on Wheels*

  A type of retailer is often situated in high-traffic areas and typically sells things to a certain market.

- *Corner Store*

  A convenience store is commonly referred to as a retailer. This store should be situated in good areas close to the gate entrances of subdivisions, schools, and Brgy. Halls, among other potential locations.

- *Long-Counter Store*

  This retailer incorporates a convenience store with regular stores, allowing customers to enter but not the act of picking the items by themselves. The customer pays at the long counter after choosing the needed items.

- *Mini Grocery*

  It is a more significant kind of retailer than the conventional sari-sari store. It strives to fulfill all its clients' shopping needs by being a one-stop-shop.

- *Kiosk/ Carts*

  A kind of retailer that is ideally located inside the malls or transport terminals. Applicable for customers seeking a quick grab of their needs.

- *Traditional/ Nipa Hut Store*

  The origin concept that probably started the sari-sari store in the country by selling sorts of goods to the neighborhood.

- *Home Store*

  A retailer attached to the owner's home and typically found in a neighborhood. The set-up is closely comparable to the traditional/ nipa hut store.

- *Portable/ Cabinet Store*

  The store functions similarly to a cabinet wherein it just closes and opens. The goods are displayed in the cabinet.

2.2.3 Inventory Management System of Sari-sari Store

The relationship between livelihood and consumption/savings motives of the sari-sari store is unique since capital is offered in-home products. Even if the store does not generate enough sales, capital can be carried over without significant losses. Unsold inventory acts as both an investment and savings in kind. Only sold items count toward production, whereas those in inventory are earmarked for future or current consumption. Variable-income households must be able to integrate income generation with savings goals. Even if the household consumes all or most of its inventory, it will have saved money. The sari-sari store helps the household combine production and consumption goals [23].

The income of a sari-sari store could positively increase by having different products in the inventory -this is done by selling various goods according to the community belonging. What the community's consumers may need daily is bought in fewer quantities (candies, biscuits, drinks such as soft drinks, and toiletries). The more variety of goods in the inventory of a sari-sari store, the better the profit to be accumulated [33].

2.2.4 Problems of Sari-sari Store in the Philippines

The household sari-sari store of 19 barangays of Naspit, Agusan del Norte, Philippines, was assessed via a self-made questionnaire regarding their accounting literacy and sustainability. Part of the questionnaire assessed the owners if they: (1) Maintain a book of accounts, (2) Prepare financial reports, (3) Maintain cash record book, (4) Voucher system of cash payments, (5) Issue sales receipt, and (6) Maintain bank accounts. The questionnaire results show that the sari-sari store entrepreneurs lack knowledge in accounting since the garnered data for their accounting literacy is 1.38 (never). In conclusion, a sari-sari store that practices basic accounting has existed for more than fifteen years of business operation. On the other hand, a sari-sari store that does not engage themselves in accounting exists just less than a year [34].

The primary factors that contribute to the failure of sari-sari stores to survive and thrive are a lack of record-keeping, an inability to manage the store in a scientific manner, the diversion of inventory and product use, and a mentality that focuses on the small-scale. The collection of debts and insufficient funds resulting from a lack of accounting were two of the most common challenges that sari-sari stores encountered. A strategy for increasing sales at sari-sari shops is to extend credit to customers and provide credit for purchases [22].

A high-rate of failures masks behind the numerous numbers of sari-sari stores in the Philippines, and that is poor financial management. Many sari-sari store owners do not track their store's costs, expenses, and sales. They overspend due to this error. Sari-sari store owners often spend more because they think they are making a lot which results in no savings [28].

2.2.5 Importance of Sari-sari Store in a Neighborhood in the Philippines

162 respondents in Leyte and Samar Islands were assessed regarding the consumption and distribution of the sari-sari store in the Philippines base of the pyramid market. In his findings, the sari-sari store in the mentioned provinces acts as middlemen between large-scale retailers (such as direct-sale companies, supermarkets, and markets) and the customers who shop at those establishments. It is highly beneficial for those individuals living in poverty since they are the ones who are unable to purchase large quantities of items from these large-scale retailers. Additionally, sari-sari stores provide the needs of an individual who needs an item in smaller quantities. The interviews support the importance of sari-sari stores for Philippine consumers, evidenced by their omnipresence [35].

Sari-sari stores are important in a neighborhood as this is where consumers nearby buy their necessities (i.e., hygiene products). Majority of the responder's respondent that items are cheaper in a sari-sari store to buy their basic goods [36].

2.2.6 Success and Failures of Sari-sari Stores

Women entrepreneurs are increasingly growing in numbers. This was fueled by their desire to earn money on their own and improve their financial situation. The success and difficulties of women micro entrepreneurs managing their 'sari-sari' stores were highlighted in the research. Five women micro-entrepreneurs who had been in business for a while were interviewed for this multiple case study. They are at least five years in business. An in-depth interview was conducted, as well as cross-case analysis. New investments, increased family income, productivity, and resourcefulness were among the stories highlighted in which they hailed as a triumph. They emphasized company breakdowns and failures in their failure stories. Further, they also emphasized that their operation was once challenged with a personal dilemma. They also revealed that failures framed them to be determined, dependable, attentive, and upbeat. Finally, they highlighted important information that women entrepreneurs should know. These are to be business oriented, people skills development, and a strong will power in which these female business owners have. They have their own way of determining success and failure in micro entrepreneurship. The researchers also emphasized that their respondents had numerous failures that formed their entrepreneurial instincts. And because of that, they now have everything they need to be successful in their business [37].

Table 2.2: Summary of Related Studies on Sari-sari Store in the Philippines

Title	Author/s	Date	Key Features
2017 Food Retail Sectoral Report [32]	Claridades-Rubio, J.	2017	This report identified the key food retailers in the Philippines for October 2017 with their respective store names or company. The sari-sari store falls under the traditional retailers, which sell a variety of basic food and goods on a smaller scale.

8 Sari-sari Store Designs and Concepts [27]	NA	(n.d.)	This web page enumerates and differentiates the different designs and concepts to consider in starting a retail business of a micro-entrepreneur.
The Filipino Sari-sari Store [23]	Malapit, H.J.	2007	This article provides an agent-based, rational choice explanation for the progression of sari-sari stores into residential neighborhoods over 25 years in the Philippines. Thus, it assessed the relationship of the sari-sari store's link between income and consumption/savings is unique because capital is offered in-house.
Income Growth Factors in Sari-Sari Stores and its Contribution to the Philippine Economy [33]	Santa Maria, Ma. S. C. C.	2019	In order to have an impact on the income growth of the sari-sari store, some factors need to be considered, and one of these is the inventory. To gain a significant sale, there should also be numerous products to sell, thus, acquired with proper inventory management.

Accounting Literacy and Sustainability Among Sari-Sari Stores: Basis on Intervention Program [34]	De Lima, M., Samson, G.C., Taghoy, J.V., and Veranio, E.L.	2020	Practicing accounting is a factor in building a sari-sari store successfully. The reason for the existence of a sari-sari store for a short period is because of a lack of knowledge to practice essential accounting, which is an important factor in a successful operation in a growing business.
Sari-Sari Store Owners Get Lift from Sun Life [23]	Arceo-Dumlao, T.	2012	This article discusses the poor financial management acquired in a sari-sari store. The existence of a sari-sari store often leads to failure for not involving themselves in organizing their inventory and the importance of financial basics.
Distribution and consumption in the Base of the Pyramid (BOP) market: The case of Sari-Sari stores in the Philippines [35]	Funahashi, T.	2013	This paper studies the distribution and consumption of retailers in the Philippines. The consumption of goods from the large firms through selling it to small-scale retailers in the Philippines which is the sari-sari store. It is also indicated that a sari-sari store is

			essential in the country and can be seen everywhere.
Importance of Sari-Sari Stores in the time of a Pandemic [36]	Alindahao, K, Donozo, M. T. and Jullian, A.	2020	The importance of sari-sari stores during the pandemic was assessed in this paper by conducting a survey. The results show that a sari-sari store is an essential part of a neighborhood as this is the most convenient way to purchase a small number of items at an affordable price.
Success and Failures of Sari-sari Stores [37]	J. C. Gano-An and G. P. Gempes	2020	This study highlights the success and difficulties of women micro entrepreneurs managing their 'sari-sari' stores. The respondents were 5 who have at least 5 years engaged in business. In managing ‘sari-sari stores’, one should possess qualities such as business oriented, people skills development, and a strong will power

2.3 Point-of-Sale System

This part tackles the differences of the point-of-sale system and how it is used in different enterprises as well as the challenges in using it.

2.3.1 Difference of Traditional Point-of-Sale and mPOS

mPOS simply means mobile point-of-sale. It differs from traditional POS in terms of device usage, data storage and capacity, and convenience for both software and hardware. It is a modern mobile solution that is built for tablet and/or smartphone use rather than desktop computer. The data storage is empowered using cloud technology hence improving the data capacity, compared to traditional POS wherein the transactional data were stored at the site of the cash register and owners are needed to manually track the sales and inventory throughout the day [38].

In terms of hardware, POS requires a hardware front as a desktop computer, but mPOS can run using a tablet/smartphone which allows a fast transaction of payments from anywhere. And in terms of software, the owner can access the store’s data from any internet connected device from any location compared to the traditional which needs to be on site [39].

2.3.2 Point-of-Sale System in Retail Business

Small businesses must create a captivating retail experience in order to sustain their competitive edge, and in order to enable this experience, the right technologies must be employed. The purpose of this case study is to evaluate and select the right point of sale (POS) and inventory management (IM) systems for a small business by considering both the sectoral needs and the user's needs. Using a Rhode Island-based company - Wildwood Inc. - as a case study, the project creates a three-step framework for real-world implementation of these systems. The checkout and inventory management practices at Wildwood Inc. are manual. Growth makes it difficult for the company to provide effective customer service. After analyzing the specific challenges Wildwood faces, a POS and implementation roadmap is created. Businesses looking to upgrade their systems can use this roadmap as a general guide [40].

2.3.3 Point-of-Sale System in Small Business

This study opted to recommend a POS system suitable for small enterprises. The result of their study revealed that a group of small retailers may utilize a shared, distant database server as a solution, but such systems are frequently unreliable and slow. The authors propose a POS system that uses local data tables implanted in each client's PC to address these concerns. A remote database server and several client POS machines with offline data handling capabilities make up this system. The findings showed that small firms can benefit significantly from a low-cost POS system based on a remote client-server approach with a local data cache [41].

2.3.4 Point-of-Sale System in a Restaurant

Point of sale (POS) systems are transaction mechanisms in a trading environment. It is used to know the price of an item, let the consumer view the price through the POS, and print a receipt or label. POS systems include handheld devices, cash registers, optical scanners, special terminals, and magnetic card readers. The POS system is used in hospitality and retail industries [42].

2.3.5 Challenges to the Efficient Use of Point-of-Sale System

The study explored the challenges of using point of sale (POS) terminals efficiently in Nigeria. The features of both users and non-users were investigated. Data was gathered in cross section from 650 respondents who were chosen at random from 20 local government areas in Lagos State, Nigeria. Friedman and Kendall ranking order tests were used to assess the data. The study's conclusion indicated that the most difficult element impeding POS efficiency is network failure, which occurs frequently, Limited quantities of POS each merchant business when they are available, and power outage in which communication via the network and POS inaccessibility at all merchant locations [43].

Table 2.3: Summary of Related Studies on Point-of-Sale System

Title	Author/s	Date	Key Features
Difference between POS and MPOS System [38]	Cookson, M.D., and Strik, P.M.R.	2021	This web page tackles how MPOS differs from POS. It is indicated that instead of using the traditional desktop computer for a business transaction, tablets and other smartphone devices are used – substituting the conventional apparatus, thus increasing flexibility during work hours.

The Difference Between POS and mPOS & Mobile Payments Explained [39]	KoronaPOS	2019	This blog discusses the differences between POS, mPOS, and mobile payments as there is confusion for the customers involved. A POS system works with external devices, making transactions easier. On the other hand, mPOS can operate just like POS however, it is much more portable, and it is easier to use than POS.
The Power of Point of Sale Improving Growth, Profit, and Customer Service in a Retail Business [40]	Cote, M., and T. McCarthy	2015	This research conducted on retail agriculture business in Wildwood Nursery Inc., wherein there is an interest in updating from a manual point of sale system to an electronic system. This highlights the benefits and pitfalls of a particular system that could have a great factor to the needs and growth of the company.
A POS system based on the remote client-server model in the	Y. G. Kim and J. Lim,	2011	This study proposed a POS system that uses local data tables implanted in each client's PC to address the concerns related to a shared, distant

small business environment [41]			database server. The findings showed that small firms can benefit significantly from a low-cost POS system based on a remote client-server approach with a local data cache.
IT PROJECT MANAGEMENT: CLASS PROJECT OF A POINT OF SALE (POS) SYSTEM IMPLEMENTATION IN A RESTAURANT [42]	A. Gillum and M. A. Rob	2011	It clearly explained what a POS actually does. It is defined as the transaction mechanisms in a trading environment and can be used in hospitality and retail industries
African Journal of Business Management Challenges to the efficient use of point of sale (POS) terminals in Nigeria [43]	O. O. Adeoti.	2013	In this study, data was gathered from 650 respondents who were chosen at random from 20 local government areas in Lagos State, Nigeria.It concludes that the most difficult element impeding POS efficiency is network failure.

2.4 Sales Forecasting

This part tackles how neural networks are made to perform sales forecasting and how machine learning algorithms are able to predict future demand.

2.4.1 A Sales Forecasting Model for the Consumer Goods with Holiday Effects

This study used SPSS, a statistics software to create an ARIMA model to forecast the trend of a seasonally adjusted series. By considering the holidays as key parameters which can affect the sales of consumer goods. Using the Adjusted Dickey-Fuller test showed that the dataset used in this study was a first-order stationary series where $d = 1$. This study also used the ACF and PACF functions to check for randomness in the dataset to define the p and q. The data series were tailed therefore it is suitable for the ARMA model. Through Akaike's Information Criterion as baseline to choose a suitable model ARIMA (1,1,1) was selected. Evaluating the model and reviewing ACF and PACF graphs show that the residual series were white noise series confirming ARIMA (1,1,1) as the best prediction model for the stationary series. The sales forecast results showed that the model has an average error (after calculation) of 4.20% in a monthly timeframe and can effectively simulate seasonality and random characteristics of cigarette sales. The model also supplied an upper and lower limit for the forecast value to supply space for empirical adjustments.[44]

2.4.2 Sales Forecasting of Goods in Shoe Retail

This study used the Prophet model to predict the sales of shoes in a time-series additive model where nonlinear trends which correspond to fit periodic seasonality and the addition of holiday effects as parameters or regressors. This study stated that the Prophet model works best with time series with seasonal effects. The model is found to be robust in shifts and missing data. Additionally, it is stated that it handles outliers well. The model used three main components namely, trend, seasonality and holidays and is defined by the equation.

$$y(t) = g(t) + s(t) + h(t) + \varepsilon$$

In hopes of reducing human influence on the distribution of goods, the implementation of this model in shoe retail aims to solve the forecasting problems of the shoe industry and provide a systematic way to provide analysis to existing data which drives business decisions.[45]

2.4.3 Increasing Sales Productivity Through Sales Forecasting Using ARIMA Models

Using local data, this study implemented an ARIMA model to perform sales forecasting for business intelligence applications. With data from January 2008 to October 2015 and performing exploratory data analysis using R aimed at predicting future sale values for different types of products with high accuracy. The model was validated using MAE (mean absolute error) and MAD (mean absolute deviation) to quantify model bias. The results show an average absolute error of 0.66 units stating a considerable amount of accuracy for the generated model but also recommended adjustments and hyperparameter tuning to generate better predictions [46].

2.4.4 Sales-forecasting of Retail Stores Using Machine Learning Techniques

Sales forecasting is the process of predicting future sales based on historical sales activity. Sales forecasting is of the utmost importance for businesses entering new markets, introducing new services or products, or experiencing rapid development. The primary purpose of a company's projection is to balance marketing and sales resources with supply capacity planning. Examining the range of forecasting techniques, such as Multiple Regression, Polynomial Regression, Ridge Regression, and Lasso Regression, as well as several boosting algorithms, such as AdaBoost and Gradient Tree Boosting, to achieve the highest level of accuracy. Multiple Regression is a statistical technique used to predict an outcome that is dependent on multiple independent predictors or variables. Simple linear regression is extended by polynomial regression. Where the model identifies a nonlinear association between the independent variable x and the dependent variable y. AdaBoost is a boosting method that is mostly employed to enhance the performance of models. It uses the outputs of other weak learners and combines them into a weighted sum before arriving at the output [47].

$$\text{MAE} = \frac{\sum_{i=1}^{n} \textit{Actual value} - \textit{Forecast value}}{n}$$

$$\text{MAD} = \frac{\sum_{i=1}^{n} |\textit{Actual value} - \textit{Forecast value}|}{n}$$

2.4.5 Machine-Learning Models for Sales Time Series Forecasting

Predicting sales is a regression issue rather than a time series problem. Using regression techniques for sales forecasting can frequently yield superior results compared to time series methods. One of the fundamental assumptions of regression algorithms is that previous data patterns will be reproduced in the future. The precision on the validation set is a crucial metric for determining the ideal number of machine learning algorithm iterations. The result of machine learning generalization is the identification of patterns in the entire data set. This effect can be used to anticipate sales when there are few previous data points for certain sales time series, such as when a new product or store is introduced. In the stacking method, the outcomes of multiple model predictions on the validation set serve as input regressors for the subsequent level of models. As the model at the next level, Lasso regression might be utilized. Using stacking makes it possible to account for discrepancies in the outputs of many models with various parameter settings, hence enhancing the accuracy of validation and out-of-sample data sets [48].

2.4.6 Inventory Management Using Machine Learning

This study used the XGBoost regression model, which is a type of machine learning algorithm that uses decision trees to perform the demand forecasting. The data that were used as an input in this study were tabulated and structured which is suitable for the XGBoost regression model. In building the XGBoost model, it requires decision trees which are built from the error factor of the previous decision trees to train the input data. The error factor that was generated from the previous trees were computed by these following steps: (1) Create a model that fits to the input data, (2) Create a model which fits to the residuals, (3) Generate an enhanced model based on the previous model, and (4) Repeat the previous steps until the error factor is well-calibrated. After the implementation of the model, training of the model comes next. Training the model is similar to other algorithms which trains the input data until it is close to the desired result. However, it has an approach called early stopping rounds which aims to avoid the repetitive training of the model when the model reaches its accuracy. This study concluded that inventory forecasting minimizes the cost of manual labor and lets the owners of the stores maintain and regulate the flow of stocks based on their demand. It also prevents overstocking and out-of-stock of items [49].

2.4.7 An Advanced Sales Forecasting Using Machine Learning Algorithm

In this study, it was defined that retail sales contribute a major role in data mining. Data mining refers to the analysis of data in search of patterns for data forecasting. Data mining in sales can systematically solve the discounts, inventory stage, and tips using the previous dataset of these kinds of data. The forecasting system for this dynamic data remains to be a challenge to develop as the input data of sales is not constant and changing every day. The rate of income of the store depends on the promotions, holidays, seasonality, and locality. This study used a Machine Learning Algorithm called Extreme Gradient Boosting in which the data sales is analyzed by predictive-trees models. The training of data is repetitive until it reaches its maximum peak, and no enhancements will be made. The testing of this study involves the prediction of sales of the Rossman shops of 1115 shops for the next 6 weeks. It includes the dataset between January 2013 to July 2015 which has a number of days of 942. The attributes they used are store ID, sales, customer open, keep type, assortment, promo, promo intervals, store, holidays, and distance. The results are that the sales will be high on Sunday but the magnitude of people going to the shops was less likely than the usual day because the stores are closed on Sunday. XGBoost found to be the most effective type of Machine Learning in predicting the sales among the Linear Regression and Random Forest Regression as they are built for linear datasets only and not applicable for nonlinear datasets [50].

2.4.8 How to Apply Machine Learning to Demand Forecasting

This article provides an overview on the application of machine learning on demand forecasting. Demand forecasting is defined as the process of predicting the demand for certain products in the future. And is aimed to improve (1) Supplier relationship management, (2) Customer relationship management, (3) Order fulfillment and logistics, (4) Marketing campaigns, (5) Manufacturing flow management. Although the majority of these processes are focused on manufacturers, it can also be applied to retailers which also provide valuable information for business intelligence and supply chain management (SCM). Because of machine learning approaches, retailers can create a robust system to automate forecast updates based on data to help drive business decisions and increase adaptability to changes.

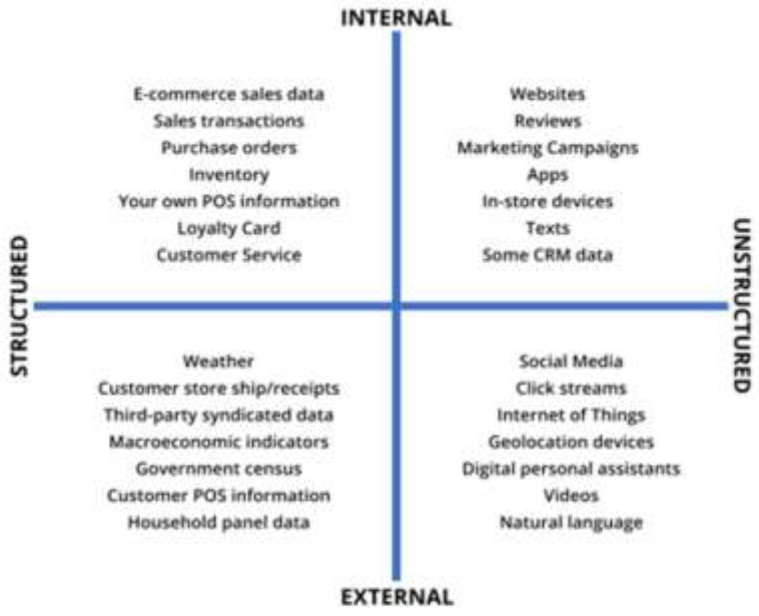


Figure 2.1: Competing Values Framework

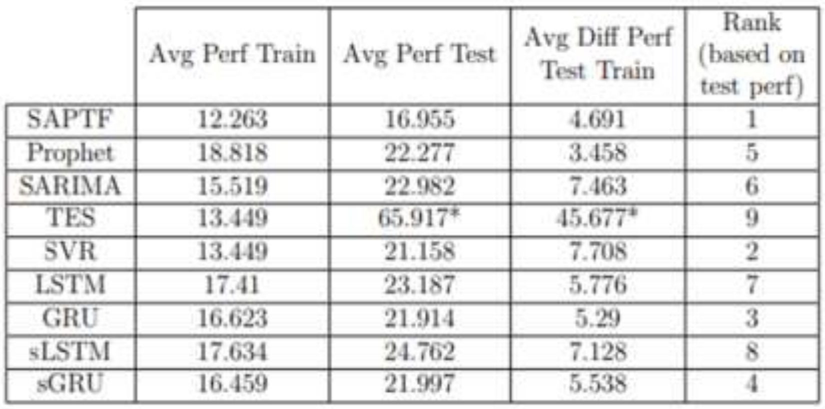
In continuation, it also encourages the identification of exogenous and endogenous variables that are either structured or unstructured. It is also important to evaluate the data according to parameters such as consistency, accuracy, validity, relevance, accessibility, and completeness. The approaches on demand forecast algorithms vary and often depend on business goal, data type, data amount, quality, forecasting period. The time series approach uses data points that occur on a certain period and is used to predict the future values of that variable, this requires successive, periodical data which has four main components for analysis: trend, seasonality, irregularity, cyclicity. ARIMA, SARIMA, Exponential Smoothing, XGBoost, KNN Regression, Random Forest, LSTM are all machine learning approaches that can be hyperparameter tuned to work for demand forecasting. Validating these models can utilize numerous metrics depending on the model, RMSE (root mean square error), MSE (mean square error), AE (absolute error), PE (percentage error), APE (absolute percentage error), are methods to quantify Forecast Error of the model and the Accuracy can be calculated by: [51]

2.4.9 Comparative Study on Time Series Forecasting Models

In this study, various machine learning approaches were compared using univariate time series using these performance metrics to measure forecast performance which accounts sensitivity to data transformation and outliers.

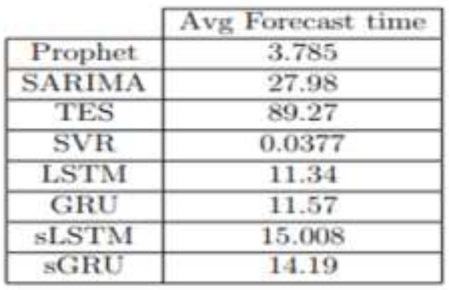
Table 2.4.: Average Train and Test Performance of Models

$$Accuracy = APE - 100\%$$

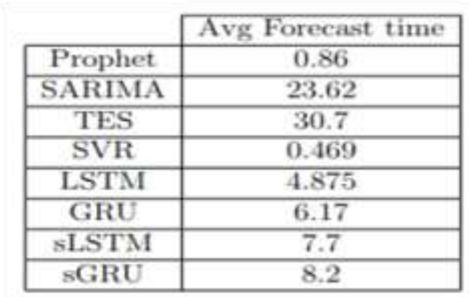


	Avg Perf Train	Avg Perf Test	Avg Diff Perf Test Train	Rank (based on test perf)
SAPTF	12.263	16.955	4.691	1
Prophet	18.818	22.277	3.458	5
SARIMA	15.519	22.982	7.463	6
TES	13.449	65.917*	45.677*	9
SVR	13.449	21.158	7.708	2
LSTM	17.41	23.187	5.776	7
GRU	16.623	21.914	5.29	3
sLSTM	17.634	24.762	7.128	8
sGRU	16.459	21.997	5.538	4

Table 2.5: Average forecast time per model on small and large datasets respectively



	Avg Forecast time
Prophet	3.785
SARIMA	27.98
TES	89.27
SVR	0.0377
LSTM	11.34
GRU	11.57
sLSTM	15.008
sGRU	14.19



	Avg Forecast time
Prophet	0.86
SARIMA	23.62
TES	30.7
SVR	0.469
LSTM	4.875
GRU	6.17
sLSTM	7.7
sGRU	8.2

The study chose three metrics, MAPE, RMSE and R2 to evaluate the performance of the model. Based on the experiments, on average SAPTF shows exceptional performance and is closely followed by SARIMA, Prophet, SVR and sGRU, while Prophet showed an exceptional average forecast time which required a certain kind of datetime indexing format. Out of all the models, SAPTF, SARIMA, TES, and Prophet are the only explainable models, whereas other models provided good forecasts but cannot be explained why.[52]

2.4.10 Critical Factors Affecting Sales in Supermarket Chains: A Critical Review of Literature

This study focused on the supermarket retail industry and factors that affect its revenues as well in Vietnam. From the results of the literature review. It identified that there are six notable factors to consider. One is product quality, second is its price, its volume of sales, geographic location of the market, competitors, and marketing policies. Resulting in six hypotheses:

H1: Increasing the quality of goods helps increase sales revenue.

H2: Low pricing strategies help increase sales revenue.

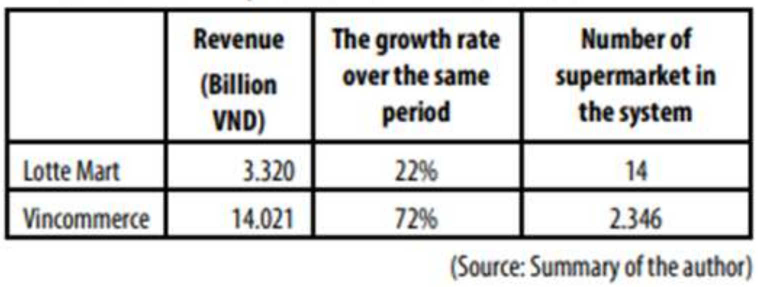
H3: Increasing sales volume help increase sales revenue.

H4: Favorable locations helps increase sales revenue.

H5: High number of competitors reduce sales revenue.

H6: Promoting marketing policies and trade help increase sales revenue.

Table 2.6: Revenue Table for Vietnam based Retail Store



	Revenue (Billion VND)	The growth rate over the same period	Number of supermarket in the system
Lotte Mart	3.320	22%	14
Vincommerce	14.021	72%	2.346

(Source: Summary of the author)

The revenue of local businesses in the retail space showed that the greater number of stores allowed more potential for growth and higher revenue in the first quarter of 2019. As much as the retail industry is characterized in the paper. The results of the study highlighted that the impact of increased revenue contributes to balance to supply and demand, stabilizing market prices and expanding economic exchanges between regions. From an enterprise perspective, it allowed businesses to meet the material needs of society providing various goods. However, this paper did not define specific parameters that represent their study objectives and suggested further research. [53]

Table 2.7: Summary of Related Studies on Sales Forecasting

Title	Author/s	Date	Key Features
A Sales Forecasting Model for the Consumer Goods with Holiday Effects [44]	Huang, X., Yang, C., and Zhang, M,	2020	This paper proposes a sales forecasting model for consumer goods based on the seasonal decomposition and ARIMA model. In order to further analyze the

			trend, forecast for the seasonally adjusted series, the study relies on the ARIMA model to perform this research.
Sales forecasting of goods in shoe retail [45]	Kaipov, I. and Nedzved, A.	2020	A consideration of machine learning-based sales forecasting approaches and methods is presented in this article through a discussion of methods and approaches. In comparison with other methods considered to produce forecasts, the Prophet forecasting method was found to be the most accurate to reality in terms of accuracy.
Increasing sales productivity through sales forecasting using ARIMA models [46]	Dassanayake, D.M.G.T., Nugaliyadde, A., and Mallawarachchi, Y.	2016	This study focuses on integrating sales forecasting with automated sales force management so that it can make intelligent business decisions based on the forecast. Many products are regularly forecasted for a variety of reasons.

			Analyzing and forecasting time series data has been made possible using statistical models. Due to their accuracy and the mathematical soundness of ARIMA models, these techniques are used to analyze time series. They have been used primarily for the forecasting of loads in the past.
Sales-forecasting of Retail Stores using Machine Learning Techniques [47]	Aich, A., Akhilesh, V., Hegde, C., and Krishna, A.	2018	This article made sales forecasts for a retail store with the implementation of various machine learning approaches and attempted to identify the one that proved to be the most effective in solving our problem statement. This method used both traditional regression techniques and boosting approaches. Comparing the two different approaches, it was discovered that the boosting algorithms produce significantly

			superior outcomes to the regular regression algorithms.
Machine-learning models for sales time series forecasting [48]	Pavlyshenko, B. M.		This paper studied predictive analytics for sales forecasting using machine-learning models. When launching new items to sell, historical data were used in training the machine learning with their specific time-series sales to predict the sales of the newly launched items. A technique was added to improve the predictive models for sales time series forecasting; this is with the work of the stack approach.
Inventory Management using Machine Learning [49]	Praveen K B.	2020	This study used the XGBoost regression model. Training the model is like other algorithms which trains the input data until it is close to the desired result. This study concluded that inventory forecasting minimizes the cost of

			manual labor and lets the owners of the stores maintain and regulate the flow of stocks based on their demand.
An Advanced Sales Forecasting Using Machine Learning Algorithm [50]	Sri, B., Ramya, S., & Vedavathi	2020	In this study, it was defined that retail sales contribute a major role in data mining. Data mining refers to the analysis of data in search of patterns for data forecasting. This study used a Machine Learning Algorithm called Extreme Gradient Boosting. The attributes they used are store ID, sales, customer open, keep type, assortment, promo, promo intervals, store, holidays, and distance. The results are that the sales will be high on Sunday but the magnitude of people going to the shops was less likely than the usual day because the stores are closed on Sunday. XGBoost found

			to be the most effective type of Machine Learning in predicting the sales among the Linear Regression and Random Forest Regression.
How to Apply Machine Learning to Demand Forecasting [51]	Molodoria, A., and Taranenko, L.	2020	The article describes how the AI engineers at MobiDev approach the task of ML demand forecasting using machine learning algorithms. This shows that it is also imperative to evaluate data with a set of parameters that are closely related to it. Also, which time series model is the most applicable to the retail sector: ARIMA, SARIMA, and Exponential Smoothing.
Comparative Study on Time Series Forecasting Models [52]	Ben Baccar, Y.	2019	According to this report, several of the most commonly used Time Series estimators have been compared with the in-house estimator developed by SAP and SAPTF, to benchmark their

			performances on a wide variety of series collected from various fields.
Critical Factors Affecting Sales Revenue in Supermarket Chains: A Critical Review Literature [53]	Huong, H. T. and Thi Thanh Binh, V.	2021	The study is theoretical, based on examining past research papers to analyze the elements influencing sales in Vietnamese supermarket systems. The study's findings reveal that factors such as the quality of items, selling prices, the volume of sales, supermarket location, rivals, and marketing practices influence the sales revenue of supermarket systems in Vietnam. The findings of the research serve as the foundation for future empirical studies.

2.5 Demand Forecasting

This part tackles the modeling and sampling strategies that should be considered in the demand forecasting.

2.5.1 Successful Demand Forecasting Modeling Strategies for Increasing Small Retail Medical Supply Profitability

In this study, forecasts are defined as the predictions that can contribute to the increased sales of a certain business. It aims to monitor the inventory flow of a certain business and to determine what will be the future demand. Its purpose is essentially needed for the business to maintain the business goals when it comes to profitability and to become globally competitive. It uses different models with various techniques such as (1) judgmental, (2) econometric, and (3) time series. With the use of time series techniques in forecasting, output can be predicted by statistical models. It also defines that the demand in the future was derived from the previous market history. It also stated that the current demand can also be a future demand. Its research method is mainly qualitative as it states that it can validate the understanding of certain phenomena with multiple input variables. It also used quantitative methods to test the significant differences of the hypothesis and to numerically explain the actual output of the system that was developed. Its target was a small retail medical supply store in Kentucky. It also stated that it used at least 20 respondents for the phenomenological design to further reach the data saturation. Consequently, it also stated that it is better to sample a portion rather than its population size because the goal of the qualitative research is to verify and understand in depth the information that a researcher wants to be evaluated. They used purposeful sampling as the respondents must have at least 5 years of successful experience in demand forecasting as they are the source of in-depth knowledge of a certain study and to be able to reach the data saturation. Data Saturation refers to the point where information is measured to be adequate and sufficient, and it refers to the point where the information is already repetitive and no additional information will be further added. In terms of numerical solving process, it refers to the point wherein it reaches its value where it will be constant. The strategies that were used to increase the profitability are (1) understanding the sale trends, (2) implementing inventory management plan, and (3) seasonality [54].

Table 2.8: Summary of Related Studies on Demand Forecasting

Title	Author/s	Date	Key Features
Successful Demand Forecasting Modeling Strategies for Increasing Small Retail Medical Supply Profitability [54]	Watkins, A.	(n.d.)	This study uses different models with various techniques such as (1) judgmental, (2) econometric, and (3) time series. They used purposeful sampling, and their respondents were at least 20 and have at least 5 years of successful experience in demand forecasting to reach the data saturation. The strategies that were used to increase the profitability are (1) understanding the

			sale trends, (2) implementing inventory management plan and (3) seasonality.

2.6 Effects of Weather on Retail Sales

This part tackles how weather can affect the consumption of the products in the retail stores and how dependent does the retail sales in terms of weather data.

2.6.1 It's the Weather: Quantifying the Impact of Weather on Retail Sales

This study tackles the paucity of research in this area by measuring the link between trading outcomes and various meteorological conditions. This study quantifies the impact of different weather conditions on daily retail sales. The weather is significant in many decision-making processes in the retail sector and is thought to have a significant impact on consumer purchasing behaviors. As a result, many retailers and other stakeholders may benefit from a greater knowledge of the scope and character of the influence of fluctuating UK weather conditions. Results show that wind is consistently determined to be the most important weather condition, and that weather influence is greatest in the summer and spring months. Health foods are revealed to be the most weather-dependent product type among the top five most weather-dependent categories, which include a variety of different product types. The weather has a much more complicated relationship with sales from out-of-town retailers than it does with traditional high street stores, with London and the Southeast seeing the most impact. The scope for additional study is highlighted along with various ramifications of these findings for retail stakeholders [55].

2.6.2 A Study of Weather's Impact on Consumption of Goods

The grocery store industry in Sweden uses various forms of forecast engines for better formulation of marketing strategies for the future demand. Its parameters were based on season, holidays, and campaigns which is defined as the selected products that sold at a discounted price for a period of time. This study wanted to incorporate weather as one of the parameters for their forecast engines. Their scope was to determine the relationship between consumption of the product and weather in the grocery store. The study wanted to validate if the consumption of the product is weather-dependent. It contains 1400 of sales data of the products. They performed regression analysis as part of their statistical analysis to achieve accurate significant results. They gathered the one-year sales data from the ICA Nara and the weather data was gathered through the Swedish Meteorological and Hydrological Institute (SMHI). They obtained the temperature, sunshine hours, wind speed, and precipitation. They assigned the covariates which is used in the regression analysis according to the food category and the parameters of weather data they obtained. They used a Logarithmic Transformation model to evaluate the nine (9) groups of food whether the consumption of it is affected by the weather. Additionally, they built simple models without the covariates of weather to make a comparison with the use of regression analysis. They predict one-week consumption of products, and they compare it with the actual sales within that week. They used the root-mean-square error for validating the accuracy of the prediction model. This study concluded that the prediction model with weather covariates had predicted the consumption rate compared with the simple model. It predicts 60% even better than the simple model. They concluded the possible errors in their study which is mainly composed of weather complications because its weather forecasts have a 45%-84% success rate in determining the weather. Hence, the relationship will be non-linear. They recommend using a non-linear model to deal with this kind of weather complications. The consumption forecast will improve because of incorporating weather as a parameter [56].

Table 2.9: Summary of Related Studies on Effects of Weather on Retail Sales

Title	**Author/s**	**Date**	**Key Features**
It's the Weather: Quantifying the Impact of Weather on Retail Sales [55]	Rose, N., & Dolega, L.	2022	This study measured the link between trading outcomes and various meteorological conditions. It also stated that weather is significant in many decision-making processes in the retail sector. The findings of this study states that wind is consistently determined to be the most important weather condition. Health foods are revealed to be the

			most weather-dependent product type among the top five most weather-dependent categories.
A Study of Weather's Impact on Consumption of Goods [56]	Krylstedt, J., & Weidlertz, A.	(n.d.)	This study was based on the parameters such as season, holidays, and campaigns. The study wanted to validate if the consumption of the product is weather-dependent. They used a Logarithmic Transformation model and simple model to find out the relationship of the covariates. They used sales and weather data for the covariates in the regression model. This study revealed that incorporating weather as a parameter, it can predict 60% better than a simple model.

2.7 Inventory Management

This section discusses the inventory control management.

2.7.1 Inventory Control Management and Repeat Purchase in Policy Reformation

In this study, it used random sampling to enable equal chances for the participants to get to participate in the conducted study. This will enable unbiased data. Additionally, the proposed solution was to have a warehouse check to identify the purchasing power of the products. Also, this study concluded that the free trading marketing department should be responsible for the number of items displayed in the shelves to maximize the rented spaces of the product brands [57].

Table 2.10: Summary of Related Studies on Inventory Management

Title	Author/s	Date	Key Features
Implementation of Inventory Control Management and Repeat Purchase in Right Goods Philippines Incorporated: Inputs to Policy Reformulation [57]	Rosario, J.M.	2022	This study focused on inventory control management. The proposed solution based on the result of his study was to have a warehouse check to identify the purchasing power of the products. Random sampling and survey-questionnaires to assess the relationship between the implementation of

			inventory control management and repeat power purchase of RPI.

2.8 Barcode Scanner

2.8.1 Process of Barcode Scanning

In this study, a barcode scanner is commonly called a POS scanner or price scanner. It is a stationary device that is used to read the information of the item inside the barcode. The barcode scanner is connected into the ports such as serial port or keyboard port to work. It can also be connected in an interface called wedge. The information read by the barcode scanner is translated through the use of software applications. The barcode scanner works by directing the beam of light against the barcode and reading the pattern of light that is being reflected. The barcode scanner converts the light energy into electrical energy. This electrical energy is converted into data by the decoder and sends the converted data into the computer [58].

2.8.2 Information Content of Barcode and Its Types

In this article, barcode is defined as machine-readable code that is printed in items with printed numbers and varying width of patterned parallel lines that is black and white. It contains encoded information that the machine only can decode. The parallel lines of black and white has a patterned algorithm to encode the specific information. Barcode contains the specific information such as product name, product specifications (size, color, type, etc.), production time and date, production location, warehouse location, and operator who produced the item. Quiet zone is the area in the barcode that is essential in reading a barcode. This zone acts as a blank space in the left and right of the barcode. Additionally, this space enables the barcode reader to determine the starting point and ending point of the barcode. The barcode would not be identifiable without this space. There are only two types of barcodes namely 1-D and 2-D barcode. 1-D consists of varying widths of vertical lines. It consists of alpha-numeric and numeric data only. It is most found in retail markets as the information in this type of barcode is only limited. The most common type of barcode is the UPC (Universal Product Code). It has 12 numerical digits. The first digit contains the standard number system character. It identifies the product category. The first set of 5 digits pertains to the product's manufacturer. It remains the same on each item because it is specific for the company. The second set of 5-digit numbers pertains to the classification of the products. The last digit is used for the error detection scheme of the barcode. The 2-D barcode can store the information in vertical and horizontal dimensions. Unlike the 1-D barcode, it has dots, squares, and geometric patterns. Examples of this are QR code, data matrix, PDF 417, Maxicode, and Aztec. It holds hundreds of characters compared with 1-D barcode [59].

Table 2.11: Summary of Related Studies on Barcode Scanner

Title	Author/s	Date	Key Features
What is a Barcode Reader (POS	Hanna, K. T.	2022	This study focused on the process of barcode scanning. It briefly explains the

scanner, Barcode Scanner, Price Scanner)? [58]			main concept of the process behind barcode scanning through the use of barcode scanner.
Barcode 101: Information You Need To Know [59]	D'Altorio, A	2022	This study focused on the components of the barcode and its types. It explains the information content of the barcode.

Chapter 3

METHODOLOGY

This chapter presents all the detailed steps and procedures to accomplish the research objectives of this development study.

3.1 Research Design

This study utilized descriptive research design. Descriptive research is the process of gathering and analyzing data about prevailing conditions and processes [60] while developmental research is the systematic study of designing, developing, and evaluating instructional programs, processes, and products that must meet criteria of internal consistency and effectiveness [61].

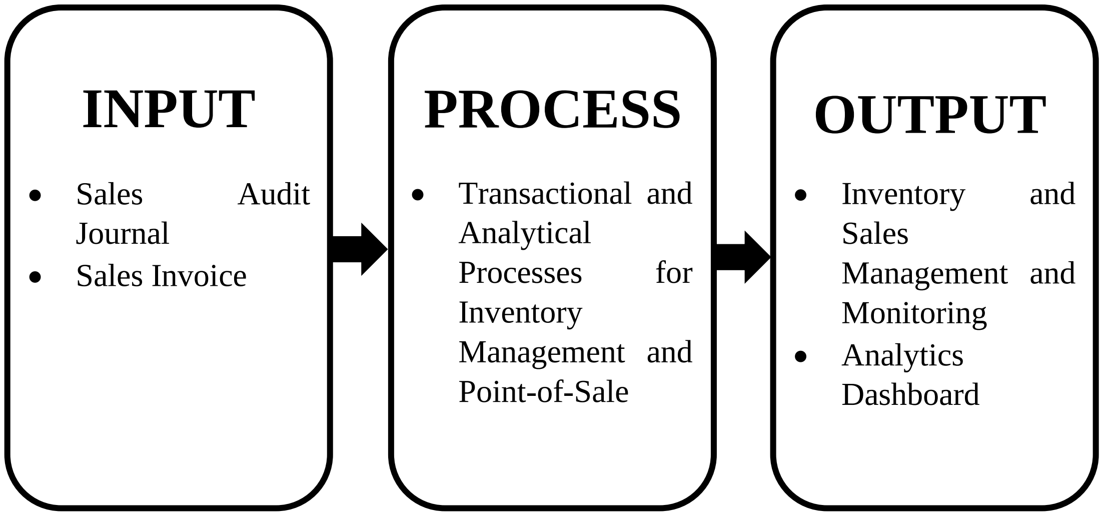

INPUT
Sales Audit Journal
Sales Invoice
PROCESS
Transactional and Analytical Processes for Inventory Management and Point-of-Sale
OUTPUT
Inventory and Sales Management and Monitoring
Analytics Dashboard

Figure 3.1: Input-Process-Output

The figure shown above illustrates the Input-Process-Output of the study. Data from confirmation in sales from the mini grocery store as input parameters are stored in a database which allows Transactional and Analytical processes. The Transactional processes can monitor and audit sales transactions while Analytical processes focus on predictive analytics of sales and demand based on the sales data using Machine Learning Algorithms which can be displayed in the mobile application.

3.2 Research Process Flow

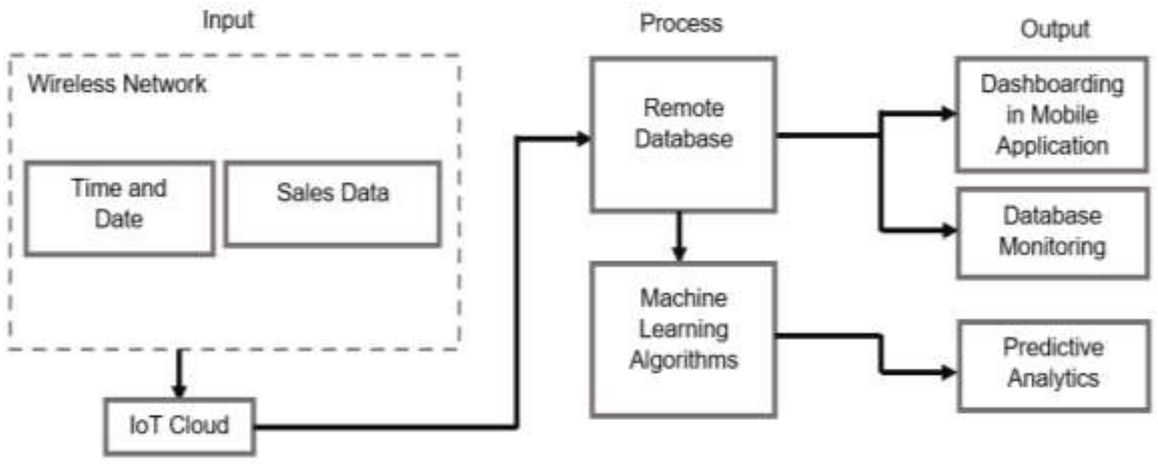


Figure 3.2: Block Diagram

Figure 3.2 shows where system starts with the implementation of a wireless network which uses:

(1) Time and date when the sales transaction occurs

(2) Sales transaction data using the POS system

As input parameters for the machine learning algorithm to forecast demand and sales. After the collection of data from the sales using the POS system, it is stored in a remote database where it can be accessed by a program to conduct machine learning algorithms for sales and demand forecasting. The results were now output through the mobile application viewable by the user. In addition, the mobile application also includes features such as sales transaction report and stock monitoring for the entire retail product inventory which can be remotely configured when the user decides to change parameters such as price and quantity.

3.3 Development of a Mobile Application using the Dart Language and Flutter Framework

This section of the paper details the steps on the development of the mobile application using dart and flutter framework. The core features of the application should be:

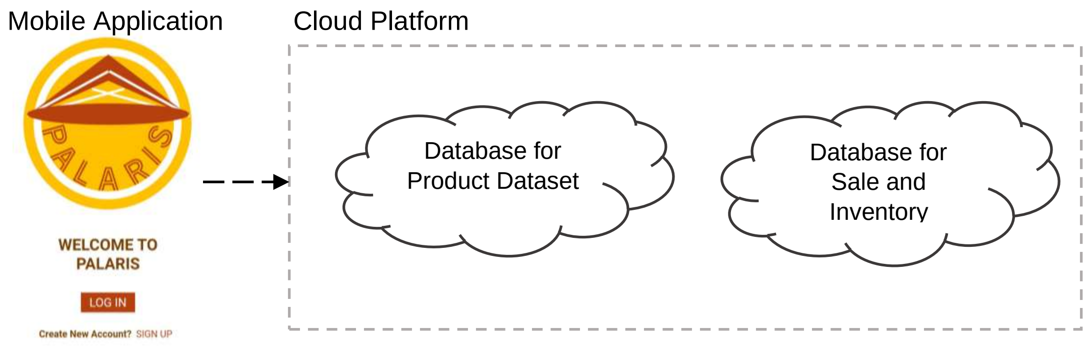

Mobile Application
Cloud Platform
PALARIS
WELCOME TO PALARIS
LOG IN
Create New Account? SIGN UP
Database for Product Dataset
Database for Sale and Inventory

Figure 3.3: Mobile Application to Cloud Platform

Figure 3.3 shows the establishment of the connection between the mobile application, Palaris, and the cloud platform that stores and manages the database. The cloud platform serves as a cloud storage that handles and manages all the information of the products, sales, and inventory.

3.3.1 Key Features of the Mobile Application

This section highlights the main features of the mobile application in terms of its functionality.

1. Track transactions and generate sales report.
2. Management of inventory and point-of-sale

3.3.2 User Interface and Backend Design

This section presents the procedures for creating and establishing the connection between the user interface and the backend design of the mobile application.

1. Conceptualization of a clean UI/UX
2. Adding necessary page interfaces:
    - Login
    - Homepage
    - Business Intelligence
        - Data Analytics
        - Audit Tracking
        - Report Generation
    - Inventory and Sales Management
        - Product info page
    - Barcode Scanner and Barcode Generator
    - Logout
3. Programming

3.3.3 Interfacing with the Cloud Database

This part enumerates the steps taken to establish a connection between the mobile application and the cloud database service as well as the steps on the backend development.

Key Features:

Provide a database for the sales transactions and basic information of the products such as item name, number of stocks and selling price.

1. Reading the documentation of the database service
2. Generating a private key for client to server connection
3. Programming and establishing the interface.
4. Add necessary features with the database (Stock & Price manipulation etc.)
5. Feature testing

3.3.4 Data Dashboard

By displaying beneficial charts and graphs helped the retail store owners to identify and monitor the performance of their business, make key decisions, establish their local Key Performance Indicators etc. mainly to provide business intelligence in an interactive and digestible way.

1. Identify necessary charts.
2. Display charts
3. Pipe necessary information from database

3.4 Development of a Sales and Demand Prediction Model using Machine Learning Algorithms SARIMA and Prophet

The methodology employed in this study aimed to develop a robust sales and demand prediction model using machine learning algorithms, specifically the SARIMA and Prophet algorithms known for their effectiveness in time series forecasting. The goal was to leverage historical sales and demand data to capture inherent patterns and seasonality, facilitating accurate predictions of future sales trends. This section presents a detailed overview of the steps involved in developing the sales and demand prediction model, including data preparation, model identification, parameter estimation, model fitting, and evaluation. The objective is to contribute to the field of sales forecasting by providing a comprehensive framework for reliable prediction, assisting businesses in making informed decisions based on anticipated sales and demand.

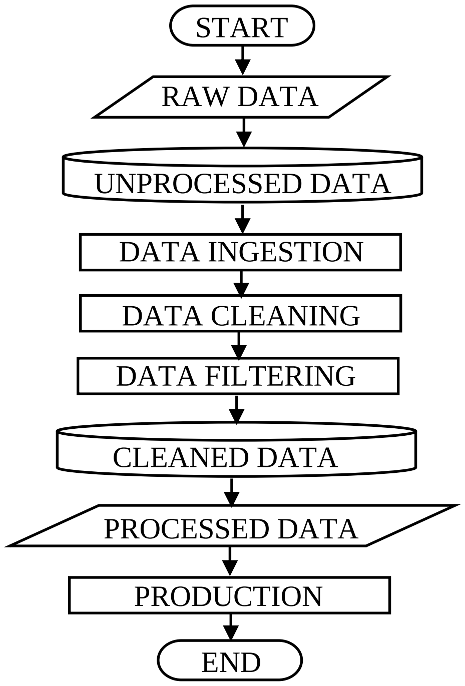


Figure 3.4: Process Flow Diagram for Time Series Modeling

Figure 3.4 illustrates the step-by-step process flow for data processing in the development of the sales and demand prediction model. The unprocessed transaction data is stored in a cloud database, and it is ingested to compute a unit which enables the data to be processed. The first stage of data processing involves ingesting the unprocessed data from the cloud storage. This ingestion should be done at a regular interval to keep the data updated. Once data is ingested to compute a unit, it undergoes cleaning where error values are either imputed or filtered. This process enhances the integrity and the accuracy of data to ensure valid and reliable information is used. After preprocessing the data, it is properly labeled to a big dataset which is used to train a model. It is important that the data is properly formatted to the standards of the machine learning algorithm to clear possible errors. This cleaned data is not uploaded to the cloud database but kept locally in spreadsheets for monitoring and is now ready for modelling.

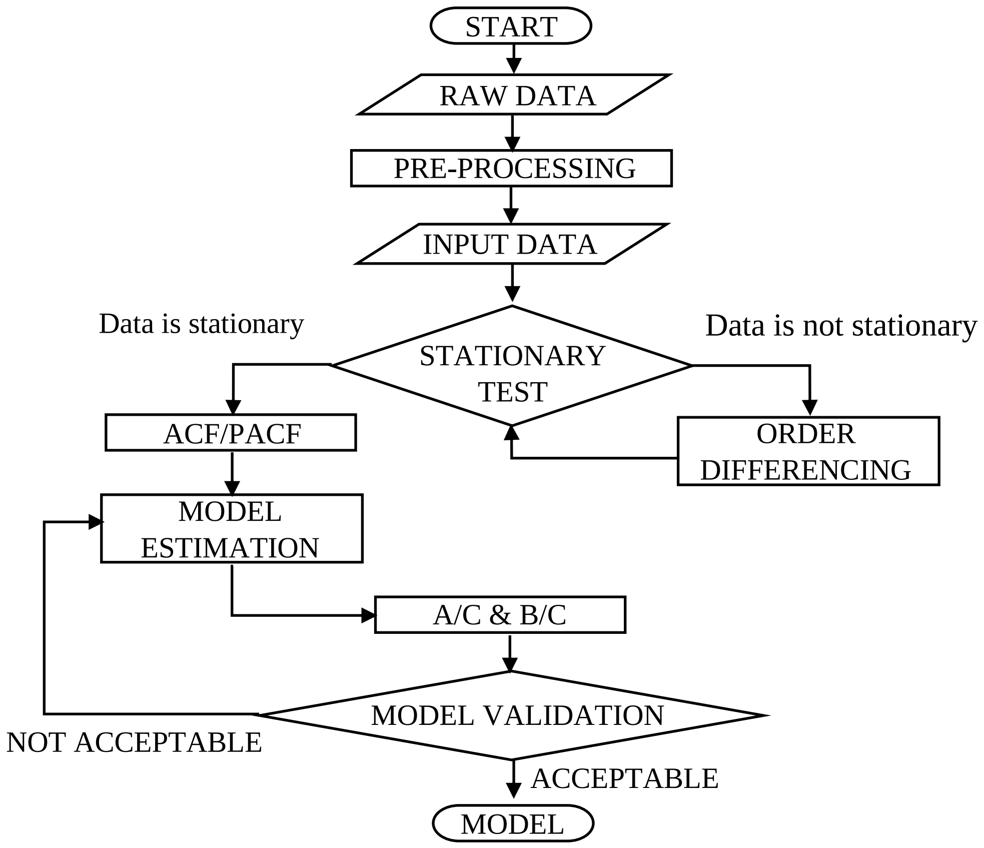


Figure 3.5: Process Flow Diagram

Figure 3.5 shows the process flow diagram of how time-series sales and demand forecasting models were developed. The process starts with gathering raw data, such as historical sales data. Then, it undergoes the pre-processing of data to visualize, manage and filter easily the raw data. This pre-processed data is the input data for the training of the time series model. The input data undergoes the process of stationary testing. This test determines if the data already has constant variances and error means. If the data passes this test, it undergoes ACF/PACF and model estimation. However, if it does not pass the test, the data undergoes order differencing. After the model estimation, the data is split-tested, and the time-series model is validated if it has acceptable error means and accuracy.

3.4.1 Univariate Approach

This study used the univariate time forecasting approach to create a prediction model for sales forecasting. Using the scikit-learn modeling toolbox to create a SARIMA model to predict future retail sales. The modeling process begins with unprocessed raw data that is subjected to data cleaning. It is essential to keep the dataset clean from null values while considering the zero values in the dataset. Establishing a base interpretation of zero values as valid or invalid data. In this study, zeroes are valid data since it is firstly possible for stores to have no sales in a day. After the data is cleaned, it is now workable input data. It is formatted to a time series data frame using the Pandas library. In addition, the time indexes are formatted in datetime before being fed to the modeling algorithm. Using the Augmented Dickey-Fuller Test (ADF), the dataset is then evaluated for stationarity. Non-stationary data is then passed through an order differencing function to make it stationary. This is done by shifting the columns by one. Another differencing method is the log transformation widely used in time series data. Once the dataset passes for stationarity, it passes through a function that evaluates the base orders for three parameters the model requires. P is the number of autoregressive terms, D is the number of non-seasonal differences, and Q is the number of lagged forecast errors in the prediction equation. Newer implementations of ARIMA, such as the auto ARIMA employ an algorithm to automate the process of deciding the value of these parameters. Lastly, using Akaike’s Information Criterion, the Bayesian Information Criterion, and other forecast metrics, to evaluate model performance. Hyperparameter tuning was be used to optimize the model, while Cross-validation techniques was used to validate the model.

3.4.2 Multivariate Approach

Like the idea of the univariate modeling approach is the multivariate approach, this method is used for machine learning problems that require other variables that can affect the predictive values. Implementing the multivariate approach relies heavily on VAR (Vector Auto Regression), which is the base method for creating a model that accounts for other factors. However, existing libraries have allowed the multivariate approach to be less complicated. Prophet is a model developed by Facebook which can easily manage multivariate forecasting. One feature of Prophet allows it to define specific dates as holidays and add parameters called regressors, which also account for changes in other variables. However, regressors are not necessarily the same as VAR while both having the same capacity to expand their dimensionality. Regressors are values that the model expects to affect the data. Parameters are expected to follow a framework and correlate significantly to the predictor. VAR, on the other hand, models using multiple predictors. Both methods are valid multivariate approaches that were considered during this project

3.4.3 Data

This section shows the guide to regressor selection, the competing values framework was used.

3.4.3.1 Data Parameters

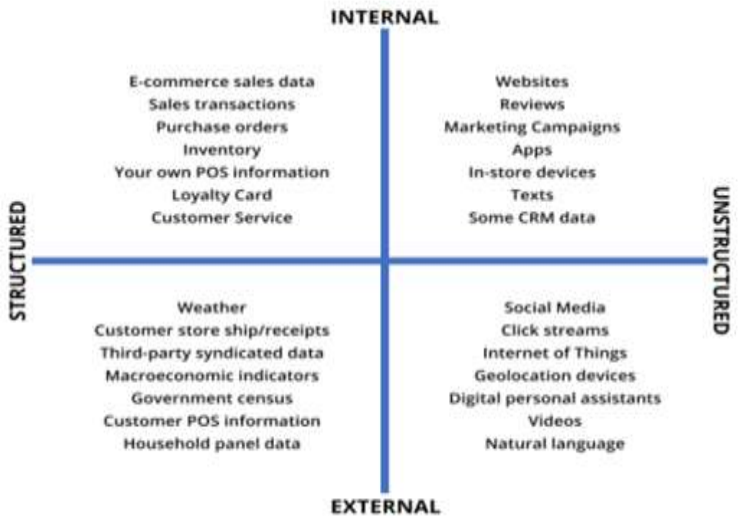


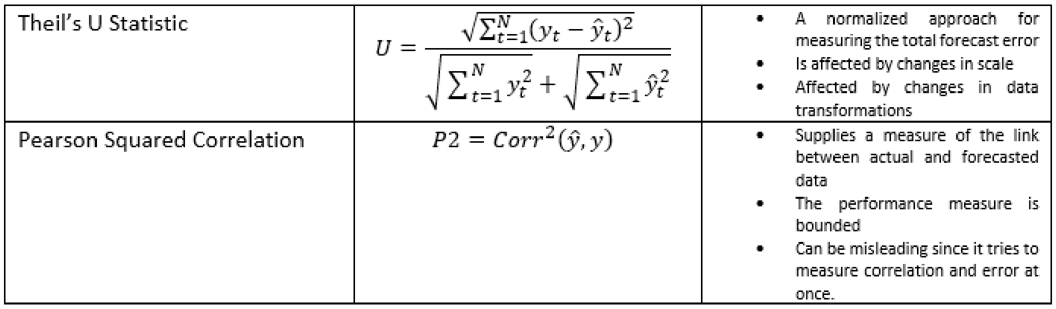
Figure 2.1: Competing Values Framework [51]

The competing values framework serves as a guide to this study in choosing necessary parameters as the regressors for the model. Macroeconomic indicators and climatological data as such can be factors that decide the sales of retail goods. Aside from these exogenous variables data from the POS system and inventory data can also be regressors for the model.

3.4.4 Validation

Cross-validation and splitting of datasets were used to test and train. This aided in evaluating the overall model performance. In addition, performing hyperparameter tunings to tweak the model parameters and computing for Akaike's Information Criterion (AIC) and Bayesian Information Criterion (BIC) can helped figuring out the best-fit model for the study among numerous models since no model is one size fits. A study comparing various time series forecasting approaches suggested practicing using three metrics in scoring a model to fully understand it as different metrics result in different insights.

Table 3.1: Forecast Metrics Summary Table [52]



Theil's U Statistic	$U = \frac{\sqrt{\sum_{t=1}^{N}(y_t - \hat{y}_t)^2}}{\sqrt{\sum_{t=1}^{N} y_t^2} + \sqrt{\sum_{t=1}^{N} \hat{y}_t^2}}$	• A normalized approach for measuring the total forecast error • Is affected by changes in scale • Affected by changes in data transformations
Pearson Squared Correlation	$P2 = Corr^2(\hat{y}, y)$	• Supplies a measure of the link between actual and forecasted data • The performance measure is bounded • Can be misleading since it tries to measure correlation and error at once.

3.5 Statistical Test

This study used a two-sample t-test to compare the actual sales data and predicted sales and demand generated from the developed sales and demand forecasting model and actual sales data. This statistical test determined if the actual and predicted values have significant differences. Additionally, it used weighted means to determine the efficiency and functionality of the mobile POS application to the store owner.

3.6 Deployment

3.6.1 Locale of the Project Study

The location of the deployment was held at Triple J store that was specifically located at 2096 Rosal Street, Batasan Hills, Quezon City.

3.6.2 Project Study Instrument

The proponents used a survey questionnaire to gather data from the store owner to determine the overall user experience and functionality of the developed mobile POS application to the store owner. The survey questionnaire has three parts: System Functionality, Hardware Functionality, and Predictive Model. Moreover, the proponents used the Likert scale to quantitatively assess the experience of the store owner of using the mobile POS application.

3.6.3 Data Gathering Procedure

The following are the data gathering procedure of the proponents,

1. The proponents gave a Non-Disclosure Agreement (NDA) to the chosen deployment area and explained the details of the NDA to the store owner.
2. The mobile POS application was installed in a handheld POS during the deployment phase to have convenience in barcode scanning and generation of sales transaction receipts.
3. The proponents of the study introduced and demonstrated to the user the main features and flow of the mobile POS application. Continuously, the proponents let the store owner to use the developed mobile POS application for one week to test the accuracy of the predicted sales and demand generated from the developed sales and demand forecasting models.
4. The proponents monitored the database remotely of the sales transactions of the store.

5. The proponents distributed survey questionnaire forms daily in the form of a Likert scale to the deployment research locale to rate and evaluate the daily performance of the mobile POS application.
6. After a week, the actual sales of the store are gathered and compared with the predicted sales and demand of the store.
7. The proponents tallied the gathered responses from the survey questionnaire forms to prepare for the statistical data analysis.

3.6.4 Data Analysis

The proponents used two-sample t-test to compare the means of the predicted and actual sales and demand of the locale of the project study. This statistical test determined whether the predicted and actual sales and demand differ significantly. Additionally, the gathered data from the survey questionnaire was computed using weighted means to determine the overall functionality of the developed mobile POS application. The results of the weighted means were determined by using the verbal interpretation table of the weighted means. The verbal interpretation interval was determined by subtracting the highest scale from the lower scale and dividing it by the highest scale. The quotient was added to the lowest scale to form the first interval. One was added to the first interval's end value to determine the succeeding interval's first value.

3.7 Project Work Plan

Project Workplans for the proponents is shown in the table below.

Tasks		2022						2023					
		July	**August**	**September**	**October**	**November**	**December**	**January**	**February**	**March**	**April**	**May**	**June**
PHASE 0: DESIGN	Learning the Programming Language and its Development Environment	🟧	🟧										
	Conceptualizing UI Design and Layout of the Mobile Application	🟧											
	Choosing Database Management Solution	🟧											
PHASE 1: MOBILE APPLICATION FOCUS	Programming the UI/ UX		🟧	🟧	🟧	🟧	🟧						
	Programming Necessary Features for the Mobile Application		🟧	🟧	🟧	🟧	🟧						
	Mobile Application Interfacing with Database			🟧	🟧	🟧	🟧						
	Debugging						🟧						
	Optimization						🟧						
PHASE 2: PRE-DEPLOYMENT	Designing the predictive Model							🟧	🟧				
	Data Gathering							🟧	🟧				
	Modelling using SARIMA and prophet							🟧	🟧	🟧			
	Data Preprocessing								🟧	🟧			
	Model Training									🟧	🟧		
	Model Testing									🟧	🟧		
	Optimization and Tuning									🟧	🟧		
	Model Evaluation									🟧	🟧		
	Procure thermal printer and barcode scanner available in the market									🟧	🟧		
	Establish a connection between mobile application and hardware.									🟧	🟧		
	Test and print necessary parameters in receipt.									🟧	🟧		
PHASE 3: DEPLOYMENT	Project Deployment and Monitoring										🟧	🟧	🟧
	Creation of Survey Materials											🟧	
	Verification and Validation of Survey Instrument											🟧	
	Distribution of Survey Forms											🟧	🟧
	Technical Evaluation												🟧
	Gathering of Responses												🟧
	Conduction of Statistical Tests												🟧
	Interpretation of Data												🟧
	End of Project Study												🟧

Chapter 4

DATA AND RESULTS

This chapter presents the interpretation of gathered data and analysis of the results based on the tests conducted.

4.1 Project Technical Description

The project study entitled “Palaris: A Mobile Point-of-Sale System Solution for Inventory Management of MSMEs with Time-series Forecasting Using SARIMA and Prophet” is a mobile POS application that offers digitalization of inventory that eliminates manual inventory. Also, this mobile POS application uses time-series forecasting that enables it to predict the future sales and demand of the mini grocery store. Additionally, it offers the generation of barcodes to the products with no barcode, such as items packed in small packaging and real-time updates of sales transaction databases through a wireless network.

SARIMA and Prophet were machine learning models used to develop the predictive analytics capability of the mobile POS application. During the initial phase, the sales and demand prediction model used historical data of the chosen deployment mini grocery store to form the prediction model. The prediction model took several calibrations to form accurate sales and demand predictions. The model has an overall 76.1% accuracy.

The developed mobile POS application enables the user to add items to the inventory and generate barcodes for the items packed in small packaging to store them in the POS. Continuously, the summary of the customer's sales transaction will be displayed after it is completed and have the option to generate a receipt after the transaction is settled.

4.2 Project Organizational Structure

This section shows how the UI/UX establishes its connection with the other core functionalities of the mobile application, Palaris. It also presents the detailed process of how Palaris works.

4.2.1 User Interface

This section shows the overall visualization of the Palaris mobile application.

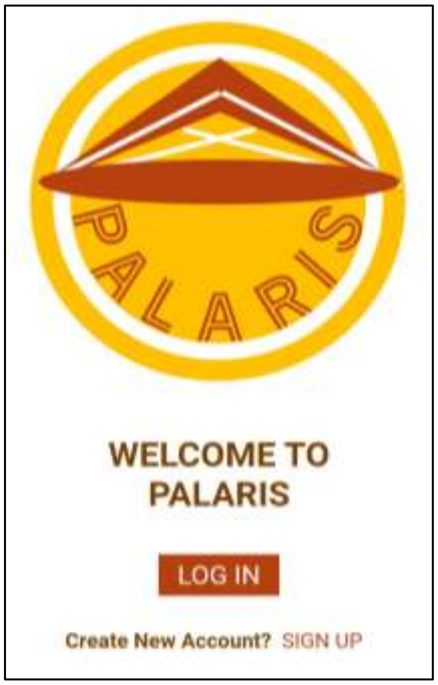
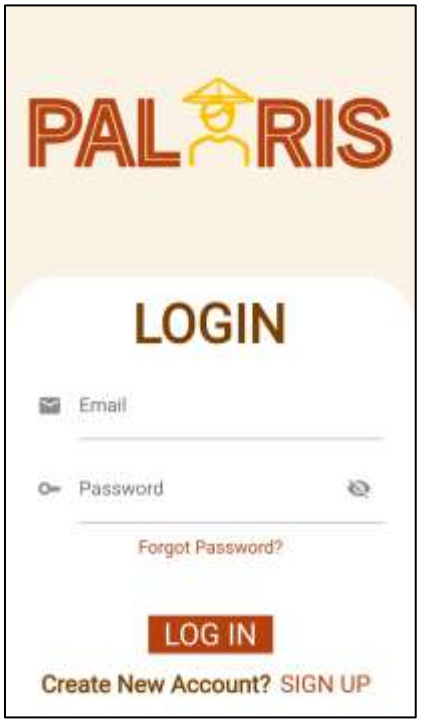
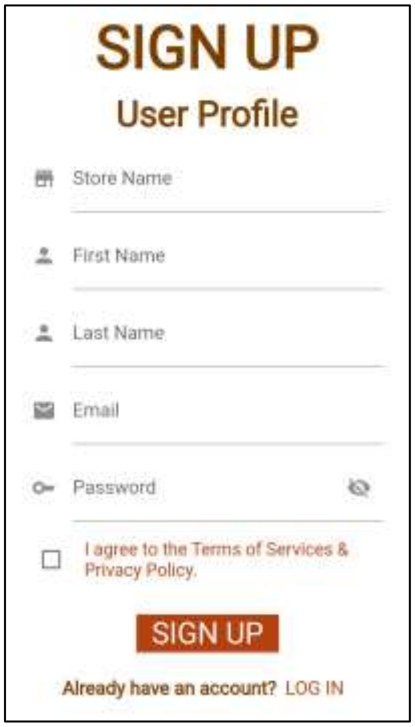


Figure 4.1: Palaris Welcome Page

Figure 4.1 is the Palaris Welcome Page user interface, where the main logo, Log In button, and Sign Up is found. From here, the user can either log in with an existing account or sign up as a new user.



(a)

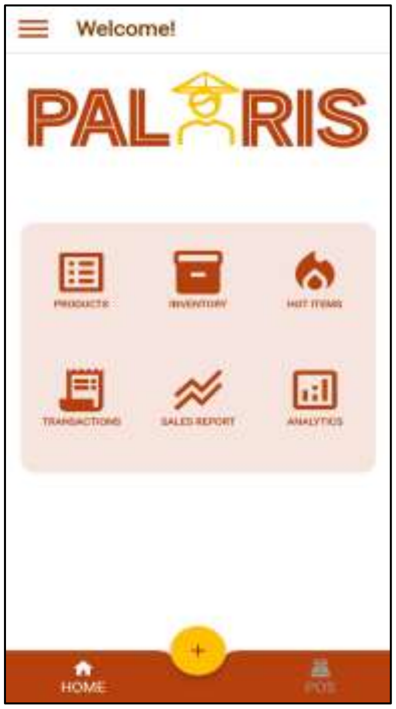

SIGN UP
User Profile
Store Name
First Name
Last Name
Email
Password
I agree to the Terms of Services & Privacy Policy.
SIGN UP
Already have an account? LOG IN

(b)

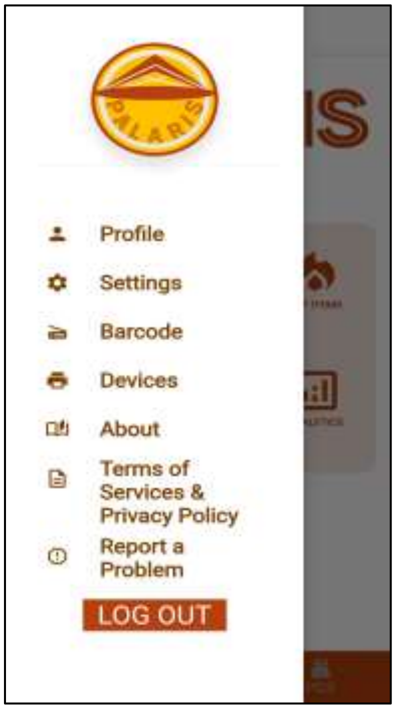
Figure 4.2: Palris User Verifier Page:

(a) Palaris Login; (b)Sign-Up Page

Figure 4.2 is the Palaris User Verifier Page, where users can enter their credentials and read the Terms of Services & Privacy Policy. The Login Page serves to authenticate the user's identity to protect all the information and sales data of the user. On the other hand, the Sign-Up Page has unique identifier fields to ensure the accuracy of the information for the users.



(a)

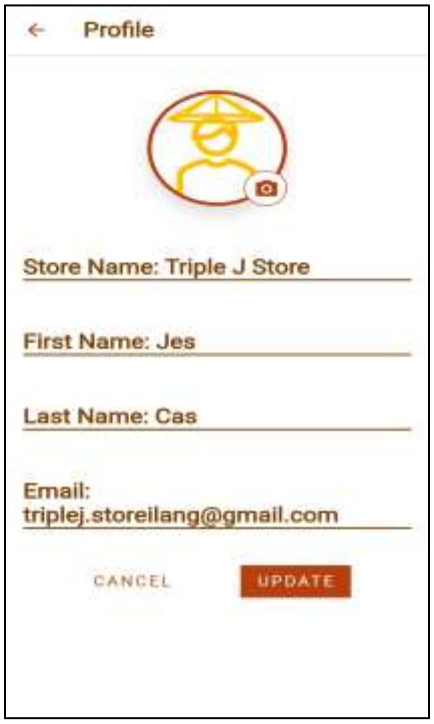


(b)

Figure 4.3: Palaris Home Page:

(a) Home Page with Features; (b) Drawer Menu at Home Page

Figure 4.3 shows the main home page of the Palaris; this is where the main functions of the Palaris are found: Products, Inventory, Hot Items, Transactions, Sales Report, Analytics, Add Product (plus button), and POS. At the upper left is where the Drawer Menu is located.


Profile

Store Name: Triple J Store

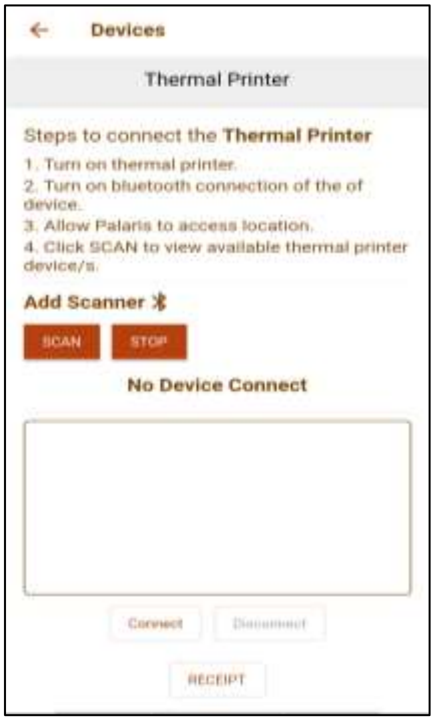
First Name: Jes

Last Name: Cas

Email:
triplej.storeilang@gmail.com

Figure 4.4: Palaris Profile Page

Figure 4.4 shows the Palaris Profile Page user interface, and it shows the personal details of the user, such as the Store Name, First Name, Last Name, and email used in creating the account. The user can update their details and upload a profile picture.


Devices

Thermal Printer

Steps to connect the **Thermal Printer**

1. Turn on thermal printer.
2. Turn on bluetooth connection of the of device.
3. Allow Palaris to access location.
4. Click SCAN to view available thermal printer device/s.

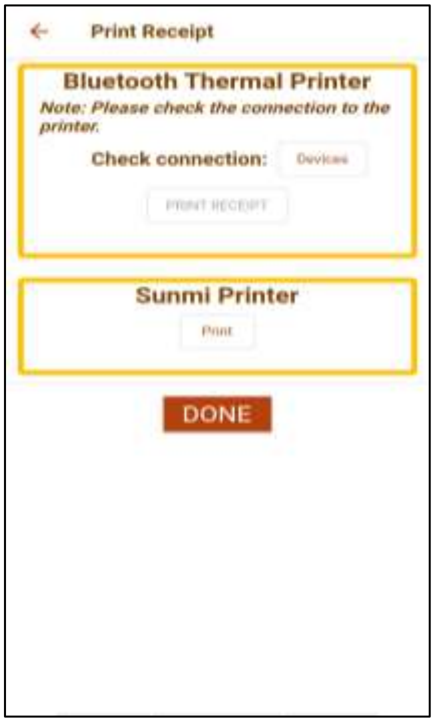
Add Scanner

SCAN STOP

No Device Connect

Connect Disconnect

RECEIPT


Print Receipt

Bluetooth Thermal Printer

Note: Please check the connection to the printer.

Check connection: Devices

PRINT RECEIPT

Sunmi Printer

Print

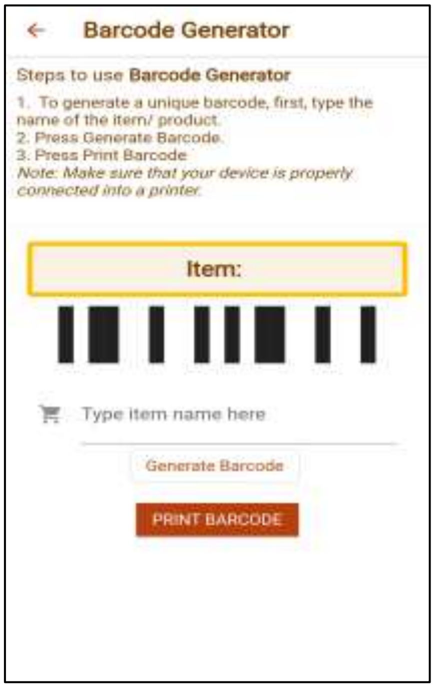
DONE

(a) (b)

Figure 4.5: Thermal Printer Compatibility with Instructions Page:

(a) Thermal Printer Compatibility; (b) Attached Handheld POS Printer Compatibility

Figure 4.5 shows the compatibility of the thermal printer; it also has instructions on how to use it for better understanding. Here, the user can perceive the available devices for thermal printers. The compatibility for thermal printer works via 2.4GHZ Bluetooth connection.


Barcode Generator

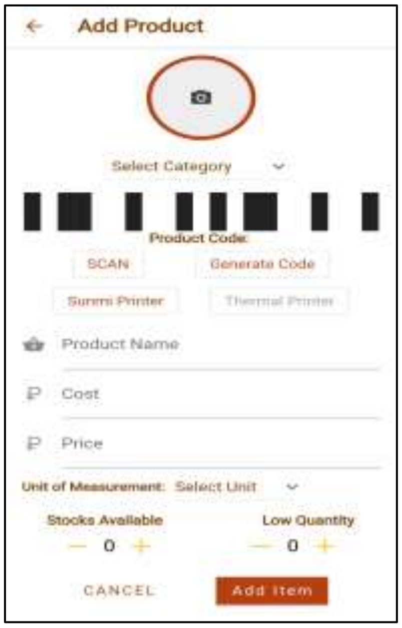
Steps to use **Barcode Generator**

1. To generate a unique barcode, first, type the name of the item/ product.
2. Press Generate Barcode.
3. Press Print Barcode

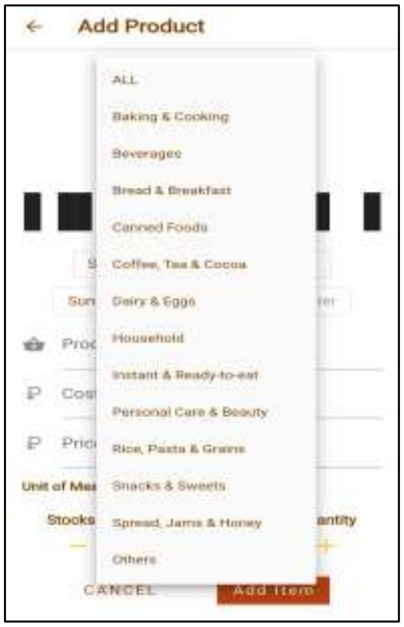
Note: Make sure that your device is properly connected into a printer.

Figure 4.6: Barcode Generator with Instructions Page

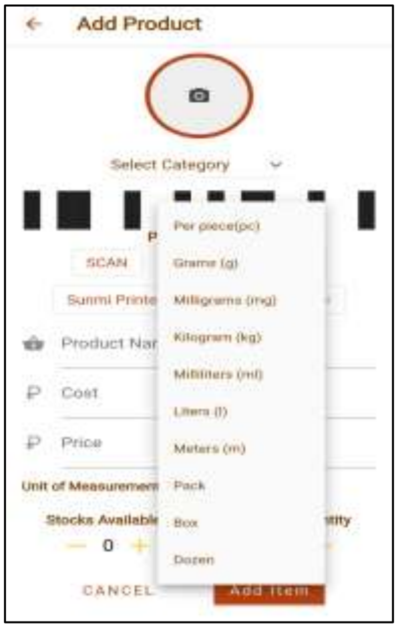
Figure 4.6 shows the Barcode Generator with instructions. Here, the user can create their unique barcode for their codeless products. The barcode generator consists of nine (9) randomized numbers. The user can also input a product together with the product name they desire to register and print the barcode for their personal copy.



(a)



(b)

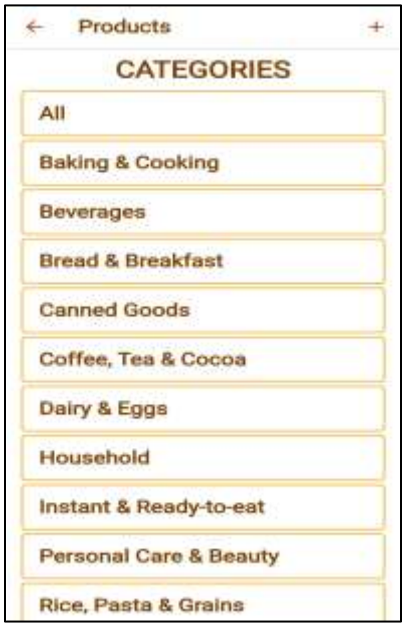


(c)

Figure 4.7: Add Products Page:

(a) *User Interface of Add Product; (b) Categories of Products; (c) Units Available*

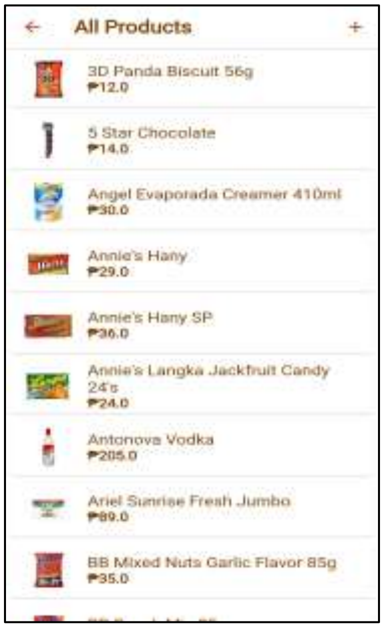
Figure 4.7 shows the Add Products Page, this is where the user can register their products.


Products

CATEGORIES

- All
- Baking & Cooking
- Beverages
- Bread & Breakfast
- Canned Goods
- Coffee, Tea & Cocoa
- Dairy & Eggs
- Household
- Instant & Ready-to-eat
- Personal Care & Beauty
- Rice, Pasta & Grains

(a)


All Products

3D Panda Biscuit 56g
₱12.0

5 Star Chocolate
₱14.0

Angel Evaporada Creamer 410ml
₱30.0

Annie's Hany
₱29.0

Annie's Hany SP
₱36.0

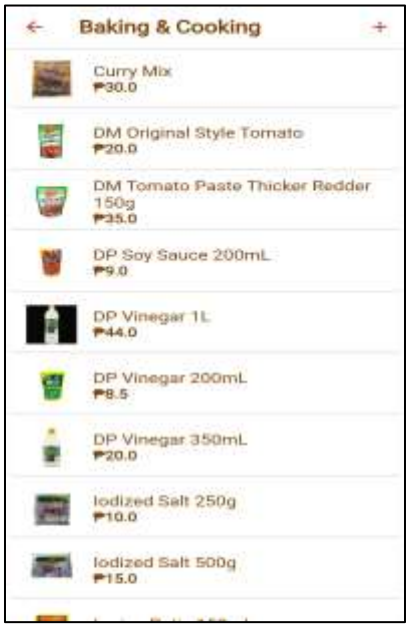
Annie's Langka Jackfruit Candy 24's
₱24.0

Antonova Vodka
₱205.0

Ariel Sunrise Fresh Jumbo
₱89.0

BB Mixed Nuts Garlic Flavor 85g
₱35.0

(b)


Baking & Cooking

Curry Mix
₱30.0

DM Original Style Tomato
₱20.0

DM Tomato Paste Thicker Redder 150g
₱35.0

DP Soy Sauce 200mL
₱9.0

DP Vinegar 1L
₱44.0

DP Vinegar 200mL
₱8.5

DP Vinegar 350mL
₱20.0

Iodized Salt 250g
₱10.0

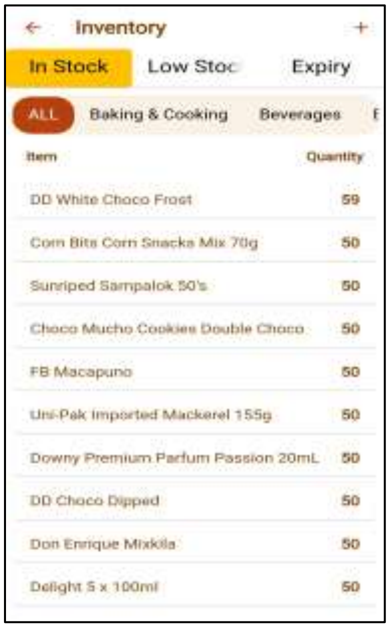
Iodized Salt 500g
₱15.0

(c)

Figure 4.8: Products Page:

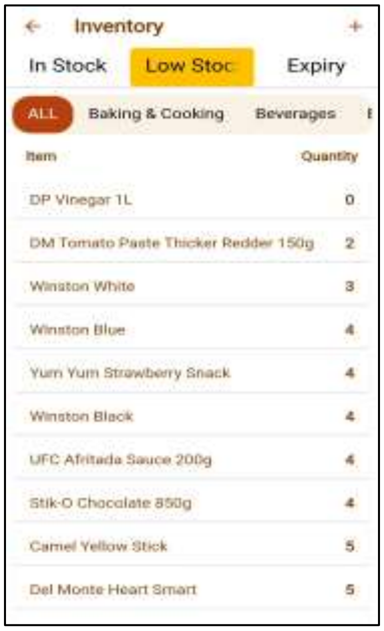
(a) *Categories; (b) All Products; (c) Sample List of Items in a Category*

Figure 4.8 shows the user interface of all the products with categories in Palaris.



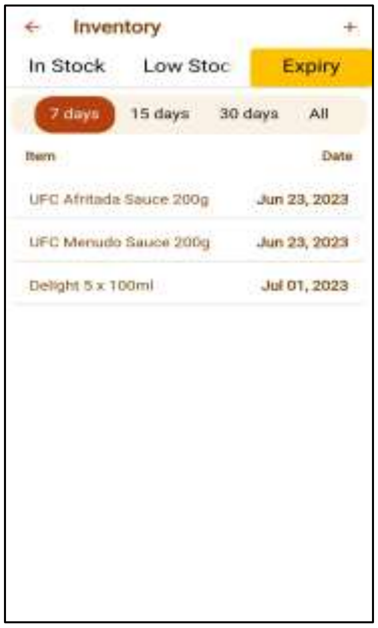
Item	Quantity
DD White Choco Frost	59
Corn Bits Corn Snacks Mix 70g	50
Sunriped Sampalok 50's	50
Choco Mucho Cookies Double Choco	50
FB Macapuno	50
Uni-Pak Imported Mackerel 155g	50
Downy Premium Parfum Passion 20mL	50
DD Choco Dipped	50
Don Enrique Mixkila	50
Delight 5 x 100ml	50

(a)



Item	Quantity
DP Vinegar 1L	0
DM Tomato Paste Thicker Redder 150g	2
Winston White	3
Winston Blue	4
Yum Yum Strawberry Snack	4
Winston Black	4
UFC Afritada Sauce 200g	4
Stik-O Chocolate 850g	4
Camel Yellow Stick	5
Del Monte Heart Smart	5

(b)



Item	Date
UFC Afritada Sauce 200g	Jun 23, 2023
UFC Menudo Sauce 200g	Jun 23, 2023
Delight 5 x 100ml	Jul 01, 2023

(c)

Figure 4.9: Inventory Page:

(a) In Stock Page; (b) Low Stock Page; (c) Expiry Page

Figure 4.9 shows Inventory's user interface, which is divided into three parts: In Stock, Low Stock, and Expiry. On the In Stock Page, the user can view the number of available items for each product, arranged from highest to lowest. The Low Stock Page is where the user can view the items number of items that are low in stock; this is arranged from lowest to highest. Lastly, on the Expiry Page, the user can view the expiration date of the products. The expiration date of the items is viewed as seven (7) days, fifteen (15) days, or thirty (30) days before the expiration date.

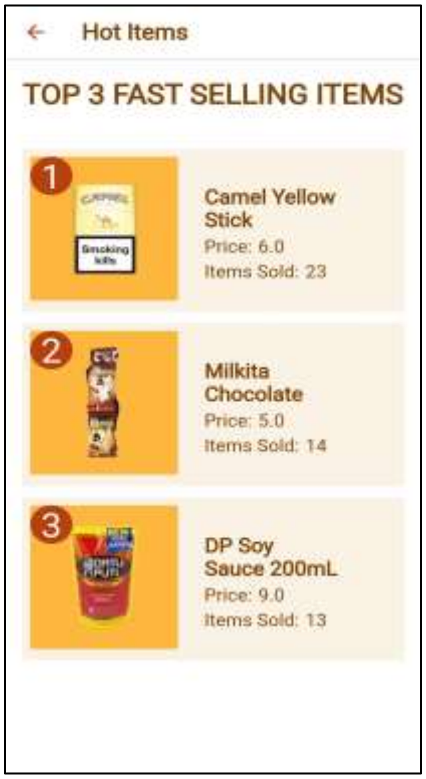
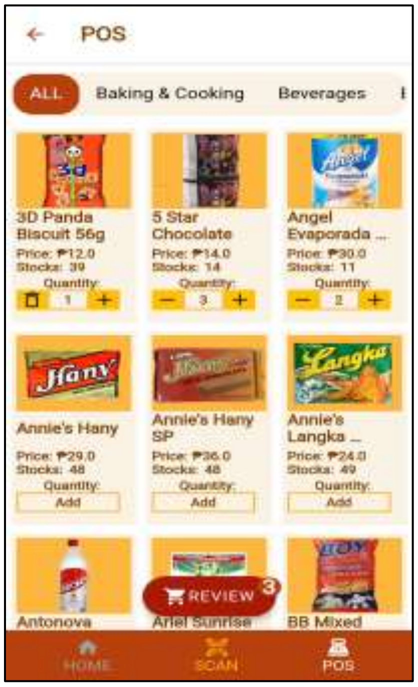


Figure 4.10: Hot Items Page

Figure 4.10 shows the user interface of the store's Top 3 Fast Selling Items. It shows the Product Name, Price, and Items Sold. This page guides the user on what items should be prioritized to buy during the restocking period to earn more profit and purchase from the supplier.



3D Panda Biscuit 56g
Price: ₱12.0
Stocks: 39
Quantity: 1

5 Star Chocolate
Price: ₱14.0
Stocks: 14
Quantity: 3

Angel Evaporada ...
Price: ₱30.0
Stocks: 11
Quantity: 2

Annie's Hany
Price: ₱29.0
Stocks: 48
Quantity: Add

Annie's Hany SP
Price: ₱36.0
Stocks: 48
Quantity: Add

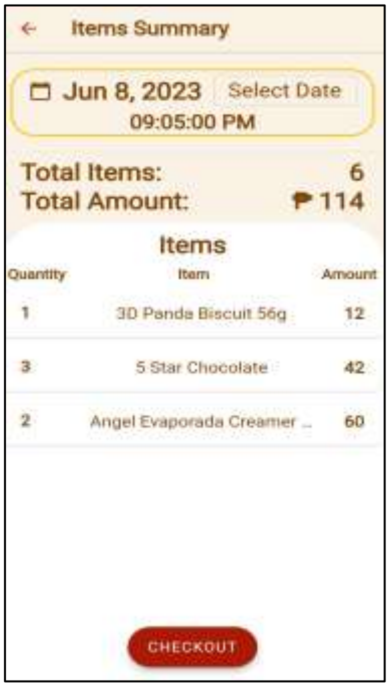
Annie's Langka ...
Price: ₱24.0
Stocks: 49
Quantity: Add

Antonova

Ariel Sunrise

BB Mixed

(a)



Jun 8, 2023
09:05:00 PM

Total Items: 6
Total Amount: ₱114

Items

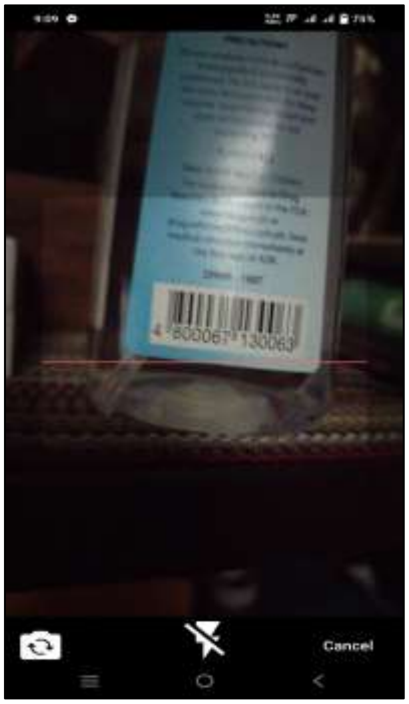
Quantity	Item	Amount
1	3D Panda Biscuit 56g	12
3	5 Star Chocolate	42
2	Angel Evaporada Creamer ...	60

(b)

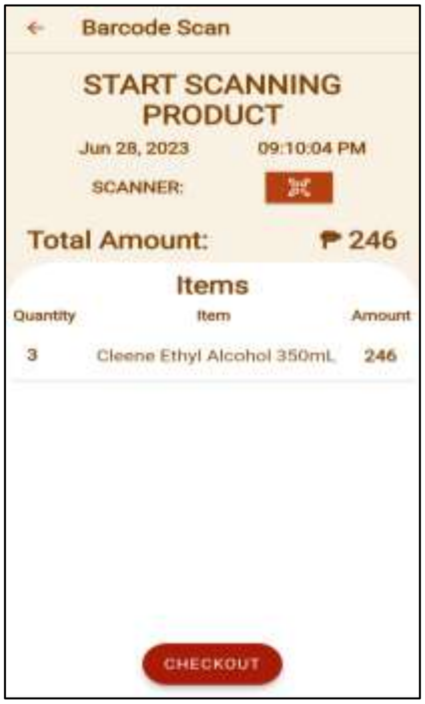
Figure 4.11: Palaris Point-of-Sale via Add Items Method:

(a) Point-of-Sale Page; (b) Items Summary Page in Point-of-Sale

Figure 4.11 shows the POS system of Palaris; this is where the process of items happens. On the POS Page, the user can also view the Product Image, Product Name, Price, Stocks, and Number of Added Items. The user can add and delete the items in the cart depending on the products purchased by the customer. Users can view the Total Items and Total Amount on the Items Summary Page.



(a)



Barcode Scan

START SCANNING PRODUCT

Jun 28, 2023 09:10:04 PM

SCANNER:

Total Amount: ₱ 246

Items

Quantity	Item	Amount
3	Cleene Ethyl Alcohol 350mL	246

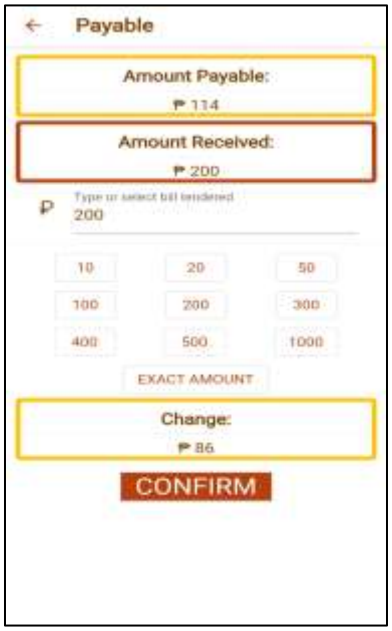
CHECKOUT

(b)

Figure 4.12: Palaris Point-of-Sale via Barcode Scanning Method

*(a) Product Barcode Scanning****;*** ***(****b) Items Summary Page in Point-of-Sale*

Figure 4.12 shows the page for the POS system of Palaris when using the barcode scanning feature. The user has the choice to use the barcode of the product in order to make the POS system at ease. After scanning the product, it will automatically go back to the item summary page, and from there, it shows the total quantity and total amount of the purchased products of the customer.



Payable

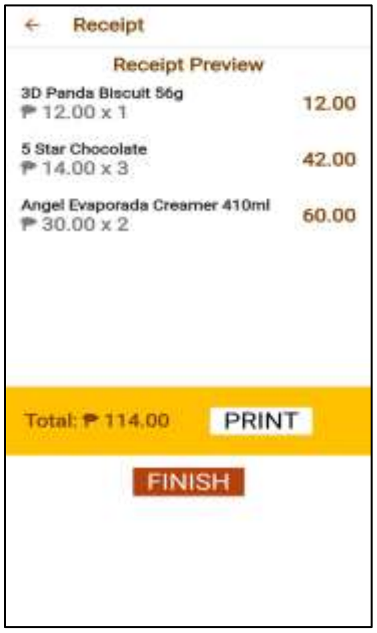
Amount Payable:
₱ 114

Amount Received:
₱ 200

Type or select bill tendered
₱ 200

Change:
₱ 86

(a)



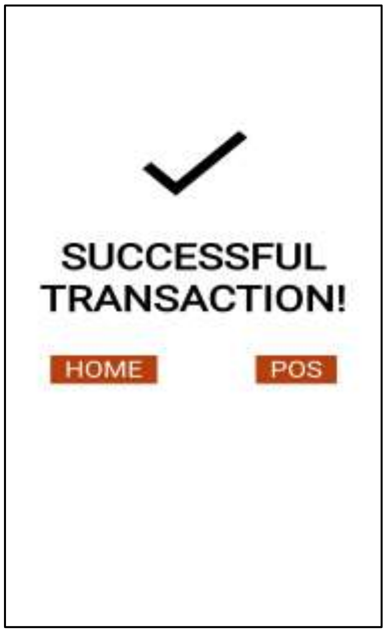
Receipt

Receipt Preview

3D Panda Biscuit 56g ₱ 12.00 x 1	12.00
5 Star Chocolate ₱ 14.00 x 3	42.00
Angel Evaporada Creamer 410ml ₱ 30.00 x 2	60.00

Total: ₱ 114.00

(b)



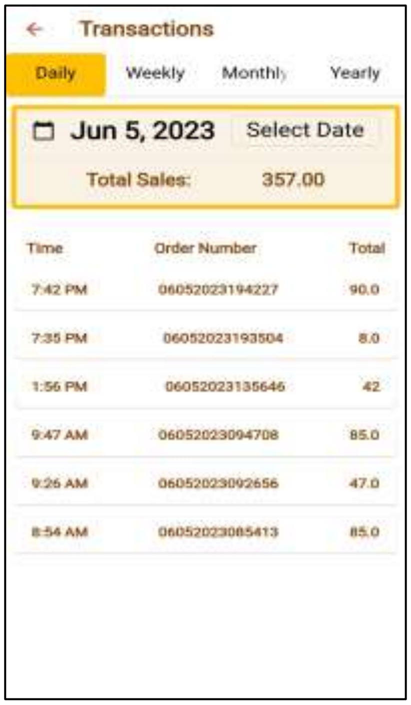
SUCCESSFUL TRANSACTION!

(c)

Figure 4.13: Palaris Payment Page Flow:

(a) Payable Amount Page; (b) Receipt Preview Print and Finish Page; (c) Confirmation Page

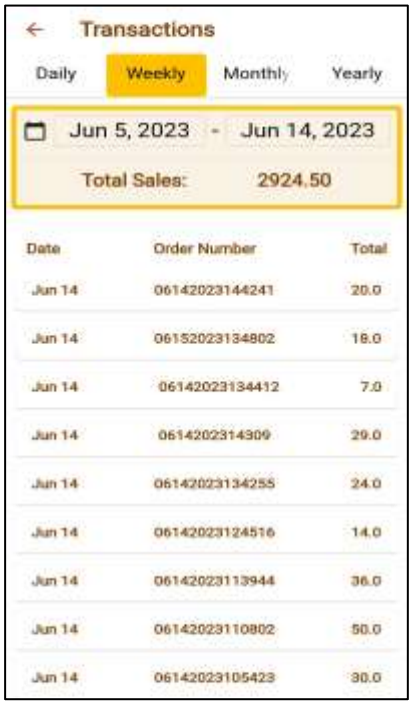
Figure 4.13 shows the Payment Process Flow; the user can select a specific amount, exact amount, and type desired amount. The payment process flow also shows the Receipt Preview, wherein the user can view the summary and total of the items again. The user has the option to print or not print a receipt.



Total Sales: 357.00

Time	Order Number	Total
7:42 PM	06052023194227	90.0
7:35 PM	06052023193504	8.0
1:56 PM	06052023135646	42
9:47 AM	06052023094708	85.0
9:26 AM	06052023092656	47.0
8:54 AM	06052023085413	85.0

(a)



Total Sales: 2924.50

Date	Order Number	Total
Jun 14	06142023144241	20.0
Jun 14	06152023134802	18.0
Jun 14	06142023134412	7.0
Jun 14	0614202314309	29.0
Jun 14	06142023134255	24.0
Jun 14	06142023124516	14.0
Jun 14	06142023113944	36.0
Jun 14	06142023110802	50.0
Jun 14	06142023105423	30.0

(b)

Figure 4.14: Transactions:

(a) Daily Transactions Page; (b) Weekly Transactions Page

Figure 4.14 shows the history of the transaction. It is customizable or viewable in four (4) parts: Daily, Weekly, Monthly, and Yearly. On these pages, the user can view the Order Number and Total. The difference between the Daily and the Weekly, Monthly, and Yearly is that it can view the time on Daily Page, while the latter can only view the Date. The Order Number is set to MM/dd/yyyy/HH:mm:ss. The Total Sales obtained for each section can also be viewed on the said pages.

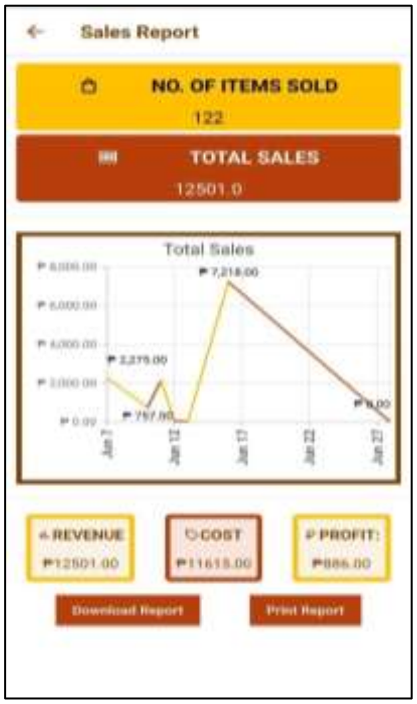
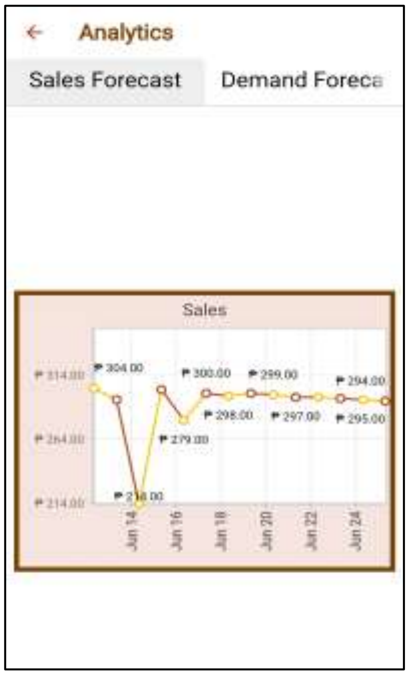
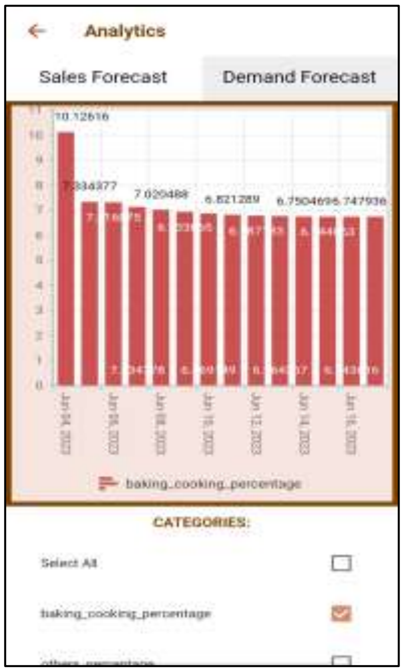


Figure 4.15: Sales Report

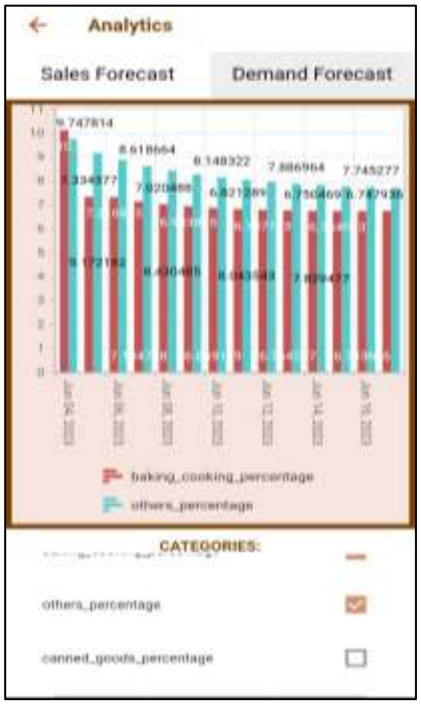
Figure 4.15 shows the Sales Report of the Palaris. In here, the user can see the summary of sales as well as download and print sales report.



(a)



(b)



(c)

Figure 4.16: Analytics Page:

(a) Sales Forecast; (b) Demand Forecast - Specific Category; (c) Demand Forecast - With Two or More Categories

Figure 4.16 shows the Analytics Page of the Palaris. The Sales Forecast Page shows the predicted sales of the user for the following days and months. The Demand Forecast Page shows the predicted products to sell per category. Thus, the page for Demand Forecast can be filtered based on the desired specific product category to view.

4.2.2 Database

This section presents the Palaris Cloud Firestore database, which includes the Cart, Orders, Predictions, and User Collections. These collections are responsible for segregating the data for the convenience of calling out data from the backend to establish in the mobile application.

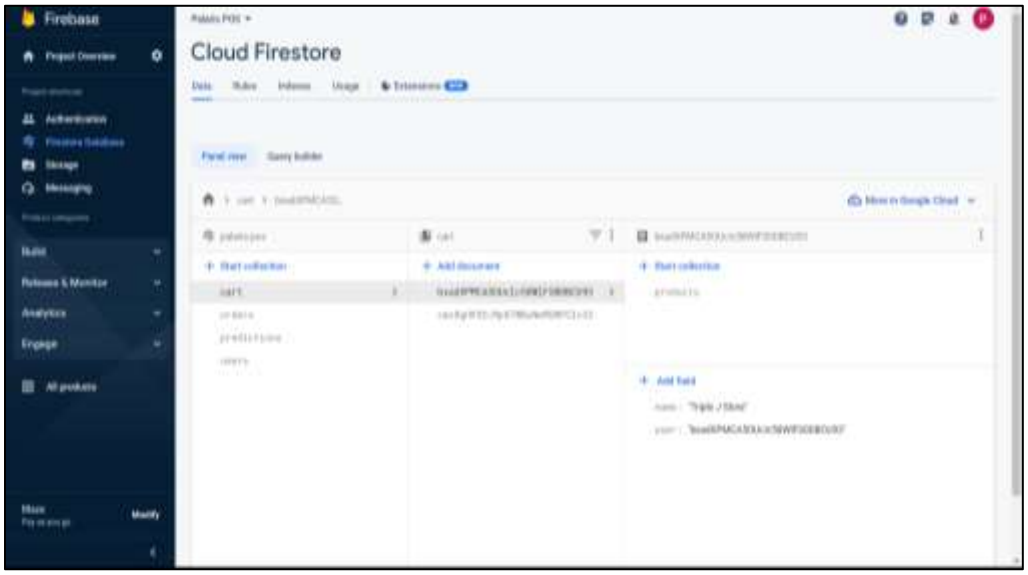


Figure 4.17: Palaris Cloud Firestore Database Main Tab

Figure 4.17 shows the Palaris Cloud Firestore Database Main tab, which contains all the collections the mobile application needs to store and manage data.

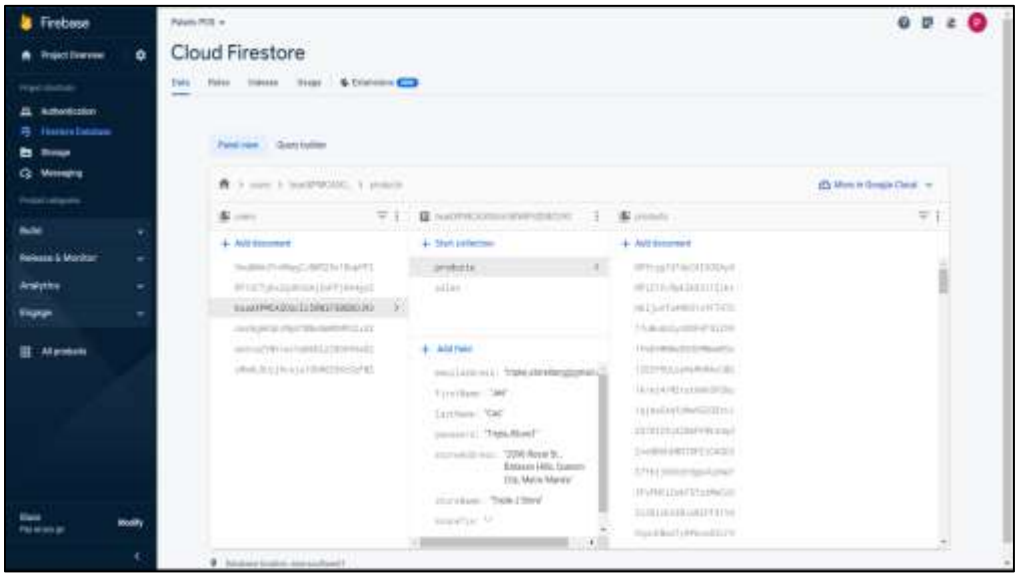


Figure 4.18: Palaris Cloud Firestore Database – Users Products Collection

Figure 4.18 shows the Palaris Cloud Firestore Database – Users Products Collection which contains the information of the users that are registered in the mobile application.

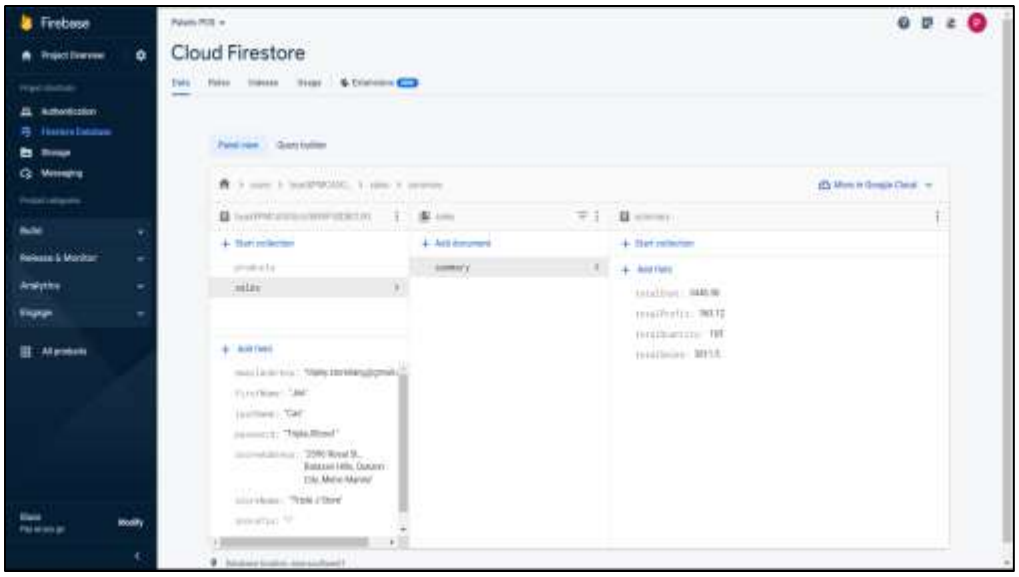


Figure 4.19: Palaris Cloud Firestore Database – Summary of Sales Collection

Figure 4.19 shows the Palaris Cloud Firestore Database – Summary of Sales Collection, which stores the mobile application's total cost and profit and is used for the mobile application's sales report feature.

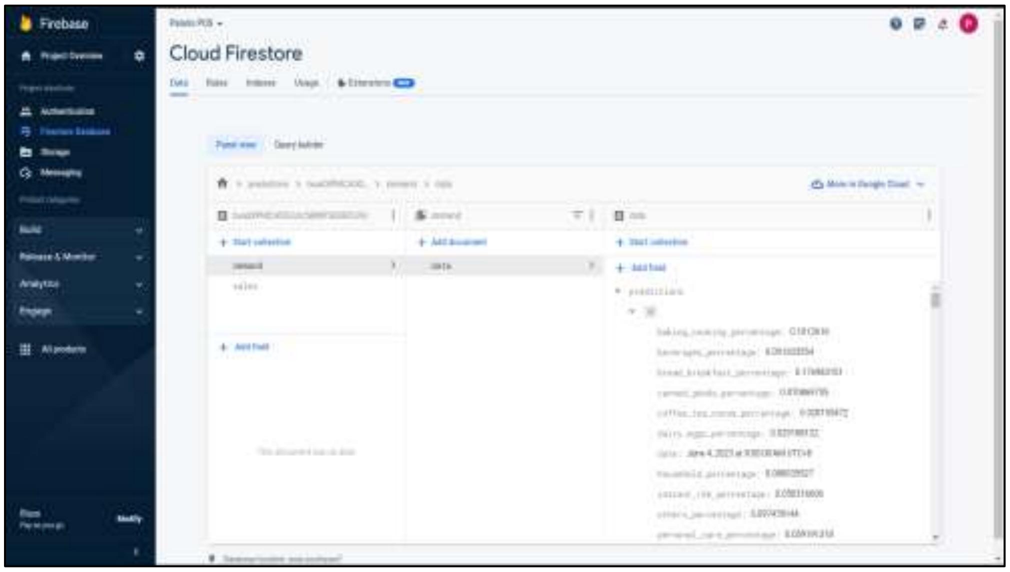


Figure 4.20: Palaris Cloud Firestore Database Predictions – Demand Data Collection

Figure 4.20 shows the Palaris Cloud Firestore Database Predictions – Demand Data Collection which stores all the product’s data and its forecasted demand in each day.

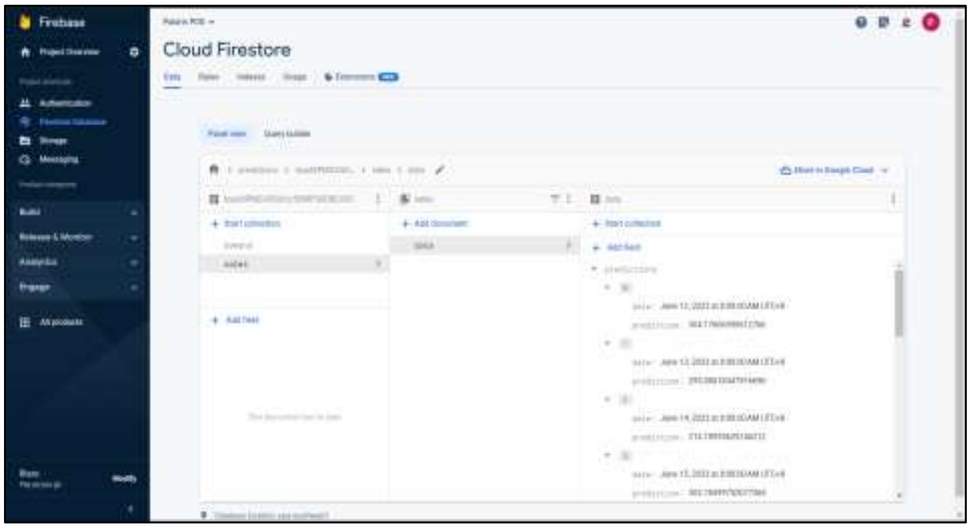


Figure 4.21: Palaris Cloud Firestore Database Predictions – Sales Data Collection

Figure 4.21 shows the Palaris Cloud Firestore Database Predictions – Sales Data Collection stores forecasted sales that were processed using a time-series model, which is used to show all the forecasted sales each day.

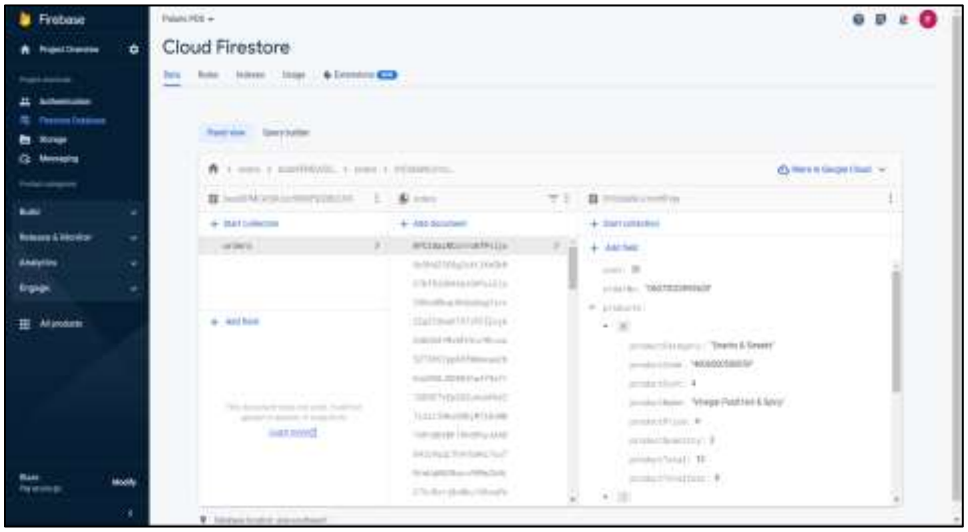


Figure 4.22: Palaris Cloud Firestore Database – Orders Collection

Figure 4.22 shows the Palaris Cloud Firestore Database – Orders Collection, which handles the store's product information, such as the product code, cost, name, price, and quantity.

.4.3 Sales and Demand Forecasting Models’ Results and Analysis (Training Phase)

Table 4.1: Actual and Predicted Values for March 2023

Date	Actual Values	Predicted Values
3/1/2023	5900	4653.417
3/2/2023	6850	4910.724
3/3/2023	9600	4985.096
3/4/2023	9000	4955.036
3/5/2023	6340	4901.032
3/6/2023	6200	4858.234
3/7/2023	5750	4829.613
3/8/2023	9703	4807.821
3/9/2023	6024	4786.931
3/10/2023	7393	4764.777
3/11/2023	9400	4741.432
3/12/2023	8840	4717.531
3/13/2023	6520	4693.495
3/14/2023	4400	4669.449
3/15/2023	2700	4645.367
3/16/2023	4270	4621.198
3/17/2023	3040	4596.912
3/18/2023	3029	4572.504
3/19/2023	4600	4547.976
3/20/2023	4329	4523.333
3/21/2023	5494	4498.576
3/22/2023	3281	4473.705
3/23/2023	5350	4448.721
3/24/2023	2300	4423.623
3/25/2023	2350	4398.411
3/26/2023	3350	4373.085
3/27/2023	4800	4347.644
3/28/2023	3300	4322.09
3/29/2023	4500	4296.421
3/30/2023	5000	4270.639

Table 4.1 shows the actual and predicted values of the total sales of the Triple J store for the month of March 2023.

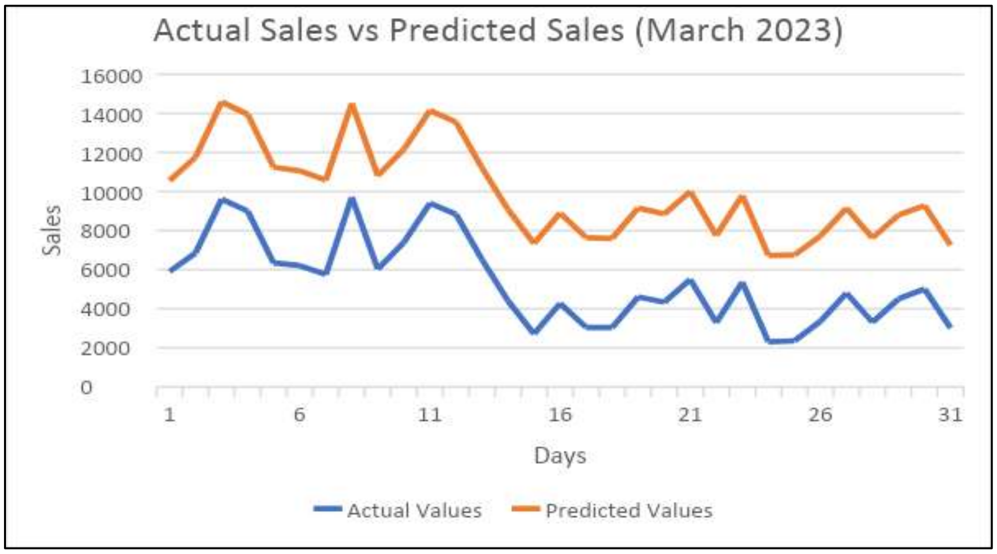


Figure 4.23: Line Plot of Actual vs Predicted Sales for March 2023

Figure 4.23 shows the line plot between actual and predicted values for the total sales of the Triple J store. The horizontal axis serves as the index and the vertical axis serves as the total sales of the Triple J store. It can be observed that there was a huge gap between actual and predicted total sales of the Triple J store.

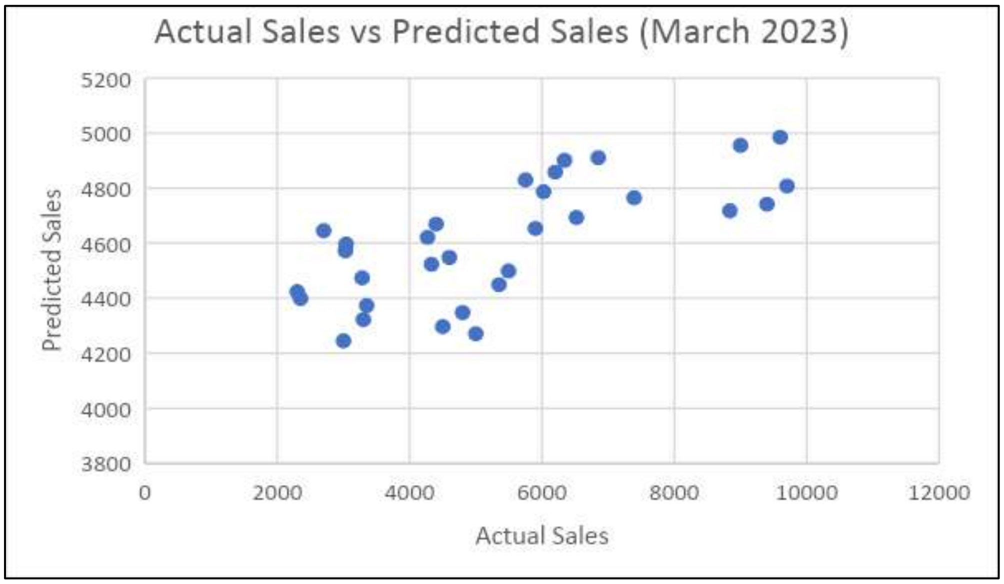


Figure 4.24: Scatter Plot of Actual vs Predicted Sales for March 2023

Figure 4.24 shows the relationship between the actual and predicted values for the total sales of the Triple J store. It can be observed that there is no correlation between the actual and predicted values for the total sales for March 2023.

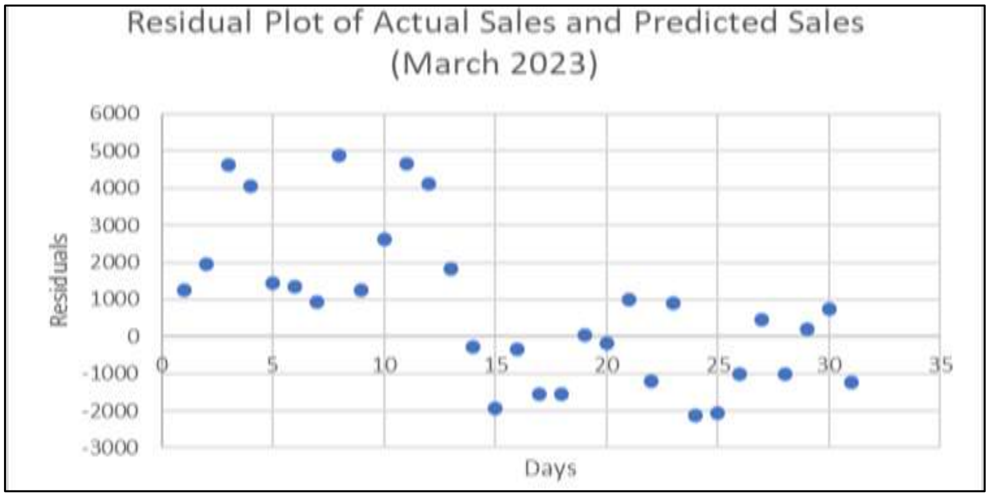


Figure 4.25: Residual Plot of Actual vs Predicted Sales for March 2023

Figure 4.25 shows the residual plot of the linear equation model for the predicted and actual values for the total sales for March 2023. It can be observed that the points lie above 0, which indicates that most of the points of residual values made by subtracting the predicted from the actual values have too low of a difference. Furthermore, points below 0 indicate that residual values made from subtracting the predicted from the actual values are too high in difference. Hence, the accuracy of the prediction for March 2023 predicts too low values from the actual values.

Table 4.2: Performance Metrics for the Sales Forecasting Model for March 2023

MSE	935243.1
RMSE	967.08
MAE	886.83
MAPE	15.81

Table 4.2 shows the performance metrics of the sales forecasting model for March 2023. It can be observed that Mean Square Error (MSE) has a significant value of 935243.1 which indicates that the data points made from the sales forecasting model are dispersedly broad from the mean, and that also indicates that the sales forecasting model predicts inaccurately because its considerable value from 0. Additionally, Root Mean Square Error (RMSE) is the model's effectiveness, and it scored 967.08, which indicates that the average error that the model's prediction has compared to the actual values has a more significant prediction error because of its significant value. Consequently, the Mean Absolute Error (MAE) is also high, and it scored 886.83, which means that, on average, the prediction distance from the actual value is 886.83. Lastly, Mean Absolute Percentage Error (MAPE) scored 15.81, which indicates that the model's prediction is less accurate than the actual values.

Table 4.3: Actual and Predicted Values for April 2023

Date	Actual Values	Predicted Values
4/1/2023	2500	5583.114
4/2/2023	3000	5388.689
4/3/2023	4000	5306.708
4/4/2023	3000	5291.316
4/5/2023	3000	5292.866
4/6/2023	Holy Week	
4/7/2023	Holy Week	
4/8/2023	Holy Week	
4/9/2023	3000	5240.85
4/10/2023	3400	5223.267
4/11/2023	2700	5206.195
4/12/2023	3000	5189.318
4/13/2023	2700	5173.351
4/14/2023	1400	5157.128
4/15/2023	1800	5140.445

Table 4.3 shows the actual and predicted values of the total sales of the Triple J store for April 2023. The Triple J store was closed from April 6 to April 8 due to Holy Week. Consequently, the store owner relocated the Triple J store in their neighborhood street on April 16 to May 7. Hence, there was no gathered data from April 16 to May 7.

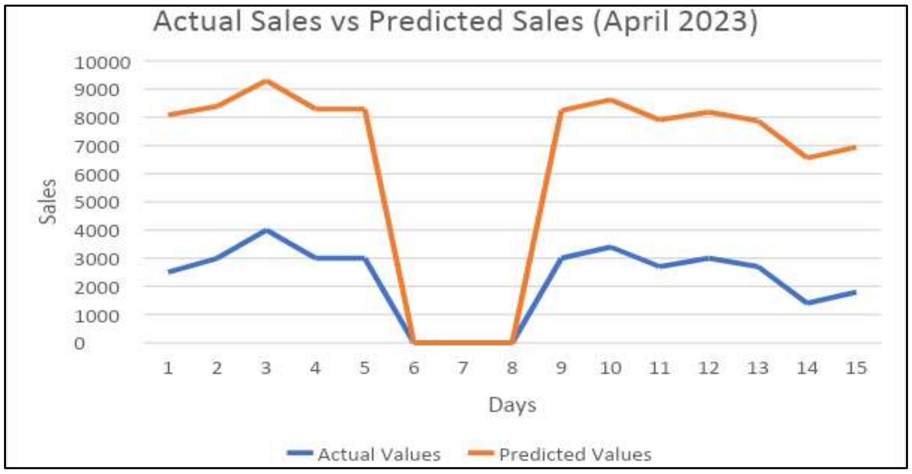


Figure 4.26: Line Plot of Actual vs Predicted Sales for April 2023

Figure 4.26 shows the line plot between actual and predicted values for the total sales of the Triple J store. There was a significant difference between actual and predicted total sales for April 2023.

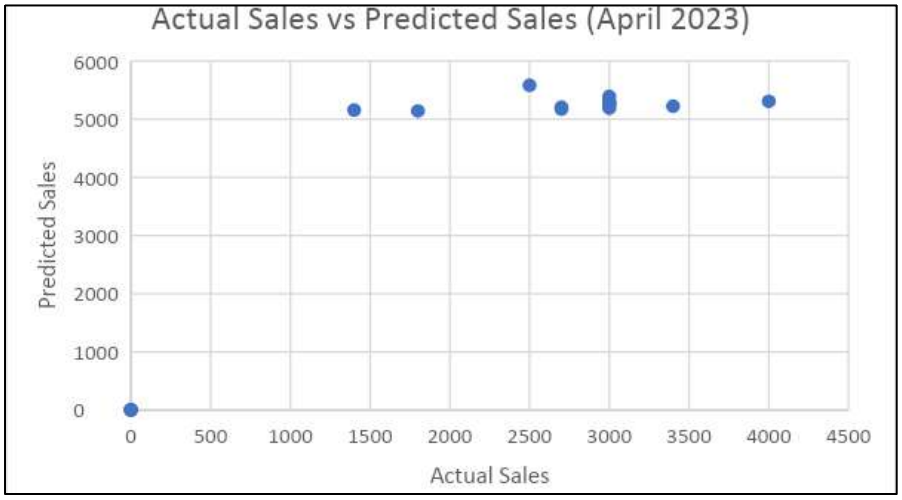


Figure 4.27: Scatter Plot of Actual vs Predicted Sales for April 2023

Figure 4.27 shows the relationship of the actual and predicted values for the total sales of the Triple J store. It can be observed that there is no correlation between the actual and predicted values for the total sales for March 2023.

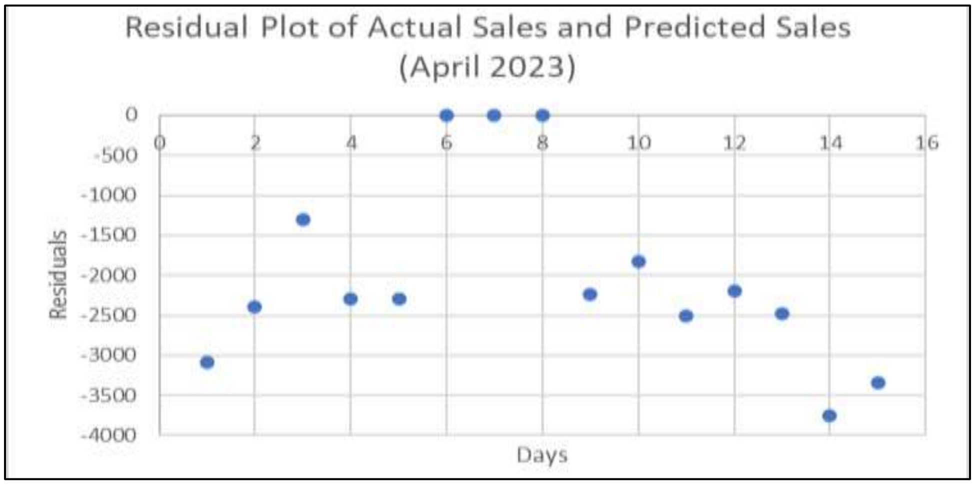


Figure 4.28: Residual Plot of Actual vs Predicted Sales for April 2023

Figure 4.28 shows the residual plot of the linear equation model for the predicted and actual values for the total sales for April 2023. It can be observed that all points lie below 0 which indicates that some residual values made from subtracting the predicted to the actual values have too high of a difference.

Table 4.4: Performance Metrics for the Sales Forecasting Model for April 2023

MSE	4432662
RMSE	2105.39
MAE	1662.78
MAPE	32.96

Table 4.4 shows the performance metrics of the sales forecasting model for April 2023. It can be observed that Mean Square Error (MSE) is more significant than in March 2023, and it scored 4432662, which indicates that the data points made from the sales forecasting model are dispersedly broad from the mean, and that also indicates that the sales forecasting model predicts inaccurately because of its immense value from 0.

Additionally, Root Mean Square Error (RMSE) scored 2105.39, which is higher than March 2023, indicating that the model's prediction's average error compared to the actual values has a more significant prediction error because of its large value. Consequently, the Mean Absolute Error (MAE) is also high, and it scored 1662.78, which is higher than March 2023, and it means that, on average, the prediction distance from the actual value is 1662.78. Lastly, Mean Absolute Percentage Error (MAPE) scored 32.96, which indicates that the model's prediction is less accurate than the actual values.

Table 4.5: Actual and Predicted Values for May 2023

Date	**Actual Values**	**Predicted Values**
5/8/2023	200	3787.266408
5/9/2023	500	3887.379511
5/10/2023	300	3755.510928
5/11/2023	700	3683.644199
5/12/2023	300	3826.358846
5/13/2023	600	3782.843387
5/14/2023	350	3719.171838
5/15/2023	800	3647.572281
5/16/2023	500	3635.757745
5/17/2023	600	3593.463052
5/18/2023	800	3559.178049
5/19/2023	300	3539.171442
5/20/2023	300	3522.38081
5/21/2023	500	3483.731334
5/22/2023	300	3464.891463
5/23/2023	300	3440.786379
5/25/2023	400	3371.376547
5/26/2023	1500	3346.230469
5/27/2023	300	3321.652825
5/28/2023	600	3291.828588
5/29/2023	300	3264.509507
5/30/2023	300	3236.390169
5/31/2023	300	3206.932235

Table 4.5 shows the actual and predicted values of the total sales of the Triple J store for May 2023. The Triple J store was closed from May 1 to May 7 due to the relocation. Additionally, the store was closed on May 24 due to personal matters of the store owner. It can be observed that the actual and predicted values have a considerable gap difference. This vast gap difference was due to the relocation of the Triple J store. The relocation of the Triple J store was due to the indifferences and disagreements between the store owner and the landlord. Hence, this circumstance is unforeseen and unexpected by the proponents. When the store owner relocated to their neighborhood, the sales dropped because the relocated store is slightly hidden from the main street. The main street contains more local consumers than their neighborhood street.

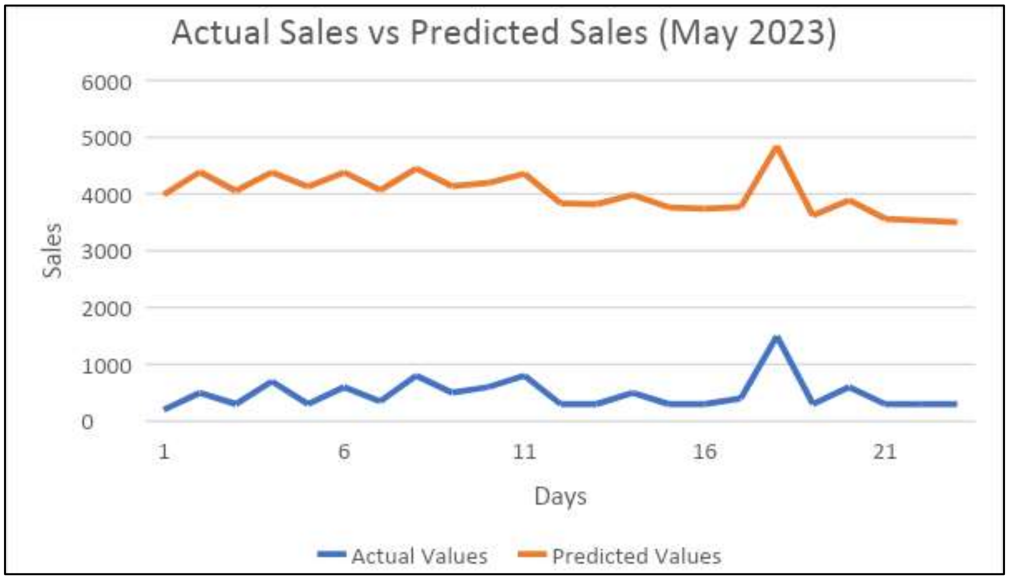


Figure 4.29: Line Plot of Actual vs Predicted Sales for May 2023

Figure 4.29 shows the line plot between actual and predicted values for the total sales of the Triple J store. The horizontal axis serves as the index, and the vertical axis serves as the total sales of the Triple J store. It can be observed that the actual sales are dropping because of the relocation factor of the store, which has caused a considerable difference between the actual and predicted values.

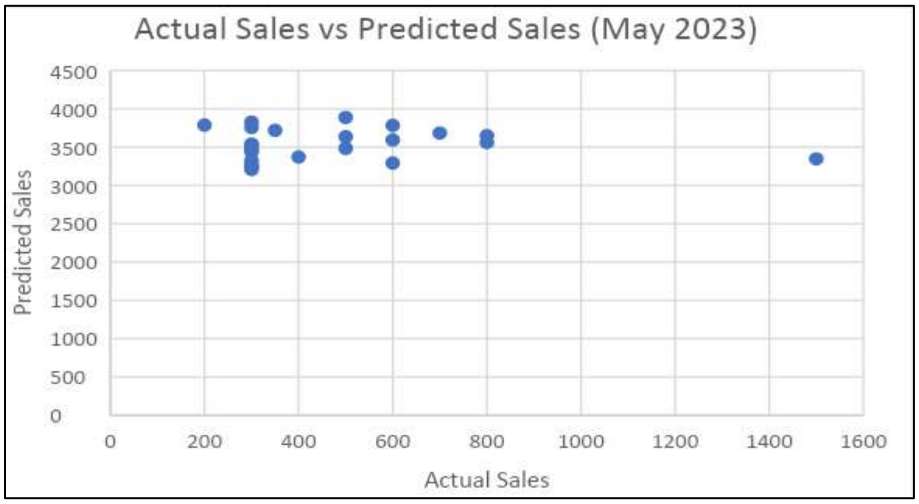


Figure 4.30: Scatter Plot of Actual vs Predicted Sales for May 2023

Figure 4.30 shows the relationship of the actual and predicted values for the total sales of the Triple J store. It can be observed that there is no correlation between the actual and predicted values for the total sales for May 2023 since the sales for this month are constantly dropping.

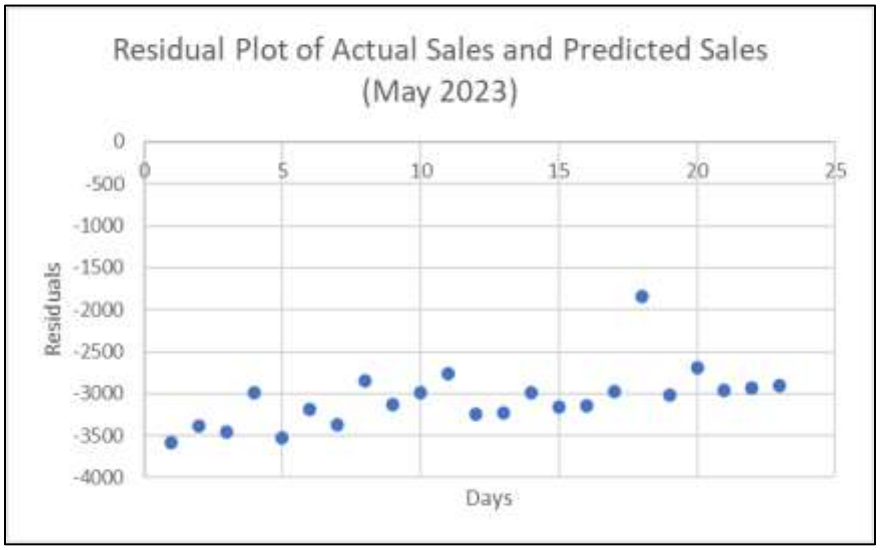


Figure 4.31: Residual Plot of Actual vs Predicted Sales for May 2023

Figure 4.31 shows the residual plot of the linear equation model for the predicted and actual values for the total sales for May 2023. It can be observed that all points that lie below 0 which indicates that all residual values made from subtracting the predicted to the actual values have too high of a difference.

Table 4.6: Performance Metrics for the Sales Forecasting Model for May 2023

MSE	7897775
RMSE	2810.3
MAE	2453.48
MAPE	inf

Table 4.6 shows the performance metrics of the sales forecasting model for April 2023. It can be observed that Mean Square Error (MSE) is more extensive than in March and April 2023, and it scored 7897775, which indicates that the data points made from the sales forecasting model are dispersedly broad from the mean, and that also indicates that the sales forecasting model predicts inaccurately because of its large value from 0. Additionally, Root Mean Square Error (RMSE) scored 2810.3, which is higher than March and April 2023, indicating that the average error that the model's prediction has compared to the actual values has a larger prediction error because of its large value. Consequently, the Mean Absolute Error (MAE) is also high, and it scored 2453.48, which is higher than March and April 2023, and it means that, on average, the prediction distance from the actual value is 2453.48. Lastly, Mean Absolute Percentage Error (MAPE) scored inf, which indicates that the model's prediction is less accurate than the actual values.

4.4 Deployment Results and Analysis (Actual Phase)

This section shows the data gathered during the deployment period.

Table 4.7: Computation for the Mean Absolute Error (MAE) of Actual and Predicted Values of Deployment

Date	Predicted Values	Actual Values	Difference	Absolute Difference	% Difference
6/5/2023	304.18	357	-52.82	52.82	14.79552
6/6/2023	295.01	255	40.01	40.01	15.6902
6/7/2023	214.11	146	68.11	68.11	46.65068
6/8/2023	303.7	171	132.7	132.7	77.60234
6/9/2023	279.66	273.5	6.16	6.16	2.252285
6/10/2023	300.93	281	19.93	19.93	7.092527
6/11/2023	298.62	334	-35.38	35.38	10.59281
6/12/2023	300.73	486	-185.27	185.27	38.1214
6/13/2023	299.92	247	52.92	52.92	21.4251
6/14/2023	297.48	313	-15.52	15.52	4.958466
6/15/2023	297.32	179	118.32	118.32	66.10056
Total	3191.66	3042.5	149.16	727.14	27.7529
Mean Absolute Error	23.89942				

Table 4.7 shows how Mean Absolute Error (MAE) is computed for the accuracy rate of the prediction model. Based on the results in Table 4.7, the Mean Absolute Error of the prediction model is 23.9% which means that the developed prediction model for sales has a 76.1% accuracy rate and 23.9% percentage error rate. Among the previous prediction models developed during the training phase, this prediction model achieved a high accuracy percentage among the rest.

Table 4.8: Results for T-test for the Data for Actual and Predicted Values of Deployment

	Predicted Values	**Actual Values**
Mean	290.1509	276.5909
Variance	679.6661	9403.741
Observations	11	11
Hypothesized Mean Difference	0	
df	11	
t Stat	0.44787	
P(T<=t) two-tail	0.662939	
t Critical two-tail	2.200985	

Table 4.8 shows the t-test results for testing the means of actual and predicted values. The test statistic (0.44787) is lower than the two-tailed critical value (2.200985). Also, the p-value (0.662939) is greater than 0.05, indicating no significant difference between actual and predicted total sales during the deployment phase. Hence, it is evident that the actual and predicted sales are accurate.

Table 4.9: Verbal Interpretation for System Functionality

Weighted Means	**Verbal Interpretation**
1 – 1.80	Very Dissatisfied/ Very Difficult/ Not effective at all/ Not at all completely/ Very Dissatisfied
1.81 – 2.60	Dissatisfied/ Difficult/ Partial effective/ Partial complete/ Dissatisfied
2.61 – 3.40	Neutral
3.41 – 4.20	Satisfied/ Easy/ Effective/ Partial complete/ Satisfied
4.21 – 5	Very Satisfied/ Very Easy/ Extremely Effective/ Completely/ Very Satisfied

Table 4.9 shows the means of the Likert scale survey questionnaire for part I, software functionality of the post-deployment of the mobile POS application; this aims to measure the store owner's satisfaction with the overall performance of the mobile POS software application.

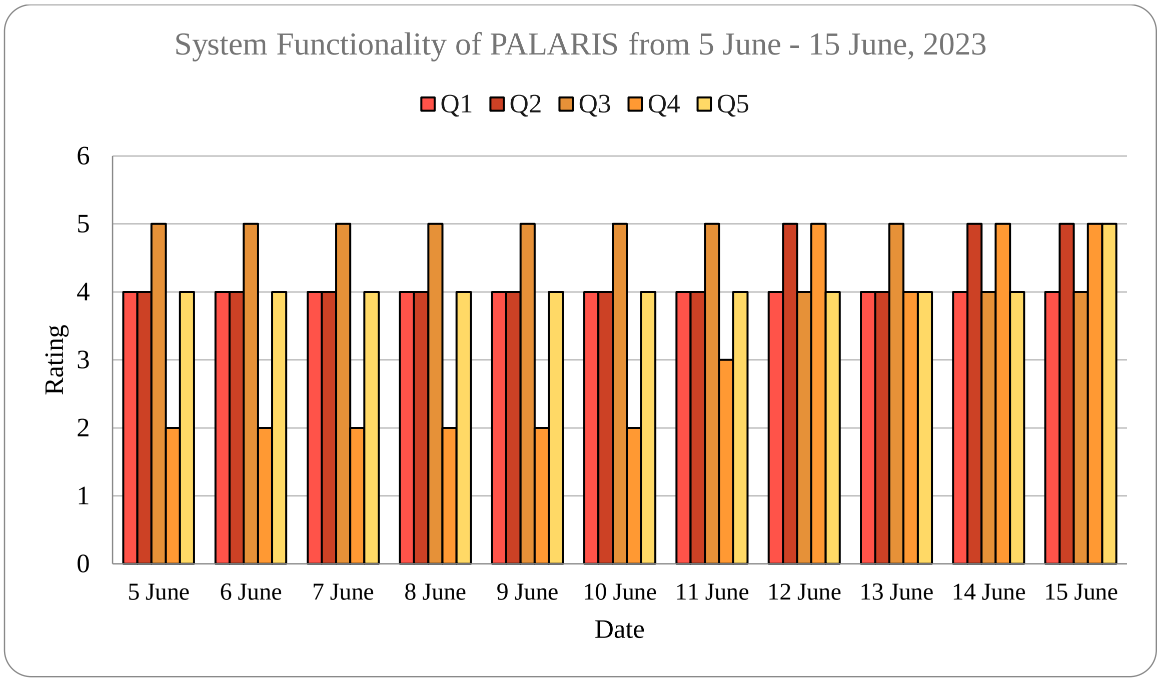


Figure 4.32: Questionnaire Survey Results for Software Functionality

In terms of system functionality, in Q1, it pertains to the overall satisfaction of the store owner with the software functionality of the app, and it has a mean of 4, which means that the store owner is satisfied based on the verbal interpretation from Table 9. Q2 pertains to easy navigation and use of the app's features, and it scored 4.27 on average, which means that for the store owner, the mobile POS application is straightforward to use and navigate. Q3 pertains to the effectiveness of the app's reporting and analytics features in providing valuable insights for her business, and it has an average of 4.73, which means that the mobile POS application is efficient in terms of the app's reporting and analytics features.

Q4 pertains to the extent to which the app meets her specific business requirements and needs, and it has an average of 3.09 and is interpreted as neutral. Lastly, Q5 pertains to how satisfied the store owner regards the app's integration with other systems or platforms, and it has an average of 4.09; this means that the store owner is satisfied with the app's integration with other systems or platforms.

Table 4.10: Verbal Interpretation for Hardware Functionality

Weighted Means	**Verbal Interpretation**
1 – 1.80	Very Difficult to Use/ Very Poorly/ Very Inaccurate and Unreliable/ / Very Dissatisfied/ Not at all completely
1.81 – 2.60	Difficult to Use / Poorly/ Inaccurate and Unreliable / Dissatisfied / Partial complete
2.61 – 3.40	Neutral
3.41 – 4.20	Easy to Use / Well/ Accurate and Reliable / Satisfied / Easy
4.21 - 5	Very Easy to Use / Very Well/ Very Accurate and Reliable / Very Satisfied / Very Easy

Table 4.10 shows the means of the Likert scale survey questionnaire for part II, hardware functionality of the post-deployment of the mobile POS application; this aims to measure the store owner's satisfaction with the overall performance of the compatibility with a hardware device (i.e., barcode scanner and thermal printer).

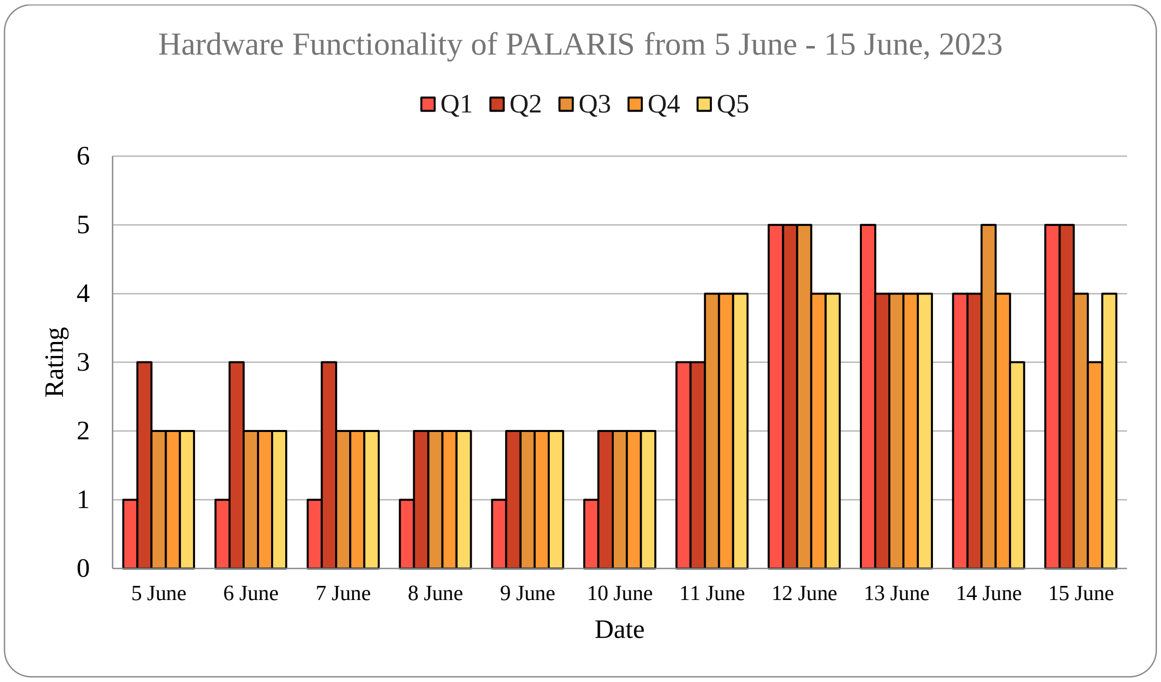


Figure 4.33: Questionnaire Survey Results for Hardware Functionality

In terms of hardware functionality, Q1 pertains to the rate of experience of the store owner with the integrated thermal printer function, and it has an average of 2.55, which means that the store owner has difficulty using the integrated thermal printer function. Consequently, Q2 pertains to how the app captures and interprets barcode information through the barcode scanner function. It has an average of 3.27, meaning the store owner is neutral regarding the barcode scanner function. Q3 pertains to the accuracy and reliability of the barcode scanner function in the app, and it has an average of 3.09, which means the store owner is neutral in rating the barcode scanner function of the app. Q4 pertains to how satisfied the store owner is with the compatibility and ease of integration with hardware components, and it has an average of 2.82, which means that the store owner is neutral in the compatibility and integration of the mobile POS application with hardware components.

Table 4.11: Verbal Interpretation for Predictive Model

Weighted Means	Verbal Interpretation
1 – 1.80	Never/ Very Inaccurate/ Very Poorly/ No Impact/ Not Confident at All
1.81 – 2.60	Rarely/ Inaccurate/ Poorly/ Minimal Impact/ Partial Confident
2.61 – 3.40	Neutral
3.41 – 4.20	Sometimes / Accurate/ Well / Significant Impact / Confident
4.21 - 5	Always / Very Accurate/ Very Well / Very Significant Impact / Very Confident

Table 4.11 shows the means of the Likert scale survey questionnaire for part III, the predictive model of the post-deployment of the mobile POS application; this aims to measure the store owner's satisfaction with the overall performance of the predictive model, which determines the possible sales and demand in the future.

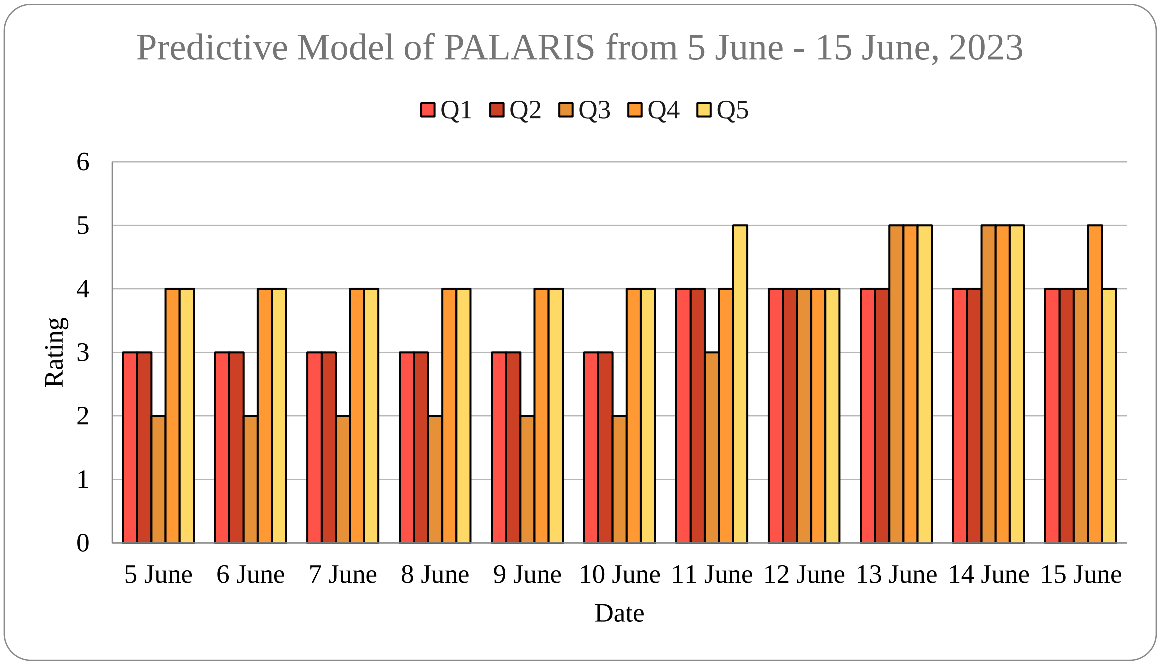


Figure 4.34: Questionnaire Survey Results for Predictive Model

In terms of a predictive model, Q1 pertains to how often the store owner depends on her business decisions on the sales predictions in the mobile POS application, and it scored 3.45, which means that sometimes the store owner depends on her business decisions in the sales predictions generated in the mobile POS application. Q2 pertains to how accurate the store owner finds the sales predictions generated in the mobile POS application, and it scored 3.36, which means that the store owner is neutral regarding the accuracy of the sales predictions generated in the mobile POS application. Q3 pertains to how the app's predictive model aligns with the actual sales performance of her business, and it scored 3, which means that the store owner is neutral in terms of alignment of the app's predictive model with the actual sales performance of her business. Q4 pertains to the rate impact of using the predictive model on her business decision-making process, and it scored 4.27, meaning that the predictive model significantly impacted her business decision-making process. Lastly, Q5 pertains to how confident the store owner is in terms of the reliability and usefulness of the app's sales forecasting model, and it scored 4.27, which means that the store owner is very confident in terms of the reliability and usefulness of the app's sales forecasting model.

Table 4.12: Overall Mean Results for Survey Questionnaire

System Functionality	4.04
Hardware Functionality	2.91
Predictive Model	3.67

Table 4.12 shows the overall mean of the post-deployment survey questionnaire regarding system functionality, hardware functionality, and predictive model. The system functionality section refers to the software capabilities of the mobile POS application, such as generating sales reports and predictive analytics for sales and demand and integrating

the mobile POS application with other systems or platforms. The hardware functionality section refers to mobile POS applications' effectiveness in integrating them with hardware components such as thermal printers and barcode scanners. Lastly, the predictive model section refers to the accuracy of the sales and demand forecasting models and how the store owner depends on her business decisions based on the sales predictions generated by the mobile POS application.

Table 4.13: Verbal Interpretation for Overall Mean Results for Post-Deployment Survey

Weighted Means	Verbal Interpretation
1 – 1.80	Not Satisfied
1.81 – 2.60	Partially Satisfied
2.61 – 3.40	Neutral
3.41 – 4.20	Satisfied
4.21 – 5	Very Satisfied

Overall, the system functionality scored 4.04, which means that the store owner is satisfied with the system functionality of the mobile POS application, such as easy navigation, generating sales reports and predictions, and meeting the specific business needs and requirements. Continuously, the hardware functionality scored 2.91, which means that the store owner is neutral in terms of the compatibility of the mobile POS application with hardware components such as barcode scanner and thermal printer and the accuracy of translating and reflecting the barcode of the products into the mobile POS application using the integrated barcode scanning capability of the mobile POS application. Lastly, the predictive model scored 3.67 overall, which means that the store owner is satisfied with the accuracy, reliability, and usefulness of the predictive model of the mobile POS application.

Chapter 5

CONCLUSIONS AND RECOMMENDATIONS

This chapter presents the summary of findings, the conclusions drawn from the results, and the possible recommendations for improvements in the study.

5.1 Summary of Findings

Sari-sari stores fall under the Micro, Small, and Medium Enterprises that can be found almost everywhere in the Philippines, and it comprises 30% to 40% of the retail sales in the Philippines because it offers affordable prices of essential products such as canned goods, instant noodles, soft drinks, and snacks. Also, it resells smaller versions of packaging of different products to cater to other people with a limited budget for the whole day. However, even though the sari-sari store comprises almost half of the retail sales in the Philippines, there still needs to be more innovative technology solutions available to alleviate the difficulty in managing the sari-sari store, especially their inventory. Manual tallying stocks of the products and items sold requires tedious and tremendous effort for the sari-sari store owner. Hence, most sari-sari store owners need more inventory management skills. In replacement for the manual inventory, a mobile POS application with sales and demand forecasting was proposed to ensure that sari-sari store owners can monitor and track their products. Also, the sari-sari store owners will be notified if the product has fewer stocks. Additionally, with the developed mobile POS application, sari-sari store owners can access their store's future sales and demand, enabling them to formulate a strategic plan to maximize their profit.

The developed mobile POS application is limited to the following: (1) it can be operated only in an Android operating system, (2) it can be available in online-access only using Wi-Fi, (3) it offers only cash-based payments, and (4) it only uses historical data of the chosen deployment mini grocery store, Triple J store which is located in 2096 Rosal Street, Batasan Hills, Quezon City to train the sales and demand forecasting model. The project study utilized descriptive and developmental research design. Also, SARIMA and Prophet were machine learning algorithms used to train the sales and demand.

5.2 Conclusion

Based on the findings and results of the study, the following are the conclusions drawn out by the proponents:

1. Dart language and Flutter framework were used to develop a mobile application for a point-of-sale system. It was proved that it has a user-friendly interface for the store owner because it is evident in the overall mean score of the system functionality of 4.04 which means that the store owner is satisfied with the software capability such as easy navigation, generating sales reports and predictions that meets with her specific business requirements and needs. Continuously, the mobile POS application proved it had established compatibility because of an overall score of 2.91 which means that the store owner is neutral in terms of the compatibility of the mobile POS application with hardware components such as the barcode scanner and thermal printer and the accuracy of translating and reflecting the barcode of the products into the mobile POS application using the integrated barcode scanning capability of the mobile POS application.
2. Time-series model was developed through SARIMA and Prophet. SARIMA analyzes historical seasonality and trends using the gathered historical sales

data of the Triple J Store to formulate the sales forecasting model. Prophet was applied in the demand forecasting because it handles the outliers in the gathered sales dataset through weighted sum of the independent variables that eliminates the dependency of the data with one another. Hence, it was used for developing the demand forecasting because this model is suited for non-linear trends that are fit daily, weekly, and yearly seasonality. These time-series forecasting models were proved to be effective in developing prediction model as it has an overall accuracy of 76.1%

3. Palaris established the compatibility feature successfully between the interface of the procured two thermal printers and handheld POS through the means of Bluetooth. Thus, it can generate sales receipt upon purchase.
4. Through field testing, the developed mobile POS application proved to be effective for the store owner in terms of system functionality and prediction model. The system functionality scored with an overall mean of 4.04. The prediction model scored an overall average of 3.67 and the hardware functionality scored 2.91.

5.3 Recommendation

For future proponents who may further conduct this project study, the proponents have the following recommendations:

1. Integrate online payment for cashless transactions.
2. Gather historical data of stores in different areas to compare whether the prediction model is still accurate.

3. Conduct a study to develop a more scalable prediction model to determine the relationship of other factors such as weather, seasons, holidays, and location to the sales and demand of the store.
4. Integrate offline persistence to the database to access and synchronize the product transactions offline to fully acquire and utilize the full functionality of the POS mobile application.
5. Deploy the mobile POS (Point-of-Sale) application and prolong the testing of the sales and demand forecasting models by deploying the mobile POS application to different stores to achieve more accurate and unbiased results of data.

References

[1] D. Nosowitz, “What Do You Call the Corner Store?,” 2016, [Online]. Available: https://www.atlasobscura.com/articles/what-do-you-call-the-corner-store

[2] “What’s a Sari-Sari Store: PH’s small neighborhood retail shop,” *What’s a Sari-Sari Store PH’s small neighborhood Retail shop*, 2016, [Online]. Available: https://primer.com.ph/tips-guides/2016/07/30/expat-guide-sari-sari-stores/

[3] P. Atty. Paranas, “The TRAIN Law and MSMEs,” *LMA Law*. https://www.lmalaw.org/index.php/blog/item/36-the-train-law-and-msmes (accessed Jun. 09, 2022).

[4] “2020 MSME Statistics,” 2020. Accessed: Jun. 10, 2022. [Online]. Available: https://www.dti.gov.ph/resources/msme-statistics/#:~:text=2020 MSME Statistics&text=Of these%2C 952%2C969 (99.51%25),at 0.49%25 (4%2C716).

[5] C. S. Lorenciana, “Starting your own sari-sari store,” 2014, [Online]. Available:https://www.pressreader.com/philippines/the-freeman/20140421/281831461725816

[6] “4 Main Reasons Why Sari-Sari Stores Fail,” 2017, [Online]. Available: https://pinoynegosyo.net/sari-sari-stores-fail-1817.html

[7] “Proper Inventory Management for your Sari-Sari Store,” 2020, [Online]. Available: https://www.condura.com/condura-guide/proper-inventory-management-for-your-sari-sari-store/

[8] M. Keenan, “What Are Stockouts And How Can I Prevent Them?,” 2021, [Online]. Available: https://www.shopify.com/ph/retail/what-causes-a-stockout#:~:text=The most obvious consequence of,recurring sales in the future.

[9] R. Vargheese and H. Dahir, “An IoT/IoE enabled architecture framework for precision on shelf availability: Enhancing proactive shopper experience,” *Proc. - 2014 IEEE Int. Conf. Big Data, IEEE Big Data 2014*, pp. 21–26, 2014, doi: 10.1109/BigData.2014.7004418.

[10] M. Waida, “What Is Inventory Management and Why Is it Important?,” 2022, [Online]. Available: https://www.wrike.com/blog/what-is-inventory-management/#What-is-inventory-management

[11] M. Freedman, “Types of POS Systems,” 2021, [Online]. Available: https://www.business.com/articles/types-of-pos-systems/

[12] A. T. Stubbs and A. Conrad, “What Is a Point of Sale (POS) System?,” 2021,[Online].Available:https://www.softwareadvice.com/resources/what-is-a-point-of-sale-system/

[13] M. Freedman, “How to Develop a Sales Report,” 2021, [Online]. Available:https://www.businessnewsdaily.com/15988-how-to-write-a-sales-report.html

[14] S. I. Lestariningati, “Mobile point of sale design and implementation,” IOP Conf. Ser. Mater. Sci. Eng., vol. 407, no. 1, 2018, doi: 10.1088/1757-899X/407/1/012094.

[15] E. C. Lewis, K. M. Harper, L. K. Poirier, and J. Gittelsohn, “Feasibility of using mobile point-of-sale technology in Baltimore City corner stores for tracking sales: A brief report,” 2021. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8883529/ (accessed Jun. 10, 2022).

[16] “What Is a Mobile Point of Sale?” https://www.verizon.com/business/small-

business-essentials/resources/what-is-mobile-point-of-sale/

[17] "Peddlr: Frequently Asked Questions," Peddlr. https://peddlr.io/faq#:~:text=Peddlr is a smart Point,to understand your business performance.

[18] "5 Useful Business Tips for Sari-Sari Store Owners," Business News Philippines. https://www.businessnews.com.ph/useful-business-tips-for-sari-sari-store-owners-20160129/

[19] A. Khanna and R. Tomar, "IoT based interactive shopping ecosystem," Proc. 2016 2nd Int. Conf. Next Gener. Comput. Technol. NGCT 2016, no. October, pp. 40–45, 2017, doi: 10.1109/NGCT.2016.7877387.

[20] A. Ramli, R. M. Zain, M. A. Razik, and A. S. Yaacob, "Micro Businesses: Do They Need Accounting?," *Int. J. Acad. Res. Bus. Soc. Sci.*, vol. 7, no. 9, Oct. 2017, doi: 10.6007/ijarbss/v7-i9/3317.

[21] A. M. N. Margate, M. C. F. Ravina, J. J. G. Pido, and M. N. Young, "Seiton: A Mobile Inventory Management System Application for Micro, Small and Medium-sized Enterprise," Dec. 2020. doi: 10.1109/ICETAS51660.2020.9484183.

[22] H. J. Malapit, "The Filipino Sari-Sari Store," 2007.

[23] T. Arceo-Dumlao, "Sari-sari store owners get lift from Sun Life, Hapinoy," *Inquirer.Net*, Oct. 12, 2012. Accessed: Jun. 09, 2022. [Online]. Available: https://business.inquirer.net/85796/sari-sari-store-owners-get-lift-from-sun-life-hapinoy

[24] R. O. Reyes, "Peddlr app to help Philippine micro, small businesses," *Sunstar*, Tacloban, Apr. 06, 2021. Accessed: Jun. 01, 2022. [Online]. Available:https://www.sunstar.com.ph/article/1890890/tacloban/business/peddlr-app-to-help-philippine-micro-small-businesses

[25] D. E. Marcial, D. P. Mupe, and D. E. Marcial, "Assessing the Challenges of inventory management for small and medium enterprises in Dumaguete City: Inputs towa... Assessing the Challenges of inventory management for small and medium enterprises in Dumaguete City: Inputs towards the development of a portable ERP."

[26] C. H. Godoy Jr, C. V. Llanillo, D. A. Ramos, J. P. Tagubase, E. Taguibao, and Q. Zacarias, "Development of Mobile Application Inventory Control System for Efren's Sari-sari store," Technological University of The Philippines, Manila, 2021. Accessed: Jun. 06, 2022. [Online]. Available:https://www.researchgate.net/publication/351442808_Development_of_Mobile_Application_Inventory_Control_System_for_Efren's_Sari-sari_store

[27] "8 Sari Sari Store Design and Concept," *Piso and Beyond!*, 2014. http://www.pisoandbeyond.com/2014/09/8-sari-sari-store-designs-and-concepts.html (accessed Jun. 09, 2022).

[28] J. Cordial, "Inventory Management Tools for Local Micro"

[29] K. S. M. Cuevas, "DEPARTMENT OF AGRIBUSINESS MANAGEMENT AND ENTREPRENEURSHIP COLLEGE OF ECONOMICS AND MANAGEMENT UNIVERSITY OF THE PHILIPPINES LOS BAÑOS DETERMINANTS AND

IMPACT OF RECORDKEEPING TO AGRIBUSINESS GROWTH: A STUDY ON PUBLIC MARKETSMALL AND MICRO ENTERPRISES IN LOS BAÑOS, LAGUNA Special Problem Report BS IN AGRIBUSINESS MANAGEMENT," 2017.

[30] W. T. Sucuahi, "Determinants of Financial Literacy of Micro Entrepreneurs in Davao City," *Int. J. Account. Res.*, vol. 1, no. 1, pp. 44–51, 2013, doi: 10.12816/0001127.

[31] R. Bee and A. Aleem, "SALES AND INVENTORY MANAGEMENT SYSTEM," 2013.

[32] Clariades-Rubio, J., "2017 Food Retail Sectoral Report"

[33] M. S. C. C. Santa Maria, "Income Growth," Nov. 2019. Accessed: Jun. 09, 2022. [Online]. Available: https://www.academia.edu/41188523/Income_Growth?auto=download

[34] M. De Lima, G. C. Samson, J. V. Taghoy, and E. L. Veranio, "Accounting Literacy and Sustainability Among Sari-Sari Stores: Basis on Intervention Program," 2020, Accessed: Jun. 09, 2022. [Online]. Available: https://www.ejournals.ph/article.php?id=16393

[35] T. Funahashi, "Distribution and consumption in the Base of the Pyramid (BOP） market-The case of Sari-Sari stores in the Philippines," Mar. 2013. Accessed: Jun. 01, 2022. [Online]. Available: https://m-repo.lib.meiji.ac.jp/dspace/bitstream/10291/16194/1/keieironshu_60_4_211.pdf

[36] A. Jullian, K. Alindahao, and M. T. Donozo, “Importance of Sari-Sari stores in the time of a Pandemic.”

[37] J. C. Gano-An and G. P. Gempes, “The Success and Failures of Sari-Sari Stores: Exploring the Minds of Women Micro-Entrepreneurs,” HOLISTICA – Journal of Business and Public Administration, vol. 11, no. 2, pp. 25–51, 2020, doi: 10.2478/hjbpa-2020-0017.

[38] M. D. Cookson and P. M. R. Stirk, “Difference between POS and MPOS system,” 2021. http://mpossystem.com/day-today-epos/

[39] KoronaPOS, “The Difference Between POS and mPOS & Mobile Payments Explained,” 2019. https://koronapos.com/blog/difference-between-pos-and-mpos/

[40] M. Cote and T. McCarthy, “The Power of Point of Sale Improving Growth, Profit, and Customer Service in a Retail Business,” Nat Genet, vol. 27, no. 1, pp. 5–6, 2015, [Online]. Available: http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&dopt=Citation&list_uids=11137984

[41] Y. G. Kim and J. Lim, “A POS system based on the remote client-server model in the small business environment,” Management Research Review, vol. 34, no. 12, pp. 1334–1350, Nov. 2011, doi: 10.1108/01409171111186432.

[42] A. Gillum and M. A. Rob, “IT PROJECT MANAGEMENT: CLASS PROJECT OF A POINT OF SALE (POS) SYSTEM IMPLEMENTATION IN A RESTAURANT,” 2011.

[43] O. O. Adeoti, “African Journal of Business Management Challenges to the efficient use of point of sale (POS) terminals in Nigeria,” vol. 7, no. 28, pp. 2801–2806, 2013, doi: 10.5897/AJBM12.1129.

[44] Zhang, Mu & Huang, Xiao-nan & Yang, Chang-bing. (2020). A Sales Forecasting Model for the Consumer Goods with Holiday Effects. Journal of Risk Analysis and Crisis Response. 10. 10.2991/jracr.k.200709.001.

[45] Kaipov, Ivan & Nedzved, Alexander. (2020). Sales forecasting of goods in shoe retail. 6. 10-17.

[46] D.M.G.T, Dassanayake & Nugaliyadde, Anupiya & Mallawarachchi, Yashas. (2016). Increasing sales productivity through sales forecasting using ARIMA models. 10.13140/RG.2.1.3040.4088.

[47] A. Krishna, V. Akhilesh, A. Aich, and C. Hegde, “Sales-forecasting of Retail Stores using Machine Learning Techniques,” *Proc. 2018 3rd Int. Conf. Comput. Syst. Inf. Technol. Sustain. Solut. CSITSS 2018*, pp. 160–166, 2018, doi: 10.1109/CSITSS.2018.8768765.

[48] B. M. Pavlyshenko, “Machine-learning models for sales time series forecasting,” Data, vol. 4, no. 1, pp. 1–11, 2019, doi: 10.3390/data4010015.

[49] Praveen K B. (2020). Inventory Management using Machine Learning. International Journal of Engineering Research And, V9(06). https://doi.org/10.17577/ijertv9is060661

[50] Sri, B., Ramya, S., & Vedavathi, K. (2020). An Advanced Sales Forecasting Using Machine Learning Algorithm. In *International Journal of Innovative Science and Research Technology* (Vol. 5, Issue 5). www.ijisrt.com

[51] Taranenko, Liudmyla & Molodoria, Anastasiia. (2022). How to Apply Machine Learning to Demand Forecasting.

[52] Ben Baccar, Yacine. (2019). Comparative Study on Time Series Forecasting Models. 10.13140/RG.2.2.32241.02408.

[53] Thi Thanh Binh, Vu & Huong, Hoang Thi. (2021). CRITICAL FACTORS AFFECTING SALES REVENUE IN SUPERMARKET CHAINS: A CRITICAL REVIEW OF LITERATURE - Journal of Science & Technology - https://jst-haui.vn/media/30/uffile-upload-no-title30687.pdf. 57. 130-134.

[54] Watkins, A. (n.d.). *ScholarWorks Successful Demand Forecasting Modeling Strategies for Increasing Small Retail Medical Supply Profitability*. https://scholarworks.waldenu.edu/dissertations

[55] Rose, N., & Dolega, L. (2022). It's the Weather: Quantifying the Impact of Weather on Retail Sales. *Applied Spatial Analysis and Policy*, *15*(1), 189–214. https://doi.org/10.1007/s12061-021-09397-0

[56] Krylstedt, J., & Weidlertz, A. (n.d.). *A Study of Weather's Impact on Consumption of Goods For Certain Weather-Dependent Products at a Small Grocery Store*. www.kth.se/sciresearch design

[57] Rosario, J. M. (n.d.). Implementation of Inventory Control Management and Repeat Purchase in Right Goods Philippines Incorporated: Inputs to Policy Reformulation. *International Journal Peer Reviewed Journal Refereed Journal Indexed Journal Impact Factor SJIF*, 2020–2021. www.wwjmrd.com

[58] Hanna, K. T. (2022, October 17). What is a Barcode Reader (POS scanner, Barcode Scanner, Price Scanner)? WhatIs.com. Retrieved February 1, 2023, from https://www.techtarget.com/whatis/definition/barcode-reader-POS-scanner-bar-code-reader-price-scanner?amp=1

[59] D'Altorio, A. (2022, November 22). Barcode 101: Information You Need To Know. Barcode Blog. Retrieved February 1, 2023, from https://www.smithcorona.com/blog/barcode-101-information/

[60] J. F. Calderon and E. C. Gonzales, Methods of Research and Thesis Writing, vol. 59.

[61] R. C. Richey, “Developmental Research: The Definition and Scope,” 1994 Natl. Conv. teh Assoc. fr Educ. Commun. Technol., pp. 714–720, 1994, [Online]. Available: http://files.eric.ed.gov/fulltext/ED373753.pdf

ANNEX I

BILL OF MATERIALS

BILL OF MATERIALS

Item #	Item Description	Quantity	Unit Cost	Total Cost
1	Sunmi V2 Pro POS Device Wireless Handheld Thermal Printer	1	₱5,500	₱5,500
2	Thermal Printer Model: BT-58D		₱1,450	₱1,450
3	Thermal Printer XP 58IINT		₱1,750	₱,1750
4	IPO Trademark Filing	1	₱2,121	₱2,121

Total Expenses: ₱10,821

ANNEX II

SOURCE CODE

Frontend UI/UX

main.dart

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(options: DefaultFirebaseOptions.currentPlatform);
  runApp(MyApp());}
class MyApp extends StatelessWidget {

  @override
  Widget build(BuildContext context) {
    return MultiProvider(
      providers:[
        ChangeNotifierProvider<SignUpAuthProvider>(create:
            (context) => SignUpAuthProvider(),),
        ChangeNotifierProvider<LoginAuthProvider>(create:
            (context) => LoginAuthProvider(),),
        ChangeNotifierProvider<CartProvider>(create: (context) => CartProvider(),),],
      builder: (context, child) {
        return GetMaterialApp(
          debugShowCheckedModeBanner: false,
          theme: ThemeData(primaryColor: Color(0xffb7410e), primarySwatch: rust,
            scaffoldBackgroundColor: Colors.white,
            textTheme: TextTheme(headline1: TextStyle(color: Color(0xff7b3f00),
                  fontWeight: FontWeight.bold,fontSize: 20,),
                bodyText1: TextStyle(color: Color(0xffb7410e),
                        fontWeight: FontWeight.bold, fontSize: 11,),
                bodyText2: TextStyle(color: Color(0xff7b3f00),
                        fontWeight: FontWeight.bold, fontSize: 20,),
                overline: TextStyle(color: Colors.white, fontWeight: FontWeight.bold,
                        fontSize: 11,)),
            elevatedButtonTheme: ElevatedButtonThemeData(
              style: ElevatedButton.styleFrom(
                backgroundColor: Color(0xffb7410e), shape: RoundedRectangleBorder(),),),),
          home: StreamBuilder(stream: FirebaseAuth.instance.authStateChanges(),
            builder: (context, userSnp) {if (userSnp.hasData) {
                return Homepage();}
              return Home();},),);},);}}
```

landing_page.dart

```
class Home extends StatelessWidget {
  const Home({Key? key}) : super(key: key);

  @override
```

```
 Widget build(BuildContext context) { return Scaffold(
    body: SingleChildScrollView(
      child: Column(
        children: <Widget>[
          Container(alignment: Alignment.center,
            padding:  const EdgeInsets.only(left:20, right:20, top:80,),
            child: const Image( image: AssetImage("asset/images/logo-palaris.png"),),),
          SizedBox(height: 45,),
          Container(alignment: Alignment.center,
            padding: const EdgeInsets.only(left:25, right:25,),
            child: RichText(text: TextSpan(text: 'WELCOME TO PALARIS',
           style: TextStyle(color: Color(0xff7b3f00), fontWeight: FontWeight.bold,
                      fontSize: 30,),),
              textAlign: TextAlign.center,),), SizedBox(height: 30,),
          Container(alignment: Alignment.center, padding: const EdgeInsets.only(
                      left:25, right:25,),
            child: ElevatedButton (child: Text ('LOG IN', style: TextStyle(
                      color: Colors.white, fontSize: 25)),
              onPressed:() => LoginPage(),
          Container(alignment: Alignment.center, padding: const EdgeInsets.only(
                      left:25, right:25, top:10,),
            child: Row(mainAxisSize: MainAxisSize.min,
              children: [Text('Create New Account?', style: TextStyle(fontSize:20,
                      color: Color(0xff7b3f00)),),
                TextButton(onPressed: () => Signup(),
                child: Text('SIGN UP', style: TextStyle(fontSize:20,
                      color: Color(0xffb7410e)),))],),),],),),);}}
```

login_page.dart

```
class LoginPage extends StatefulWidget {
  const LoginPage ({Key? key}) : super (key: key);

  @override
  State<LoginPage> createState() => _LoginPageState();}
class _LoginPageState extends State<LoginPage> {
  TextEditingController emailAddress = TextEditingController();
  TextEditingController password = TextEditingController(); bool _isObscure = true;

  @override
  Widget build(BuildContext context) {
    LoginAuthProvider loginAuthProvider = Provider.of<LoginAuthProvider>(context);
    return Scaffold(
      resizeToAvoidBottomInset: false, backgroundColor:const Color(0xfff9f2e4),
      body: Stack(
        children: [Container(padding:  const EdgeInsets.only(left:20, right:20, top:100,),
child: const Image(image: AssetImage("asset/images/palaris-home-page.png"),),),
Container(margin: EdgeInsets.only(
      top: MediaQuery.of(context).size.height*0.35),
width: double.infinity, height: 600, decoration: const BoxDecoration(
      color: Colors.white,
 borderRadius: BorderRadius.only(topRight: Radius.circular(50),
      topLeft: Radius.circular(50)),),
  child: Column(crossAxisAlignment: CrossAxisAlignment.center,
  children:<Widget> [SizedBox(height:15,), Text('LOGIN',
      style:  TextStyle(fontSize: 45,
   fontWeight: FontWeight.bold, color: Color(0xff7b3f00)),),
   SizedBox(height: 15,),
   Padding(padding: EdgeInsets.only(left:30, right:30),
    child: TextField(controller: emailAddress,
      keyboardType: TextInputType.emailAddress,
     style:TextStyle(color:Color(0xFF7b3f00)), decoration: InputDecoration(
     icon:   Icon(Icons.mail,), labelText: 'Email', hintText: 'Enter valid email id',
      hintStyle: TextStyle(color: Color(0xFF7b3f00)),),),),
   SizedBox(height:15,),
   Padding(padding: const EdgeInsets.only(left: 30, right:30,),
    child: TextField(controller: password, style: const TextStyle(
      color: Color(0xFF7b3f00)),
     obscureText: _isObscure, decoration: InputDecoration(
      icon: new Icon(Icons.key,),
      labelText: 'Password', hintText: 'Enter Secure Password',
      hintStyle: const TextStyle(color: Color(0xFF7b3f00)),
      suffixIcon: IconButton(icon: Icon(_isObscure ? Icons.visibility_off :
       Icons.visibility),
      onPressed: (){setState(() {_isObscure =!_isObscure;});},),),),),
   TextButton(style: ButtonStyle(
      foregroundColor: MaterialStateProperty.all(Colors.white)),
    onPressed: (){}, child: const Text('Forgot Password?',
     style: TextStyle(color: Color(0xffb7410e), fontSize: 15),),),
   SizedBox(height:35,),
   Column(children: [loginAuthProvider.loading == false? Container(
         padding: const EdgeInsets.only(left:25,right:25),
         child: ElevatedButton(
          onPressed: (){loginAuthProvider.loginPageValidation(
             emailAddress: emailAddress, password: password,
             context: context,);}, child: const Text('LOG IN',
             style: TextStyle(color: Colors.white, fontSize: 25),),),)
       : const Center(child: CircularProgressIndicator(),),],),
   Container(padding: const EdgeInsets.only(left:25, right:25, ,),
    child: Row(mainAxisSize: MainAxisSize.min,
     children: [Text('Create New Account?'), TextButton(
        onPressed: () => Signup(),
                    child: Text('SIGN UP', style:TextStyle(fontSize:20,
                          color: Color(0xffb7410e))],)],);}}
```

signup.dart

```
class Signup extends StatefulWidget {
  const Signup ({Key? key}) : super (key: key);

  @override
  State<Signup> createState() => _SignupState();}
class _SignupState extends State<Signup> {
  TextEditingController storeName = TextEditingController();
  TextEditingController firstName = TextEditingController();
  TextEditingController lastName = TextEditingController();
  TextEditingController emailAddress = TextEditingController();
  TextEditingController password = TextEditingController();
  bool _isObscure = true; bool _isChecked = false;

  @override
  Widget build(BuildContext context){
    SignUpAuthProvider signUpAuthProvider =
Provider.of<SignUpAuthProvider>(context);
    return Scaffold(
      body: SingleChildScrollView(
        child: Column(crossAxisAlignment: CrossAxisAlignment.center,
          children: [const SizedBox(height: 20,),
            Container(padding:const EdgeInsets.only(left:25, right:25, top:20,),
               child: Text('SIGN UP', style: TextStyle(color: Color(0xff7b3f00),
                      fontSize: 50, fontWeight: FontWeight.bold)),),
            SizedBox(height: 10,),
            Container(padding:const EdgeInsets.only(left:25, right:25, top:0, bottom:0),
              child: const Text('User Profile',style: TextStyle(color: Color(0xff7b3f00),
                      fontSize: 30)),),
            SizedBox(height: 10,),
            Container(padding:const EdgeInsets.only(left:25, right:25, top:0, bottom:0),
              child: TextFormField(controller: storeName,
                      style: TextStyle(color: Color(0xff7b3f00)),
                decoration: const InputDecoration(icon: Icon(Icons.store),
                      labelText: 'Store Name',),
                onSaved: (String? value){}, validator: (String? value){
                  return (value != null && value.contains('@')) ?
                      'Do not use the @ char.': null;},),),
            SizedBox(height: 10,),
            Container(padding: const EdgeInsets.only(left:25, right:25,),
              child: TextFormField(controller: firstName,
                style: TextStyle(color: Color(0xff7b3f00)),
                decoration: const InputDecoration(icon: Icon(Icons.person),
            labelText: 'First Name',),
    onSaved: (String? value){}, validator: (String? value){
      return (value != null && value.contains('@')) ?
            'Do not use the @ char.': null;},),),
SizedBox(height: 10,),
Container(padding:const EdgeInsets.only(left:25, right:25, top:0, bottom:0),
  child: TextFormField(controller: emailAddress,
            keyboardType: TextInputType.emailAddress,
    style: TextStyle(color: Color(0xff7b3f00)), decoration: InputDecoration(
        icon: Icon(Icons.mail),
        labelText: 'Email', hintText: 'Enter valid email id',
        hintStyle: TextStyle(color: Color(0xFF7b3f00)),),),),
SizedBox(height: 10,),
Container(padding: const EdgeInsets.only(left: 25, right:25,),
  child: TextFormField(obscureText: _isObscure, controller: password,
    style: TextStyle(color: Color(0xff7b3f00)), decoration: InputDecoration(
      icon: Icon(Icons.key),
      labelText: 'Password', hintText: 'Enter secure password',
      hintStyle: TextStyle(color: Color(0xFF7b3f00)),
      suffixIcon: IconButton(icon: Icon(_isObscure ? Icons.visibility_off :
            Icons.visibility),
          onPressed: (){setState(() {_isObscure =!_isObscure;});},),),),),
SizedBox(height: 10,),
Container(padding: const EdgeInsets.only(left:20, right:20,),
  child: Row(mainAxisSize: MainAxisSize.min,
    children: [Checkbox(value: _isChecked, onChanged: (value) {
          setState(() {_isChecked = value ?? false;});},
            checkColor: Color(0xffb7410e),),
      Expanded(child: TextButton(onPressed: () => Terms(),
            child: Center(
                child: Text('I agree to the Terms of Services & Privacy Policy.',
                    style:TextStyle(fontSize:15, color: Color(0xffb7410e)),),)),)],),),
SizedBox(height: 20,),
Column(children: [signUpAuthProvider.loading == false ? Container(
          padding: const EdgeInsets.only(left:25, right:25,),
          child: ElevatedButton(onPressed: (){
              signUpAuthProvider.signUpValidation(
                storeName: storeName, firstName: firstName, lastName: lastName,
                emailAddress: emailAddress, password: password, context: context,);},
            child: Text('SIGN UP', style: TextStyle(color: Colors.white,
                fontSize: 25)),),),)
        : const Center(child: CircularProgressIndicator(),),],),
SizedBox(height: 5,),
Container(padding: const EdgeInsets.only(left:25,right:25,),
  child: Row(mainAxisSize: MainAxisSize.min,
    children: [Text('Already have an account?', style:TextStyle(fontSize:16,)),
         TextButton(onPressed: () => LoginPage(),
           child: Text('LOG IN', style:TextStyle(fontSize:16,
             color: Color(0xffb7410e)),))],),),],),),),);}}
```

business_page.dart

```
class BusinessPage extends StatefulWidget {
  const BusinessPage ({Key? key}) : super (key: key);

  @override
  State<BusinessPage> createState() => _BusinessPageState();}
class _BusinessPageState extends State<BusinessPage> {
  TextEditingController storeName = TextEditingController();
  TextEditingController address = TextEditingController();
  TextEditingController tin = TextEditingController();

  @override
  Widget build(BuildContext context){
    return Scaffold(
      appBar: AppBar(elevation: 0, backgroundColor: Colors.white,
        actions: <Widget>[TextButton(child: Text('SKIP'),
          onPressed: () => BannerPage(),),],),
      body:SingleChildScrollView(
        child: Column(crossAxisAlignment: CrossAxisAlignment.center,
          children: <Widget>[SizedBox(height: 40,),
            Container(padding:const EdgeInsets.only(left:25, right:25,),
              child: const Text('Business Profile',style: TextStyle(color: Color(0xff7b3f00),
                  fontSize: 35)),),
            SizedBox(height: 10,),
            Container(padding:EdgeInsets.only(left: 15, right: 15 ),
              child: RichText(text: TextSpan(
                  text: 'The following information are required for the receipt.',
                style: TextStyle(color: Color(0xff7b3f00), fontWeight: FontWeight.normal,
                  fontSize: 18,),),
                textAlign: TextAlign.center,), alignment: Alignment.centerLeft,),
            SizedBox(height: 40,),
            Container(padding:const EdgeInsets.only(left:25, right:25, top:0, bottom:0),
              child: TextFormField(controller: storeName,
                style: TextStyle(color: Color(0xff7b3f00)),
                decoration: const InputDecoration(icon: Icon(Icons.storefront),
                  labelText: 'Store Name:',),
                onSaved: (String? value){}, validator: (String? value){
                  return (value != null && value.contains('@')) ?
                  'Do not use the @ char.': null;},),),
            SizedBox(height: 10,),
            Container(padding:const EdgeInsets.only(left:25, right:25,),
              child: TextFormField(controller: address,
                keyboardType: TextInputType.streetAddress,
                style: TextStyle(color: Color(0xff7b3f00)),
              decoration: const InputDecoration(icon: Icon(Icons.pin_drop),
                labelText: 'Address:',
                hintText: 'Street No, Barangay, City, Province',
                hintStyle: TextStyle(color: Color(0xFF7b3f00)),),
                onSaved: (String? value){}, validator: (String? value){
                  return (value != null && value.contains('@')) ?
                  'Do not use the @ char.': null;},),),
            SizedBox(height: 10,),
            Container(padding:const EdgeInsets.only(left:25, right:25,),
              child: TextFormField(controller: tin, keyboardType: TextInputType.number,
                style: TextStyle(color: Color(0xff7b3f00)),
                decoration: InputDecoration(icon: Icon(Icons.account_balance),
                  labelText: 'TIN No:',
                  hintText: 'Put - if none', hintStyle: TextStyle(color: Color(0xFF7b3f00)),),),),
            SizedBox(height:40,),
            Container(padding: const EdgeInsets.only(left:25, right:25, top:50, ,),
              child: ElevatedButton(onPressed: (){saveDataToFirestore();
                  Navigator.push(context, MaterialPageRoute(builder: (_)=> BannerPage()));},
                child: Text('PROCEED', style: TextStyle(color: Colors.white,
                  fontSize: 25),),),),],),),);}}
```

banner.dart

```
class BannerPage extends StatefulWidget {
  const BannerPage({Key? key}) : super(key: key);

  @override
  _BannerPageState createState() => _BannerPageState();}
class _BannerPageState extends State<BannerPage> {
  int _currentImageIndex = 0;
  final List<String> bannerImages = [
          "asset/images/Banner1.png", "asset/images/Banner2.png",
          "asset/images/Banner3.png", "asset/images/Banner4.png",];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: LayoutBuilder(builder: (context, constraints) {
          return ListView(children: [CarouselSlider(items: bannerImages.map((imagePath) {
                return Container(margin: EdgeInsets.all(6.0), decoration: BoxDecoration(
                    borderRadius: BorderRadius.circular(8.0),image: DecorationImage(
                      image: AssetImage(imagePath),fit: BoxFit.cover,),),);}).toList(),
              options: CarouselOptions(height: constraints.maxHeight * 0.7,
                enlargeCenterPage: true, enableInfiniteScroll: true,
                autoPlay: true, aspectRatio: 16 / 9, autoPlayCurve: Curves.fastOutSlowIn,
                autoPlayAnimationDuration: Duration(milliseconds: 800),
                viewportFraction: 0.8, onPageChanged: (index, reason) {
                  setState(() {_currentImageIndex = index;});},),),
            Row(mainAxisAlignment: MainAxisAlignment.center,
              children: bannerImages.map((imagePath) {
                      int index = bannerImages.indexOf(imagePath);
                return Container(width: 8, height: 8,
                  margin: EdgeInsets.symmetric(horizontal: 4),
                  decoration: BoxDecoration(shape: BoxShape.circle,
                      color: _currentImageIndex == index
                        ? Color(0xff7b3f00) : Colors.grey,),);}).toList(),),
            Container(alignment: Alignment.center, padding: EdgeInsets.all(16.0),
              child: TextButton(child: Text('GET STARTED',
                  style: TextStyle(fontSize: 18),), onPressed: () => StartBanner(),),],);},),);}}
```

start_banner.dart

```
class StartBanner extends StatefulWidget {
  const StartBanner ({Key? key}) : super (key: key);

  @override
  State<StartBanner> createState() => _StartBannerState();}
class _StartBannerState extends State<StartBanner> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      resizeToAvoidBottomInset: false, backgroundColor:const Color(0xfff9f2e4),
      body: Stack(
        children: [Container(padding: EdgeInsets.only(left:20, right:20, top:100,),
            child: const Image(image: AssetImage("asset/images/palaris-home-page.png"),),),
          SizedBox(height: 15,),
          Container(margin: EdgeInsets.only(top: MediaQuery.of(context).size.height*0.35),
            width: double.infinity, height: 600, decoration: const BoxDecoration(
            color: Colors.white, borderRadius: BorderRadius.only(topRight:
                  Radius.circular(50), topLeft: Radius.circular(50)),),
            child: Column(crossAxisAlignment: CrossAxisAlignment.center,
              children:<Widget> [SizedBox(height:150,),
                Text('Start your POS, Now!', style:  TextStyle(fontSize: 35,
                    fontWeight: FontWeight.bold,
                color: Color(0xff7b3f00)),textAlign: TextAlign.center,),
                SizedBox(height:20,),
                Container(padding: const EdgeInsets.only(left:25,right:25),
                  child: ElevatedButton(onPressed: () => AddProduct(),
                    child: Text('Add items', style: TextStyle(color: Colors.white,
                    fontSize: 25),),),), SizedBox(height: 15,),
                Container(padding: const EdgeInsets.only(left:25,right:25,),
                  child: Row(mainAxisSize: MainAxisSize.min,
                    children: [TextButton(onPressed: () => Homepage(),
                        child: Text('SKIP',style:TextStyle(fontSize:20,
                            color: Color(0xffb7410e))))],),),],),),],),);}}
```

homepage.dart

```
class Homepage extends StatefulWidget{
  const Homepage({Key? key}) : super(key: key);

  @override
  State<Homepage> createState() => _HomepageState();}
class _HomepageState extends State<Homepage> {

  @override
  void initState() {getCurrentUserDataFunction(); super.initState();}

  @override
  Widget build(BuildContext context){
    return Scaffold(
      appBar: AppBar(elevation: 1,
        backgroundColor: const Color(0xffffffff),
        title: Text("Welcome!", style: Theme.of(context).textTheme.headline1,),
            leading: Builder(builder: (BuildContext context){
          return IconButton(iconSize: 40, color: Theme.of(context).primaryColor,
            icon:  const Icon(Icons.menu,),
          onPressed:(){Scaffold.of(context).openDrawer();},
          tooltip: MaterialLocalizations.of(context).openAppDrawerTooltip,);}),),
      drawer: Drawer(
        child: Container(padding: const EdgeInsets.all(24), color: const Color(0xffffffff),
          child: SingleChildScrollView(
            child: Wrap(runSpacing: 10,
              children: [DrawerHeader(child: Center(child: Container(width: 130, height: 130,
                    decoration: BoxDecoration(border: Border.all(width: 4,
                        color: Theme.of(context).primaryColor,),
                      boxShadow: [BoxShadow(spreadRadius: 2, blurRadius: 10,
                          color: Colors.black.withOpacity(0.1), offset: const Offset(0, 10),),],
                      shape: BoxShape.circle, image: DecorationImage(fit: BoxFit.cover,
                        image: AssetImage("asset/images/logo-palaris.png"),),),),),),),
                ListView(shrinkWrap: true, physics: NeverScrollableScrollPhysics(),
                  scrollDirection: Axis.vertical,
                  children: [
                    ListTile(leading: Icon(Icons.person, size: 25, color: Color(0xff7b3f00)),
                      title: Text('Profile', style: Theme.of(context).textTheme.displayLarge,),
                      onTap: () => Profile(),),
              ListTile(leading: Icon(Icons.settings, size: 25, color: Color(0xff7b3f00)),
                title: Text('Settings', style: Theme.of(context).textTheme.displayLarge,),
                onTap: () => SettingsPage(),),
              ListTile(leading: Icon(Icons.scanner_rounded, size: 25,
                color: Color(0xff7b3f00)),
                title: Text('Barcode', style: Theme.of(context).textTheme.displayLarge,),
                onTap: () => Barcode(),),
              ListTile(leading: Icon(Icons.print, size: 25, color: Color(0xff7b3f00)),
                title: Text('Devices', style: Theme.of(context).textTheme.displayLarge,),
                onTap: () => Devices (),),
              ListTile(leading: Icon(Icons.auto_stories_outlined, size: 25,
                color: Color(0xff7b3f00)),
                title: Text('About', style: Theme.of(context).textTheme.displayLarge,),
                onTap: () => About(),),
              ListTile(leading: Icon(Icons.description_outlined, size: 25,
                color: Color(0xff7b3f00)),
                title: Text('Terms of Services & Privacy Policy',
                style: Theme.of(context).textTheme.displayLarge,),
                onTap: () => Terms(),),
              ListTile(leading: Icon(Icons.report_gmailerrorred, size: 25,
                color: Color(0xff7b3f00)),
                title: Text('Report A Problem',
                  style: Theme.of(context).textTheme.displayLarge,),
                onTap: () => Report(),),
              SizedBox(height: 10),
              Center(child: ElevatedButton(
                  onPressed: () {
                     showDialog(context: context, builder: (BuildContext context) {
                        return SimpleDialog(title: Text("Are you sure you want to logout?"),
                          children: <Widget>[
              Row(mainAxisAlignment: MainAxisAlignment.spaceBetween,
                        children: [SimpleDialogOption(onPressed: handleLogout,
                    child: const Text('YES'),),
                          SimpleDialogOption(onPressed: () {Navigator.pop(context);},
                            child: const Text('CANCEL'),),],));},);},
                  child: const Text('LOG OUT', style: TextStyle(color: Colors.white,
                      fontSize: 25),),),),],),),),
body: SingleChildScrollView(
  child: Column(children: <Widget>[SizedBox(height: 10,),
      Container(padding:  const EdgeInsets.only(left:20, right:20, top:10, bottom:0),
        child: const Image(
            image: AssetImage("asset/images/palaris-home-page.png"),),),
      SizedBox(height: MediaQuery.of(context).size.height*0.10,),
      Container(margin: const EdgeInsets.only(left: 15, right: 15),
        padding: EdgeInsets.only(top: 15, bottom: 15),
        decoration: BoxDecoration(color:const Color(0xfff5e5de),
     border: Border.all(color:const Color(0xfff5e5de),),
     borderRadius: BorderRadius.only(topRight: Radius.circular(15),
       topLeft: Radius.circular(15), bottomLeft: Radius.circular(15),
       bottomRight: Radius.circular(15),),),
   child: Column(children: [
 Row(mainAxisAlignment: MainAxisAlignment.spaceEvenly,
         mainAxisSize: MainAxisSize.max,
         children: [Container(padding: EdgeInsets.only(top:10), height: 120,
             width: MediaQuery.of(context).size.height*0.14,
             decoration: BoxDecoration(color:const Color(0xfff5e5de),
             border: Border.all(color: Color(0xfff5e5de),), borderRadius:
               BorderRadius.all(Radius.circular(15) ,),),),
             child: Column(children: [
                 IconButton(color: Theme.of(context).primaryColor,
                     iconSize: 60, padding: const EdgeInsets.only(left:5.0, right:5),
                     onPressed: () => Products()));}, icon:const Icon(Icons.list_alt)),
                 RichText(text: TextSpan(text: 'PRODUCTS',
                   style: Theme.of(context).textTheme.bodyLarge,),),],),),
           Container(padding: EdgeInsets.only(top:10), height: 120,
             width: MediaQuery.of(context).size.height*0.14,
             decoration: BoxDecoration(color:const Color(0xfff5e5de),
               border: Border.all(color: Color(0xfff5e5de),),
                 borderRadius: BorderRadius.only(
                   topRight: Radius.circular(15), topLeft: Radius.circular(15),
                   bottomLeft: Radius.circular(15),bottomRight: Radius.circular(15),),),
             child: Column(children: [
                 IconButton(color: Theme.of(context).primaryColor,
                     iconSize: 60, padding: const EdgeInsets.only(left:5.0, right:5),
                     onPressed: () => Inventory(), icon:const Icon(Icons.inventory)),
                 RichText(text: TextSpan(text: 'INVENTORY',
                   style: Theme.of(context).textTheme.bodyLarge,),),],    ),  ),
           Container(padding: EdgeInsets.only(top:10), height: 120,
             width: MediaQuery.of(context).size.height*0.14,
             decoration: BoxDecoration(color:const Color(0xfff5e5de),
               border: Border.all(color: Color(0xfff5e5de),),
                borderRadius: BorderRadius.only(
                  topRight: Radius.circular(15), topLeft: Radius.circular(15),
                  bottomLeft: Radius.circular(15),bottomRight: Radius.circular(15),),),
             child: Column(children: [
                 IconButton(color: Theme.of(context).primaryColor, iconSize: 60,
                     padding: const EdgeInsets.only(left:5.0, right:5),
                     onPressed: () => HotItems(), icon: Icon(Icons.fire)),
                 RichText(text: TextSpan(text: 'HOT ITEMS',
                   style: Theme.of(context).textTheme.bodyLarge,),),],),),],),
     SizedBox(height: 15,),
     Row(mainAxisAlignment: MainAxisAlignment.spaceEvenly,
              mainAxisSize: MainAxisSize.max,
              children: [Container(padding: EdgeInsets.only(top:10), height: 120,
                width: MediaQuery.of(context).size.height*0.14,
                decoration: BoxDecoration(color:const Color(0xfff5e5de),
                border: Border.all(color: Color(0xfff5e5de),),
                  borderRadius: BorderRadius.only(
                     topRight: Radius.circular(15), topLeft: Radius.circular(15),
                     bottomLeft: Radius.circular(15),bottomRight: Radius.circular(15),),),
                child: Column(children: [
                    IconButton(color: Theme.of(context).primaryColor,
                     iconSize: 60, padding: const EdgeInsets.only(left:5, right:5),
                        onPressed: () => Transactions(), icon:const Icon(Icons.receipt_long)),
                    RichText(text: TextSpan(text: 'TRANSACTIONS',
                      style: Theme.of(context).textTheme.bodyLarge,),
                      textAlign: TextAlign.center,),],),),
               Container(padding: EdgeInsets.only(top:10), height: 120,
                width: MediaQuery.of(context).size.height*0.14,
                decoration: BoxDecoration(color:const Color(0xfff5e5de),
                border: Border.all(color: Color(0xfff5e5de),),
                  borderRadius: BorderRadius.only(
                    topRight: Radius.circular(15), topLeft: Radius.circular(15),
                    bottomLeft: Radius.circular(15),bottomRight: Radius.circular(15),),),
                child: Column(children: [
                    IconButton(color: Theme.of(context).primaryColor,
                        iconSize: 60, padding: const EdgeInsets.only(left:5.0, right:5),
                        onPressed: () => SalesReport(), icon: Icon(Icons.line_chart)),
                    RichText(text: TextSpan(text: 'SALES REPORT',
                      style: Theme.of(context).textTheme.bodyLarge,),
                      textAlign: TextAlign.center,),],),),
               Container(padding: EdgeInsets.only(top:10), height: 120,
                width: MediaQuery.of(context).size.height*0.14,
                decoration: BoxDecoration(color:const Color(0xfff5e5de),
                 border: Border.all(color: Color(0xfff5e5de),),
                   borderRadius: BorderRadius.only(
                     topRight: Radius.circular(15), topLeft: Radius.circular(15),
                     bottomLeft: Radius.circular(15),bottomRight: Radius.circular(15),),),
                child: Column(children: [
                    IconButton(color: Theme.of(context).primaryColor,
                        iconSize: 60, padding: const EdgeInsets.only(left:5, right:5),
                        onPressed: () => Analytics(), icon: Icon(Icons.analytics_outlined)),
                    RichText(text: TextSpan(text: 'ANALYTICS',
                     style: Theme.of(context).textTheme.bodyLarge,),
                     textAlign: TextAlign.center,),],),],),],),
floatingActionButtonLocation: FloatingActionButtonLocation.centerDocked,
floatingActionButton: FloatingActionButton(onPressed: () => const AddProduct()
  backgroundColor: const Color(0xffffc000), foregroundColor: Color(0xffa81c07),
    child: const Icon(Icons.add),),
   bottomNavigationBar: BottomAppBar(notchMargin: 0.5,
    shape: const CircularNotchedRectangle(),
    color: const Color(0xffb7410e),
    child: Container(padding: const EdgeInsets.only(left:40, right: 40), height: 60,
      child: Row(mainAxisSize: MainAxisSize.max,
       mainAxisAlignment: MainAxisAlignment.spaceBetween,
       children: <Widget>[MaterialButton(minWidth: 50, onPressed: (){},
         child:Column(mainAxisAlignment: MainAxisAlignment.center,
          children: const <Widget>[Icon(Icons.home, color: Colors.white,),
           Text('HOME', style: TextStyle(color: Colors.white),),],),),
        MaterialButton(minWidth: 50,
         onPressed: () => Pos(),
         child:Column(mainAxisAlignment: MainAxisAlignment.center,
          children: const <Widget>[Icon(Icons.point_of_sale, color: Colors.grey,),
           Text('POS', style: TextStyle(color: Colors.grey),),],),),],),),),);}}
```

add_product.dart

```
class AddProduct extends StatefulWidget{
 const AddProduct ({Key? key}) : super(key: key);

 @override
 State<AddProduct> createState() => _AddProductState();}
class _AddProductState extends State<AddProduct> {
 double screenHeight=0; double screenWidth=0;
 String? dropDownValue; String? unitValue;
 String _productCode = ‘’; String _productImage = ‘’;
 int _productStocks = 0; int _productLowQuantity = 0;
 DateTime _selectedDate = DateTime.now();
 final int _productQuantity = 0; final int _soldCount = 0;
 final TextEditingController _productName = TextEditingController();
 final TextEditingController _productCost = TextEditingController();
 final TextEditingController _productPrice = TextEditingController();

 @override
 Widget build(BuildContext context){
  screenHeight = MediaQuery.of(context).size.height;
  screenWidth  = MediaQuery.of(context).size.width;
  return Scaffold(
   appBar: AppBar(elevation: 1, backgroundColor: const Color(0xffffffff),
    title: Text(‘Add Product’, style: Theme.of(context).textTheme.headline1,),
    leading: Builder(builder: (BuildContext){
     return IconButton(color: Theme.of(context).primaryColor,
        icon: Icon(Icons.arrow_back), onPressed: () => Homepage(),);},),),
   body: Container(padding: const EdgeInsets.only(left:16, right: 16),
child: SingleChildScrollView(
  child: Column(children: [const SizedBox(height: 10,),
      GestureDetector(onTap: () {
        showDialog(context: context, builder: (BuildContext context){
                return SimpleDialog(children: <Widget>[
                    SimpleDialogOption(
                      onPressed: () {imgFromGallery(); Navigator.of(context).pop();},
                      child: const Text('Pick From Gallery'),),
                    SimpleDialogOption(
                      onPressed: () {imgFromCamera(); Navigator.of(context).pop();},
                      child: const Text('Take A New Picture'),),],);});},
        child: CircleAvatar(
          radius: 55, backgroundColor: Theme.of(context).primaryColor,
          child: _photo != null ? ClipRRect(borderRadius: BorderRadius.circular(50),
            child: Image.file(_photo!, width: 100, height: 100, fit: BoxFit.fitHeight,),)
              : Container(decoration: BoxDecoration(color: Colors.grey[200],
                borderRadius: BorderRadius.circular(50)), width: 100, height: 100,
            child: Icon(Icons.camera_alt,color: Colors.grey[800],),),),),
      const SizedBox(height: 10,),
      Center(child: Column(mainAxisAlignment: MainAxisAlignment.center,
          children: [DropdownButton<String>(
                value: dropDownValue, hint: const Text('Select Category',
                style:TextStyle(fontSize: 15, color: Color(0xff7b3f00),)),
              icon: const Icon(Icons.keyboard_arrow_down),
              items: <String>["ALL", "Baking & Cooking", "Beverages",
                "Bread & Breakfast", "Canned Foods", "Coffee, Tea & Cocoa",
                "Dairy & Eggs", "Household", "Instant & Ready-to-eat",
                "Personal Care & Beauty", "Rice, Pasta & Grains",
                "Snacks & Sweets", "Spread, Jams & Honey", "Others"
              ].map<DropdownMenuItem<String>>((String value){
                return DropdownMenuItem<String>(value: value,
                  child: Text(value,
                    style: TextStyle(fontSize: 13, color: Color(0xff7b3f00),),),);}).toList(),
              onChanged: (newValue){
                setState(() {dropDownValue = newValue!;});},),  ],  ),),
      const SizedBox(height: 10,),
      Container(height: 50, child: SfBarcodeGenerator(
                value: _productCode, symbology: Codabar(),),),
      Center(
        child:  Row(mainAxisAlignment: MainAxisAlignment.center,
            mainAxisSize:MainAxisSize.max,
          children: [RichText(text: TextSpan(text: 'Product Code: ',style: TextStyle(
              color: Color(0xff7b3f00), fontWeight: FontWeight.bold, fontSize: 16,),),),
            RichText(text: TextSpan(text: _productCode,
              style: TextStyle(color: Color(0xff7b3f00),
              fontWeight: FontWeight.bold, fontSize: 16,),),),],),),
Center(
  child: Row(mainAxisAlignment: MainAxisAlignment.spaceEvenly,
    mainAxisSize: MainAxisSize.max,
    children: [OutlinedButton(
          onPressed: () => scanProductBarcode(), child: Text('SCAN')),
      OutlinedButton(
          onPressed: (){setState(() {_productCode = randomNumber();});},
          child: Text('Generate Code')),],),),
Center(child: Row(mainAxisAlignment: MainAxisAlignment.spaceEvenly,
    mainAxisSize: MainAxisSize.max,
    children: [OutlinedButton(onPressed: printSunmiBarcode,
          child: Text('Sunmi Printer')),
      OutlinedButton(child: Text('Thermal Printer'), onPressed:  _connected?()
          async {Map<String, dynamic> config = Map(); config['width'] = 40;
            config['height'] = 90; config['gap'] = 2;
            List<LineText> list = [];
            list.add(LineText(type: LineText.TYPE_TEXT,
                content: _productName.text, weight: 2, width: 2, height: 1,
                align: LineText.ALIGN_CENTER, linefeed: 1,),);
            list.add(LineText(type: LineText.TYPE_BARCODE,
                content:_productCode,
                size:25, align: LineText.ALIGN_CENTER, linefeed: 1));
            await bluetoothPrint.printLabel(config, list);}:null),],),),
TextFormField(controller: _productName,
  style: TextStyle(color:Color(0xFF7b3f00)),
  decoration: const InputDecoration(icon:   Icon(Icons.shopping_basket,),
    labelText: 'Product Name', hintText: 'ex. Pancit Canton',
    hintStyle: TextStyle(color: Color(0xFF7b3f00)),),),
TextFormField(controller: _productCost,
  style: TextStyle(color:Color(0xFF7b3f00)),
  keyboardType: const TextInputType.numberWithOptions(decimal: true),
  decoration: const InputDecoration(icon:   Icon(Icons.currency_ruble,),
    labelText: 'Cost', hintText: 'Original Price',
    hintStyle: TextStyle(color: Color(0xFF7b3f00)),),),
TextFormField(controller: _productPrice,
  style:const TextStyle(color:Color(0xFF7b3f00)),
  keyboardType: const TextInputType.numberWithOptions(decimal: true),
  decoration: const InputDecoration(icon:   Icon(Icons.currency_ruble,),
    labelText: 'Price', hintText: 'Selling Price',
    hintStyle: TextStyle(color: Color(0xFF7b3f00)),),),
Row(children: [
     RichText(text: TextSpan(text: 'Unit of Measurement:',
      style: TextStyle(
      color: Color(0xff7b3f00),fontWeight: FontWeight.bold, fontSize: 15,),),
      textAlign: TextAlign.center,),
    SizedBox(width: 10,),
    DropdownButton(value:unitValue, hint: const Text(‘Select Unit’,
          style:TextStyle(fontSize: 15, color: Color(0xff7b3f00),)),
        icon: const Icon(Icons.keyboard_arrow_down),
        items: <String>[
          “Per piece(pc)”, “Grams (g)”, “Milligrams (mg)”, “Kilogram (kg)”,
          “Milliliters (ml)”, “Liters (l)”, “Meters (m)”, “Pack”, “Box”, “Dozen”,
        ].map<DropdownMenuItem<String>>((String value){
          return DropdownMenuItem<String>(value:value,
            child: Text(value,style:const TextStyle(fontSize: 13,
        color: Color(0xff7b3f00),),),);}).toList(),
        onChanged: (newValue){setState(() {unitValue = newValue!;});}),],),
SizedBox(height: 10,),
Container(padding: const EdgeInsets.only(right:25, left: 25),
  child: Row(mainAxisSize: MainAxisSize.max,
    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    children: [RichText(text: const TextSpan(text: ‘Stocks Available’,
        style: TextStyle(color: Color(0xff7b3f00), fontWeight: FontWeight.bold,
          fontSize: 16,),),),
      RichText(text: const TextSpan(text: ‘Low Quantity’,
        style: TextStyle(color: Color(0xff7b3f00), fontWeight: FontWeight.bold,
          fontSize: 16,),),),],),
Container(padding:const EdgeInsets.only(right:30, left: 30),
  child: Row(mainAxisSize: MainAxisSize.max,
    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    children: [NumericStepButton(maxValue: 1000,
        onChanged: (value) {_productStocks = value;},),
      NumericStepButton(maxValue: 1000,
        onChanged: (value) {_productLowQuantity = value;},)],),),
const SizedBox(height: 10,),
Row(mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: [TextButton(onPressed: (){Get.back();},
      child: const Text(‘CANCEL’,
          style: TextStyle(fontSize: 14, letterspacing: 2.2,),),),
    ElevatedButton(onPressed: () async {
        bool productCodeExists = await checkIfProductCodeExists(_productCode);
        if (productCodeExists) {
          ScaffoldMessenger.of(context).showSnackBar(
            SnackBar(content: Text(‘The product already exists.’),
              duration: const Duration(seconds: 3),),);
        } else {Database.addProduct(ProductModel(
              productUnit: unitValue.toString(), productCode: _productCode,
              productName: _productName.text.trim(),
              productCategory: dropDownValue.toString(),
              productImage: _productImage.toString(),
              productCost: double.parse(_productCost.text), expDate: _selectedDate,
              productPrice: double.parse(_productPrice.text),
                    productStocks: int.parse(_productStocks.toString()),
                    productLowQuantity: int.parse(_productLowQuantity.toString()),
                    productQuantity: int.parse(_productQuantity.toString()),
                    soldCount: int.parse(_soldCount.toString()),), context,);
                  Get.to(() => const Homepage());}},
                child: const Text(‘Add Item’,
                  style: TextStyle(fontSize: 14,letterspacing: 2.2,),),)],),],),),),);}}
```

barcode.dart

```
class Barcode extends StatefulWidget{
  const Barcode({super.key});

  @override
  State<Barcode> createState() => _BarcodeState();}
class _BarcodeState extends State<Barcode> {
  TextEditingController itemName = TextEditingController();
  String valueText = '';
  String itemCode = '';
  BluetoothPrint bluetoothPrint = BluetoothPrint.instance;
  bool _connected = false;
  BluetoothDevice? _device;
  String tips = 'No Device Connect';
  List<BluetoothDevice> _devices = [];
  String _deviceMsg ="";
  final f = NumberFormat("\$#####.00","en_US");

  @override
  void initState(){super.initState();
    WidgetsBinding.instance.addPostFrameCallback((_)=>{initPrinter()});}

  @override
  Widget build(BuildContext context){
    return Scaffold(
        appBar: AppBar(elevation: 1, backgroundColor: Color(0xffffffff),
          title: Text('Barcode Generator',style: Theme.of(context).textTheme.headline1,),
          leading: Builder(builder: (BuildContext){
            return IconButton(color: Theme.of(context).primaryColor,
              icon: const Icon(Icons.arrow_back),
              onPressed: () => Homepage()));},);},),),
        body:SingleChildScrollView(
          child: Column(children: [
              Container(padding: EdgeInsets.all(10),
                child: Column(children: [Row(
                      children: [
                        Container(child: RichText(text: TextSpan(text: 'Steps to use',
          style: TextStyle(color: Color(0xff7b3f00),
            fontWeight: FontWeight.normal, fontSize: 18,),),),
        alignment: Alignment.centerLeft,),
      Container(child: RichText(text: TextSpan(text: ' Barcode Generator',
          style: TextStyle(color: Color(0xff7b3f00),
            fontWeight: FontWeight.bold, fontSize: 18,),),),
        alignment: Alignment.centerLeft,),],),
      SizedBox(height: 10,),
      Container(child: RichText(text: TextSpan(text: '1.  To generate a unique
          barcode, first, type the name of the item/ product.',
        style: TextStyle(color: Color(0xff7b3f00),
        fontWeight: FontWeight.normal, fontSize: 16,),),),
       alignment: Alignment.centerLeft,),
    Container(child: RichText(text: TextSpan(text: '2. Press Generate Barcode.',
      style: TextStyle(color: Color(0xff7b3f00),
        fontWeight: FontWeight.normal, fontSize: 16,),),),
      alignment: Alignment.centerLeft,),
    Container(child: RichText(text: TextSpan(text: '3. Press Print Barcode',
      style: TextStyle(color: Color(0xff7b3f00),
        fontWeight: FontWeight.normal,fontSize: 16,),),),
      alignment: Alignment.centerLeft,),
    Container(child: RichText(text: TextSpan(text: 'Note: Make sure that your
          device is properly connected into a printer.',
      style: TextStyle(color: Color(0xff7b3f00),
        fontStyle: FontStyle.italic, fontSize: 16,),),),
    alignment: Alignment.centerLeft,),],),),
SizedBox(height: 10,), Divider(), SizedBox(height: 10,),
Container(margin: EdgeInsets.only(left: 20,right: 20),
  padding: EdgeInsets.all(10), height: 60, width: double.infinity,
  decoration: BoxDecoration(color: Color(0xfff9f2e4),
    border: Border.all(width: 5, color: Color(0xffffc000),),
    borderRadius: BorderRadius.all(Radius.circular(5)),),
  child: Text('Item: $valueText'), alignment: Alignment.center,),
SizedBox(height: 10,),
Container(height: 80, child: Container(width: 300,
    child: SfBarcodeGenerator(value: itemCode, symbology: Codabar(),
      showValue: true,),),),
SizedBox(height: 5,),
Padding(padding: EdgeInsets.only(left:30, right:30),
  child: TextFormField(controller: itemName,
    style:TextStyle(color:Color(0xFF7b3f00)),
    decoration: InputDecoration(icon:   Icon(Icons.shopping_cart,),
      labelText: 'Type item name here',),),),
Row(mainAxisAlignment: MainAxisAlignment.spaceBetween,
  mainAxisSize: MainAxisSize.min,
  children: [OutlinedButton(
              onPressed: (){setState(() {
                valueText = itemName.text; itemCode=randomNumber();});},
              child: Text('Generate Barcode')),
            SizedBox(width: 5,),],),
          SizedBox(height: 10,),
          ElevatedButton(
            child: Text('PRINT BARCODE'),
            onPressed: printSunmiBarcode,),
          SizedBox(height: 100,),],),));}}
```

products.dart

```
class Products extends StatelessWidget{
  Products({super.key});

  @override
  Widget build(BuildContext context){
    return Scaffold(
      appBar: AppBar(elevation: 1,backgroundColor: const Color(0xffffffff),
        title: Text('Products', style: Theme.of(context).textTheme.headline1,),
        leading: Builder(builder: (BuildContext){
          return IconButton(color: Theme.of(context).primaryColor,
            icon: const Icon(Icons.arrow_back),
            onPressed: () => Homepage()));},);  },),
        actions: [Builder(builder: (BuildContext) {
            return IconButton(color: Theme.of(context).primaryColor,
              icon: const Icon(Icons.add), onPressed: () => AddProduct()));},);},),],),
      body: SingleChildScrollView(child: Column(children: [SizedBox(height: 10,),
            Container(padding:EdgeInsets.only(left: 15, right: 15, ),
              child: RichText(text: TextSpan(text: 'CATEGORIES',
                style: TextStyle(color: Color(0xff7b3f00),
                fontWeight: FontWeight.bold, fontSize: 30,),),
                textAlign: TextAlign.center,),), SizedBox(height: 10,),
            Container(
              padding: EdgeInsets.only(left: 15, right: 15,),
              child: Column(
                children: [
                  ListTile(shape: RoundedRectangleBorder(side: BorderSide(width: 2,
                      color:Color(0xffFDB73E)),borderRadius: BorderRadius.circular(5)),
                      title: Text('All', style: Theme.of(context).textTheme.headline1,),
                      onTap: () =>const AllProducts(),),);}),SizedBox(height: 4,),
                  ListTile(shape: RoundedRectangleBorder(side: BorderSide(width: 2,
                      color:Color(0xffFDB73E)),borderRadius: BorderRadius.circular(5)),
                      title: Text('Baking & Cooking', style:
                      Theme.of(context).textTheme.headline1,),
                      onTap: () =>const BakingAndCooking(),),);}),SizedBox(height: 4,),
                  ListTile(shape: RoundedRectangleBorder(side: BorderSide(width: 2,
   color:Color(0xffFDB73E)),borderRadius: BorderRadius.circular(5)),
   title: Text('Beverages', style: Theme.of(context).textTheme.headline1,),
   onTap: () =>const Beverages(),),);}), SizedBox(height: 4,),
ListTile(shape: RoundedRectangleBorder(side: BorderSide(width: 2,
   color:Color(0xffFDB73E)), borderRadius: BorderRadius.circular(5)),
   title: Text('Bread & Breakfast',style:
   Theme.of(context).textTheme.headline1,),
   onTap: () =>const BreadAndBreakfast(),),);}),SizedBox(height: 4,),
ListTile(shape: RoundedRectangleBorder(side: BorderSide(width: 2,
   color:Color(0xffFDB73E)), borderRadius: BorderRadius.circular(5)),
   title: Text('Canned Goods',style: Theme.of(context).textTheme.headline1,),
   onTap: () =>const CannedGoods(),),);}), SizedBox(height: 4,),
ListTile(shape: RoundedRectangleBorder(side: BorderSide(width: 2,
   color:Color(0xffFDB73E)), borderRadius: BorderRadius.circular(5)),
   title: Text('Coffee, Tea & Cocoa',style:
   Theme.of(context).textTheme.headline1,),
   onTap: () =>const CoffeeTeaAndCocoa(),),);}), SizedBox(height: 4,),
ListTile(shape: RoundedRectangleBorder(side: BorderSide(width: 2,
   color:Color(0xffFDB73E)), borderRadius: BorderRadius.circular(5)),
   title: Text('Dairy & Eggs',style: Theme.of(context).textTheme.headline1,),
   onTap: () =>const DairyAndEggs(),),);}), SizedBox(height: 4,),
ListTile(shape: RoundedRectangleBorder(side: BorderSide(width: 2,
   color:Color(0xffFDB73E)), borderRadius: BorderRadius.circular(5)),
   title: Text('Household',style: Theme.of(context).textTheme.headline1,),
   onTap: () =>const Household(),),);}), SizedBox(height: 4,),
ListTile(shape: RoundedRectangleBorder(side: BorderSide(width: 2,
   color:Color(0xffFDB73E)), borderRadius: BorderRadius.circular(5)),
   title: Text('Instant & Ready-to-eat',style:
   Theme.of(context).textTheme.headline1,),
   onTap: () =>const InstantProducts(),),);}), SizedBox(height: 4,),
ListTile(shape: RoundedRectangleBorder(side: BorderSide(width: 2,
   color:Color(0xffFDB73E)), borderRadius: BorderRadius.circular(5)),
   title: Text('Personal Care & Beauty',style:
   Theme.of(context).textTheme.headline1,),
   onTap: () =>const PersonalCare(),),);}), SizedBox(height: 4,),
ListTile(shape: RoundedRectangleBorder(side: BorderSide(width: 2,
   color:Color(0xffFDB73E)), borderRadius: BorderRadius.circular(5)),
   title: Text('Rice, Pasta & Grains',style:
   Theme.of(context).textTheme.headline1,),
   onTap: () =>const RicePastaGrains(),),);}),SizedBox(height: 4,),
ListTile(shape: RoundedRectangleBorder(side: BorderSide(width: 2,
   color:Color(0xffFDB73E)), borderRadius: BorderRadius.circular(5)),
   title: Text('Snacks & Sweets',style:
   Theme.of(context).textTheme.headline1,),
   onTap: () =>const SnacksAndSweets(),),);}), SizedBox(height: 4,),
ListTile(shape: RoundedRectangleBorder(side: BorderSide(width: 2,
   color:Color(0xffFDB73E)), borderRadius: BorderRadius.circular(5)),
   title: Text('Spreads, Jams & Honey',style:
  Theme.of(context).textTheme.headline1,),
   onTap: () =>const SpreadJamsAndHoney(),),);}), SizedBox(height: 4,),
ListTile(shape: RoundedRectangleBorder(side: BorderSide(width: 2,
   color:Color(0xffFDB73E)), borderRadius: BorderRadius.circular(5)),
   title: Text('Others',style: Theme.of(context).textTheme.headline1,),
   onTap: () =>const Others(),),);}), SizedBox(height: 10,),],),),],),),);}}
```

products_all.dart

```
class AllProducts extends StatefulWidget{
  const AllProducts({super.key});

  @override
  State<AllProducts> createState() => _AllProductsState();}
class _AllProductsState extends State<AllProducts> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(elevation: 1, backgroundColor: const Color(0xffffffff),
        title: Text('All Products', style: Theme.of(context).textTheme.headline1,),
        leading: Builder(builder: (BuildContext) {
          return IconButton(color: Theme.of(context).primaryColor,
            icon: const Icon(Icons.arrow_back),
            onPressed: () =>Products()));},);},),
        actions: [Builder(builder: (BuildContext) {
            return IconButton(color: Theme.of(context).primaryColor,
              icon: const Icon(Icons.add), onPressed: () => const ADDProduct()));},);},),],),
      body: SingleChildScrollView(
        child: StreamBuilder<QuerySnapshot>(stream: Database.productsRef
            .orderBy('productName', descending: false).snapshots(),
          builder: (BuildContext context,
            AsyncSnapshot<QuerySnapshot> snapshot) {
            if (snapshot.hasError) {
              return Center(child: Text('Some error occurred ${snapshot.error}'));}
            if (snapshot.connectionState == ConnectionState.waiting) {
            return const Center(heightFactor: 50, child: CircularProgressIndicator(
            valueColor: AlwaysStoppedAnimation<Color>(Colors.amberAccent,)),);} else {
              return ListView.builder(shrinkWrap: true,
                  physics: NeverScrollableScrollPhysics(), scrollDirection: Axis.vertical,
                  itemCount: snapshot.data == null ? 0 : snapshot.data!.docs.length,
                  itemBuilder: (_, index) {DocumentSnapshot documentSnapshot =
                    snapshot.data!.docs[index];
```

```
                String id = snapshot.data!.docs[index].id;
                  return Card(child: ListTile( title: Text(documentSnapshot['productName'],
                      style: const TextStyle(color: Color(0xff7b3f00),fontSize: 16,),),
                      leading: SizedBox(width: 50, height: 50,
                        child: documentSnapshot['productImage'].isNotEmpty
                          ? Image.network(documentSnapshot['productImage'])
                          : Image(image: AssetImage("asset/images/default_image.png")),),
                      subtitle: Text('₱${documentSnapshot['productPrice']}',
                        style: const TextStyle(color: Color(0xff7b3f00),fontSize: 14,),),
                      onTap: () => ProductDetail(documentSnapshot: documentSnapshot,
                                productId: id,)));},));});}},)));}
```

products_baking.dart

```
class BakingAndCooking extends StatefulWidget{
  const BakingAndCooking({super.key});

  @override
  State<BakingAndCooking> createState() => _BakingAndCookingState();}
class _BakingAndCookingState extends State<BakingAndCooking> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(elevation: 1, backgroundColor: const Color(0xffffffff),
          title: Text('Baking & Cooking', style: Theme.of(context).textTheme.headline1,),
          leading: Builder(builder: (BuildContext) {
            return IconButton(color: Theme.of(context).primaryColor,
              icon: const Icon(Icons.arrow_back), onPressed: ()  =>Products()));},);},),
          actions: [Builder(builder: (BuildContext) {
              return IconButton(color: Theme.of(context).primaryColor,
                icon: const Icon(Icons.add), onPressed: => const AddProduct()));},);},),],),
        body: SingleChildScrollView(
            child: StreamBuilder<QuerySnapshot>(stream: Database.productsRef
                  .where('productCategory', isEqualTo: 'Baking & Cooking')
                  .orderBy('productName', descending: false).snapshots(),
              builder: (BuildContext context,
                  AsyncSnapshot<QuerySnapshot> snapshot) {
                if (snapshot.hasError) {
                  return Center(child: Text('Some error occurred ${snapshot.error}'));}
                if (snapshot.connectionState == ConnectionState.waiting) {
                  return const Center(heightFactor: 50,
                    child: CircularProgressIndicator( valueColor: AlwaysStoppedAnimation
                        <Color>(Colors.amberAccent,)),);} else {
                  return ListView.builder(shrinkWrap: true,
                      physics: NeverScrollableScrollPhysics(),scrollDirection: Axis.vertical,
                      itemCount: snapshot.data == null ? 0 : snapshot.data!.docs.length,
                    itemBuilder: (_, index) {DocumentSnapshot documentSnapshot =
                      snapshot.data!.docs[index]; String id = snapshot.data!.docs[index].id;
                      return Card(child: ListTile(title: Text(documentSnapshot['productName'],
                              style: const TextStyle(color: Color(0xff7b3f00),fontSize: 16,),),
                            leading: SizedBox(width: 50,height: 50,
                              child: documentSnapshot['productImage'].isNotEmpty
                                  ? Image.network(documentSnapshot['productImage'])
                                  : Image(image: AssetImage("asset/images/default_image.png")),),
                            subtitle: Text('₱${documentSnapshot['productPrice']}',
                              style: const TextStyle(color: Color(0xff7b3f00),fontSize: 14,),),
                            onTap: () {Navigator.of(context).push =>ProductDetail(
                         documentSnapshot: documentSnapshot,productId: id,)));},)); });}},)));}}
```

product_details.dart

```
class ProductDetail extends StatefulWidget{
  final DocumentSnapshot documentSnapshot;
  final String productId;

  const ProductDetail({required this.documentSnapshot, required this.productId, Key?
key}) :super(key:key);

  @override
  State<ProductDetail> createState() => _ProductDetailState();}

class _ProductDetailState extends State<ProductDetail> {
  User? user = FirebaseAuth.instance.currentUser;

  @override
  Widget build(BuildContext context){final size = MediaQuery.of(context).size;
    return Scaffold(
        appBar: AppBar(elevation: 1, backgroundColor: Color(0xfff9f2e4),
          title: Text(widget.documentSnapshot['productName'],
          style: Theme.of(context).textTheme.headline1,),
          leading: Builder(builder: (BuildContext){
            return IconButton(color: Theme.of(context).primaryColor,
              icon: const Icon(Icons.arrow_back), onPressed: (){Get.back();},);},),),
        body: Stack(children: [
            Container(height: size.height *0.40, width: MediaQuery.of(context).size.width,
                padding: const EdgeInsets.all(20), color:Color(0xfff9f2e4),
                child: widget.documentSnapshot['productImage'].isNotEmpty
                    ? Image.network(widget.documentSnapshot['productImage'])
                    : Image(image: AssetImage("asset/images/default_image.png")),),
            SingleChildScrollView(child: Column(children: [
                  Container(margin: EdgeInsets.only(top: MediaQuery.of(context).
                    size.height*0.35), width: double.infinity, height: 600,
decoration: const BoxDecoration(color: Colors.white,
  borderRadius: BorderRadius.only(topRight: Radius.circular(50),
  topLeft: Radius.circular(50)),),
child: Column(crossAxisAlignment: CrossAxisAlignment.center,
    children:[ SizedBox(height:35,),
      Padding(padding: EdgeInsets.only(left:30, right:30),
        child: RichText(text: TextSpan(text: widget.
          documentSnapshot['productName'],
          style: TextStyle(color: Color(0xff7b3f00),
            fontWeight: FontWeight.bold, fontSize: 30,),),),),
      SizedBox(height: 25,),
      Row(children: [Container(padding:EdgeInsets.only(left: 15 ),
            child: RichText(text: TextSpan(text: 'Category:' , style: TextStyle(
              color: Color(0xff7b3f00), fontWeight: FontWeight.normal,
              fontSize: 20,),),),), alignment: Alignment.centerLeft,),
          Container(padding:EdgeInsets.only(left: 15 ),
            child: RichText(text:
            TextSpan(text: widget.documentSnapshot['productCategory'],
              style: TextStyle(color: Color(0xff7b3f00),
              fontWeight: FontWeight.normal, fontSize: 20,),),),
            alignment: Alignment.centerLeft,),],), SizedBox(height: 20,),
      Row(children: [
          Container(padding:EdgeInsets.only(left: 15 ),
            child: RichText(text: TextSpan(text: 'Price: ' , style: TextStyle(
              color: Color(0xff7b3f00), fontWeight: FontWeight.normal,
              fontSize: 20,),),),), alignment: Alignment.centerLeft,),
          Container(padding:EdgeInsets.only(left: 15 ),
            child: RichText(text: TextSpan(text: '₱
              ${widget.documentSnapshot['productPrice']}', style: TextStyle(
              color: Color(0xff7b3f00), fontWeight: FontWeight.normal,
              fontSize: 20,),),),), alignment: Alignment.centerLeft,),],),
      SizedBox(height: 20,),
      Row(children: [
          Container(padding:EdgeInsets.only(left: 15 ),
            child: RichText(text: TextSpan(text: 'Cost: ' , style: TextStyle(
              color: Color(0xff7b3f00), fontWeight: FontWeight.normal,
              fontSize: 20,),),),), alignment: Alignment.centerLeft,),
          Container(padding:EdgeInsets.only(left: 15 ),
            child: RichText(text: TextSpan(text: '₱
                ${widget.documentSnapshot['productCost']}',
                style: TextStyle(color: Color(0xff7b3f00),
                  fontWeight: FontWeight.normal, fontSize: 20,),),),
            alignment: Alignment.centerLeft,),],), SizedBox(height: 20,),
      Row(children: [
          Container(padding:EdgeInsets.only(left: 15 ),
            child: RichText(text: TextSpan(text: 'Stocks: ' , style: TextStyle(
                                      color: Color(0xff7b3f00), fontWeight: FontWeight.normal,
                                      fontSize: 20,),),), alignment: Alignment.centerLeft,),
                                    Container(padding:EdgeInsets.only(left: 15 ),
                                     child: RichText(text: TextSpan(text:
                                     widget.documentSnapshot['productStocks'].toString(),
                                       style: TextStyle(color: Color(0xff7b3f00),
                                         fontWeight: FontWeight.normal, fontSize: 20,),),),
                                     alignment: Alignment.centerLeft,),],), SizedBox(height: 50,),
                                  Row(mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                                   children: [
                                    Container(child: IconButton(icon: Icon(Icons.edit), iconSize: 40,
                                      color: Color(0xffb7410e),
                                      onPressed: () =>EditPage(
                                          documentSnapshot: widget.documentSnapshot,
                                          docId: widget.productId),); },),),SizedBox(width: 10,),
                                    Container(child: IconButton(icon: Icon(Icons.delete_forever,),
                                        iconSize:40, color: Color(0xffb7410e),
                                        onPressed: () => AlertDialog(
                                             title: Text("Are you sure to delete this product?"),
                                             content: Row(
                                              mainAxisAlignment: MainAxisAlignment.spaceBetween,
                                              children: [TextButton(
                                                onPressed: () {deleteProduct(widget.productId);
                                                 Navigator.pop(context); },child: Text('YES'),),
                                               TextButton(onPressed: () async {
                                                 Navigator.pop(context); },
                                                child: const Text('CANCEL'),),],),)); })),],),]),),],),),],));
```

edit_product.dart

```
class EditPage extends StatefulWidget {
  final DocumentSnapshot documentSnapshot;
  final String docId;

  const EditPage({required this.documentSnapshot, required this.docId, Key? key})
      : super(key:key);

  @override
  State<EditPage> createState() => _EditPageState();}
class _EditPageState extends State<EditPage> {
  User? user = FirebaseAuth.instance.currentUser;
  String? _docId;
  final TextEditingController _productName = TextEditingController();
  final TextEditingController _productPrice = TextEditingController();
  final TextEditingController _productCost = TextEditingController();
  final TextEditingController _productStocks = TextEditingController();
  @override
  void initState() {super.initState();
    _docId = widget.docId;
    updateData(_docId!); }

  @override
  Widget build(BuildContext context) {return AlertDialog(
      title: const Text('Edit Item'),
      content: new Column(mainAxisSize: MainAxisSize.min,
        crossAxisAlignment: CrossAxisAlignment.start,
        children: <Widget>[
          TextFormField(controller: _productName,
            style: TextStyle(color:Color(0xFF7b3f00)),
            decoration: InputDecoration(icon: Icon(Icons.shopping_cart,),
              labelText: 'Item Name', hintStyle: TextStyle(color: Color(0xFF7b3f00)),),),
          TextFormField(controller: _productPrice, keyboardType: TextInputType.number,
            style:TextStyle(color:Color(0xFF7b3f00)),
            decoration: InputDecoration(icon:   Icon(Icons.currency_ruble,),
              labelText: 'Price', hintStyle: TextStyle(color: Color(0xFF7b3f00)),),),
          TextFormField(controller: _productCost, keyboardType: TextInputType.number,
            style:TextStyle(color:Color(0xFF7b3f00)),
            decoration: InputDecoration(icon:   Icon(Icons.currency_ruble,),
              labelText: 'Cost', hintStyle: TextStyle(color: Color(0xFF7b3f00)),),),
          TextFormField(controller: _productStocks, keyboardType: TextInputType.number,
            style:TextStyle(color:Color(0xFF7b3f00)),
            decoration: InputDecoration(icon:   Icon(Icons.add_box_outlined,),
              labelText: 'Stocks', hintStyle: TextStyle(color: Color(0xFF7b3f00)),),),
          SizedBox(height: 15,),
          Row(mainAxisAlignment: MainAxisAlignment.spaceBetween,
            children: [TextButton(onPressed: () {saveData(_docId!); },
                child: const Text('Save'),),
              TextButton(onPressed: () {Navigator.of(context).pop();},
                child: const Text('Close'),),],)],),);; }}
```

inventory.dart

```
class Inventory extends StatefulWidget{
  const Inventory({Key? key}): super (key:key);

  @override
  _InventoryState createState() => _InventoryState();}
class _InventoryState extends State<Inventory>
    with SingleTickerProviderStateMixin{
  late TabController tabController;
```

```
 @override
 void initState(){tabController = TabController(length: 3, vsync: this); super.initState();}
 @override
 void dispose(){tabController.dispose();super.dispose();}

 @override
 Widget build(BuildContext context){ return Scaffold(
     appBar: AppBar(elevation: 1, backgroundColor: const Color(0xffffffff),
       title: Text('Inventory',style: Theme.of(context).textTheme.headline1,),
       leading: Builder(builder: (BuildContext){
         return IconButton(color: Theme.of(context).primaryColor,
           icon: const Icon(Icons.arrow_back), onPressed: () =>const Homepage()));},); },),
       actions: [IconButton(onPressed: () => AddProduct()));},color:Color(0xffb7410e),
           icon: Icon(Icons.add),),],
       bottom: TabBar(unselectedLabelColor: Colors.black, labelColor: Colors.black,
         labelStyle: TextStyle(fontSize: 20), unselectedLabelStyle: TextStyle(fontSize: 20),
         indicatorColor: Colors.white, indicatorWeight: 4, indicator: BoxDecoration(
           color: Color(0xffffc000), borderRadius: BorderRadius.circular(5),),
         controller: tabController,
         tabs: [Tab(text: 'In Stock',), Tab(text:'Low Stock'), Tab(text:'Expiry'),],), ),
     body:TabBarView(controller: tabController,
       children: [Instock(),Lowstock(),Expiry(),],),);}}
```

instocks.dart

```
class Instock extends StatefulWidget{
 const Instock ({super.key});

 @override
 State<Instock> createState() => _InstockState();}

class _InstockState extends State<Instock>with SingleTickerProviderStateMixin  {
 late TabController tabController;
 @override
 void initState() {tabController = TabController(length: 14, vsync: this);
   super.initState();}
 @override
 void dispose() {tabController.dispose();super.dispose();}
 @override
 Widget build(BuildContext context){ return Scaffold(
       body: Stack(children: [SizedBox(height: 20,),
             Padding(padding:EdgeInsets.only(left: 5,right: 5),
               child: Container(height: MediaQuery.of(context).size.height,
                 child: Column(children: [SizedBox(height: 10,),
                       Container(width: MediaQuery.of(context).size.height,
                         decoration: BoxDecoration(color: Color(0xfff9f2e4),
                              borderRadius: BorderRadius.circular(20)),
                            child: Column(children: [Padding(padding: EdgeInsets.all(2),
                                  child: TabBar(unselectedLabelColor: Colors.black,
                                    labelColor: Colors.white, labelStyle: TextStyle(fontSize: 16),
                                    unselectedLabelStyle: TextStyle(fontSize: 16),
                                    indicatorColor: Colors.white, indicatorWeight: 2,
                                    indicator: BoxDecoration(color: Color(0xffb7410e),
                                      borderRadius: BorderRadius.circular(20),),
                                    controller: tabController, isScrollable: true,
                                    tabs: [Tab(text: 'ALL',), Tab(text: 'Baking & Cooking'),
                                      Tab(text: 'Beverages'), Tab(text: 'Bread & Breakfast'),
                                      Tab(text: 'Canned Foods',), Tab(text: 'Coffee, Tea & Cocoa'),
                                      Tab(text: 'Dairy & Eggs'), Tab(text: 'Household'),
                                      Tab(text: 'Instant & Ready-to-eat'),
                                      Tab(text: 'Personal Care & Beauty'),
                                      Tab(text: 'Rice, Pasta & Grains'), Tab(text: 'Snacks & Sweets'),
                                      Tab(text: 'Spread, Jams & Honey',), Tab(text: 'Others',),],),),],),),
                          Expanded(child: TabBarView(controller: tabController,
                              children: [Instocks_All(),Instocks_Baking(),Instocks_Beverages(),
                                Instocks_Bread(),Instocks_Canned(),Instocks_Coffee(),
                                Instocks_Dairy(),Instocks_Household(),Instocks_Instant(),
                                Instocks_Personal(),Instocks_Rice(),Instocks_Snacks(),
                                Instocks_Spread(),Instocks_Others(),],),),]),),),],),);}}
```

instocks_all.dart

```
class Instocks_All extends StatefulWidget {
  const Instocks_All({Key? key}) : super(key: key);

  @override
  State<Instocks_All> createState() => _Instocks_AllState();}

class _Instocks_AllState extends State<Instocks_All> {
  User? user = FirebaseAuth.instance.currentUser;

  @override
  Widget build(BuildContext context) {return Scaffold(
      body: SingleChildScrollView(child: Column(children: [
            ListTile(title: RichText(text: TextSpan(text: 'Item', style: TextStyle(
                  color: Color(0xff7b3f00), fontWeight: FontWeight.bold, fontSize: 16,),),),
              trailing: RichText(text: TextSpan(text: 'Quantity', style: TextStyle(
                  color: Color(0xff7b3f00), fontWeight: FontWeight.bold, fontSize: 16,),),),),
            StreamBuilder<QuerySnapshot>(stream: Database.productsRef
                  .orderBy('productStocks', descending: true).snapshots(),
              builder: (BuildContext context, AsyncSnapshot<QuerySnapshot> snapshot) {
                if (snapshot.hasError) {return Center(
    child: Text('Some error occurred ${snapshot.error}'),); }
  if (snapshot.connectionState == ConnectionState.waiting) {return const Center(
    heightFactor: 50, child: CircularProgressIndicator(
      valueColor: AlwaysStoppedAnimation<Color>(Colors.amberAccent,),),),); }
  final products = snapshot.data?.docs ?? [];

  return FutureBuilder<QuerySnapshot>(future: Database.productsRef.get(),
   builder: (BuildContext context, AsyncSnapshot<QuerySnapshot> snapshot) {
    if (snapshot.hasError) {return Center(
      child: Text('Some error occurred ${snapshot.error}'),); }

    final productsDocs = snapshot.data?.docs ?? [];
    final productLowQuantities = productsDocs
      .map((doc) => doc['productLowQuantity'] as int).toList();

    if (products.isEmpty) {
     return Container(padding: EdgeInsets.only(top: 70, right: 20, left: 20),
      child: Text("No products found!", style: TextStyle(
        color: Color(0xff7b3f00), fontWeight: FontWeight.bold, fontSize: 20,),
      ), alignment: Alignment.center,); } else {
     return ListView.builder(shrinkWrap: true,
       physics: NeverScrollableScrollPhysics(),scrollDirection: Axis.vertical,
       itemCount: products.length,
       itemBuilder: (_, index){ final documentSnapshot = products[index];
        final id = documentSnapshot.id;
        final productStocks =
        documentSnapshot['productStocks'] as int;
        if (productLowQuantities.any(
        (lowQuantity) => productStocks > lowQuantity)) {
         return Card(elevation: 1, child: ListTile(dense: true,
           title: Text("${documentSnapshot['productName']}", maxLines: 1,
            overflow: TextOverflow.clip,
            style: TextStyle(fontSize: 15, color: Color(0xff7b3f00),),),
           trailing: Text("$productStocks",maxLines: 1,
            overflow: TextOverflow.clip,
            style: TextStyle(fontSize: 15,color: Color(0xff7b3f00), ), ),),);
        } else {return Container();}},);}},);},),],),),);}}
```

instocks_categ.dart

```
class Instocks_All extends StatefulWidget {
 const Instocks_All({Key? key}) : super(key: key);

 @override
 State<Instocks_All> createState() => _Instocks_AllState();}
class _Instocks_AllState extends State<Instocks_All> {
```

```
User? user = FirebaseAuth.instance.currentUser;

@override
Widget build(BuildContext context) {return Scaffold(
   body: SingleChildScrollView(child: Column(children: [
        ListTile(title: RichText(text: TextSpan(text: 'Item', style: TextStyle(
             color: Color(0xff7b3f00), fontWeight: FontWeight.bold, fontSize: 16,),),),
          trailing: RichText(text: TextSpan(text: 'Quantity', style: TextStyle(
             color: Color(0xff7b3f00), fontWeight: FontWeight.bold, fontSize: 16,),),),),
        StreamBuilder<QuerySnapshot>(stream: Database.productsRef
             .orderBy('productStocks', descending: true).snapshots(),
          builder: (BuildContext context, AsyncSnapshot<QuerySnapshot> snapshot) {
            if (snapshot.hasError) {return Center(
               child: Text('Some error occurred ${snapshot.error}'),); }
            if (snapshot.connectionState == ConnectionState.waiting) {
             return const Center(heightFactor: 50, child: CircularProgressIndicator(
                valueColor: AlwaysStoppedAnimation<Color>(Colors.amberAccent,),),); }
            final products = snapshot.data?.docs ?? [];
            return FutureBuilder<QuerySnapshot>(
             future: Database.productsRef.get(),
             builder: (BuildContext context,
                AsyncSnapshot<QuerySnapshot> snapshot) {
               if (snapshot.hasError) {return Center(
                  child: Text('Some error occurred ${snapshot.error}'),); }

               final productsDocs = snapshot.data?.docs ?? [];
               final productLowQuantities = productsDocs
                  .map((doc) => doc['productLowQuantity'] as int).toList();

               if (products.isEmpty) {
                return Container(padding: EdgeInsets.only(top: 70, right: 20, left: 20),
                  child: Text("No products found!", style: TextStyle(
                     color: Color(0xff7b3f00), fontWeight: FontWeight.bold, fontSize: 20,),
                  ), alignment: Alignment.center,); } else {
                return ListView.builder(shrinkWrap: true,
                   physics: NeverScrollableScrollPhysics(),scrollDirection: Axis.vertical,
                   itemCount: products.length,
                   itemBuilder: (_, index){ final documentSnapshot = products[index];
                    final id = documentSnapshot.id;
                    final productStocks =documentSnapshot['productStocks'] as int;
                    if (productLowQuantities.any(
                    (lowQuantity) => productStocks > lowQuantity)) {
                     return Card(elevation: 1, child: ListTile(dense: true,
                        title: Text("${documentSnapshot['productName']}", maxLines: 1,
                          overflow: TextOverflow.clip,
                          style: TextStyle(fontSize: 15, color: Color(0xff7b3f00),),),
                        trailing: Text("$productStocks",maxLines: 1,
                          overflow: TextOverflow.clip,
                          style: TextStyle(fontSize: 15,color: Color(0xff7b3f00), ),),),);
                    } else {return Container();}},);}},);},),],),),);}}
```

lowstocks.dart

```
class Lowstock extends StatefulWidget{
  const Lowstock ({super.key});

  @override
  State<Lowstock> createState() => _LowstockState();}
class _LowstockState extends State<Lowstock>with SingleTickerProviderStateMixin  {
  late TabController tabController;

  @override
  void initState() {tabController = TabController(length: 14, vsync: this);
    super.initState();}

  @override
  void dispose() {tabController.dispose(); super.dispose();}

  @override
  Widget build(BuildContext context){ return Scaffold(
        body: Stack(children: [SizedBox(height: 10,),
              Padding(padding:EdgeInsets.only(left: 5,right: 5),
                child: Container(height: MediaQuery.of(context).size.height,
                  child: Column(children: [SizedBox(height: 10,),
                        Container(width: MediaQuery.of(context).size.height,
                          decoration: BoxDecoration(color: Color(0xfff9f2e4),
                              borderRadius: BorderRadius.circular(20)),
                          child: Column(children: [Padding(padding: EdgeInsets.all(2),
                                child: TabBar(unselectedLabelColor: Colors.black,
                                  labelColor: Colors.white, labelStyle: TextStyle(fontSize: 16),
                                  unselectedLabelStyle: TextStyle(fontSize: 16),
                                  indicatorColor: Colors.white, indicatorWeight: 2,
                                  indicator: BoxDecoration(color: Color(0xffb7410e),
                                    borderRadius: BorderRadius.circular(20),),
                                  controller: tabController, isScrollable: true,
                                  tabs: [Tab(text: 'ALL',), Tab(text: 'Baking & Cooking'),
                                    Tab(text: 'Beverages'), Tab(text: 'Bread & Breakfast'),
                                    Tab(text: 'Canned Foods',),Tab(text: 'Coffee, Tea & Cocoa'),
                                    Tab(text: 'Dairy & Eggs'), Tab(text: 'Household'),
                                    Tab(text: 'Instant & Ready-to-eat'),
                                    Tab(text: 'Personal Care & Beauty'),
                                    Tab(text: 'Rice, Pasta & Grains'), Tab(text: 'Snacks & Sweets'),
                Tab(text: 'Spread, Jams & Honey',), Tab(text: 'Others',),],),),],),),
          Expanded(child: TabBarView(controller: tabController,
            children: [LowStocks_All(),LowStocks_Baking(),
              LowStocks_Beverages(),LowStocks_Bread(),LowStocks_Canned(),
              LowStocks_Coffee(),LowStocks_Dairy(),LowStocks_Household(),
              LowStocks_Instant(),LowStocks_Personal(),LowStocks_Rice(),
              LowStocks_Snacks(),LowStocks_Spread(),LowStocks_Others(),],
           ),),]),),),],),); }}
```

lowstocks_all.dart

```
class LowStocks_All extends StatefulWidget {
  const LowStocks_All({Key? key}) : super(key: key);

  @override
  State<LowStocks_All> createState() => _LowStocks_AllState();}
class _LowStocks_AllState extends State<LowStocks_All> {

  @override
  Widget build(BuildContext context) {return Scaffold(
      body: SingleChildScrollView(child: Column(children: [
          ListTile(title: RichText(text: TextSpan(text: 'Item', style: TextStyle(
               color: Color(0xff7b3f00), fontWeight: FontWeight.bold, fontSize: 16,),),),
            trailing: RichText(text: TextSpan(text: 'Quantity', style: TextStyle(
               color: Color(0xff7b3f00), fontWeight: FontWeight.bold, fontSize: 16,),),),),
          StreamBuilder<QuerySnapshot>(
            stream: Database.productsRef
               .orderBy('productStocks', descending: false).snapshots(),
            builder: (BuildContext context,
               AsyncSnapshot<QuerySnapshot> snapshot) {
              if (snapshot.hasError) {
                return Center(child: Text('Some error occurred ${snapshot.error}'),); }
              if (snapshot.connectionState == ConnectionState.waiting) {
                return const Center(heightFactor: 50, child: CircularProgressIndicator(
                    valueColor: AlwaysStoppedAnimation<Color>(Colors.amberAccent,),),); }

              final products = snapshot.data?.docs ?? [];

              return FutureBuilder<QuerySnapshot>(
                future: Database.productsRef.get(),
                builder: (BuildContext context, AsyncSnapshot<QuerySnapshot> snapshot) {
                  if (snapshot.hasError) {return Center(
                      child: Text('Some error occurred ${snapshot.error}'),); }
                  final productsDocs = snapshot.data?.docs ?? [];
                  final productLowQuantities = productsDocs
                      .map((doc) => doc['productLowQuantity'] as int).toList();
                  if (products.isEmpty) {
                    return Container(padding: EdgeInsets.only(top: 70, right: 20, left: 20),
                      child: Text("No products found!", style: TextStyle(
                          color: Color(0xff7b3f00), fontWeight: FontWeight.bold,
                          fontSize: 20,),), alignment: Alignment.center,); } else {
                    return ListView.builder(shrinkWrap: true,
                      physics: NeverScrollableScrollPhysics(),scrollDirection: Axis.vertical,
                      itemCount: products.length,
                      itemBuilder: (_, index){ final documentSnapshot = products[index];
                        final id = documentSnapshot.id;
                        final productStocks =
                        documentSnapshot['productStocks'] as int;
                        if (productLowQuantities.any(
                                (lowQuantity) => productStocks <= lowQuantity)) {
                          return Card(elevation: 1,
                            child: ListTile(dense: true,
                              title: Text("${documentSnapshot['productName']}", maxLines: 1,
                                overflow: TextOverflow.clip,
                                style: TextStyle(fontSize: 15, color: Color(0xff7b3f00),),),
                              trailing: Text("$productStocks",maxLines: 1,
                                overflow: TextOverflow.clip,
                                style: TextStyle(fontSize: 15,color: Color(0xff7b3f00), ),),),);
                        } else {return Container();}},);}},);},),],),),);}}
```

lowstocks_categ.dart

```
class LowStocks_Baking extends StatefulWidget {
  const LowStocks_Baking({Key? key}) : super(key: key);

  @override
  State<LowStocks_Baking> createState() => _LowStocks_BakingState();}
class _LowStocks_BakingState extends State<LowStocks_Baking> {

  @override
  Widget build(BuildContext context) {return Scaffold(
      body: SingleChildScrollView(child: Column(children: [
          ListTile(title: RichText(text: TextSpan(text: 'Item', style: TextStyle(
                color: Color(0xff7b3f00), fontWeight: FontWeight.bold, fontSize: 16,),),),
            trailing: RichText(text: TextSpan(text: 'Quantity', style: TextStyle(
                color: Color(0xff7b3f00), fontWeight: FontWeight.bold, fontSize: 16,),),),),
          StreamBuilder<QuerySnapshot>(stream: Database.productsRef
                .where('productCategory', isEqualTo: 'Baking & Cooking')
                .orderBy('productStocks', descending: false).snapshots(),
            builder: (BuildContext context, AsyncSnapshot<QuerySnapshot> snapshot) {
              if (snapshot.hasError) {
```

```
          return Center(child: Text('Some error occurred ${snapshot.error}'),); }
        if (snapshot.connectionState == ConnectionState.waiting) {
          return const Center(heightFactor: 50, child: CircularProgressIndicator(
              valueColor: AlwaysStoppedAnimation<Color>(Colors.amberAccent,),),),); }
        final products = snapshot.data?.docs ?? [];
        return FutureBuilder<QuerySnapshot>(future: Database.productsRef.get(),
          builder: (BuildContext context, AsyncSnapshot<QuerySnapshot> snapshot) {
            if (snapshot.hasError) {return Center(
                child: Text('Some error occurred ${snapshot.error}'),); }
            final productsDocs = snapshot.data?.docs ?? [];
            final productLowQuantities = productsDocs
                .map((doc) => doc['productLowQuantity'] as int).toList();
            if (products.isEmpty) {
              return Container(padding: EdgeInsets.only(top: 70, right: 20, left: 20),
                child: Text("No products found!", style: TextStyle(
                    color: Color(0xff7b3f00), fontWeight: FontWeight.bold,
                    fontSize: 20,),), alignment: Alignment.center,); } else {
              return ListView.builder(shrinkWrap: true,
                physics: NeverScrollableScrollPhysics(),scrollDirection: Axis.vertical,
                itemCount: products.length,
                itemBuilder: (_, index){ final documentSnapshot = products[index];
                  final id = documentSnapshot.id;
                  final productStocks =
                  documentSnapshot['productStocks'] as int;
                  if (productLowQuantities.any(
                          (lowQuantity) => productStocks <= lowQuantity)) {
                    return Card(elevation: 1,
                      child: ListTile(dense: true,
                        title: Text("${documentSnapshot['productName']}", maxLines: 1,
                          overflow: TextOverflow.clip,
                          style: TextStyle(fontSize: 15, color: Color(0xff7b3f00),),),
                        trailing: Text("$productStocks",maxLines: 1,
                          overflow: TextOverflow.clip,
                          style: TextStyle(fontSize: 15,color: Color(0xff7b3f00), ),),),);
                  } else {return Container();}},);}},);},),],),),);}}
```

expiry.dart

```
class Expiry extends StatefulWidget{
  const Expiry ({super.key});

  @override
  State<Expiry> createState() => _ExpiryState();}
class _ExpiryState extends State<Expiry>with SingleTickerProviderStateMixin  {
  late TabController tabController;
```

```
 @override
 void initState() {tabController = TabController(length: 4,vsync: this); super.initState();}

 @override
 void dispose() {tabController.dispose();super.dispose();}

 @override
 Widget build(BuildContext context){ return Scaffold(body:Stack(
        children: [Padding(padding:EdgeInsets.only(left: 5,right: 5),
            child: Container(height: MediaQuery.of(context).size.height,
              child: Column(children: [SizedBox(height: 10,),
                    Container(width: MediaQuery.of(context).size.height,
                      decoration: BoxDecoration(color: Color(0xfff9f2e4),
                          borderRadius: BorderRadius.circular(20)),
                      child: Column(children: [Padding(padding: EdgeInsets.all(2),
                            child: TabBar(unselectedLabelColor: Colors.black,
                              labelColor: Colors.white, labelStyle: TextStyle(fontSize: 16),
                              unselectedLabelStyle: TextStyle(fontSize: 16),
                              indicatorColor: Colors.white, indicatorWeight: 2,
                              indicator: BoxDecoration(color: Color(0xffb7410e),
                                borderRadius: BorderRadius.circular(20),),
                              controller: tabController, isScrollable: true,
                              tabs: [Tab(text: '7 days',),Tab(text: '15 days'),
                                Tab(text: '30 days'), Tab(text: 'All'),],),),],),),
                    Expanded(child: TabBarView(controller: tabController,
                        children: [Expiry_Seven(), Expiry_Fifteen(),
                          Expiry_Thirty(), Expiry_All(),],),),]),),),],));}}
```

expiry_fifteen.dart

```
class Expiry_Fifteen extends StatefulWidget {
 const Expiry_Fifteen({Key? key}) : super(key: key);

 @override
 State<Expiry_Fifteen> createState() => _Expiry_FifteenState();}
class _Expiry_FifteenState extends State<Expiry_Fifteen> {
 User? user = FirebaseAuth.instance.currentUser;

 @override
 Widget build(BuildContext context) {return Scaffold(
     body: SingleChildScrollView(child: Column(children: [
           ListTile(title: RichText(text: TextSpan(text: 'Item', style: TextStyle(
                 color: Color(0xff7b3f00), fontWeight: FontWeight.bold, fontSize: 16,),),),
             trailing: RichText(text: TextSpan(text: 'Date', style: TextStyle(
                 color: Color(0xff7b3f00), fontWeight: FontWeight.bold, fontSize: 16,),),),),
           StreamBuilder<QuerySnapshot>(stream: Database.productsRef
                 .orderBy('expDate', descending: false).snapshots(),
             builder: (BuildContext context, AsyncSnapshot<QuerySnapshot> snapshot) {
              if (snapshot.hasError) {return Center(
                 child: Text('Some error occurred ${snapshot.error}'),); }
              if (snapshot.connectionState == ConnectionState.waiting) {
                 return const Center(heightFactor: 50, child: CircularProgressIndicator(
                    valueColor: AlwaysStoppedAnimation<Color>(Colors.amberAccent,),),); }
              final products = snapshot.data?.docs ?? [];
              DateTime today = DateTime.now();
              DateTime expiryThreshold = today.add(Duration(days: 15));
              List<QueryDocumentSnapshot> expiringProducts = products.where((product){
                 Timestamp timestamp = product.get('expDate');
                 DateTime expiryDate = timestamp.toDate();
                 return expiryDate.isBefore(expiryThreshold); }).toList();
              if (expiringProducts.isEmpty) {
                 return Container(padding: EdgeInsets.only(top: 70, right: 20, left: 20),
                 child: Text("No products found!", style: TextStyle(color: Color(0xff7b3f00),
                   fontWeight: FontWeight.bold, fontSize: 20,),),
                 alignment: Alignment.center,); } else {
                 return ListView.builder(shrinkWrap: true,
                 physics: NeverScrollableScrollPhysics(),scrollDirection: Axis.vertical,
                 itemCount: expiringProducts.length,
                 itemBuilder: (_, index){ final documentSnapshot = expiringProducts[index];
                    final id = documentSnapshot.id;
                    List<QueryDocumentSnapshot> documents = expiringProducts;
                    List<DateTime> dates = [];
                    List<String> formattedDates = [];
                    for (var document in documents) {
                   Timestamp timestamp = document.get('expDate');
                   DateTime date = timestamp.toDate();
                   dates.add(date);
                   String formattedDate = DateFormat('MMM dd, yyyy').format(date);
                   formattedDates.add(formattedDate); }
                    String formattedDate = formattedDates[index];
                    return Card(elevation: 1,
                   child: ListTile(dense: true,
                    title: Text("${documentSnapshot['productName']}", maxLines: 1,
                     overflow: TextOverflow.clip,
                     style: TextStyle(fontSize: 15, color: Color(0xff7b3f00),),),
                    trailing: Text("$formattedDate",maxLines: 1,
                     overflow: TextOverflow.clip,
                     style: TextStyle(fontSize: 15,color: Color(0xff7b3f00), ),),),); },);}},),],
```

hot_items.dart

```
class HotItems extends StatefulWidget{
 const HotItems({super.key});

 @override
 State<HotItems> createState() => _HotItemsState();}
class _HotItemsState extends State<HotItems> {
 User? user = FirebaseAuth.instance.currentUser;

 @override
 Widget build(BuildContext context){ return Scaffold(
    appBar: AppBar(elevation: 1, backgroundColor: const Color(0xffffffff),
      title: Text('Hot Items',style: Theme.of(context).textTheme.headline1,),
      leading: Builder(builder: (BuildContext){
        return IconButton(color: Theme.of(context).primaryColor,
          icon: const Icon(Icons.arrow_back), onPressed: () => const Homepage()));},);},),
    body:SingleChildScrollView(child: Column(children: [SizedBox(height: 20,),
          Text("TOP 3 FAST SELLING ITEMS",style: TextStyle(fontSize: 25.0,
              fontWeight: FontWeight.bold, color:Color(0xff7b3f00)),
            textAlign: TextAlign.center,), SizedBox(height: 20,),
        StreamBuilder<QuerySnapshot>(stream: FirebaseFirestore.instance
            .collection('users').doc(user!.uid).collection('products')
            .orderBy('soldCount', descending: true).limit(3).snapshots(),
          builder: (context, snapshot) {
            if (snapshot.hasError) {return Text('Error: ${snapshot.error}'); }
            if (snapshot.connectionState == ConnectionState.waiting) {
              return CircularProgressIndicator();}
            final topItems = snapshot.data!.docs;
            return GridView.builder(physics: const NeverScrollableScrollPhysics(),
              shrinkWrap: true, gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
                  crossAxisCount: 1, crossAxisSpacing: 1, mainAxisExtent: 170,
                  mainAxisSpacing: 12), itemCount: topItems.length,
              itemBuilder: (_, index){ final documentSnapshot = topItems[index];
                return Container(margin: EdgeInsets.only(left: 15, right: 15),
                  decoration: BoxDecoration(borderRadius: BorderRadius.circular(2,),
                    color: Color(0xfff9f2e4),),
                  child: Column(crossAxisAlignment: CrossAxisAlignment.start,
                    children: [Padding(padding: const EdgeInsets.all(8),
                        child: Row(crossAxisAlignment: CrossAxisAlignment.start,
                          children: [Stack(children: [
                                Container(height: 150, width: 140,
                                  decoration: BoxDecoration(color:Color(0xffFDB73E),
                                    border: Border.all(color: Color(0xffFDB73E),width: 2,),
                                    borderRadius: BorderRadius.only(topRight: Radius.circular(2),
                                    topLeft: Radius.circular(2), bottomLeft:
                                  Radius.circular(2),bottomRight: Radius.circular(2),),),
                                  child:ClipRRect(borderRadius: const BorderRadius.only(
                                      topLeft: Radius.circular(16.0), topRight: Radius.circular(16.0),),
                                    child: documentSnapshot['productImage'].isNotEmpty
                                      ? Image.network(documentSnapshot['productImage'],
                                      height: double.infinity, width: double.infinity, fit: BoxFit.cover,)
                                      : Image.asset("asset/images/default_image.png",
                                      height: double.infinity, width: double.infinity,
                                      fit: BoxFit.contain,),),),
                                Positioned(top: 0, left: 0, child: Container(height: 40, width: 40,
                                    decoration: BoxDecoration(shape: BoxShape.circle,
                                      border: Border.all(width:1,
                                      color: Theme.of(context).primaryColor,),
                                      color: Theme.of(context).primaryColor,),
                                    child: RichText(text:
                                      TextSpan(text:"${popularproducts.elementAt(index)['rank']}",
                                      style: TextStyle(fontSize: 35, color: Color(0xffffffff),),),
                                      textAlign: TextAlign.center,),),),),],), SizedBox(width: 20,),
                               Container(padding: EdgeInsets.all(5),
                                width: MediaQuery.of(context).size.width*0.35,
                                child:Column(children: [SizedBox(height: 25,),
                                  Text(documentSnapshot['productName'],
                                    overflow: TextOverflow.ellipsis, maxLines: 2,
                                    textAlign: TextAlign.left, style: TextStyle(fontSize: 18,
                                      color: Color(0xff7b3f00), fontWeight: FontWeight.bold,),),
                                  const SizedBox(height: 5,),
                                  Row(children: [RichText(text: TextSpan(text:"Price:",
                                       style: TextStyle(fontSize: 18, color: Color(0xff7b3f00),),),
                                       textAlign: TextAlign.left,), const SizedBox(width: 5,),
                                      RichText(text: TextSpan(text:
                                      "${documentSnapshot['productPrice']}", style: TextStyle(
                                        fontSize: 18, color: Color(0xff7b3f00),),),
                                       textAlign: TextAlign.right,),],), const SizedBox(height: 5,),
                                  Row(children: [RichText(text: TextSpan(text:"Items Sold:",
                                       style: TextStyle(fontSize: 18, color: Color(0xff7b3f00),),),
                                       textAlign: TextAlign.left,), const SizedBox(width: 5,),
                                      RichText(text: TextSpan(text:
                                      "${documentSnapshot['soldCount']}", style: TextStyle(
                                        fontSize: 18, color: Color(0xff7b3f00),),),
                                       textAlign: TextAlign.right,),],),],),
                                alignment: Alignment.center,), const SizedBox(width: 8,),],),)],
                  ),); },);},),],),),);}}
```

transactions.dart

```
class Transactions extends StatefulWidget{
  const Transactions ({Key? key}) : super (key:key);

  @override
  _TransactionsState createState() => _TransactionsState();}
class _TransactionsState extends State<Transactions>
    with SingleTickerProviderStateMixin{
  late TabController tabController;

  @override
  void initState(){tabController = TabController(length:4, vsync:this); super.initState();}
  @override
  void dispose(){tabController.dispose();super.dispose();}
  @override
  Widget build(BuildContext context){ return Scaffold(
      appBar: AppBar(elevation: 1, backgroundColor: Color(0xffffffff),
        title: Text('Transactions',style: Theme.of(context).textTheme.displayLarge,),
        leading: Builder(builder: (BuildContext){
          return IconButton(color: Theme.of(context).primaryColor,
            icon: const Icon(Icons.arrow_back), onPressed: () => Homepage()));},);},),
        bottom:TabBar(unselectedLabelColor: Colors.black, labelColor: Colors.black,
         labelStyle: TextStyle(fontSize: 16), unselectedLabelStyle: TextStyle(fontSize: 16),
         indicatorColor: Colors.white, indicatorWeight: 4,
         indicator: BoxDecoration(color: Color(0xffffc000),
            borderRadius: BorderRadius.circular(5),), controller: tabController,
         tabs: [Tab(text:'Daily',),Tab(text:'Weekly'),Tab(text:'Monthly'),Tab(text:'Yearly')],),
      body:SingleChildScrollView(child: Column(children: [
            Padding(padding:EdgeInsets.symmetric(horizontal:5),
              child: Container(height: MediaQuery.of(context).size.height,
                child: Column(children: [SizedBox(height: 10,),
                      Expanded(
                        child: TabBarView(controller: tabController,
                          children: [Daily(),\Weekly(),Monthly(),Yearly(),],),),]),),),
            SizedBox(height: 100,),],),),); }}
```

transactions_daily.dart

```
class Daily extends StatefulWidget {
  const Daily({Key? key}) : super(key: key);

  @override
  State<Daily> createState() => _DailyState();}
class _DailyState extends State<Daily> {
  late User? user; late DateTime selectedDate;
  double totalSales = 0.0; double totalCost = 0.0; double totalProfit = 0.0;

  @override
  void initState() {super.initState();
    user = FirebaseAuth.instance.currentUser; selectedDate = DateTime.now();
    updateTotalSales();}
```

```
@override
void didUpdateWidget(Daily oldWidget) {
  super.didUpdateWidget(oldWidget);
  selectedDate = DateTime.now();
  updateTotalSales();}

@override
Widget build(BuildContext context) {return Scaffold(
    body: SingleChildScrollView(child: Column(children: [
          Container(height: 120,
            decoration: BoxDecoration(color: Color(0xfff9f2e4),
              border: Border.all(width: 5, color: Color(0xffffc000),),
              borderRadius: BorderRadius.all(Radius.circular(5)),),
            alignment: Alignment.center,
            child: Column(children: [
                Row(mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                  children: [Icon(Icons.calendar_today_rounded),
                    Text("${DateFormat('MMM d, yyyy').format(selectedDate)}",
                      style: TextStyle(fontSize: 24, color: Colors.black),),
                    OutlinedButton(onPressed: () async {
                        final DateTime? dateTime = await showDatePicker(context: context,
                          initialDate: selectedDate, firstDate: DateTime(1980),
                          lastDate: DateTime(3000),
                          initialDatePickerMode: DatePickerMode.day,
                          builder: (BuildContext context, Widget? child) {
                            return Theme(data: ThemeData.light().copyWith(
                                colorScheme: ColorScheme.light(primary: Colors.amber,
                                  onPrimary: Colors.black,), dialogBackgroundColor: Colors.white,
                              ), child: child!,); },);
                        if (dateTime != null) {
                          setState(() {selectedDate = dateTime; updateTotalSales();});}},
                      child: Text("Select Date",
                        style: TextStyle(fontSize: 20, color: Colors.black),),),],),
                Divider(), SizedBox(height: 5,),
                Row(mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                  children: [RichText(text: TextSpan(text: "Total Sales:",
                        style: TextStyle(color: Color(0xff7b3f00), fontWeight: FontWeight.bold,
                          fontSize: 20,),),),
                    RichText(text: TextSpan(text: totalSales.toStringAsFixed(2),
                        style: TextStyle(color: Color(0xff7b3f00), fontWeight: FontWeight.bold,
                          fontSize: 20,),),),],),],),), SizedBox(height: 10,), Divider(),
          SizedBox(height: 10,),
          Container(padding: EdgeInsets.only(left: 15, right: 15),
            child: Row(mainAxisSize: MainAxisSize.max,
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: <Widget>[
                RichText(text: TextSpan(text: 'Time', style: TextStyle(
                      color: Color(0xff7b3f00), fontWeight: FontWeight.bold,
                      fontSize: 16,),), textAlign: TextAlign.left,),
                RichText(text: TextSpan(text: 'Order Number', style: TextStyle(
                      color: Color(0xff7b3f00), fontWeight: FontWeight.bold,
                      fontSize: 16,),),),
                RichText(text: TextSpan(text: 'Total',style: TextStyle(
                      color: Color(0xff7b3f00), fontWeight: FontWeight.bold,
                      fontSize: 16,),), textAlign: TextAlign.right,),],),),
          SizedBox(height: 10,),
          StreamBuilder<QuerySnapshot>(
            stream: _firestore.instance.collection('orders').doc(user!.uid).collection('orders')
                .where('timestamp', isGreaterThanOrEqualTo:
                      Timestamp.fromDate(selectedDate))
                .where('timestamp', isLessThan:
                      Timestamp.fromDate(selectedDate.add(Duration(days: 1))))
                .orderBy('timestamp', descending: true).snapshots(),
            builder: (BuildContext context, AsyncSnapshot<QuerySnapshot> snapshot) {
              if (snapshot.hasError) {
                return Center(child: Text('Some error occurred ${snapshot.error}')); }
              if (snapshot.connectionState == ConnectionState.waiting) {
                return const Center(heightFactor: 50,
                  child: CircularProgressIndicator(
                    valueColor: AlwaysStoppedAnimation<Color>(Colors.amberAccent),),); }
              if (snapshot.hasData && snapshot.data!.docs.isEmpty) {
                return Container(padding: EdgeInsets.only(top: 70, right: 20, left: 20),
                  child: Text("No transactions found!",style: TextStyle(
                      color: Color(0xff7b3f00), fontWeight: FontWeight.bold, fontSize: 20,),),
                  alignment: Alignment.center,); } else {
                return ListView.builder(shrinkWrap: true,
                  physics: NeverScrollableScrollPhysics(),
                  scrollDirection: Axis.vertical,
                  itemCount: snapshot.data == null ? 0 : snapshot.data!.docs.length,
                  itemBuilder: (_, index) {
                    DocumentSnapshot documentSnapshot = snapshot.data!.docs[index];
                    String id = snapshot.data!.docs[index].id;
                    List<QueryDocumentSnapshot> documents = snapshot.data!.docs;
                    List<DateTime> dates = []; List<String> formattedDates = [];

                    for (var document in documents) {
                      Timestamp timestamp = document.get('timestamp');
                      DateTime date = timestamp.toDate();dates.add(date);
                      String formattedDate = DateFormat('h:mm a').format(date);
                      formattedDates.add(formattedDate); }
                    String formattedTime = formattedDates[index];
```

```
                      return Card(elevation: 1,
                        child: ListTile(dense: true,
                          leading: RichText(text: TextSpan(text: "$formattedTime",
                            style: TextStyle(color: Color(0xff7b3f00),
                              fontWeight: FontWeight.bold, fontSize: 15,),),),
                          title: RichText(text: TextSpan(text: "${documentSnapshot['orderNo']}",
                            style: TextStyle(color: Color(0xff7b3f00),
                              fontWeight: FontWeight.bold, fontSize: 15,),),
                            textAlign: TextAlign.center,),
                          trailing: RichText(text: TextSpan(text: "${documentSnapshot['sales']}",
                            style: TextStyle(color: Color(0xff7b3f00),
                              fontWeight: FontWeight.bold, fontSize: 15,),),),),); },);}},),
          SizedBox(height: 50,),],),),),); }}
```

sales_report.dart

```
class SalesReport extends StatefulWidget{
  SalesReport({Key? key}) : super(key:key);

  @override
  State<SalesReport> createState() => _SalesReportState();}
class _SalesReportState extends State<SalesReport> {
  final SunmiReceipt sunmiReceipt = SunmiReceipt();
  User? user = FirebaseAuth.instance.currentUser;
  late GlobalKey<SfCartesianChartState> _cartesianChartKey;
  late List<SalesData> data;
  int? totalQuantity; double? totalSales; double? totalCost; double? totalProfit;
  @override
  void initState(){
        fetchTotal();
        _zoomPanBehavior = ZoomPanBehavior(
        enablePinching: true, enableDoubleTapZooming: true,
        enableSelectionZooming: true,
        selectionRectBorderColor: Colors.red, selectionRectBorderWidth: 1,
        selectionRectColor: Colors.grey);
    _cartesianChartKey = GlobalKey();super.initState();}
  late ZoomPanBehavior _zoomPanBehavior;

  @override
  Widget build(BuildContext context){ return Scaffold(
      appBar: AppBar(elevation: 1, backgroundColor: Color(0xffffffff),
        title: Text('Sales Report',style: Theme.of(context).textTheme.displayLarge,),
        leading: Builder(builder: (BuildContext){
          return IconButton(color: Theme.of(context).primaryColor,
            icon: const Icon(Icons.arrow_back), onPressed: () =>Homepage()));},);},),),
      body:SingleChildScrollView(child: Column(children: [SizedBox(height: 10,),
Container(margin: EdgeInsets.only(left: 10,right: 10),
  padding: EdgeInsets.only(left: 5,right: 5), height: 80,
  decoration: BoxDecoration(color: Color(0xffffc000),
    border: Border.all(width: 5, color: Color(0xffffc000),),
    borderRadius: BorderRadius.all(Radius.circular(5)),),
  alignment: Alignment.center,
  child: Column(children: [SizedBox(height: 10,),
      Row(mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: [Icon(Icons.shopping_bag_outlined,size: 20,color: Colors.black ,),
          RichText(text: TextSpan(text: "NO. OF ITEMS SOLD",
            style: TextStyle(color: Colors.black, fontWeight: FontWeight.bold,
              fontSize: 20,),),textAlign: TextAlign.center,),],), Divider(),
      RichText(text: TextSpan(text: "$totalQuantity",style: TextStyle(
          color: Colors.black, fontWeight: FontWeight.normal, fontSize: 18,),),),],),),
SizedBox(height: 5,),
Container(margin: EdgeInsets.only(left: 10,right: 10),
  padding: EdgeInsets.only(left: 5,right: 5), height: 80,
  decoration: BoxDecoration(color: Color(0xffb7410e),
    border: Border.all(width: 5, color: Color(0xffb7410e),),
    borderRadius: BorderRadius.all(Radius.circular(5)),),
  alignment: Alignment.center, child: Column(
    children: [SizedBox(height: 10,),
      Row(mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: [Icon(Icons.money,size: 20,color: Color(0xffffffff) ,),
          RichText(text: TextSpan(text: "TOTAL SALES",style: TextStyle(
              color: Color(0xffffffff), fontWeight: FontWeight.bold, fontSize: 20,),),
            textAlign: TextAlign.center,),],), Divider(),
      RichText(text: TextSpan(text: "$totalSales",style: TextStyle(
          color: Color(0xffffffff), fontWeight: FontWeight.normal, fontSize: 18,
            ),),),],),), SizedBox(height: 10,),Divider(),
Container(padding: EdgeInsets.all(5), child: SalesChart(),),Divider(),
SizedBox(height: 10,),
Row(mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: [Container(margin: EdgeInsets.only(left: 10,right: 10),
      padding: EdgeInsets.only(left: 5,right: 5), height: 80,
      decoration: BoxDecoration(color: Color(0xfff9f2e4), border: Border.all(
          width: 5, color: Color(0xffffc000),),
        borderRadius: BorderRadius.all(Radius.circular(5)),),
      alignment: Alignment.center,
      child: Column(children: [SizedBox(height: 10,),
          Row(mainAxisAlignment: MainAxisAlignment.spaceEvenly,
            children: [Icon(Icons.bar_chart,size: 15,color: Color(0xff7b3f00) ,),
              RichText(text: TextSpan(text: "REVENUE",style: TextStyle(
                  color: Color(0xff7b3f00), fontWeight: FontWeight.bold, fontSize: 16,
                ),), textAlign: TextAlign.center,),],), Divider(),
          RichText(text: TextSpan(text: "₱${totalSales?.toStringAsFixed(2)}",
                style: TextStyle(color: Color(0xff7b3f00),
                  fontWeight: FontWeight.normal, fontSize: 15,),),),],),),
          Container(margin: EdgeInsets.only(left: 10,right: 10),
            padding: EdgeInsets.only(left: 5,right: 5), height: 80,
            decoration: BoxDecoration(color: Color(0xfff5e5de),
              border: Border.all(width: 5, color: Color(0xffb7410e),),
              borderRadius: BorderRadius.all(Radius.circular(5)),),
            alignment: Alignment.center,
            child: Column(children: [SizedBox(height: 10,),
                Row(mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                  children: [Icon(Icons.sell_outlined,size: 15,color: Color(0xff7b3f00) ,),
                    RichText(text: TextSpan(text: "COST",style: TextStyle(
                        color: Color(0xff7b3f00), fontWeight: FontWeight.bold, fontSize: 16,
                      ),), textAlign: TextAlign.center,),],), Divider(),
                RichText(text: TextSpan(text: "₱${totalCost?.toStringAsFixed(2)}",
                  style: TextStyle(color: Color(0xff7b3f00),
                    fontWeight: FontWeight.normal, fontSize: 15,),),),],),),
          Container(margin: EdgeInsets.only(left: 10,right: 10),
            padding: EdgeInsets.only(left: 5,right: 5), height: 80,
            decoration: BoxDecoration(color: Color(0xfff9f2e4),border: Border.all(
                width: 5, color: Color(0xffffc000),),
              borderRadius: BorderRadius.all(Radius.circular(5)), ),
            alignment: Alignment.center, child: Column(
              children: [SizedBox(height: 10,),
                Row(mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                  children: [Icon(Icons.currency_ruble,size: 15,color: Color(0xff7b3f00) ,),
                    RichText(text: TextSpan(text: "PROFIT:", style: TextStyle(
                        color: Color(0xff7b3f00), fontWeight: FontWeight.bold, fontSize: 16,
                      ),), textAlign: TextAlign.center,),],), Divider(),
                RichText(text: TextSpan(text: "₱${totalProfit?.toStringAsFixed(2)}",
                  style: TextStyle(color: Color(0xff7b3f00),
                    fontWeight: FontWeight.normal, fontSize: 15,),),),],),),],),
      SizedBox(height: 10,),
      Row(mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        mainAxisSize: MainAxisSize.max,
        children: [
          ElevatedButton(onPressed: downloadSummaryReports,
            child: Text('Download Report'),),
          ElevatedButton(onPressed: () async {
              try {await sunmiReceipt.printSummary();
                ScaffoldMessenger.of(context).showSnackBar(
                  SnackBar(content: Text('Receipt printed successfully'),),);
              } catch (error) {ScaffoldMessenger.of(context).showSnackBar(
                  SnackBar(content: Text('Error printing receipt'),),);} },
            child: Text('Print Report'),),],), SizedBox(height: 50,),],),),); }
Future<void> _renderChartAsImage() async {
    final ui.Image data =
    await _cartesianChartKey.currentState!.toImage(pixelRatio: 3.0);
    final ByteData? bytes =
    await data.toByteData(format: ui.ImageByteFormat.png);
    final Uint8List imageBytes =
    bytes!.buffer.asUint8List(bytes.offsetInBytes, bytes.lengthInBytes);
    await Navigator.of(context).push<dynamic>(
      MaterialPageRoute<dynamic>(builder: (BuildContext context) {
            return Scaffold(body: Image.memory(imageBytes)); }),);}}
class SalesData{
  final String month; final double sales;
  final Color? color; SalesData(this.month, this.sales, this.color);}
```

sales_forecast.dart

```
class SalesForecast extends StatefulWidget {
  SalesForecast({Key? key}) : super(key: key);

  @override
  State<SalesForecast> createState() => _SalesForecastState();}
class _SalesForecastState extends State<SalesForecast> {
  User? user = FirebaseAuth.instance.currentUser;
  List<SalesData> chartData = <SalesData>[];
  late ZoomPanBehavior _zoomPanBehavior;

  @override
  void initState() {
    _zoomPanBehavior = ZoomPanBehavior(zoomMode: ZoomMode.x,
            enablePinching: true,
      enableDoubleTapZooming: true, enableSelectionZooming: true,
selectionRectBorderColor: Colors.red,
      selectionRectBorderWidth: 1, selectionRectColor: Colors.grey,); super.initState();}

  @override
  Widget build(BuildContext context) {return _showChart();}
  Widget _showChart() {return Scaffold(
      body: StreamBuilder<DocumentSnapshot<Map<String, dynamic>>>(
        stream:
_firestore.collection("predictions").doc(user!.uid).collection("sales").doc("data").
            snapshots(),
        builder: (context, snapshot) {if (!snapshot.hasData || !snapshot.data!.exists) {
            return Center(child: CircularProgressIndicator());}
          List<SalesData> chartDataList = [];
          var data = snapshot.data!.data();
          if (data != null && data.containsKey("predictions")) {
            List<dynamic> predictions = data['predictions'];
            for (var prediction in predictions) {
              DateTime date = (prediction['date'] as Timestamp).toDate();
              int y = (prediction['prediction'] as double).toInt();
              chartDataList.add(SalesData(date, y));}}
          if (chartDataList.isEmpty) {
            return Center(child: Text("No data found",style: TextStyle(fontSize: 16),),);}
          int minYValue = chartDataList.map((data) => data.y).
            reduce((a, b) => a < b ? a : b);
          return Padding(padding: EdgeInsets.all(8),
            child: Container(height: 300
              child: SfCartesianChart(zoomPanBehavior: _zoomPanBehavior,
                backgroundColor: Color(0xfff5e5de), plotAreaBackgroundColor:Colors.white,
                plotAreaBorderWidth: 5, borderColor: Color(0xff7b3f00), borderWidth: 5,
                margin: EdgeInsets.all(5),
                primaryXAxis: DateTimeAxis(labelRotation: -90,
                   labelStyle: TextStyle(fontSize: 14,
                    fontWeight: FontWeight.bold,),),
                primaryYAxis: NumericAxis(numberFormat: NumberFormat.currency(
                        locale: 'en_In',
                    symbol: "₱",),minimum: minYValue.toDouble(),),
                title: ChartTitle(text: 'Sales'),
                legend: Legend(isVisible: false, offset: Offset(10, 50),),
                tooltipBehavior: TooltipBehavior(enable: true),
                series: <ChartSeries<SalesData, DateTime>>[
                  LineSeries<SalesData, DateTime>(dataSource: chartDataList,
                    xValueMapper: (SalesData data, _) => data.x,
                    yValueMapper: (SalesData data, _) => data.y,
                    pointColorMapper: (SalesData sales, _) => sales.color,
                    name: 'Sales', dataLabelSettings: DataLabelSettings(isVisible: true),
                    color: Theme.of(context).primaryColor, xAxisName: 'Days',
                        yAxisName: 'Sales',),],),),);},),);}}

class SalesData {final DateTime x; final int y; final Color color;
  SalesData(this.x, this.y): color = Color(0xff7b3f00);}
```

demand_forecast.dart

```
class DemandForecast extends StatefulWidget {
  DemandForecast({Key? key}) : super(key: key);

  @override
  _DemandForecastState createState() => _DemandForecastState();}
class _DemandForecastState extends State<DemandForecast> {
  List<ChartData> chartData = [];List<String> distinctCategories = [];
  Set<String> selectedCategories = {};
  @override
  void initState() {super.initState();fetchData();selectAllCategories();
  void selectAllCategories() {setState(() {selectedCategories =
distinctCategories.toSet();});}
  @override
  Widget build(BuildContext context) {return LayoutBuilder(
     builder: (context, constraints) {return Scaffold(
        body: Center(child: Column(children: [
            Expanded(flex: 3, child: chartData.isNotEmpty? Container(
               height: constraints.maxHeight * 0.75, child: SfCartesianChart(
                 zoomPanBehavior: ZoomPanBehavior(
                   enablePanning: true, enablePinching: true,),
                 backgroundColor: Color(0xfff5e5de),
                   plotAreaBackgroundColor: Colors.white,
                 plotAreaBorderWidth: 5, borderColor: Color(0xff7b3f00), borderWidth: 5,
                 margin: EdgeInsets.all(5), primaryXAxis: CategoryAxis(
                   labelRotation: 90, ),
                 primaryYAxis: NumericAxis(),isTransposed: true, series: _getSeries(),
                 legend: Legend(isVisible: true, position: LegendPosition.bottom,
                   toggleSeriesVisibility: true,
                   overflowMode: LegendItemOverflowMode.wrap,),),)
                 : CircularProgressIndicator(),),
            SizedBox(height: 10), Text('CATEGORIES:', style: TextStyle(fontSize: 16),),
            Expanded(flex: 1,
              child: ListView(padding: EdgeInsets.symmetric(vertical: 8, horizontal: 16),
               children: [CheckboxListTile(title: Text('Select All',
                   style: TextStyle(fontSize: 14),),
                   value: selectedCategories.length == distinctCategories.length,
                   onChanged: (value) {setState(() {if (value!) {
                        selectedCategories = distinctCategories.toSet();} else {
         selectedCategories.clear();}});},),…distinctCategories.map((category) {return
         CheckboxListTile(title: Text(category, style: TextStyle(fontSize: 14),),
                    value: selectedCategories.contains(category), onChanged: (value) {
                      setState(() {if (value!) {selectedCategories.add(category); } else {
                         selectedCategories.remove(category);}});},);}).toList(),],),),],),),);},);
List<ChartSeries<ChartData, String>> _getSeries() {
    final List<String> visibleCategories = selectedCategories.toList();
    final int colorCount = visibleCategories.length;
    final List<Color> colors = _getDistinctColors(colorCount);
    return visibleCategories.map((category) {final bool isVisible = true; final Color
categoryColor =colors[visibleCategories.indexOf(category) % colorCount];
      return BarSeries<ChartData, String>(
        dataSource: chartData.where((data) => data.category == category).toList(),
        xValueMapper: (ChartData data, _) =>
           DateFormat('MMM dd,yyyy').format(data.date),
        yValueMapper: (ChartData data, _) => data.percentage,
        dataLabelSettings: DataLabelSettings(isVisible: true),
        color: categoryColor, // Use the category color for the bar serieisVisible: isVisible,
        name: category,); }).toList();}
  List<Color> _getDistinctColors(int count) {final List<Color> colors = [];
    for (int i = 0; i < count; i++) {final double hue = (360 / count) * i;
      final Color color = HSVColor.fromAHSV(1.0, hue, 0.6, 0.8).toColor();
            colors.add(color);}
    return colors;}}

class ChartData {final String category; final double percentage; final DateTime date;
  ChartData({required this.category,required this.percentage, required this.date,});}
```

pos.dart

```
class Pos extends StatefulWidget{
  const Pos ({super.key});

  @override
  State<Pos> createState() => _PosState();}
class _PosState extends State<Pos>with SingleTickerProviderStateMixin {
  final CartServices _cart = CartServices();
  late TabController tabController;

  @override
  void initState(){tabController = TabController(length:14, vsync:this);
      super.initState();}

  @override
  void dispose(){tabController.dispose();
      super.dispose();}
  @override
  Widget build(BuildContext context){
    final _cartProvider = Provider.of<CartProvider>(context);
    _cartProvider.getCartTotal();
    return Scaffold(
      appBar: AppBar(toolbarHeight: 70, elevation: 1,
        backgroundColor: const Color(0xffffffff),
        leading: Builder(builder: (BuildContext){
          return IconButton(color: Theme.of(context).primaryColor,
            icon: const Icon(Icons.arrow_back), onPressed: () => Homepage()));},);},),
        title: Text('POS',style: Theme.of(context).textTheme.headline1,),),
      body:SingleChildScrollView(child: Column(children:[ SizedBox(height: 10,),
              Padding(padding:EdgeInsets.only(left: 5,right: 5),
                child: Container(height: MediaQuery.of(context).size.height,
                  child: Column(children: [
                      Container(width: MediaQuery.of(context).size.height,
                    decoration: BoxDecoration(color: Color(0xfff9f2e4),
                      borderRadius: BorderRadius.circular(20)),
                    child: Column(children: [Padding(padding: EdgeInsets.all(2),
                          child: TabBar(unselectedLabelColor: Colors.black,
                            labelColor: Colors.white,
                            labelStyle: TextStyle(fontSize: 16),
                            unselectedLabelStyle: TextStyle(fontSize: 16),
                            indicatorColor: Colors.white, indicatorWeight: 2,
                            indicator: BoxDecoration(color: Color(0xffb7410e),
                              borderRadius: BorderRadius.circular(20),),
                            controller: tabController, isScrollable: true,
                            tabs: [Tab(text: 'ALL',), Tab(text:'Baking & Cooking'),
                              Tab(text:'Beverages'), Tab(text:'Bread & Breakfast'),
                              Tab(text:'Canned Goods',),
                              Tab(text:'Coffee, Tea & Cocoa'),
                              Tab(text:'Dairy & Eggs'),Tab(text:'Household'),
                              Tab(text:'Instant & Ready-to-eat'),
                              Tab(text:'Personal Care & Beauty'),
                              Tab(text:'Rice, Pasta & Grains'), Tab(text:'Snacks & Sweets'),
                              Tab(text:'Spread, Jams & Honey',), Tab(text:'Others'),],),),],),),
                  Expanded(
                    child: TabBarView(controller: tabController,
                      children: const [PosAll(),PosBaking(),PosBeverages(),PosBread(),
                        PosCanned(),PosCoffee(),PosDairy(),PosHousehold(),PosInstant(),
                        PosPersonalCare(),PosRice(),PosSnacks(),PosSpread(),PosOthers(),
                      ],),),],),),),],),),
        floatingActionButtonLocation: FloatingActionButtonLocation.centerFloat,
        floatingActionButton: FloatingActionButton.extended(
          label:Badge(badgeContent: Text('${_cartProvider.cartQty}', style: TextStyle(
                color: Colors.white,),),
            badgeColor: Theme.of(context).primaryColor,
            position: BadgePosition.topEnd(top:-18, end:-25),
            child: Container(alignment: Alignment.center,
              child: Row(children: [Icon(Icons.shopping_cart), Text('REVIEW',),],),),
            alignment: Alignment.topRight,), tooltip: 'REVIEW',
          onPressed: (){_cart.calculateTotalQuantity();
            Navigator.push(context,MaterialPageRoute(builder: (context)=> ItemSummary()
          ));},
          backgroundColor: Color(0xffa81c07), foregroundColor: Colors.white, ),
        bottomNavigationBar: BottomAppBar(color: const Color(0xffb7410e),
          child: Container(padding: EdgeInsets.only(left:40, right: 40), height: 60,
            child: Row(mainAxisSize: MainAxisSize.max,
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: <Widget>[MaterialButton(minWidth: 50,
                  onPressed: ()=>const Homepage()));},
                  child:Column(mainAxisAlignment: MainAxisAlignment.center,
                children: <Widget>[Icon(Icons.home, color: Colors.grey,),
                  Text('HOME', style: TextStyle(color: Colors.grey),),],),),
            MaterialButton(minWidth: 50, onPressed: () => POSBarcode(),));},
              child:Column(mainAxisAlignment: MainAxisAlignment.center,
                children: <Widget>[
                  Icon(Icons.qr_code_scanner_outlined, color: Color(0xffffc000),),
                  Text('SCAN', style: TextStyle(color: Color(0xffffc000),),),],),),
            MaterialButton(minWidth: 50, onPressed: (){},
              child:Column(mainAxisAlignment: MainAxisAlignment.center,
                children: <Widget>[Icon(Icons.point_of_sale, color: Colors.white,),
                  Text('POS', style: TextStyle(color: Colors.white),),],),),],),),),); }}
```

pos_all.dart

```
class PosAll extends StatefulWidget{
  const PosAll ({super.key});

  @override
  State<PosAll> createState() => _PosAllState();}
class _PosAllState extends State<PosAll> {

  @override
  Widget build(BuildContext context){ return Scaffold(
      body:SingleChildScrollView(child: Column(
        children:[ const SizedBox(height: 15,),
          StreamBuilder<QuerySnapshot>(
            stream: _firestore.instance.collection("users")
                .doc(FirebaseAuth.instance.currentUser!.uid)
                .collection("products").orderBy('productName',
                    descending: false).snapshots(),
            builder: (BuildContext context,
                AsyncSnapshot<QuerySnapshot> snapshot) {
              if (snapshot.hasError) {return Center(
                  child: Text('Some error occurred ${snapshot.error}')); } else {
                return  GridView.builder(
                  physics: const NeverScrollableScrollPhysics(),shrinkWrap: true,
                  gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
                      crossAxisCount: 3, crossAxisSpacing: 10, mainAxisExtent: 230,
                      mainAxisSpacing: 10),
                  itemCount: snapshot.data == null ? 0 : snapshot.data!.docs.length,
                  itemBuilder: (_, index){
                    DocumentSnapshot documentSnapshot = snapshot.data!.docs[index];
                    String id = snapshot.data!.docs[index].id;
                    return Container(decoration: BoxDecoration(
                        borderRadius: BorderRadius.circular(2,),
                        color: const Color(0xfff9f2e4),),
alignment: Alignment.center,
child: Column(crossAxisAlignment: CrossAxisAlignment.start,
   children: [Padding(padding: const EdgeInsets.only(left:5, right: 5),
      child: Column(crossAxisAlignment: CrossAxisAlignment.start,
        children: [const SizedBox(height: 5,),
          Container(width: double.infinity, height: 90,
            decoration: BoxDecoration(color:const Color(0xffFDB73E),
              border: Border.all(color: const Color(0xffFDB73E),width: 2,),
              borderRadius: const BorderRadius.only(
              topRight: Radius.circular(2), topLeft: Radius.circular(2),
                bottomLeft: Radius.circular(2),
                    bottomRight: Radius.circular(2),),),
            alignment: Alignment.center,
            child: ClipRRect(borderRadius: const BorderRadius.only(
                topLeft: Radius.circular(2.0), topRight: Radius.circular(2.0),),
              child: documentSnapshot['productImage'].isNotEmpty
                  ? Image.network(documentSnapshot['productImage'],
                height: double.infinity, width: double.infinity,
                fit: BoxFit.cover,)
                  : Image.asset("asset/images/default_image.png",
                height: double.infinity, width: double.infinity,
                fit: BoxFit.cover,),),),
          Container(width: double.infinity, height: 40,
            alignment: Alignment.centerLeft,
            child: Text(documentSnapshot['productName'], maxLines: 2,
              overflow: TextOverflow.ellipsis,
              style: const TextStyle(fontSize: 15, ),),),
          const SizedBox(height: 5,),
          Container(height: 30, width: double.infinity,
            alignment: Alignment.centerLeft,
            child:  Column(children: [
                Row(children: [const Text("Price: ",
                        style: TextStyle(fontSize: 12,)),
                    Text('₱${documentSnapshot['productPrice']}',
                      style: const TextStyle(fontSize: 12,),),],),
                Row(children: [const Text("Stocks:",
                        style: TextStyle(fontSize: 12,)),
                    const SizedBox(width: 5,),
                    Text(documentSnapshot['productStocks'].toString(),
                      style: const TextStyle(fontSize: 12,),),],),],),),
          SizedBox(height: 5,),
          Container(alignment: Alignment.center,
            child: const Text('Quantity: ', style: TextStyle(fontSize: 12),),),
          Row(mainAxisSize: MainAxisSize.max,
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: [
                                           Counter(key: ValueKey(documentSnapshot['productCode']),
                                            documentSnapshot: documentSnapshot,docId: id,)]),],),)]));
                                             },);}},),const SizedBox(height: 500,)],),),); }}
```

pos_categ.dart

```
class PosBaking extends StatefulWidget{
  const PosBaking ({super.key});

  @override
  State<PosBaking> createState() => _PosBakingState();}
class _PosBakingState extends State<PosBaking> {

  @override
  Widget build(BuildContext context){ return Scaffold(
      body:SingleChildScrollView(child: Column(
          children:[ const SizedBox(height: 15,),
            StreamBuilder<QuerySnapshot>(
              stream: FirebaseFirestore.instance.collection("users")
                  .doc(FirebaseAuth.instance.currentUser!.uid).collection("products")
                  .where('productCategory', isEqualTo: 'Baking & Cooking')
                  .orderBy('productName', descending: false).snapshots(),
              builder: (BuildContext context, AsyncSnapshot<QuerySnapshot> snapshot) {
                if (snapshot.hasError) {return Center(
                    child: Text('Some error occurred ${snapshot.error}')); } else {
                  return  GridView.builder(physics: const NeverScrollableScrollPhysics(),
                    shrinkWrap: true,
                    gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
                        crossAxisCount: 3, crossAxisSpacing: 10, mainAxisExtent: 230,
                        mainAxisSpacing: 10),
                    itemCount: snapshot.data == null ? 0 : snapshot.data!.docs.length,
                    itemBuilder: (_, index){ DocumentSnapshot documentSnapshot =
                      snapshot.data!.docs[index]; String id = snapshot.data!.docs[index].id;
                      return Container(decoration: BoxDecoration(
                            borderRadius: BorderRadius.circular(2,), color: Color(0xfff9f2e4),),
                          alignment: Alignment.center,
                          child: Column(crossAxisAlignment: CrossAxisAlignment.start,
                              children: [Padding(padding: const EdgeInsets.only(left:5, right: 5),
                                  child: Column(crossAxisAlignment: CrossAxisAlignment.start,
                                    children: [const SizedBox(height: 5,),
                                      Container(width: double.infinity, height: 90,
                                        decoration: BoxDecoration(color:const Color(0xffFDB73E),
                                          border: Border.all(color: const Color(0xffFDB73E),width: 2,),
                                          borderRadius: const BorderRadius.only(
                                           topRight: Radius.circular(2), topLeft: Radius.circular(2),
                                            bottomLeft: Radius.circular(2),
                                         bottomRight: Radius.circular(2),),),
                                       alignment: Alignment.center,
                                       child: ClipRRect(borderRadius: const BorderRadius.only(
                                         topLeft: Radius.circular(2.0),topRight: Radius.circular(2.0),),
                                        child: documentSnapshot['productImage'].isNotEmpty
                                          ? Image.network(documentSnapshot['productImage'],
                                         height: double.infinity, width: double.infinity,
                                         fit: BoxFit.cover,)
                                          : Image.asset("asset/images/default_image.png",
                                         height: double.infinity, width: double.infinity,
                                         fit: BoxFit.cover,), ),),
                                      Container(width: double.infinity, height: 40,
                                       alignment: Alignment.centerLeft,
                                       child: Text(documentSnapshot['productName'], maxLines: 1,
                                        style: const TextStyle(fontSize: 15, ),),), SizedBox(height: 5,),
                                      Container(height: 30, width: double.infinity,
                                       alignment: Alignment.centerLeft, child:  Column(children: [
                                         Row(children: [Text("Price: ",style: TextStyle(fontSize: 12,)),
                                           Text('₱${documentSnapshot['productPrice']}',
                                            style: const TextStyle(fontSize: 12,),),],),
                                         Row(children: [Text("Stocks:",
                                              style: TextStyle(fontSize: 12,)),SizedBox(width: 5,),
                                           Text(documentSnapshot['productStocks'].toString(),
                                            style: const TextStyle(fontSize: 12,),),],),],),),
                                      SizedBox(height: 5,),
                                      Container(alignment: Alignment.center,
                                       child: const Text('Quantity: ', style: TextStyle(fontSize: 12),),),
                                      Row(mainAxisSize: MainAxisSize.max,
                                        mainAxisAlignment: MainAxisAlignment.spaceBetween,
                                        children: [
                                         Counter(documentSnapshot: documentSnapshot, docId: id,)]
                                      ),],),)])); },);}},),const SizedBox(height: 500,)],),),); }}
```

pos_barcode.dart

```
class POSBarcode extends StatefulWidget{
  const POSBarcode({super.key});

  @override
  State<POSBarcode> createState() => _POSBarcodeState();}
class _POSBarcodeState extends State<POSBarcode> {
  CartServices _cart = CartServices();
  ProductServices _product = ProductServices();
  User? user = FirebaseAuth.instance.currentUser;
  String scannedProductCode = '';
  bool isLoading = false;
```

```
String productCode = '';
String productCategory = '';
String productName = '';
double productPrice = 0;
double productCost = 0;
int productStocks = 0;
int productQuantity = 0;

@override
Widget build(BuildContext context){ MediaQueryData queryData;
  queryData = MediaQuery.of(context);
  final _cartProvider = Provider.of<CartProvider>(context);
  _cartProvider.getCartTotal();
  return Scaffold(backgroundColor: Color(0xfff9f2e4),
      appBar: AppBar(elevation: 1, backgroundColor: Color(0xfff9f2e4),
        title: Text('Barcode Scan',style: Theme.of(context).textTheme.headline1,),
        leading: Builder(builder: (BuildContext){
          return IconButton(color: Theme.of(context).primaryColor,
            icon: const Icon(Icons.arrow_back), onPressed: () => Pos()));},);},),),
      body:SingleChildScrollView(child: Column(children: [SizedBox(height: 20,),
            Container(padding:EdgeInsets.only(left: 15 ,right: 15),
              child: RichText(text: TextSpan(text: 'START SCANNING PRODUCT',
                style: TextStyle(color: Color(0xff7b3f00), fontWeight: FontWeight.bold,
                  fontSize: 30,),), textAlign: TextAlign.center,),), SizedBox(height: 10,),
            Row(mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: [SizedBox(width: 5,),
                Container(padding:EdgeInsets.only(left: 20),
                  child: Text(DateFormat.yMMMd().format(DateTime.now()),
                    style:  TextStyle(fontSize: 16),),
                  alignment: Alignment.centerLeft,),
                Container(padding:EdgeInsets.only(left: 20),
                  child: Text(DateFormat('KK:mm:ss a').format(DateTime.now()),
                    style:  TextStyle(fontSize: 16),),
                  alignment: Alignment.centerLeft,), SizedBox(width: 5,),],),
            SizedBox(height: 10,),
            Row(mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: [SizedBox(width: 5,),
                Container(child: Text('SCANNER:', style: TextStyle(fontSize: 16),),),
                ElevatedButton(onPressed: (){scanProductBarcode();},
                    child: Icon(Icons.qr_code_scanner_rounded)), SizedBox(width: 5,),],),
            SizedBox(height: 20,),
            Row(mainAxisSize: MainAxisSize.max,
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: [Container( padding:EdgeInsets.only(left: 20),
                  child: Text('Total Amount:',style:  TextStyle(fontSize: 24),),),
                Container(padding:EdgeInsets.only(right: 20),
                child: Text('₱ ${_cartProvider.subTotal.toStringAsFixed(0)}',
                 style:  TextStyle(fontSize: 24),),),],), SizedBox(height: 10,),
          Container(width: double.infinity, decoration: const BoxDecoration(
                color: Colors.white, borderRadius: BorderRadius.only(topRight:
                Radius.circular(50), topLeft: Radius.circular(50)),),
              child: Column(crossAxisAlignment: CrossAxisAlignment.center,
                children: [SizedBox(height: 10,), Text('Items', textAlign: TextAlign.center,
                  style:  TextStyle(fontSize: 24),), SizedBox(height: 10,),
                 Container(padding: EdgeInsets.only(left: 7,right: 7),
                  child: Row(mainAxisSize: MainAxisSize.max,
                   mainAxisAlignment: MainAxisAlignment.spaceBetween,
                   children: <Widget>[
                    RichText(text: TextSpan(text: 'Quantity', style: TextStyle(
                        color: Color(0xff7b3f00), fontWeight: FontWeight.bold,
                        fontSize: 16,),),),
                    SizedBox(width: 5,),
                    RichText(text: TextSpan(text: 'Item', style: TextStyle(
                        color: Color(0xff7b3f00), fontWeight: FontWeight.bold,
                        fontSize: 16,),),), SizedBox(width: 5,),
                    RichText(text: TextSpan(text: 'Amount', style: TextStyle(
                        color: Color(0xff7b3f00), ontWeight: FontWeight.bold,
                        fontSize: 16,),),),],),), SizedBox(height: 5,), CartList(),
                 Container(height: 50, margin: EdgeInsets.only(bottom:
                 MediaQuery.of(context).size.height*0.5),
                  decoration: const BoxDecoration(color: Colors.white,
                   borderRadius: BorderRadius.only(bottomLeft: Radius.circular(50),
                   bottomRight: Radius.circular(50)),),),]),),]),),
    floatingActionButtonLocation: FloatingActionButtonLocation.centerFloat,
    floatingActionButton: FloatingActionButton.extended(
      label: Text('CHECKOUT',), tooltip: 'CHECKOUT',
      onPressed: () => Payable()));},backgroundColor: Color(0xffa81c07),
      foregroundColor: Colors.white,),); }}
```

item_summary.dart

```
class ItemSummary extends StatefulWidget{
  const ItemSummary ({super.key});

  @override
  State<ItemSummary> createState() => _ItemSummaryState();}
class _ItemSummaryState extends State<ItemSummary> {
  DateTime selectedDate = DateTime.now();
  final CartServices _cart = CartServices();
  int totalQuantity = 0;

  @override
```

```
void initState() {super.initState();
  _cart.calculateTotalQuantity().then((quantity) {
    setState(() {totalQuantity = quantity; });});}

@override
Widget build(BuildContext context){
  MediaQueryData queryData;
  queryData = MediaQuery.of(context);
  final _cartProvider = Provider.of<CartProvider>(context);
  _cartProvider.getCartTotal();
  return Scaffold(backgroundColor: Color(0xfff9f2e4),
    appBar: AppBar(elevation: 1,backgroundColor: Color(0xfff9f2e4),
      title: Text('Items Summary',style: Theme.of(context).textTheme.headline1,),
      leading: Builder(builder: (BuildContext){
        return IconButton(color: Theme.of(context).primaryColor,
          icon: const Icon(Icons.arrow_back), onPressed: ()=> Pos()));},);},),),
    body:SingleChildScrollView(child: Column(children: [SizedBox(height: 20,),
            Container(margin: EdgeInsets.only(left: 10,right: 10),
              padding: EdgeInsets.only(left: 5,right: 5), height: 80,
              decoration: BoxDecoration(color: Color(0xfff9f2e4), border: Border.all(
                  width: 2, color: Color(0xffffc000),),
                borderRadius: BorderRadius.all(Radius.circular(25)),),
              alignment: Alignment.center,
              child: Column(children: [
                  Row(mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                    children: [Icon(Icons.calendar_today_rounded, color: Color(0xff7b3f00),),
                      Text(DateFormat.yMMMd().format(selectedDate),
                        style:  TextStyle(fontSize: 24,color: Color(0xff7b3f00)),),
                      OutlinedButton(onPressed: () async {
                            final DateTime? dateTime = await showDatePicker(context: context,
                              initialDate: selectedDate,
                              firstDate: DateTime(1980),
                              lastDate: DateTime(3000),);
                            if (dateTime != null){ setState(() {selectedDate = dateTime; });}},
                          child: Text("Select Date",style:  TextStyle(fontSize: 20,
                                  color:Color(0xff7b3f00)),)),],),
                  Center(child: Text(DateFormat('KK:mm:ss a').format(DateTime.now()),
                      style:  TextStyle(fontSize: 20),),),],),),
            SizedBox(height: 10,), Divider(),
            Row(mainAxisSize: MainAxisSize.max,
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: [Container(padding:EdgeInsets.only(left: 20),
                  child: Text('Total Items:',style:  TextStyle(fontSize: 24),),),

                Container(padding:EdgeInsets.only(right: 20),
                  child: Text('$totalQuantity',style:  TextStyle(fontSize: 24),),),],),
            SizedBox(height: 2,),
            Row(mainAxisSize: MainAxisSize.max,
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: [Container(padding:EdgeInsets.only(left: 20),
                  child: Text('Total Amount:',style:  TextStyle(fontSize: 24),),),
                Container(padding:EdgeInsets.only(right: 20),
                  child: Text('₱ ${_cartProvider.subTotal.toStringAsFixed(0)}',
                    style:  TextStyle(fontSize: 24),),),],),
            SizedBox(height: 10,),
            Container(width: double.infinity, decoration: const BoxDecoration(
                color: Colors.white, borderRadius: BorderRadius.only(topRight:
                    Radius.circular(50), topLeft: Radius.circular(50)),),
              child: Column(crossAxisAlignment: CrossAxisAlignment.center,
                children: [SizedBox(height: 10,),
                  Text('Items', textAlign: TextAlign.center, style:  TextStyle(fontSize: 24),),
                  SizedBox(height: 10,),
                  Container(padding: EdgeInsets.only(left: 7,right: 7),
                    child: Row(mainAxisSize: MainAxisSize.max,
                      mainAxisAlignment: MainAxisAlignment.spaceBetween,
                      children: <Widget>[RichText(text: TextSpan(text: 'Quantity',
                          style: TextStyle(color: Color(0xff7b3f00),
                            fontWeight: FontWeight.bold, fontSize: 16,),),),
                        SizedBox(width: 5,),
                        RichText(text: TextSpan(text: 'Item', style: TextStyle(
                            color: Color(0xff7b3f00), fontWeight: FontWeight.bold,
                            fontSize: 16,),),), SizedBox(width: 5,),
                        RichText(text: TextSpan(text: 'Amount', style: TextStyle(
                            color: Color(0xff7b3f00), fontWeight: FontWeight.bold,
                            fontSize: 16,),),),],),),
                  SizedBox(height: 5,), CartList(),
                  Container(height: 50,
                    margin: EdgeInsets.only(bottom: MediaQuery.of(context).size.height*0.5),
                    decoration: const BoxDecoration(color: Colors.white,
                      borderRadius: BorderRadius.only(bottomLeft: Radius.circular(50),
                      bottomRight: Radius.circular(50)),),),  ])),])),
        floatingActionButtonLocation: FloatingActionButtonLocation.centerFloat,
        floatingActionButton: FloatingActionButton.extended(
          label: Text('CHECKOUT',), tooltip: 'CHECKOUT', onPressed: ()  => Payable()));},
          backgroundColor: Color(0xffa81c07), foregroundColor: Colors.white,),); }}
```

payable.dart

```
class Payable extends StatefulWidget{
  const Payable({Key? key});

  @override
  State<Payable> createState() => _PayableState();}
class _PayableState extends State<Payable> {
  OrderServices _orderServices = OrderServices();
  ProductServices _productServices = ProductServices();
  CartServices _cart = CartServices();

  DateTime selectedDate = DateTime.now();
  late String orderNumber;
  String amountRcvd = '';
  late TextEditingController moneyRcvd;
  String change = '';
  int totalQuantity = 0;

  @override
  void initState() {super.initState();
    moneyRcvd = TextEditingController(text: amountRcvd);
    orderNumberGenerator();
    _cart.calculateTotalQuantity().then((quantity) {setState(() {totalQuantity = quantity;
      });});}

  void updateChange() {
    double payable = Provider.of<CartProvider>(context, listen: false).subTotal;
    double received = double.tryParse(amountRcvd) ?? 0;
    double changeAmount = received - payable;
    setState(() {change = changeAmount.toStringAsFixed(0); });}

  Future<String> orderNumberGenerator() async {
    DateTime now = selectedDate;
    orderNumber = '${now.month.toString().padLeft(2,
'0')}${now.day.toString().padLeft(2, '0')}${now.year}${now.hour.toString().padLeft(2,
'0')}${now.minute.toString().padLeft(2, '0')}${now.second.toString().padLeft(2, '0')}';
    return orderNumber; }

  @override
  Widget build(BuildContext context){
    final _cartProvider = Provider.of<CartProvider>(context);
    _cartProvider.getCartTotal();

    return Scaffold(appBar: AppBar(elevation: 1,
        backgroundColor: const Color(0xffffffff),
        title: Text('Payable',style: Theme.of(context).textTheme.displayLarge,),
        leading: Builder(builder: (BuildContext){
          return IconButton(color: Theme.of(context).primaryColor,
            icon: const Icon(Icons.arrow_back), onPressed: () => ItemSummary()));},);},),),
      body: SingleChildScrollView(child: Column(children: [SizedBox(height: 10,),
          Container(margin: EdgeInsets.only(left: 10,right: 10),
  padding: EdgeInsets.only(left: 5,right: 5), height: 80,
  decoration: BoxDecoration(border: Border.all(width: 5,
     color: Color(0xffffc000),),
    borderRadius: BorderRadius.all(Radius.circular(5)),),
  alignment: Alignment.center,
  child: Column(children: [SizedBox(height: 10,),
     Row(mainAxisAlignment: MainAxisAlignment.spaceEvenly,
       children: [
         RichText(text: TextSpan(text: "Amount Payable:",
             style: TextStyle(color: Color(0xff7b3f00), fontWeight: FontWeight.bold,
               fontSize: 20,),), textAlign: TextAlign.center,),],), Divider(),
     RichText(text: TextSpan(
         text: '₱ ${_cartProvider.subTotal.toStringAsFixed(0)}', style: TextStyle(
           color: Color(0xff7b3f00), fontWeight: FontWeight.normal, fontSize: 18,),
       ),),],),), SizedBox(height: 5,),
Container(margin: EdgeInsets.only(left: 10,right: 10),
  padding: EdgeInsets.only(left: 5,right: 5), height: 80,
  decoration: BoxDecoration(border: Border.all(width: 5,
     color: Color(0xffb7410e),),
    borderRadius: BorderRadius.all(Radius.circular(5)),),
  alignment: Alignment.center,
  child: Column(children: [SizedBox(height: 10,),
     Row(mainAxisAlignment: MainAxisAlignment.spaceEvenly,
       children: [
         RichText(text: TextSpan(text: "Amount Received:", style: TextStyle(
               color: Color(0xff7b3f00), fontWeight: FontWeight.bold, fontSize: 20,),
           ), textAlign: TextAlign.center,),],), Divider(),
     RichText(text: TextSpan(text: "₱ $amountRcvd",style: TextStyle(
           color: Color(0xff7b3f00), fontWeight: FontWeight.normal, fontSize: 18,
         ),),),],),),
Padding(padding: EdgeInsets.only(left: 30, right: 30), child: TextFormField(
    controller: moneyRcvd, style: TextStyle(color: Color(0xFF7b3f00)),
    decoration: InputDecoration(
      icon: Icon(Icons.currency_ruble, color: Color(0xFF7b3f00)),
      labelText: 'Type or select bill tendered',),
    onChanged: (value) {setState(() {amountRcvd = value; updateChange();});},),
SizedBox(height: 5,), Divider(),
Row(mainAxisSize: MainAxisSize.max,
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: [OutlinedButton(onPressed: () {setState(() {amountRcvd = "10";
         moneyRcvd.text = "10";updateChange();});},child: Text('10'),),
    OutlinedButton(onPressed: () {setState(() {amountRcvd = "20";
         moneyRcvd.text = "20";updateChange();});},child: Text('20'),),
    OutlinedButton(onPressed: () {setState(() {amountRcvd = "50";
         moneyRcvd.text = "50";updateChange();});},child: Text('50'),),],),
Row(mainAxisSize: MainAxisSize.max,
          mainAxisAlignment: MainAxisAlignment.spaceEvenly,
          children: [OutlinedButton(onPressed: () {setState(() {amountRcvd = "100";
                moneyRcvd.text = "100";updateChange();});},child: Text('100'),),
            OutlinedButton(onPressed: () {setState(() {amountRcvd = "200";
                moneyRcvd.text = "200";updateChange();});},child: Text('200'),),
            OutlinedButton(onPressed: () {setState(() {amountRcvd = "300";
                moneyRcvd.text = "300";updateChange();});},child: Text('300'),),],s),
        Row(mainAxisSize: MainAxisSize.max,
          mainAxisAlignment: MainAxisAlignment.spaceEvenly,
          children: [OutlinedButton(onPressed: () {setState(() {amountRcvd = "400";
                moneyRcvd.text = "400";updateChange();});},child: Text('400'),),
            OutlinedButton(onPressed: () {setState(() {amountRcvd = "500";
                moneyRcvd.text = "500";updateChange();});},child: Text('500'),),
            OutlinedButton(onPressed: () {setState(() {amountRcvd = "1000";
                moneyRcvd.text = "1000";updateChange();});},child: Text('1000'),),],),
        OutlinedButton(onPressed: () {setState(() {
            amountRcvd = _cartProvider.subTotal.toStringAsFixed(0);
            moneyRcvd.text = _cartProvider.subTotal.toStringAsFixed(0);
            updateChange();});},child: Text('EXACT AMOUNT'),),
        Container(margin: EdgeInsets.only(left: 10, right: 10),
          padding: EdgeInsets.only(left: 5, right: 5), height: 80,
          decoration: BoxDecoration(border: Border.all(width: 5,
            color: Color(0xffffc000),),
            borderRadius: BorderRadius.all(Radius.circular(5)),),
          alignment: Alignment.center,
          child: Column(children: [SizedBox(height: 10,),
            Row(mainAxisAlignment: MainAxisAlignment.spaceEvenly,
              children: [
                RichText(ext: TextSpan(text: "Change:", style: TextStyle(
                    color: Color(0xff7b3f00), fontWeight: FontWeight.bold,
                    fontSize: 20,),), textAlign: TextAlign.center,),],),
            Divider(),
            RichText(text: TextSpan(text: "₱ $change",style: TextStyle(
                color: Color(0xff7b3f00), fontWeight: FontWeight.normal,
                fontSize: 18,),),),],),), SizedBox(height: 10,),
        Container(padding: const EdgeInsets.only(left: 25, right: 25),
          child: ElevatedButton(onPressed: () {orderNumberGenerator();
            _saveOrder(_cartProvider); _productServices.updateProductStocks();
            _productServices.updateSoldCount();
            Navigator.push(context, MaterialPageRoute(builder: (_)=> Receipt()));},
            child: Text('CONFIRM', style: TextStyle(color: Colors.white, fontSize: 25),),),
        ),],),),);}
_saveOrder(CartProvider cartProvider){
  _orderServices.saveOrder({
    'products' : cartProvider.cartList,
    'sales' : cartProvider.subTotal,
    'cost' : cartProvider.cartTotalCost,
    'timestamp' : selectedDate,
    'orderNo' : orderNumber,
    'totalQuantity' : totalQuantity, });}}
```

receipt.dart

```
class Receipt extends StatefulWidget{

  @override
  State<Receipt> createState() => _ReceiptState();}
class _ReceiptState extends State<Receipt> {
  final CartServices _cart = CartServices();
  final OrderServices _orders = OrderServices();

  @override
  void initState() {_orders.saveTotal();super.initState();}

  @override
  Widget build(BuildContext context){
    final _cartProvider = Provider.of<CartProvider>(context);
    _cartProvider.getCartTotal();
    return Scaffold(
      appBar: AppBar(elevation: 1, backgroundColor: const Color(0xffffffff),
        title: Text('Receipt', style: Theme.of(context).textTheme.headline1,),
        leading: Builder(builder: (BuildContext){
          return IconButton(color: Theme.of(context).primaryColor,
            icon: const Icon(Icons.arrow_back), onPressed: () => Payable()));},);},),),
      body:SingleChildScrollView(child: Column(
          children: [
            SizedBox(height: 10,),
            Container(child: Text('Receipt Preview', style: TextStyle(fontSize: 20),),),
            SizedBox(height: 400, child: ReceiptList(),),
            Container(color: Color(0xffffc000), padding: EdgeInsets.all(20),
              child: Row(children: [
                  Text("Total: ₱ ${_cartProvider.subTotal.toStringAsFixed(2)}",
                    style: TextStyle(fontWeight: FontWeight.bold),),
                  Expanded(child: Center(
                      child: OutlinedButton(onPressed: () => PrintPage()));},
                          style: ElevatedButton.styleFrom(backgroundColor: Colors.white,),
                          child: Text('PRINT',style: TextStyle(color: Colors.black, fontSize: 25),))
                    ),),],),), SizedBox(height: 10,),
            Container(padding: const EdgeInsets.only(left:25,right:25),
              child: ElevatedButton(onPressed: (){_orders.saveTotal();_cart.deleteAll();
                  Navigator.push(context, MaterialPageRoute(builder: (_)=> PosFinish()));},
                child: Text('FINISH', style: TextStyle(color: Colors.white, fontSize: 25),),),
        ),],),),); }}
```

print_page.dart

```
class PrintPage extends StatefulWidget{

  @override
  _PrintPageState createState()=> _PrintPageState();}

class _PrintPageState extends State<PrintPage>{

  final SunmiReceipt sunmiReceipt = SunmiReceipt();
  final CollectionReference _cartCollection =
FirebaseFirestore.instance.collection('cart').doc(user!.uid).collection('products');
  final CollectionReference _usersCollection =
FirebaseFirestore.instance.collection('users');
  final CollectionReference _ordersCollection =
FirebaseFirestore.instance.collection('orders').doc(user!.uid).collection('orders');

  BluetoothPrint bluetoothPrint = BluetoothPrint.instance;
  bool _connected = false;
  String tips = 'No Device Connect';
  final f = NumberFormat("\$#####.00","en_US");

  @override
  void initState(){
    super.initState();
    WidgetsBinding.instance.addPostFrameCallback((_)=>{initPrinter()});}
  Future<void> initPrinter() async {
    bluetoothPrint.startScan(timeout: Duration(seconds: 2));
    bool isConnected = await bluetoothPrint.isConnected??false;
    bluetoothPrint.state.listen((state){
      print('****************** cur device status: $state');
      switch (state){
        case BluetoothPrint.CONNECTED:
          setState(() {_connected = true; tips = 'Connect Success'; });
          break;
        case BluetoothPrint.DISCONNECTED:
          setState(() {_connected = false; tips = 'Disconnect Success'; });
          break;
        default:
          break;
      }
    });
    if (!mounted) return;
    if (isConnected){
```

```
    setState(() { _connected=true; });}}

  @override

  Widget build(BuildContext context){
    return Scaffold(
      appBar: AppBar(elevation: 1, backgroundColor: Color(0xffffffff),
        title: Text('Print Receipt',style: Theme.of(context).textTheme.displayLarge,),
        leading: Builder(builder: (BuildContext){
          return IconButton(color: Theme.of(context).primaryColor,
            icon: const Icon(Icons.arrow_back), onPressed: (){Get.back();},);},),),
      body: SingleChildScrollView(child: Column(children: [SizedBox(height: 5,),
            Container(margin: EdgeInsets.only(left: 10,right: 10),
              padding: EdgeInsets.only(left: 5,right: 5), height: 200,
              decoration: BoxDecoration(border: Border.all(width: 5,
                  color: Color(0xffffc000),),
                borderRadius: BorderRadius.all(Radius.circular(5)),),
              child: Column(children: [
                  Center(child: Text('Bluetooth Thermal Printer', style: TextStyle(fontSize: 25),
                    ),), SizedBox(height: 5,),
                  Container(padding: EdgeInsets.only(left: 8, right: 5),
                    child: Text('Note: Please check the connection to the printer.',
                      style: TextStyle(fontSize: 18, fontStyle: FontStyle.italic),),
                    alignment: Alignment.center,),
                  Row(mainAxisAlignment: MainAxisAlignment.center,
                    children: [SizedBox(width: 20,), Text('Check connection:'),
                      SizedBox(width: 8,),
                      OutlinedButton(onPressed: () => Devices()));},child: Text('Devices'))],),
                  OutlinedButton(child: Text('PRINT RECEIPT'),
                      onPressed:  _connected?() async {Map<String, dynamic> config = Map();
                        DocumentSnapshot userSnapshot = await
_usersCollection.doc(user!.uid).get();
                        String storeName = userSnapshot['storeName'];
                        String storeAddress = userSnapshot['storeAddress'];
                        String storeTin = userSnapshot['storeTin'];
                        double totalAmount = 0;
                        String qty = '';
                        QuerySnapshot orderSnapshot = await _ordersCollection
                            .orderBy('timestamp', descending: true).limit(1).get();
                        if (orderSnapshot.docs.isNotEmpty) {
                        DocumentSnapshot orderDoc = orderSnapshot.docs.first;
                        String orderNumber = orderDoc['orderNo'];
                        Timestamp timestamp = orderDoc['timestamp'];
                        String orderTotalQty = orderDoc['totalQuantity'].toString();

                        DateTime dateTime = timestamp.toDate();
                    String formattedDateTime = DateFormat('MM/dd/yy
HH:mm:ss').format(dateTime);

                    config['width'] = 40;
                    config['height'] = 70;
                    config['gap'] = 2;

                    List<LineText> list = [];
                    list.add(LineText(type: LineText.TYPE_TEXT,
                       content: '${storeName.toUpperCase()}',
                       weight: 2, width: 2, height: 2,
                       align: LineText.ALIGN_CENTER, linefeed: 1,),);
                    list.add(LineText(type: LineText.TYPE_TEXT,
                       content: '$storeAddress',
                       weight: 2, width: 2, height: 2,
                       align: LineText.ALIGN_CENTER, linefeed: 1,),);
                    list.add(LineText(type: LineText.TYPE_TEXT,
                       content: 'TIN: $storeTin', weight: 2,
                       width: 2, height: 2,align: LineText.ALIGN_LEFT, linefeed: 1,),);
                    list.add(LineText(type: LineText.TYPE_TEXT,
                       content: 'Order No: $orderNumber', weight: 2, width: 2,
                       height: 2, align: LineText.ALIGN_LEFT, linefeed: 1,),);
                    list.add(LineText(type: LineText.TYPE_TEXT,
                       content: 'Date: $formattedDateTime', weight: 2,
                       width: 2, height: 2, align: LineText.ALIGN_LEFT, linefeed: 1,),);
                    list.add(LineText(type: LineText.TYPE_TEXT,
                          content: '**********************************************',
                          weight: 1, align: LineText.ALIGN_CENTER, linefeed: 1));

                    QuerySnapshot snapshot = await _cartCollection.get();
                    List<QueryDocumentSnapshot> documents = snapshot.docs;
                    for (var document in documents) {
                    Map<String, dynamic> data = document.data() as Map<String, dynamic>;
                    String productName = data['productName'];
                    double productPrice = data['productPrice'];
                    int productQuantity = data['productQuantity'];
                    double productTotal = data['productTotal'];

                    totalAmount += productTotal;
                    qty = orderTotalQty;

                      list..add(LineText(
                       type: LineText.TYPE_TEXT,
                       content: productQuantity.toString(),
                       weight: 0, width: 2, align: LineText.ALIGN_LEFT, linefeed: 1,))
                       ..add(LineText(type: LineText.TYPE_TEXT, content: productName,
                       weight: 0, width: 2,align: LineText.ALIGN_CENTER, linefeed: 1,))
                       ..add(LineText(type: LineText.TYPE_TEXT,
                       content: productTotal.toStringAsFixed(2), weight: 0,
                       width: 2, align: LineText.ALIGN_RIGHT,linefeed: 1,));
                      list.add(LineText(type: LineText.TYPE_TEXT,
                        content: '@ ${productPrice.toStringAsFixed(2)}',
                        align: LineText.ALIGN_CENTER, linefeed: 1,),); }
                    list.add(LineText(type: LineText.TYPE_TEXT,
                          content: '---------$qty Item(s)----------',
                          weight: 1, align: LineText.ALIGN_CENTER,linefeed: 1));
                    list.add(LineText(type: LineText.TYPE_TEXT,
                       content: 'TOTAL: ${totalAmount.toStringAsFixed(2)}',
                       weight: 2, width: 2, height: 1, align: LineText.ALIGN_CENTER,
                       linefeed: 1,),);
                    list.add(LineText(type: LineText.TYPE_TEXT,
                          content: '*****************************',
                          weight: 1, align: LineText.ALIGN_CENTER,linefeed: 1));
                    list.add(LineText(type: LineText.TYPE_TEXT,
                       content: "Acknowledgement receipt",weight: 2,width: 2,
                       height: 1,align: LineText.ALIGN_CENTER, linefeed: 1,),);
                    list.add(LineText(type: LineText.TYPE_TEXT,
                       content: "See you next purchase!", weight: 2, width: 2,
                       height: 1, align: LineText.ALIGN_CENTER, linefeed: 1,),);
                    list.add(LineText(type: LineText.TYPE_TEXT,
                       content: 'Made by Palaris POS', weight: 2, width: 2,
                       height: 2, align: LineText.ALIGN_CENTER, linefeed: 1,),);
                    list.add(LineText(type: LineText.TYPE_TEXT,
                       content: "          ",weight: 2, width: 2, height: 2,
                       align: LineText.ALIGN_CENTER, linefeed: 1,), );
                    await bluetoothPrint.printLabel(config, list); }}:null),],),),
        SizedBox(height: 20,),
        Container(margin: EdgeInsets.only(left: 10,right: 10),
          padding: EdgeInsets.only(left: 5,right: 5), height: 100,
          decoration: BoxDecoration(border: Border.all(width: 5,
              color: Color(0xffffc000),),
            borderRadius: BorderRadius.all(Radius.circular(5)),),
          child:Column(children: [
              Center(child: Text('Sunmi Printer', style: TextStyle(fontSize: 25),),),
              OutlinedButton(onPressed: () async {
                  try {await sunmiReceipt.printReceipt();
                    ScaffoldMessenger.of(context).showSnackBar(
                      SnackBar(content: Text('Receipt printed successfully'),),);
                  } catch (error) {
                    ScaffoldMessenger.of(context).showSnackBar(
                      SnackBar(content: Text('Error printing receipt'),),); }},
                child: const Text('Print'),),],),),
        Divider(),
        Container(padding: const EdgeInsets.only(left:25,right:25),
          child: ElevatedButton(onPressed: (){Get.back();},
            child: Text('DONE', style: TextStyle(color: Colors.white, fontSize: 25),),),),],
    ),),); }}
```

pos_finish.dart

```
class PosFinish extends StatelessWidget{
  const PosFinish({super.key});

  @override
  Widget build(BuildContext context){
    return Scaffold(

      body:SingleChildScrollView(child: Column(
          children: [SizedBox(height: 160,),
            Container(padding: const EdgeInsets.only(left:25,right:25),
              child: Icon( Icons.check,size: 150,color: Colors.black,),
              alignment: Alignment.center, ),
            SizedBox(height: 20,),
            Container(padding: const EdgeInsets.only(left:15,right:15),
              child:Text('SUCCESSFUL TRANSACTION!',
                overflow: TextOverflow.visible,
                style: TextStyle(fontSize: 40, color: Colors.black,
                    fontWeight: FontWeight.bold), textAlign: TextAlign.center,),
              alignment: Alignment.center,),
            SizedBox(height: 40,),
            Row(mainAxisSize: MainAxisSize.max,
              mainAxisAlignment: MainAxisAlignment.spaceEvenly,
              children: [Container(padding: const EdgeInsets.only(left:25,right:25),
                  child: ElevatedButton(onPressed: ()=> Homepage()));},
                    child: Text('HOME', style: TextStyle(color: Colors.white, fontSize: 25),), ),),
                SizedBox(width: 10,),
                Container(padding: const EdgeInsets.only(left:25,right:25),
                  child: ElevatedButton(onPressed: (_)=> Pos()));},
                    child: Text('POS', style: TextStyle(color: Colors.white, fontSize: 25),),),
                ),],),],),),); }}
```

Backend

user_model.dart

```
class UserModel {
  final String storeName;
  final String firstName;
  final String lastName;
  final String emailAddress;
  final String password;
  final String userUid;

  UserModel({
    required this.storeName,
    required this.firstName,
    required this.lastName,
    required this.emailAddress,
    required this.password,
    required this.userUid,
  });

  factory UserModel.fromDocument(DocumentSnapshot doc) {
    return UserModel(
      storeName: doc['storeName'],
      firstName: doc['firstName'],
      lastName: doc['lastName'],
      emailAddress: doc['emailAddress'],
      password: doc['password'],
      userUid: doc['userUid'],
    );
  }

  Future<String?> getUser(String email) async {
    try {
      CollectionReference users =
      FirebaseFirestore.instance.collection('users');
      final snapshot = await users.doc(email).get();
      final data = snapshot.data() as Map<String, dynamic>;
      return data['firstName'];
    } catch (e) {
      return 'Error fetching user';
    }
  }
}
```

product_model.dart

```
class ProductModel {
  ProductModel({
    this.productUnit,
    this.productCode,
    this.productCategory,
    this.productName,
    this.productCost,
    this.productPrice,
    this.productStocks,
    this.productLowQuantity,
    this.productImage,
    this.timeStamp,
    this.productQuantity,
    this.soldCount,
    this.expDate,
  });
  final String? productUnit;
  final String? productCode;
  final String? productCategory;
  final String? productName;
  final double? productCost;
  final double? productPrice;
  final int? productStocks;
  final int? productLowQuantity;
  final String? productImage;
  final Timestamp? timeStamp;
  final int? productQuantity;
  final int? soldCount;
  final DateTime? expDate;

  ProductModel.fromJson(Map<String, Object> json)
      : this(
    productUnit: json['productUnit'] as String,
    productCode: json['productCode'] as String,
    productCategory: json['productCategory'] as String,
    productName: json['productName'] as String,
    productCost: json['productCost'] as double,
    productPrice: json['productPrice'] as double,
    productStocks: json['productStocks'] as int,
    productLowQuantity: json['productLowQuantity'] as int,
    productImage: json['productImage'] as String,
    timeStamp: json['timeStamp'] as Timestamp,
    productQuantity: json['productQuantity'] as int,
    soldCount: json['soldCount'] as int,
    expDate: json['expDate'] as DateTime,
  );

  Map<String, Object> toJson() {
    return {
      'productUnit' : productUnit as String,
      'productCode': productCode as String,
      'productCategory': productCategory as String,
      'productName': productName as String,
      'productCost': productCost as double,
      'productPrice': productPrice as double,
      'productStocks': productStocks as int,
      'productLowQuantity': productLowQuantity as int,
      'productImage': productImage as String,
      'productQuantity': productQuantity as int,
      'soldCount': soldCount as int,
      'expDate': expDate as DateTime,
    };
  }
}
```

cart_model.dart

```
class CartModel {
  CartModel({
    required this.id,
    required this.productId,
    required this.productCode,
    required this.productCategory,
    required this.productName,
    required this.productInitialPrice,
    required this.productPrice,
    required this.productStocks,
    required this.productQuantity,
    required this.unitTag,
    required this.productImage,
    required this.totalCount,
    required this.totalPrice,
  });
  String id;
  String productId;
  String productCode;
  String productCategory;
  String productName;
  double productInitialPrice;
  double productPrice;
  int productStocks;
  int productQuantity;
  String unitTag;
  String productImage;

  double totalPrice;
  int totalCount;
  CartModel.fromJson(Map<String, Object> json)
      : this(
    id: json['id'] as String,
    productCode: json['productCode'] as String,
    productId: json['productId'] as String,
    productName: json['productName'] as String,
    productCategory: json['productCategory'] as String,
    productInitialPrice: json['productInitialPrice'] as double,
    productPrice: json['productPrice'] as double,
    productQuantity: json['productQuantity'] as int,
    productStocks: json['productStocks'] as int,
    unitTag: json['unitTag'] as String,
    productImage: json['productImage'] as String,
    totalPrice: json['totalPrice'] as double,
    totalCount: json['totalCount'] as int,
  );

  Map<String, Object> toJson() {
    return {
      'productCode': productCode,
      'productCategory': productCategory,
      'productName': productName,
      'productInitialPrice': productInitialPrice,
      'productPrice': productPrice,
      'productStocks': productStocks,
      'productQuantity': productQuantity,
      'unitTag': unitTag,
      'productImage': productImage,
      'totalPrice': totalPrice,
      'totalCount': totalCount,
    };
  }
}
```

cart_provider.dart

```
class CartProvider with ChangeNotifier {
  CartServices _cart = CartServices();
  double subTotal = 0.0;
  double cartTotalCost = 0.0; // Added variable
  int cartQty = 0;
  int itemQty = 0;
  late DateTime timestamp;
  late QuerySnapshot snapshot;
  List cartList = [];
  Future<double?> getCartTotal() async {
    var cartTotal = 0.0;
    var cartTotalCost = 0.0; // Added variable
    List newList = [];
    QuerySnapshot snapshot =
    await _cart.cart.doc(_cart.user!.uid).collection('products').get();

    snapshot.docs.forEach((doc) {
      if (!newList.contains(doc.data())) {
        newList.add(doc.data());
        this.cartList = newList;
        notifyListeners();
      }
      cartTotal = cartTotal + doc['productTotal'];
      cartTotalCost = cartTotalCost + doc['productTotalCost']; // Calculate cartTotalCost
    });

    this.subTotal = cartTotal;
    this.cartTotalCost = cartTotalCost; // Assign calculated cartTotalCost
    this.cartQty = snapshot.docs.length;
    this.snapshot = snapshot;
    notifyListeners();

    return cartTotal;
  }
}
```

functions.dart

```
// initialize Firebase Firestore and Authentication
  final FirebaseFirestore _firestore = FirebaseFirestore.instance;
  User? user = FirebaseAuth.instance.currentUser;
  FirebaseAuth auth = FirebaseAuth.instance;
// function to save data to Firestore
  Future<void> saveDataToFirestore() async {
    String storeNameText = storeName.text; String addressText = address.text; String
tinText = tin.text;
    try {await _firestore.collection('users').doc(user!.uid).update({
        'storeName': storeNameText, 'storeAddress': addressText, 'storeTin': tinText,});
      print('Data saved to Firestore');
    } catch (e) { print('Error saving data to Firestore: $e');} }
// function to get user data
  Future getCurrentUserDataFunction() async {
    await _firestore.collection("users").doc(user!.uid).get().then(
          (DocumentSnapshot documentSnapshot) {if (documentSnapshot.exists) {
          userModel = UserModel.fromDocument(documentSnapshot);
        } else {print("Document does not exist in the databases }
// function to handle logout
  void handleLogout() async {
  FlutterSecureStorage storage = FlutterSecureStorage();
    try {await auth.signOut();
      print('User logged out successfully.');
      DefaultCacheManager().emptyCache();
      await storage.deleteAll();
      clearAndRefreshData().then((_) {
        Navigator.pushAndRemoveUntil(context, MaterialPageRoute(builder: (context) =>
Home()),
              (Route<dynamic> route) => false,);});
    } catch (e) {print('Error logging out: $e');}
  }
// function to save image to Firebase Storage
  File? _photo;
  Future imgFromGallery() async {
    ImagePicker imagePicker = ImagePicker();
    XFile? file = await imagePicker.pickImage(source: ImageSource.gallery);
    List<int> compressedImage = (await FlutterImageCompress.compressWithFile(
      file!.path, quality: 5, // Set the desired image quality (0 to 100)
    )) as List<int>;
    Uint8List compressedImageData = Uint8List.fromList(compressedImage);
    String uniqueFileName = DateTime.now().microsecondsSinceEpoch.toString();

    Reference referenceRoot = FirebaseStorage.instance.ref();
    Reference referenceDirImages = referenceRoot.child('${user!.uid}/images');
    Reference referenceImageToUpload = referenceDirImages.child(uniqueFileName);
    try {await referenceImageToUpload.putData(compressedImageData);
      _productImage = await referenceImageToUpload.getDownloadURL();
    } catch (error) {}setState(() {if (file != null) {_photo = File(file.path);
      } else {print('No image selected.');}});
  }
  Future imgFromCamera() async {
    ImagePicker imagePicker = ImagePicker();
    XFile? file = await imagePicker.pickImage(source: ImageSource.camera);
    print('$file?.path');
    List<int> compressedImage = (await FlutterImageCompress.compressWithFile(
      file!.path, quality: 5, // Set the desired image quality (0 to 100)
    )) as List<int>;
    Uint8List compressedImageData = Uint8List.fromList(compressedImage);
    File tempFile = await
File('${file.path}_compressed').writeAsBytes(compressedImageData);
    String uniqueFileName= DateTime.now().microsecondsSinceEpoch.toString();
    Reference referenceRoot = FirebaseStorage.instance.ref();
    Reference referenceDirImages = referenceRoot.child('${user!.uid}/images');
    Reference referenceImageToUpload = referenceDirImages.child(uniqueFileName);
    try {await referenceImageToUpload.putFile(tempFile);
      _productImage = await referenceImageToUpload.getDownloadURL();
    } catch(error){}setState(() {if (file != null) {_photo = File(file.path);
      } else {print('No image selected.');}});
  }
// function to scan barcode
  Future<void> scanProductBarcode() async {
    String barcodeScanRes;
    try {barcodeScanRes = await FlutterBarcodeScanner.scanBarcode(
          '#ff6666', 'Cancel', true, ScanMode.DEFAULT);
    } on PlatformException {barcodeScanRes = 'Failed to get platform version.';}
    if (!mounted) return;
    setState(() {_productCode = barcodeScanRes;});
  }
// function to connect thermal printer
  BluetoothPrint bluetoothPrint = BluetoothPrint.instance;
  bool _connected = false;
  BluetoothDevice? _device;
  String tips = 'No Device Connect';
  List<BluetoothDevice> _devices = [];
  String _deviceMsg ="";
  final f = NumberFormat("\$#####.00","en_US");
// function to print barcode using thermal printer
  Future<void> initPrinter() async {
    bluetoothPrint.startScan(timeout: Duration(seconds: 2));
    bool isConnected = await bluetoothPrint.isConnected??false;
    bluetoothPrint.state.listen((state){
      print('****************** cur device status: $state');
      switch (state){case BluetoothPrint.CONNECTED:
          setState(() {_connected = true;
            tips = 'Connect Success';}); break;
        case BluetoothPrint.DISCONNECTED:
          setState(() {_connected = false;
            tips = 'Disconnect Success';}); break; default: break;}});
    if (!mounted) return;
    if (isConnected){setState(() {_connected=true;});}
  }
  String r = '';
  String randomNumber(){
    String r = Random().nextInt(99999999).toString().padLeft(9, '4');
    return r;
  }
// function to print barcode using handheld pos
  Future<void> printSunmiBarcode() async {
    await SunmiPrinter.bindingPrinter();
    await SunmiPrinter.initPrinter();
    await SunmiPrinter.setAlignment(SunmiPrintAlign.CENTER);
    await SunmiPrinter.startTransactionPrint(true);
    await SunmiPrinter.printText("${_productName.text}",
        style: SunmiStyle(bold: true, fontSize: SunmiFontSize.LG, align:
SunmiPrintAlign.CENTER,));
    await SunmiPrinter.setAlignment(SunmiPrintAlign.CENTER);
    await SunmiPrinter.printBarCode(_productCode, barcodeType:
SunmiBarcodeType.CODE128,
        textPosition: SunmiBarcodeTextPos.TEXT_UNDER, height: 80, width: 500);
    await SunmiPrinter.lineWrap(2);
    await SunmiPrinter.lineWrap(2);
    await SunmiPrinter.exitTransactionPrint(true);
  }
// function to check and prevent double entry of items
  Future<bool> checkIfProductCodeExists(String productCode) async {
    User? user = FirebaseAuth.instance.currentUser;
    final QuerySnapshot<Map<String, dynamic>>
        snapshot = await _firestore.collection('users').doc(user!.uid).collection('products')
            .where('productCode', isEqualTo: productCode).get();
    return snapshot.docs.isNotEmpty;}
// function to fetch demand forecast
  Future<void> fetchData() async {
    final docSnapshot = await _firestore.collection('predictions').doc(user!.uid)
        .collection('demand').doc('data').get();
    final predictions = docSnapshot['predictions'];
    Set<String> categories = {};
    for (var prediction in predictions) {
      for (var field in prediction.keys) {
        if (field != 'date') {final category = field; categories.add(category); }}}
    setState(() {chartData = [];
      for (var prediction in predictions) {final DateTime date = (prediction['date'] as
Timestamp).toDate();
        for (var field in prediction.keys) {if (field != 'date') {final category = field;
            final rawPercentage = double.parse(prediction[field].toString());
            final percentage = rawPercentage * 100;
            chartData.add(ChartData(category: category, percentage: percentage, date:
date));}}}
      distinctCategories = categories.toList();});}
// function to delete the product in database
  Future<void> deleteProduct(String docId) async {
    try {final documentReference = FirebaseFirestore.instance.collection('users')
          .doc(user!.uid).collection('products').doc(docId);
      await documentReference.delete();
      Navigator.of(context).pop();} catch (e) {print(e.toString());}}
// function to update data in database
```

```
  Future<void> updateData(String docId) async {
    try {final documentReference =
_firestore.instance.collection('users').doc(user!.uid).collection('products').doc(_docId);
      await documentReference.get().then((documentSnapshot) {
        if (documentSnapshot.exists) {
          final data = documentSnapshot.data();
          _productName.text = data!['productName'];
          _productPrice.text = data['productPrice'].toString();
          _productCost.text = data['productCost'].toString();
          _productStocks.text = data['productStocks'].toString();}});} catch (e) {
      print(e.toString());}}
// function to save data in database
  Future<void> saveData(String docId) async {
    try {final documentReference =
_firestore.instance.collection('users').doc(user!.uid).collection('products').doc(_docId);
      await documentReference.update({
        'productName': _productName.text,
        'productPrice': double.parse(_productPrice.text),
        'productCost': double.parse(_productCost.text),
        'productStocks': int.parse(_productStocks.text), });
      final updatedDocumentSnapshot = await documentReference.get();
      Navigator.pushReplacement ()=> ProductDetail(
            documentSnapshot: updatedDocumentSnapshot, productId: widget.docId,),),);
    } catch (e) {print(e.toString());}}
// function to update total sales
void updateTotalSales() async {
    DateTime startDate = DateTime(selectedDate.year, selectedDate.month,
selectedDate.day);
    DateTime endDate = startDate.add(Duration(days: 1));
    QuerySnapshot snapshot = await FirebaseFirestore.instance.collection('orders')
        .doc(user!.uid).collection('orders')
        .where('timestamp', isGreaterThanOrEqualTo: Timestamp.fromDate(startDate))
        .where('timestamp', isLessThan: Timestamp.fromDate(endDate)).get();
    double sales = 0.0; double cost = 0.0; double profit = 0.0;
    for (var doc in snapshot.docs) {sales += doc.get('sales').toDouble();
      cost += doc.get('cost').toDouble();profit = sales - cost; }
    setState(() {totalSales = sales; totalCost = cost; totalProfit = profit; });}
// function for fetching data of total sales, cost & profit
void fetchTotal() async {
    DocumentSnapshot snapshot = await FirebaseFirestore.instance.collection('users')
        .doc(user!.uid).collection('sales').doc('summary').get();
    if (snapshot.exists) {setState(() {totalQuantity = snapshot['totalQuantity'];
        totalSales = snapshot['totalSales']; totalCost = snapshot['totalCost'];
        totalProfit = snapshot['totalProfit']; });
      print('Total Quantity: $totalQuantity'); print('Total Sales: $totalSales');
      print('Total Cost: $totalCost'); print('Total Profit: $totalProfit');
```

```
} else {print('Document does not exist'); }}
// function for downloading csv file of sales report
void downloadSummaryReports() async {
    List<List<dynamic>> csvData = [
      ['Total Quantity', 'Total Sales', 'Total Cost', 'Total Profit'],
      [totalQuantity, totalSales, totalCost, totalProfit],];
    String csvString = const ListToCsvConverter().convert(csvData);
    Directory? externalStorageDir = await getExternalStorageDirectory();
    if (externalStorageDir != null) {
      String downloadsPath = '${externalStorageDir.parent.path}/Download';
      Directory downloadsDir = Directory(downloadsPath);
      if (!downloadsDir.existsSync()) {downloadsDir.createSync(recursive: true); }
      String filePath = '$downloadsPath/summary_reports.csv';
      File file = File(filePath);
      await file.writeAsString(csvString);
      final scaffoldKey = ScaffoldMessenger.of(context);
      scaffoldKey.showSnackBar(
        SnackBar(content: Text('Downloaded successfully.'),
          backgroundColor: Color(0xffb7410e),),);
      // Show a toast or message to indicate the file is saved
      print('CSV file saved at: $filePath');
    } else {print('Unable to save the file'); }}
// functions for pos scanner
Future<void> scanProductBarcode() async {String barcodeScanRes;
    try {barcodeScanRes = await FlutterBarcodeScanner.scanBarcode(
        '#ff6666', 'Cancel', true, ScanMode.DEFAULT,);
      setState(() {scannedProductCode = barcodeScanRes; });
      handleScannedProductCode();
    } on PlatformException {barcodeScanRes = 'Failed to get platform version.'; }}
// function for scanned product code
  Future<void> handleScannedProductCode() async {
    if (scannedProductCode.isNotEmpty && scannedProductCode != '-1') {
      await compareProductCode();
    } else {Navigator.pop(context); }}
  Future<void> compareProductCode() async {setState(() {isLoading = true; });
    try {
      // Query the product database using the scanned product code
      var snapshot = await _product.product.doc(user!.uid).collection('products')
          .where('productCode', isEqualTo: scannedProductCode).get();
      if (snapshot.docs.isNotEmpty) {
        var productData = snapshot.docs[0].data();
        productCode = productData['productCode'];
        productCategory = productData['productCategory'];
        productName = productData['productName'];
        productPrice = productData['productPrice'];
        productQuantity = productData['productQuantity'];
        productCost = productData['productCost'];
        productStocks = productData['productStocks'];
        // Add more fields as per your database structure

        if (productStocks > 0 && productQuantity < productStocks) {
          await addToCart();
        } else {showDialog(context: context,
            builder: (BuildContext context) {return AlertDialog(title: Text('Out of Stock'),
                content: Text('The product is out of stock or the stock limit has been reached.'),
                actions: [TextButton(child: Text('OK'),
                    onPressed: () {Navigator.of(context).pop();},),],);},);}
      } else {}} catch (e) { print('Error: $e'); }
    setState(() {isLoading = false; });}
// function to add item in cart
  Future<void> addToCart() async {
    try { var cartItem = await _cart.cart.doc(user!.uid).collection('products').doc(
          scannedProductCode).get();
      if (cartItem.exists) {
        int currentQuantity = cartItem['productQuantity'];
        currentQuantity++;
        double currentPrice = productPrice * currentQuantity;
        double currentCost = productCost * currentQuantity;
        await cartItem.reference.update({
          'productQuantity': currentQuantity,
          'productTotal': currentPrice,
          'productTotalCost': currentCost,
        });
      } else {
        double currentPrice = productPrice;
        double currentCost = productCost;
        await _cart.cart.doc(user!.uid).collection('products').doc(
            scannedProductCode).set({
          'productCode' : productCode,
          'productCategory' : productCategory,
          'productName': productName,
          'productPrice': productPrice,
          'productQuantity': 1,
          'productTotal': currentPrice,
          'productTotalCost': currentCost, });}
      print('Product added to cart successfully!');
    } catch (e) { print('Error: $e'); }}
```

Model

```
import pickle
import os
import numpy as np
import pandas as pd
import firebase_admin
import matplotlib.pyplot as plt
import seaborn as sns
import calendar

from datetime import datetime
from pmdarima.arima import auto_arima
from firebase_admin import credentials
from firebase_admin import firestore

def pushSales(user):
    # Initialize the Firebase Admin SDK with credentials
    if not firebase_admin._apps:
        cred = credentials.Certificate('./files/jsonCred.json')
        firebase_admin.initialize_app(cred)

    # Create a Firestore client
    db = firestore.client()
    # Loading the prediction data
    data = pd.read_csv(f"./{user}/forecast.csv")
    # Convert data to a dict list with timestamps
    from google.cloud.firestore_v1 import SERVER_TIMESTAMP
    predictions = [{"date": pd.Timestamp(row["date"]).to_pydatetime(),
            "prediction": row["sales"]}
        for index, row in data.iterrows()]
    # Write the data to the Firestore collection
    doc_ref =
db.collection("predictions").document(user).collection("sales").document("data")
    doc_ref.set({"predictions": predictions})
    return print(f"Predictions uploaded to user {user}.")

def pushDemand(user):
    # Initialize the Firebase Admin SDK with credentials
    if not firebase_admin._apps:
        cred = credentials.Certificate('./files/jsonCred.json')
        firebase_admin.initialize_app(cred)
    # Create a Firestore client
    db = firestore.client()
    # Loading the prediction data
    data = pd.read_csv(f"./{user}/demand.csv")
   # Convert data to a dictionary list with timestamps
    from google.cloud.firestore_v1 import SERVER_TIMESTAMP
    predictions = []
    for index, row in data.iterrows():
        prediction = {"date": pd.Timestamp(row["date"]).to_pydatetime()}
        for column in data.columns:
            if column != "date":
                prediction[column] = row[column]
        predictions.append(prediction)
    # Write the data to the Firestore collection
    doc_ref =
db.collection("predictions").document(user).collection("demand").document("data")
    doc_ref.set({"predictions": predictions})
    return print(f"Predictions uploaded to user {user}.")

def pullRaw(user):
    cred = credentials.Certificate('./files/jsonCred.json')
    firebase_admin.initialize_app(cred)
    db = firestore.client()
    orders_ref = db.collection('orders').document(user).collection('orders')
    # Retrieve all documents in the collection
    orders = orders_ref.get()
    # Store document data in a list
    documents = []
    for order in orders:
        document_data = order.to_dict()
        documents.append(document_data)
    # Convert the list of document data to a DataFrame
    df = pd.DataFrame(documents)
    # Saving the df to a local file
    df.to_csv(f"./{user}/rawData.csv", index=False)
    # Message Prompt
    return print(f"Data ingestion from user {user} success")

def processSales(user):
    # reading the data extracted from the firebase
    df = pd.read_csv(f"./{user}/rawData.csv")
    ds_path = f"./{user}/dataset.csv"
    if not os.path.exists(ds_path):
        # Create an empty DataFrame with the required columns
        ds_columns = ['date', 'sales', 'area_change']
        ds = pd.DataFrame(columns=ds_columns)
        ds.to_csv(ds_path, index=False)
    # reading the current dataset
    ds = pd.read_csv(ds_path)
    # timestamp conversion and index setting
    df['timestamp'] = pd.to_datetime(df['timestamp'])
    df.set_index('timestamp', inplace=True)
    ds['date'] = pd.to_datetime(ds['date'])
    # aggregation of to daily sales
    df  = df.resample('D').sum()
    df.reset_index(inplace=True)
    # strftime removal
    df['timestamp'] = df['timestamp'].dt.strftime('%Y-%m-%d')
    df.rename(columns = {'timestamp':'date'}, inplace = True)
    # dropping uneccessary columns
    df.drop(columns=['orderNo', 'totalQuantity', 'cost'], inplace=True)
    # add area change column
    df['area_change'] = 1
    # combi_ds
    combi_ds = pd.concat([df, ds]).drop_duplicates()
    combi_ds['date'] = pd.to_datetime(combi_ds['date'])
    combi_ds.sort_values(by=['date'], inplace=True)
    combi_ds = combi_ds[~combi_ds.duplicated(subset='date')]
    # Convert 'date' column to datetime format
    combi_ds['date'] = pd.to_datetime(combi_ds['date'], format='%m/%d/%Y')
    # Set 'date' column as the index
    combi_ds.set_index('date', inplace=True)
    # Generate a complete date range with a frequency of 1 day
    complete_date_range = pd.date_range(start=combi_ds.index.min(),
end=combi_ds.index.max(), freq='D')
    # Reindex the data frame with the complete date range
    combi_ds = combi_ds.reindex(complete_date_range)
    # Reset the index
    combi_ds.reset_index(drop=False, inplace=True)
    combi_ds.rename(columns={'index': 'date'}, inplace=True)
    # fill missing values
    combi_ds['sales'].interpolate(method='linear', inplace=True)
    combi_ds['area_change'].fillna(1, inplace=True)
    # Saving the df to a local file
    combi_ds.to_csv(f"./{user}/dataset.csv", index=False)
    return print(f"Preprocess success.")

def checkUsers():
    # Initialize Firebase Admin SDK
    cred = credentials.Certificate('./jsonCred.json')  # Replace with the path to your service
account key JSON file
    firebase_admin.initialize_app(cred)
    db = firestore.client()
    # Retrieve documents from the users collection
    users_ref = db.collection('users')
    docs = users_ref.get()
    # Create local folders for each document and delete non-existing folders
    existing_folders = set()
    for doc in docs:
        # Create a folder with the document ID as the folder name
        folder_name = doc.id
        os.makedirs(folder_name, exist_ok=True)
        print(f"Created folder: {folder_name}")
        existing_folders.add(folder_name)
    # Delete non-existing folders
    local_folders = set(os.listdir('.'))
    folders_to_delete = local_folders - existing_folders
    for folder in folders_to_delete:
        if os.path.isdir(folder):
            os.rmdir(folder)
            print(f"Deleted folder: {folder}")

def reModel(user):
    # importing datasets
    df = pd.read_csv(f"./{user}/dataset.csv")
    df['date'] = pd.to_datetime(df['date'])
    train_df = df.copy()
    # fit an ARIMA model to the training data
    model = auto_arima(df['sales'], exog=['area_change'],
                       seasonal_test='ocsb', seasonal=True, m=12,
                       information_criterion='aic', stepwise=True, alpha=0.5,
                       trace=True, max_iter=100)
    model_fit = model.fit(train_df['sales'])
    # saving the model
    with open(f"./{user}/model.pkl", "wb") as f:
        pickle.dump(model_fit, f)
    # get model orders
    order = model_fit.order
    seasonal_order = model_fit.seasonal_order
    modelName = "SARIMA"
    # Open the text file for writing
    with open(f"./{user}/results.txt", 'w') as file:
        # Write the model name
        file.write("Model: {}\n".format(modelName))
        file.write("\n")
        # Write the model orders
        file.write("Model Orders:\n")
        file.write("p = {}\n".format(order[0]))
        file.write("d = {}\n".format(order[1]))
        file.write("q = {}\n".format(order[2]))
        file.write("\n")
        # Write the seasonal orders
        file.write("Seasonal Orders:\n")
        file.write("P = {}\n".format(seasonal_order[0]))
        file.write("D = {}\n".format(seasonal_order[1]))
        file.write("Q = {}\n".format(seasonal_order[2]))
        file.write("s = {}\n".format(seasonal_order[3]))
        file.write("\n")
    return "reModel Successful"

def modelPredict(user):
    with open(f"./{user}/model.pkl", 'rb') as file:
        model = pickle.load(file)
    current_date = datetime.now()
    days_in_month = calendar.monthrange(current_date.year, current_date.month)[1]
    time = 7
    pred = np.round(model.predict(n_periods=time),0)
    date_df = pd.read_csv(f"./{user}/dataset.csv")
    start_date = pd.to_datetime(date_df['date'].iloc[-1]) + pd.DateOffset(days=1)
    date_index = pd.date_range(start=start_date, periods= len(pred))
    forecast = pd.DataFrame({'date': date_index, 'sales':   pred})
    forecast.to_csv(f"./{user}/forecast.csv", index=False)
```

ANNEX III

USER MANUAL

PALARIS

Version 1.0
App Size: 26 MB
Supports Android 6.0 (Marshmallow) and above

User Manual Guide

Team **MicroPOS** is grateful for choosing the **Palaris** to help you manage your inventory. This user manual will help you to navigate yourself through the app. Carefully read each part.

Application Overview

The **Palaris** mobile application is a mobile point-of-sale (POS) system enabling retailers to manage their inventory with proper minimum requirements.

Palaris is developed by Electronics Engineering students from the Technological University of the Philippines – Manila. The **Palaris** mobile application aims to aid common problems such as inventory management, sales and expiration date tracking, and sales and demand forecasting for MSMEs, specifically the mini-grocery stores.

Organization of the Manual

The user manual is consists of the following:

INSTRUCTIONS

MINIMUM DEVICE REQUIREMENTS

PALARIS SPECIFICATIONS

1. Installation
2. Account Setup
3. Navigation
4. Features
   - Add Items
   - Barcode Generator
   - Products
   - Inventory
   - Hot Items
   - Transaction
   - Sales Report

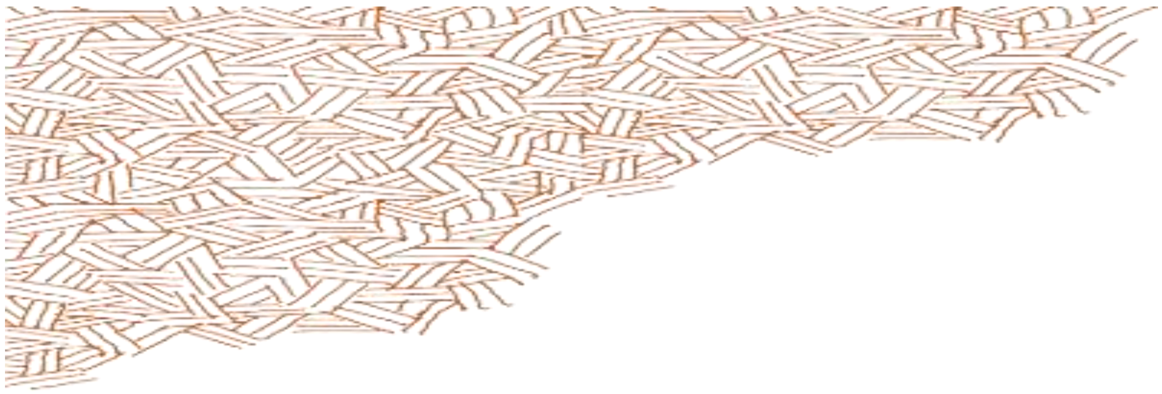
- Analytics
- Point-of-Sale or POS

MINIMUM DEVICE REQUIREMENTS

- Android 6.0 (Marshmallow) and above
- 2GB RAM
- 4GB Phone Storage
- Wi-Fi and Bluetooth connection ready
- At Least a 2MP camera
- With a sim slot (For data connectivity if Wi-Fi is unavailable)

PALARIS SPECIFICATIONS

1. Installation
    - Locate the **Palaris** mobile application on your device.
    - Select and install.
    - Once downloaded, launch the application by tapping on the icon
2. Account Setup
    - After the mobile application is launched, the Welcome page will be the first thing you will see.
    - You can log in to your account if you have an existing account.
    - If you do not have an account, tap the sign-up button. From there, the Sign-Up page will appear; enter the necessary details such as name, email address, password, and store name.
    - Tap on the box to agree with the Terms and Conditions
    - Upon setting up the account, you can now log in to your account and directly access all exclusive features available in the mobile application.
3. Navigation
    - The homepage shows key features such as products, inventory, hot items, transactions, sales reports, and analytics.
    - A menu drawer can be seen at the upper left part of the homepage, and this is where the basic features of mobile applications are located. Barcode Generator



and Devices (i.e., Thermal Printer and Sunmi Handheld POS) are also located in the menu drawer.

4. Features

- **Add Items -** this allows you to add the product you want, considering the item has a barcode. You can customize the details of your items, such as
  - ❖ Item Name
  - ❖ Stock Number (Low Stock and In Stock)
  - ❖ Quantity of Items
  - ❖ Add Product Image
  - ❖ Expiration Date
  - ❖ Category
  - ❖ Unit
  - ❖ Barcode Generator
  - ❖ Thermal Printer.

  This part already has categories for the items,
  - ❖ Baking & Cooking
  - ❖ Beverages
  - ❖ Bread & Breakfast
  - ❖ Canned Goods
  - ❖ Coffee, Tea & Cocoa
  - ❖ Dairy & Eggs
  - ❖ Household
  - ❖ Instant & Ready to Eat
  - ❖ Personal Care & Beauty
  - ❖ Rice, Pasta & Grains
  - ❖ Snacks & Sweets
  - ❖ Spreads, Jams, & Honey
  - ❖ Others
- **Barcode Generator** – allows you to create a unique barcode for the product, fill in the product name, and fill up all the necessary details.
- **Products –** this section allows users to view and edit all the added items.
- **Inventory –** the list of stocks available, items on low stock, and expiration date of the products.

- **Hot Items –** this section is where you can locate the most sellable items of the week.
- **Transaction** – this is where the summary of the total of the items sold daily, weekly, monthly, and yearly can be located. It is customizable and historically viewable.
- **Sales Report –** this graph the sold items per month. Here, you can see a graph if you are progressively selling or not.
- **Analytics –** this is where the analytics for Sales Forecast and Demand Forecast can be located. This predicts your total sales and the possible demand for items in the future.
- **POS –** is where a transaction happens wherein the user can add the items to the cart or scan the barcode. This works like a usual POS where checkout, payment process, and thermal print happen.

ANNEX IV

PROGRESS DOCUMENTATION

PROGRESS DOCUMENTATION

Project Study Group Consultation with Thesis Adviser, Engr, Romeo L. Jorda Jr.

DATA GATHERING

Gathering of Data from Various Stores

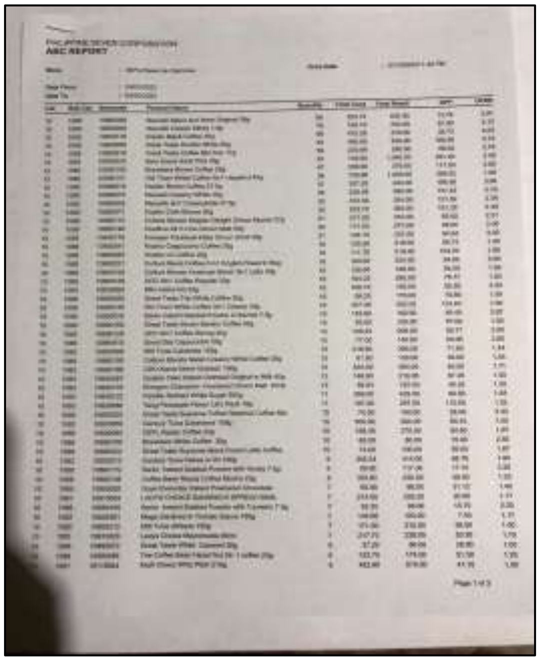
MINI GROCERY STORES VISITED

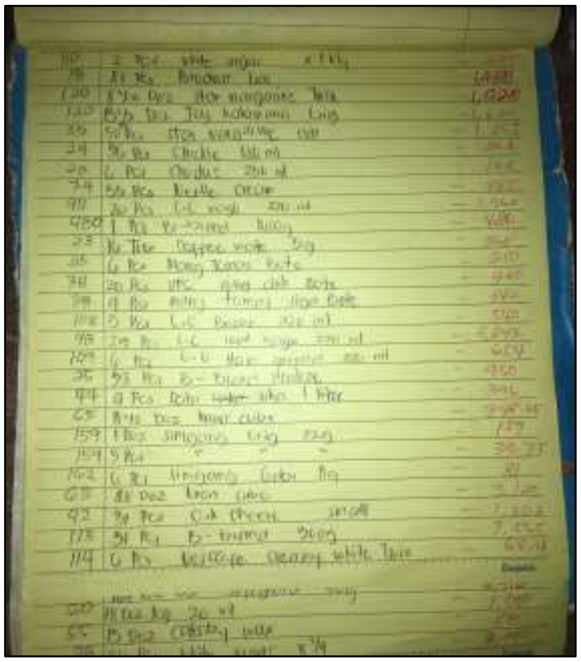
Physical Layout of the Visited Store for Data Gathering

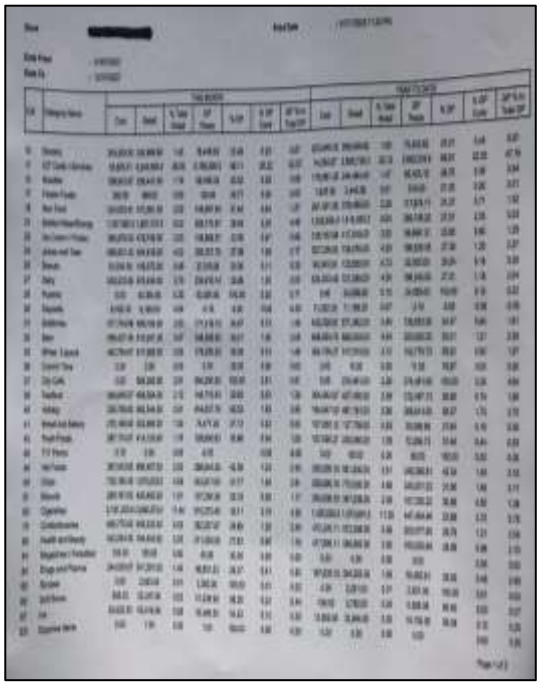
Physical Layout of the Visited Store for Data Gathering

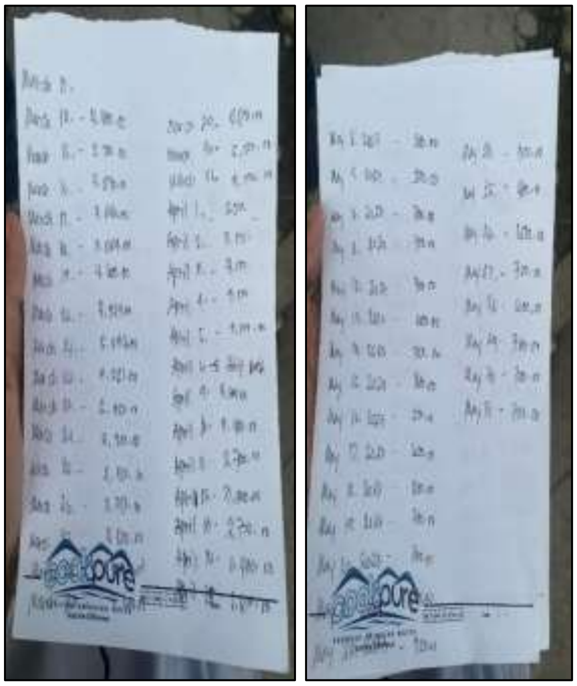
SAMPLE DATA GATHERED FROM VISITED STORES

Gathered Data of Various of Stores for Training of the Sales and Demand Forecasting



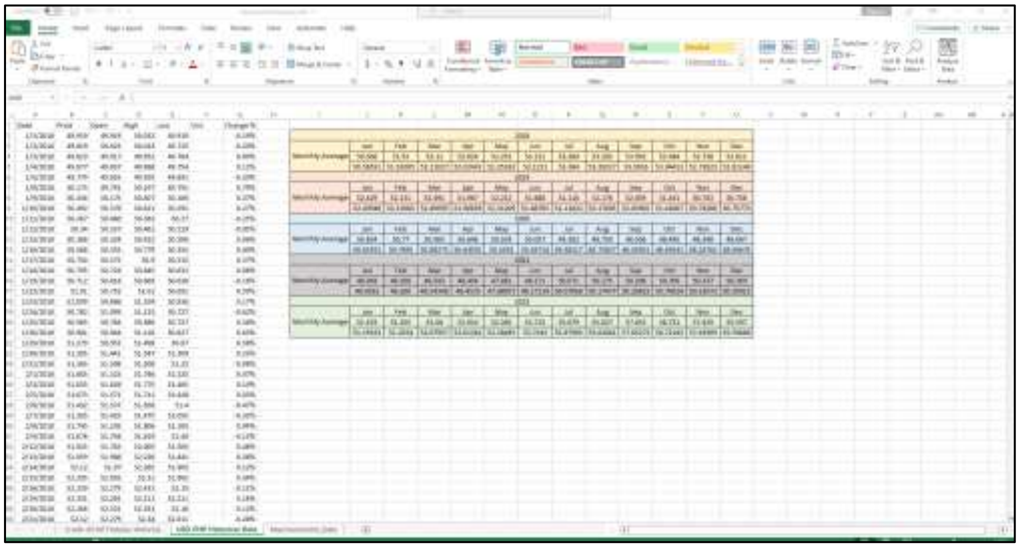




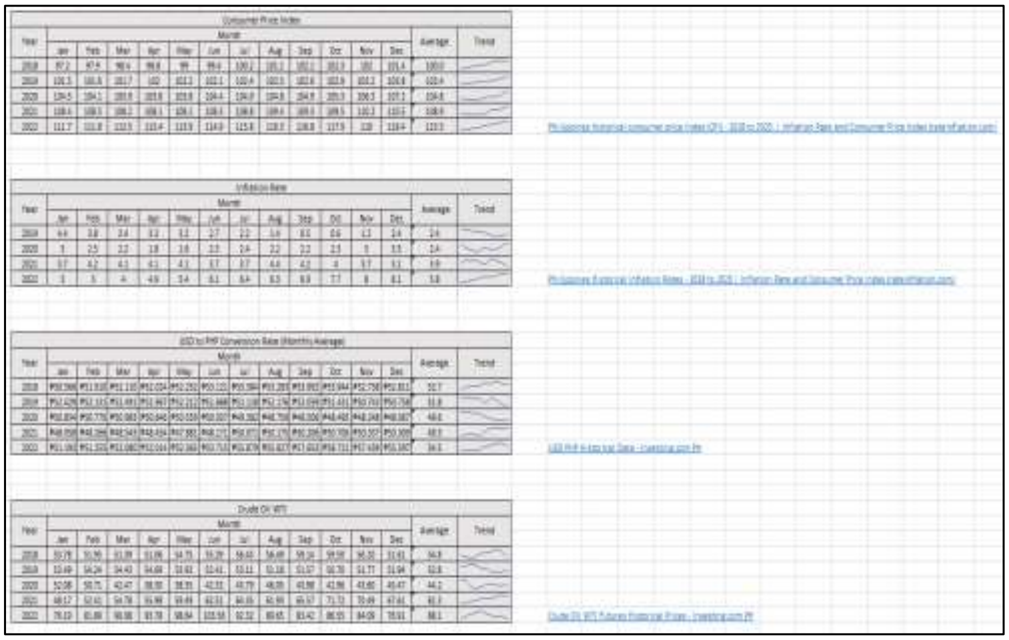


DATE GATHERED FOR TRAINING THE MODEL

Pre-processed Data for Sales and Demand Forecasting



Date	Price	Open	High	Low	Vol.	Change %
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]



Consumer Price Index														
Year	Month												Average	Trend
	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec		
2018	97.2	97.4	98.4	98.8	99	99.4	100.2	101.1	102.1	101.3	102	101.4	100.0	
2019	100.3	100.5	101.7	102	102.2	102.1	102.4	102.5	102.6	102.9	103.2	103.8	102.4	
2020	104.5	104.1	103.8	103.8	103.8	104.4	104.8	104.8	104.8	105.3	106.3	107.1	104.8	
2021	108.6	108.5	108.2	108.1	108.1	108.1	108.6	109.4	109.3	109.5	110.3	110.7	108.9	
2022	111.7	111.8	112.5	113.4	113.9	114.9	115.8	118.2	118.8	117.9	118	118.4	115.3	

Philippines Historical Consumer Price Index (CPI) - 2018 to 2022 \| Inflation Rate and Consumer Price Index (rate-inflation.com)

Inflation Rate														
Year	Month												Average	Trend
	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec		
2019	4.4	3.8	3.4	3.2	3.2	2.7	2.2	1.6	0.9	0.6	1.3	2.4	2.4	
2020	3	2.5	2.2	1.8	1.8	2.2	2.4	2.2	2.2	2.3	3	3.3	2.4	
2021	3.7	4.2	4.1	4.1	4.1	3.7	3.7	4.4	4.2	4	3.7	3.1	3.9	
2022	3	3	4	4.9	5.4	6.1	6.4	6.3	6.9	7.7	8	8.1	5.8	

Philippines Historical Inflation Rates - 2018 to 2022 \| Inflation Rate and Consumer Price Index (rate-inflation.com)

USD to PHP Conversion Rate (Monthly Average)														
Year	Month												Average	Trend
	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec		
2018	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	52.7	
2019	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	51.8	
2020	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	49.6	
2021	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	49.3	
2022	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	54.5	

USD PHP Historical Data - Investing.com PH

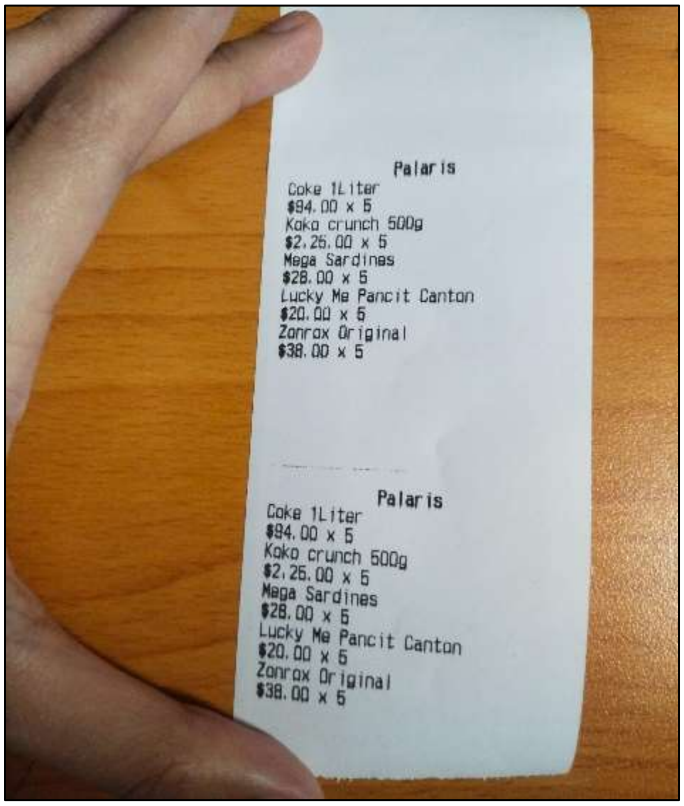
Crude Oil WTI														
Year	Month												Average	Trend
	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec		
2018	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	64.9	
2019	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	56.8	
2020	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	44.2	
2021	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	61.3	
2022	78.13	85.88	98.98	93.78	98.94	105.58	92.52	89.65	83.42	86.55	84.08	78.91	88.1	

Crude Oil WTI Futures Historical Prices - Investing.com PH

APPLIED CHANGES PROGRESS DEFENSE

Thermal Printer Procurement

Initial Layout for Customizable Receipt of Palaris

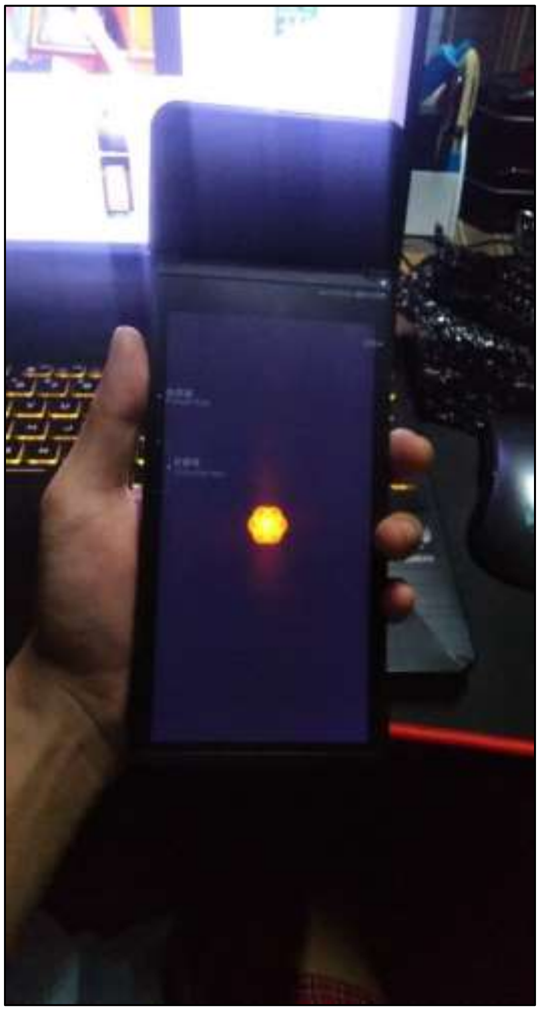


```
         Palaris
Coke 1Liter
$94.00 x 5
Koko crunch 500g
$2,25.00 x 5
Mega Sardines
$28.00 x 5
Lucky Me Pancit Canton
$20.00 x 5
Zonrox Original
$38.00 x 5
```

```
         Palaris
Coke 1Liter
$94.00 x 5
Koko crunch 500g
$2,25.00 x 5
Mega Sardines
$28.00 x 5
Lucky Me Pancit Canton
$20.00 x 5
Zonrox Original
$38.00 x 5
```

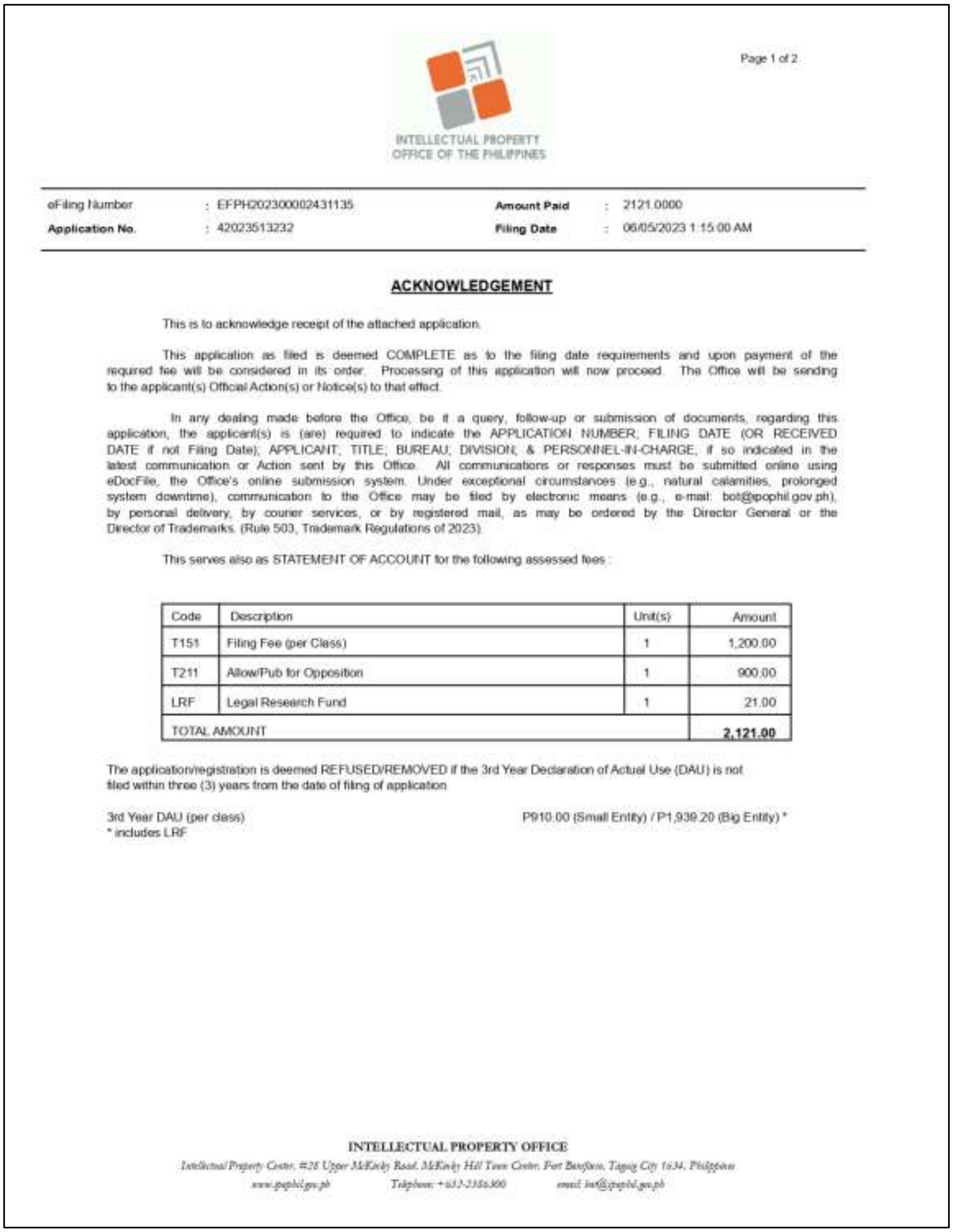
CHANGES DURING PRE-FINAL DEFENSE

Procurement of Handheld POS



FILING OF TRADEMARK TO ITSO

Acquired documents for filing trademark.



Page 1 of 2

INTELLECTUAL PROPERTY OFFICE OF THE PHILIPPINES

eFiling Number	: EFPH202300002431135	**Amount Paid**	: 2121.0000
Application No.	: 42023513232	**Filing Date**	: 06/05/2023 1:15:00 AM

ACKNOWLEDGEMENT

This is to acknowledge receipt of the attached application.

This application as filed is deemed COMPLETE as to the filing date requirements and upon payment of the required fee will be considered in its order. Processing of this application will now proceed. The Office will be sending to the applicant(s) Official Action(s) or Notice(s) to that effect.

In any dealing made before the Office, be it a query, follow-up or submission of documents, regarding this application, the applicant(s) is (are) required to indicate the APPLICATION NUMBER; FILING DATE (OR RECEIVED DATE if not Filing Date); APPLICANT; TITLE; BUREAU; DIVISION; & PERSONNEL-IN-CHARGE, if so indicated in the latest communication or Action sent by this Office. All communications or responses must be submitted online using eDocFile, the Office's online submission system. Under exceptional circumstances (e.g., natural calamities, prolonged system downtime), communication to the Office may be filed by electronic means (e.g., e-mail: bot@ipophil.gov.ph), by personal delivery, by courier services, or by registered mail, as may be ordered by the Director General or the Director of Trademarks. (Rule 503, Trademark Regulations of 2023)

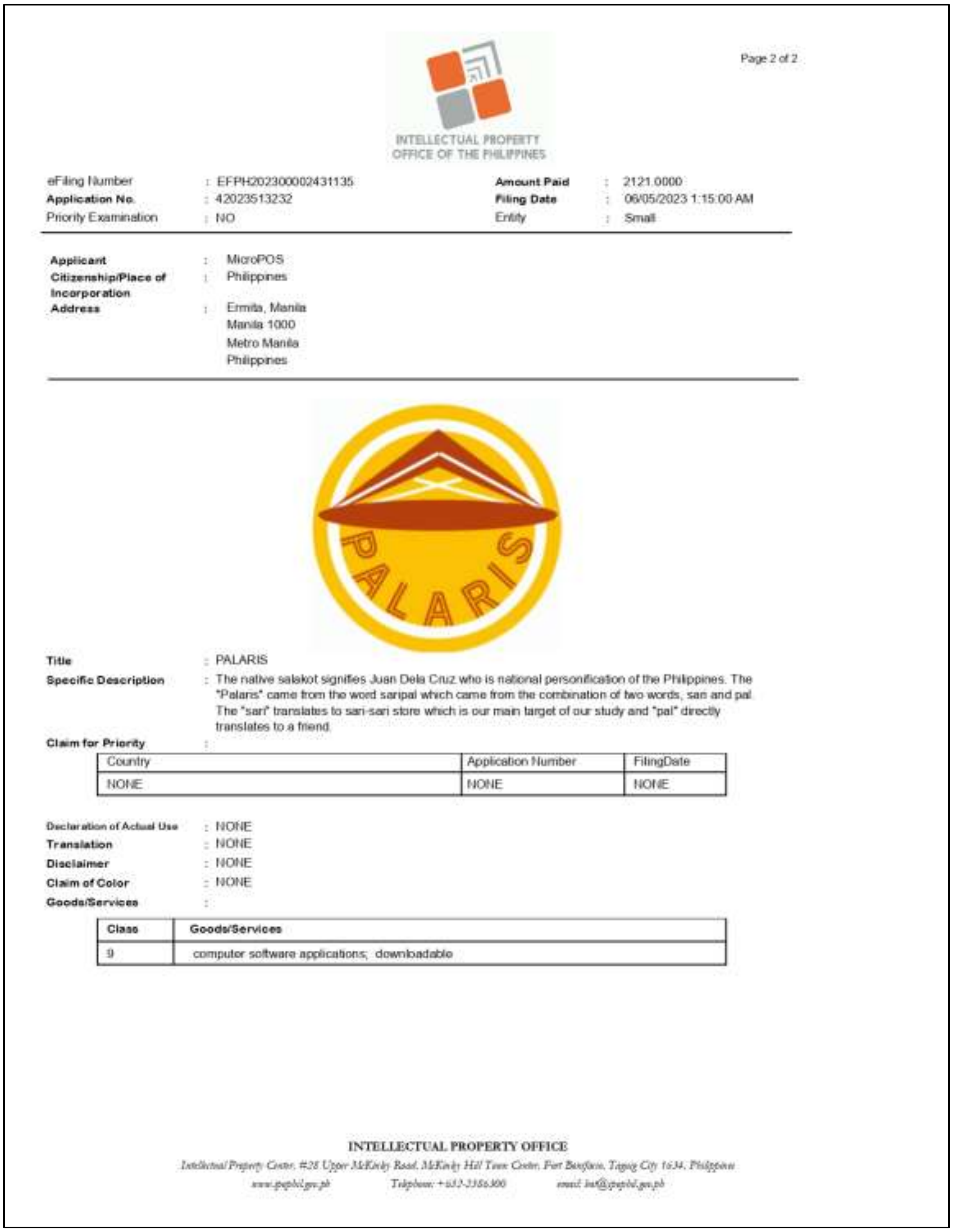
This serves also as STATEMENT OF ACCOUNT for the following assessed fees :

Code	Description	Unit(s)	Amount
T151	Filing Fee (per Class)	1	1,200.00
T211	Allow/Pub for Opposition	1	900.00
LRF	Legal Research Fund	1	21.00
TOTAL AMOUNT			**2,121.00**

The application/registration is deemed REFUSED/REMOVED if the 3rd Year Declaration of Actual Use (DAU) is not filed within three (3) years from the date of filing of application

3rd Year DAU (per class) P910.00 (Small Entity) / P1,939.20 (Big Entity) *
* includes LRF

INTELLECTUAL PROPERTY OFFICE
Intellectual Property Center, #28 Upper McKinley Road, McKinley Hill Town Center, Fort Bonifacio, Taguig City 1634, Philippines
www.ipophil.gov.ph *Telephone: +632-2386300* *email: bot@ipophil.gov.ph*



Page 2 of 2

INTELLECTUAL PROPERTY
OFFICE OF THE PHILIPPINES

eFiling Number	: EFPH202300002431135	**Amount Paid**	:	2121.0000
Application No.	: 42023513232	**Filing Date**	:	06/05/2023 1:15:00 AM
Priority Examination	: NO	Entity	:	Small

Applicant : MicroPOS
Citizenship/Place of Incorporation : Philippines
Address : Ermita, Manila
Manila 1000
Metro Manila
Philippines

PALARIS

Title : PALARIS

Specific Description : The native salakot signifies Juan Dela Cruz who is national personification of the Philippines. The "Palaris" came from the word saripal which came from the combination of two words, sari and pal. The "sari" translates to sari-sari store which is our main target of our study and "pal" directly translates to a friend.

Claim for Priority :

Country	Application Number	FilingDate
NONE	NONE	NONE

Declaration of Actual Use : NONE
Translation : NONE
Disclaimer : NONE
Claim of Color : NONE
Goods/Services :

Class	Goods/Services
9	computer software applications; downloadable

INTELLECTUAL PROPERTY OFFICE
Intellectual Property Center, #28 Upper McKinley Road, McKinley Hill Town Center, Fort Bonifacio, Taguig City 1634, Philippines
www.ipophil.gov.ph Telephone: +632-2386300 email: inf@ipophil.gov.ph

DEPLOYMENT SITE VISITATION

Actual Visitation of the Store for Deployment on March 27, 2023

ANNEX V

DEPLOYMENT

PALARIS MOBILE APPLICATION DEMONSTRATION

Demonstrating on How to Use the Palaris

Group Picture with Mrs. Jesmar Casiding, the co-owner of the Triple J Store on June 5, 2023

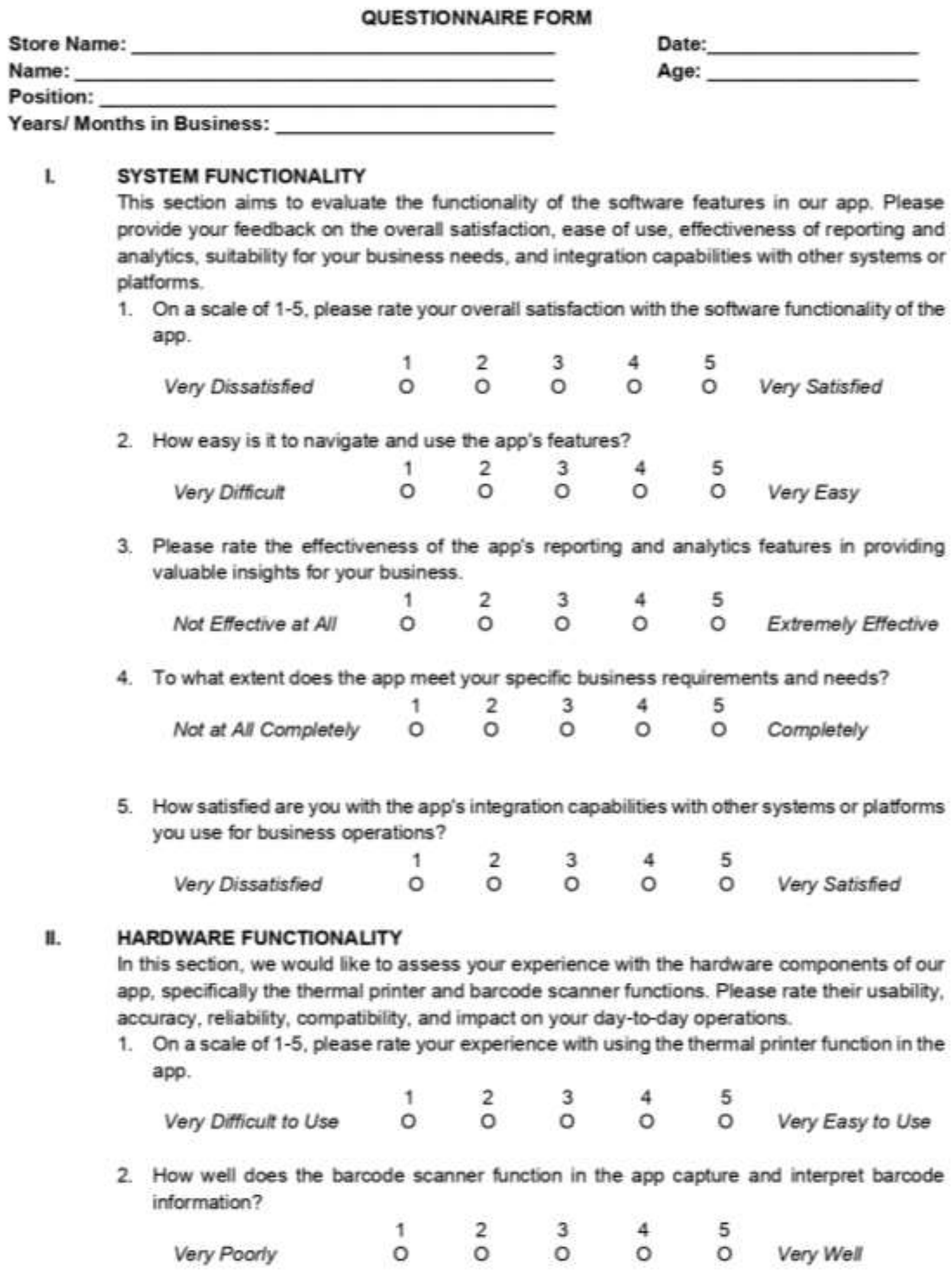
Group Picture with Mrs. Jesmar Casiding, the co-owner of the Triple J Store on June 13, 2023

FIELDWORK

Market Research for Pitching for Final Defense

ANNEX VI

SURVEY FORM



QUESTIONNAIRE FORM

Store Name: ______________________ **Date:** ______________

Name: ______________________ **Age:** ______________

Position: ______________________

Years/ Months in Business: ______________________

I. SYSTEM FUNCTIONALITY

This section aims to evaluate the functionality of the software features in our app. Please provide your feedback on the overall satisfaction, ease of use, effectiveness of reporting and analytics, suitability for your business needs, and integration capabilities with other systems or platforms.

1. On a scale of 1-5, please rate your overall satisfaction with the software functionality of the app.

	1	2	3	4	5	
Very Dissatisfied	O	O	O	O	O	*Very Satisfied*

2. How easy is it to navigate and use the app's features?

	1	2	3	4	5	
Very Difficult	O	O	O	O	O	*Very Easy*

3. Please rate the effectiveness of the app's reporting and analytics features in providing valuable insights for your business.

	1	2	3	4	5	
Not Effective at All	O	O	O	O	O	*Extremely Effective*

4. To what extent does the app meet your specific business requirements and needs?

	1	2	3	4	5	
Not at All Completely	O	O	O	O	O	*Completely*

5. How satisfied are you with the app's integration capabilities with other systems or platforms you use for business operations?

	1	2	3	4	5	
Very Dissatisfied	O	O	O	O	O	*Very Satisfied*

II. HARDWARE FUNCTIONALITY

In this section, we would like to assess your experience with the hardware components of our app, specifically the thermal printer and barcode scanner functions. Please rate their usability, accuracy, reliability, compatibility, and impact on your day-to-day operations.

1. On a scale of 1-5, please rate your experience with using the thermal printer function in the app.

	1	2	3	4	5	
Very Difficult to Use	O	O	O	O	O	*Very Easy to Use*

2. How well does the barcode scanner function in the app capture and interpret barcode information?

	1	2	3	4	5	
Very Poorly	O	O	O	O	O	*Very Well*

3. Please rate the accuracy and reliability of the barcode scanner function in the app.

	1	2	3	4	5	
Very Inaccurate and Unreliable	O	O	O	O	O	*Very Accurate and Reliable*

4. How satisfied are you with the compatibility and ease of integration with the hardware components (e.g., barcode scanner, thermal printer)?

	1	2	3	4	5	
Very Dissatisfied	O	O	O	O	O	Very Satisfied

5. To what extent does the hardware functionality contribute to a smooth and efficient workflow in your day-to-day operations?

	1	2	3	4	5	
Not at All Completely	O	O	O	O	O	*Very Easy*

III. **PREDICTIVE MODEL**

This section focuses on the sales forecasting model provided by our app. We are interested in understanding how often you base your business decisions on predictions, your perception of their accuracy, alignment with actual sales performance, impact on decision-making, and overall confidence in the model.

1. How often do you base your business decisions on the sales predictions provided by the app's predictive model?

	1	2	3	4	5	
Never	O	O	O	O	O	*Always*

2. On a scale of 1-5, how accurate do you find the sales predictions generated by the app's predictive model?

	1	2	3	4	5	
Very Inaccurate	O	O	O	O	O	*Very Accurate*

3. How well does the app's predictive model align with the actual sales performance of your business?

	1	2	3	4	5	
Very Poorly	O	O	O	O	O	*Very Well*

4. Please rate the impact of using the predictive model on your business decision-making process.

	1	2	3	4	5	
No Impact	O	O	O	O	O	*Very Significant Impact*

5. How confident are you in the reliability and usefulness of the app's sales forecasting model?

	1	2	3	4	5	
Not Confident at All	O	O	O	O	O	Very Confident

IV. COMMENTS AND SUGGESTIONS

Your input is valuable to us. In this section, we invite you to share any additional comments, suggestions, or specific feedback you may have regarding the app's features, functionality, or any other aspect. Feel free to provide detailed insights that can help us further improve the app to better serve your needs.

Please share any comments, suggestions, or specific feedback you have regarding the app's features, functionality, or any other aspect you would like to address.

ANNEX VII

PHOTOS FROM PREVIOUS DEFENSE

TOPIC DEFENSE

TITLE DEFENSE

PROGRESS DEFENSE

PRE-FINAL DEFENSE

FINAL DEFENSE

ANNEX VIII

PROPONENT'S INFORMATION

SEAMUS ROD C. AQUINO
Phase 3 Block 37 M Lot 13 Barangay 12, Dagat-dagatan Ave. Caloocan
0917-858-1597
seamussyyy@gmail.com

OBJECTIVE

Highly motivated electronics engineering student with a strong passion for data sciences and a drive to excel in the field of data engineering and machine learning. Seeking a challenging role as a data engineer or a machine learning engineer, where I can leverage my technical expertise, analytical mindset, and skills to develop, innovate solutions for real-world problems. Committed to continuous learning and improvement to stay updated with emerging technologies to contribute to the advancement of the field.

EDUCATIONAL ATTAINMENT

College
Bachelor of Science in Electronics Engineering
Technological University of the Philippines
Ayala Blvd., Ermita, Manila
(2018 - 2023)

SKILLS

- Programming
    - Python
    - MATLAB
    - R
    - SQL
- Data Analysis
- Machine Learning
- Time Series Modeling
- Algorithmic Thinking

DEVELOPMENT

- **Project SPARTA - Computing**
    - Computing in Python
    - Data Management Fundamentals
    - SQL for Business Users
    - Essential Excel Skills for Data Preparation and Analysis
    - Getting Grounded in Analytics
- **CISCO Networking Academy**
    - PCAP: Programming Essentials in Python
    - Networking Basics
    - Packet Tracer

EXPERIENCE

Chairperson – Web Application Committee
TUP-M APPRECIATE 2023
Ayala blvd. Ermita Manila
(Jan 2023 – May 2023)

Cadet Engineer
Dai-ichi Electronics Manufacturing
F. Mariano, Pasig City
(Aug 2022 to Oct 2022)

Manager – OECES Technical and Production Team
TUP-M ECE Dept
Ayala blvd. Ermita Manila
(Sep 2020 – Sep 2021)

TRICIER DANE T. ACIERTO, CCNA
Block 10 Lot 15 Phase 3 Orchids St., Mary Homes Subd., Molino 4 Bacoor, Cavite
0998-950-1007
aciertodane@gmail.com

OBJECTIVE
To secure a challenging position where I can effectively contribute my skills, certifications, and experiences as an Electronics Engineer.

EDUCATIONAL ATTAINMENT

College
Bachelor of Science in Electronics Engineering
Technological University of the Philippines
Ayala Blvd., Ermita, Manila
(2018 - 2023)

SKILLS
- Knowledgeable in Microsoft Excel, Word, and other MS Office Applications
- Knowledgeable in IP Planning and Design for Networks of IPv4 and IPv6
- Knowledgeable in TCP/IP and OSI model
- Knowledgeable in Splicing Fiber Optic Cable and Implementation
- Knowledgeable in configuring DHCPv4 and DHCPv6.
- Knowledgeable in configuring SCCP and SIP (VoIP) using Wired and Wireless
- Knowledgeable in configuring Cisco Access Point (AP)
- Knowledgeable in configuring Cisco 1800 and 2800 series routers on different protocols such as RIP, RIPng, OSPFv2, OSPFv3, EIGRP, EIGRP for IPv6, BGP, VPN, etc.
- Knowledgeable in configuring Cisco catalyst switches such as 2950, 2960, 3550, 3560 and 3750 with protocols such as VLAN, VTP, STP, RSTP, PVST, Rapid-PVST, ETHERCHANNEL, HSRP, Inter-VLAN Routing, and Port-Security

CERTIFICATIONS
- Fortinet NSE 3 Network Security Associate (May 2023 to May 2025)
- Fortinet NSE 2 Network Security Associate (March 2023 to March 2025)
- Cisco Certified Network Associate (CCNA) (February 2023 to February 2026)
- Fortinet NSE 1 Network Security Associate (September 2022 to September 2024)

WORK EXPERIENCE
Network Engineer
Amdocs
Pasig City
(May 2023 to present)

Intern
Converge Training and I.T. Services
San Roque, Marikina City
(August 2022 to March 2023)

REFERENCES
Engr. Mark Anthony V. Melendres
Chief Executive Officer (CEO)
MGKK Information Communication Technology Services
0977-196-8226

MARIAN GAIL C. DICHOSO
Block 2 Lot 6, Villa Laserna, Barangay Labas, Santa Rosa, Laguna, 4026
09062157097
dichosomariangail@gmail.com

OBJECTIVE
To join in an organization that will allow me to apply and contribute my knowledge as to develop my skills and abilities.

EDUCATIONAL ATTAINMENT

College
Bachelor of Science in Electronics Engineering
Technological University of the Philippines
Ayala Blvd., Ermita, Manila
(2018 - 2023)

SKILLS

- Basic understanding of MATLAB, Arduino, and certain programming languages such as Python and Dart. Knowledgeable in developing mobile application using Flutter platform. Proficiency in Microsoft Word, PowerPoint and Excel for reports and presentation.

CERTIFICATIONS

- Fortinet NSE 1 Network Security Associate (May 2023 to May 2025)
- Fortinet NSE 3 Network Security Associate (May 2023 to May 2025)
- Fortinet NSE 2 Network Security Associate (March 2023 to March 2025)
- Professional Protege: Terra Hertz Telecommunications Online Training Four-weeks (May 7-29,2022)
    - Mobile Wireless Spectrum in the Philippines
    - Wireless Network Fundamentals (2G up to 6G)
    - Spectrum Analyzer Fundamentals
    - Site Solution & Simulation Tool
    - Resume Creation
    - Interview Skills
    - Negotiation Skills
    - Financial Skills

- Master IP Addressing and Subnetting for CCN (May 2022)

WORK EXPERIENCE
Network Operations Center Intern
Lemcon Philippines Incorporated
#41 SFB 4 Laguna Technopark, Inc. Phase 8, Brgy. Loma, Binan City Laguna
August 2022

REFERENCES

ENGR. CHERRY G. PASCION
Assistant Professor 1
Technological University of the Philippines-Manila
Mobile:09055581972
Email:cherry_pascion@tup.edu.ph

JEE ANNE M. ORTEGA
Block 63 Lot 1 Brgy. Santo Cristo Dasmariñas City, Cavite
0976-272-7431
jeeanne.ortega@tup.edu.ph

OBJECTIVE

To secure a position in innovative projects and collaborative environments, leveraging my expertise in software development and electronics engineering. As a results-driven and detail-oriented student, I aim to contribute my passion for technology and problem-solving abilities to deliver efficient solutions.

EDUCATIONAL ATTAINMENT

College
Bachelor of Science in Electronics Engineering
Technological University of the Philippines
Ayala Blvd., Ermita, Manila
(2023)

SKILLS

- **Hard Skills**
  - Programming Skills
    - Intermediate level in Flutter, Dart, Java, Kotlin, Firebase
    - Familiar in Python, R, JavaScript, SQL, C#, HTML, CSS, SASS, Bootstrap
  - Editing and Design Skills
    - Proficient in Canva, and Adobe Photoshop
    - Familiar with Adobe XD, Figma, and Dreamweaver for designing user interfaces
  - Development Tools
    - Experience in Unity 3D for game development
  - Automation Skills
    - Zapier, Zoho CRM, Power Automate
- **Soft Skills**
  - Teamwork
  - Communication
  - Detail-Oriented
  - Leadership
  - Critical Thinking
  - Time Management

CERTIFICATIONS

- Master IP Addressing ad Subnetting for CCNA (May 2022)

WORK EXPERIENCE

Junior Developer Intern
Staff Domain, Inc.
Ortigas Center, Metro Manila
(August 2022 to October 2022)

REFERENCES
Aileen Mae D. So
Web Developer
Staff Domain, Inc.
0917-629-9466

RAIAH SHERINA L. VARGAS
Block 20, Lot 8 Phase 2, Casimiro Westville Homes, Brgy. Ligas III, Bacoor, Cavite
0969-142-8895
raiahsherina.vargas@gmail.com

OBJECTIVE
Seeking a career opportunity to expand my knowledge and to help me improve my skills in my chosen field that could contribute to my growth as a person and to the company I work with.

EDUCATIONAL ATTAINMENT

College
Bachelor of Science in Electronics Engineering
Technological University of the Philippines
Ayala Blvd., Ermita, Manila
(2018 - 2023)

SKILLS
- Basic Cisco Packet Tracer Knowledge
- Operation of Devices (Function Generator, Multi-testers, Oscilloscope
- Simulation in Multisim, Designspark PCB, TinkerCAD, GNS3. LTSspice XVII and Electronic Workbench
- Proficiency in Microsoft Programs (Excel, PowerPoint Presentation, Word)
- Knowledgeable in design and scale

CERTIFICATIONS
- Fortinet NSE 3 Network Security Associate (April 2023 to April 2025)
- Fortinet NSE 2 Network Security Associate (April 2023 to April 2025)
- Fortinet NSE 1 Network Security Associate (April 2023 to April 2025)
- Asia Open RAN Academy in Partnership with Department of Information and Communication Technology (DICT) (April- May 2023)
- Asia Open RAN Academy in Partnership with Rakuten Symphony (May to June 2023)
- Master IP Addressing and Subnetting for CCNA – FREE COURSE (May 2022)

WORK EXPERIENCE
Intern
Asian Institute of Aviation – Planters Aviation Corporation
Makati City
(August 2022 to September 2022)

REFERENCES
Kriztah Maebelle Onella
Marketing
Asian Institute of Aviation – Planters Aviation Corporation
0905-519-5017