8-25-2022 10:30 AM

Towards a Novel and Intelligent e-commerce Framework for Smart-Shopping Applications

Susmitha Hanumanthu, *The University of Western Ontario*

Supervisor: Haque, Anwar, *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Master of Science degree in Computer Science

Abstract

Nowadays, with the advancement of market digitalization accompanied by internet technologies, consumers can buy products from anywhere in the world. Finding the best-offered deal from numerous e-commerce sites and online stores is overwhelming, time-consuming, and often not very effective. Customers need to visit many online stores to find their desired product at the desired price. Also, the option of finding a product in the future time that is not currently available is limited in the current e-commerce platform. To address these limitations, there is a need to develop a new one-stop e-shopping model that would allow customers to search for any product in the marketplace from one central application. This thesis proposes a novel and intelligent e-shopping application framework that takes specific product/s input from the users and searches the e-commerce sites to find the best deal. Our proposed framework is also capable of learning from customers' complex behavior and their shopping experience, offering product recommendations accordingly. Our developed SmartCart prototype is a novel, effective, intelligent, scalable, and one-stop solution for shoppers to find desired products in real-time. Our proposed novel solution addresses some of the significant limitations of today's e-shopping platforms and is expected to enhance today's digitally savvy e-shopping consumers.

Keywords

E-Commerce, machine learning, data analytics, real-time processing, system scalability, android application, map visualization, MongoDB, smart shopping.

Summary for Lay Audience

Today's digital world has changed the way people shop. The current marketplace consists of online and offline stores offering various services to attract buyers. An average shopper now has access to numerous sellers and is exposed to a wide range of deals on identical products through multiple platforms. Consumers are forced to visit these individual platforms to find the best deal on their chosen product and thus spend considerable time on shopping research. This highlighted the need for a single e-commerce platform that consolidates existing deals and presents them to the end-user through a central e-shopping application. Everyday shopping also does not end at finding the availability and best price for one single product. This research identified the need for a complete shopping solution and proposed an intelligent solution to address that challenge. The thesis presents an innovative smart shopping framework that identifies the stores according to the user preferences, provides a centralized product lookup, and finds the stores that offer the desired products. The user is notified whenever a previously unavailable product becomes available. This novel model considers user preferences while locating the products and gives back real-time information about these products in a few seconds. Users' shopping behaviour and experience are also monitored as part of our developed prototype, and our recommendation system can offer relevant products based on users' shopping trend and preferences.

Acknowledgments

Firstly, I would like to express my sincere gratitude to my supervisor Prof. Anwar Haque for giving me the opportunity to conduct research on this exciting problem that directly impacts the online shopping experience and users' productivity. Thank you for your continuous support and guidance throughout my Master's program. Your insights and expertise have fast-tracked the completion of this research project in time.

To Roshini, thank you for mentally supporting and encouraging me through this journey. Eternally grateful to my parents for their support and encouragement.

I would like to extend my gratitude to Pavani, for helping me during code blocks and for technical support.

Table of Contents

List of Tables

List of Figures

Chapter 1

1 Introduction

1.1 Motivation

The rapid developments in communication and network technology changed many aspects of society, from business to recreation [1]. New forms of communication have introduced new ways of business [1]. These advancements resulted in E-Commerce emerging as a new economic activity. Electronic Commerce (EC) refers to the use of the Internet and modern communication technology for any form of business operation management or information exchange [2]. While globalization made it possible for people from one corner of the world to interact with people from other, digitalization has increased the usage of internet-supported electronic devices such as mobile phones, tablets, and tablets. These two factors created a domino effect in the field of e-commerce, as consumers can now purchase products from all over the world with a single click. A buyer looks up a product on the internet, chooses from the recommended options, and completes the payment. This gained significant popularity from the customers as it eliminates the travel and waiting times. The increased interest from the consumers has made many businesses to offers their services online through a web or mobile interface. These factors collectively introduced a new flourishing market on the internet, commonly called online shopping.

Online shopping has created new opportunities for small-scale and individual businesses. They can reach new consumer groups through digital marketing. The comfort of browsing and completing the purchase at home increased the consumer base for these stores. Today's marketplace includes both online and offline stores. Traditional in-store shopping has seen a dip as the customers prioritize convenience and reliability [3]. These businesses have either moved online or adapted a hybrid approach to sustain the competition.

The conveniences introduced by online shopping created new and higher requirements from the buyers [2]. The overwhelming amount of available purchasing options demanded considerable effort from the shoppers. Research shopping [4] emerged as standard practice where an intense comparison of price, ratings, and reviews on a product across all the selling platforms before arriving at a purchasing decision. Companies have recognized the commercial value of the routine and introduced innovative applications which present the ongoing deals on a particular product from different vendors to the user. These applications significantly reduce the search cost and help make decisions in a shorter time. The success of these applications encouraged more and more industries to integrate these tools into their business model. While some enterprises achieved significant results, others witnessed no financial

gains. Travel and tourism are among the most benefitted industries as the user is one click away from the best deals on flights. The websites and tools that compare the price and ratings of electronic devices have experienced immense growth in popularity, increasing the traffic to their parent sites. The common attribute of these industries is that the buyer is expecting a one-time purchase of a particular product. When integrated into daily shopping needs such as grocery and personal care products, these applications did not produce the same results. The increased search cost has discouraged the shoppers from actively using them. The current e-commerce marketspace lacks a single solution that offers an efficient search, maintains the communication with the user and assists in online and offline shopping.

1.2 Contribution - SmartCart

The needs for everyday shopping are very different from occasional shopping, and the current smart shopping applications are not equipped enough to handle these differences. An amateur shopper expects maximum benefits with minimal effort and resource usage. The consumer is expected to look for individual products and select the store of their choice for that product. This creates confusion as most existing applications do not provide provisions to save these selections. As a part of this study the following This study identified the limitations of the current e-commerce framework concerning day-to-day requirements and presented them below.

1. Lack of a central search system that handles the complete product search for the user.
2. The search results are based on user individual product selections and do not present all the available buying options for the complete product list.
3. The consumer is not aware of the availability of the product.
4. There is an absence of a price computation engine that adds up the total cost in every available store.
5. The cart list of most applications is built with data from the positive search results, which is missing information that can be useful to the user.

This thesis makes the following contributions to developing an intelligent e-commerce framework:

1. A comprehensive analysis of popular e-commerce applications is conducted to understand the current state-of-the-art, and a comparative view is presented.
2. The limitations of the existing framework and the new requirements are identified based on consumer needs and demands.
3. A comprehensive framework that addresses all the identified concerns is proposed and a prototype is developed. The features include:
    i. Personalized user content and interface are provided to the user.

ii. Users are given the ability to control their search vicinity. They can choose the location and the distance they are willing to travel to complete their shopping.
iii. A unified central search is developed, where only a product list is required from the user as the input. The search looks for the product's availability in the partner stores in the selected region and compiles a list for every store.
iv. The results include a custom map of stores with at least one product with total cost as the marker. This visualization technique presents valuable information with a single look.
v. More information about the location, distance, and the number of available products, along with an option to save the store, is provided to the user.
vi. The user is presented with two different lists - cart and store items. Cart items contain all the products searched by the user during one session, whereas the store items are the list of products available in the selected store.
vii. A notification service is added to the framework to inform the user about the availability of their desired product in the future.
viii. A recommendation system is proposed that can learn from the users' shopping behavior and experience, and apply the knowledge, later on, to serve the users better.

4. SmartCart – an android application is implemented as a proof of concept.

1.3 Thesis Outline

The rest of the thesis is organized as follows: A current literature review and an analysis of the existing innovative shopping applications are provided in Chapter 2. A comprehensive overview of the process for data collection, pre-processing techniques involved, and the final records is presented in Chapter 3. Chapter 4 describes the application's features, design, including the user interactions and interfaces. The implementation of the application is explained in Chapter 5. Chapter 6 proposed a recommendation system based on users' shopping behavior and experience. Chapter 7 talks about the technical challenges and setbacks experienced during the development cycle of our prototype. Concluding remarks and future work directions are presented in Chapter 8.

Chapter 2

2 Related Work

This chapter provides insights into the recent developments in the e-commerce industry. Section 2.1 discusses the academic research and studies on online and smart shopping. Currently available smart shopping applications are studied in Section 2.2. The features offered by these applications and their limitations are explored in detail. The results are tabulated and presented at the end.

2.1 Literature Review

E-commerce has been steadily growing popular throughout the years. To understand the phenomenon of online shopping, various studies focusing on different aspects are conducted. The study on e-commerce is widespread and cannot be constrained to a single focus point. A survey of published papers and journals on e-commerce from different fields is reviewed as a part of this thesis. The authors of [5],[6], and [7] analyzed consumer behavior and presented the factors influencing a consumer's decision to move towards online shopping. The advantages and disadvantages of online shopping were discussed in detail. Security is the main concern for people to take a back step for online shopping. Authors of [8],[9], and [10] performed a comparative analysis of online and offline shopping models. The pros and cons of each medium are discussed and presented from a consumer's perspective. A different perspective and the importance of human and social factors that influence the usability of an online commerce platform are presented in [11]. They stated privacy, trustworthiness, and easy navigation through the platform as the key indicators a user looks for in any application. The research is not just confined to understanding consumer behavior and attitude towards smart shopping. Numerous studies were focused on devising solutions to the problems faced by consumers in the real world and giving them a smooth shopping experience. Virtual Cart, an android application, was implemented in [12] to assist the user in online and offline shopping. The application offers many features such as directions to the store, a product aisle locator, a cyber cart, and order confirmation. An RFID-based bar code scanner was used in studies [13],[14], [15], and [16] to eliminate the waiting for billing by calculating the price while placing the product in the cart. These papers were focused on improving the offline shopping experience of the customers. The research on online shopping concentrates mainly on saving a shopper's time visiting every website to find the best deal on their desired product. A comparison of online products using data mining was presented in [17]. Web crawling and scraping techniques were used to gather the product information of various sites, and a GUI was developed for user interactions. A Price comparison system using Lucene was presented in [18]. The proposed approach employs a multi-threaded crawler to gather information and uses the Lucene library to implement the product search. The search mechanism is

supposedly faster than the traditional search as Lucene uses indexing for data storing and retrieval. Most works focused on online and offline shopping as separate entities and provided individual solutions. Developments in the field of Artificial intelligence and machine learning have seen tremendous leaps in the past years. These new learnings are incorporated into the existing frameworks to make intelligent applications. Several studies proposed various ways of integrating machine learning techniques into the e-commerce field of study. The authors of [19] proposed an e-commerce sales chatbot that uses natural language processing and machine learning to answer some simple customer queries. [20] analyses the different recommendation techniques currently employed in various online stores and presents their advantages and disadvantages. The research suggests the integration of semantics in the existing recommendation framework to achieve better results. [21] identifies different online shopping tasks where e-learning methods can be introduced. These tasks include customer service, fraud detection, cross-border transactions, and commerce forecast. Multiple learning algorithms for each task are presented to solve common e-commerce problems.

This thesis combines the two models as the consumer would want the best deal out of both marketplaces and presents a framework that supports both. It distinguishes itself from the existing literature as it is focused on providing a one-stop smart shopping application that addresses many of the practical concerns of a consumer in both worlds. It highlights the research gaps in the current e-commerce, such as the lack of a centralized search engine and price computation engine and provides a solution. The research focuses on identifying consumers' requirements, developing efficient solutions, and integrating them into a single framework.

2.2 Industry Tools Review

The current pool of shopping applications available on the internet offers various additional features to market themselves to ordinary people. Some of the popular comparison engines studied in detail have an overview of their features and limitations. The widely available applications in North America considered for this analysis include:

1. Flipp
2. ShopSavvy
3. Reebee
4. Instacart
5. Shopbot.ca

2.2.1 Flipp

With more than 50M downloads, Flipp is one of the most popular mobile-based price comparison applications. It offers services in the regions of Canada and the US. It is a one-stop marketplace for savings and deals [22]. Weekly flyers of all the local retail and global retail stores are made available to the user in one place. The user can look up a particular product using the search functionality, and all the available deals on that product are displayed to the user. A product list is constructed based on the user search and is accessible to the user to be used as a shopping list. The major drawback of the application is that while it displays the deals available in different stores, there is no option to select the store. The application successfully reduces the time spent as the user can avoid visiting the web pages of each store and search for the product. At the same time, the user is forced to make a manual effort to search for a product in every available store and keep a note which offers the best deal. The data of offline stores are limited to the products featured on the flyer. Overall, Flipp is mainly used as a one-stop for flyers rather than for price comparison.

2.2.2 ShopSavvy

Developed as a mobile application, ShopSavvy presents the user with local and online buying options for a searched product. This application allows the user to scan the barcode as an input in addition to the traditional search offered by other applications. Users can create a waiting list for a selected product, and the app notifies the user if there is a remarkable drop in the price. The user interface is not easy to navigate and can confuse an everyday shopper. If the user selects an online deal, the app redirects them to the selected store, which can be distracting if the intention is to just browse for information. There is a significant disparity between the data available for online and offline stores. The support for in-store shopping is notably lacking as the application does not offer a way to organize the searched product in a selected store.

2.2.3 Reebee

Another mobile application that offers similar functionalities as Flipp. Reebee displays all the available flyers to the user. It distinguishes the ongoing and upcoming flyers, which helps the users plan their in-store shopping. The basic search functionality lets the user do a manual look for a product, and the shopping list is made available to the user. Additionally, the user can now select the store which offers the best deal for a particular product, and a shopping list is created especially for the store. This facilitates the user to understand what products are offered at a better price at a single store. This is a step-up from the other applications, but the user is still expected to make some manual effort. The store-based shopping list does not include all the products, and there is no other way than searching and selecting

the products from a couple of stores to compare the total bill. The in-store product data is limited to the flyer products.

2.2.4 Instacart

Instacart is essentially a grocery delivery or pick-up service application. It lets the user select the products from a store and deliver selected products to them. The search functionality of this application comes with hidden price comparison engines, thus making it a subject for this research. The available deals are presented to the user as options when they look up a particular product, allowing the user to choose the best deal. Like Reebee, Instacart creates a new shopping cart for different stores, displaying the total amount of those products. Again, user effort is needed to select all the products in all available stores to properly understand the total savings. The process is tedious and time-consuming. There is no support for a complete shopping list where users can access all the searched products. The product database of Instacart is comparatively more extensive than its counterparts as it provides additional services which benefit the companies. So, they are willing to share their data.

2.2.5 Shopbot.ca

A web-based application designed for product and price comparison. The services are extended to 15 categories, including electronics, pets, office supplies, and many more. In addition to manual search, the user can browse via brands, categories, merchants, and many more. Price filtering is also available to the user. The interface is outdated and not interactive as the mobile apps. There is no feature to create a shopping list as the application is not catered to everyday needs. The product data is made up of online stores and does not provide any provisions for in-store shopping.

The primary purpose of any consumer-based application is to offer services tailored to users' needs. Some of the common requirements for such applications include an easy-to-use interface and the ability to navigate to different functionalities without difficulties. It is impractical to assume that users would complete their product search in one go. Hence, it is essential for e-commerce applications to provide a means for users to save their selections so they can continue in the future. One deciding factor defining the usefulness of such applications is the amount of time and effort saved by a user. An efficient application needs to minimize the time it takes to search and select a store. Another feature that increases the usability of an application is the quality of the information directly visible to the user. So an e-commerce application needs to be able to present price, location, availability of the product and other deciding attributes in a user-friendly fashion. The e-commerce applications mentioned in Table 1 are compared against these features to asses their usability.

Name of the Application	PLATFORM	FEATURES	LIMITATIONS
Flipp	Android/IOS	• Offers all digital flyers at one place • Provides individual product search and selection • Shopping list is created based on searched items	• Increased search cost if more than one product. • Limited Product database • No provisions to store the selected the store with best price.
ShopSavvy	Android/IOS	• Allows the user to search either by manual entry or bar code scan. • Redirects to the purchasing option whenever possible.	• Clustered and complicated user interface. • Increased search cost if a greater number of products are involved. • Lack of offline store product information.
Reebee	Android/IOS	• Information about current and upcoming flyers at different stores. • Overall shopping list is made of searched products • List of products selected in a particular store is available to user.	• Overall search cost is still high as list items needs to be searched individually and selected from each store. • No provisions for price calculation.
Instacart	Android/IOS	• Creates a shopping cart and price for each store based on the products selected in it. • Great collection of online and offline stores and their product information. • Integrated the on-delivery option.	• The price calculation involves only products selected for the store and not the entire shopping list. • The cart list cannot be used as traditional shopping list.
Shopbot.ca	Web based	• Provision to Categorial search is provided. • Can filter the price.	• Does not include everyday products. • No inclusion of offline stores • Selected product information cannot be stored

Table 1: Comparative analysis of existing smart-shopping applications

The results of the analysis are presented in table 1. This analysis demonstrates a definite gap between the expectations of the user and the features offered. The most visible drawback is the clustered and

complicated user interface. These applications make it harder for the user to navigate to different between functionalities as everything is packed on the main screen. The UI of SmartCart is designed to be user-friendly, and a panel is provided for the user to navigate between different functionalities. The screens are restricted to contain only the necessary information they are designed for, avoiding clustering on a single screen. Another notable flaw is the lack of a medium to display user-specific information. A shopper would be interested to see their previous searched or selected items and maybe want to continue from where they left. Some of the above-discussed applications address this issue by creating a cart items list, but it is restricted only to successful product searches. The framework proposed in this study provides multiple lists to the end-user. The search activity retains the information of previously added products. Cart displays all the products the customer has looked for irrespective of availability. A different component provides the list of products available along with the information of the store they selected.

The primary reason for using any of these applications is to save money without spending hours finding the deals. Even with these applications, the user is expected to spend considerable time exploring the best deals. To address this problem, a unified central search is integrated into the presented framework. The user is expected to provide just the list of products as the input, and the search operation looks for these products in all the available partner stores and compiles a list for every store to present to the user. A complete solution approaches the user when any new information of value to the user is added. The existing applications send many promotional notifications to the user. SmartCart's notification service notifies users whenever a product of their choice becomes available. Only relevant information is being passed without clogging the user's notification. The study recognized the flaws in the existing applications and the research gaps in the intelligent shopping area. A one-stop application that addresses all the discussed issues is presented as a solution.

Chapter 3

3 Data

This chapter is entirely focused on the data involved in the functioning of our proposed framework/application prototype i.e., SmartCart. Various methods, tools, and techniques are studied to integrate the real-time or close to real-time e-commerce data. More details on these are presented in the challenges section of this thesis. On weighing the pros and cons of these methods, generating synthetic data using existing datasets is marked as the better approach. Section 3.1 discusses the characteristics of different datasets needed for the application. The analysis of existing e-commerce datasets and their drawbacks from this specific application perspective is provided in Section 3.2. The Data generation and pre-processing are discussed in Section 3.3, and a look at refined data is presented in Section 3.4.

3.1 Characteristics and type of data

SmartCart is expected to present the user with product and vendor information. So, at higher level a retail dataset with product information and a location dataset with the local store information is needed. For the functioning of the tool, the product dataset should have at least the name, price, and description details. The Location dataset must have the name, and address columns to differentiate between different affiliates of the same store. These are basic expectations from each dataset.

The ideal and final product dataset is expected to have a product name, price, store name, address, and any other miscellaneous product-related information. The actual real-time data of the supermarkets was considered to be included in the application. The initial approach for data collection was to implement spiders/crawlers for all the websites and extract the product information. Most of the websites have anti-crawling principles making it impossible to crawl. The spiders were blocked by the websites after, thus preventing them from collecting the data. Another approach is to generate the data. The existing retail datasets were analyzed to select the one that fits the requirements of the application.

3.2 Analysis of existing e-commerce databases

Several E-commerce datasets are available on the internet, and they offer a different set of data. Some datasets focus on the financial front and have information about yearly sales across different stores, and some highlight the nutritional value of different food items. The following datasets are evaluated to implement this solution, and a detailed analysis is given.

1. Instacart Market Basket Analysis (Kaggle) [23]
2. Online retail Dataset (UCI Machine Learning repository) [24]
3. Food Data Central [25]

4. Open Food Facts [26]
5. Open Grocery Database Project [27]

3.2.1 Instacart Market Basket Analysis

It is actual real-time data provided by Instacart a grocery delivery application to Kaggle. Kaggle is an online data analysis and machine learning community. Currently, there are 50,000 datasets available for public access [23]. The initial intention was to develop a prediction to guess the user's next buy based on this data. Now, it is used by many data analysts and students who focus on the grocery industry. The dataset comprises six CSV files with information about aisles, departments, orders, products, and the history of orders. The file contains details of the products previously ordered by Instacart customers. The details include product_id, product_name, aisle_id, and department_id of 49688 products. These products are categorized into 22 departments, and the details are saved in departments.csv. Product and department files can be used for this research.

3.2.2 Online Retail Dataset

A transactional dataset with information about online transactions of a UK-based retail store over one year from 01/12/2010 to 09/12/2011. It is donated to the UCI Machine Learning repository under a public license [24]. The repository is a collection of different databases, domain theories, and data generators created to assist the machine learning community. The products provided belong to a single all-occasion gift category, and the properties of this data include quantity, price, invoice data, description, and country of the purchase. A total of 541909 transactions are documented in this dataset.

3.2.3 Food Data Central

A nutrient-focused food database maintained by Food Data Central [25]. This dataset provides unique information such as calories, nutrients, proteins, and nutritional values of various food items. A wide range of data, including the methods to calculate the nutrient value and various representations of the data, is provided by this dataset. Of all the tables provided, the food table with 424252 rows is the focus of this application as it contains the name and descriptions of various food products, which can form a sub-category for our application dataset.

3.2.4 Open Food Facts

An independent non-profit organization run by internet users [26]. Most of the information is uploaded by an everyday user, and it resonates highly with real-time data. It is a visual dataset that displays the product with its image. Offers the user an option to filter the data according to their needs based on

category, country, source of information, and much more. A total of 52959 products available for the Canada region are considered for this tool.

3.2.5 Open Grocery Database Project

A few researchers and developers recognized the lack of a reliable, easy-to-use, open-source grocery database and developed this project to provide a solution. Available via grocery.com, Open Grocery Database [27] is a UPC (Universal Product Code) centric database with information about 100k products. This database has a wide range of products ranging from groceries, dairy, everyday necessities to pet products. It is made of two downloadable files, one contains the information about the available brands, and the other has the product and brand name.

All these datasets come with their pros and cons. The size of the dataset corresponds to the amount of product information available in the dataset. Some of the datasets, such as the Online Retail Dataset, may have repeated products based on transactions rather than a product. Data diversity refers to the variety of the products available. Since the tool's focus is to assist in everyday shopping, the dataset, which includes products from various categories, is rated high. Only a portion of these datasets is needed for this application. Hence, measuring the ease of access to that desired data is essential. The dataset with minimal but valuable information is given priority. Food data central provides the product information along with the ingredients, and it is tedious to extract just the product name. While Open Grocery dataset has the advantage of large dataset, Instacart data has a proper categorization which can be an added asset to the application A tabulated analysis of these five datasets against the application's requirements is presented below in table 2. After careful analysis, Instacart Market Basket Analysis database is considered suitable for this study.

DATA CHARACTERICS	MARKET BASKET ANALYSIS	ONLINE RETAIL DATASET	FOOD DATA CENTRAL	OPEN FOOD FACTS	OPEN GROCERY DATASET
SIZE OF THE DATASET	49688 products	541909 transactions with repeated product data	424252	52959 products	More than 100k
DATA DIVERSITY	High	Low	Low	Low	High
EASE OF EXTRACTION.	Moderate	Easy	Difficult	Difficult	Easy
DATASET SPECIFIC CHARACTERISTICS	Product information is limited to purchased products.	Product information is not given priority.	Focused more on ingredients of the food rather than the food item	Too big to download and poor readability	Products scattered across different categories. Price of the product is unavailable.

Table 2: Evaluation of retail and food datasets

Unlike product datasets, there are no credible sources for location datasets available on the Internet. The store locations need to be accurate to use as the APIs employed for the framework deal with real-time addresses, and the application dataset will be constantly verified against the real data. There is no single source to access the location information about all the selected stores in the city of London. There are numerous blogs on the internet that states the store details but there is no standard way to comparing them.

3.3 Data Collection and Pre-Processing

As mentioned in the above section, this framework requires two datasets for effective functioning. One dataset contains the information about the stores on board to share their internal data. The other dataset comprises all the products with details currently available in those stores. As there is no existing dataset, we have generated the dataset to suit the needs of the application. The location dataset that has been created manually limits the scope of the application to the city of London, Ontario. This action also ensures the credibility of the store locations. The most popular and commonly used grocery retail chain stores such as Food Basics, FreshCo, Metro, Real Canadian Superstore, Sobeys, and Walmart are included in this dataset. The locations of various branches of these stores have been taken from the official websites. A CSV file containing the store name, address, and zip code is generated. A snapshot of the CSV is shown in the figure below.

A	B	C	D
Id	Store Name	Address	ZipCodes
1	Sobeys Extra Oxford & Wonderland	661 wonderland Rd N, London, ON, N6H 0H9	N6H OH9
2	Loblaws Richmond	1740 Richmond St N, London, ON, N5X 2S7	N5X 2S7
3	Loblaws Wonderland	3040 Wonderland Rd S, London, ON, N6L 1A6	N6L 1A6
4	Freshco Trafalgar & Highbury	1298 Trafalgar St, London, ON, N5Z 1H9	N5Z 1H9
5	Freshco Huron & Adelaide	1080 Adelaide Street, London, ON, N5Y 2N1	N5Y 2N1
6	Freshco Commissioners	645 commissioners Rd E, London, ON, N6C 2T9	N6C 2T9
7	Freshco Wonderland & Pine	Unit 1, 981 Wonderland Rd S, London, ON, N6K 4L3	N6K 4L3
8	Food and Drug Basics	509 Commissioners Rd W, London, ON, N6J 1Y5	N6J 1Y5
9	Food Basics Ernest Avenue	1401 Ernest Avenue, London, N6E 2P6	N6E 2P6
10	Food Basics Oxford Street	1299 Oxford Street E, London, ON, N5Y 4W5	N5Y 4W5
11	Food Basics Wonderland Road	1225 Wonderland Road N, London, ON, N6G 2B9	N6G 2V9
12	Food Basics Commissioners Road East	1200 Commissioners Road E, London, N5Z 4R3	N5Z 4R3
13	Walmart Whiteoaks	1105 Wellington Rd, London, ON, N6E 1V4	N6E 1V4
14	Walmart SuperCentre Argyle Mall	330 Clarke Rd E, London, ON, N5W 6G4	N5W 6G4
15	Walmart SuperCentre Fanshaw Pk Road	1280 Fanshawe Park Rd W, London, ON, N6G 5B1	N6G 5B1
16	Walmart Highbury Ave	1275 Highbury Ave N, London, ON, N5Y 1A6	N5Y 1A6
17	Sobeys North London	1595 Adelaide Street North, London, ON, N5X 4E8	N5V 4E8
18	Metro Wellington rd	395 Wellington Road, London, ON, N6V 4P9	N6C 5Z6
19	Metro Oxford St W	301 Oxford St W, London, ON, N6H 1S6	N6H 1S6
20	Metro Adelaide St	1030 Adelaide St N, London, ON, N5Y 2M9	N5Y 2M9
21	Real Canadian Superstore Oxford	1205 Oxford St, London, ON, N6H 1V9	N5Y 3J8
22	Metro Clarke Rd	155 Clarke Rd, London, ON, N5W 5C9	N5W 5C9
23	Metro Commissioners rd W	1244 Commissioners Rd W, London, ON, N6K 1C7	N6K 1C7

Figure 1: Store Data

The analysis of the existing datasets revealed that no single dataset had met the requirements of this thesis. The dataset needs to be generated synthetically by utilizing the product information provided in one of the analyzed datasets. The product names from the product.csv file of Instacart Market Basket analysis are extracted to create the synthetic data. As a next step, store and location information is added to these products. This will create an inventory for each store and replicates the actual data. The location dataset from earlier is used for this purpose. There are some notable drawbacks to the dataset generated. In the real world, a single product is available in multiple stores. In the case of retail chains, the same product may have different prices at different locations. These factors must be considered to make the synthetic data resemble the real-time data. The following steps are taken to address these issues.

1. The extracted product name dataset is multiplied by ‘6’ before integrating it with the store location dataset. This ensures that every single product is available at six stores. Measures have been taken so that the replicated products would never be assigned to the same store.
2. Postal code is included as a separate column, and store name along with it is used to recognize the different associates of chain stores.

Thus, the existing public market basket analysis dataset and the manually generated location dataset are used to generate a synthetic product dataset. Python and its libraries Pandas and NumPy are used for data pre-processing. The final snipper of product dataset is shown in the figure 2. The column product

name represents the name of the product, store name is the name of the store in which the product is currently available, and price represents the price of the product in that store.

A	B	C	D	E
	Name	Zipcode	Store	Price
0	Chocolate Sandwich Cookies	N6H OH9	Sobeys Extra Oxford & Wonderland	7
1	Chocolate Sandwich Cookies	N5X 2S7	Loblaws Richmond	13
2	Chocolate Sandwich Cookies	N6L 1A6	Loblaws Wonderland	5
3	Chocolate Sandwich Cookies	N5Z 1H9	Freshco Trafalgar & Highbury	11
4	Chocolate Sandwich Cookies	N5Y 2N1	Freshco Huron & Adelaide	11
5	Chocolate Sandwich Cookies	N6C 2T9	Freshco Commissioners	17
6	All-Seasons Salt	N5X 2S7	Loblaws Richmond	16
7	All-Seasons Salt	N6H OH9	Sobeys Extra Oxford & Wonderland	5
8	All-Seasons Salt	N6L 1A6	Loblaws Wonderland	18
9	All-Seasons Salt	N5Z 1H9	Freshco Trafalgar & Highbury	7
10	All-Seasons Salt	N5Y 2N1	Freshco Huron & Adelaide	14
11	All-Seasons Salt	N6C 2T9	Freshco Commissioners	17
12	Robust Golden Unsweetened Oolong Tea	N6L 1A6	Loblaws Wonderland	6
13	Robust Golden Unsweetened Oolong Tea	N6H OH9	Sobeys Extra Oxford & Wonderland	13
14	Robust Golden Unsweetened Oolong Tea	N5X 2S7	Loblaws Richmond	15
15	Robust Golden Unsweetened Oolong Tea	N5Z 1H9	Freshco Trafalgar & Highbury	15
16	Robust Golden Unsweetened Oolong Tea	N5Y 2N1	Freshco Huron & Adelaide	16
17	Robust Golden Unsweetened Oolong Tea	N6C 2T9	Freshco Commissioners	13
18	Smart Ones Classic Favorites Mini Rigatoni With '	N5Z 1H9	Freshco Trafalgar & Highbury	15
19	Smart Ones Classic Favorites Mini Rigatoni With '	N6H OH9	Sobeys Extra Oxford & Wonderland	14
20	Smart Ones Classic Favorites Mini Rigatoni With '	N5X 2S7	Loblaws Richmond	12
21	Smart Ones Classic Favorites Mini Rigatoni With '	N6L 1A6	Loblaws Wonderland	4
22	Smart Ones Classic Favorites Mini Rigatoni With '	N5Y 2N1	Freshco Huron & Adelaide	8
23	Smart Ones Classic Favorites Mini Rigatoni With '	N6C 2T9	Freshco Commissioners	10
24	Green Chile Anytime Sauce	N5Y 2N1	Freshco Huron & Adelaide	7
25	Green Chile Anytime Sauce	N6H OH9	Sobeys Extra Oxford & Wonderland	11
26	Green Chile Anytime Sauce	N5X 2S7	Loblaws Richmond	13
27	Green Chile Anytime Sauce	N6L 1A6	Loblaws Wonderland	17

Figure 2 : Product and Store Data

Chapter 4

4 Design

The design of the tool, along with its features and the decisions involved, are explained in this chapter. A detailed description of the features to be included in the framework is presented in Section 4.1. The system requirements and constraints are presented in Section 4.2. A high-level use case diagram of the application is illustrated in Section 4.3. Section 4.4 explains the various design stages, such as interface design, wireframing, prototyping, and the actual mobile interface design. The importance of data visualization in presenting the results and performative analysis of three visualization techniques within the application context are provided in Section 4.4.

4.1 Novel Features of the Application

The comprehensive analysis of the existing online shopping applications and the supporting e-commerce framework provided the necessary groundwork for understanding the actual needs of a consumer. These requirements are considered while designing the framework's features. The proposed solution offers the following features to the end-user.

4.1.1 User Profile and Preferences

User profiles are the driving force for any organization as they provide insights into who the user is and what they are looking for [28]. The information gathered can help understand the individual user and cater to their specific needs. The submitted solution captures every piece of data entered by the user during their interaction with the application. Basic personal information of the user gained during the registration phase is presented through a Profile Screen. It is allowed to be edited by the user at any point in time. In addition, user selections such as searched items, cart items, and store information, if selected, are preserved in the application until manually deleted. This encourages usability as they can continue where they left off without any additional effort. The framework can also save the user preferences such as their selected radius and store them in the device for future actions, thus creating a personalized environment for the user.

4.1.2 Location Selection

An internet product search provides a shopper with thousands of results, many of which can be unfruitful. So, presenting only relevant and valuable results to the user is essential and can be achieved by incorporating location-based services into the framework. Location-based services refer to the software services that use geographical data to provide services to their users [29]. SmartCart uses the geolocation

given by the user to perform any requests. Users are given a choice to enter their location manually or can allow the device to detect it. A store locator is designed to identify the partner stores within the user's location. Provisions to further filter these outlets are made available to the user in two ways. The user can either set a radius to restrict their buyer options within that vicinity, choose a particular retail chain, or both. The product search is now restricted to the stores that conform to the user preferences. The results are accompanied by navigation to the store if the user chooses any in-person shopping option. This feature guarantees that only obtainable and feasible buying options are suggested to the user.

4.1.3 Novel Centralized Product Search

A search engine is a crucial component of every shopping application. The current mechanism has two major flaws. User is expected to find the best deal on a product by checking all the available platforms. While some applications offer a unified search, this is restricted to a single product and requires the user to look for their desired product list individually. The present market space of the above-mentioned applications is either restricted to stores with an online presence or to a certain category of products. A novel search approach that addresses these limitations is presented in this study. Partner stores are expected to include local offline stores, big retail chain stores, and online outlets. Users can search for all of their products in the common space, and the application will compile the list of available products in every store within their search vicinity. The extra effort of looking for the best deal for each product is now replaced with a single consolidated search. A compiled list of available and unavailable products at each store is presented to the user.

4.1.4 Price Computing Engine

Finances are a significant factor in any purchasing decision, and all the research and browsing done by a shopper is to get the product they need at a lower price. For a list of products, the total price at one store is substantial rather than deals on the individual product at different stores. Considering this behavior, an innovative price computing engine is included in the proposed framework. In addition to presenting the available products in each store, the application computes the total price and includes it in the results. Consumers have more information about their expected spending even before confirming the purchase. The knowledge about available products and the cost at filtered stores will assist the shoppers in making better purchasing decisions quickly.

4.1.5 Push Notifications

Communication is the key to keeping a user interested and engaging with any service. Notifications are a great way to update the user about recent additions or modifications to the application. Most e-

commerce applications end-user interactions after a product search, irrespective of the result. The standard expectation is that users will continue looking every day until they find it, which may not be the ideal scenario. An innovative way to approach this situation is to note failed searches or unavailable products and notify the user when they become available. SmartCart is configured with a notification service that sends push notifications to the user whenever a product previously unavailable becomes available in their selected store. The decision to register for this service is given to every user after their store selection.

4.1.6 Highly Scalable

Scalability is considered during every design and implementation decision. Backend-as-a-service is used to host the application and is equipped to scale horizontally and vertically if required. The application database is created on a cluster configured to handle load imbalances and route the requests to appropriate nodes. From the front-end perspective, the requests to the server are restricted to a minimum level. Frequently used data is stored as cache and fetched from the device whenever needed, thus avoiding the additional database calls. Users can be logged in even after exiting the application to avoid unnecessary authentication requests.

4.1.7 Real-time Information

One main characteristic of e-commerce data is its vulnerability to change. A product can be available at one minute and unavailable at the next. Other parameters such as price and quantity, are also expected to change at any time. The proposed framework is designed to adapt to these changes quickly. The database for the application is selected by giving more priority to writing requests. The cost of inserting a new product is kept at a minimum. The search mechanism is designed to request the database for products for every session to avoid stale data and maintain the real-time data at all times. Notifications are sent to a user in real-time whenever there is an update on the product they registered for. Partners are expected to notify any changes to their product catalogues, and they will be reflected in the database soon after.

4.2 System Requirements

For quicker and more efficient implementation of the prototype, the scope of the application is currently constrained to certain categories and regions. The city of London in Ontario, Canada, is selected for the initial developments, and the store finder is limited to this region. The product catalogue is restricted to everyday utilities such as grocery, personal care products, pharmacies, etc. The product data utilized in this project is generated synthetically by applying various pre-processing techniques for existing datasets

to make it close to the actual database of products. Offline stores are given the same importance as online, and the data reflects that. The local retail stores of Canada like Sobeys, Loblaws, Food Basics, FreshCo, along with Walmart, Metro, and Costco, are considered for data generation. The application is made to work by considering the above constraints and the assumptions that the local stores are willing to become partner stores and share their data.

4.3 Application Design

The detailed analysis of the current market, consumer behavior, and the existing shopping engines has revealed the need for a smart e-commerce tool. Some of the factors considered during the design phase of the tool include accessibility, easy navigation between various workflows, and visual representation of the results. Emphasis is placed on displaying only quality and relevant data as the final result. To maximize usability and accessibility, the tool is developed as a mobile application. Figure 3 shows a use case diagram depicting the different use cases and user interactions.

The native Mobile application model is chosen to develop the application for faster and more efficient implementation as the focus of the thesis is to show the significance of a user-friendly price calculation tool. This allows leveraging the platform's operating system features without worrying about compatibility. Currently, Android and iOS are the biggest platforms for mobile applications. [30] provided a comparative study on the performances of a single functionality application on both platforms. After careful analysis of this study results, requirements of the application, available resources for each platform, and knowledge stack, Android is decided as the platform for implementation

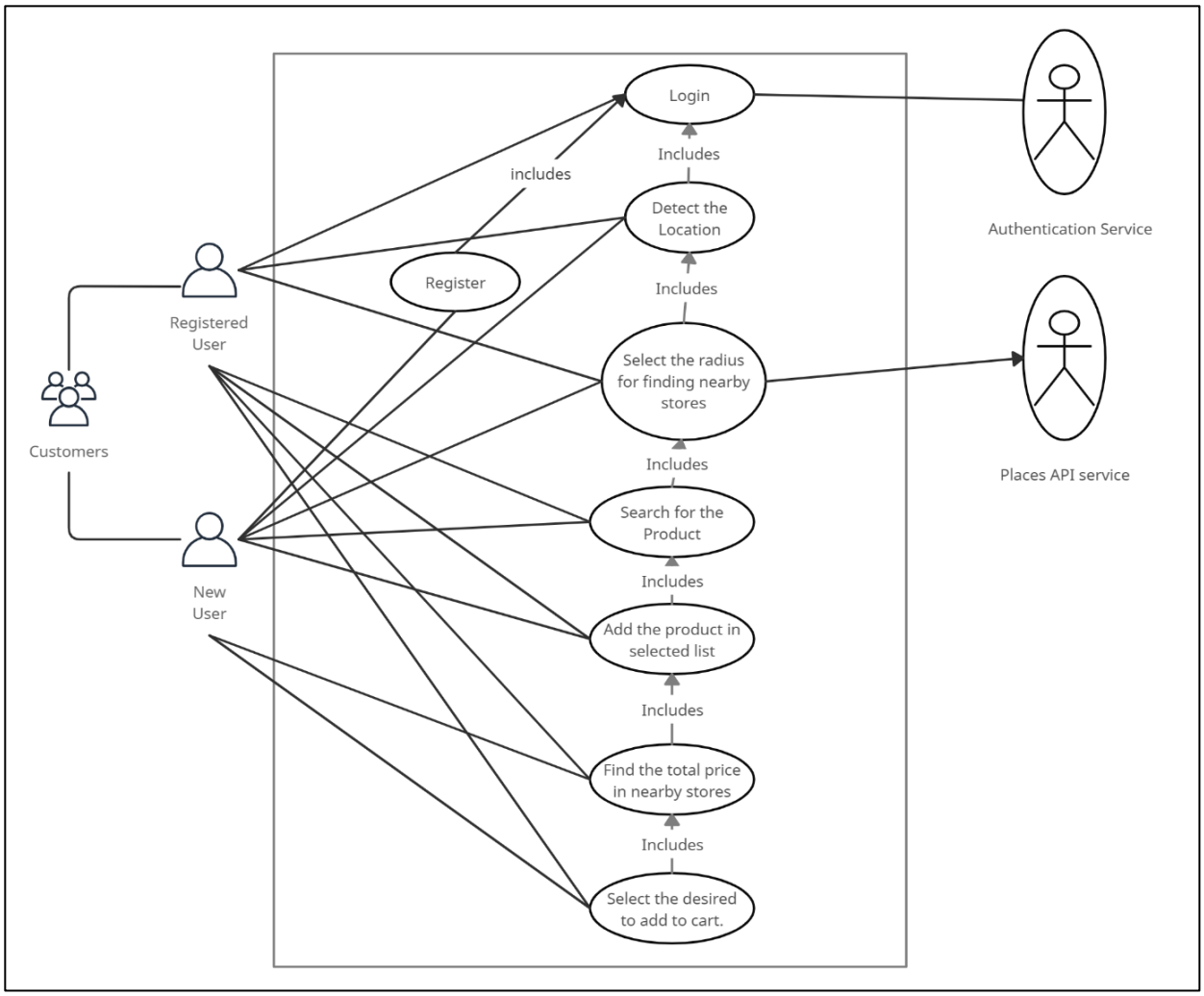

Figure 3 : High-level use case diagram

4.4 Interface Design

The use case diagram provides an insight into the various functionalities provided to the user and the required interactions for it. Android uses activities and fragments to control the workflow. Considering the dependencies of each use cases and the android components, a screen flow diagram illustrating the activities and fragments needed for the application is created. Figure 4. The dotted rectangles refer to the individual screens. The services and adapters required for the application are also shown.

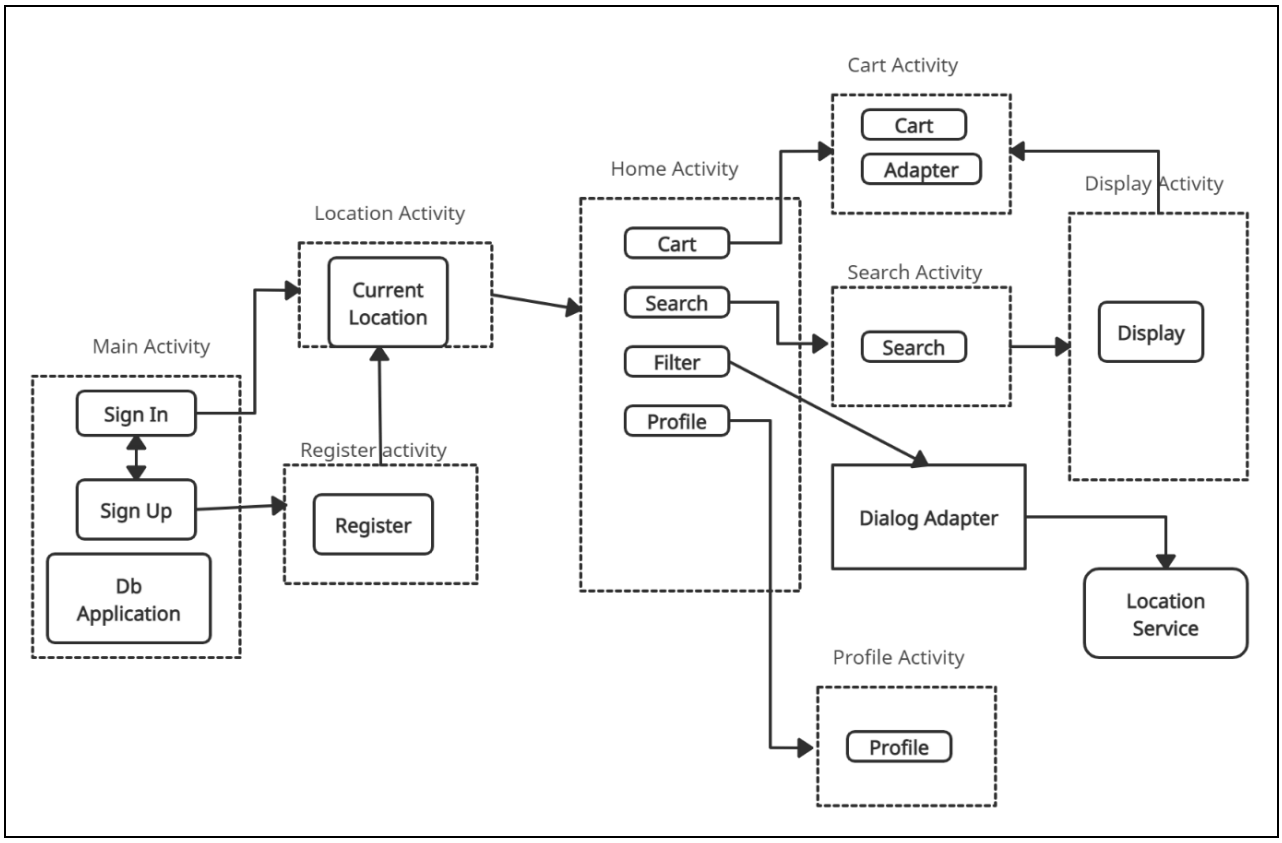

Figure 4 : Screen Flow Diagram

4.4.1 Wireframes and Prototypes

Wireframes are essentially a skeleton for any product development [31]. It contains critical information about the elements displayed on the user interface. They help shape abstract ideas into a proper outline with structure, layout, and direction. Design flaws can be captured in the initial phases by testing using these wireframes. A traditional pen and paper approach was utilized to design the initial wireframes for this project. The sketched wireframes are shown below figure 5.

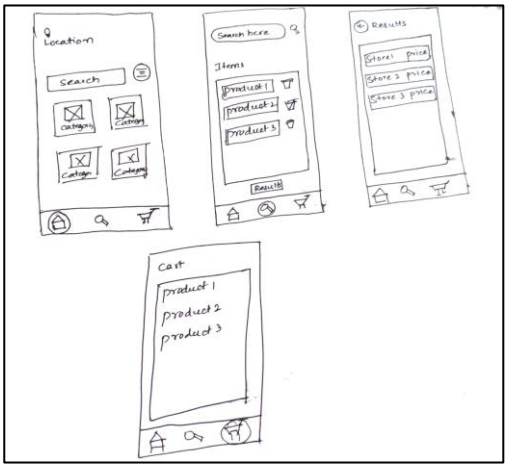

Figure 5: Pen and paper traced wireframes

A prototype illustrates the working model of the user interface for the application [31]. It focuses on user interactions, which include navigation between screens and outcomes of certain interactive events. These provide more insights into the usability of the application. A low-fidelity prototype has been created using Figma. Figma is a prototyping tool that provides many features to replicate the exact user journey and helps in improving the user experience. Some of the designed prototype screens are shown below in figure 6.

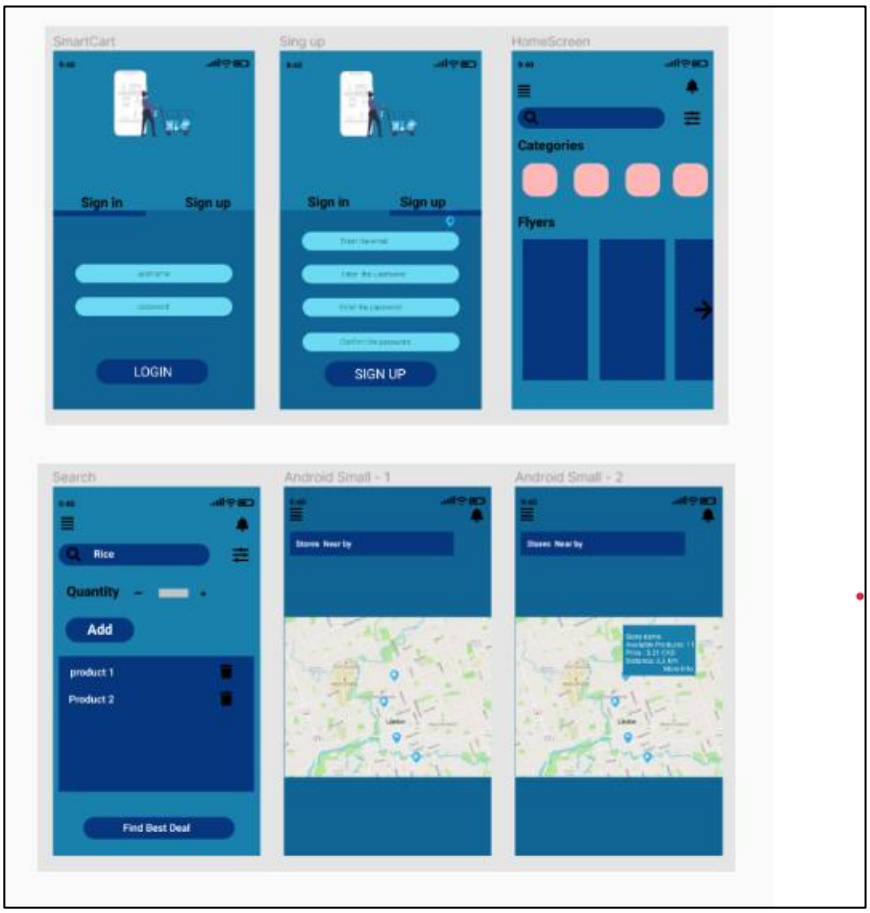

Figure 6: Protypes designed using Figma

4.4.2 Mobile Interface Design

It is explicitly stated from the beginning that representing the data in most understandable format is given at most importance. Most existing applications fail to provide the available data in a simple and useful way. To make this application stand out from the heaps of price comparison applications, various data visualization techniques are studied to integrate visual elements for the display of results.

4.4.2.1 Data Visualization

The human mind is very visual [32]. Hence, presenting the data in a simple and visually pleasing amplifies the analyzing capabilities of a user. Visualization refers to forming a sensible picture in a person's brain [33]. Data visualization commonly refers to presenting information in a graphical or pictorial form [33]. These techniques improve the efficiency of recognizing the data by a human as visual stimulation helps them understand the data easily. During online shopping and price comparison, most of the time is spent on aggregating the price and product information and thereby making a decision. This thesis uses the concept and techniques of data visualization to greatly reduce this time by presenting the direct price of products for the corresponding store.

The most common visualization techniques include graphs, charts, tables, dashboards, and maps [34]. A detailed analysis of integrating bar graphs, tables, and maps into the proposed tool is conducted, and final design decisions are taken based on them.

4.4.2.1.1 Bar Graphs

Bar graphs are usually used to represent categorical data. The x-axis presents the different categories, and the y-axis represents the values of a common attribute. The height of the bar corresponds to the value of the attribute for that category. For this application, stores can be shown on the x-axis and total cost in CAD on the y-axis. The peak of each bar gives the total cost for all the products in that store. With this approach, the user can instantly see which store offers the lowest price. To facilitate the user, data labels on top of each bar are added to display the number of products available in that store. A simple bar graph is presented in figure 7 below, and the user can infer the following information with a single glance at the graph.

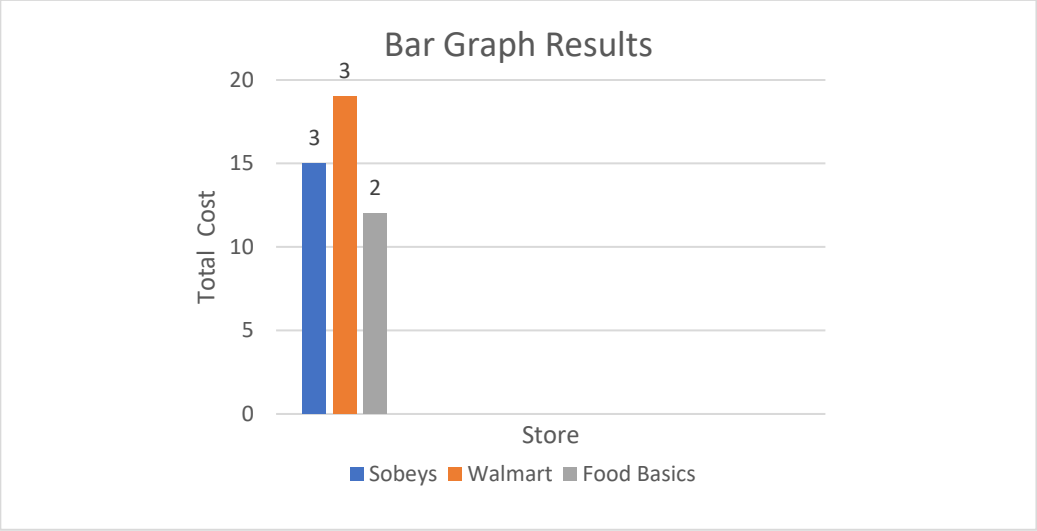

Figure 7: Bar Graph Visualization

1. The application identifies three stores based on the user location and product inputs.
2. An estimate of the total price can be seen from the y-axis for each store.
3. The data labels on each bar show that only two products are available in Food Basics.

With the above information, the user can conclude in a short time that Sobeys offers all the products at the best price. There are certain limitations to this type of visualization.

4.4.2.1.2 Tabular Visualization

One of the most popular and common visualization types is the tabular format. The rows and columns of the table can display a large amount of data in an orderly fashion. From the application perspective, rows can represent stores, and different decision factors can be expressed as columns. The tabular visualization results are displayed in figure 8.

Store Name	Total Cost (CAD)	Products Available	Distance (KM)	Address
Sobeys	15	3	3.2	190 Fleming Dr
Walmart	19	3	1.2	348 Oxford St
Food Basics	12	2	2.5	324 Western Rd

Figure 8: Tabular Visualization

The user can analyze the table data and conclude the savings on each store. This format provides information about the price, number of products, distance of the store from the user's current location, and the store's address. Compared to the bar graph, the user may not locate the store with the best price as it is not as noticeable as the bars. The additional information can be helpful if they are looking for an in-store experience.

4.4.2.1.3 Map Visualization

Map visualization is primarily used to analyze and display geospatial data, and this visualization method is preferred when there is any location-based or geographical data. A custom google map is designed to integrate this technique into the proposed tool. The usual marker is replaced with a textbox that displays

the price at the location of the store. The click event on the marker is created to display the available product information, distance, address, and directions to the store.

Maps are interactive and intuitive when compared to bar graphs and tables. This visualization supports the “one glance” attribute of the bar graph and the “additional information” characteristic of the table. The visual stimulation is higher in maps as they look more familiar and casual than the other two. Figure 9 shows the visual representation of tool results in maps.

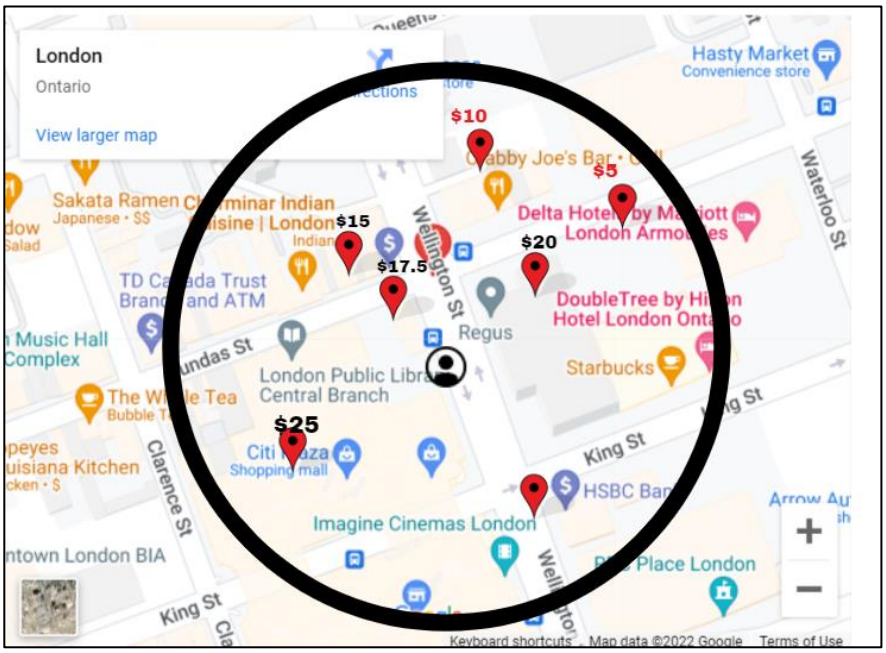

Figure 9: Map Visualization

The result of a comparative analysis is presented in table 3. From the analysis, Maps are found be the most expressive and useful visualization type. The final interface was designed by following the basic structure from the prototype and introducing interactive UI elements present in android studio. The results are displayed by integrating google maps into the application.

Visualization Type	Data inference	Available Data	User Interact ability
Bar Graph	Easy	Low	Low
Table	Moderate	High	Low
Maps	Easy	High	High

Table 3: Tabulated comparison of visualization techniques

Chapter 5

5 Implementation

This chapter focuses on the core implementation of the application. The architecture of the application and the software development model employed to develop it are discussed in Section 5.1. Section 5.2 introduces the programming languages, IDEs, libraries, frameworks used in the project and third-party APIs. The use of MongoDB as a database and back-end server is explained in Section 5.3. The application development is discussed in Section 5.4. The complete workflow is demonstrated as a case study in Section 5.5.

5.1 Architecture and Software Development Model

5.1.1 Architecture

The architecture of the SmartCart is shown below in figure 10. It is developed using Backend as a service architecture. It is made of two layers: the UI and Baas layers. The UI layer is responsible for the user interactions and relays the user requests to the Baas Layer.

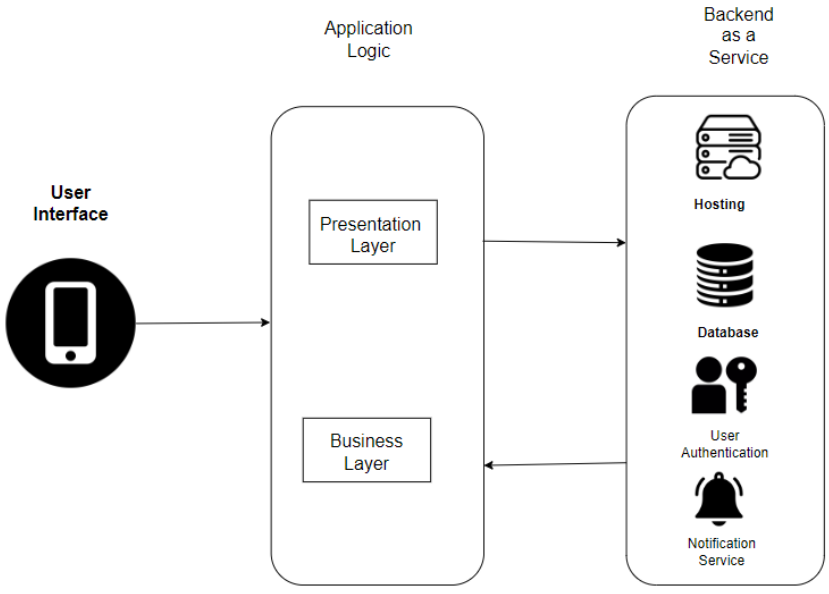

Figure 10: Application architecture

5.1.1.1 Backend as a Service Architecture

Back-end as a Service, commonly referred to as Baas, is a cloud-based architecture model where the back-end development of the application is automated and handled by a service provider [35]. The workload of the development team is reduced as the usual back-end functionalities such as storage and data management, database access, authentication, APIs, and hosting of the application are managed by a third-party service. Reduced infrastructure cost and scalability make it attractive for application developers. By subscribing to BaaS, the developer can avoid the hassles of logging; a dashboard of usage and analytics is provided by almost all the providers. MongoDB Realm is chosen as the service provider for SmartCart

5.1.2 Rapid Application Development (RAD)

The advantages and disadvantages of different software development models was analyzed in [36]. Considering the project requirements and timelines, the rapid application development software model is adopted in building this application. First introduced by James Martin in 1991, it has quickly gained traction as a faster and quality system development model. A shorter planning phase followed by an elaborated implementation, testing, and feedback phase is the norm for RAD. An initial prototype of the application is developed based on the primary ideas and presented for feedback. The shortcomings are then discussed, and the project re-enters the implementation phase to address them. Figure 11 presents the flow in RAD.

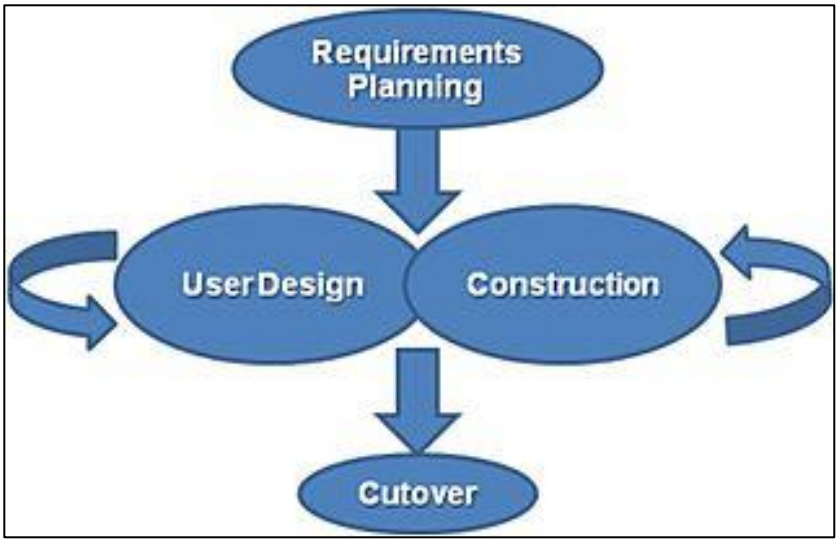

Figure 11: Rapid Application Development Model [37]

An initial prototype with just product search and price calculation was developed to test the tool's functionality. Feedback on user interactions and additional features were implemented in the next phase.

5.2 Programming languages, IDEs, and Third-Party tools

This section provides complete information on the tech stack involved in the application development. Section 5.2.1 discusses Java as the primary programming language, while the Android development platform and its features are described in Section 5.2.2. The development tool Android Studio is explained in Section 5.2.3, and Section 5.2.4 presents information on the third-party APIs integrated into the application.

5.2.1 Java Programming language

Java is one of the most popular high-level, object-oriented programming languages. It used to be the official language for android application development before Kotlin. The abundance of community support and development resources on Java makes it the first choice for any developer. It is a platform independent programming language which aligns with the platform independent development requirement for android. The OOPS concept of JAVA such as classes, objects, and polymorphism greatly helps in defining the application functionalities. The rich set of libraries and SDK support available ease the development burden. In addition to the above advantages, JAVA provides a secured approach with no memory leaks.

5.2.2 Android Development Platform

Android is a software stack for mobile devices that includes an operating system and middleware, created by Google's Open Handset Alliance. It provides capabilities to utilize the device features such as GPS, camera and much more. It is an open-source project and can run on various devices such as mobile phones, tablets, etc. It provides a unified approach and ensures that a developer does not need to worry about compatibility on different devices. The components of Android development used in this project are described in the following sections.

5.2.2.1 Activities

Activity refers to a screen that provides the user interface and handles user interactions for the application. It extends the Activity class and overrides the OnCreate() method. This method is responsible for the initial layout of the screen. Every newly created activity needs to be declared in the manifest file. Activities usually contain a specific feature or functionality associated with it.

5.2.2.2 Fragments

It is a part of the interface bounded to an activity. Fragments are commonly used to design the reusable components of the UI. They introduce modularity into the application by dividing the UI into discrete chunks.

5.2.2.3 Application

The Application is a base class and contains all the other components. It is instantiated before any class. Custom Application classes are essential in scenarios where a task needs to be run before the first activity, global initialization of an object and sharing immutable data across all components. The objects defined inside the OnCreate() method are expected to be alive throughout the application life cycle. It is common to instantiate the database connection in the Application class to make sure only one connection is established during a session.

5.2.2.4 Shared Preferences

SharedPreferences APIs are used to store a small set of key-value pairs on the device file storage. It supports the storage of primitive datatypes such as int, float, string. It is extended to custom objects using gson. Mobile applications are expected to keep the user information available until the user logouts. DB calls can be expensive in such cases and android presents the option to store the details such as username, and search results on the physical device through shared preferences.

5.2.2.5 Location Manager

An android class to access the device location services. This class offers methods to get the user's current location and to notify the system whenever the location is updated. It is integrated with various location providers such as GPS and network based to guarantee the accuracy and availability of the location information at all times.

5.2.2.6 Intent

Intents are commonly used to start an activity, services and send information between two activities. Extras are used to carry data in the form of key-value pairs.

5.2.3 Tools and Third-Party APIs

5.2.3.1 Android Studio

Android studio is the official Integrated Development Environment (IDE) for android app development [38]. In addition to the powerful code editor, it offers various features which make it easier to develop

an application. It offers support for all android devices; hence the developer does not have to worry about compatibility across different devices. A powerful and flexible Gradle-based build system to run the app. The written application can be tested simultaneously either on a feature-rich emulator or by connecting the actual device to the system over USB or Wi-Fi. Git version control can be used to track the progress and changes made to the application. Different features of this IDE are presented in the following section.

5.2.3.1.1 Gradle Build System

Gradle is an open-source build system that is used to automate the building, testing, and deployment stages of an application[39]. It is flexible enough to build every type of software. Android studio utilizes Gradle to build the system whenever changes are pushed. An android plugin for Gradle was developed to accommodate all the needs of android studio. The build files are named as build.gradle. Every project has two build files one for project-level and the other for the module level. The Gradle files also handle the dependencies for the project. The required dependencies are mentioned in the module level file and build system is responsible for making them available.

5.2.3.1.2 Android Virtual Device

Android Studio provides a faster way to test the changes to the application. A virtual device can be configured to replicate the behavior of an android phone, tablet, or OS devices. The configured device can be accessed via device manager. Emulators help in stimulating these devices and test the application. Multiple versions and OS can be tested using AVD in lesser time when compared to physical device.

5.2.3.1.3 USB Debugging

Android Studio provides support for running and debugging of the application on an actual device. Android device settings are configured to enable USB debugging. This feature allows the developer to implement and test the application simultaneously.

5.2.3.2 FourSquareAPI

One of the most critical features of this project is to suggest places based on user location and within the chosen radius. To implement this feature, an API that returns the nearby grocery stores is needed. FOURSQUARE PlacesAPI is used to complete this action. Foursquare is a technology and cloud platform company, and its primary products include Pilgrim SDN and Places API.

5.2.3.2.1 Places API

Places API was launched in November 2009 and currently has a database of more than 105 million places worldwide and an API that provides location data. It is widely used in many top companies, including Apple, Samsung, Microsoft, Tencent, etc.

A Foursquare account is needed to access the available APIs [40]; it provides the user with a Client ID and Client Secret, which is used for authorization while fetching data.

For this project, places API specifically, the venues have been used. Venues is a location-based API that finds different types of stores such as grocery stores, Entertainment, Shopping units, and much more in the vicinity of a particular given location. A typical request to the API is made of client id, client secret, radius, latitude-longitude coordinates of the location, category of the venue, and maximum expected results. Many parameters provide the user with a very comprehensive response. The parameters mentioned earlier are included in the API request of the project. Radius refers to the distance in kilometers the user selected for the search. The official documentation of foursquare provides the details of the unique ID for different types of venues, and the category for supermarkets is selected for this project. The request is shown in figure 12.

CategoryID: 52f2ab2ebcbc57f1066b8b46

```
https://api.foursquare.com/v2/venues/search?
client_id=&client_secret=&
v=20180323&ll=42.9834,-81.233&
radius=20000&
categoryId=52f2ab2ebcbc57f1066b8b46
```

Figure 12: FourSquare Place API Request

The response received from the API is a JSON object with quite a lot of information about the selected category places. Primarily the response contains the details of the venue such as name, location, distance from the selected location, latitude and longitude coordinates of the venue, postal code, and formatted address. A small snippet of the response is shown below. Fig 13.

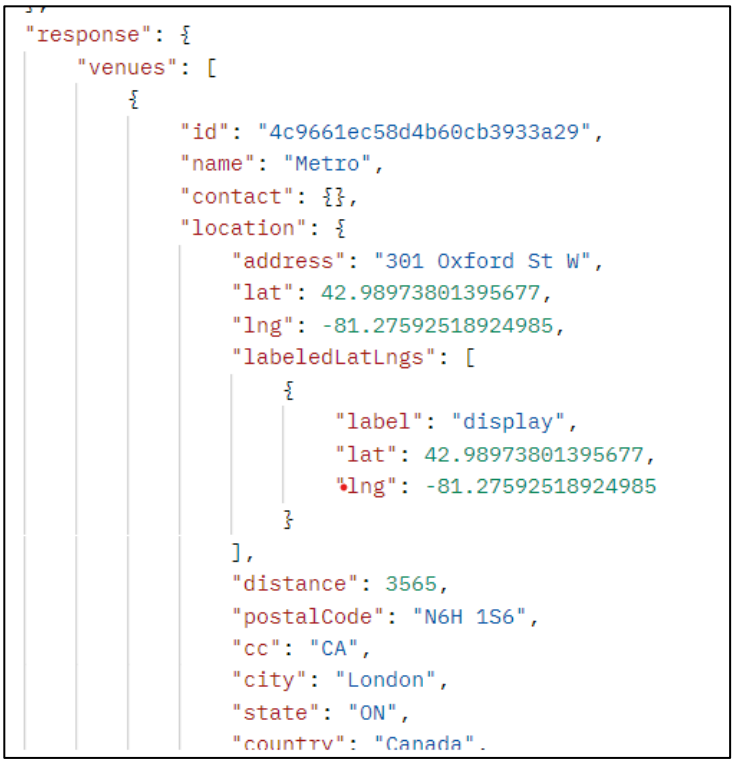

Figure 13: FourSquare Places API response

5.2.3.3 Geocoding API

An Android API service supported by google to handle the geo-location and maps events. Geocoding refers to the process of converting addresses into geographic coordinates (latitude and longitude) [41]. The conversion of these coordinates into human readable format is called Reverse Geocoding [41]. The API supports both conversions. In addition to these, location detection through postal code is also provided. For the implementation of SmartCart, Geocoder class and its method for finding address through coordinates and postal code are used.

getFromLocation(latitude, longitude, maxresults, geocoderListener)

getFromLocationName(postal_code, maxresults, geocoderListener)

5.2.3.4 Firebase Cloud Messaging

Firebase Cloud Messaging (FCM) is a cross-platform messaging solution that allows sending messages at no cost [42]. The implementation of FCM includes two main components. An application server to build, target, and send messages, and a client app, usually an android, apple, or web application, receives the information. FCM allows the server to send messages to individual devices running the application or multiple devices simultaneously. Firebase leverages topics to send messages to various devices. Firebase generates an encrypted token whenever a device installs the application and uses this token to

identify the devices. Devices can be subscribed and unsubscribed to any topic at any time, and the messages are sent only to the devices that are subscribed to that token. FCM, along with topics, are used to send push notifications to the user about any changes in the products.

A project is created in the Firebase console and integrated with the project in Android Studio. The cloud messaging settings are enabled to create a server key. This key, along with the sender ID, is used to configure the notifications in the MongoDB server.

5.3 MongoDB

A detailed overview of MongoDB and its integration with the proposed solution is presented in this chapter. The official documentation of MongoDB describes it as a complete development platform for data-driven applications [43]. It offers a cloud-based, faster, reliable, and scalable platform for web and mobile use cases. It is primarily a NoSQL document-based database that extended its services to DBAAS and BAAS. MongoDB Atlas provides a multi-cloud support database management system, and Mongo DB Realm offers services for seamless and serverless application development. SmartCart is developed using Atlas and Realm to support the database and application hosting requirements. The integrated features, along with their functionality, are explained below.

5.3.1 MongoDB Atlas – Database as a Service

MongoDB Atlas is a fully managed cloud database management offered as a service. Provisioning, deployment, maintenance, and upgrades to the database are managed by atlas, thus reducing the amount of time and resources spent on just the database component. It is hosted on AWS, Google Cloud, and Azure and is accessible in 80+ regions [44]. This allows the application to be accessed from anywhere. It offers built-in fault tolerance and data recovery mechanism to make the database more reliable. A minimum of the three-node replica is distributed on the availability cloud zones and are pulled into the application in case of any unexpected failures [45]. Secure database management is guaranteed with integrated end-to-end encryption and role-based access. It reduces the operation cost by allocating resources based on the requests to the system. This is achieved through auto-scaling by supporting both horizontal scaling and vertical scaling. Atlas provides a web-based UI to simplify the cluster configuration. A sharded cluster is deployed to handle the horizontal scaling. The developer has complete access to create a sharded cluster and decide the number of shards. Placement of replica sets, distribution of the queries, and handling of the write operations are entirely taken off the developer’s hand. Vertical scaling is reduced to a cluster configuration based on usage.

Data is stored as collections in the Atlas cluster. A database is created using the Atlas UI offered by MongoDB., and collections are then created to store data in the form of documents. A free version of the basic cluster is used to deploy SmartCartDb. The cluster configuration is shown in figure 14 and the sharded cluster information is displayed in figure 15.

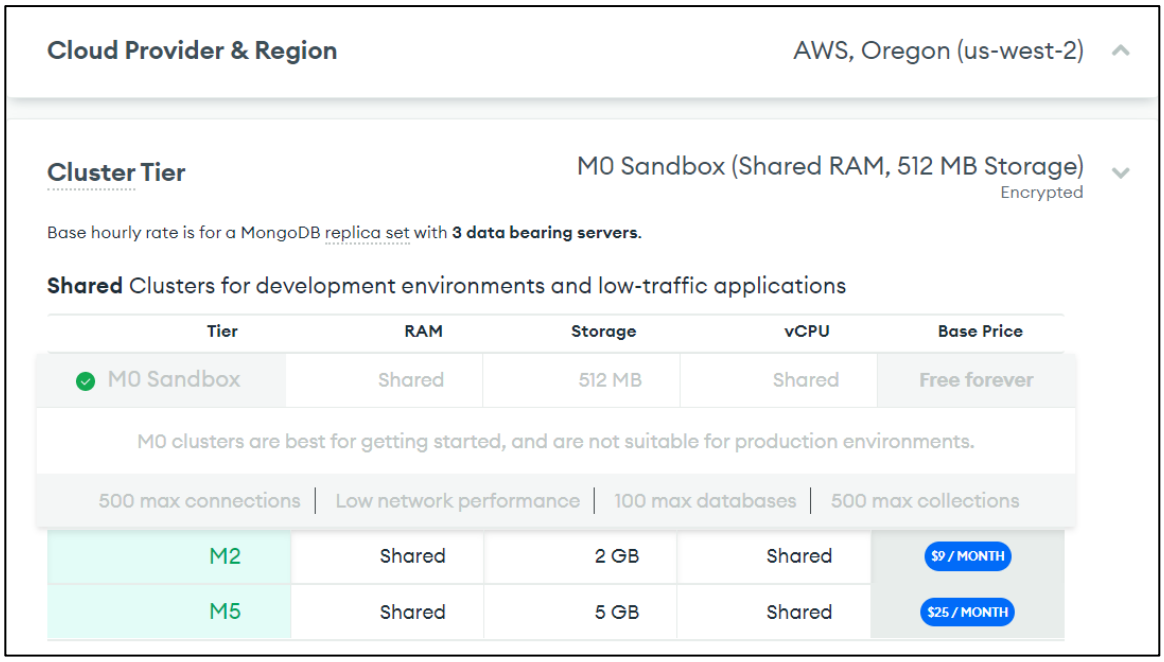

Figure 14: Cluster Configuration

Figure 15: Three replica set of sharded Cluster

The database is comprised of four different collections. UserCollection is assigned to store all the personal information of the user attained via the application. ProductCategory is a list of documents with product names along with their category specifically created to assist the search functionality. The StoreCollection contains the name, address, and postal code of the stores involved with the application. ProductandStoreCollection is the heart of the application, which is comprised of products with their pricing in that store. The schema of all these collections is shown in the following figures. Only the admin has configured the access to modify the Product, Store, and ProductandStore collections whenever

needed. The general schema of the documents in each collection is shown in the below figures. Fig 16 ProductCategory Collection, Fig 17 ProductandStore Collection, Fig 18 Store Collection, Fig 19 User Collection.

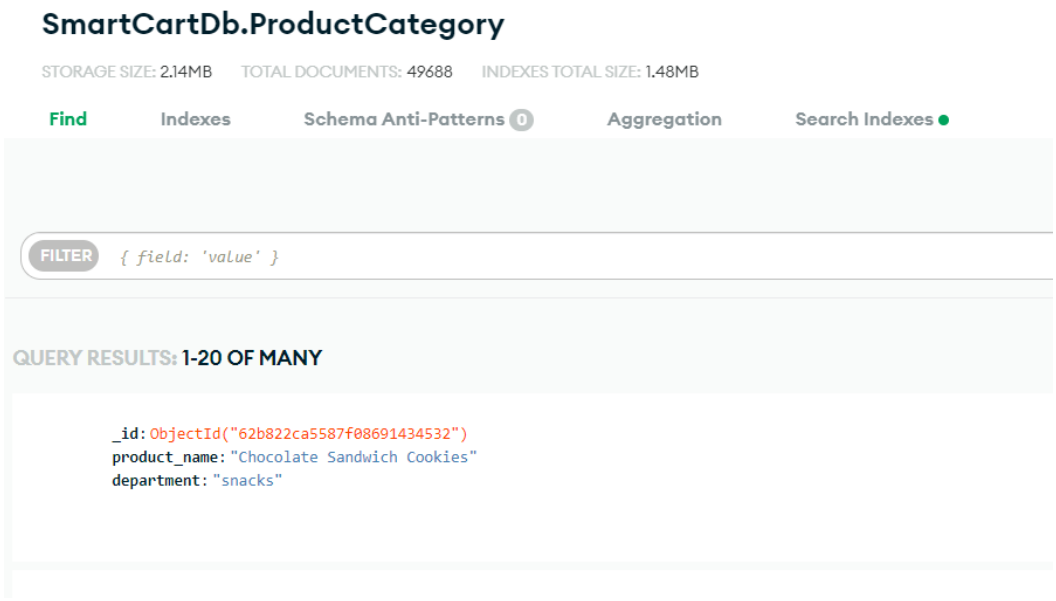

Figure 16: Atlas Product Collection

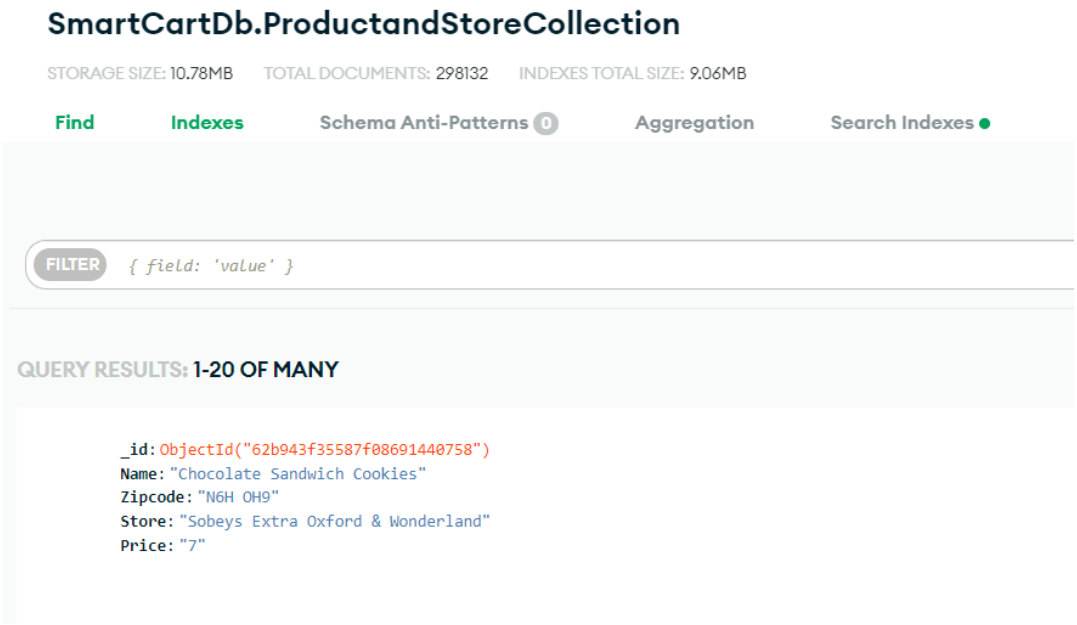

Figure 17: Atlas ProductandStoreCollection

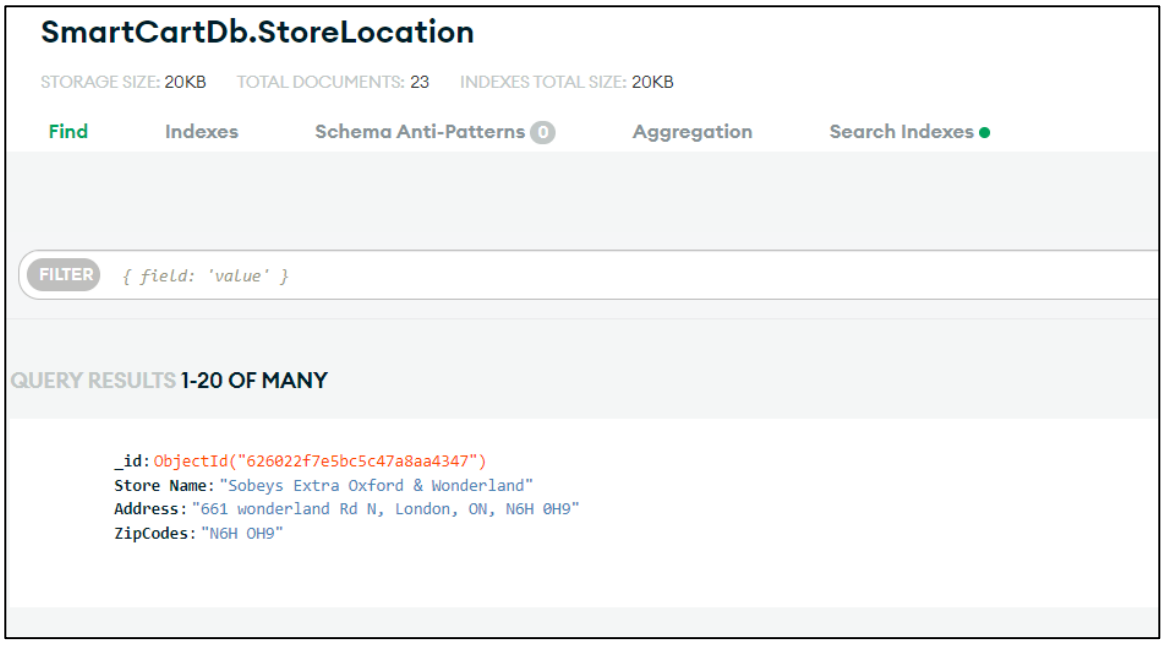

Figure 18: Atlas Store Collection

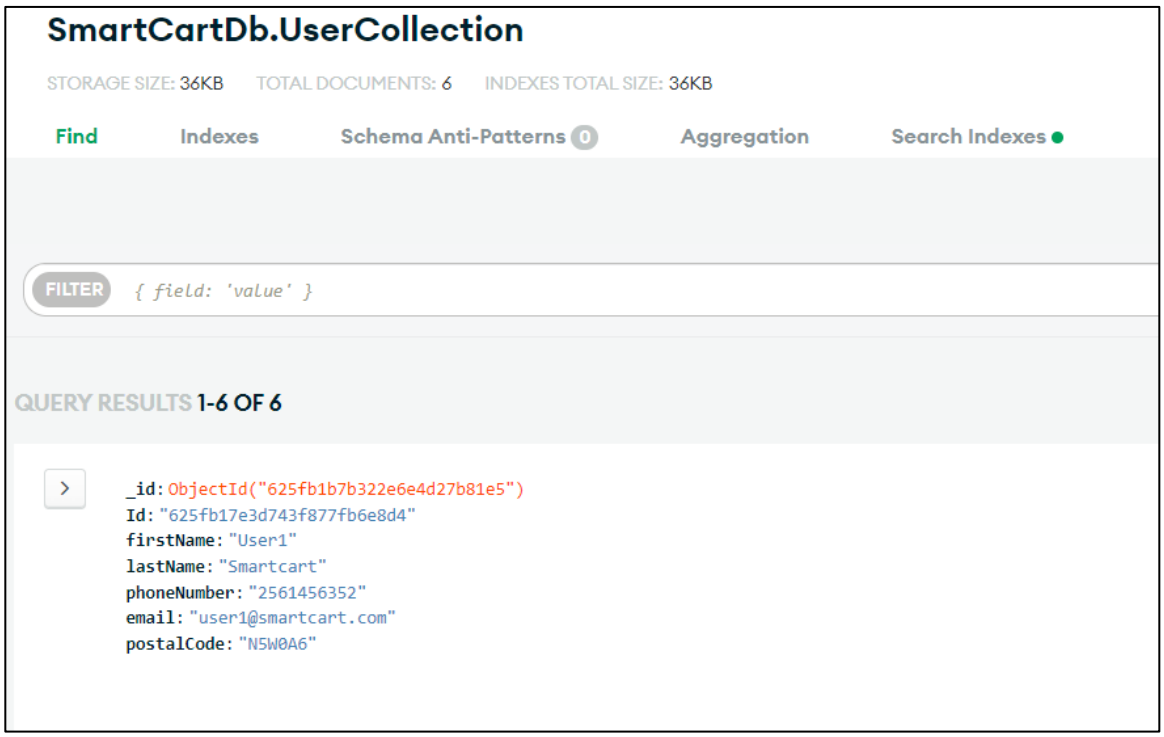

Figure 19: Atlas User Collection.

A distinct collection named savedLists is created and used to store information about the shopping carts made by the user. This collection contains all the shopping lists saved by the user and information can be used to understand the user behavior, identify popular products, and create a recommendation system. A snippet of the collection is shown in figure 20.

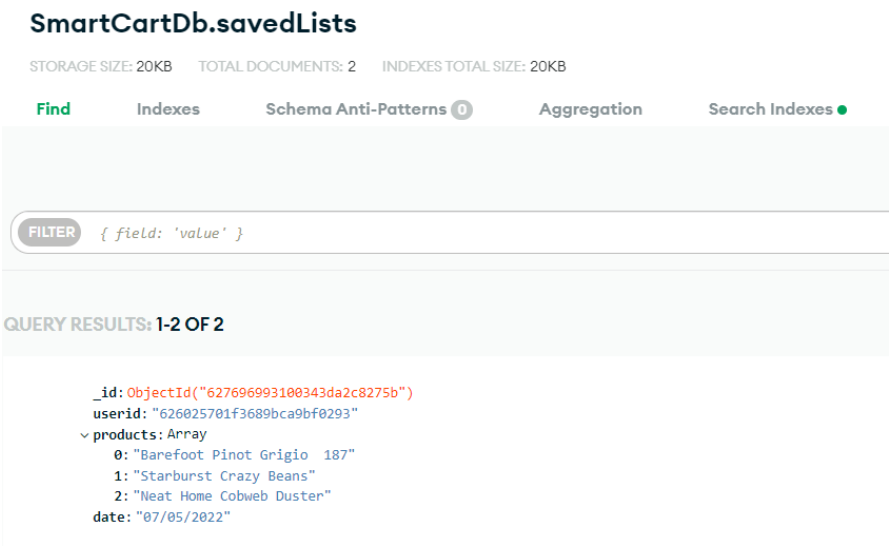

Figure 20: savedLists Collection.

5.3.2 MongoDB Realm – Backend as a Service

The official documentation of MongoDB Realm describes it as a complete development platform for data-driven applications [46]. It acts as a backend service and offers application-level services to build and deploy mobile and web-based applications at a faster rate. Some of the offered features include User Authentication, Realm Sync, Schema, rules, etc. Realm UI is offered to configure and access these services according to the application's needs. It is an Offline-first model which allows the application to be available even in offline environments, a valuable feature for the development of mobile-based applications. A Realm app is configured for a selected cluster to be used along with the database. The creation of the app is shown below in Fig 21.

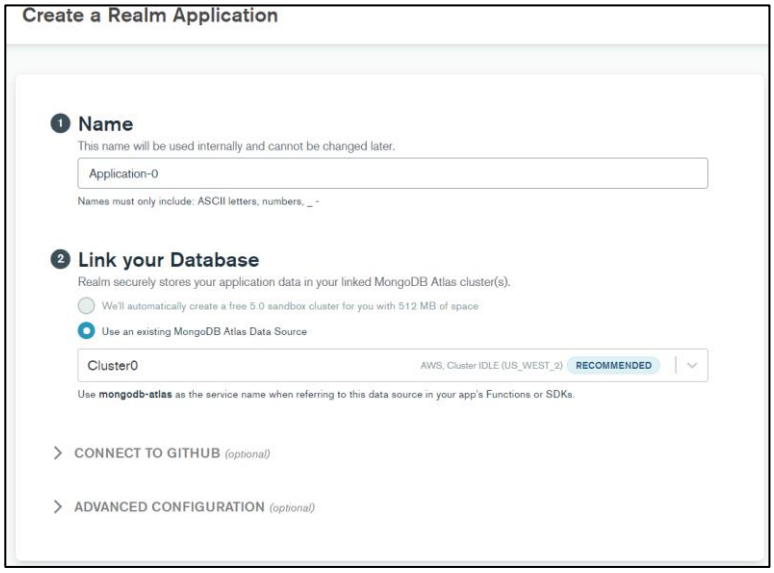

Figure 21: Realm app creation

5.3.2.1 Realm Sync

It synchronizes the data on client devices with its counter source on the Mongo database. This helps to maintain the consistency of the data across different devices. Realm Sync handles all the network-related access and conflicts as a background thread without interfering with the application logic. It supports the offline-first approach by sending the local changes on the data to the server. The Live Objects feature of sync ensures that data updated through one device is reflected across all the related devices.

5.3.2.2 User Authentication

Realm offers identification for an existing user and registration and authorization for a new user as a part of its authentication service. This service handles user access to the database and any requests against the database. Users are required to provide a set of credentials to authenticate themselves. Realm supports several built-in authentication providers such as Anonymous users – allows access to anyone and everyone; email/password – a valid email and password are used as login credentials and API keys. In addition to these, OAuth through users' Facebook, Google, and Apple ID is also supported.

A new realm user is created whenever the provider validates a unique identity, and the login requests followed are verified against this user. Trivial information such as name and mobile number can be associated with this user by enabling the Custom data feature. A collection can be configured to store user-specific information and retrieved along with the user object by mapping an attribute in the collection with the realm user Id field. This allows unnecessary database calls to fetch user-related information. The data can be accessed via the getCustomData method.

5.3.2.3 Schema and Rules

Schema refers to the general structure of the data rather than the values. The realm supports several primitive schema types such as string, int, arrays, and even objects. The defined schema validates the objects and maintains the sync between different devices.

Rules are used to restrict access to collections and prevent unauthorized reads and writes. They are configured for the entire collection and applied during the dynamic execution of the query. Different levels of access are provided to users based on the roles.

5.3.2.4 Push Notifications

MongoDB supports integrating mobile apps with the Firebase Cloud Messaging (FCM) service to provide push notifications [47]. FCM credentials (Sender ID and Legacy API Key) must be configured in the config tab of the Push Notifications screen. MongoDB server can access the Topics created using

the FCM libraries and are being used to send messages to subscribed users. MongoDB configuration to integrate firebase is shown in figure 22 below.

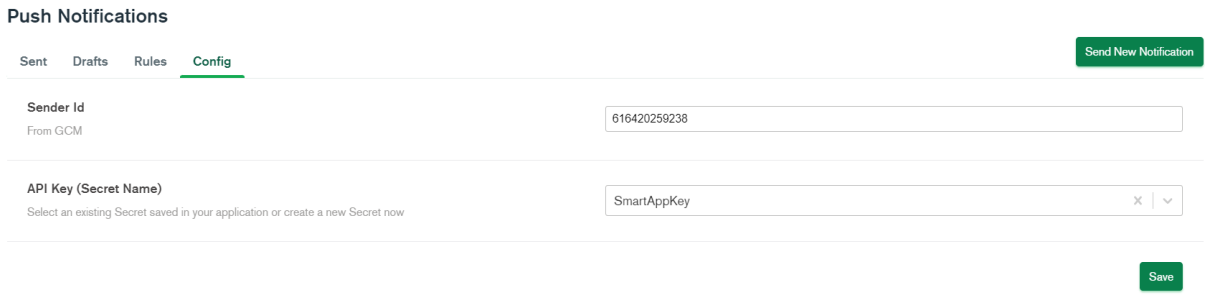

Figure 22: Firebase Notification Configuration in MongoDB.

5.3.2.5 Triggers and Functions

Triggers are created to respond to events, use pre-defined schedules, and execute configured logic [48]. Triggers continuously listen to the collection they are configured for and fire the linked function when the event occurs. There are three kinds of triggers supported by MongoDB Database, Authentication and Scheduled triggers. A database trigger responds to document insert, changes, or deletion. The change event is passed as an argument to the linked function and can be used to customize the messages.

Functions are a piece of server-side JavaScript code written to define the application's behavior. These functions can be called directly from the client application or the server. Functions and triggers are collectively used to create a notification service that sends messages to users whenever a product is added.

SendFCM function is written to build the notification. It contains the sender list and message. The function contains the configuration to call the firebase service to send the message to its recipients. A database trigger is created on the ProductandStoreCollection and is configured to fire whenever a new product is added to the collection. The trigger is then associated with the SendFCM. The newly added document is then passed as an argument to the function, and a custom message is built using that data. The configured trigger and the function are shown in figures below.

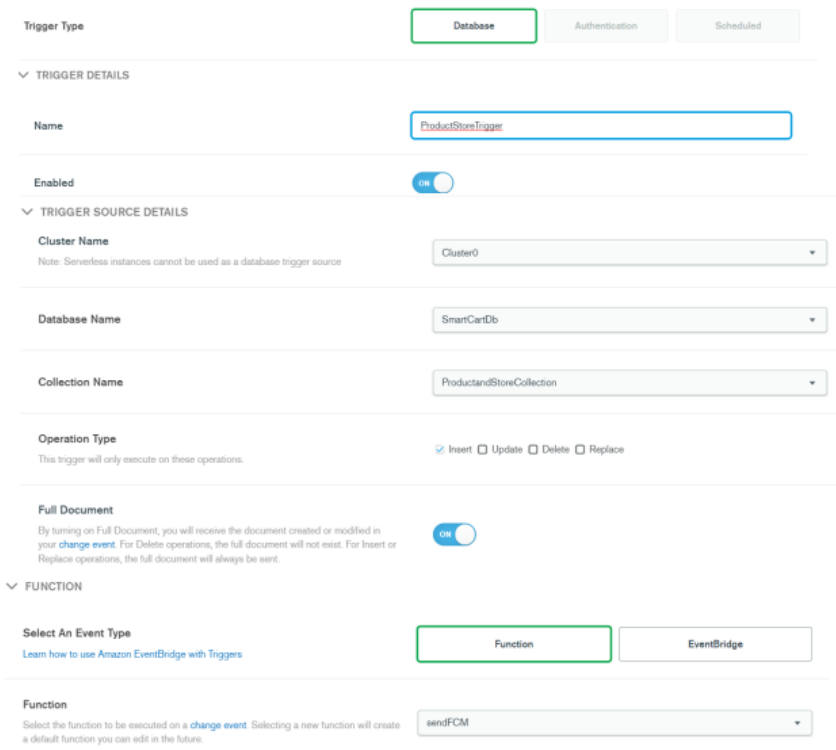

Figure 23: Database trigger to register changes to ProductandStoreCollection.

```
exports = function(changeEvent){
  const {fullDocument } = changeEvent;
  const { Name, Zipcode ,Store} = fullDocument;
  let tt = Name+" "+Zipcode;
  tt = tt.split(' ').join('_');
  const topicName = "/topics/"+tt;
  console.log(tt);
  const togroup = topicName;
  const message = {
    "to":togroup,
    "notification": {
      "title": "Product Available",
      "body": Name+" is currently available at "+ Store+"."
    }
  };

  // Send push notification
  const gcm = context.services.get('gcm');
  const result = gcm.send(message);
  console.log(JSON.stringify(result));
  return result;
};
```

Figure 24: SendFCM function.

5.3.3 MongoDB Integration

MongoDB has open-source SDKs for all the popular programming languages. Java SDK provides APIs to create, and access on-devices databases and for connecting to realm backend [49]. The complete process of integrating Atlas and Realm into SmartCart for backend service is explained.

Realm App is created and configured to use the existing Atlas cluster as the data source. The App can be accessed via AppId provided on the Realm UI. Email/Password as the authentication provider and UserCollection for custom data are configured for this App. Rules for UserCollection allow users to read

all data while writing only their data. Other collections are restricted to read-only as default for any roles. Any interactions with the database are directed through the realm app. Android Studio builds the app configuration and accesses it using the provided appId string. The following figures show the defined configurations for SmartCart. Fig 25 App Id , Fig 26 User Authorization Configuration Fig 27 Custom Data Configuration, Fig 28 Collection access rules configuration.

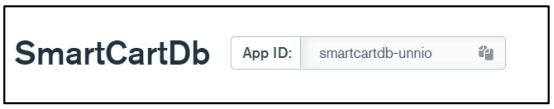

Figure 25: Realm AppId

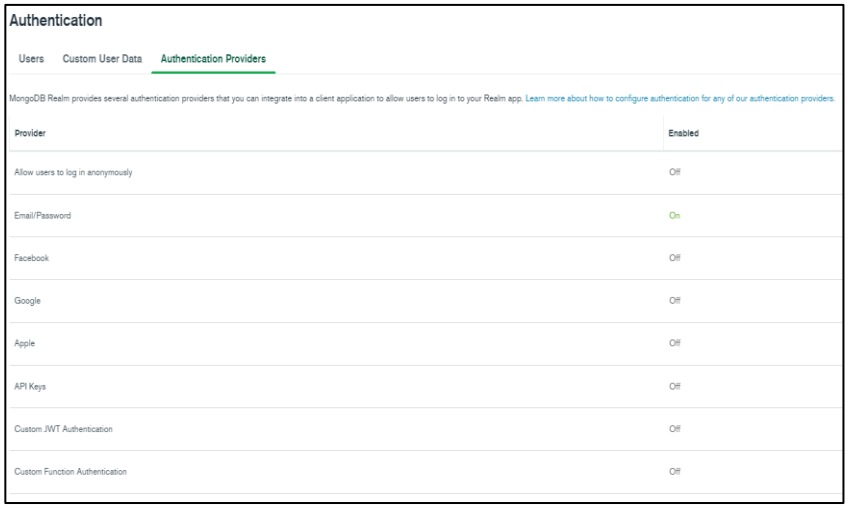

Figure 26: User Authentication Configuration

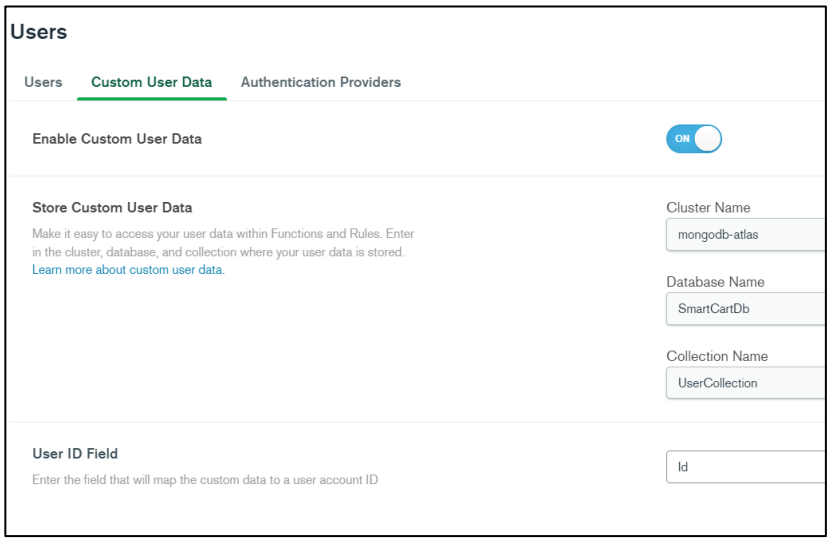

Figure 27: Custom Data Configuration

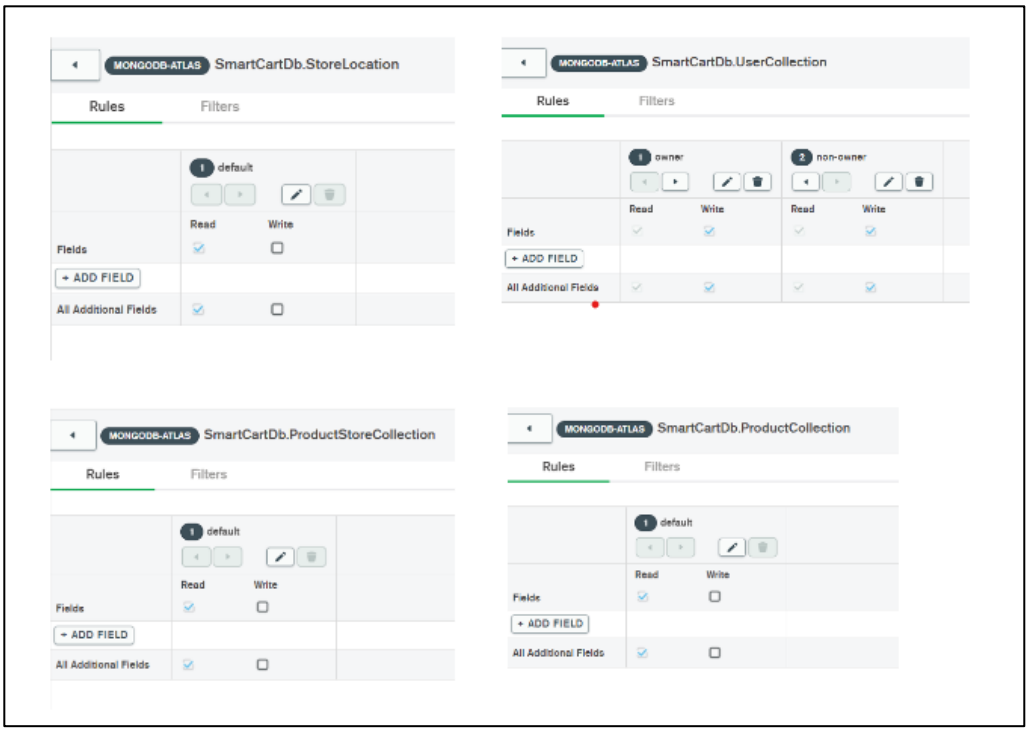

Figure 28: Collection access rules configuration

5.4 Application Implementation

The functional implementation is presented in this chapter. Section 5.4.1 describes the Java Classes used in the project. The Application class is described in Section 5.4.2. Login and sign-up are discussed in Section 5.4.3. The location service and methods are explained in Section 5.4.4. The home screen implementation, including the store finder code, is presented in chapter 5.4.5. The search functionality and the actual methodology for calculating the price are explained in Section 5.4.6. The final ouput screen to the user is explained in Section 5.4.7. Section 5.4.8 talks about the cart and profile activities of the application. The store activity which contains information about the selected store is discussed in Section 5.4.9 and the notification service is explained in 5.4.10. The Shared preferences of the application are managed through a Preference Manager. Section 5.4.11 explains the Preference Manager in detail.

5.4.1 Java Classes

Classes and objects are used to define the attributes and methods to access the data from the Atlas Collections. Serialization, and conversion of data into byte stream is necessary for storing the documents in shared preferences and to share them across different screens. Serializable classes following the common structure of the document in a collection are created for Product, Store and Product Store. These classes are defined to contain the additional information required for the functioning of the application.

A special DisplayData class is created to store the results of the tool. The class diagram below figure 29 shows different classes implemented in the application and their attributes.

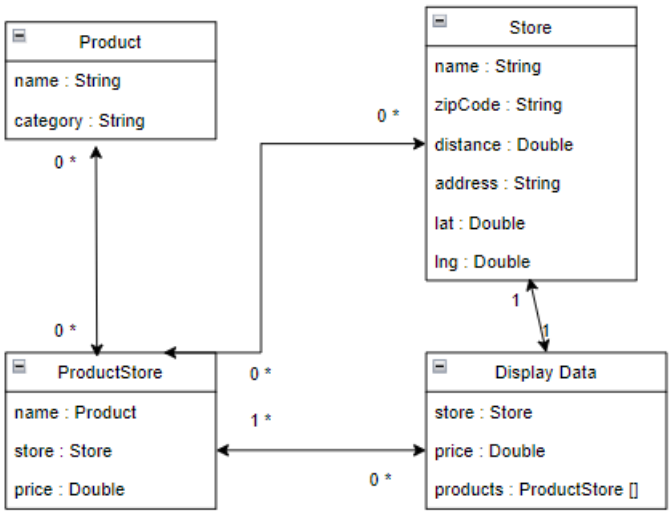

Figure 29: Class diagram

5.4.2 Application Class

The realm object needs to be initialized only once for every application cycle to access the app and its services. Android context needs to be provided to initialize it. The context can be of activity, fragment, or application; the Android application class ensures that only one realm object is created and used all over the code. So, a RealmApplication java class is created and added to the manifest.xml file. The application OnCreate() is the first code to run and thus guarantees that the realm object is instantiated and made available for accessing the services and storage. The following code is used for initialization.

```
Realm.init(this)
```

Additional build scripts and plugins are required to support the integration of the MongoDB realm into the android studio. The following changes (fig 30) are included in the Gradle build files for the same.

```
buildscript {
    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath "com.android.tools.build:gradle:7.1.2"
        classpath "io.realm:realm-gradle-plugin:10.10.0"
    }
}

apply plugin: 'realm-android'
realm{
    syncEnabled = true
}
```

Figure 30: Android Studio Realm dependencies

5.4.3 Login and Sign-Up

Login and Sign-up are the necessary functions for any mobile application. It helps in identifying and presenting the custom information of the user. SmartCart uses the Authentication service offered by the MongoDB realm for authentication and authorization. The email / Password identification provider is integrated into the application. The user is expected to provide a valid and existing username and password for a successful login. An object of the Credentials class is created with the given inputs, and an asynchronous request is sent to Realm API.

Login Activity is responsible for the login interactions of the user. Two fragments, LoginTab and SignUpTab, are implemented for the activity and are being managed by LoginAdapter. Existing user events are handled by LoginTab and new users by SignUpTab. LoginTab takes username and password as user inputs and validates them against the existing user data through LoginService. Sign-up is divided into two activities; SignUpTab handles the username validations, such as checking for existing usernames. Register Activity handles collecting personal information such as name, phone number, and postal code. This data is being stored as custom user data. MongoDB Realm allows the customer data to be associated with the realm user.

LoginService is responsible for all the interactions with the realm app. It contains methods to log in and register a new user. A new App object is configured to connect to the realm app using a string appId provided by realm. Code is implemented to catch errors and display the appropriate message to the user. On successful user creation and login, the user is prompted to Location Activity. The UI of Login

functionality is shown in figure 31. Realm provides an APK for smooth integration with android [50] and the application is connected to realm backend through the following app configuration.

App app = new App(new AppConfiguration.Builder(appID).build());

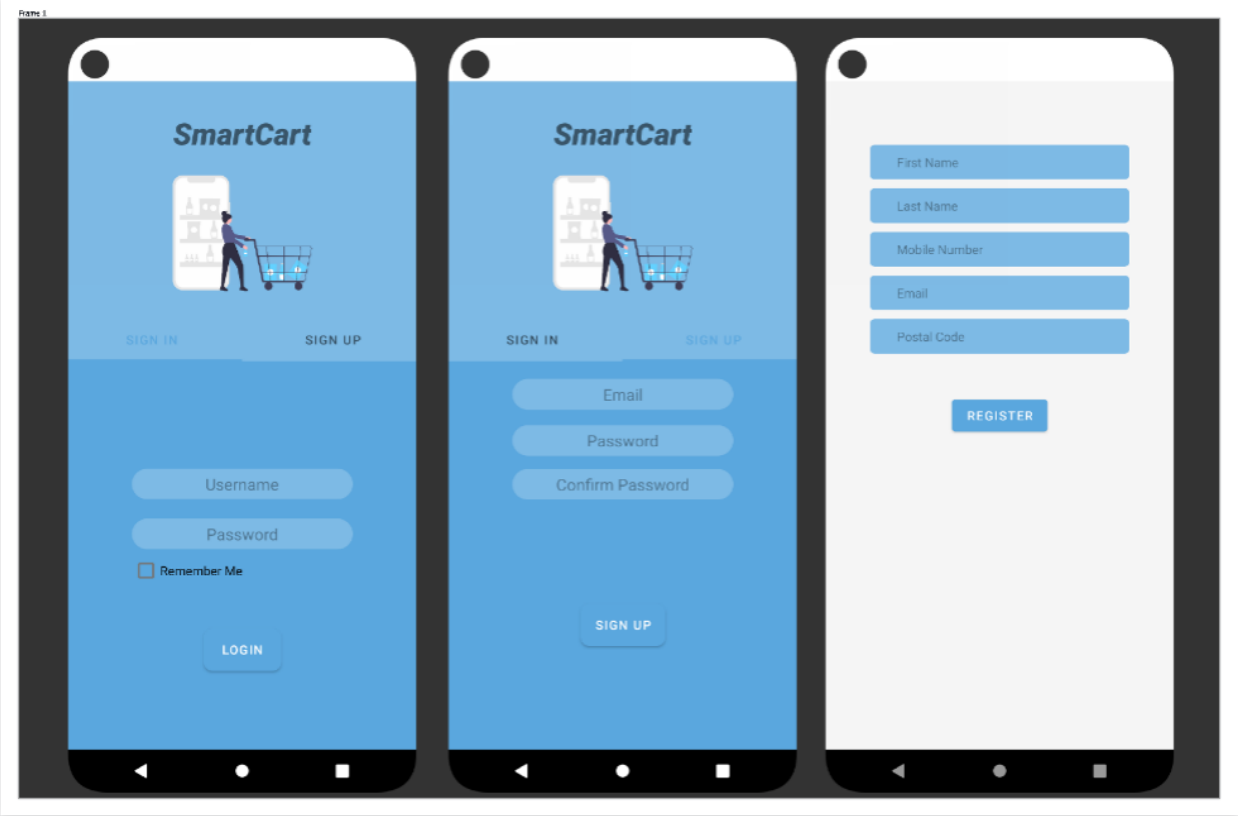

Figure 31: Login and Sign-Up

5.4.4 Location Detection

A successful login prompts the user to the location detection screen. Automatic location identification through a ‘Find Location’ button and a text field to enter a postal code are the two input options available to the user. The first method uses Android’s LocationManager class to find the location’s latitude and longitude values. Out of all the available providers, GPS_PROVIDER and NETWORK_PROVIDER are incorporated into the tool.

The android device allows mobile applications to access its GPS and network information if the user provides access to it. SmartCart checks for necessary permissions and requests the user if they are not provided. The permissions requested for location access are as follows. The ACCESS_INTERNET, to connect to the internet, FINE_LOCATIONS is necessary to access the network and GPS providers, while COARSE_LOCATION is responsible for the network provider. The following permissions are added to the manifest.xml file

<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

The getllastknownlocation provides the latitude and longitude values of the current location. Geocoder is used to convert the values into a readable address regardless of the selected method. It supports deriving the address from latitude, longitude, and postal code. The result of a successful request is displayed on the screen for further confirmation, and Toast messages are displayed for observed errors when required. Continuous requests to the location manager are restricted by disabling the UI elements. The address, latitude, and longitude values are stored in shared preferences before proceeding to the home screen. The UI design is shown below in figure 32.

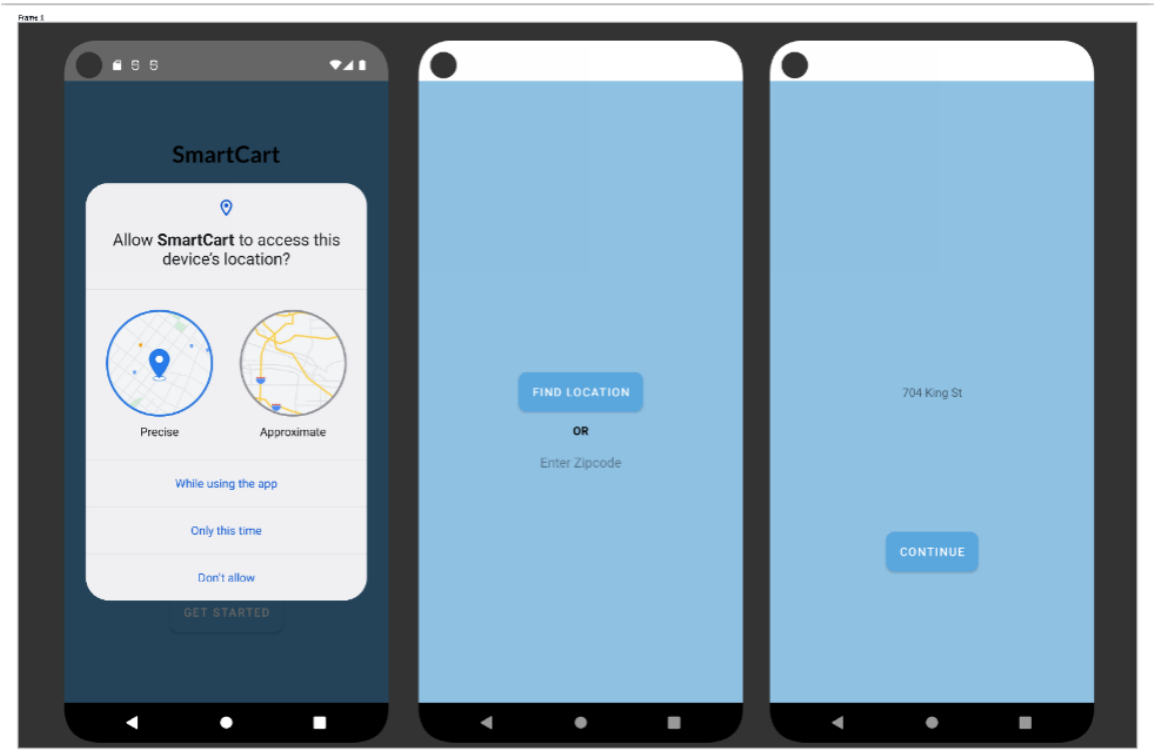

Figure 32: A - Location Permission Request B- Location Detection and Display

5.4.5 Home

Home activity acts as the main screen for user interactions and provides options for navigating to other features of the application. The screen is customized to display user-specific information such as their location and a welcome banner with the user's first name. The getCustomData method of realm user is used to get the user information. Previously selected cart items, if available, can be accessed via a cart

button at the top of the screen. A search bar is created to direct the search activity if all the constraints are met. The home screen is shown in figure 33.

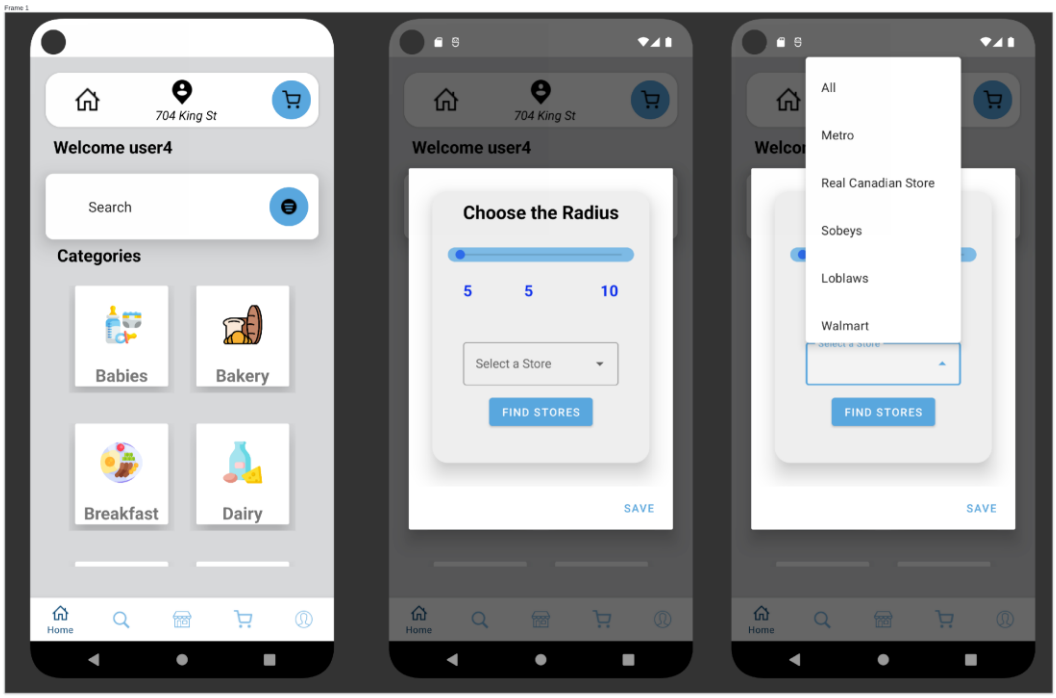

Figure 33: Home Screen and Location Filter

Customization is the primary function of this activity. The filter button allows the user to select custom parameters and values for the search method. Currently, the user can select a custom radius, and the tool will look for stores within that radius. An optional store selection feature is provided to the user where they can filter the nearby store of their choice. The OnClick() event on the button pops out a dialogue box with a seek bar to adjust the radius and a button to find the available stores for that radius. Users can change the radius until they find a considerate number of stores based on the textview, which displays the number of stores, and exit the dialogue through a save button. A request to FourSqaureAPI with radius, latitude, longitude and CategoryID as parameters is made. The request is designed to run as a background process to avoid overhead on the main Java thread. The response is captured and processed to store the venue details. MongoDB Atlas StoreCollection is queried asynchronously to load the details of the stores available to the application. The response of both requests is then compared, and the common stores are saved as nearby stores. The save button stores the radius and stores information in shared preferences. These values are considered for all the upcoming sessions until the user logouts or explicitly changes them. The details of nearby stores are required for the search activity; in case of no selection from the user, their access to search is denied, and they are redirected to the filter dialogue.

5.4.6 Search

Product search and price calculation are the primary functionalities of this activity. The click event to search activity queries the MongoDB atlas Product collection for all the product names. A progress bar is displayed to the user until the products are loaded to avoid null pointer and empty queries. Once the products are loaded, the search screen is presented with a search bar and button. The search bar is implemented as an AutoCompleteTextView to suggest the existing products in the form of a dropdown. The search button click provides the desired search results. A card is now showcased with the product name and an add button. Once added, the product will be displayed in the selected product list called Items. Users can delete a product anytime.

Once the required products are added to the list, the done event is executed by OnClick () button. A series of methods are executed after this event. The order of the events is as follows:

1. The application checks whether any stores are available as per the user's constraints. If not, the user is prompted to filter fragments to re-enter valid parameters for search.
2. A query is created to find the selected products in the products database using the '$and' and '$ or' operation of MongoDB.
3. The query is executed asynchronously, and on successful retrieval, the results are stored as a Product Store data object.
4. The current results need to be grouped according to the store in which they are available. A HashMap <String, ArrayList<ProductStore> is created to achieve the desired data format. The postal code of the store is being used as a key, and ProductStore objects with the same key are stored as value.
5. The HashMap is then iterated over the stores' list, and a new Object of class DisplayData is generated. The total price is generated in the same loop and stored as the Price variable of the object along with a store and list of ProductStore objects.
6. As the final step, the user is directed to the Map Activity screen, where the price is used to mark the store's location on google maps.

An additional feature of adding all the selected items into the cart is provided to the user as an Add to Cart Button at the lower end of the screen. The complete search functionality is shown in figure 34 and figure 35.

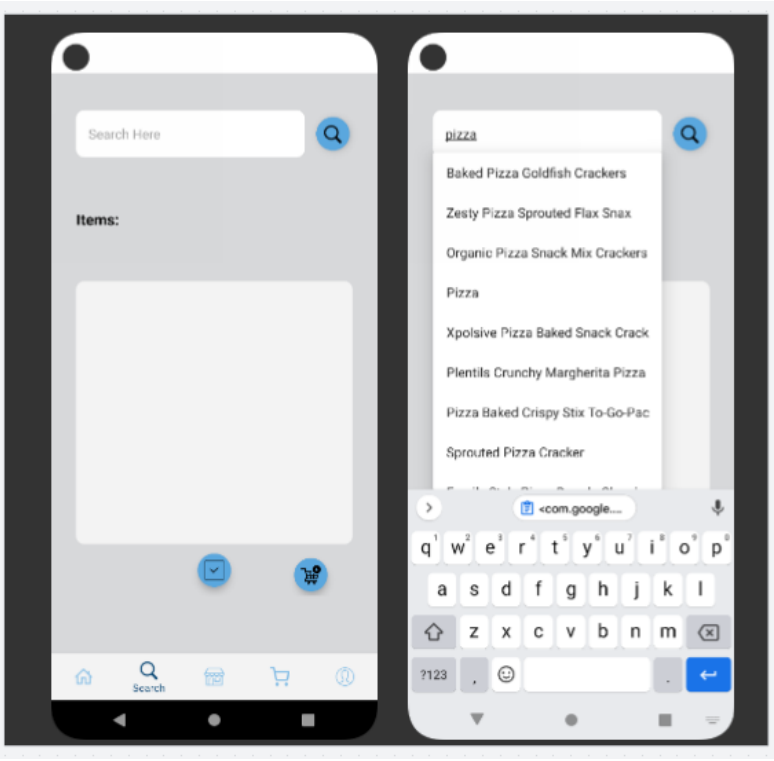

Figure 34: Search screen enabled with AutoComplete

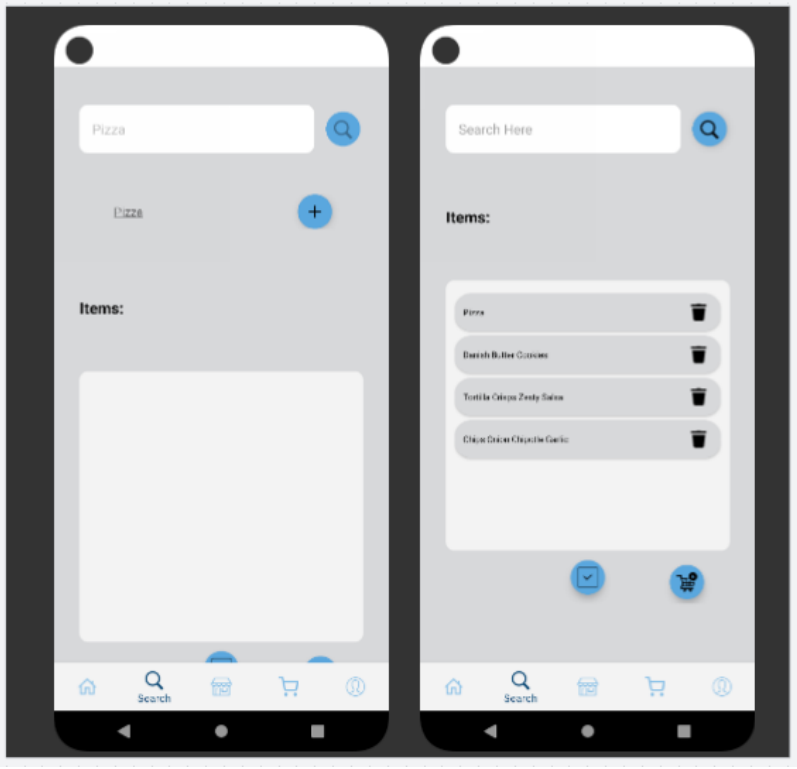

Figure 35: Product Search, Select and Add

5.4.7 Map Results

Android supports the integration of maps into its lifecycle through an inbuilt map activity. Google map is loaded as a fragment inside the activity. Google Maps API key and maps utility dependency are included in the project code to access and customize the maps [51]. Android intents are used to pass the display information to the map activity from the search. Back navigation to search activity is added on the main screen, along with an option to add products to the cart. Meta data with API key to manifest file, google's library to support maps in android along with utils package for assisting in custom design are added to Gradle build.

implementation 'com.google.android.gms:play-services-maps:18.0'

implementation 'com.google.maps.android:android-maps-utils:2.3'

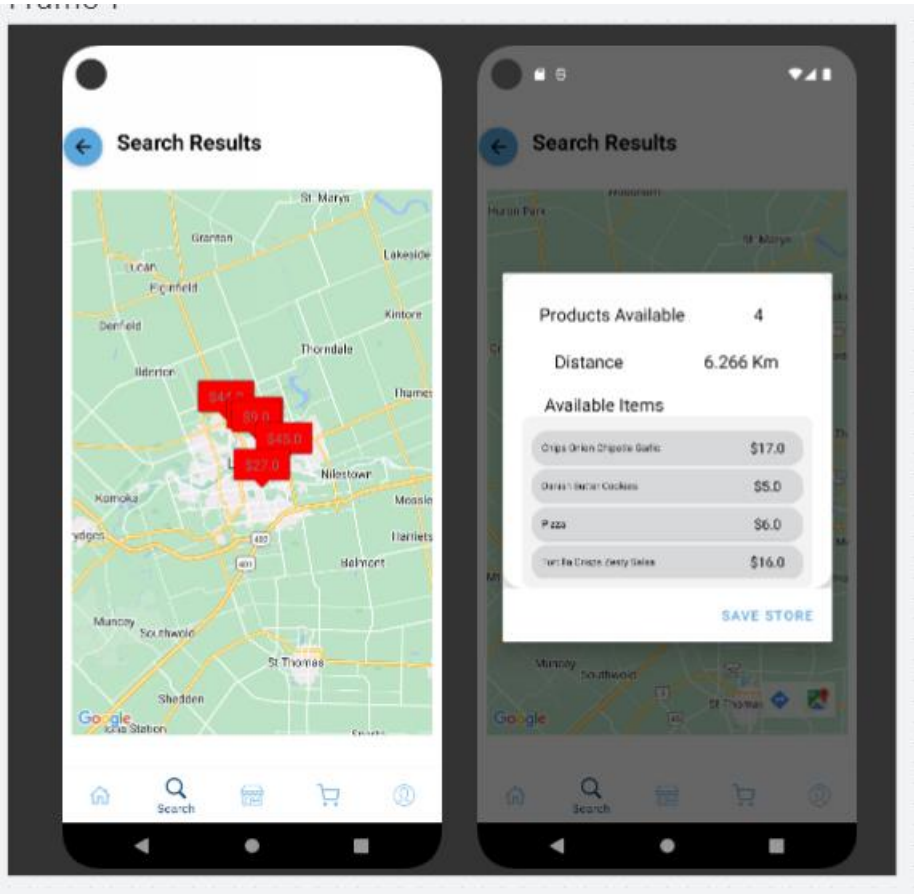

Figure 36: Results

The end result screen is shown in figure 36. The location is marked with the price and more information is presented to the user with a click event. A simple tap on the marker opens up a new display window for the user. This contains information about the number of products available in that store, the distance

of the store from the user's location, and an option to select and save this store for future use. The save store click event directs the user to a new activity.

5.4.8 Cart and Profile

All the selected products are saved in the shared preferences following the price calculation in the Search Activity. The products can be added to the cart through button click events in search and map activity. A cart list is designed to replace the traditional shopping list and contains all the products, irrespective of availability. An option to save the shopping list is also provided to the user. This data can be used to further improve the personalization of users.

The personal details of the user are presented on this screen. Allows the user to modify the information. The user's Custom Data is accessed via the "getCustomData" method of the logged-in user object. Users can exit the application using the logout screen provided in this activity.

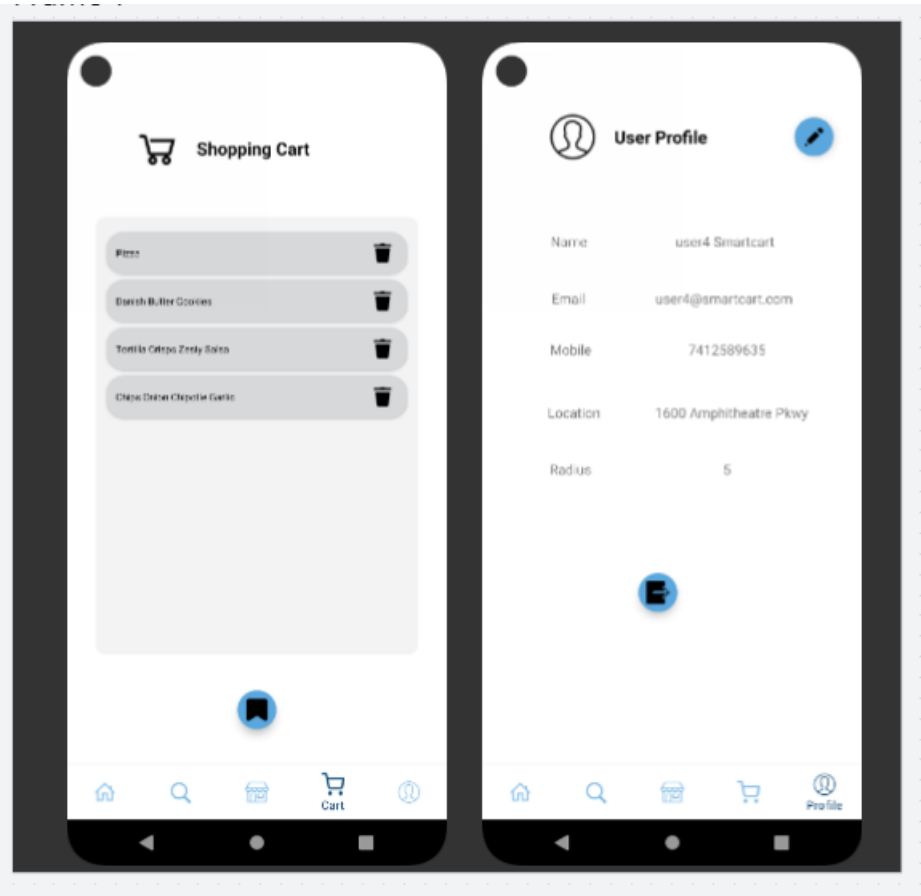

Figure 37: Shopping Cart and Profile Screens.

5.4.9 Store Information

Store activity is created to provide a means to keep the information of the store user selected until a new selection is made. The UI contains information about the store, such as name, location, distance from the user's given location, products available in that store, and the total cost for it. In addition, a special feature to receive notifications about unavailable products is provided to the user on this screen. Notify button on the screen allows the user to subscribe to the notification service. The figure below shows the store information screen.

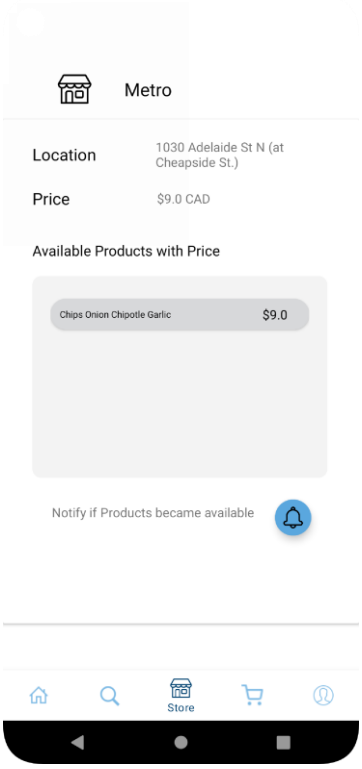

Figure 38: Store Information User Screen.

5.4.10 Notification Service

An exclusive notification service is configured to maintain communication with the users in their inactive state. The main functionality of this service is to notify the users about the availability of their desired products. In the real-world, a single store may not have all the products in its inventory, and the user inevitably ends up with some unavailable products. The notification service listens to the database collection and sends a message to the user when any product in their unavailable list is added to the store they selected. The following steps are followed to connect the application to the Firebase Cloud Messaging.

1. A new android app is created in the Firebase console, and the package name and SHA1 key from the android studio app is used to create it.

2. A google.json file taken from the firebase console is added to the project's app folder.
3. The following dependencies are then added to the build.gradle file at project and module level
4. A PushNotificationService class that extends the FirebaseMessagingService is created inside the project, and the following code is added to the AndroidManifest.xml file.

The triggered message from the SendFCM message can now receive the message and display it to the end-user. Users can subscribe to notifications on their own. The topic names are created in product_name_zipcode format so that it is easier to identify when a product is added to a particular store. The OnClick() notification button method subscribes the user to their product topic if it exists or creates a new one. The user is now subscribed to all the unavailable products and is notified whenever one of them is added to the collection. Firebase's “SubscribeToTopic” method is used to register the users to their topics. The final notification sent to the user is shown below.

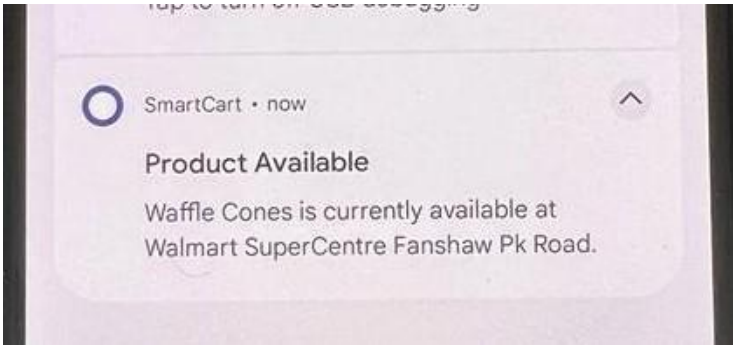

Figure 39: Notification message from the server.

5.4.11 Preference Manager

The navigation between the screens is restricted because of the dependency on the data from the earlier screen. For example, nearby stores information from the home screen is required by the search activity and should be made available throughout the application life cycle. Shared preferences are used to store globally needed information. The value of the radius selected, nearby stores, previously added items, cart items, and the latitude and longitude values are stored in the shared preferences. Android provides support only for the primitive data types such as String, Int, and Long to be saved in local storage. A preference manager class with methods to support the custom objects is created. The GSON library is used to convert the custom objects into JSON and then store them as a String.

GSON Dependency: ‘com.google.code.gson:gson:2.8.8’

5.5 Case Study: SmartCart

This section explains the workflow of the application from an end-user perspective and demonstrates the features of the application addressing the discussed research gap.

1. Interactive, User-friendly, and personalized interface:
2. Price Calculation and Comparison based on user inputs.
3. Easy-to-understand display of the results
4. Automatic creation and unrestricted access to the shopping list.
5. Notifications to the user.

5.5.1 Interactive, User-friendly, and Personalized interface

Visual and straightforward UI elements are utilized to make the interface interactive and easy to use, even for an amateur shopper. Most functionalities and features, such as product addition/deletion, are operated through buttons. Seek bar is used for radius selection, and information about the number of stores is provided right after choosing it as text so that unnecessary back and forth interaction between the end-result screen and store selection component can be avoided. Main application features can be accessed from anywhere with the help of the bottom navigation bar. User location and welcome banner with their first name is shown on the home screen to make it more personal for the user. The profile screen is designed to display and edit the custom information of the user. Figure 40 highlights some of the interactive UI elements.

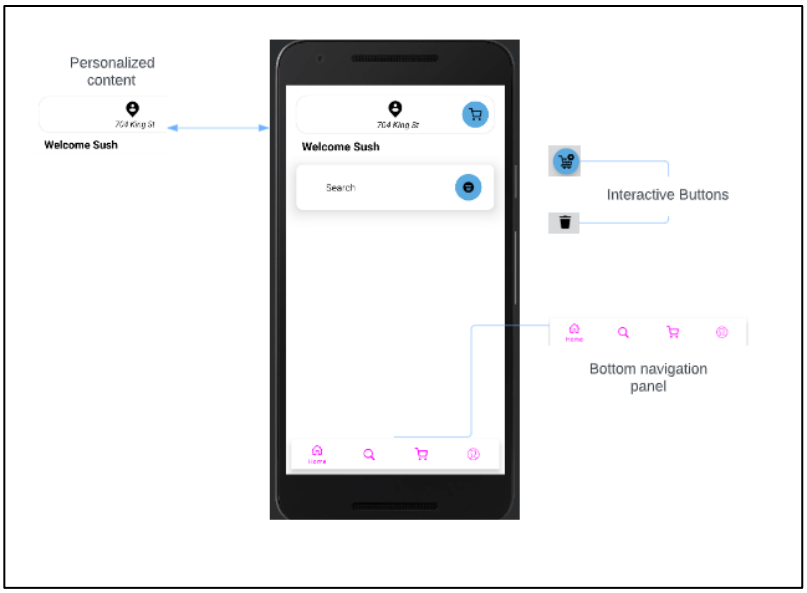

Figure 40: Interactive personalized UI

5.5.2 Price Calculation

The Home screen allows the user to adjust the search parameters of the screen. Toast messages are displayed to the user in the absence of required parameters. The user can search and add the products to compare the price all the once as a single activity. They can review and make additions/deletions to the

list with just one event. A click event presents the store results to the user. The user is responsible for selecting the search radius and the products. The application looks for the availability of products, calculates the cost, and provides the results for user comparison. The individual search and selection of products are avoided, thus saving time. The user is not expected to be knowledgeable of every available store as the tool is responsible for finding nearby stores. Effort and time spent are significantly reduced with these features. The UI workflow from user input to output is shown in figure 41.

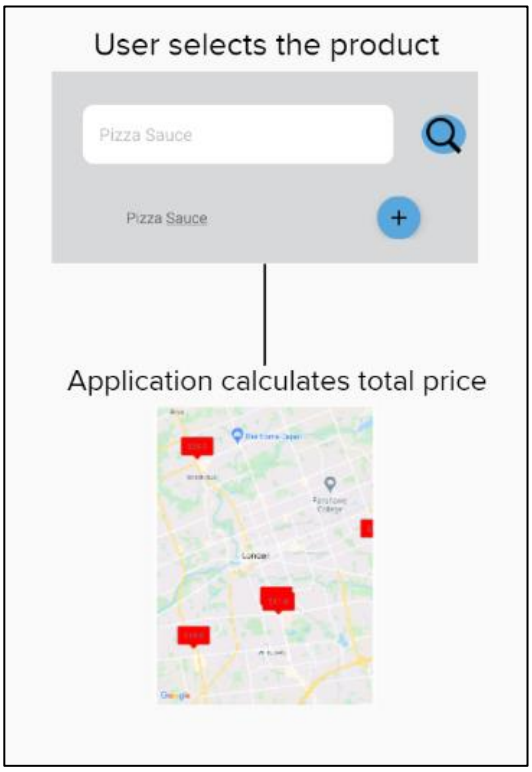

Figure 41: Application input and output

5.5.3 Easy-to-Understand Display of Results

With a single look, the user can identify the closest store, the store with the lowest price, and the total cost and marked location of all the stores. More details on the number of products available and address can be uncovered with a button click. Unnecessary navigation and interactions are avoided through this visualization technique, and just the critical data is pictured to the user. A detailed view of the results is presented in figure 42.

Figure 42: Visual display of results

5.5.4 Automatic creation and unrestricted access to shopping list

One of the essential features of any smart shopping application is to organize the selected items and make them accessible to the user. SmartCart collects the selected items and makes them available to the user through cart items. The list is strictly made from the selected items, irrespective of their availability. This assists the user with their in-store shopping. In addition, it provides a store information screen which contains individual lists of products available in the selected store.

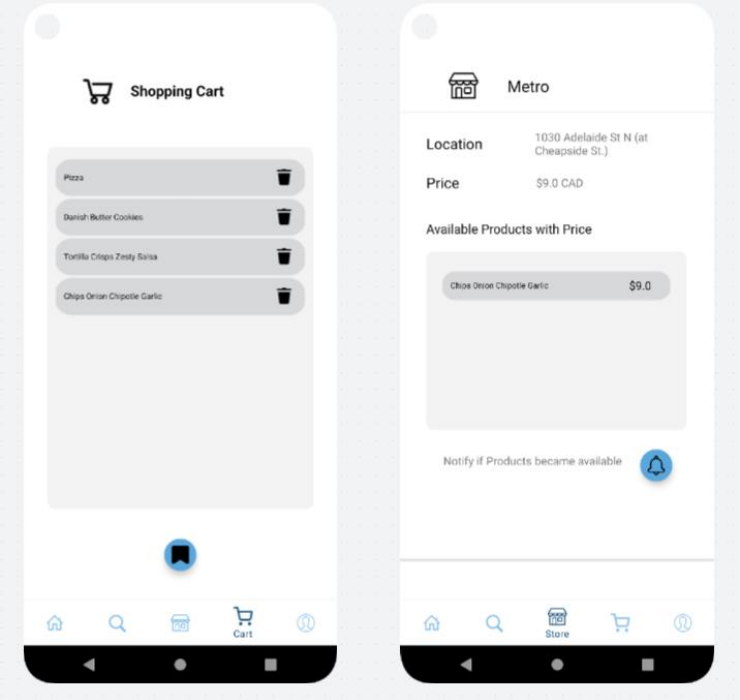

Figure 43: Store and Shopping Cart Screens

5.5.5 Notifications to the user

The consumer is provided with an option to get notified whenever an unavailable product becomes available in the store of their choice. A notify button at the end of the Store information screen is created for this purpose. A notification is sent to the end user right after the product becomes available. The notification along with the button is shown in Figure 44.

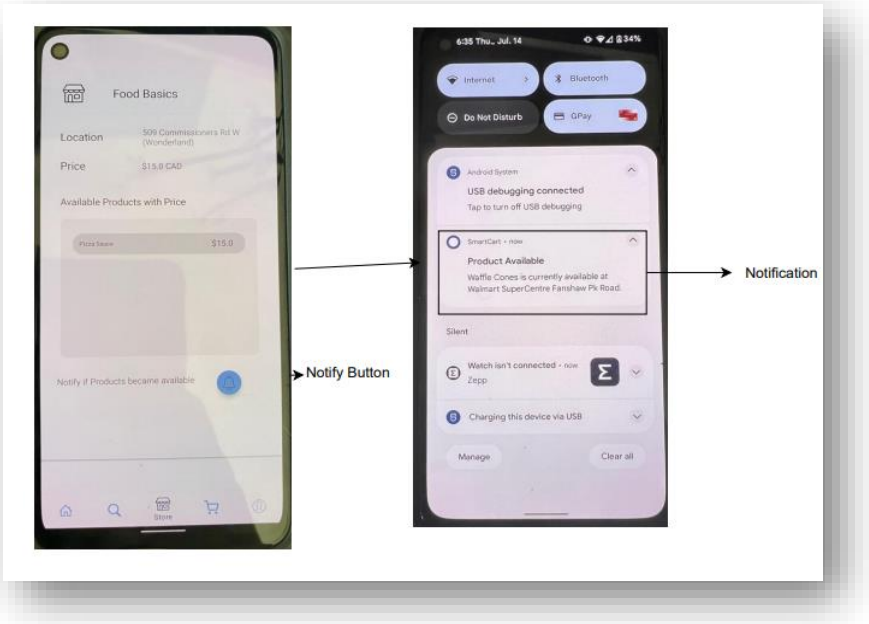

Figure 44: Notification workflow

Chapter 6

6 Recommendation System

This chapter explains the proposed recommendation system for SmartCart. The background information about various machine learning techniques and recommendation systems is presented in Section 6.1. The algorithms employed in implementing the recommendation system are discussed in Section 6.2. The dataset and its pre-processing are described in detail in Section 6.3. Section 6.4 explains the implementation of the recommendation system and the results.

6.1 Background

This chapter gives a brief overview of our proposed recommendation systems. An introduction to various technical terms and relevant algorithms and techniques are presented in this section.

6.1.1 Recommendation System

The accessibility of a massive amount of online information has created a data overload problem for a regular users [55]. It complicated the process of finding relevant and valuable content on the internet [55]. This inconvenience has created the need for a guided search engine that directs the user towards related information more easily, leading to the development of recommendation systems. In broader terms, a recommendation system is a filtering tool based on user preferences and necessities. They are being incorporated into most fields, such as entertainment in the form of book, movie, and music recommendations, e-commerce, gaming, and social media applications. In E-commerce, recommendation systems can be classified into personalized and non-personalized. Personalized recommendation focuses on suggesting and directing the user to the items or products based on their history and interests. In contrast, in a non-personalized recommendation system, all the users receive the same product recommendations. Recommendation systems greatly depend on the feedback or rating given by the user [56]. Explicit rating refers to the scenario where the user specifically ranks a particular item. For example, e-commerce sites collect star ratings from the user after purchase, which can be used to understand a user's preferences. However, it is not always possible to obtain an explicit rating. To overcome this issue, the recommenders infer the user preferences by observing various factors such as search and purchase history [56]. This kind of rating is known as implicit rating. Most practical and real-time recommendation systems work based on implicit feedback.

Based on the filtering techniques, recommendation systems can be broadly classified into three types: content-based, Collaborative, and hybrid recommender systems [55]. A content-based recommendation system focuses on the items and groups them into profiles based on their features. A user profile is built based on their liking of different items. The recommendation system then suggests to the users the items that belong to the same profile as the items in their profile. The major drawback of this approach is that system to limited to the user's past choices. Another disadvantage of these systems is that the differences between objects in the same group are not considered. A collaborative recommendation system measures the similarity between users and uses it in product suggestions [55]. This technique groups the users based on their preferences, likes, and dislikes, creating a neighborhood for that user. They create a collection of nearest neighbors [57]. These algorithms expand the user's interests as they can suggest products that the user never sees. Collaborative filtering can further be divided into memory-based and model-based approaches [55].

The memory-based collaborative approach uses a special utility matrix to recommend new items. A utility matrix is a user-item matrix created by considering the feedback of the user against each item. A user-memory-based collaborative approach calculates the user rating of a new item based on neighborhood user ratings for the same item. In an item-based process, an item-neighborhood consisting of similar items the user has previously rated is created [55]. The major setback for memory-based techniques is that only existing users can utilize the feature, as the model uses a utility matrix as input. Model-based methods use data mining and machine learning algorithms to predict the user's feedback [55]. These techniques build the model based on existing data and predict the ratings using the generated model and user profile as the input. They analyze the existing data using machine learning algorithms such as Matrix Factorization, Deep Neural networks, and dimensionality reduction techniques.

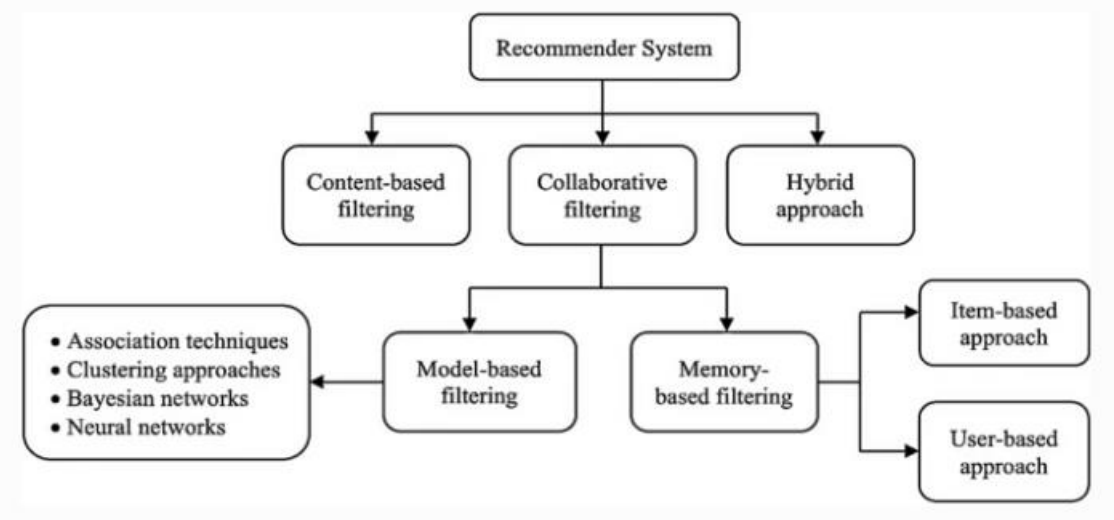

Figure 45: Types of recommendation system[55].

6.1.2 Matrix Factorization

Matrix factorization is a class of collaborative filtering algorithms that work by decomposing the utility matrix into the product of two lower dimensionality rectangular matrices [58]. They are used to discover hidden features, called latent features, based on the similarities in users' preferences and interactions [59]. There are various methods of matrix factorization and Singular value decomposition is used in the proposed recommendation system.

6.1.3 Hyperparameters

Hyperparameter is a parameter whose value controls the learning process [60]. These values determine the parameters for the final model. The values are set before the training process begins and cannot be changed during the training [61]. These parameters are not a part of the resulting model but help decide the model's structure. Some of the most commonly used hyperparameters are epochs, learning rate, regularization parameter, and the number of latent factors. Epochs refer to the number of times the training dataset is passed to the learning algorithm. This hyperparameter regulates the model from under and overfitting. The learning rate controls the changes to the model after each estimated error. A small learning rate can result in the model being stuck. Hence it is important to choose an optimal value. The regularization parameter controls the flexibility of the model and prevents it from overfitting.

6.1.4 Grid Search

Grid search is an optimization technique used to tune the hyperparameters. It gives the optimal value for hypermeters to result in an accurate model. Many libraries provide inbuilt Grid search functions, which can be imported and incorporated while training the model. A set of values is given for each hyperparameter; grid search finds the best combination of these values to build a precise prediction model.

6.1.5 Root Mean Square Error

The root-mean-square-error is used to measure the difference between the values predicted by a model and the observed values [62]. It is the square root of the difference between the predicted values and actual output values. It measures the accuracy of the models developed through various learning algorithms.

6.1.6 Surprise and Spark

Surprise is an easy-to-use python scikit library. It has many built-in machine learning algorithms suitable for recommendation systems. PySpark is an Apache spark interface for python. It has pre-defined

machine learning algorithms which aid in training the models. Both of these libraries are used in this thesis for the learning process.

6.2 Algorithms

This explains the background information about the learning techniques and algorithms that are used in this project.

6.2.1 Singular Value Decomposition (SVD)

The Singular Value Decomposition (SVD) is a common dimensionality reduction technique used in machine learning [63]. It is a matrix factorization technique that reduces the number of features of a dataset by reducing the dimensionality space [63]. It creates a matrix with users as rows and items as columns; the ratings determine the elements. It decomposes the matrix into three other matrices and extracts the factors from a high-level utility matrix factorization. The mathematical form can be shown as follows:

- U is the user matrix * latent factors.
- V is the item matrix * latent factors.
- S is the diagonal matrix showing the strength of each latent factor.

$$A = USV^T \text{ [63]}$$

The rating of an item by a user is given by $R\_u_i = X^T\_i * Y\_u$ where X_i is the vector for item and Y_u for user in the latent space. The loss can be minimized by the square error difference between the product of R_u_i and the expected rating.

$$Min(x, y) \sum_{(u,i)\in K} (r_{ui} - x_i^T y_u)^2$$

To avoid the overfitting of the data, regularization is used and a penalty is added to the dataset. A bias term is added to reduce the error [63]. The final equation with bias term and regularization term is as follows:

$$Min(x, y, b_i, b_u) \sum_{(u,i)\in K} (r_{ui} - x_i^T y_u - \mu - b_i - b_u)^2 + \lambda(||x_i||^2 + ||y_u||^2 + b_i^2 + b_u^2)$$

6.2.2 Stochastic Gradient Descent

Gradient Descent is one of the most common cost optimization algorithms in machine learning. Stochastic Gradient decent is a variant of the same algorithm designed to handle the large datasets. The In Stochastic Gradient Descent, a few samples are selected randomly instead of the whole data set for each iteration. The weights are updated accordingly as follows:

$$for\ i\ in\ range(m)\ :$$

$$w_j := w_j - \alpha \frac{\partial J_i}{\partial w_j}$$

J_i is the cost of ith training example [64]

The loss function of SVD can be optimized using this technique and an accurate model can be developed. SGD calculates the derivative of the loss function, but for each variable in the model. So, for each sample, get the derivatives of each variable, set them all to zero, solve for the feature weights, and update each feature. The features of the SVD lossfunction (L) are user bias, item bias, user latent factor, anditem latent factor. Therefore, SGD updates each feature with each sample. User bias (b_u) updates are done as follows. Similarly other features can be updated to minimize the loss function and get accurate results.

$$b_u \leftarrow b_u - \eta \frac{\partial L}{\partial b_u}$$

6.2.3 Alternating Least Squares

Alternating least squares is a matrix factorization algorithm that factorizes the matrix R into two factors, U and V, such that $R = U^T V$ [65]. In the context of recommendation, U is the user matrix, and V is the item matrix. The i^{th} column of the user matrix is denoted by u_i and the i^{th} column of the item matrix is v_i. The matrix R can be called the ratings matrix with $(R)_{i,j} = r_{i,j}$ [65] . To find the user and item matrix, the following problem is solved:

$$\arg\min_{U,V} \sum_{\{i,j|r_{i,j}\neq 0\}} \left(r_{i,j} - u_i^T v_j\right)^2 + \lambda \left(\sum_i n_{u_i} \|u_i\|^2 + \sum_j n_{v_j} \|v_j\|^2\right) \quad [65]$$

with λ being the regularization factor, n_{ui} being the number of items the user i has rated and n_{vi} being the number of times the item j has been rated [65]. The hyperparameters number of latent factors and regularization factors are fine tuned to create an accurate model.

6.3 Data

The accuracy of the model is heavily dependent on the training data. Hence, having a large and precise dataset representing the problem statement is crucial. The Instacart Market Basket Analysis dataset available on Kaggle comprises the actual orders made by the customers over one year [23]. The dataset contains multiple CSV files, each file including varied information about users, their orders, and products. The aisles.csv and departments.csv have information about aisles and departments, respectively, along with ids for each. The order_products_prior.csv and order_products_train.csv files contain the information about the orderid, and ids of the products included in that order. The orders.csv

file provides knowledge about the order and the user who ordered it. The products.csv comprises product name, id, aisleid, and departmentid. The required data for the recommendation system is spread across different files, So some pre-processing is needed to present the training data in the required format. As the first step, the users are merged with the products they ordered through different orders. This can be achieved by integrating the orders_products_prior and order_products_train with the orders files, with orderid being the common column. The next step is to take note of the number of times a user has ordered a product. A dataset with userid, productid, and a column called count, which represents the order count, is created. This newly created dataset is used as the training dataset for the recommendation model. A snippet of the dataset is shown in the figure 46.

user_id	product_id	count
1	196	11
1	12427	10
1	10258	10
1	25133	9
1	13032	4
1	46149	4
1	26405	3
1	26088	3
1	49235	3
1	13176	2
1	38928	2
1	39657	2
1	14084	1
1	30450	1
1	41787	1
1	17122	1
1	10326	1
1	35951	1
1	27845	1
2	32792	10
2	24852	8
2	47209	8
2	19156	6
2	1559	6
2	16589	6
2	18523	6
2	33754	5

Figure 46: Pre-processed dataset for training the model.

6.4 Implementation

A successful e-commerce recommendation system should be able to navigate the users to the products they might be interested in purchasing or potentially missed while compiling their list. The proposed recommendation system is expected to suggest ten products similar to the ones ordered by the user and their likings. The current dataset does not provide information about whether a customer liked a product or not. With a lack of explicit feedback, implicit feedback is inferred based on the number of times a user has ordered a particular product. A new rating system has been created based on the reorder numbers. Any product ordered more than five times is given a rating of 5 for that user, and the other rating follows the decreasing order. A snippet of the data frame with userid, productid, and rating is presented in the figure below 47.

user_id	product_id	rating
1	196	5
1	12427	5
1	10258	5
1	25133	5
1	13032	4
...	...	...
206209	23909	1
206209	48697	1
206209	755	1
206209	6825	1
206209	42606	1

Figure 47: Data frame with implicit ratings.

Collaborative filtering techniques are being used to implement the recommendation system as the purpose is to suggest similar products which the user has never ordered. These techniques focus on understanding the user-to-user and item-to-item similarities and will come up with the best-fit recommendations. The singular value decomposition algorithm is used for the matrix factorization to analyze the features of the input data. Two learning algorithms, Stochastic gradient descent (SGD) and Alternating least errors (ALS) are employed to train the model. Python is the main language for training the model and implementing the recommendation system. The rich set of existing libraries with pre-defined machine learning algorithms is the primary reason for choosing python. The libraries Surprise and Spark are imported to utilize their SGD and ALS algorithms, respectively.

The training dataset is divided into two parts, 75 percent being train data and the rest 25 as test data. This helps in gauging the performance of both algorithms. The trained models are tested against the test data, and the one with the least error rate is chosen to develop the final recommendation system. Hyperparameter tuning is later performed on both models using GridSearch method to find the best set of parameters for the accurate functioning of the model. The graphs of the grid search for both models are presented below in figure 48 and figure 49. The root-mean-square-error for SGD trained model is 1.23 and for ALS is 1.28. Hence SGD trained model is used to implement the recommendation system.

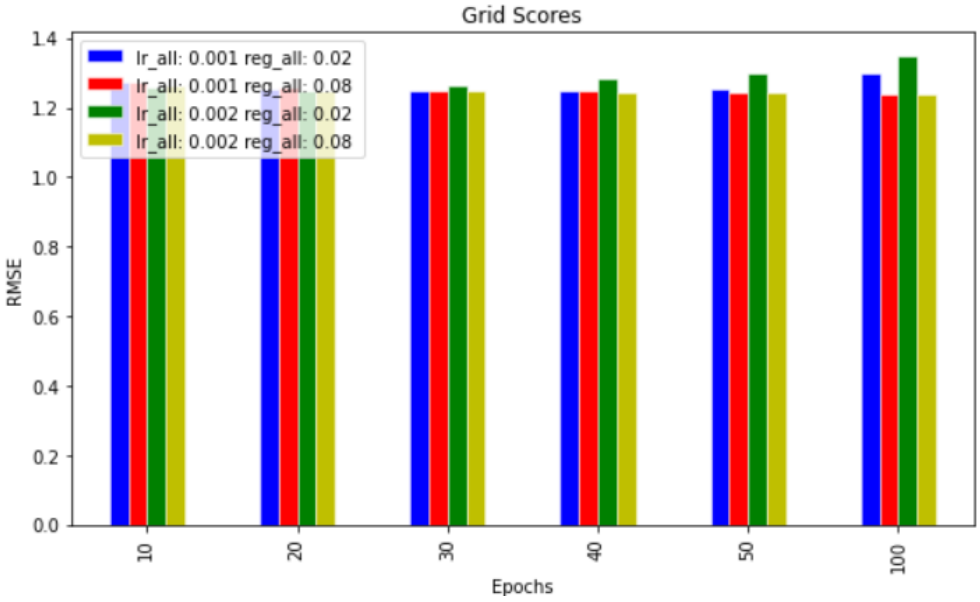

Figure 48: Grid Search Hyperparameter tuning graph for SGD.

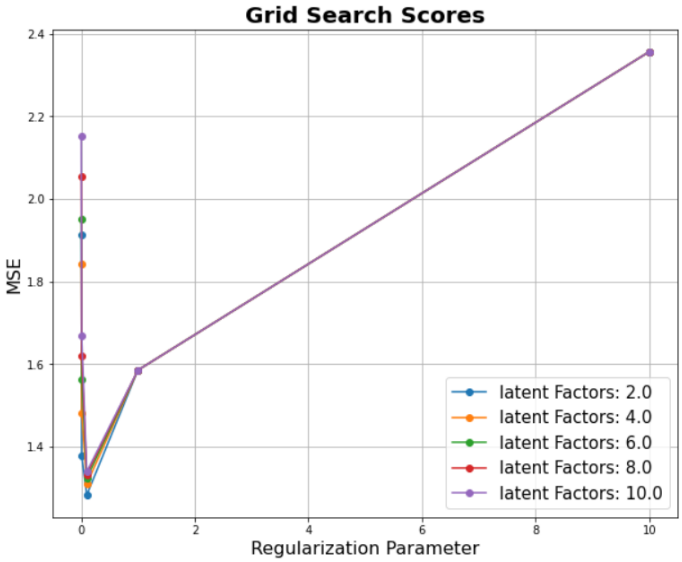

Figure 49: Grid Search hyperparameter tuning graph for ALS.

The system takes userid as the input and generates a list of products with their predicted ratings. These ratings indicate the user's likability towards that product. The function takes userid and identifies the products which have never been ordered by the user. These productids and userid are then inputted into the model for predictions. The model generates a data frame with productid, userid, and predicted ratings. The data is merged with other files to understand and improvise the recommendations. The newly developed data frame has additional information about the product, such as the name, aisle, and department it belongs to. The generated data frame is shown in figure 50.

	product_id	predictions	Unnamed: 0	product_name	aisle_id	department_id	aisle	department
0	17469	3.072820	17468	Lo-Carb Energy Drink	64	7	energy sports drinks	beverages
1	47231	2.834695	47230	Ultra-Purified Water	115	7	water seltzer sparkling water	beverages
2	19660	2.832166	19659	Spring Water	115	7	water seltzer sparkling water	beverages
3	48041	2.830864	48040	DanActive Vanilla Probiotic Dairy Drink	120	16	yogurt	dairy eggs
4	9292	2.828631	9291	Half And Half Ultra Pasteurized	84	16	milk	dairy eggs
...	...	...	...	...	...	...	...	...
49529	13878	1.288207	13877	Daily Grind Black Peppercorn	104	13	spices seasonings	pantry
49530	14626	1.278443	14625	Organic Cherry/Watermelon Fruity Hearts Treat ...	6	2	other	other
49531	19423	1.276199	19422	Nature's Colors Decorating Set	104	13	spices seasonings	pantry
49532	13373	1.258225	13372	Food Coloring	97	13	baking supplies decor	pantry

Figure 50: Predicted ratings by the model.

After multiple evaluations with different users, we noticed that products from one kind of department are rated high and are being recommended at the top. The products are internally divided into different aisles, and aisles containing the same category of products are grouped together into departments. This relationship is leveraged to alter the list. The recommendation list is further grouped by aisle id to diversify the recommendations while still maintaining accuracy. The highest rated product from each aisle is then included in the list, thus allowing the inclusion of various departments. As the final output, the user is presented with the top ten products from the recommendation list as the similar products they might be interested in. The following figure shows the suggested products for a given user based on the ratings predicted by the trained model.

aisle	product_id	product_name
milk	29447	Milk, Organic, Vitamin D
energy sports drinks	17469	Lo-Carb Energy Drink
soy lactosefree	43394	Organic Lactose Free Whole Milk
bread	44741	Honey Whole Wheat
eggs	44177	Large Grade A Eggs
yogurt	4962	Yotoddler Organic Pear Spinach Mango Yogurt
fresh fruits	39276	Bananas
baby food formula	42152	Toddler Drink Go & Grow Powder
water seltzer sparkling water	13128	Purified Alkalkine Water with Minerals pH10
dog food care	23876	Chunky Beef Meaty Ground Dog Food

Figure 51: Recommendation list generated by the model for a particular user.

Chapter 7

7 Implementation Challenges

This section provides a summary of the challenges and setbacks faced during the tool's development. The biggest challenge in implementing this project is obtaining real-time or close to real-time product data. Web scraping has been employed to extract the data from the digital flyers. Different scraping tools and techniques are exploited to gather the product data. The underlying legalities of scraping made it challenging to extract the data, as many corporations do not agree with the principles of scraping. Bot detection, IP blocking, and explicit disagreement to scraping through robots.txt files have caused a hindrance to the data collection procedure. Most local grocery stores do not display all the product data on their site but rather showcase a flyer with the ongoing sales and promotions making it impossible to gain complete product information. Therefore, scraping has been scratched out, and had to examine other data generation techniques.

A significant amount of time has been spent deciding the primary programming language for the development. Popular languages are evaluated based on their support for multi-platform, the libraries and frameworks for data analysis, the developer community, and the availability of online resources. Based on the initial research, Python is selected as the development language. The rich library and framework support for scraping and data analysis have made Python an inevitable choice. Even after withdrawing from scraping, Python was still considered the application language because of its rich pre- and post-processing techniques.

While it was an excellent programming language for data extraction and analysis, there is a visible lack of support for mobile application development. A survey on existing frameworks that assist Python in mobile application development was conducted, and Kivy was considered for this implementation. An efficient server was developed, but the UI design was not up to the standards. The lack of interactive widgets and limited user interactions have decreased the marketability of the initial product. Another round of analysis resulted in the development of the application with Java. Additional efforts were required to refactor and redesign the tool to include the features offered by the new language.

Android studio has inbuilt support for various databases, such as SQLite, Rooms, etc., through libraries and packages. These commonly used databases are considered unfit as the application prerequisites conform to a NoSQL database. Therefore, substantial time and effort went into finding a suitable android-supported NoSQL database. The lack of resources specific to mobile and android development and lesser community support has made it challenging to integrate MongoDB into the application. The

Notification service required integration of Firebase Cloud Messaging (fcm) with the MongoDB App Services. While MongoDB supports the push notification feature, the official documentation did not include any information about the application. At one point, Firestore, the official database of Firebase, was considered to replace the existing MongoDB database. The new firestore database did not offer the flexibilities of MongoDB, such as offline first and scalability. Hence the application was moved back to MongoDB Atlas. This back and forth caused a notable setback in the development phase. After numerous trial and error methods, using third-party applications to integrate fcm and MongoDB, we successfully integrated both the services and developed the notification service.

Chapter 8

8 Conclusion

8.1 Summary

With the expansion of the online marketplace and the launch of applications that provide easy access to these stores, a novice consumer is overwhelmed to access the available product data. The difficulty of navigating through each site and finding the best deal has opened doors for a new kind of application called innovative shopping applications. Some of these applications assist the user in finding desired products nearby their location, while others help identify the best deals by comparing the price across various purchasing options. However, there is no single tool targeted toward addressing all the existing problems in the e-commerce space. A buyer needs an application that helps the user in product search, compares prices from nearby available stores, calculates the total cost of that shopping session, and presents this information in a visually beneficial way. These applications can significantly reduce the time spent selecting a store every week for grocery and basic needs shopping. This thesis is aimed to bridge this gap and demonstrate the benefits of a smart shopping tool and the features it can offer a user to improve their shopping experience. The proposed solution can allow the user to select their radius for their search, locates the nearby partner stores, provides product search, and display the price of all the selected products in the stores under the selected vicinity of the application. The framework also notifies the user whenever a previously unavailable product becomes available. As part of our proposed framework, an android application was developed to present the working model of this solution. The research leverages data visualization techniques to present the results simply and efficiently. E-commerce is a growing industry and is expected to reach new heights in the coming years. All the design decisions are taken by considering this expansion, and the application is competent to scale up when needed.

8.2 Limitations and Future Work

The current implementation of SmartCart is based on the synthetic data generated through pre-processing the existing datasets. In the future, the application is expected to integrate real-time data from various Canadian retail stores. The product catalogue can be extended to include other categories such as Furniture, Automobiles, local small-scale business, and the food and restaurant industry. The application's reach is restrained to the city of London, which can be improved by extending it to other cities and provinces in Canada. A real-time system, particularly e-commerce applications, must be updated regularly as the data changes frequently. The proposed framework was built on one-time data

and did not consider the design requirements for repeated data updations and modification. Various database update techniques will be studied to integrate the optimized solution in future work. Privacy is one of the biggest concerns for consumers as information stored by these applications can be used to infer customer habits and behaviors. While requesting the user's permission at the required stages, the designed framework in this thesis does not completely adhere to all privacy policies and guidelines. There are database confidentiality rules and user privacy protection laws which safeguard a user's rights. Measures must be taken to understand and introduce encryption and other techniques to shield sensitive information.

A payment and delivery module can be included in the existing tool allowing the user to have a complete shopping experience. The customer information acquired from the application and their product search is proven valuable for many business units. The proposed recommendation system, which suggests the products to the individual user based on their previous orders, needs to be integrated into the application. A deeper analysis of the shopping and search history can help understand a product's buying cycle; a notification module can be incorporated to remind the user to include the product in their next buy at the end of the cycle. As an alternative to the presented recommendation system, a mathematical formulation using linear programming techniques to find the best fit can be implemented. A comparison of both models can aid in developing an effective recommendation engine. The current engine takes user and product information as features to feed the recommendation model. In addition, time can also be included as a feature. The time of the purchase is an important attribute as it can introduce seasonal recommendations into the framework. In the future, time can be captured along with the user's history. Information about popular products and stores can be sent back to manufacturers for market research. With these additional features, SmartCart can function as a complete shopping assistant to amateur consumers in acquiring the best deals and fulfilling their everyday shopping needs.

References

[1] Alqahtani, Abdullah & Goodwin, Robert. (2012). E-commerce Smartphone Application. International Journal of Advanced Computer Science and Applications. 3. 10.14569/IJACSA.2012.030810.

[2] Song, Xia & Yang, Shiqi & Huang, Ziqing & Huang, Tao. (2019). The Application of Artificial Intelligence in Electronic Commerce. Journal of Physics: Conference Series. 1302. 032030. 10.1088/1742-6596/1302/3/032030.

[3] BalaPriya, S., Srinivasan, N., (2022). Emerging Practices of Digital Commerce Using Artificial Intelligence Techniques in Online Merchandising Business Platforms. International Journal of Mechanical Engineering (Vol. 7 No.1). Kalahari JournalsB. Li, J. Springer, G. Bebis, and M. Hadi, “A survey of network flow applications,” *J. Netw. Comput. Appl.*, vol. 36, no. 2, pp. 567–581, 2013.

[4] Verhoef, P. C., Neslin, S. A., & Vroomen, B. (2007). Multichannel Customer Management: Understanding the research-shopper phenomenon. *International Journal of Research in Marketing*, *24*(2), 129–148. https://doi.org/10.1016/j.ijresmar.2006.11.002

[5] S. R. Sujithra and R. K. Kavitha, "Consumer Online Buying Behaviour - A Perspective Study Analysis using R," 2021 International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA), 2021, pp. 1-6, doi: 10.1109/ICAECA52838.2021.9675566.

[6] Raj, Dr. Lekh. (2016). Customer Preferences towards Online Shopping. Asian Journal of Advanced Basic Sciences.

[7] Pritam P. Kothari and Shivganga S. Maindargi, "A Study on Customers Attitude towards Online Shopping in India and its Impact: With Special Reference to Solapur City", International Journal of Advance research Ideas and Innovations in Technology, vol. 2, no. 6, pp. 1-10, 2016.

[8] Sarkar, Raja & Das, Sabyasachi. (2017). Online Shopping vs Offline Shopping : A Comparative Study.

[9] SUKHWINDER KAUR ,Dr. Vikramjit Kaur , "COMPARATIVE STUDY ON ONLINE VS. OFFLINE SHOPPING", International Journal of Creative Research Thoughts (IJCRT), ISSN:2320-2882, Volume.6, Issue 1, pp.1460-1470, January

2018, Available at :http://www.ijcrt.org/papers/IJCRT1705208.pdf

[10] B, L., & Manohar, H. L. (n.d.). *Online vs. offline shopping – A comparative study*. JETIR. Retrieved August 2, 2022, from https://www.jetir.org/view?paper=JETIRES06091

[11] R. Safavi, "Human/social factors influencing usability of E-commerce websites and systems," 2009 International Conference on Application of Information and Communication Technologies, 2009, pp. 1-5, doi: 10.1109/ICAICT.2009.5372597.

[12] K. P. Vidya et al., "Virtual Cart: Novel Approach for Revamping Smart Shopping Experience," 2018 IEEE Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER), 2018, pp. 135-140, doi: 10.1109/DISCOVER.2018.8674117.

[13] V. Perera, L. Ekanayake, A. Bandara, D. Shakya and U. S. Oruthota, "IOT Based Smart Shopping System," 2021 10th International Conference on Information and Automation for Sustainability (ICIAfS), 2021, pp. 225-229, doi: 10.1109/ICIAfS52090.2021.9606124.

[14] V. V., P. K. P. and C. R. S., "Smart Shopping Cart," 2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET), 2018, pp. 1-4, doi: 10.1109/ICCSDET.2018.8821103.

[15] A. Kumar, A. Gupta, S. Balamurugan, S. Balaji and R. Marimuthu, "Smart Shopping Cart," 2017 International conference on Microelectronic Devices, Circuits and Systems (ICMDCS), 2017, pp. 1-4, doi: 10.1109/ICMDCS.2017.8211723.

[16] S. R. Subudhi and R. N. Ponnalagu, "An Intelligent Shopping Cart with Automatic Product Detection and Secure Payment System," 2019 IEEE 16th India Council International Conference (INDICON), 2019, pp. 1-4, doi: 10.1109/INDICON47234.2019.9030331.

[17] Shah, R., Pathan, K., Masurkar, A., Rewatkar, S., & Vengurlekar, P.N. (2016). Comparison of E-commerce Products using web mining. International journal of scientific and research publications, 6.

[18] Jianxia Chen and Ri Huang, "A price comparison system based on Lucene," 2013 8th International Conference on Computer Science & Education, 2013, pp. 117-120, doi: 10.1109/ICCSE.2013.6553894.

[19] M. M. Khan, "Development of An e-commerce Sales Chatbot," 2020 IEEE 17th International Conference on Smart Communities: Improving Quality of Life Using

ICT, IoT and AI (HONET), 2020, pp. 173-176, doi: 10.1109/HONET50430.2020.9322667.

[20] S. Shaikh, S. Rathi and P. Janrao, "Recommendation System in E-Commerce Websites: A Graph Based Approached," 2017 IEEE 7th International Advance Computing Conference (IACC), 2017, pp. 931-934, doi: 10.1109/IACC.2017.0189.

[21] S. Yin and X. Luo, "A Review of Learning-Based E-commerce," 2021 16th International Conference on Intelligent Systems and Knowledge Engineering (ISKE), 2021, pp. 483-490, doi: 10.1109/ISKE54062.2021.9755410.

[22] *Platforms*. Flipp. (2022, February 25). Retrieved April 29, 2022, from https://corp.flipp.com/platforms/

[23] *Instacart Market Basket Analysis*. Kaggle. (n.d.). Retrieved April 29, 2022, from https://www.kaggle.com/c/instacart-market-basket-analysis

[24] UCI Machine Learning Repository: Online Retail Data Set. (n.d.). Retrieved April 29, 2022, from https://archive.ics.uci.edu/ml/datasets/online+retail

[25] Fooddata Central. (n.d.). Retrieved April 29, 2022, from https://fdc.nal.usda.gov/index.html

[26] *Open food facts - canada*. Open Food Facts - Canada. (n.d.). Retrieved April 29, 2022, from https://ca.openfoodfacts.org/

[27] 30, A. N., 24, C. S. A., 25, G. G. M., 6, V. M., 13, M. K. A., 2, E. J., 30, R. J., 1, K. J., 14, M. I. C. H. A. E. L. P. O. L. I. T. Z. M., 23, ddd J., & 17, D. V. J. (2017, January 24). *Open grocery database project*. Grocery.com. Retrieved April 29, 2022, from https://www.grocery.com/open-grocery-database-project/

[28] *The benefits of user profiling worth shouting about*. Hurree's Marketing Blog. (n.d.). Retrieved July 14, 2022, from https://blog.hurree.co/blog/the-benefits-of-user-profiling-worth-shouting#:~:text=Brings%20in%20more%20users,be%20attracted%20to%20your%20app.

[29] Wikimedia Foundation. (2022, April 15). *Location-based service*. Wikipedia. Retrieved July 14, 2022, from https://en.wikipedia.org/wiki/Location-based_service

[30] Gyorodi, Robert & Zmaranda, Doina & Georgian, Vlad & Győrödi, Cornelia. (2017). A Comparative Study between Applications Developed for Android and iOS. International Journal of Advanced Computer Science and Applications. 8. 10.14569/IJACSA.2017.081123.

[31] Murphy, C., designer, A. T. A. A., & Author, A. T. (2018, March 1). A comprehensive guide to wireframing and Prototyping. Smashing Magazine. Retrieved April 29, 2022, from https://www.smashingmagazine.com/2018/03/guide-wireframing-prototyping/

[32] Aparicio, Manuela & Costa, Carlos. (2014). Data Visualization. Communication Design Quarterly. 3. 7-11. 10.1145/2721882.2721883.

[33] Sadiku, Matthew & Shadare, Adebowale & Musa, Sarhan & Akujuobi, Cajetan & Perry, Roy. (2016). DATA VISUALIZATION. International Journal of Engineering Research and Advanced Technology (IJERAT). 12. 2454-6135.

[34] *What is data visualization? definition, examples, and learning resources*. Tableau. (n.d.). Retrieved April 29, 2022, from https://www.tableau.com/learn/articles/data-visualization

[35] *Backend as a service – what is a baas? - back4app blog*. (n.d.). Retrieved April 29, 2022, from https://blog.back4app.com/backend-as-a-service-baas/.

[36] Sarker, Iqbal & Faruque, Md & Hossen, Ujjal & Rahman, Atikur. (2015). A Survey of Software Development Process Models in Software Engineering. International Journal of Software Engineering and its Applications. 9. 55-70. 10.14257/ijseia.2015.9.11.05.

[37] Wikipedia Foundation. (2022, February 10). *Rapid application development*. Wikipedia. Retrieved April 29, 2022, from https://en.wikipedia.org/wiki/Rapid_application_development.

[38] *Meet android studio : Android developers*. Android Developers. (n.d.). Retrieved April 29, 2022, from https://developer.android.com/studio/intro.

[39] *Community home*. Gradle Docs. (n.d.). Retrieved April 29, 2022, from https://docs.gradle.org/current/userguide/what_is_gradle.html

[40] *Get started*. Foursquare Developer Documentation. (n.d.). Retrieved April 29, 2022, from https://developer.foursquare.com/docs/places-api-getting-started.

[41] Google. (n.d.). Google. Retrieved April 29, 2022, from https://developers.google.com/maps/documentation/geocoding/overview

[42] Google. (n.d.). *Firebase Cloud messaging | send notifications across platforms at no-cost*. Google. Retrieved July 14, 2022, from https://firebase.google.com/products/cloud-

messaging?gclid=CjwKCAjw_b6WBhAQEiwAp4HyILFu7TN_qfOw5cnec53ZPUF9L_ggo29bsFqMDm7_NTUOSeA9J_fv7xoCfKEQAvD_BwE&gclsrc=aw.ds

[43] *The Application Data Platform*. MongoDB. (n.d.). Retrieved April 29, 2022, from https://www.mongodb.com/.

[44] *MongoDB atlas database: Multi-cloud database service*. MongoDB. (n.d.). Retrieved April 30, 2022, from https://www.mongodb.com/atlas/database

[45] *Reliability*. MongoDB. (n.d.). Retrieved April 29, 2022, from https://www.mongodb.com/cloud/atlas/reliability.

[46] *MongoDB Realm Application Services — MongoDB Realm*. (n.d.). Www.mongodb.com. Retrieved April 29, 2022, from https://www.mongodb.com/docs/realm/cloud/.

[47] Push Notifications [Deprecated] - Atlas App Services. (n.d.). Retrieved July 14, 2022, from https://www.mongodb.com/docs/atlas/app-services/reference/push-notifications/

[48] Triggers Overview - Atlas App Services. (n.d.). Retrieved July 15, 2022, from https://www.mongodb.com/docs/atlas/app-services/triggers/overview/

[49] "MongoDB Realm Java SDK — MongoDB Realm." *Www.mongodb.com*, www.mongodb.com/docs/realm/sdk/java/. Accessed 29 Apr. 2022.

[50] Sharma, M. (2022, April 8). *Introduction to the realm SDK for Android*. MongoDB. Retrieved April 29, 2022, from https://www.mongodb.com/developer/how-to/introduction-realm-sdk-android/

[51] Google. (n.d.). Google. Retrieved April 29, 2022, from https://developers.google.com/maps/documentation/android-sdk/utility

[52] Mahesh, B. (2020). Machine learning algorithms-a review. International Journal of Science and Research (IJSR).[Internet], 9, 381-386.

[53] El Naqa, I., Murphy, M.J. (2015). What Is Machine Learning?. In: El Naqa, I., Li, R., Murphy, M. (eds) Machine Learning in Radiation Oncology. Springer, Cham. https://doi.org/10.1007/978-3-319-18305-3_1

[54] Baştanlar, Y., Özuysal, M. (2014). Introduction to Machine Learning. In: Yousef, M., Allmer, J. (eds) miRNomics: MicroRNA Biology and Computational Analysis. Methods in Molecular Biology, vol 1107. Humana Press, Totowa, NJ. https://doi.org/10.1007/978-1-62703-748-8_7.

[55] Roy, D., Dutta, M. A systematic review and research perspective on recommender systems. J Big Data 9, 59 (2022). https://doi.org/10.1186/s40537-022-00592-5.

[56] Y. Hu, Y. Koren and C. Volinsky, "Collaborative Filtering for Implicit Feedback Datasets," 2008 Eighth IEEE International Conference on Data Mining, 2008, pp. 263-272, doi: 10.1109/ICDM.2008.22.

[57] X. Li, "Research on the Application of Collaborative Filtering Algorithm in Mobile E-Commerce Recommendation System," 2021 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC), 2021, pp. 924-926, doi: 10.1109/IPEC51340.2021.9421092.

[58] Wikimedia Foundation. (2022, July 26). Matrix factorization (recommender systems). Wikipedia. Retrieved July 29, 2022, from https://en.wikipedia.org/wiki/Matrix_factorization_(recommender_systems)

[59] Chen, D. (2020, July 9). Recommendation system-matrix factorization. Medium. Retrieved July 29, 2022, from https://towardsdatascience.com/recommendation-system-matrix-factorization-d61978660b4b.

[60] Wikimedia Foundation. (2022, July 27). Hyperparameter (machine learning). Wikipedia. Retrieved July 28, 2022, from https://en.wikipedia.org/wiki/Hyperparameter_(machine_learning)

[61] Nyuytiymbiy, K. (2022, March 28). Parameters and hyperparameters in machine learning and Deep Learning. Medium. Retrieved July 28, 2022, from https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac

[62] Wikimedia Foundation. (2022, June 15). Root-mean-square deviation. Wikipedia. Retrieved July 28, 2022, from https://en.wikipedia.org/wiki/Root-mean-square_deviation

[63] Chen, D. (2020, August 6). *Recommender system-singular value decomposition (SVD) & truncated SVD*. Medium. Retrieved July 29, 2022, from https://towardsdatascience.com/recommender-system-singular-value-decomposition-svd-truncated-svd-97096338f361#:~:text=Singular%20value%20decomposition%20(SVD)%20is%20a%20collaborative%20filtering%20method%20for,factor%20model%20for%20matrix%20factorization.

[64] Dutta, S. (2020, October 29). *Why stochastic gradient descent works?* Medium. Retrieved August 30, 2022, from https://towardsdatascience.com/https-towardsdatascience-com-why-stochastic-gradient-descent-works-9af5b9de09b8

[65] Alternating least squares. Apache Flink 1.2 Documentation: Alternating Least Squares. (n.d.). Retrieved July 29, 2022, from https://nightlies.apache.org/flink/flink-docs-release-

1.2/dev/libs/ml/als.html#:~:text=Examples-,Description,and%20is%20called%20latent%20factors.

Curriculum Vitae

Name:	Susmitha Hanumanthu
Post-secondary Education and Degrees:	University College of Engineering Osmania University, Hyderabad, India 2013-2017 B.E. The University of Western Ontario London, Ontario, Canada 2021-2022 M.Sc.
Related Work Experience	Teaching Assistant The University of Western Ontario 2021-2022