Dailify

Easing your everyday essentials

2018 - 2022

A project report of Phase-II submitted to

Rajiv Gandhi Proudyogiki Vishwavidyalaya,

towards the partial fulfillment of the requirements for the degree of

Bachelor of Technology in

Computer Science Engineering

Guided by:
Dr. Sonika Shrivastava
Assistant Professor
Dept. of Computer Engg.

Dr. Vandan Tewari
Associate Professor
Dept. of Computer Engg.

Submitted by
0801CS181003 - Abhi Jain
0801CS181007 - Ajinkya Dandvate
0801CS181008 - Ajinkya Taranekar
0801CS181032 - Krati Jain
0801CS181039 - Maulik Ramnani

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING
SHRI G.S. INSTITUTE OF TECHNOLOGY AND SCIENCE,
INDORE (M.P.)

SHRI G.S. INSTITUTE OF TECHNOLOGY AND SCIENCE

INDORE (M.P.)

DEPARTMENT OF COMPUTER ENGINEERING

2021 - 2022

RECOMMENDATION

The project report of Phase-II entitled **"Dailify"** submitted by: **0801CS181003 - Abhi Jain, 0801CS181007 - Ajinkya Dandvate, 0801CS181008 - Ajinkya Taranekar, 0801CS181032 - Krati Jain, 0801CS181039 - Maulik Ramnani**, students of B.E. IV year in the session 2021-2022, towards partial fulfillment of the degree of **Bachelor of Technology Computer Science Engineering** of Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal is a satisfactory account of their work.

Dr. Sonika Shrivastava
Project Guide
Dept. of Computer Engg.

Dr. Urjita Thakar
Professor & Head
Dept. of Computer Engg.

Dr. Vandan Tewari
Co. Project Guide
Dept. of Computer Engg.

Dean Academics
S.G.S.I.T.S, Indore

SHRI G.S. INSTITUTE OF TECHNOLOGY AND SCIENCE

INDORE (M.P.)

DEPARTMENT OF COMPUTER ENGINEERING

A Govt. Aided Autonomous Institute, Affiliated to RGPV, Bhopal

2021 - 2022

CERTIFICATE

This is to certify that the Phase II of project work entitled **"Dailify"** submitted by **0801CS181003 - Abhi Jain, 0801CS181007 - Ajinkya Dandvate, 0801CS181008 - Ajinkya Taranekar, 0801CS181032 - Krati Jain, 0801CS181039 - Maulik Ramnani**, students of B.Tech. IV year in the session 2021-2022, towards partial fulfillment of the degree of **Bachelor of Technology in Computer Engineering** of Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal is a satisfactory account of their work.

Internal Examiner
Date:

External Examiner
Date:

DECLARATION

We, **Abhi Jain (0801CS181003), Ajinkya Dandvate (0801CS181007), Ajinkya Taranekar (0801CS181008), Krati Jain (0801CS181032), and Maulik Ramnani (0801CS181039)**, declare that the Major Project (Phase-2) work on **"Dailify"** is our own work conducted under supervision of **Dr. Sonika Shrivastava, Assistant Professor, Computer Engg. S.G.S.I.T.S. Indore(M.P.) and Dr. Vandan Tewari, Associate Professor, Computer Engg. S.G.S.I.T.S. Indore(M.P.).**

I further declare that to the best of our knowledge this project work does not contain any part of any work which has been submitted for the award of any degree or any other work either in this University or in any other University/websites without proper citation and is plagiarism free.

0801CS181003 - Abhi Jain

0801CS181007 - Ajinkya Dandvate

0801CS181008 - Ajinkya Taranekar

0801CS181032 - Krati Jain

0801CS181039 - Maulik Ramnani

Date :

ACKNOWLEDGEMENT

With great pleasure and sense of obligation we express our heartfelt gratitude to our esteemed guides **Dr. Sonika Shrivastava**, Assistant Professor, S.G.S.I.T.S. Indore and **Dr. Vandan Tewari**, Associate Professor, S.G.S.I.T.S. Indore, whose constant encouragement enabled us to work enthusiastically. Our project guides, in spite of their heavy work commitments and busy schedule, have taken the time to bestow upon us their invaluable guidance and support.

We are grateful to **Dr. Urjita Thakar**, Head, Department of Computer Engineering, S.G.S.I.T.S. Indore, for the facilities provided in completion of this project.

We sincerely wish to express, our gratefulness to all the members of staff of Computer Engineering Department who have extended their cooperation at all times and have contributed in their own way in developing the project.

0801CS181003 - Abhi Jain
0801CS181007 - Ajinkya Dandvate
0801CS181008 - Ajinkya Taranekar
0801CS181032 - Krati Jain
0801CS181039 - Maulik Ramnani

Place: Indore
Date:

ABSTRACT

With increasing prices of fuel, delivery services are not feasible and economical for business, thus curbside service is a popular omnichannel retail alternative, intended to enhance the convenience of online customers. Focusing on the pick-up stage of curbside service, we develop an integrated system for customers as well as store owners to ensure quality checks and support small scale businesses. A multi-step development procedure involving checks on every aspect of service for effectiveness and growth. The study also indicates that store associates with such methodology will continue to be seen as an important source of knowledge and insight, but the relationship between the shopper and employee will need to be digitized. For 80% of consumers, digital communications with store associates over the next 6 months are "likely" or "very likely." As well, store inventory visibility will become the centrepiece of all associate and customer experiences. This includes showing accurate, real-time store inventory online, filtering stores by product availability and calculating the effective route of customers on their way to the store.

Table of Contents

Recommendation ii

Certificate iii

Declaration iv

Acknowledgement v

Abstract vi

List of Figures x

List of Tables xii

1 INTRODUCTION 1

1.1 Preamble 2

1.2 Need of the Project 2

1.3 Problem Statement 3

1.4 Project Objective 3

1.5 Proposed Approach 3

1.6 Organization of the Report 3

2 BACKGROUND 5

2.1 Tools and Technologies 6

3 LITERATURE REVIEW 9

3.1 Summary of reviewed Literature 10

3.1.1 Curbside pickups with a cost-effective strategy 10

3.1.2 Our Solution 11

3.1.3 Comparison with existing apps 11

3.1.4 Drawbacks of traditional grocery delivery apps. 11

3.1.5 Curbside pickup boon to cost-effective way out. 12

3.1.6 Comparison of US Based Curbside grocery app with Dailify. . . . 13

3.1.7 Scale of Impact 13

4 ANALYSIS **15**

4.1 Existing Systems 16

4.2 Detailed Problem Statement 16

4.3 Requirement Analysis 16

4.3.1 Functional Requirements 16

4.3.2 Data Flow Diagram (Level 0) 17

4.3.3 Data Flow Diagram (Level 1) 17

4.3.4 Use Cases 18

4.3.5 Activity Diagram 18

4.4 Non Functional Requirements 19

4.5 System Requirements 20

4.5.1 Software Requirements 20

4.6 Hardware Requirements 20

5 DESIGN **21**

5.1 System Architecture 22

5.2 Software Design 23

5.2.1 Sequence Diagram 23

5.3 Data Design 24

5.3.1 ER Diagram 24

5.4 Algorithms / Flowchart Design 25

6 IMPLEMENTATION **28**

6.1 Frontend Implementation Status 29

6.2 Backend Implementation Status 32

6.3 Deployment Status 35

7 TESTING & RESULT **38**

7.1 Datasets Used 39

7.1.1 Blinkit Inventory 39

7.1.2 Indore Grocery Stores 41

7.2 Testing and Improvements 43

7.3 Results 45

7.3.1 Consumer Acceptance 45
7.3.2 Store Owners Feedback 46

8 CONCLUSION **47**

8.1 Conclusion 48
8.2 Limitations of Proposed Approach 48
8.3 Future Enhancements 48

Appendices

A **49**

A.1 Response Time 50
A.2 Haversine Formula 50
A.3 ElasticSearch 51
A.3.1 Features it supports: 51

Bibliography **52**

LIST OF FIGURES

Fig. No.	Title	Page No.
3.1	Graph showing interest rate of users in curbside grocery service.	10
4.1	Data Flow Diagram: Level 0	17
4.2	Data Flow Diagram: Level 1	17
4.3	Use Case Diagram	18
4.4	Activity Diagram	19
5.1	System Architecture	22
5.2	Sequence Diagram	24
5.3	Entity Relationship Diagram	25
5.4	Store Finding Algorithm	26
5.5	Store Finding Algorithm: Input	26
5.6	Store Finding Algorithm: Output	27
6.1	Onboarding Process	29
6.2	Dashboard Screens	30
6.3	Product Screens	30
6.4	Store Screens	31
6.5	Order Screens	31
6.6	Eureka Server	32
6.7	Open API Docs: Consumer Service	33
6.8	Open API Docs: Store Service	33
6.9	Open API Docs: Inventory Service	34
6.10	Open API Docs: Order Service	34
6.11	Open API Docs: Store Finding Algorithm Service	35
6.12	Azure VM Glances Output	36
6.13	Github Organisation	36
6.14	Consumer App CI/CD	37
6.15	Order Service CI/CD	37
7.1	Inventory Group by Categories	39

7.2 Top 15 brands in our Database 40
7.3 Inventory Group by Ranking of Products 40
7.4 Top 10 Costly Products 41
7.5 Grocery Shops in Indore 41
7.6 Grocery shops group by locality 42
7.7 Grocery shops group by type 42
7.8 Inventory Searching without ES 43
7.9 Inventory Searching with ES 43
7.10 Store Finding Algorithm without Optimisations 44
7.11 Store Finding Algorithm with Optimisations 44
7.12 User Survey 45
7.13 Store Survey 46

A.1 Haversine Formula 51
A.2 Elastic Search v/s PostgreSQL BenchMark 51

LIST OF TABLES

Table No.	Table Title	Page No.
3.1	Table showing how Dailify benefits over tradition grocery delivery app . . .	12
3.2	Table showing how Dailify benefits over tradition grocery delivery app . . .	14

Chapter 1

INTRODUCTION

This chapter presents a brief introduction to the problem, thereby building the context needed to understand the need for the proposed system. It also states the problem statement, expatiates the objectives, the proposed approach and the organization of the report.

1.1 Preamble

In this fast-paced world, grocery shopping for a home is difficult and time taking. For instance, a user (who is living in a metropolitan city) may not have enough time to spend on buying goods, he may opt to go for an easier method, by purchasing online on Amazon Pantry or Flipkart wholesale, in this case, various small scale businesses left out in this process. Through this proposal, we overcome the problem of not only purchasing goods online, while moving towards a destination but uplifting small businesses thereby saving time and making the growth of the business. With just one click of a button, a user can select products, enter a location, and while on the way, we will tell stores to prepare your order, and when a user reaches the store, he gets the order ready for pickup.

1.2 Need of the Project

Today, supermarkets are losing customers because of rising fuel prices and the need for curbside pickup services. This is the first challenge that supermarkets are facing. The second challenge they are facing is that grocery delivery apps like BigBasket, Amazon Pantry, and Flipkart wholesale take away business from them. To combat this issue, supermarkets can provide an alternative to these apps with their own grocery delivery app. The third challenge they are facing is that people want to buy groceries online but also want to pick them up themselves at the store.

One solution to these challenges would be for supermarkets to offer curbside pickup services. Fuel costs would ease up because people will not have to make as many trips to stores which means less time spent waiting in lines or driving around looking for parking spots. Customers can directly order simply with a few clicks. Also, the promotion of small scale businesses will be at growth. So we need a utopia for the whole scenario.

1.3 Problem Statement

The Indian grocery business is huge and many companies have been working on creating mobile applications that can work as a platform for this grocery-delivery app and management. A lot of progress has been made in this field but there are still some issues like customer retention and a cost-efficient model for delivery. So there is a need for developing an economical AI-based curbside service to enhance the convenience of online customers.

1.4 Project Objective

- To implement a grocery curbside service in India's market.
- To design a proper Inventory Management for stores, allowing economically managing their whole inventory on finger touches.
- To design and analyze a Route Optimization algorithm for customers to get a route to the best shop available with the grocery items in the cart.
- To design an effective business model which will be profitable throughout the journey.

1.5 Proposed Approach

Dailify is a curbside pickup service that provides an easy and efficient pickup experience by allowing shoppers to order products online and then pick them up at the store. This model eliminates the need for costly return deliveries, which for some retailers is costly and inefficient. Dailify is available to all types of stores including grocery stores, pharmacies, convenience stores, and hotels.

1.6 Organization of the Report

- **Chapter 1:** A brief introduction about the project giving the need for the project, the problem is going to solve, objectives, and also the approach to be used.
- **Chapter 2:** This chapter describes the literature survey done to study the concepts of the existing system. And the description of studies done in research papers.

- **Chapter 3:** This Chapter describes existing systems, the detailed problem statement of the project work. It deals with a detailed analysis of the project with the functional requirements and non-functional requirements, system components, use case diagram and activity diagrams for each sub-problems

- **Chapter 4:** This chapter focuses on design details consisting of the basic architecture of the system and the flow of activities of each sub-problems.

- **Chapter 5:** This chapter revolves around the development of system modules and provides their description.

- **Chapter 6:** This chapter revolves around the implementation deals with providing the full logical view of the modules.

- **Chapter 6:** This chapter concludes the whole report till now.

Chapter 2

BACKGROUND

This chapter provides the background on information systems, existing solutions and enlists the chosen tools and technologies.

2.1 Tools and Technologies

The technology stack used in this project is a combination of programming languages, frameworks, and tools that helped us build this web/mobile app.

CLIENT

- **Android Mobile:** Mobile applications.
- **Websites:** Admin Panel is an admin-specific platform that allows for access and manipulation of data within the user interface of the site. In our project, the Admin panel (also known as dashboard UI) contains almost everything that is needed like a chart, table, and a nice small card for showing info. Admin panel gives insights into stock and other data that is required.

SOFTWARE TOOLS

- **Backend Tools:**
  - **IntelliJ IDEA** IntelliJ IDEA is an integrated development environment for JVM languages. It is integrated with tools like compiler, debugger, version control system, build tools,
  - **MySQL Workbench** MySQL Workbench is a visual database design tool. It provides an integrated environment for development, administration, database design, creation and maintenance for MySQL database systems.
- **Frontend Tools:**
  - **Visual Studio:** Visual Studio is a code editor used for software development. It helps in creating websites, web apps, and web services. The software contains completion tools, compilers, and other features to facilitate the software development process.
  - **Android Studio:** Android Studio is used for Flutter Applications which helps in running various tasks perfectly on the Web, macOS app, Tablet also on both Android and iOS phones.

PROGRAMMING LANGUAGES

- **Java 11:** Java 11 is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible.
- **Dart:** Dart is a programming language designed for client development, such as for the web and mobile apps. It is developed by Google and can also be used to build server and desktop applications. Dart is an object-oriented, class-based, garbage-collected language with C-style syntax.
- **Python:** Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation.

TECHNOLOGIES

- **Flutter:** It is a mobile UI framework. It helps in creating natively compiled applications for mobile, web, and desktop from a single codebase. Dart is used to code flutter apps.
- **SpringBoot:** It is a framework that provides a platform to get started with an auto configurable production-grade Spring application.
- **FastAPI:** FastAPI is a Web framework for developing RESTful APIs in Python. It enables us to use a REST interface to call commonly used functions to implement applications. It helps in calling common building blocks for an app.

DATABASE

- **MySQL:** It is an open-source relational database management system that helps in managing database services to deploy cloud-native applications. It is used for a wide range of purposes, including data warehousing, e-commerce, and logging applications.
- **ElasticSearch:** Elasticsearch is a search engine based on the Lucene library. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents.

DEPLOYMENT

- **Service:** Azure VM Instance
- **Operating System:** Ubuntu 20.04 LTS
- **Hardware Specifications**

- **Size:** Standard B4ms (2 vcpus)
- **RAM:** 4GB
- **Storage:** 256GB SSD

Chapter 3

LITERATURE REVIEW

In this chapter, the literature review related to the notification system module is given. The content here is presented after reviewing the related literature from the papers referred to in the topic of study. Further sections in this chapter elaborates the purpose.

3.1 Summary of reviewed Literature

In this section, the literature review is related to the comparison from existing organisation that are in currently in operation. Also, a brief overview of how curbside services are better than traditional approaches. The content present here is presented after reviewing the related literature from the papers referred to in the topic of study.

3.1.1 Curbside pickups with a cost-effective strategy

Data shows that at least 30% of all products ordered online are returned as compared to 8.89% in brick-and-mortar stores, adding to the reverse logistic costs to be borne by the retailers. Amidst the rising fuel prices, businesses are facing huge losses over offering free delivery services to retain customers. Online sales come at a higher cost, though, and bring new headaches, such as the need to train employees for new roles.

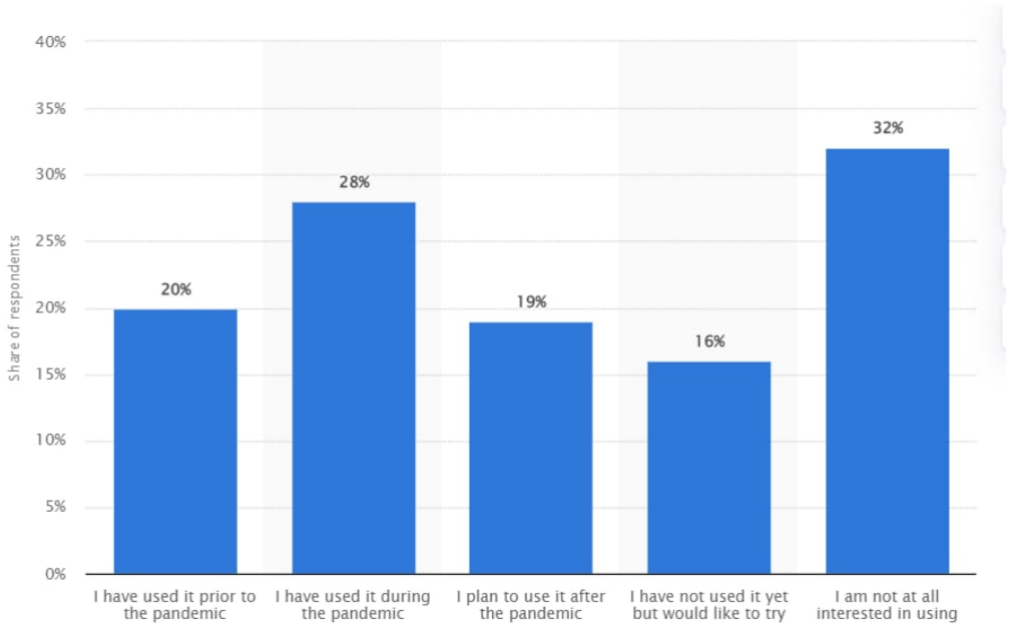


Figure 3.1: Graph showing interest rate of users in curbside grocery service.
The graph shows the on growing interest in curbside services during the pandemic time in US region

In a survey conducted in September 2020, regarding the impact of coronavirus

(COVID-19) on ordering online, curbside pickup in India, only about 20 percent of the respondents had used it before multiple apps were aligned with the pandemic. About 32 percent of the respondents reported that they were not interested in using this service.

Whereas if we compare in outside countries 85% of Shoppers Have Increased Curbside Pick-Up Since COVID-19, 79% Say a Contact less Store Pickup is Very Important to Them. There is a difference in the opinions due to geographical differences and less exposure. As curbside pickup is testing retailers and forcing them to change how they operate and train employees. It is also adding new costs that cut into their profits.

3.1.2 Our Solution

With our solution **"Dailify"** we provide consumers with an edge to effortlessly and economically make a purchase from the best available retailer and provide small/new retailers to scale up and compete in the market.

From online order to pick up, a curbside pickup process consists of five steps:

1. Store inventory visibility
2. Curbside purchase capability online
3. Alert for store staff to pick the order and ready it for customer pickup
4. Notification for customers that their order is ready for curbside pickup
5. Customer check-in at store and delivery of the order to customer's vehicle

3.1.3 Comparison with existing apps

There are multiple apps aligning with the idea of purchasing groceries online like:

- JioMart, BigBasket, Amazon Fresh are apps that provide grocery delivery services from a central warehouse.
- BlinkIt- Provides 10 min hyperlocal delivery.

3.1.4 Drawbacks of traditional grocery delivery apps.

There are several drawbacks with the apps and does not completely solve the use case of the above-mentioned problem statement.

	Traditional Apps	**Dailify**
Curbside provision	No	Yes
Customer Satisfaction	Customers may or may not be satisfied with the quality of the product as the possibility of low-grade quality product can be delivered leading to discontentment.	Customers can assure the quality at the moment itself and would not face any discomfort after purchase
Suggestions and optimization of the route	Generally, grocery apps don't provide any suggestions based on the distance.	It provides apt suggestions to the customers based on the route and the source and destination users select.
Delivery Cost	With increasing petrol prices, delivery cost adds up much more to the price of items for getting brought to our doors.	Eliminates the need for costly return deliveries.
Retailer support	Generally promotes giant holders to capture the market.	Recommendations to small scale retailers to grow their businesses.

Table 3.1: Table showing how Dailify benefits over tradition grocery delivery app

- There is limited functionality and no way of suggesting the shops based on the source and destination. Consumers have to manually select the shop they wanted to purchase before multiple aligned from.
- None of the above apps support curbside pickup. All are limited to the delivery services which add up a lot of cost at the end.

3.1.5 Curbside pickup boon to cost-effective way out.

Pickup orders are boosting sales for restaurants and grocers at better margins than delivery. But conventional pickup solutions often get customers complaining about long waiting times. We offer on-the-way pickup services for customers by knowing the details of their commute.

The customer enters the start and end location of the commute and prepares the cart of items. Based on these inputs and the data of our partner stores we recommend the

best place to get the items via our intelligent algorithms that optimizes the cost, travel time, and waiting time.

Upon selecting the desired store and checking out, the recipient store gets the order details and the customer's ETA. Once the customer reaches the store and verifies the content a unique QR code associated with the order is made available to the stores which the customer scans to complete the order and payments(if any).

We use a distributed vendor architecture in which equal opportunities are provided to all the vendors. With our advanced analytical tools, we help businesses better understand their customer shopping patterns. Our inventory management tool comes in handy for stock predictions and refills.

3.1.6 Comparison of US Based Curbside grocery app with Dailify.

Currently in India there are no such organizations which provide curbside services due to lack of popularity. But, in the west it's been shown a success among the GenZ. Here is a table (Table 3.2) which delivers the points on which we had added our comparison. Some of the known organizations are:

1. Target
2. Kroger
3. Walmart

3.1.7 Scale of Impact

Time spent on placing an order could nearly be reduced by up to 30% with the help of our store recommendation engine. Dedicated collection spots help in faster order collection and instant correction. Helping avoid the time spent in-store Ailes and billing ques. This ensures customer satisfaction and peace of mind. Small scale businesses can join the trend with our platform where they can offer their services and get exposure to new consumer bases. Inventory optimization and sales analyses help businesses scale and get an extra workforce during peak times. Regular customer feedback helps fill up the blind spots. This ensures businesses don't lose any opportunity to outshine.

	Target	**Kroger**	**Walmart**	**Dailify**
Curbside provision	Target offers pickup and drive-up options in addition to grocery delivery.	Kroger is another great way to avoid crowds in the store as well as impulse buys.	Walmart is really good about allowing substitution, while ordering groceries.	Dailify provides a wide range of products with store discounts and additional offers on the app.
Services	In drive-up service, park in one of the designated spots in front of the store for your order. When your order is ready, you'll receive an email and a notification from the Target app.	Once you're finished shopping, you can indicate whether or not you would like to allow substitutions for any out-of-stock items.	From running list of items in your cart on the right side of the screen. This makes it easy to stay under budget and keep track of what you're purchasing. Then reserve a time slot for your delivery.	To order grocery user can just select products and set its route, Dailify will give set of shops to look for which is on the way to his/her destination. Users can pickup from store when he/she gets a notification.
Local shops/ mega marts	Local shops are not connected to their services. They use their separate super mega mart setup at various places across cities to provide customer with service			Dailify provides scaling of small scale local businesses to growth.
Suggestions and optimization of the route	These apps use the mega marts location based on which nearest mart is being notified to collect orders while the user is arriving at the store, where the user could simply walkin parking lot and purchase the goods in the car itself.			Dailify uses a smart algorithm to quickly get users location along with shops nearby to fulfill the shopping list.
Delivery Cost	No additional hidden cost.	Kroger charges a service fee starting at $4.95 when ordering groceries online.	Pickup is free with a $35 online grocery order, but you can pay a service fee to get your order faster.	Grocery pickup is free, with no minimum purchasing.

Table 3.2: Table showing how Dailify benefits over tradition grocery delivery app

Chapter 4

ANALYSIS

This chapter includes the findings and details of the analysis phase. It presents the detailed problem statement and further provides the requirement analysis for the system. This also includes various analysis on artifacts such as Data Flows, Use Cases, Activity Diagrams.

4.1 Existing Systems

- Curbside is a common concept in the western world but hasn't quite found its way into the Indian markets yet which is dominated by delivery services.
- Products like Swipe by and Curbside PickUp provide a quick way for shopkeepers to set up their store for this facility but have limited functionality and are expensive to use.
- Grocery delivery apps like Grofers and BigBasket are currently leading the consumer grocery segment with retail giants holding most of the stock in large centralized warehouses effectively killing the business of small scale retailers, moreover, no returns/claims are accepted on groceries as these are perishable products.
- USP of such products is the ease of delivery in contrast to the quality and affordability that we aim to bring in with Dailify.

4.2 Detailed Problem Statement

The traditional way of buying and selling is now no more harmonious with the current generation's vital needs. Life in such a fast lane and its everlasting growing pace needs a better blend of technology to confront the momentum of everyone. So here we come up with the solution which not only took the edge off in wasting the time and energy in the long queues but also provided you with the best route to choose for the product you need to be packaged.

4.3 Requirement Analysis

4.3.1 Functional Requirements

"Functional requirements describe what a system should do." Requirements that need to be fulfilled for the users in the desired system are the Functional requirements. These requirements are gathered by interacting and asking questions about Who? What? How?

to the users about the final product. More requirements may be added further in the later release of the product. The functional requirements of our system are explained below : Dailify shall allow for:

- The system should verify the user both consumer and store owner.
- The system should provide a structured list of products and search hassle-free in the inventory.

4.3.2 Data Flow Diagram (Level 0)

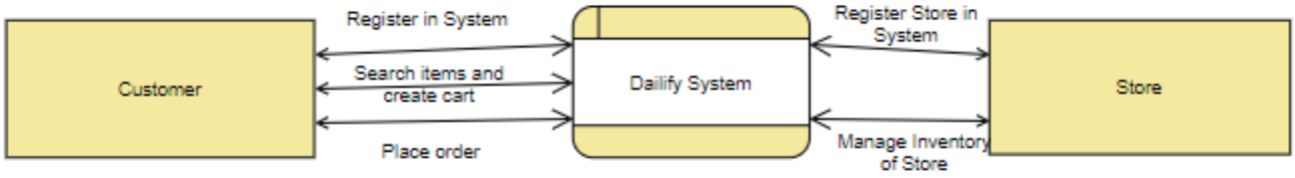


Figure 4.1: Data Flow Diagram: Level 0
A level 0 data flow diagram for Dailify System

4.3.3 Data Flow Diagram (Level 1)

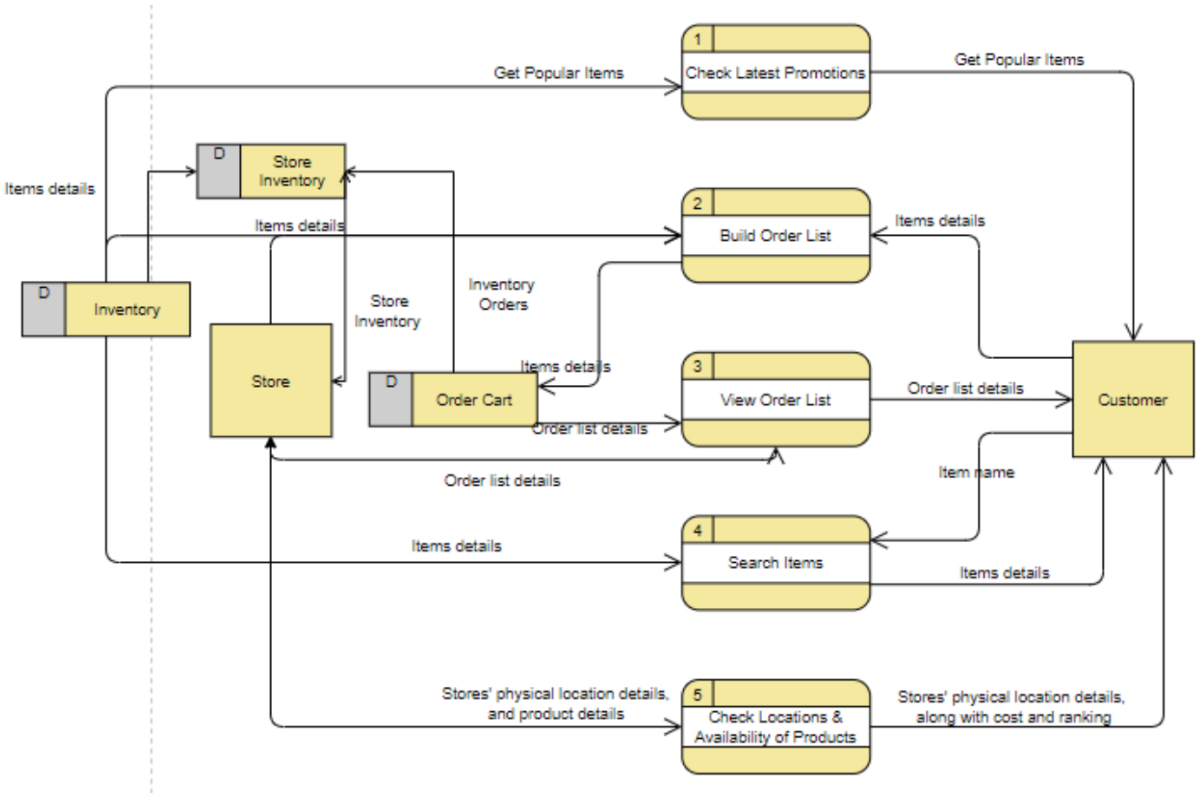


Figure 4.2: Data Flow Diagram: Level 1
A level 1 data flow diagram for Dailify System

4.3.4 Use Cases

A use case diagram is a graphical depiction of a user's possible interactions with a system. Use case analysis is used to identify requirements of the system associated with the users of the systems. Users are the actors in the Use-cases. Actions which the users can perform, is shown in the Use Case Diagram. Further is the use case diagram obtained after the use case analysis.

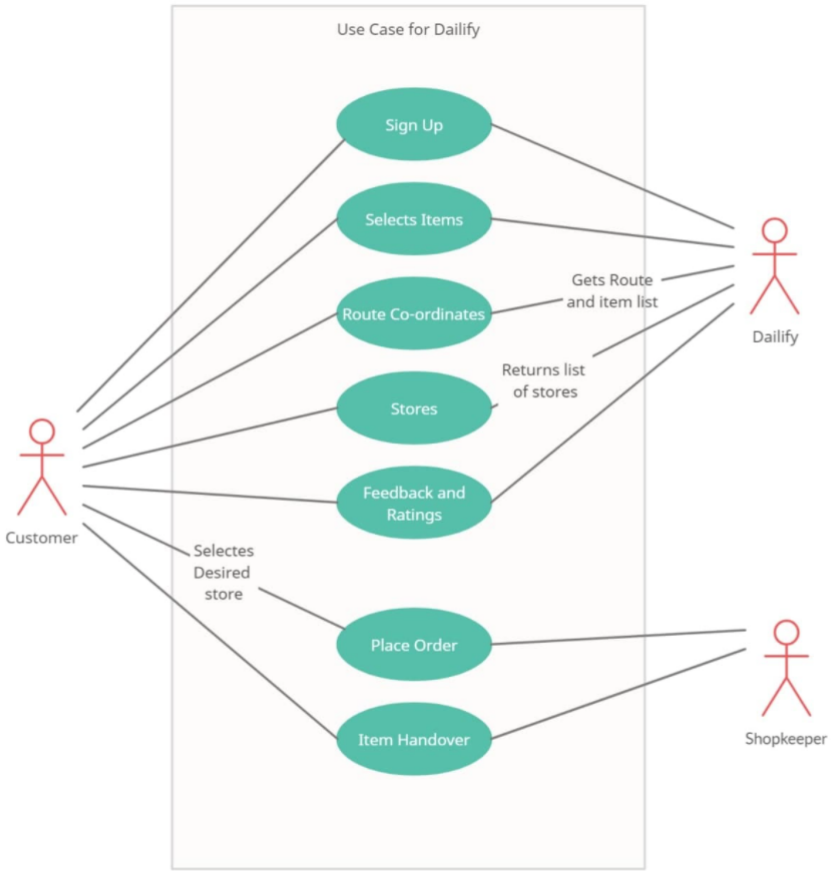


Figure 4.3: Use Case Diagram
A use diagram for Dailify System

4.3.5 Activity Diagram

In UML, the activity diagram is used to demonstrate the flow of control within the system rather than the implementation. It models the concurrent and sequential activities. The activity diagram helps in envisioning the workflow from one activity to another. It put emphasis on the condition of flow and the order in which it occurs. The flow can be sequential, branched, or concurrent, and to deal with such kinds of flows, the activity diagram has come up with a fork, join, etc.

It is also termed as an object-oriented flowchart. It encompasses activities composed of a set of actions or operations that are applied to model the behavioral diagram.

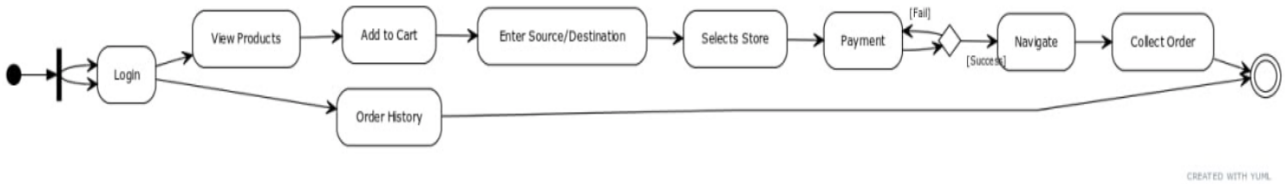


Figure 4.4: Activity Diagram
An activity diagram for Dailify System

4.4 Non Functional Requirements

These requirements are the several answers to the question "How is the system supposed to do". The Non-Functional requirements of the proposed system are as follows:

- **Performance:** Quick Availability of the data increases the performance of the application. The application should not take more than 3 seconds to load the initial screen. Should take less response time to complete desired tasks.
- **Usability:** Making the user interface more interactive by proper placing of buttons, icons, etc increases usability. Using icons related to the information makes it easy for the user to see the information quickly. Users should be able to use the app without any previous knowledge of interacting with such applications.
- **Availability:** Commonplace from where the user has the access to install and update the application. The contact of either developer or handling team should be given in the application.

Non-functional requirements of our system are mentioned below:

- The system shall perform efficient searches and ensure high performance.
- The system should be compatible with the android version of 4.4 and higher.
- The system shall assure data integrity.

4.5 System Requirements

4.5.1 Software Requirements

- **Android Client:** Requires an Android version greater than 6.
- **Website Client:** Any Browser.

4.6 Hardware Requirements

- **Android Client:** Having at least 2GB RAM.
- **Website Client:** Any Browser.

Chapter 5

DESIGN

This chapter includes the overall design of the intended project. The purpose of this chapter is to throw light on the different components of Dailify. It also elucidates the adopted architecture and proposed approach for each module undertaken. Interaction between different components along with their importance is being presented here.

5.1 System Architecture

The below diagram depicts the architecture of the system:

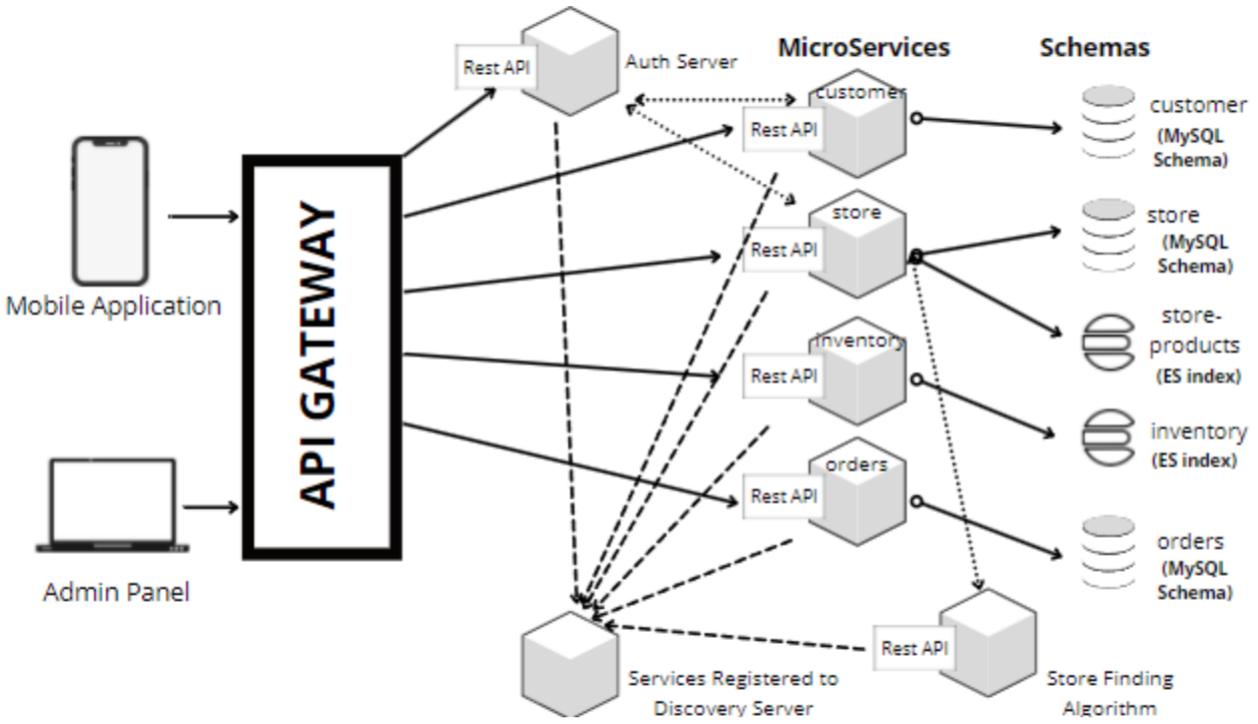


Figure 5.1: System Architecture
The overall System Architecture Design as per now for the Dailify System

1. Here we are considering microservices-based architecture. Different services are listed in the architecture diagram

2. All requests made from a mobile app or UI will go to different services via the API gateway. API gateway will take care of routing requests to services. This will authenticate and authorize the user and send back the token ID. This token is used for further communication

3. Different services like, user registration and management service, order service, payment service will use transactional databases. We will use the MySQL relational database.

4. Right now, we have used MySQL to store, inventory and store-inventory mapping, but upon further research, the analysis we came to know that ElasticSearch is the most prominent option to overcome time complexity. We had added information about the different stores, their inventory mapped, price, offers, etc which is being stored in JSON document storage in ElasticSearch. Also, the common product information is added to ElasticSearch. Whenever a customer searches for an item it will be fetched from the elastic search. Elastic search provides fast scalable search options (benchmark 50ms in ElasticSearch is >600 ms for SQL)

5. Once the user selects the product, quantity, and the distance which he is willing to deviate, also select source and destination. He will go to the checkout option and then do the payment.

6. Different payment gateways and payment options are integrated with the system and upon successful payments, the order is successfully placed.

7. (FUTURE) Once the order is placed all the information is sent to the central message Queue like Kafka or RabbitMQ. The order processing unit reads the order info and then notifies the selected store about the order. It also gets information like preparation time from the store and estimated pickup time from the source to the store coordinates and other details.

8. (FUTURE) The user gets a push notification about the order approval. The order processing and tracking service will work together and the user can track their order status, route to the store, etc.

9. (FUTURE) The customer is notified when the order is ready for pickup.

5.2 Software Design

5.2.1 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence in the field of software engineering. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of scenario. The sequence diagram for the architecture can be seen below:

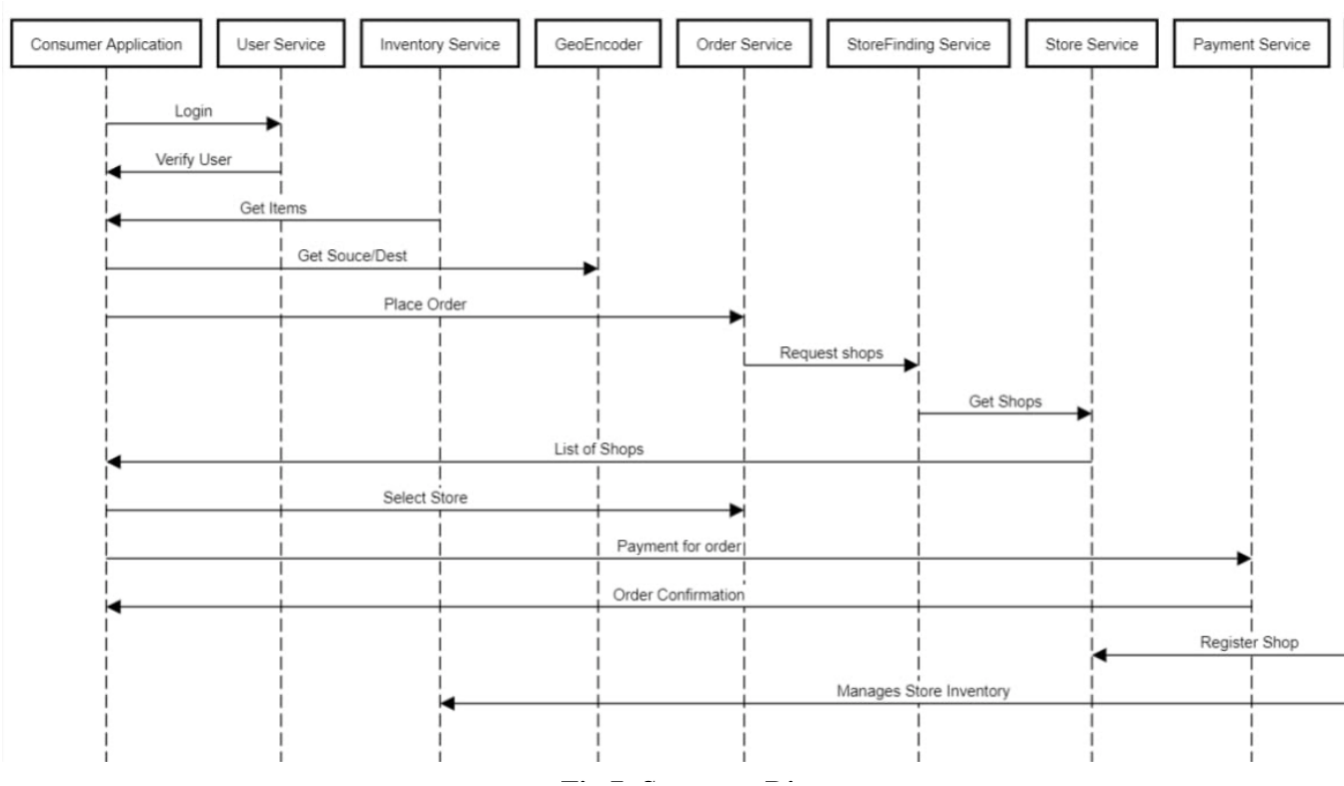


Figure 5.2: Sequence Diagram
The overall Sequence Diagram as per now for the Dailify System

5.3 Data Design

5.3.1 ER Diagram

An entity–relationship model describes interrelated things of interest in a specific domain of knowledge. A basic ER model is composed of entity types and specifies relationships that can exist between entities. Below is the ER Diagram, based on Monolithic Architecture:

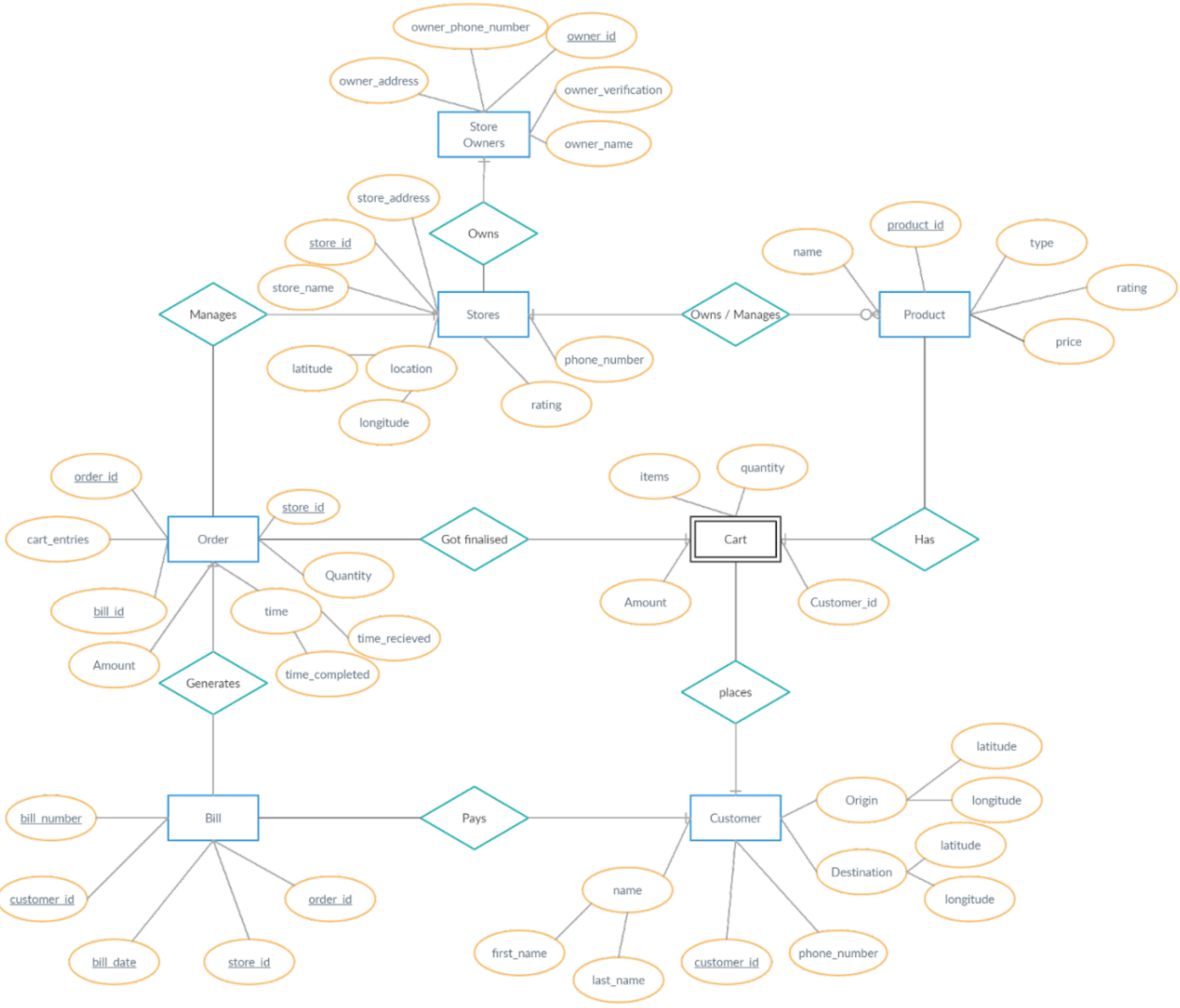


Figure 5.3: Entity Relationship Diagram
The overall ER Diagram as per now for the Dailify System

5.4 Algorithms / Flowchart Design

Based on the given input on right, We find the routes using TomTom API. Next, using city we find the stores present in the given city. Based on the stores available, we check whether a particular store has all the products available, and if all products are available we calculate the distance between coordinates on route and stores using the Haversine formula to find the nearest point from the route, which a user can deviate from.

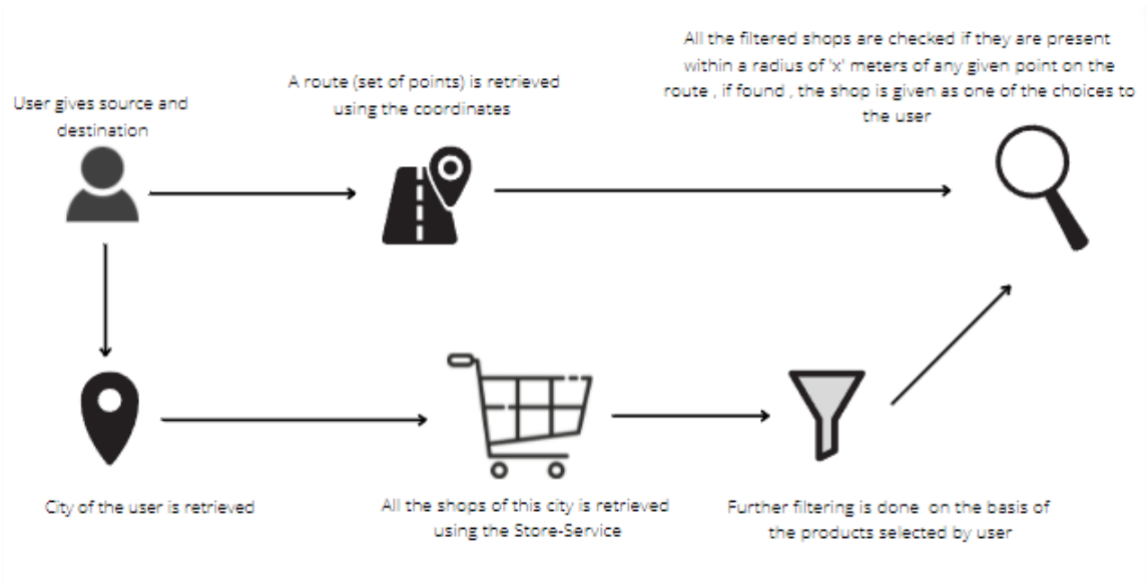


Figure 5.4: Store Finding Algorithm
Figure represents the algorithm proposed to implement the store finding algorithm for Dailify System

```
{
    "city": "Indore",
    "coordinatesPoint": [
        {"latitude": 22.71620,"longitude": 75.89654},
        {"latitude": 22.72135,"longitude": 75.89490}
    ],
    "maxAllowedDistance": 1.5,
    "products": [
        {
            "productId": "8Vbjt30B7fOZsTKevZR8",
            "quantity": 2
        },
        {
            "productId": "albpt30B7fOZsTKeGJUi",
            "quantity": 5
        }
    ]
}
```

Figure 5.5: Store Finding Algorithm: Input
Suppose, we give input in the form as this

[{"store":{"id":116,"storeName":"Apna Enterprises","storeAddress":{"street":"Radio Colony Indore","city":"Indore","state":"Madhya Pradesh","country":"India","pinCode":452001,"coordinates":{"latitude":22.7245,"longitude":75.8893}},"phoneNumber":"7312528062"},"products":[{"productId":"8Vbjt30B7fOZsTKevZR8","productName":"Ching's Secret Schezwan (250 g) Chutney","productImage":"https://uploads.dailify.tech/inventory/8Vbjt30B7fOZsTKevZR8/8Vbjt30B7fOZsTKevZR8_1639467695846.jpg","productQuantityDescription":"250 g","amount":72.0,"quantity":2,"discount":10.0},{"productId":"albpt30B7fOZsTKeGJUi","productName":"Top Ramen Curry Veg Noodles","productImage":"https://uploads.dailify.tech/inventory/albpt30B7fOZsTKeGJUi/albpt30B7fOZsTKeGJUi_1639468046678.jpg","productQuantityDescription":"280 g","amount":68.8,"quantity":5,"discount":14.0}],"totalAmount":488.0,"distance":0.672789149986115,"route":"https://api.tomtom.com/routing/1/calculateRoute/22.7162,75.89654:22.7245,75.8893:22.72135,75.8949/json?avoid=unpavedRoads&key="},{"store":{"id":70,"storeName":"Shree Krishna General Stores","storeAddress":{"street":"Radio Colony Indore","city":"Indore","state":"Madhya Pradesh","country":"India","pinCode":452001,"coordinates":{"latitude":22.7236,"longitude":75.8869}},"phoneNumber":"7312494384"},"products":[{"productId":"8Vbjt30B7fOZsTKevZR8","productName":"Ching's Secret Schezwan (250 g) Chutney","productImage":"https://uploads.dailify.tech/inventory/8Vbjt30B7fOZsTKevZR8/8Vbjt30B7fOZsTKevZR8_1639467695846.jpg","productQuantityDescription":"250 g","amount":72.0,"quantity":2,"discount":10.0},{"productId":"albpt30B7fOZsTKeGJUi","productName":"Top Ramen Curry Veg Noodles","productImage":"https://uploads.dailify.tech/inventory/albpt30B7fOZsTKeGJUi/albpt30B7fOZsTKeGJUi_1639468046678.jpg","productQuantityDescription":"280 g","amount":72.8,"quantity":5,"discount":9.0}],"totalAmount":508.0,"distance":0.8574039694131037,"route":"https://api.tomtom.com/routing/1/calculateRoute/22.7162,75.89654:22.7236,75.8869:22.72135,75.8949/json?avoid=unpavedRoads&key="},{"store":{"id":96,"storeName":"Dalal Shri Ram Trading Co.","storeAddress":{"street":"Radio Colony Indore","city":"Indore","state":"Madhya Pradesh","country":"India","pinCode":452001,

Figure 5.6: Store Finding Algorithm: Output

We get a list of shop, with it is found one shop on the route with a minimum deviation of 1500 m along with the total amount and a URL to check the route in the frontend.

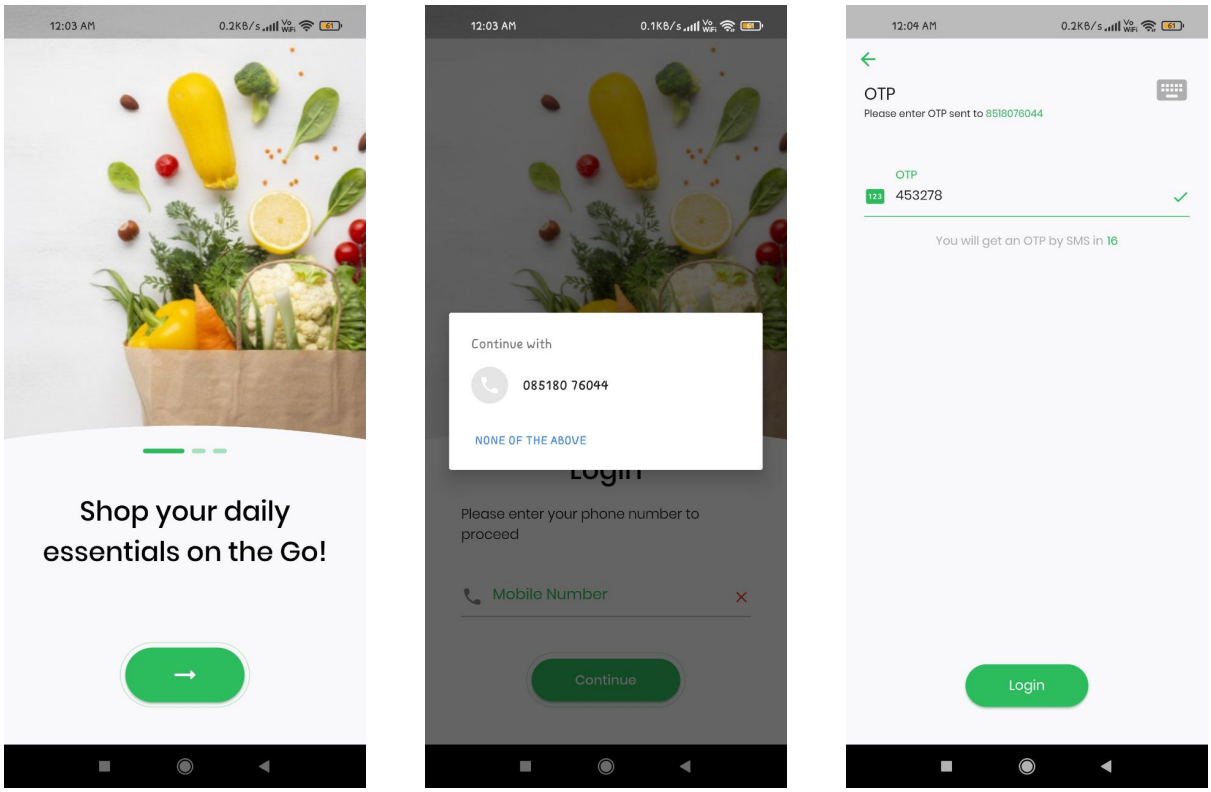
Chapter 6

IMPLEMENTATION

This chapter includes the overall implementation done so for the intended project. The purpose of this chapter is to showcase implementation of Dailify done so far from both Frontend as well as Backend perspective.

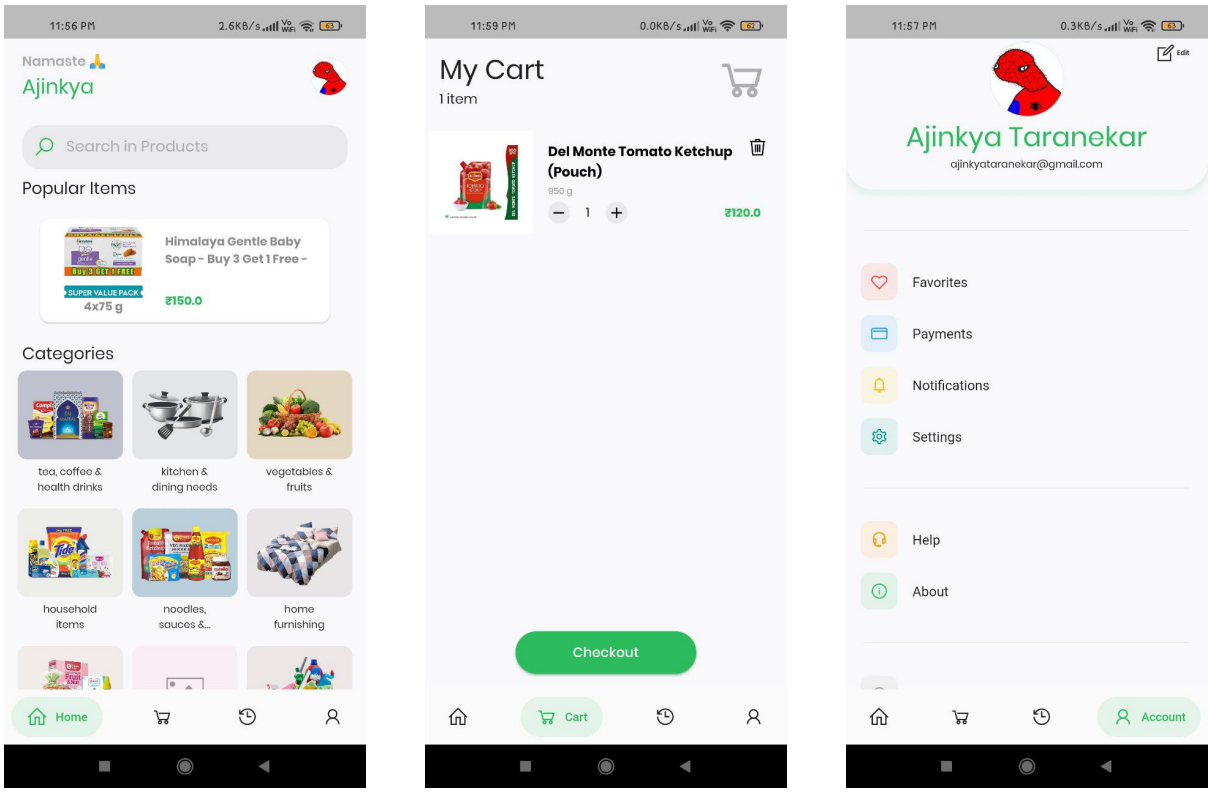
6.1 Frontend Implementation Status

Consumer App Implementation done using Flutter Framework. It fully connected with backend services with local storage as Sembast to maintain states of App.



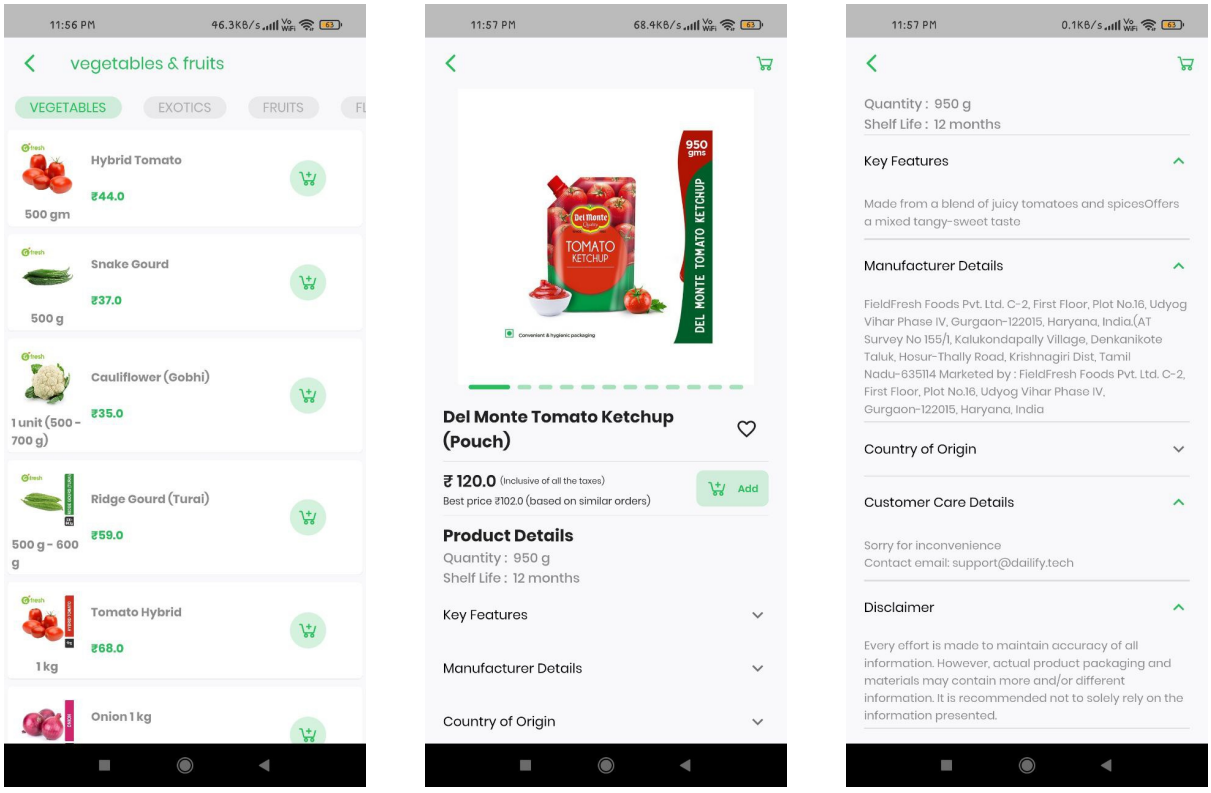
(a) Welcome Screen (b) Login Screen (c) OTP Screen

Figure 6.1: Onboarding Process



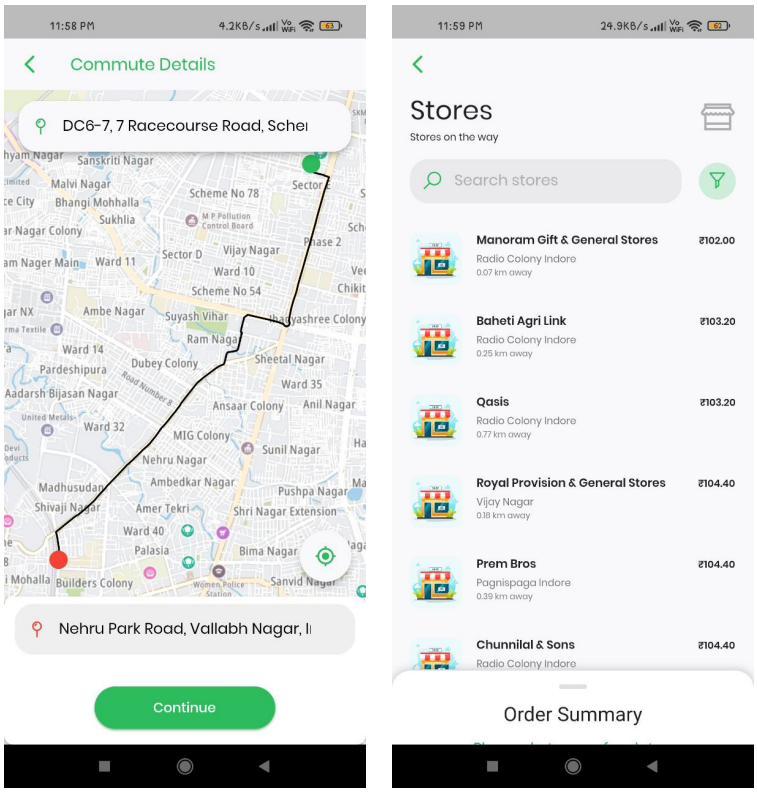
(a) Home Screen (b) Cart Screen (c) Account Screen

Figure 6.2: Dashboard Screens



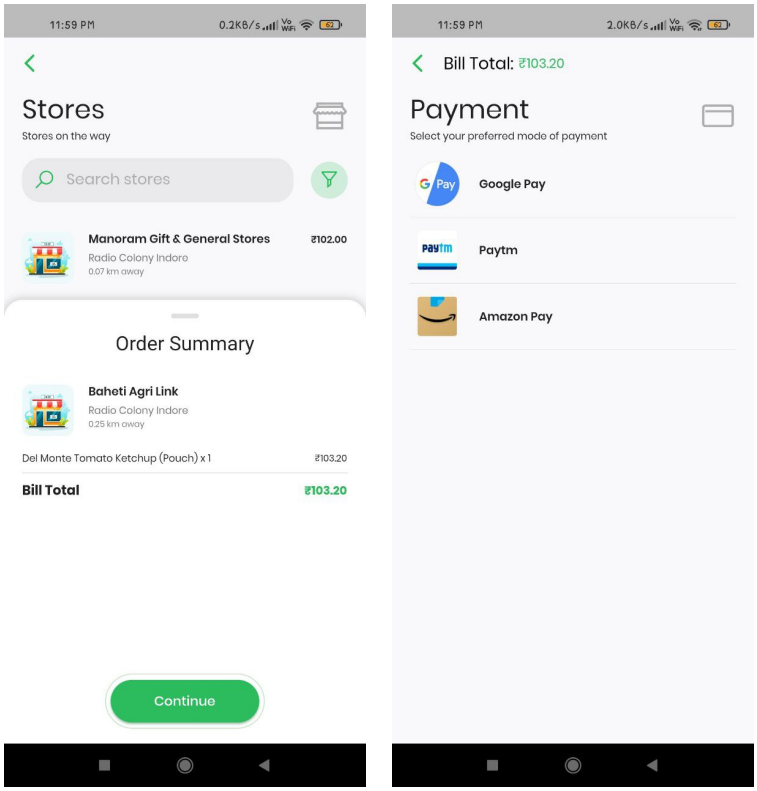
(a) Products List Screen (b) Product Screen (c) Product Details Screen

Figure 6.3: Product Screens



(a) Commute Detail Screen (b) Stores List Screen

Figure 6.4: Store Screens



(a) Order Summary View (b) Bill Payment Screen

Figure 6.5: Order Screens

6.2 Backend Implementation Status

Backend consists of various services for which we have created documentation using Open API or Swagger 3.0, links for docs here: https://api.dailify.tech/docs

Below are the images of services documentation pages, where each service is registered on Eureka Server.

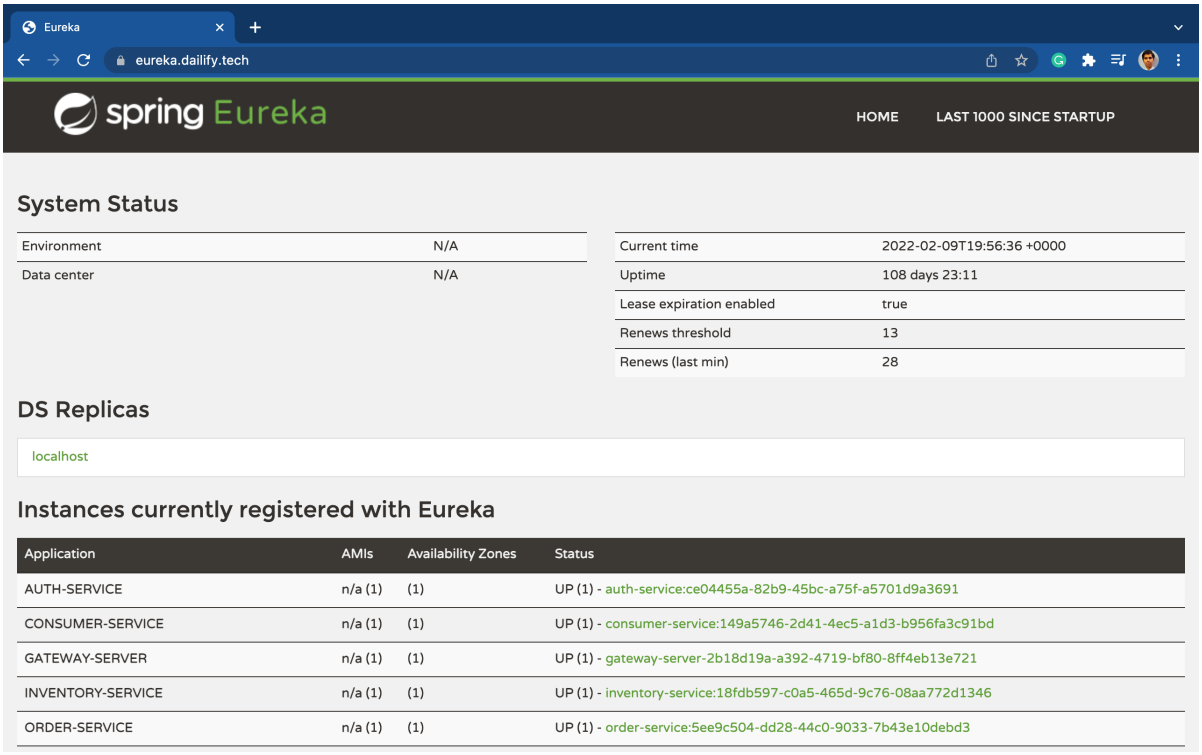


Figure 6.6: Eureka Server

Eureka Server Status currently stating the list of services right now deployed over Azure.

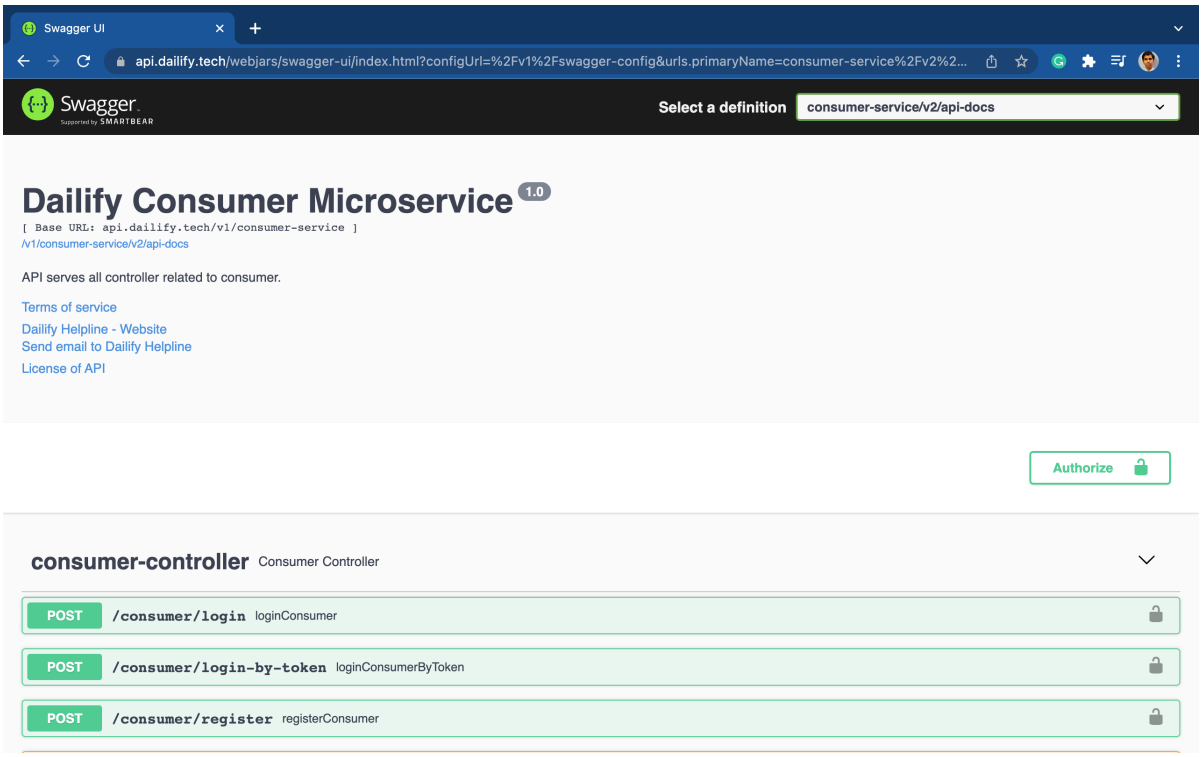


Figure 6.7: Open API Docs: Consumer Service
Consumer Service documentation page supported with Open API documentation or Swagger 3.0

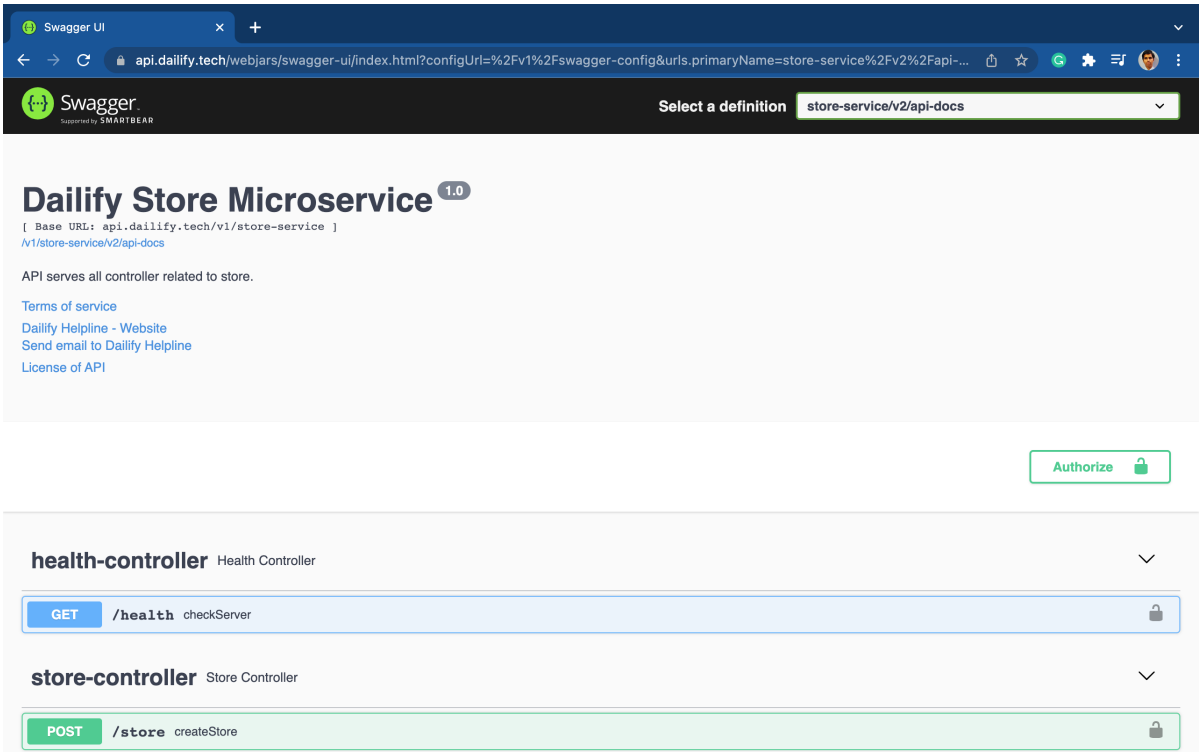


Figure 6.8: Open API Docs: Store Service
Store Service documentation page supported with Open API documentation or Swagger 3.0

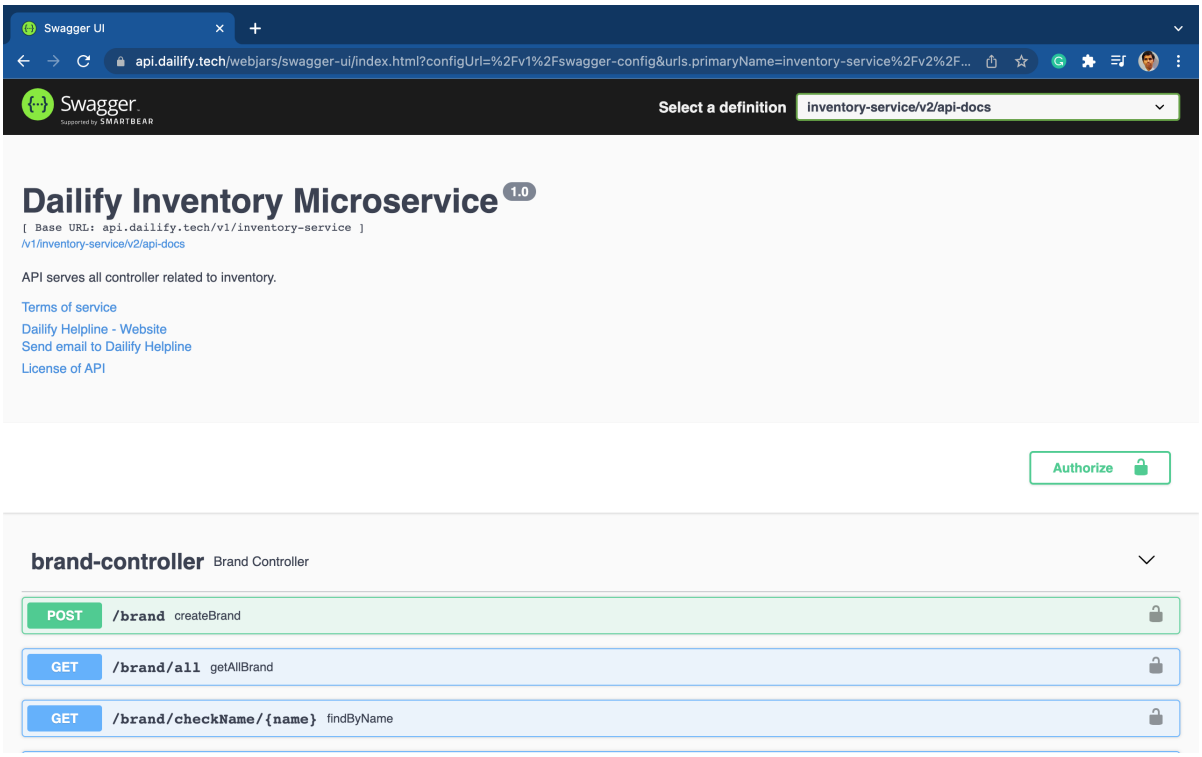


Figure 6.9: Open API Docs: Inventory Service
Inventory Service documentation page supported with Open API documentation or Swagger 3.0

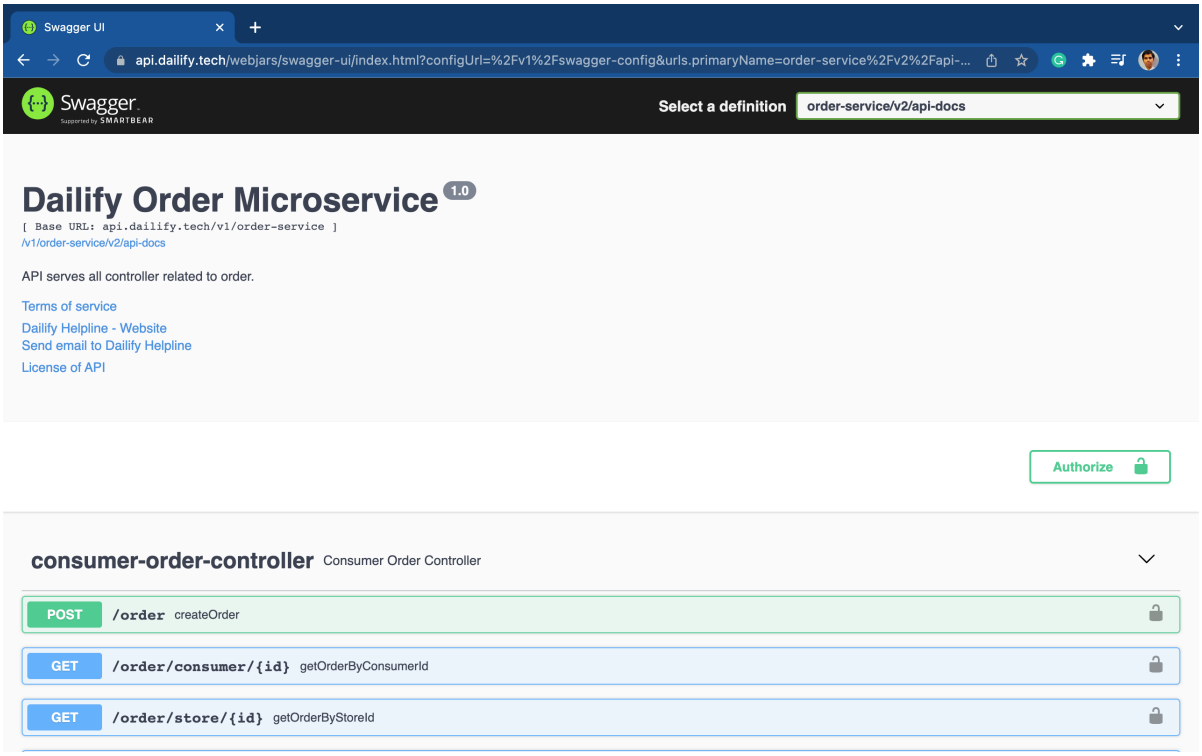


Figure 6.10: Open API Docs: Order Service
Order Service documentation page supported with Open API documentation or Swagger 3.0

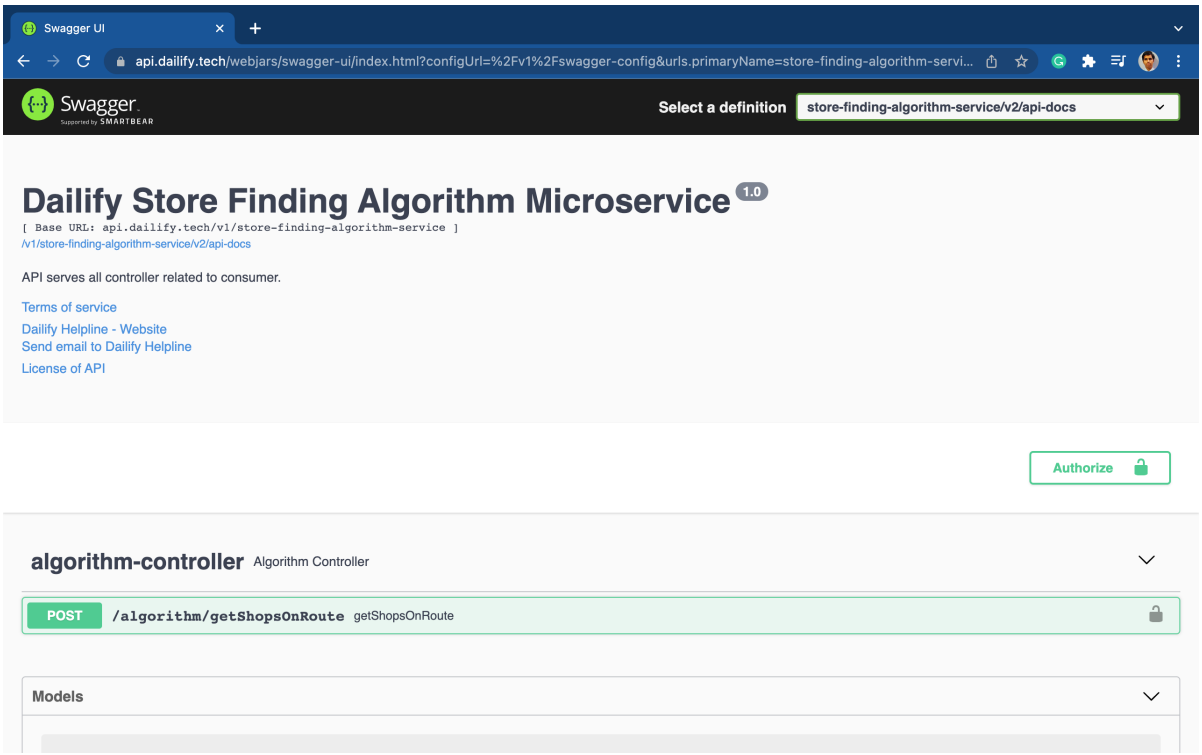


Figure 6.11: Open API Docs: Store Finding Algorithm Service
Store Finding Algorithm Service documentation page supported with Open API documentation or Swagger 3.0

6.3 Deployment Status

Backend is deployed on an Azure Virtual Machine instance currently running with Central-India Datacenter. More over we have registered ourselves with domain as dailify.tech

The complete CI/CD is done through GitHub Actions, which automatically deploy the backend code pushed on master branch. While the frontend is build and release and an APK of app to the Telegram Channel.

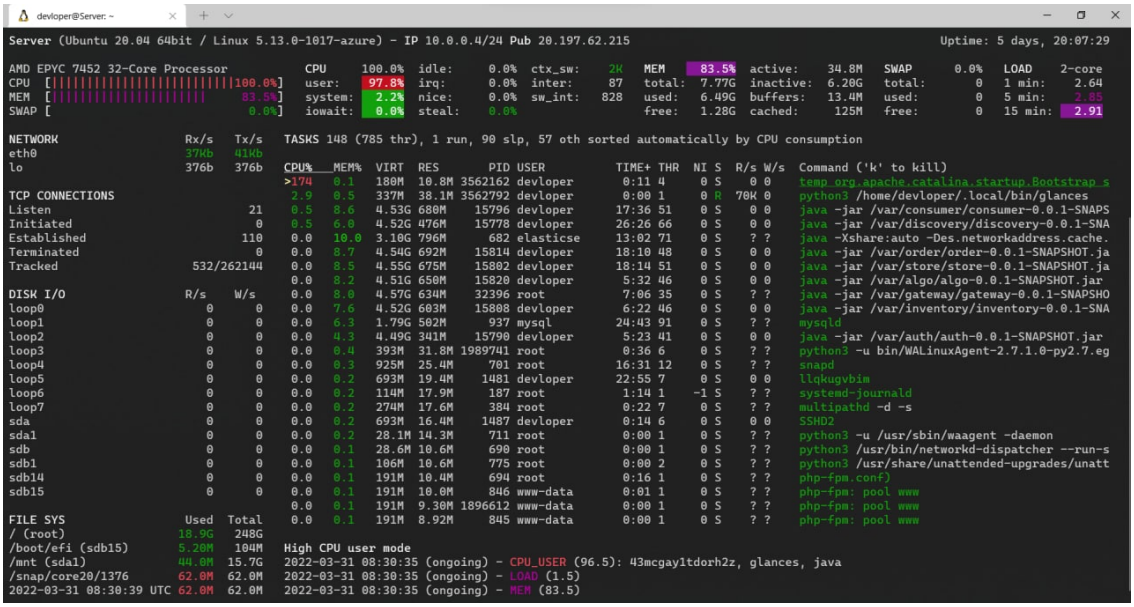


Figure 6.12: Azure VM Glances Output
The VM info deployed on Azure using glances command.

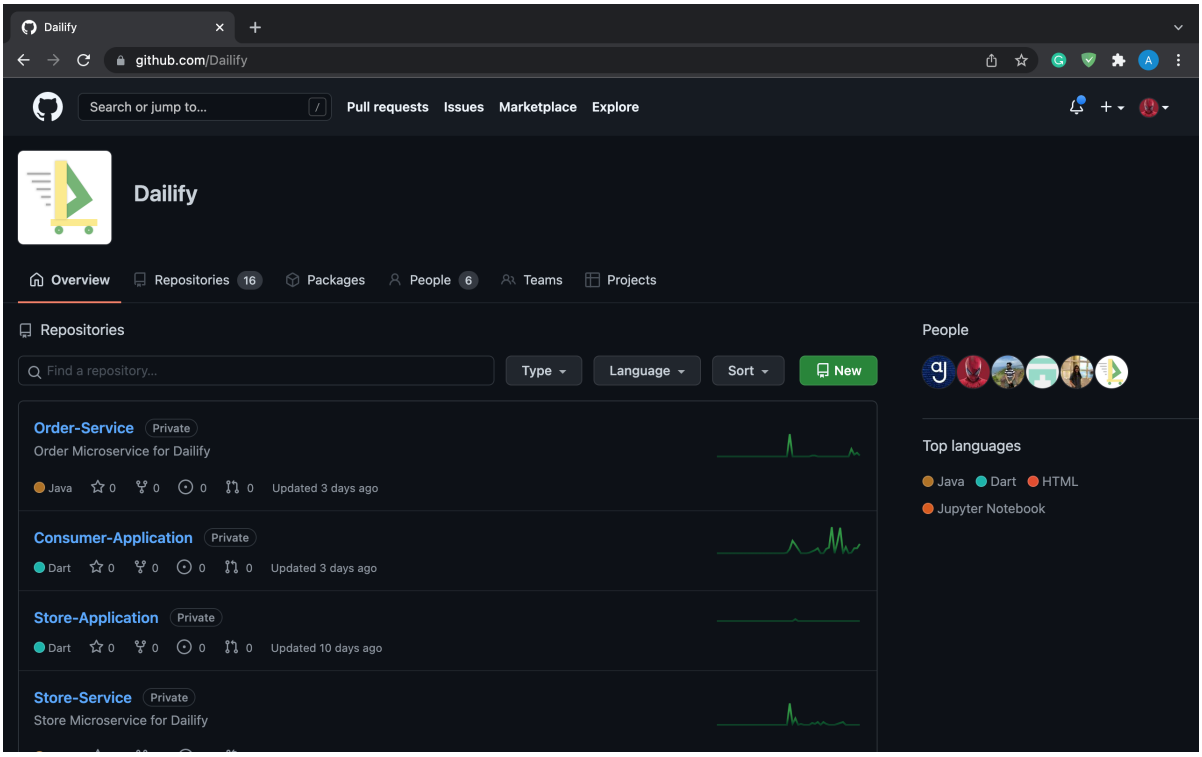


Figure 6.13: Github Organisation
Dailify Github Organisation with listed repositories.

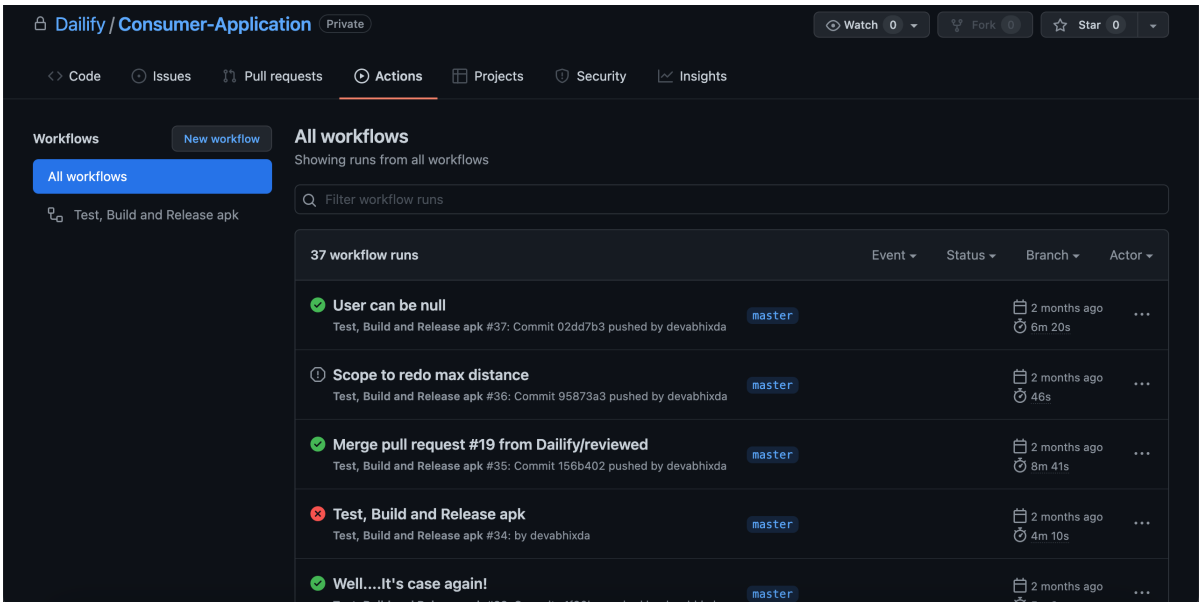


Figure 6.14: Consumer App CI/CD
Github action fetches the latest code, create an APK and send that to the Telegram Channel.

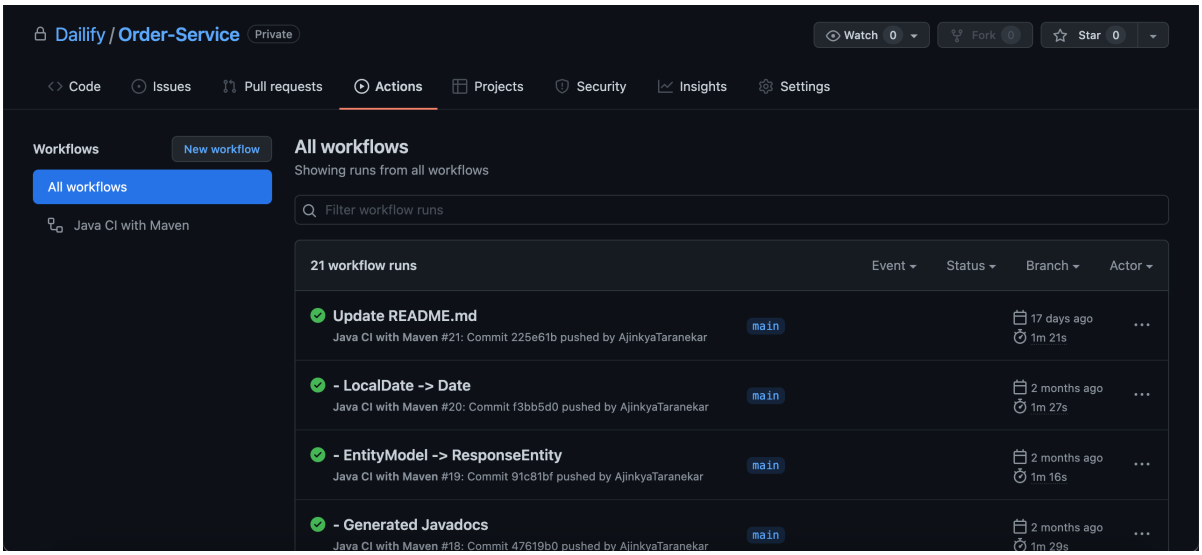


Figure 6.15: Order Service CI/CD
Github action fetches the latest code, creates a jar which is then deployed to the servers.

Chapter 7

TESTING & RESULT

In this chapter, the details of testing and result phase are covered. In the first section, the dataset used for the product is discussed. In the second section, we test main components i.e. product searching and store finding algorithm efficiency by comparing different approach. The last section has a survey of the final product, from the users themselves.

7.1 Datasets Used

7.1.1 Blinkit Inventory

To explore inventory we explored various retails and grocery selling applications, such as JioMart, Blinkit, BigBasket, DMart etc. But overall we concluded that Blinkit(formerly Grofers) inventory was quiet managed and cleaned up rather than rest of the three.

To scrap data from Blinkit, we spent a week analysing how they have written codes, what are network calls taking place, understanding cookies and APIs.

The data was fetched from a python script which cleaned the data map with out model and synchronise with our elasticsearch database.

The data analysis of database is as follows:

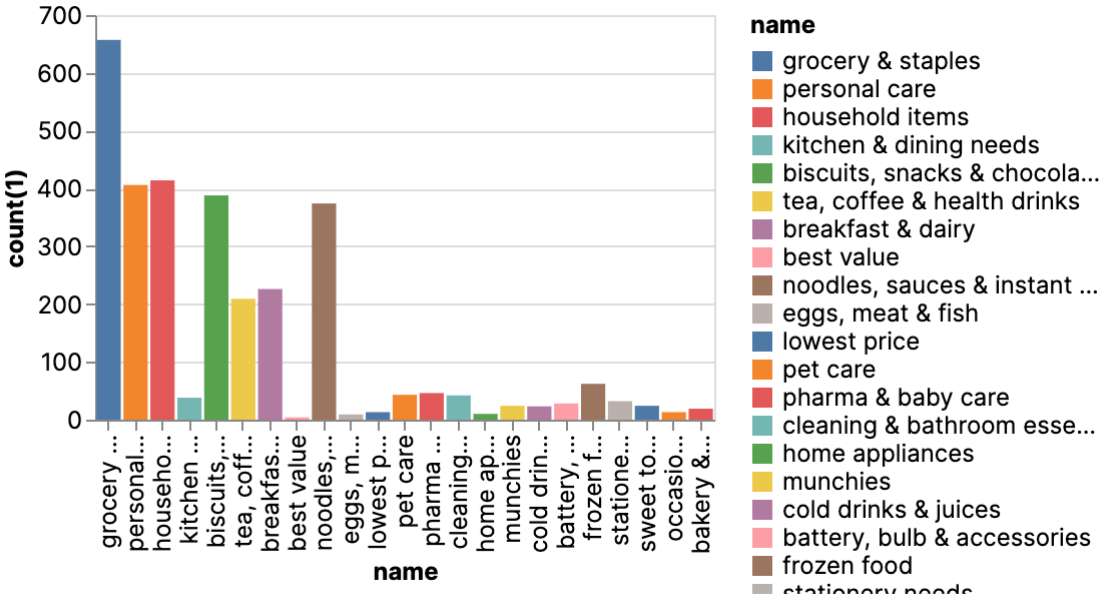


Figure 7.1: Inventory Group by Categories
Figure represents the inventory group by categories of products

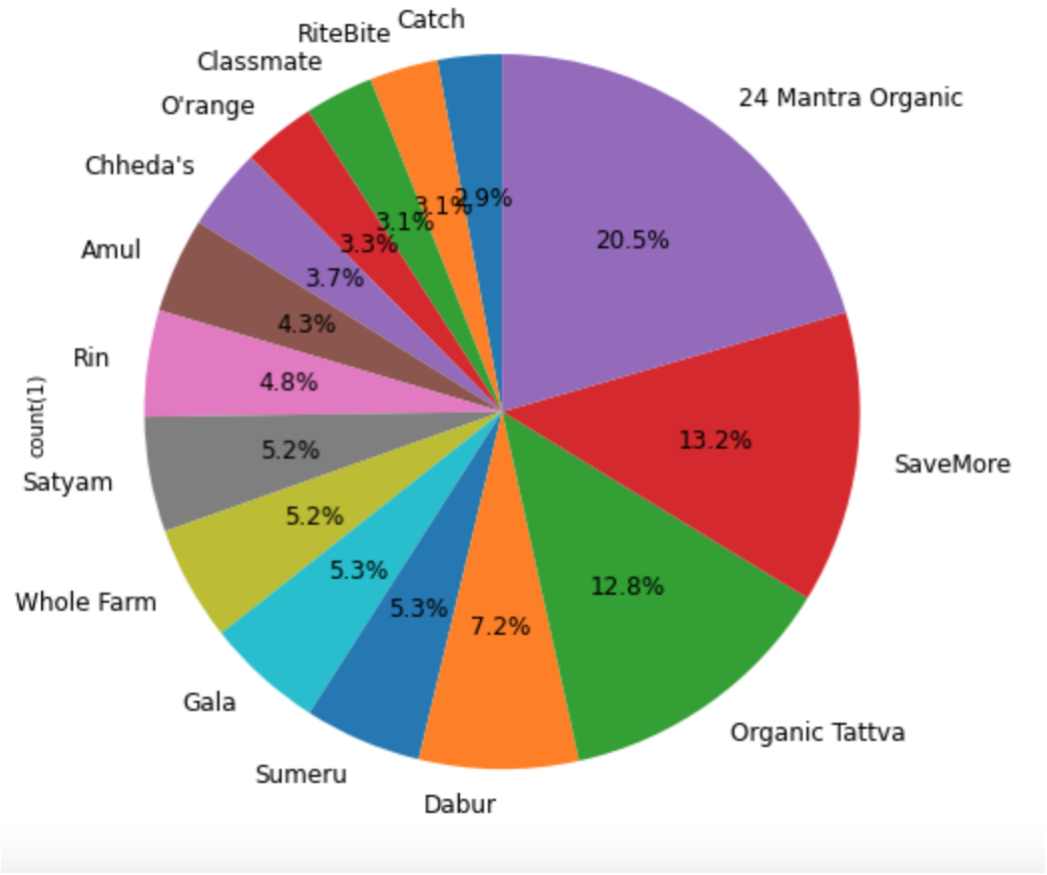


Figure 7.2: Top 15 brands in our Database
Figure represents the top 15 brands in our database

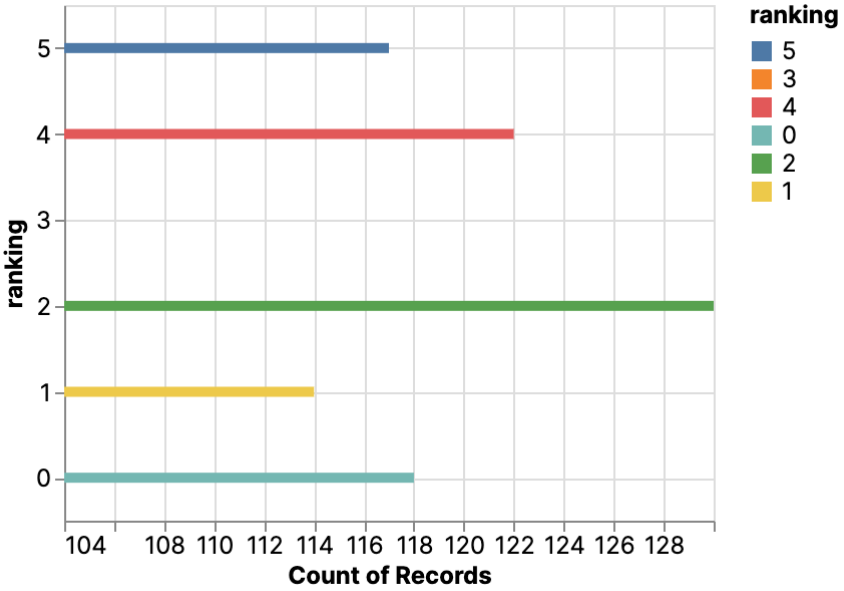


Figure 7.3: Inventory Group by Ranking of Products
Figure represents the inventory group by ranking of products

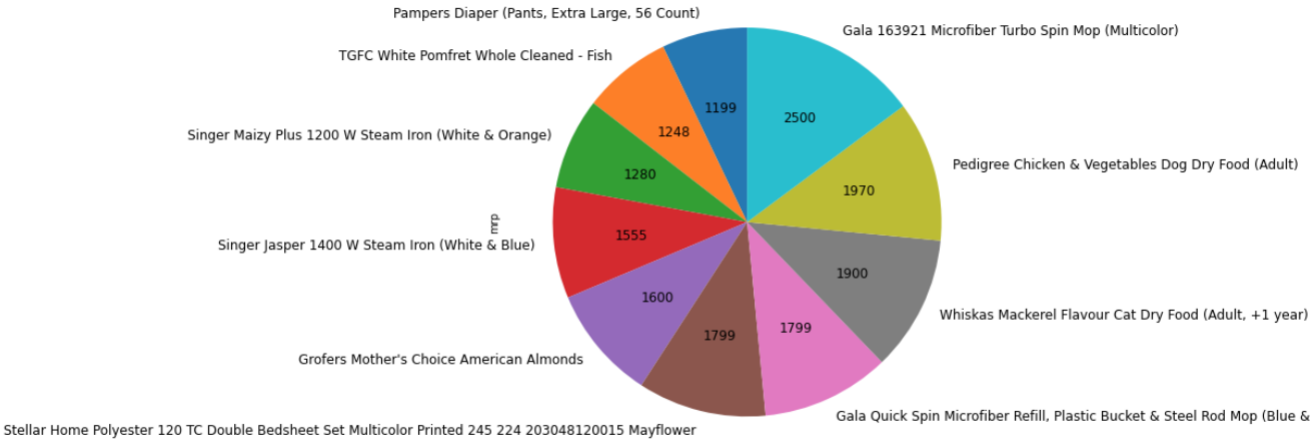


Figure 7.4: Top 10 Costly Products
Figure represents the top 10 costly products in inventory

7.1.2 Indore Grocery Stores

We collected grocery stores in Indore region using JustDial, for which the records were web scrapped. We got around 150 stores in Indore, for which data was analysed and cleaned. This data was inserted to our databases.

Information gathered during data analysis:

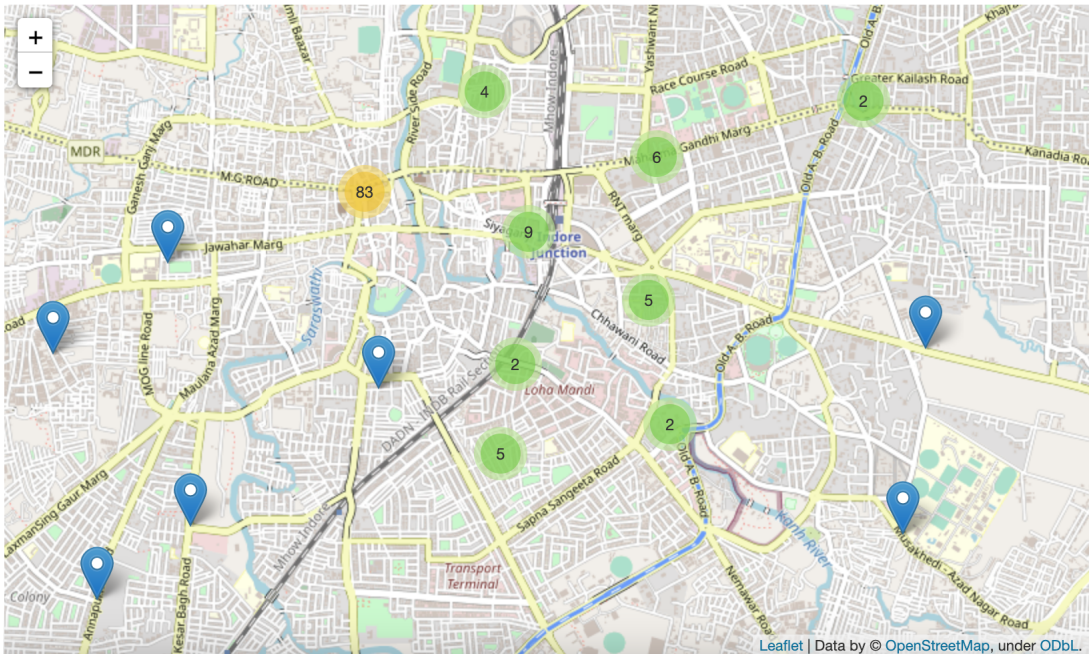


Figure 7.5: Grocery Shops in Indore
Figure represents the grocery shops map in Indore region created using folium library

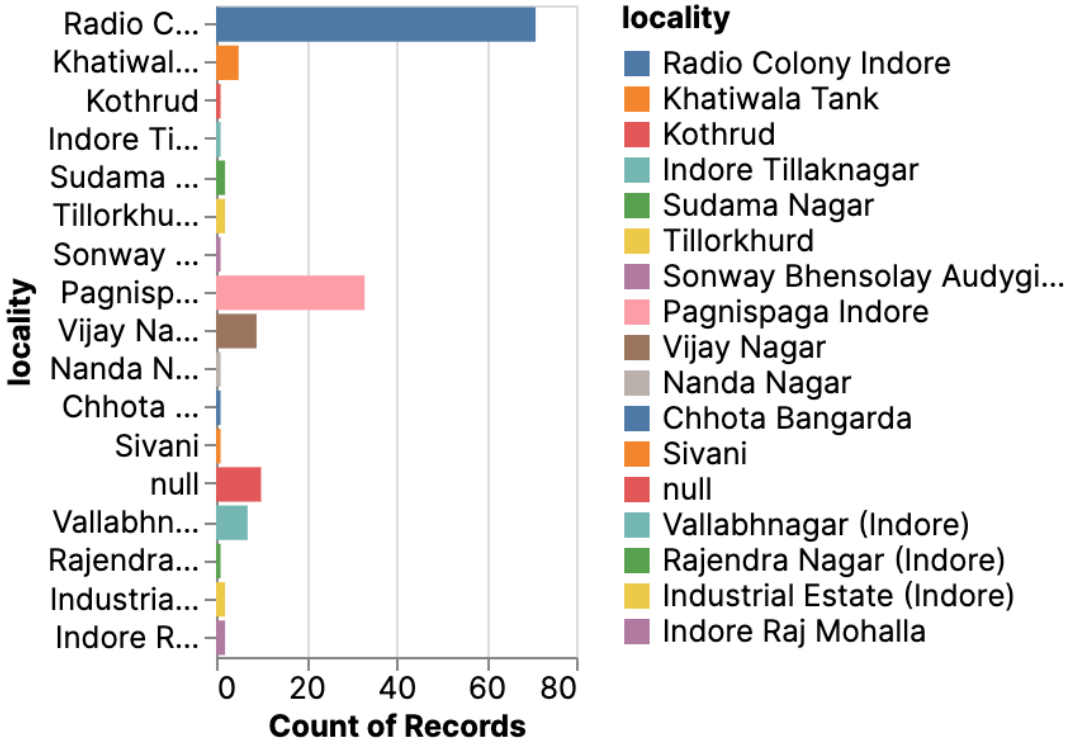


Figure 7.6: Grocery shops group by locality
Figure represents the grocery shops group by Indore Locality

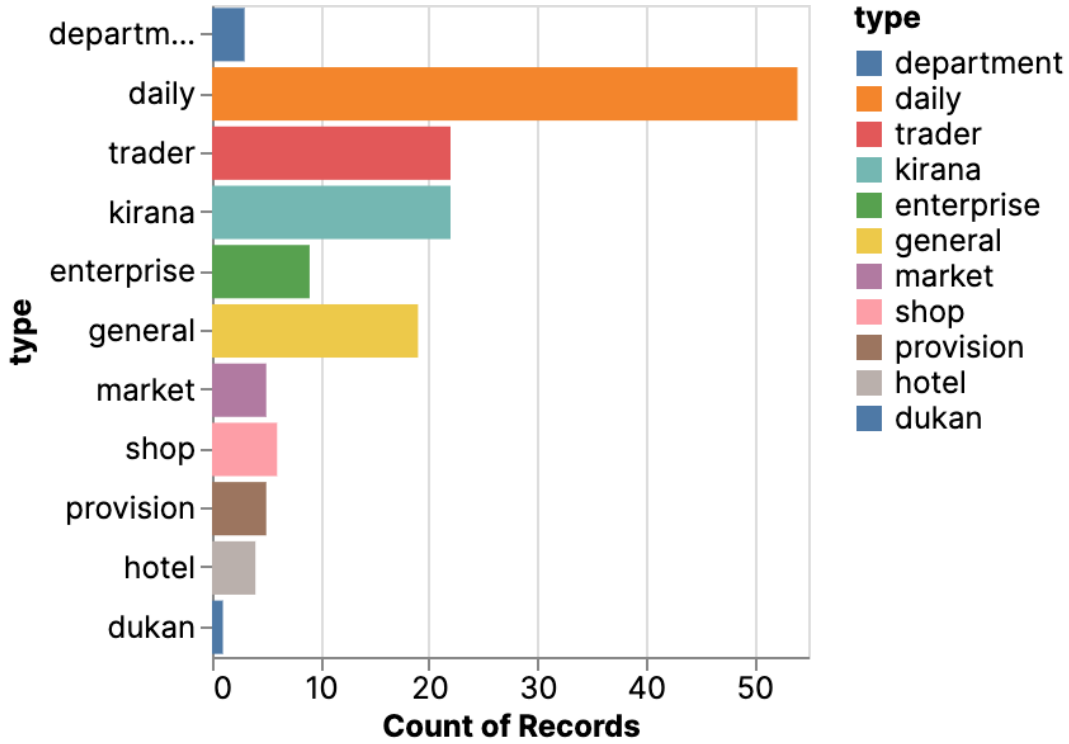


Figure 7.7: Grocery shops group by type
Figure represents the types of grocery shops in database

7.2 Testing and Improvements

Major improvement was introduction of Elasticsearch for product searching and indexing. This resulted in **85%** improvement in response time for searches.

The Fig 7.8 was an partial optimised searching using fuzzy matching of name "poha" in inventories name where we get result in 353 ms. But in Fig 7.9 ElasticSearch did the magic, now each searches were within 53 ms.

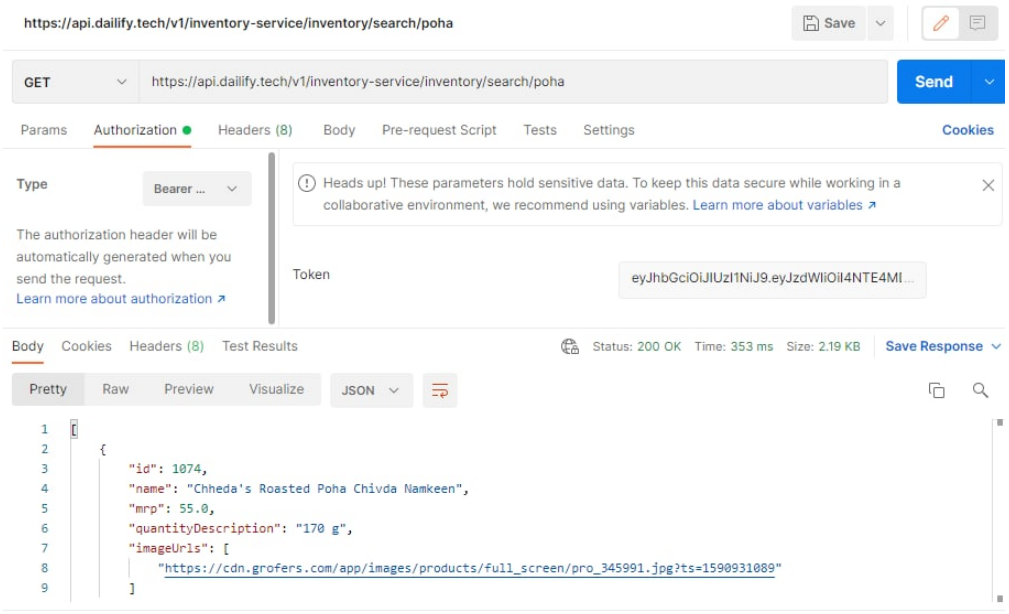


Figure 7.8: Inventory Searching without ES
Figure represents the postman testing of searching poha in database using Fuzzy Searching

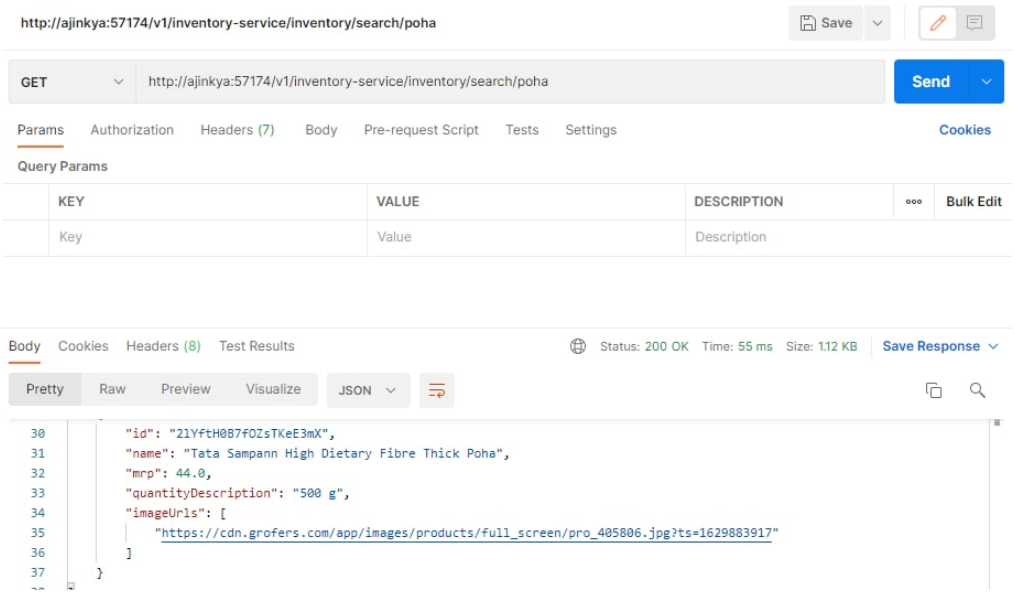


Figure 7.9: Inventory Searching with ES
Figure represents the postman testing of searching poha in database with ElasticSearch

The Fig 7.10 was an brute force approach in $O(n^3)$ time complexity, where we get result in 3-4 seconds. But in Fig 7.11 ElasticSearch introduction made querying lot easier, also we introduced Binary Searching, this made store finding algo to run in within 1 seconds or less.

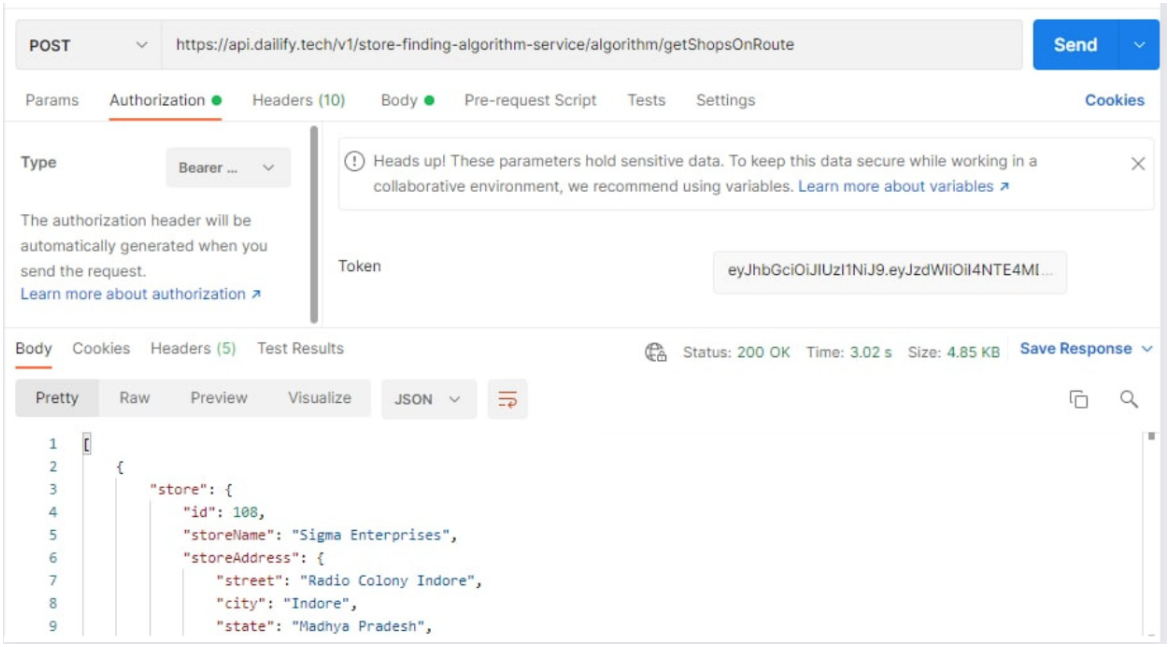


Figure 7.10: Store Finding Algorithm without Optimisations
Figure represents the postman testing of searching stores with given body in database without optimizations

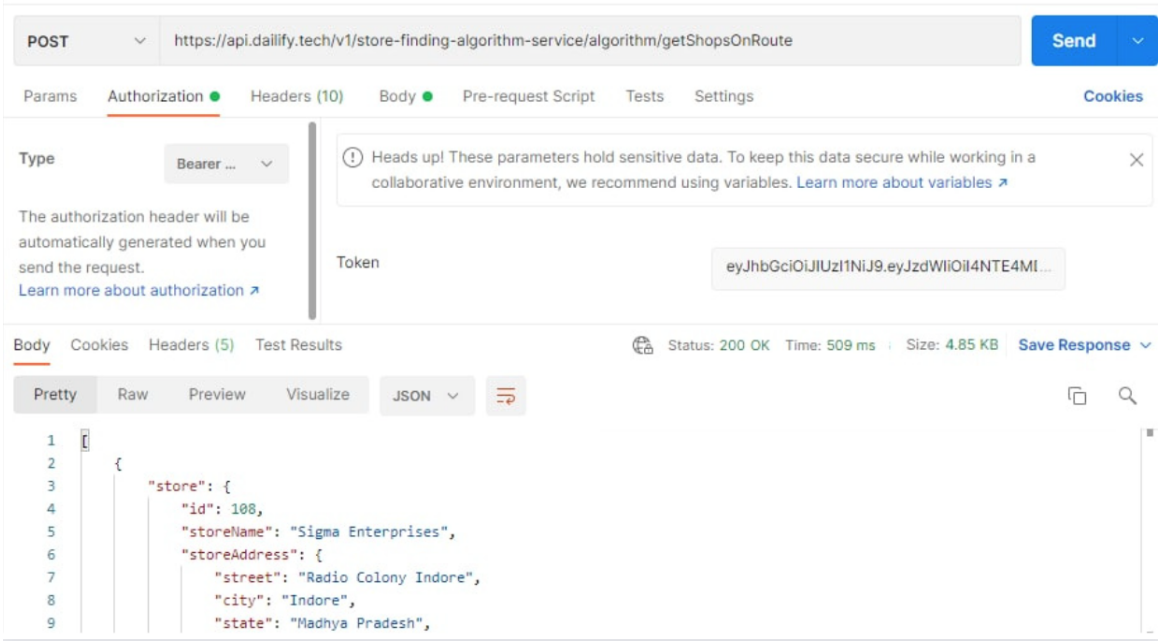


Figure 7.11: Store Finding Algorithm with Optimisations
Figure represents the postman testing of searching stores with given body in database with optimizations

7.3 Results

7.3.1 Consumer Acceptance

We sent the APK of our application to a group of 50 friends and relatives to test the field result of the application, each user was asked to use the application for a period of 15 days during their commute for shopping groceries. We also on boarded a group of 10 store owners to experience Dailify, at the end of 15 day period we conducted a survey to understand how Dailify has impacted the grocery shopping experience for users. Below figure shows the result of the survey.

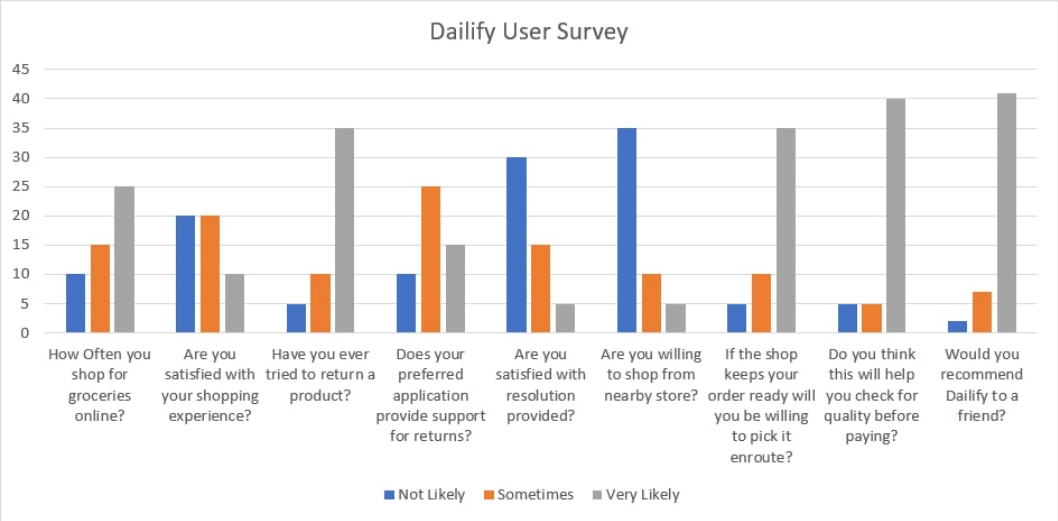


Figure 7.12: User Survey

The survey shows that people faced issue with product quality and returns but are still not willing to shop from nearby stores because of long queues and waiting time which was our main assumption. Curbside pickup solution helps to solve both of this problems which is evident from the survey results.

7.3.2 Store Owners Feedback

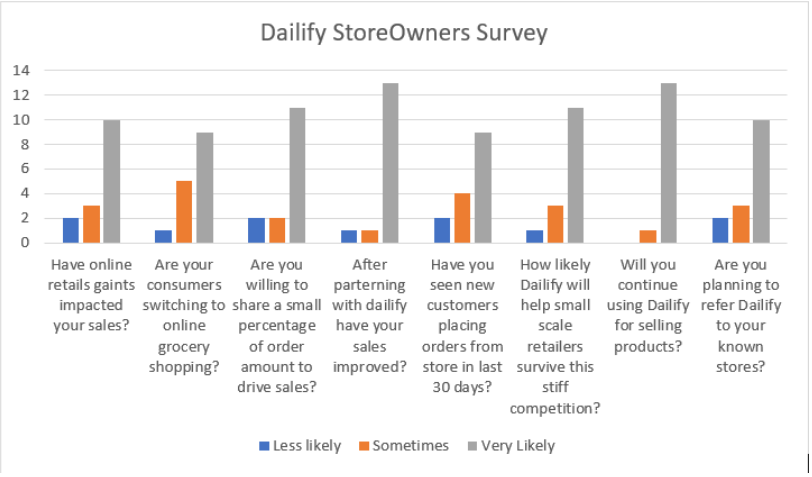


Figure 7.13: Store Survey

The survey shows that most of the owners are losing business due to Retails giants like JioMart, Grofers, Big Basket. People prefer convenience of online shopping(As significant from user survey). Empowering small scale shop owners with a platform to sell their products to customer conveniently ensures that customers keep coming back and the main drawing factor is price and quality of the goods offered.

Chapter 8

CONCLUSION

This chapter presents the conclusion of the work done in the analysis and design phase. In Section 8.1 the conclusion of the proposed approach is discussed. Section 8.2 explains the limitations of the proposed approach. The Future Scope of the proposed approach is described in Section 8.3.

8.1 Conclusion

The main aim of this project was to design and analyze the given problem statement, thus bringing a closer look at how grocery pickup service can be brought to India, also uplifting small scale businesses. Thus we present the Dailify to the market, where based on analysis and design a user could register, and check out a shopping list while moving from one place to another in just a few steps. Stores registered on Dailify will get more outreach along with a newer set of customers. This will surely act as a medium for promotion in the online world. The proposed approach uses an optimized brute force approach towards matching stores and predicting the optimal route for the customer using Binary Search and Haversine Algorithm method, along with using ElasticSearch as Database for Inventory and Store Products Map. Which takes a total of less than 800 milliseconds to find stores.

8.2 Limitations of Proposed Approach

The proposed approach has the following limitations:

- The proposed approach towards the algorithm works only when we find a shop that has all products.
- The proposed approach emphasises more on improving time complexity, along with an efficient way to store products.

8.3 Future Enhancements

Enhancements that can be done in future are as follows:

- Implementation of the message queue to notify stores when an order is placed.
- Rightnow UPI payment is added, we will add proper payment gateway to app, such that payment is first done to us than based on commission we will redirect to store's wallet.

Appendix A

A.1 Response Time

Response Times: The 3 Important Limits

by Jakob Nielsen on January 1, 1993

Summary: There are 3 main time limits (which are determined by human perceptual abilities) to keep in mind when optimizing web and application performance. Excerpt from Chapter 5 in my book Usability Engineering, from 1993:

The basic advice regarding response times has been about the same for thirty years [Miller 1968; Card et al. 1991]:

- **0.1 seconds** is about the limit for having the user feel that the system is reacting instantaneously, meaning that no special feedback is necessary except to display the result.
- **1.0 seconds** is about the limit for the user's flow of thought to stay uninterrupted, even though the user will notice the delay. Normally, no special feedback is necessary during delays of more than 0.1 but less than 1.0 seconds, but the user does lose the feeling of operating directly on the data.
- **10 seconds** is about the limit for keeping the user's attention focused on the dialogue. For longer delays, users will want to perform other tasks while waiting for the computer to finish, so they should be given feedback indicating when the computer expects to be done. Feedback during the delay is especially important if the response time is likely to be highly variable since users will then not know what to expect.

A.2 Haversine Formula

The haversine formula determines the great-circle distance between two points on a sphere given their longitudes and latitudes. Important in navigation, it is a special case of a more general formula in spherical trigonometry, the law of haversines, that relates the sides and angles of spherical triangles.

Haversine formula:

$$a = \sin^2(\Delta\varphi/2) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda/2)$$
$$c = 2 \cdot \text{atan2}(\sqrt{a}, \sqrt{(1-a)})$$
$$d = R \cdot c$$

where φ is latitude, λ is longitude, R is earth's radius (mean radius = 6,371km); note that angles need to be in radians to pass to trig functions!

Figure A.1: Haversine Formula
Haversine Formula used to calculate stores near the given points on route

A.3 ElasticSearch

Elasticsearch (ES) is a database that provides distributed, near real-time search and analytics for different types of data. It is based on the Apache Lucene™ library and is developed in Java. It works on structured, unstructured, numerical and geospatial data. The data is stored in the form of schema-less JSON documents.

A.3.1 Features it supports:

Some of the major features that Elasticsearch has to offer are:

- Lightening fast full-text search.
- Security analytics and infrastructure monitoring.
- Can be scaled to thousands of servers and can handle petabytes of data.
- Can be integrated with Kibana to provide real-time visualisation of Elasticsearch.
- Use of machine learning to automatically model the behaviour of your data in real-time.

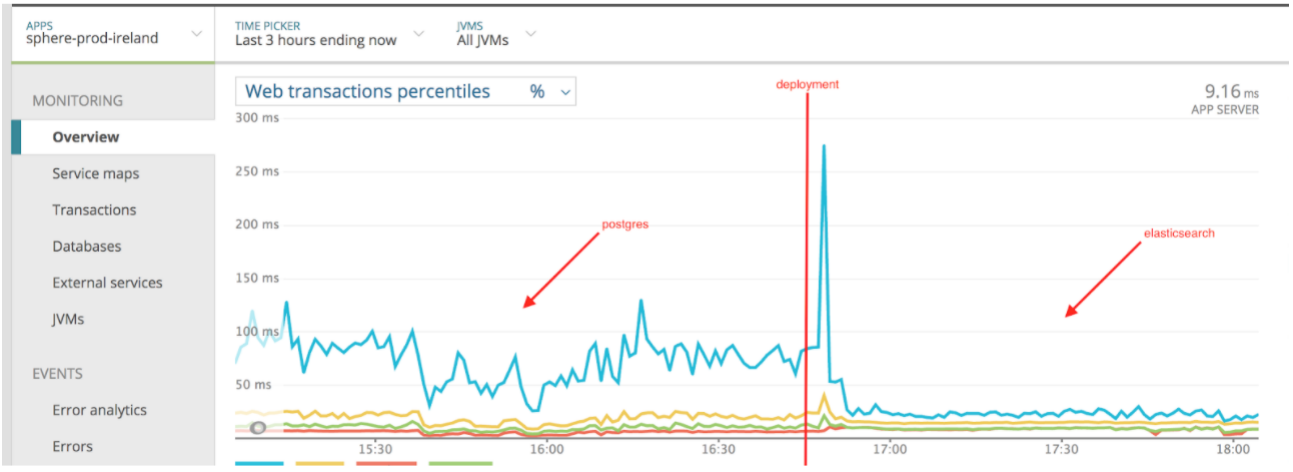


Figure A.2: Elastic Search v/s PostgreSQL BenchMark
Comparison between time on response from elastic search and postgre sql.

Bibliography

[1] Fotouhi, Hossein, et al. 'Assessing the Effects of Limited Curbside Pickup Capacity in Meal Delivery Operations for Increased Safety during a Pandemic'. Transportation Research Record: Journal of the Transportation Research Board, vol. 2675, no. 5, May 2021, pp. 436–52. DOI.org (Crossref), https://doi.org/10.1177/0361198121991840.

[2] 85% of Shoppers Have Increased Curbside Pick-Up Since COVID-19, 79% Say a Contactless Store Pickup Is Very Important to Them. 24 Sept. 2020, https://www.businesswire.com/news/home/20200924005537/en/85-of-Shoppers-Have-Increased-Curbside-Pick-Up-Since-COVID-19-79-Say-a-Contactless-Store-Pickup-is-Very-Important-to-Them.

[3] Narendra. 'Architecture and Design Principles behind the Swiggy's Delivery Partners App'. Medium, 13 Nov. 2019, https://bytes.swiggy.com/architecture-and-design-principles-behind-the-swiggys-delivery-partners-app-4db1d87a048a.

[4] Sengar, Ravinder Singh. 'E-Commerce/Food Delivery App (Zomato/Swiggy/1mg)'. Medium, 3 Feb. 2022, https://medium.com/@ravindersengar/end-to-end-design-of-a-food-delivery-app-like-zomato-or-swiggy-52b96f3a635f.

[5] 'Increased Use of Online Grocery Shopping "Here to Stay"'. Supermarket News, 25 Aug. 2021, https://www.supermarketnews.com/online-retail/increased-use-online-grocery-shopping-here-stay.

[6] '6 Best Grocery Pickup Services'. Clark Howard, 8 Apr. 2020, https://clark.com/shopping-retail/best-grocery-pickup-services/.