Project Report

on

Khaas Food (Online Grocery)

By

Robin Jose

Under the Guidance of

Prof. Leena Deshmukh

Master of Computer Application

P.E. Society's

Modern Institute of Business Studies, Nigdi, Pune-44

Savitribai Phule Pune University

2021-22

Seema Apartment, Office. 4, behind Rahul Theatre,
Shivajinagar, Pune, Maharashtra 411005
www.gtlinfotech.com , Email : gtlinfotech1@gmail.com
Contact : 9699775400 / 9623034217

INTERNSHIP COMPLETION LETTER

This is to certify that **Mr. Robin Jose**, Master of Computer Application student final year at the P.E.S. Modern Institute of Business Studies, Yamunanagar Nigdi, Pune Maharashtra has completed a partial project titled "**Khaas Food**" in Java Technology with GTL Infotech Shivajinagar Pune as part of his fourth-semester project.

The duration of his project was from 04 April 2022 to 10 August 2022 under the guidance of Mr. Amit Sonar and Mr. Abhishek Kumar.

Note: In accordance with our privacy policies, we are unable to provide source code or data to interns.

Best Regards,
Amit Sonar
GTL Infotech
www.gtlinfotech.com
gtlinfotech1@gmil.com
9623034217

Progressive Education Society's

Modern Institute of Business Studies

Nigdi, Pune-44

Certificate

This is to certify that project entitled **Online Grocery Shopping** for **GTL INFOTECH** is submitted by **Robin Jose** a student of **Master of Computer Application** has successfully completed the MCA-IV semester Project during academic year **2021-22.**

This report is submitted as partial fulfillment of the requirement of degree in **MCA** as per the rules of Savitribai Phule Pune University, Pune.

Dr. Maithili Arjunwadkar	**Internal Guide**	**External Examiner**
Director	**Prof. Leena Deshmukh**	
PES MIBS		

ACKNOWLEDGEMENT

It is my proud privilege to express my profound gratitude to the entire management of Progressive Education Society's Modern Institute of Business Studies and teachers of the institute for providing me with the opportunity to avail the excellent facilities and infrastructure of the institute. The knowledge and values inculcated have proved to be of immense help to my career.

I am grateful to **Dr. Maithili Arjunwadkar (Director, MIBS)** for their astute guidance, constant encouragement and sincere support for this project work.

I express my sincere gratitude to **Prof. Leena Deshmukh, MIBS** for their inspiration, constructive suggestion, mastermind analysis and affectionate guidance in my work, without which this project work completion would have been impossible for me. Sincere thanks to all my seniors and colleagues at the company for their support and assistance throughout the project.

I would like to thank all the faculties for providing me with an opportunity to pursue my industrial training, as it is an important part of the MCA course.

I feel proud and privileged in expressing my deep sense of gratitude to all those who have helped me in presenting this assignment. I would be failing in my endeavor if I do not place my acknowledgement.

DECLARATION

I the undersigned solemnly declare that the report of the project work entitled **Khaas Food (Online Grocery)** is based my own work carried out during the course of my study under the supervision of **Prof. Leena Deshmukh** I assert that the statements made and conclusions drawn are anoutcome of the project work.

I further declare that to the best of my knowledge and belief that the project report does not contain any part of any work which has been submitted for the award of any other degree/diploma/certificate in this University or any other University.

Robin Jose

INDEX

1 : Introduction

1.1 Company Profile

"GTL Infotech" is a leading software development company based in Pune.
With the establishment of GTL Infotech in 2010, we aimed to create a work environment that is informal yet highly professional. We encouraged our staff to identify the organization's goals as their own.
Using the latest technology to provide efficient and cost-effective information management solutions. In every aspect of our operations, we strive for excellence.
As a company, we are highly motivated to provide professional, high-quality software.

Our Aim: "To be automated and to see automated"

About our software services and products

We recognized the potential of the Indian Software Industry during its infancy and developed various enterprise applications and off-the-shelf software products tailored to customer requirements.

- ✓ Business Soft
    - Hotel Management Software
    - Hospitality Software
- ✓ Retail Soft
    - Civil Inventory System
    - EIS
- ✓ Edu Soft
    - Library Management System
    - Accounting System
- ✓ Android Applications

1.2 Abstract

It has been developed with the primary purpose of selling grocery items online through an e-commerce website with the aim of making it as convenient as possible for consumers to buy groceries online.
To ensure that the project has the maximum impact on the local level, it has been designed specifically for the Bangladesh country. The convenience of shopping for groceries online in Bangladesh is unmatched. You can order groceries online with cash on delivery in Dhaka, Chittagong, as well as other cities in Bangladesh.

It is now possible to order food online and have it delivered to your home. With this project, Online E-Commerce Shopping, we are providing all the facilities to assist the shopkeeper in managing the shop on the website as well as controlling the entire e-Commerce business transactions from the website.

In addition, it can help admin keep track of what items are available for sale on a daily basis and keep their site updated with new products as they become available. The purpose of this project is to manage various product categories, products, special offers, galleries, orders, payments, and customer feedback.

Throughout this project, customers will be able to view a variety of products, add them to their cart, and place an order with ease. Throughout the course of the project, we have developed a lot of modules in order to perform all of the operations of this system.

There are several modules that can be used to manage your products, your types, your orders, and so on.

1.3 Existing System and Need for System

Existing System

- The user visits a supermarket to purchase products, where the store offers limited alternatives.
- In order to add the items to their cart, they compare the brand and packaging of the product.
- To make payment for the products selected by the user, the user must wait in line at the checkout counter
- In some cases, the product the user is looking for may not be available in the store, so he or she will need to travel again to purchase the product.

Need for System

- It takes a considerable amount of time to shop for groceries offline
- The supermarket does not operate on a 24/7 basis
- Reviewing, comparing, and contrasting products based on their brands, descriptions, and reviews
- Need for hassle-free checkout with online payment and cash-on-delivery option
- A real-time notification service based on the availability of the product
- The delivery of groceries from a store to a residence is required
- Reduction of carbon emissions in order to improve the environment

1.4 Scope of System

- The system should help customers to search for and compare products

- In addition to viewing the products, customers have the opportunity to view real-time feedback from other customers.
- The system should allow customers to pay using their desired payment mode: Online or COD
- The system should allow the customer to view and give feedback on past orders

- The system should allow the admin to add and manage, products and categories

- The system should allow the admin to add, manage delivery partners and assign orders.

1.5 Operating Environment – Hardware and Software

Software Specifications

- Back-end Language : Java 17 (Spring Boot 2.4)
- Front-end Language : TypeScript (Angular 8)
- IDE : IntelliJ IDEA, VS Code
- Database : PostgreSQL 14
- Server : Embedded Tomcat

Hardware Specifications

- Hard Disk : 10 GB (minimum)
- Processor : Intel i5+ / AMD Ryzen 5+
- RAM : 4 GB (minimum)
- Hosting : Linux Hosting/ Window Hosting

1.6 Detail Description of Technology Used

- **Java**

Java is a programming language and computing platform first released by Sun Microsystems in 1995. It has evolved from humble beginnings to power a large share of today's digital world, by providing the reliable platform upon which many services and applications are built. New, innovative products and digital services designed for the future continue to rely on Java, as well. Simple.

- **Spring-Boot**

Spring Boot is an open-source Java-based framework used to create a micro-service. It is developed by Pivotal Team and is used to build stand-alone and production ready spring applications. This chapter will give you an introduction to Spring Boot and familiarizes you with its basic concepts.

Spring Boot automatically configures your application based on the dependencies you have added to the project by using *@EnableAutoConfiguration* annotation. For example, if MySQL database is on your class path, but you have not configured any database connection, then Spring Boot auto-configures an in-memory database.

The entry point of the spring boot application is the class contains *@SpringBootApplication* annotation and the main method.

Spring Boot automatically scans all the components included in the project by using *@ComponentScan* annotation.

- **HTML 5**

A servlet is a web component, managed by a container that generates dynamic content. Servlets are small, platform independent Java classes compiled to an architecture neutral byte code that can be loaded dynamically into and run by a web server. Servlets interact with web clients via a request response paradigm implemented by the servlet container. This request-response model is based on the behavior of the Hypertext Transfer Protocol (HTTP)

- **TypeScript**

TypeScript is a free and open-source programming language developed and maintained by Microsoft. It is a strict syntactical superset of JavaScript and adds optional static typing to the language. It is designed for the development of large applications and transpiles to JavaScript. TypeScript may be used to develop JavaScript applications for both client-side and server-side execution (as with Node.js or Deno). Multiple options are available for transpilation. The default TypeScript Compiler can be used, or the Babel compiler can be invoked to convert TypeScript to JavaScript.

- **Angular**

Angular is a development platform, built on TypeScript. As a platform, Angular includes: A component-based framework for building scalable web applications A collection of well-integrated libraries that cover a wide variety of features, including routing, forms management, client-server communication, and more. A suite of developer tools to help you develop, build, test, and update your code.

- **PostgreSQL**

PostgreSQL is a powerful, open-source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads. The origins of PostgreSQL date back to 1986 as part of the POSTGRES project at the University of California at Berkeley and has more than 30 years of active development on the core platform.

PostgreSQL has earned a strong reputation for its proven architecture, reliability, data integrity, robust feature set, extensibility, and the dedication of the open source community behind the software to consistently deliver performant and innovative solutions. PostgreSQL runs on all major operating systems, has been ACID-compliant since 2001, and has powerful add-ons such as the popular PostGIS geospatial database extender. It is no surprise that PostgreSQL has become the open-source relational database of choice for many people and organizations.

With PostgreSQL, developers can build applications, administrators can protect data integrity and build fault-tolerant environments, and anyone can manage data no matter how big or small. As well as being free and open source, PostgreSQL is highly extensible. Without recompiling your database, you can define your own data types, build custom functions, or even write code from different programming languages!

2: Proposed system

2.1 Study of Similar Systems (If required research paper can be included)

- **Chris Hand, Francesca Dall'Olmo Riley, Patricia Harris, Jaywant Singh and Ruth Rettie - Online grocery shopping**

  According to Keynote (2007), "the UK is considered to have one of the world's most developed Internet grocery industries". Yet, while the online food and grocery market is reported to have grown by almost 34 per cent in 2006, online grocery purchases are estimated to still account for only 1.6 per cent of total UK grocery sales (Keynote, 2007).

  In contrast, overall internet sales in the same year accounted for as much as 10 per cent of all retail sales (BBC News, 2008). Mintel (2007) conclude that online shopping for food remains a niche market. The fast growth rate of the online grocery market presents a challenge for supermarket chains competing for share, in terms of balancing their online and offline investments.

  A better understanding of the triggers which influence the adoption (and the discontinuation) of online grocery shopping is vital for the strategic management of this sector, both in the "more developed" UK market and elsewhere.

  The aim of our research is to identify triggers for the adoption and discontinuation of online grocery shopping. Specifically, the objective of this study is to establish the role of situational factors in the process of adoption of online grocery shopping behaviour. This knowledge will assist online retailers in their customer recruitment and retention efforts.

  Our research makes a significant contribution to the literature by extending knowledge of online purchase behaviour beyond the much-researched influence of consumer traits, assessing the impact of situational factors and their role as triggers for the adoption of online grocery shopping.

  Shopping for groceries online is arguably a discontinuous innovation (Hansen, 2005), requiring a significant change in behaviour (see Robertson, 1967): online shoppers forfeit the social interaction of offline shopping and the potential to evaluate groceries prior to purchase.

For online grocery shopping to develop beyond its current "niche" size, retailers need to understand not only what triggers consumers to change their purchase behavior, but also the extent to which their online shopping experience reinforces the adoption process.

.

- **KAVITHA.R - A Study on Consumers acuity towards online grocery shopping**

EMERGENCE OF E-GROCERY

Groceries have been traditionally bought in stores, where consumers have an opportunity to touch and inspect products before the purchase and also control the transportation of fragile products. E-grocery retailing is still at the infancy stage and the volume of internet groceries buying leaves much to be desired for e-grocers in India to sustain profitable growth in the long run. The thought of purchasing perishable products including fish over the internet provokes some skepticism among most customers as the reliability and accountability of the service comes into question. E-grocery retailers can build brand equity, generating repeat business and this will result in market penetration in this industry. The real benefit of online grocery shopping is convenience. By ordering online, one can quickly search for the products one needs and order them without having to physically walk through those long aisles.

OBJECTIVES OF THE STUDY

To ascertain the consumer perception towards online grocery shopping.

To determine whether online grocery shopping will be beneficial and on what factors.

To determine whether consumers are aware of the availability of online grocery sites.

To analyze whether product and service quality affect consumers buying behavior online.

SCOPE OF STUDY

E-grocery is happening in India in a big way. The E-shopping will

replace traditional in-store shopping in the near future. Though the traditional shopping is still to stay but the online buying could change the way people experience shopping grocery. The study is to identify whether people are changing the way they shop their food and grocery from the next door Kirana store to online web stores. The study is conducted to reveal the consumer's attitude on e-grocery and their preference towards the same. This study will help to find out whether e-grocery will exceed the preference of using the traditional way of shopping and increase the response of online grocery shopping in future. There are numerous opportunities for innovative new services.

METHODOLOGY

The survey was administered to explore consumers' perceptions of Online Grocery Shopping in Coimbatore. An online questionnaire was developed to collect the required data in the most efficient manner possible. The questionnaire was distributed via electronic mail to the potential participants. The unit of analysis was any grocery consumer with or without experience in Online Grocery Shopping who lives in Coimbatore

- **Shipra Agrawal , Snehal, Tushar Kandhar - CONSUMER PERCEPTION TOWARDS ONLINE GROCERY SHOPPING .**

The Online grocery retailing or e-tailing can be branched into two parts: a supermarket that also delivers products or a standalone e-commerce company that only undertakes delivery activities. They usually charge a delivery fee for the service provided. While online grocery retailing still remains the smallest retail channel having only 0.2% of the market share, it is the one that is growing the fastest and has the highest potential. At present, analysts estimate online grocery sales between $500 million to a little over $1 billion with expectations of sales growing to over $5 billion in the next 3-4 years in India.

Online grocery retailing or e-tailing can be branched into two parts: a supermarket that also delivers products or a standalone e-commerce company that only undertakes delivery activities. They usually charge a delivery fee for the service provided. While online grocery retailing still

remains the smallest retail channel having only 0.2% of the market share, it is the one that is growing the fastest and has the highest potential. At present, analysts estimate online grocery sales between $500 million to a little over $1 billion with expectations of sales growing to over $5 billion in the next 3-4 years in India.

With the increasing internet availability and rising demand for budget smartphones, online retailing took its first steps into India and a number of companies emerged offering products at competitive prices and earned consumer trust through timely and assured door delivery of their products. Since the launch of companies like Bigbasket and Grofers, India has seen a hike in online grocery shopping which has impacted the traditional shops and store retailing. With this came the online payment system that changed the game of online retailing. Technology, efficient management of logistics and large investments became the key driver of online retailing in India. The increasing penetration of companies like Bigbasket and Amazon into the Indian market for online grocery shopping leaves us in thought if the Indian market is ready to accept the change towards the organised sector

2.2 Feasibility Study

A feasibility study is an evaluation of a proposal designed to determine the difficulty in carrying out a designated task. Generally, a feasibility study precedes technical development and project implementation.

- **System Feasibility**

The assessment is based on an outline design of system requirements in terms of Input, Processes, Output, Fields, Programs, and Procedures. This can be quantified in terms of volumes of data, trends, frequency of updating, etc. in order to estimate whether the new system will perform adequately or not. Technological feasibility is carried out basically to determine whether the company has the capability in terms of software, hardware, personnel and expertise to handle the completion of the project.

- **Economic Feasibility**

Economic analysis is the most frequently used method for evaluating the effectiveness of a new system. More commonly known as cost/benefit analysis, theprocedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. If benefits outweigh costs, then thedecision is made to design and implement the system. An entrepreneur must accurately weigh the cost versus benefits before taking an action.

- **Operational Feasibility**

It is a measure of how well a proposed system solves the problems, and takes advantages of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development. This project provides interactive interface to generate report as per the requirements and also user can update the information efficiently.

2.3 Objectives of Proposed System

- To be able to search for and compare products
- To be able to view the products and compare them based on real-time feedback from other customers
- To be able to pay using their desired payment mode: Online or COD
- To be able to view and give feedback on past orders
- To be able to add and manage, products and categories
- To be able to add, manage delivery partners and assign orders
- To be able to get order status details
- To be able to view feedback

2.4 Users of System

Customer: Any person who wants to buy products and make payments for them will be able to use this system in order to do so. Enabling customers to check the availability and various brand of products online. Customers can opt for cash on delivery or online payment as per their choice.

Admin: The person will be able to access and manage all information related to this system. Providing a web-based grocery store where customers can purchase products online without physically going to the store. The admin assigns orders to the delivery partner as soon as they are placed.

Delivery Partner: The person will be able to deliver the products and collect the payment if the user has chosen cash on delivery as the payment method. As part of the delivery process, the delivery partner will be able to update the order status according to the delivery status. As the delivery partner tries to eliminate user annual interactions with the physical store, they play an important role in online applications.

3: Analysis & Design

3.1 System Requirements (Functional and Non - Functional)

- **Functional Requirements**

Requirement Analysis is the first phase of software development process. This phase focuses to understand the problem. Requirement Analysis is on identifying what is need from these systems, not how the system will achieve its goals. In this phase often at least two parties are involved in Software Development-a client and a developer. The developer has to develop the system to satisfy the clients' needs. The developer and client arrange a meeting and discuss his/her own views. The developer asks the clients for his/her needs. After a meeting the developer understands what the requirements of the client are. Before starting of the development process, the developer analyses, test the requirements which are given by the clients. According to those requirements the developer starts development process. Hence the developer needs a user's problem.

In the software requirement we are dealing with the requirements of the proposed system, that's the capabilities of that system, which is yet to be developed, should have. The software requirement specification (SRS) is a document that completely describes what the proposed software should do without describing how the software will do it. So the basic goal of Requirement Phase is to produce the SRS, which describes the complete external behavior of the proposed software.

- **Non-Functional Requirements**

The following performance characteristics were taken care of in developing the systems:

- ✓ **User Friendliness:** There is a lot of simplicity and ease of use to the system. It is also possible for a native user to use the system effectively and without any difficulties.

- ✓ **User Satisfaction:** User expectations are met by the system.

- ✓ **Error Handling:** Response to user errors and undesired situations has been taken care of to ensure that the system operations without halting in case ofsuch situation and proper are given to user.

- ✓ **Safety:** A catastrophic behavior can be avoided by the program.

- ✓ **Robustness:** The system recovers from undesired events without humanintervention.

- ✓ **Security:** "Khaas Food – Online Grocery" This system protects information by utilizing a password mechanism. As a result, only authorized individuals have access to the databases

- ✓ **Validations:** This results in a thorough testing of the details, and the system is such that likely to changes and modifications can be easily incorporated in it. online bus ticket reservation full presentation review.

- ✓ **Portability:** The system can move to a new hardware/ operating system aftermaking minor modifications to it.

- ✓ **Exception Handling:** To ensured that the system does not halt in case of undesired situation or events exception conditions are taken care of providing the corresponding exception responses while developing the system

- ✓ **Code Optimization:** Due to its inheritance concept, it is possible to reduce coding since the majority of the codes are reused to reduce repeated coding and also the result set is reused where necessary to reduce repetitive coding. A function organization that is built into the various processes can be traced back to the inbuilt functions. There is a holistic approach to system analysis, which is based on the identification of the inputs and outputs of the system. Analyzing a system involves becoming aware of the problem at hand, as well as identifying the relevant and decisional variables which will assist in solving it.

- **Modules**

  - ❖ Category Management: Used for managing the product categories.
  - ❖ Product Management: Used for managing the product.
  - ❖ Product Search: Used for finding and comparing products.
  - ❖ Order Management: Used to managing the orders by assigning it.
  - ❖ Feedback Management: Used for managing the booking details.
  - ❖ Customer Management: Used for managing the customer information.
  - ❖ Login Management: Used for managing the login details.
  - ❖ Delivery Partner Management: used to managing the user of the system.

- **System Analysis**

System analysis is a process of gathering and interpreting facts, diagnosing problems and the information about the bus booking tickets to recommend improvement on the system. It is problem solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is studied to the minutest detail and analyzed. The system analyst plays the role of the interrogator and dwells deep into the working of the present system. The system is viewed as a whole and the input to the system are identified.

- **UML Diagram**

It is essential that any programmer must thoroughly know the language he or she uses when designing and analyzing. The programmer would have to analyze the program and then knows the problem he needs to solve. He would then perform the process of coding while applying the process of design which he presented previously. Finally, he or she would have to test the program in order to ascertain compatibility with customer requirement.

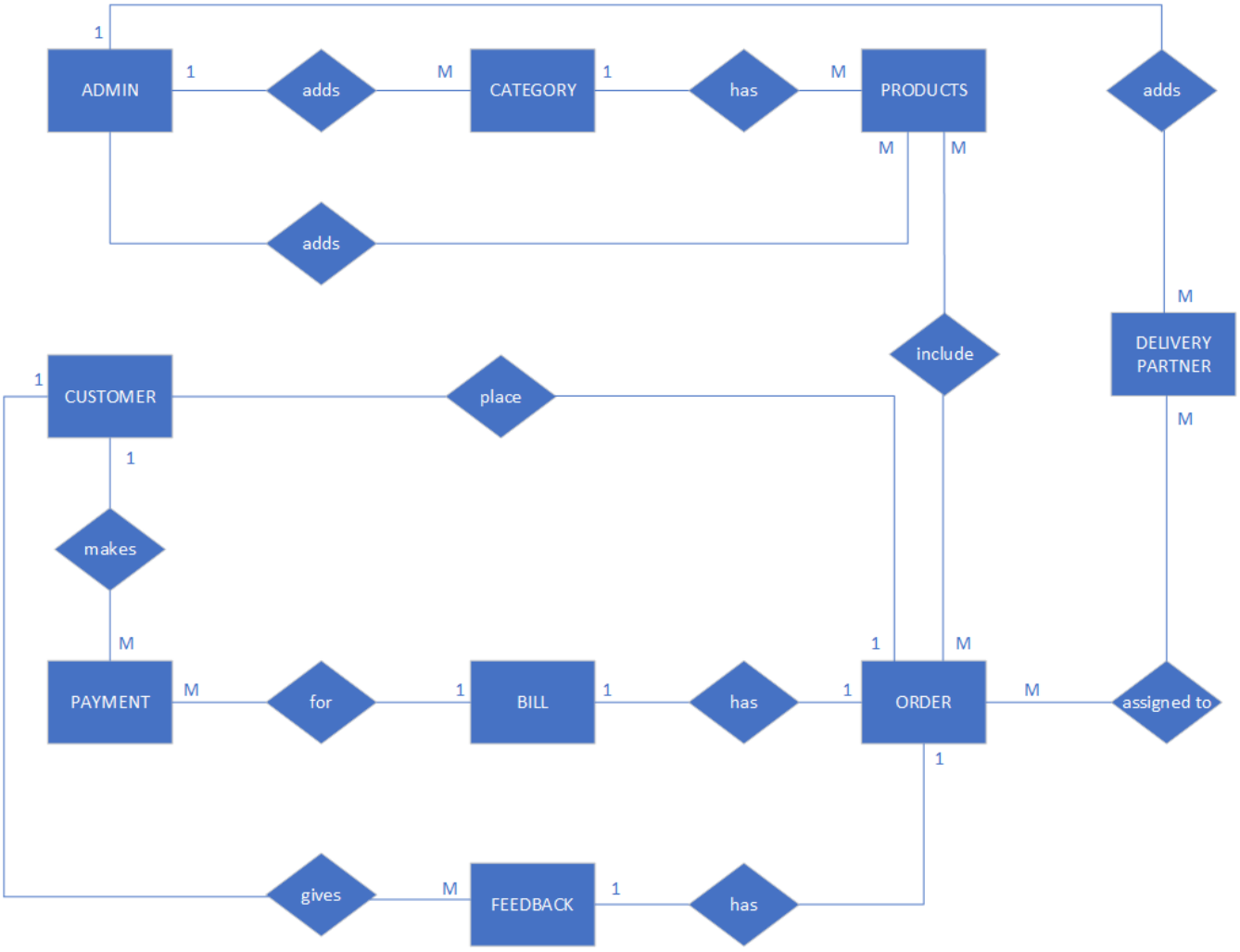
The process we have mentioned, including Analysis, Design, Coding and Testing, identify the project's Software Development Life Cycle (SDLC) as any project would have to go through all these processes using the appropriate methodology.Otherwise, chaos would ensue.

We shall use the Interactive and incremental development methodology in order to develop a prototype system. This process is characterized with flexibility and revision whenever necessary in all phases. The process would begin with an initial plan and concluded with interaction among the various phases and components.

As to designing the processes used in describing and interactions in this project, weshall be using the UML (Unified Modeling Language).

3.2 Entity Relationship Diagram (ERD)

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database.



3.3 Table Structure

Table Name: **Admin**
Description: Contains information about Admin.

Sr No	***Attribute***	***Data-Type***	***Size***	***Constraints***
1	ADMIN_ID	Bigint	10	Primary Key
2	FIRST_NAME	varchar	20	Not Null
3	LAST_NAME	varchar	20	Not Null
4	EMAIL	varchar	50	Not Null
5	PASSWORD	varchar	20	Not Null
6	ADDRESS	varchar	200	Not Null
7	MOBILE	bigint	10	Not Null
8	IS_ACTIVE	boolean	1	Not Null
9	CREATED_BY	Bigint	10	Not Null
10	CREATED_ON	Date		Not Null

Table Name: **Delivery Partner**
Description: Contains information about Delivery Partner.

Sr No	***Attribute***	***Data-Type***	***Size***	***Constraints***
1	DELIVERY_PARTNER_ID	bigint	10	Primary Key
2	FIRST_NAME	varchar	20	Not Null
3	LAST_NAME	varchar	20	Not Null
4	EMAIL	varchar	50	Not Null
5	PASSWORD	varchar	20	Not Null
6	PANCARD	varchar	10	Not Null
7	MOBILE	bigint	10	Unique
8	PINCODE	int	6	Not Null
9	IS_ACTIVE	boolean	1	Not Null
10	CREATED _ON	date		Not Null
11	ADMIN_ID	bigint	10	Foreign Key

Table Name: **Customer**
Description: Contains information about Customers

Sr No	***Attribute***	***Data-Type***	***Size***	***Constraints***
1	CUSTOMER_ID	bigint	10	Primary Key
2	FIRST_NAME	varchar	20	Not Null
3	LAST_NAME	varchar	20	Not Null
4	EMAIL	varchar	50	Not Null
5	PASSWORD	varchar	20	Not Null
6	MOBILE	bigint	10	Unique
7	STATE	varchar	20	Not Null
8	CITY	varchar	20	Not Null
9	BILLING_ADDRESS	varchar	200	Nullable
10	SHIPPING_ADDRESS	varchar	200	Nullable
11	IS_ACTIVE	boolean	1	Not Null
12	CREATED_ON	date		Not Null

Table Name: **Category**
Description: Contains information about Product Categories.

Sr No	***Attribute***	***Data-Type***	***Size***	***Constraints***
1	CATEGORY_ID	bigint	10	Primary Key
2	CATEGORY_NAME	varchar	20	Not Null
3	IS_AVAILABLE	boolean	1	Not Null
4	CREATED_ON	date		Not Null
5	ADMIN_ID	bigint	10	Foreign Key

Table Name: **Product**
Description: Contains information about Products.

Sr No	***Attribute***	***Data-Type***	***Size***	***Constraints***
1	PRODUCT_ID	bigint	10	Primary Key
2	PRODUCT_NAME	varchar	20	Not Null

3	PRODUCT_PRICE	numeric	6,2	Not Null
4	PRODUCT_QUANTITY	numeric	6	Not Null
5	PRODUCT_DESCRIPTION	varchar	500	Not Null
6	PRODUCT_DISCOUNT	numeric	2	Nullable
7	GST_HS_CODE	varchar	10	Nullable
8	IS_AVAILABLE	boolean	1	Not Null
9	CREATED_ON	date		Not Null
10	ADMIN_ID	bigint	10	Foreign Key
11	CATEGORY_ID	bigint	10	Foreign Key

Table Name: **Order**
Description: Contains information about Order.

Sr No	***Attribute***	***Data-Type***	***Size***	***Constraints***
1	ORDER_ID	bigint	10	Primary Key
2	TOTAL_AMOUNT	varchar	20	Not Null
3	ORDER_STATUS	boolean	1	Not Null
4	CREATED_ON	date		Not Null
5	CUSTOMER_ID	bigint	10	Foreign Key

Table Name: **Product Order**
Description: Contains information about Product and Order.

Sr No	***Attribute***	***Data-Type***	***Size***	***Constraints***
1	PRODUCT_ORDER_ID	bigint	10	Primary Key
2	QUANTITY	numeric	4	Not Null
3	PRODUCT_ID	bigint	10	Foreign Key
4	ORDER_ID	bigint	10	Foreign Key

Table Name: **Bill**
Description: Contains information about payment bill.

Sr No	***Attribute***	***Data-Type***	***Size***	***Constraints***
1	BILL_ID	Bigint	10	Primary Key
2	BILL_DATE	Date		Not Null
3	BILL_STATUS	Boolean	10	Not Null
4	PACKING_CHARGE	numeric	5,2	Nullable
5	DELIVERY_CHARGE	numeric	5,2	Nullable
6	GST	numeric	5,2	Not Null
7	OFFER_DISCOUNT	numeric	5,2	Nullable
8	FINAL_AMOUNT	numeric	10,2	Not Null
9	ORDER_ID	bigint	10	Foreign Key

Table Name: **Payment**
Description: Contains information about payment.

Sr No	***Attribute***	***Data-Type***	***Size***	***Constraints***
1	PAYMENT_ID	bigint	10	Primary Key
2	PAYMENT_TYPE	varchar	20	Not Null
3	TRANSACTION_ID	bigint	10	Not Null
4	PAYMENT_STATUS	boolean	1	Not Null
5	PAYMENT_DATE_TIME	date		Not Null
6	BILL_ID	bigint	10	Foreign Key

Table Name: **Order Delivery**
Description: Contains information about Order and Delivery Partner Mapping.

Sr No	***Attribute***	***Data-Type***	***Size***	***Constraints***
1	ORDER_DELIVERY_ID	bigint	10	Primary Key
2	ORDER_ DELIVERY_STATUS	varchar	50	Not Null
3	DELIVERY_DATE	date		Not Null
4	ADMIN_ID	bigint	10	Foreign Key
5	ORDER_ID	bigint	10	Foreign Key
6	DELIVERY_PARTNER_ID	bigint	10	Foreign Key

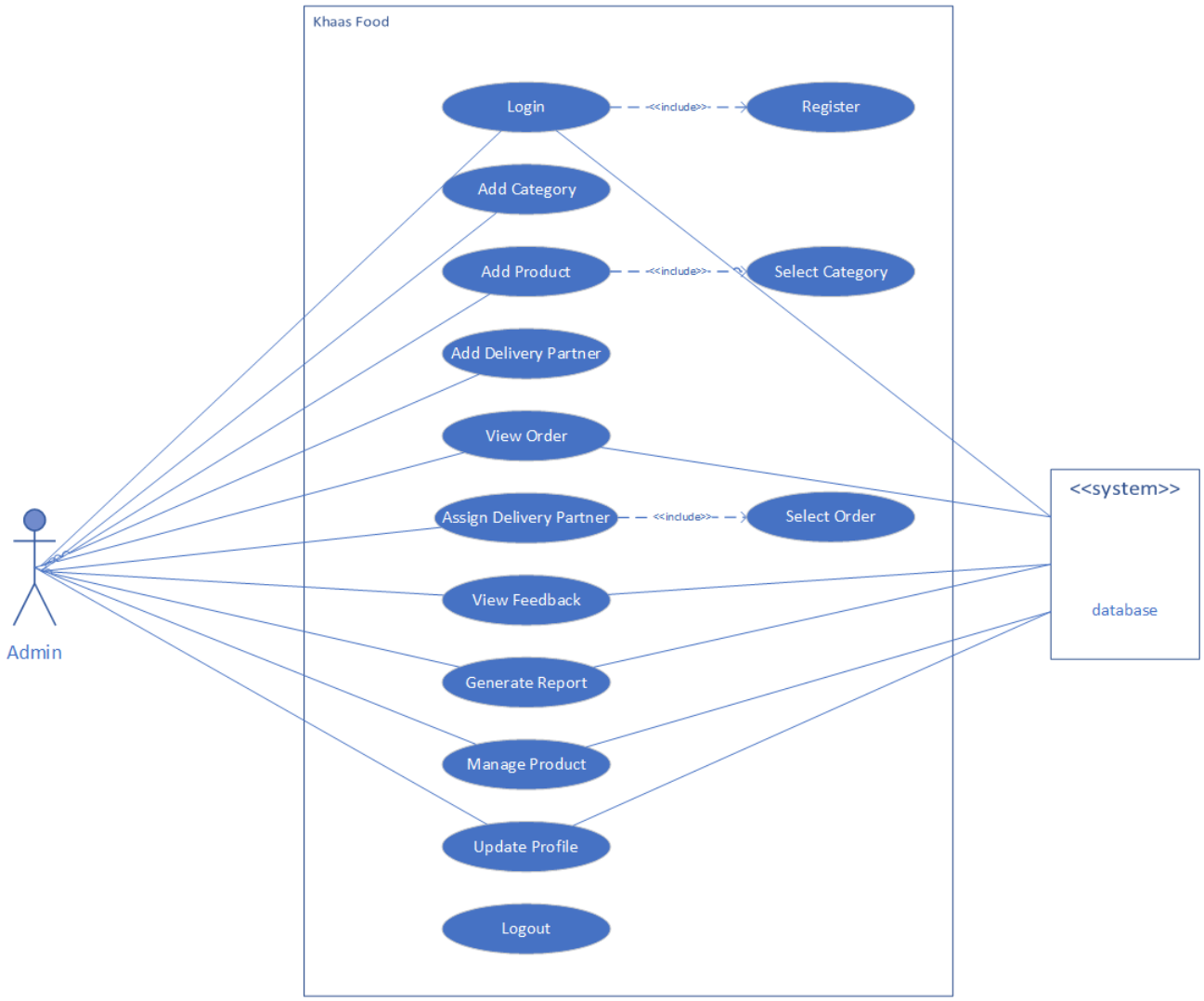
Table Name: **Feedback**
Description: Contains information about Customer Feedback.

Sr No	Attribute	Data-Type	Size	Constraints
1	FEEDBACK_ID	bigint	10	Primary Key
2	DELIVERY_RATING	bit	1	Nullable
3	ORDER_RATING	bit	1	Nullable
4	IS_REVIEWED	boolean	1	Not Null
5	CREATED_ON	date		Not Null
6	ORDER_ID	bigint	10	Foreign Key
7	ADMIN_ID	bigint	10	Foreign Key

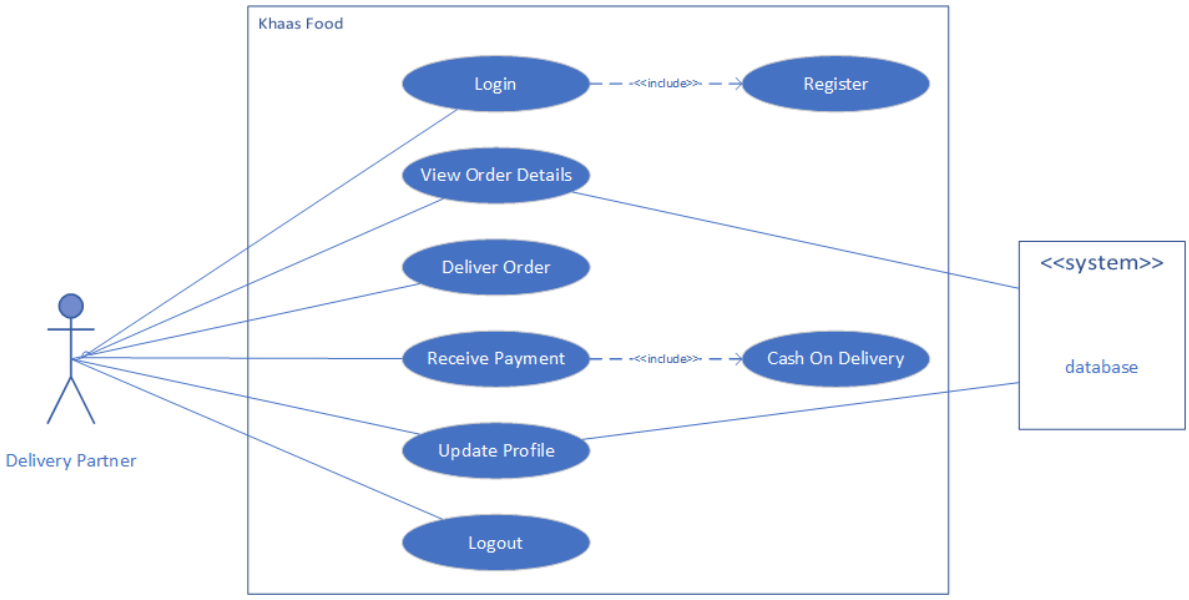
3.4 Use Case Diagrams

A use case diagram is a dynamic or behavior diagram in UML. Use case diagrams model the functionality of a system using actors and use cases. Use cases are a setof actions, services, and functions that the system needs to perform.

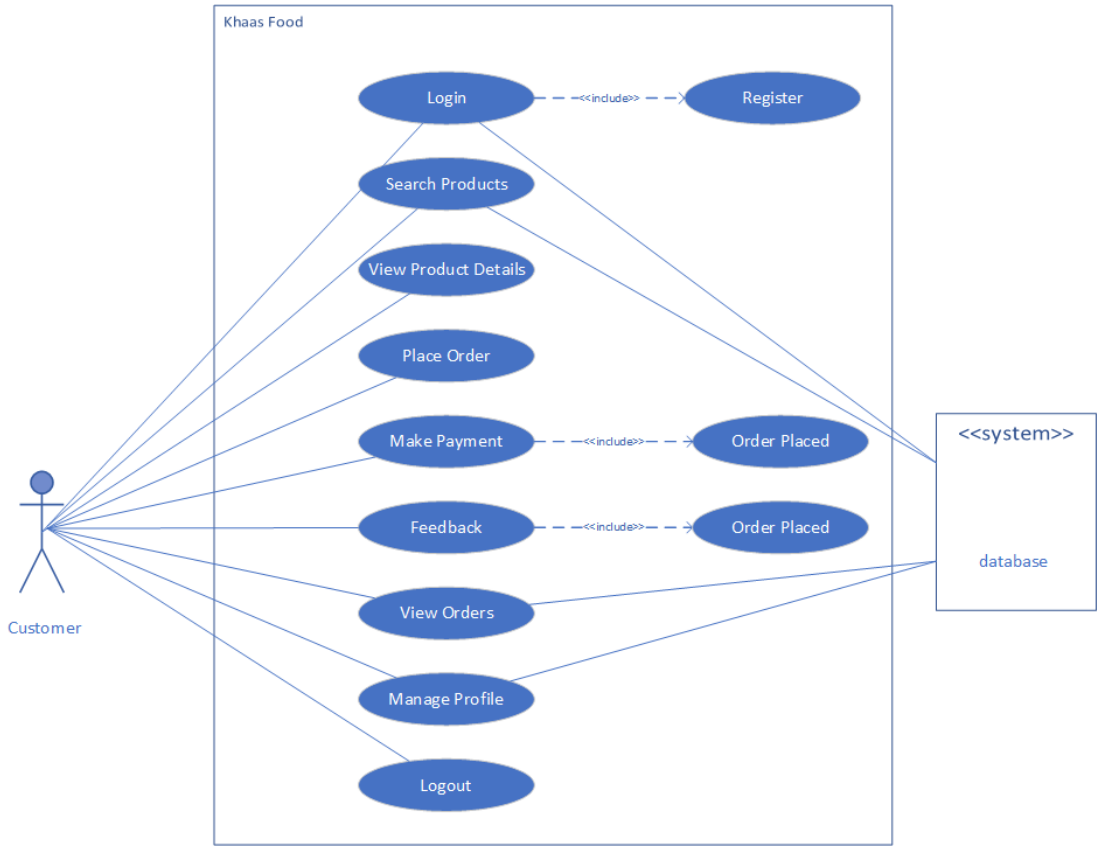
- **Use case diagram for Admin:**



- **Use case diagram for Delivery Partner**

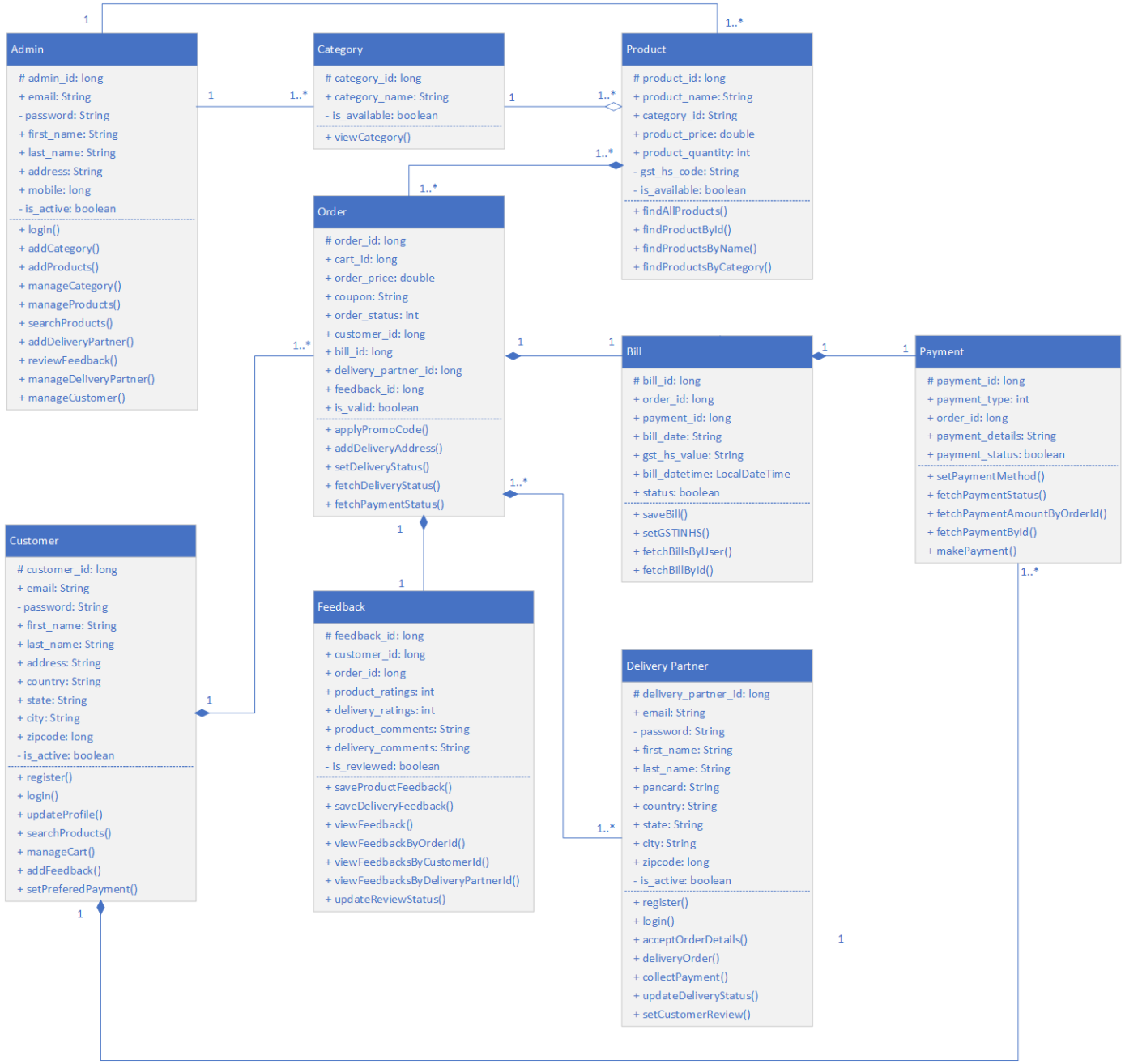


- **Use case diagram for Customer**



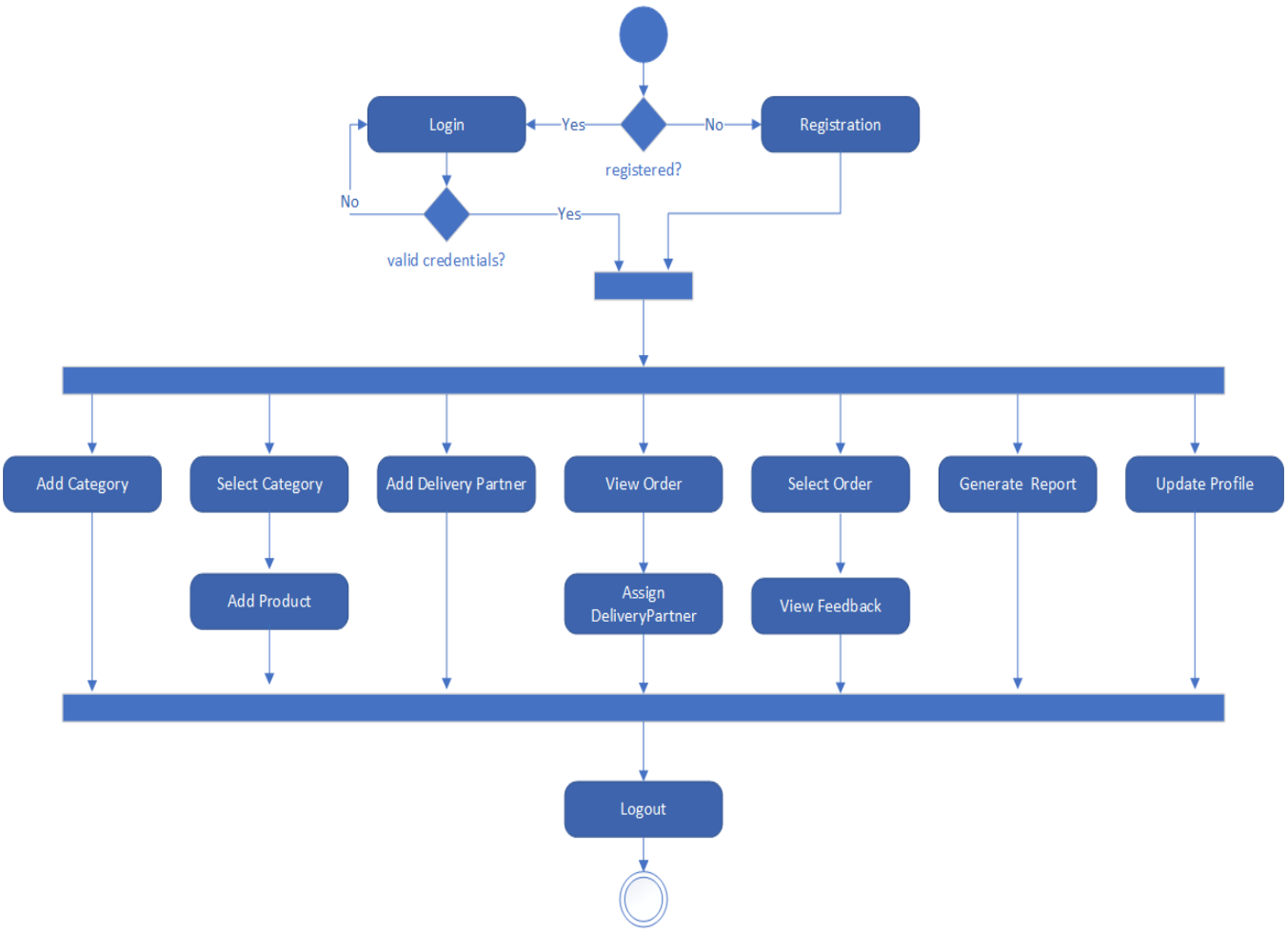
3.5 Class Diagram

Class diagram is a type of structure diagram that shows the structure of the classes, attributes, operations and relationship among them. Given below is the class diagram of the proposed system which shows classes such as User and Owner.

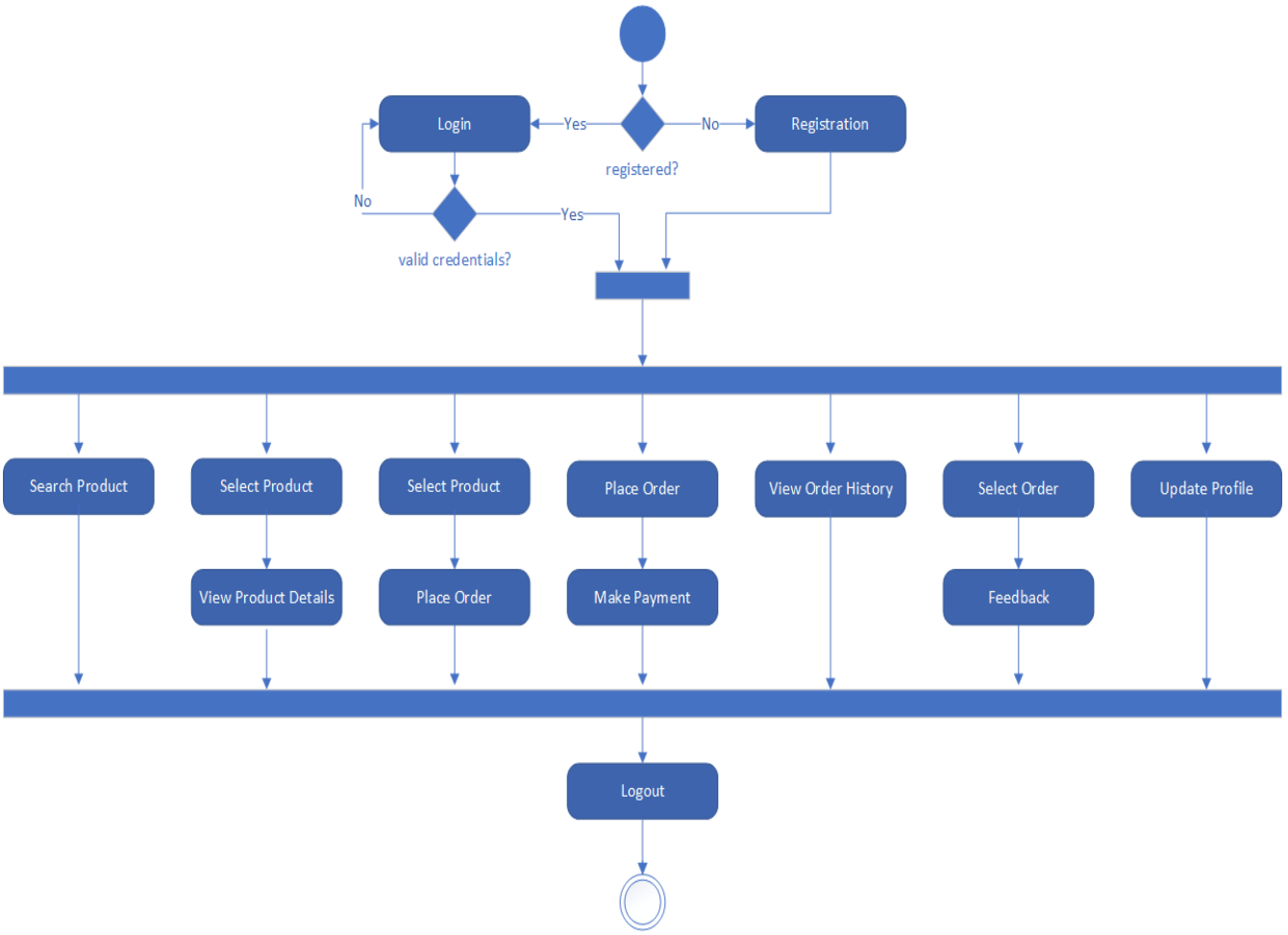


3.6 Activity Diagrams

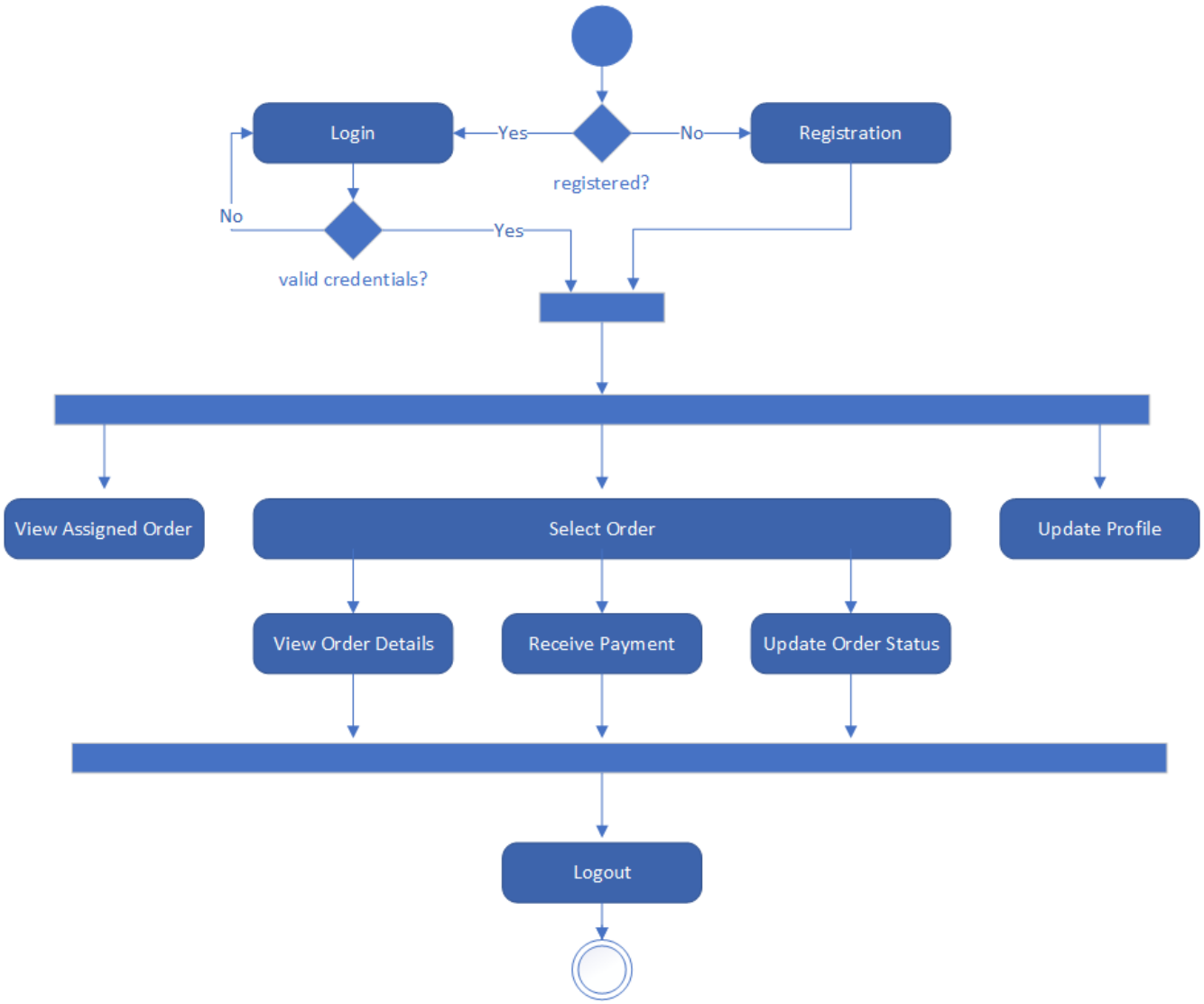
- **Activity diagram for Admin**



- **Activity diagram for Customer**

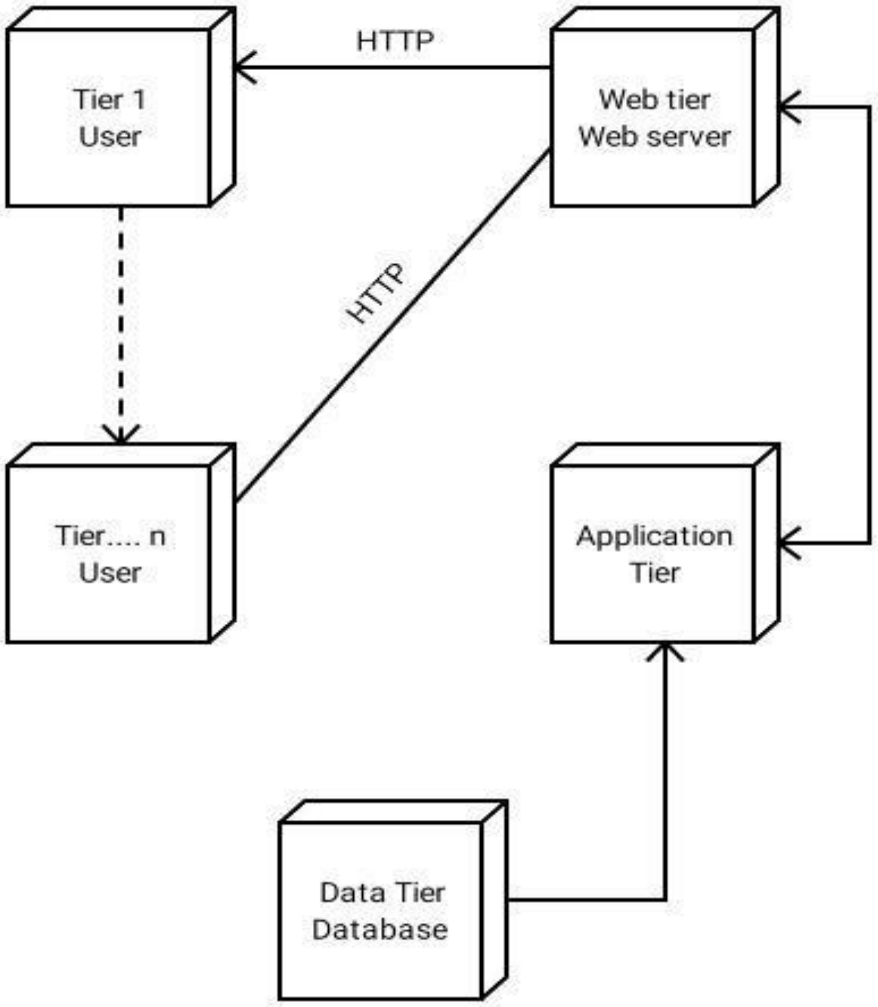


- **Activity diagram for Delivery Partner**

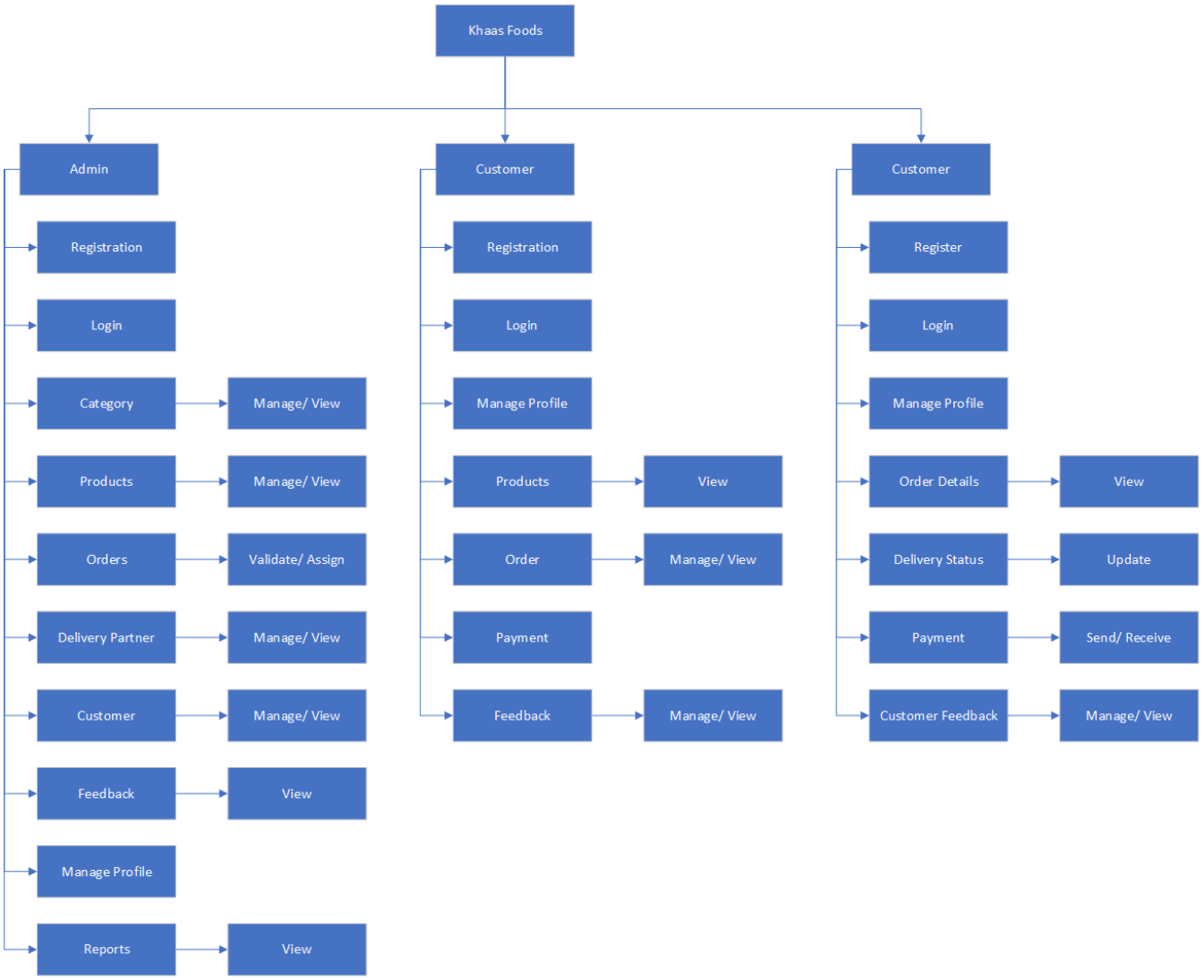


3.7 Deployment Diagram

A UML deployment diagram is a diagram that shows the configuration of run time processing nodes and the components that live on them. Deployment diagrams is a kind of structure diagram used in modeling the physical aspects of an object-oriented system. They are often be used to model the static deployment view of a system (topology of the hardware).

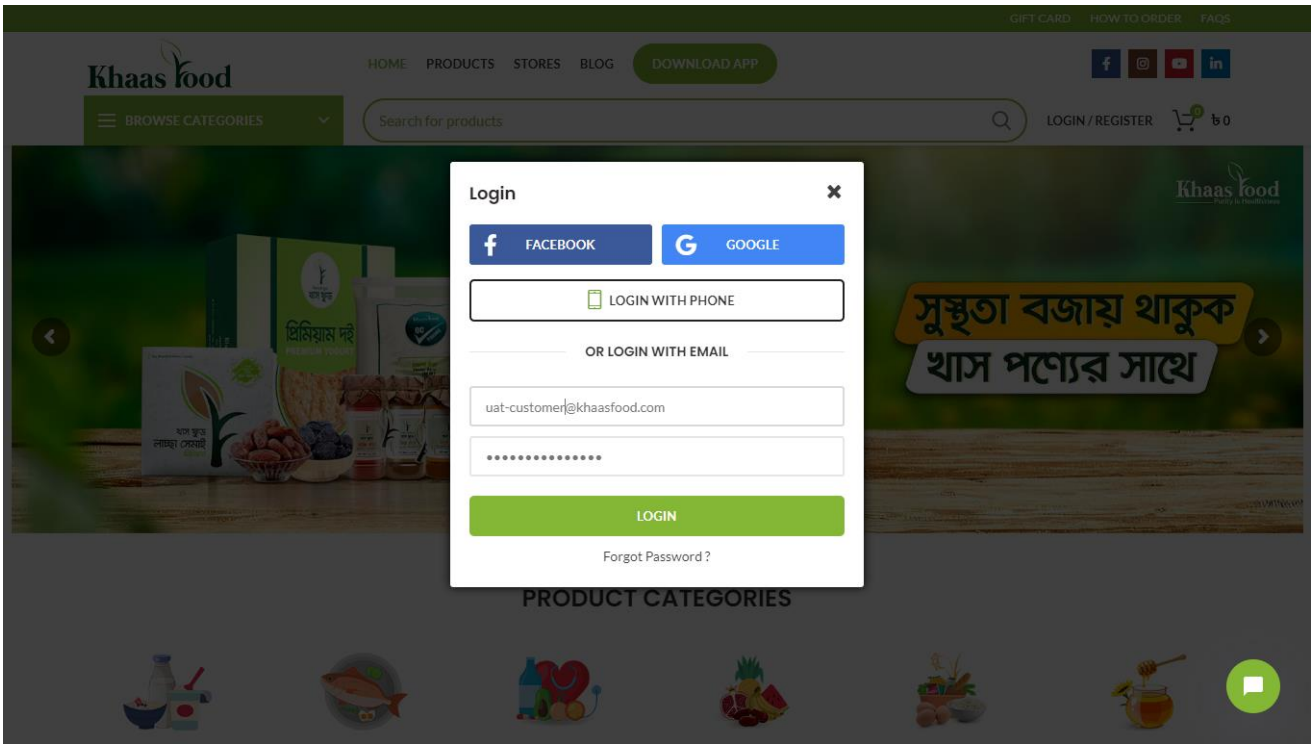


3.8 Module Hierarchy Diagram

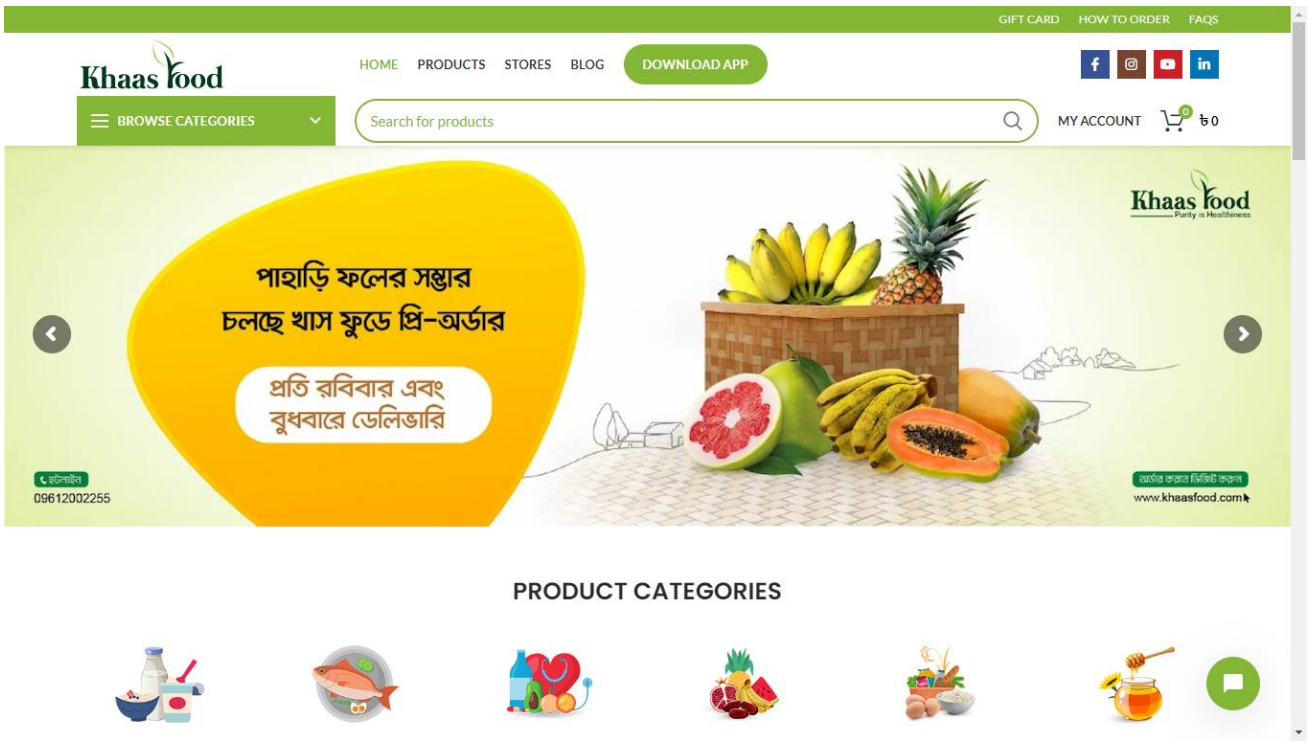


3.9 Sample Input and Output Screens

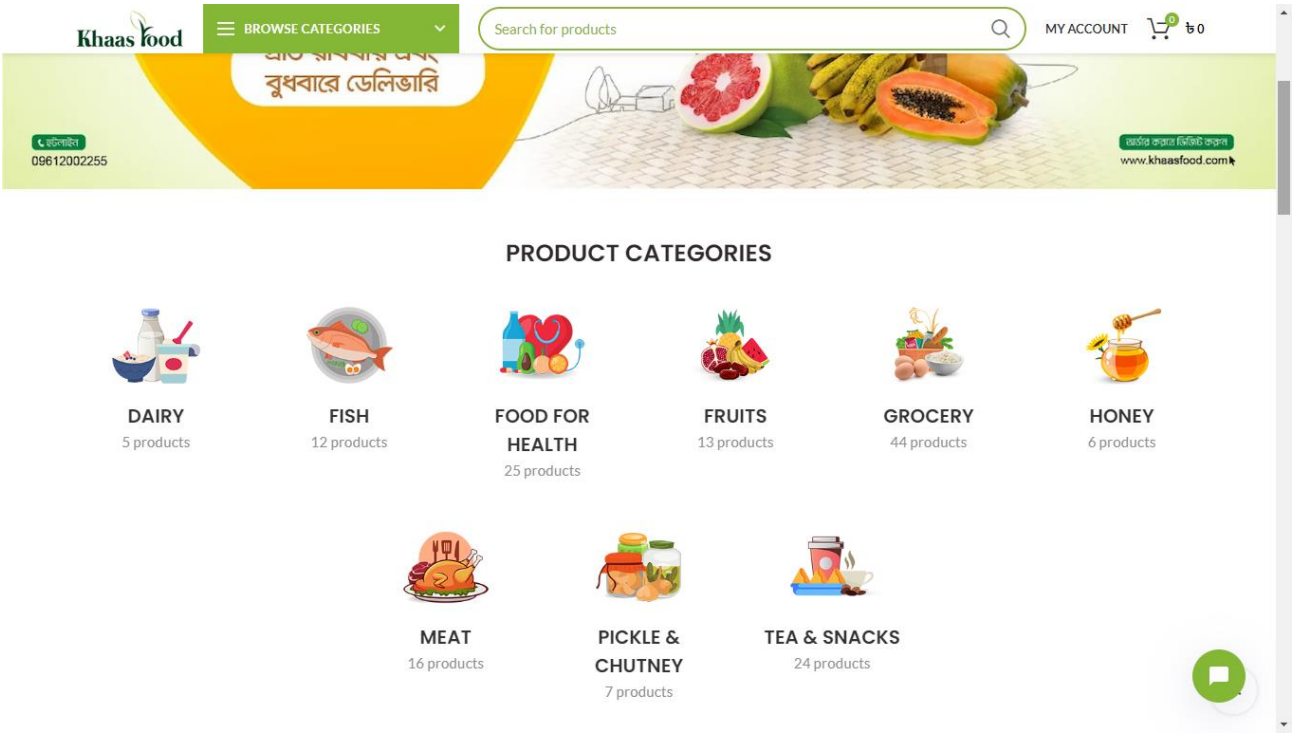
Customer - ***Register and Login Page***



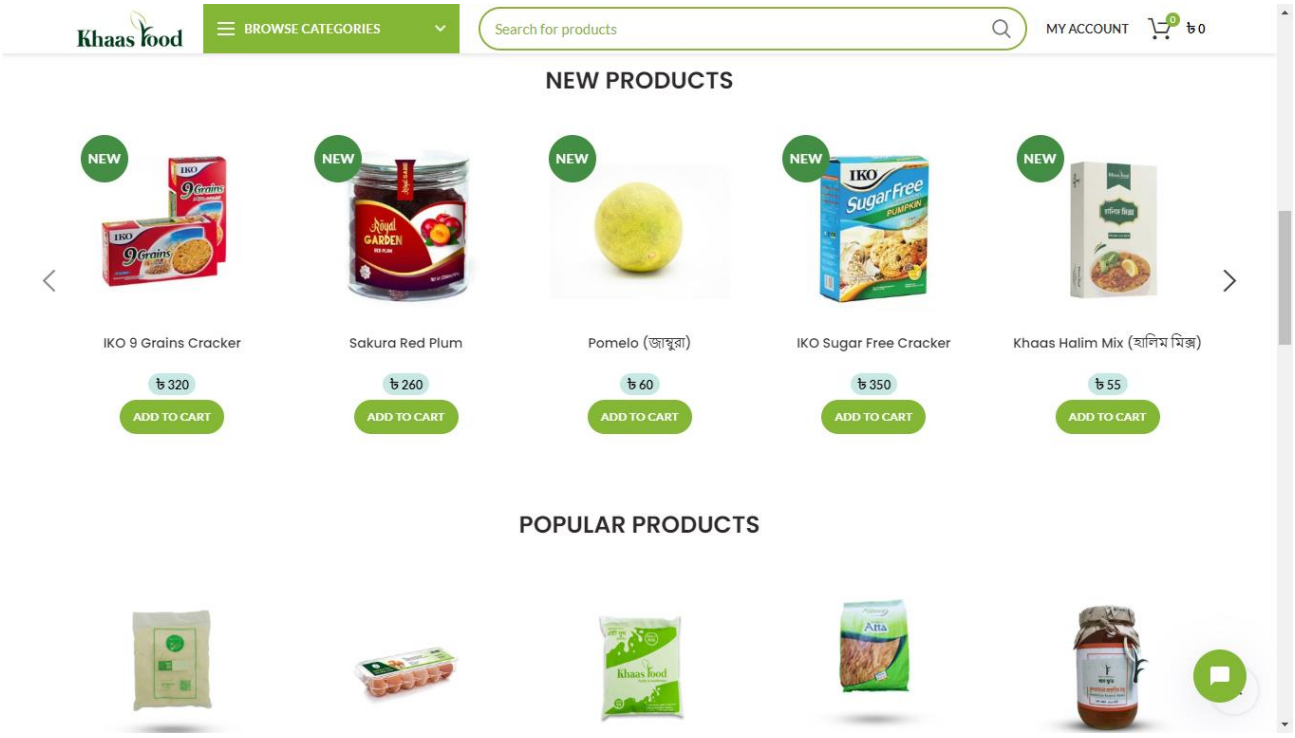
Customer – ***Dashboard***



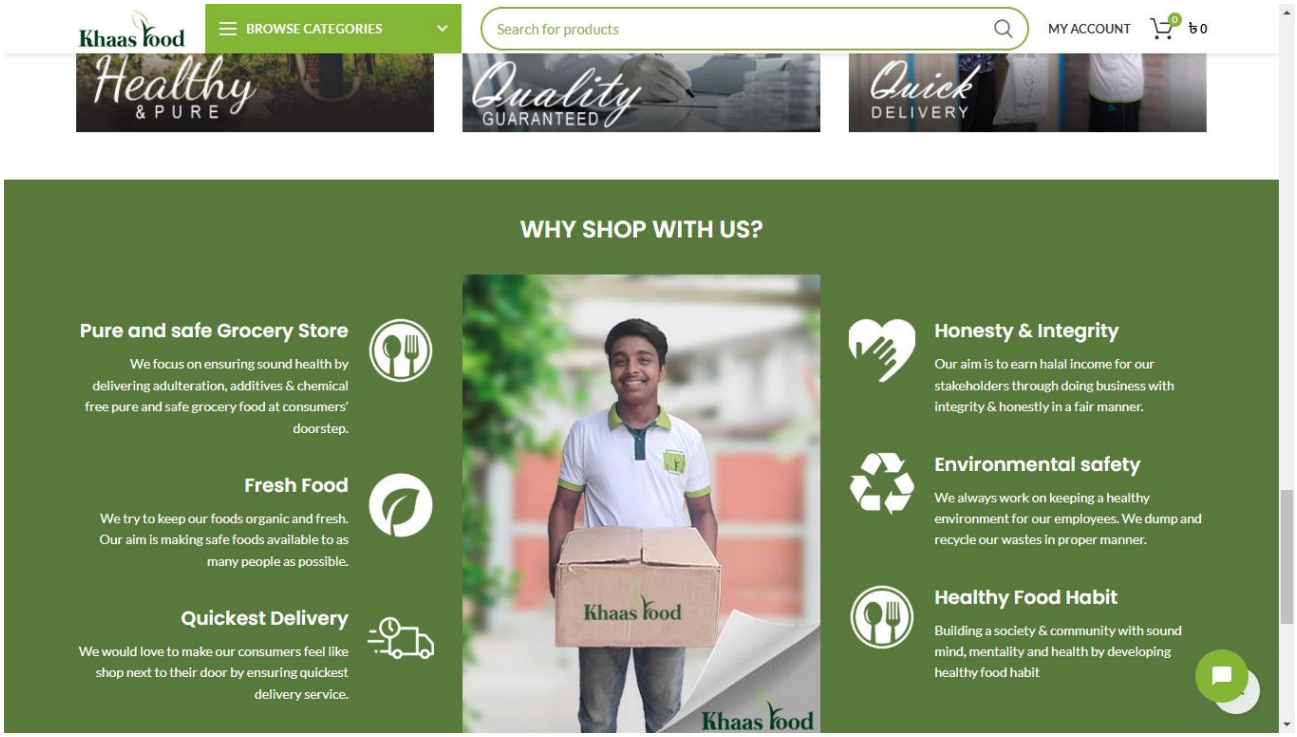
Customer – ***Product Categories***



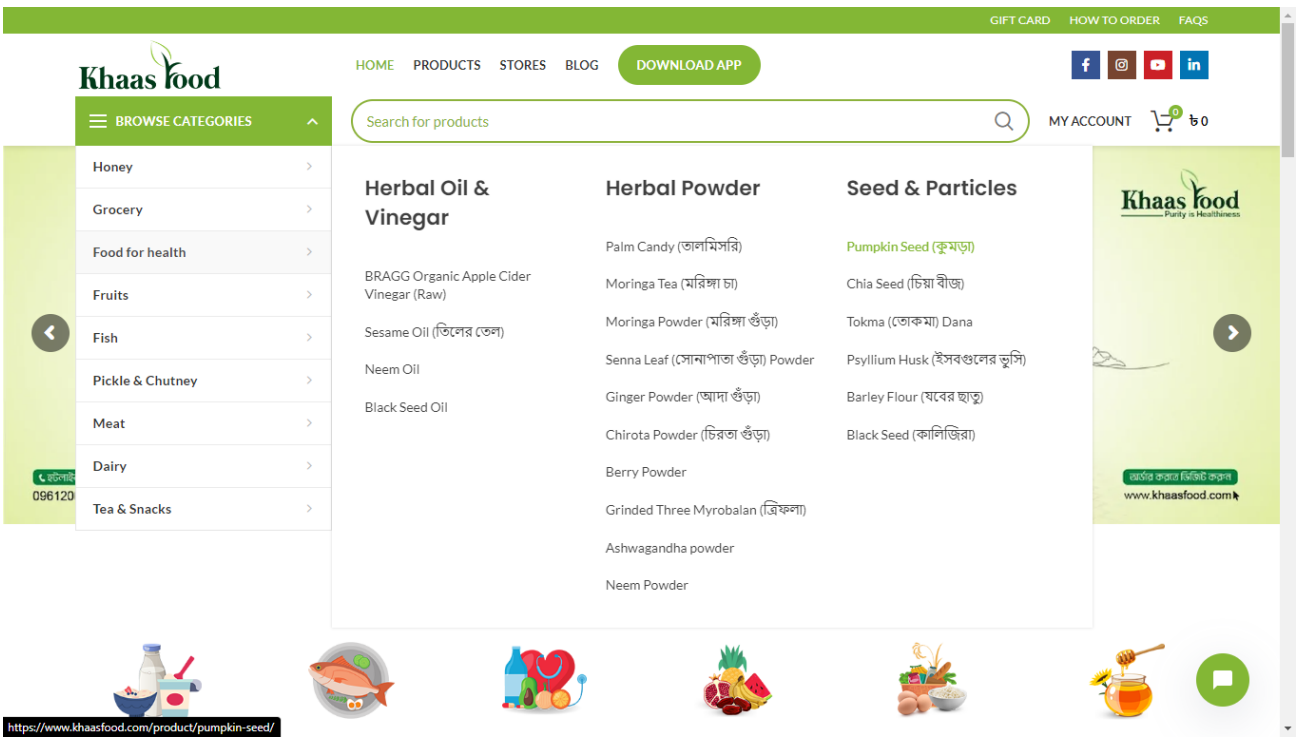
Customer – ***New & Popular Products***



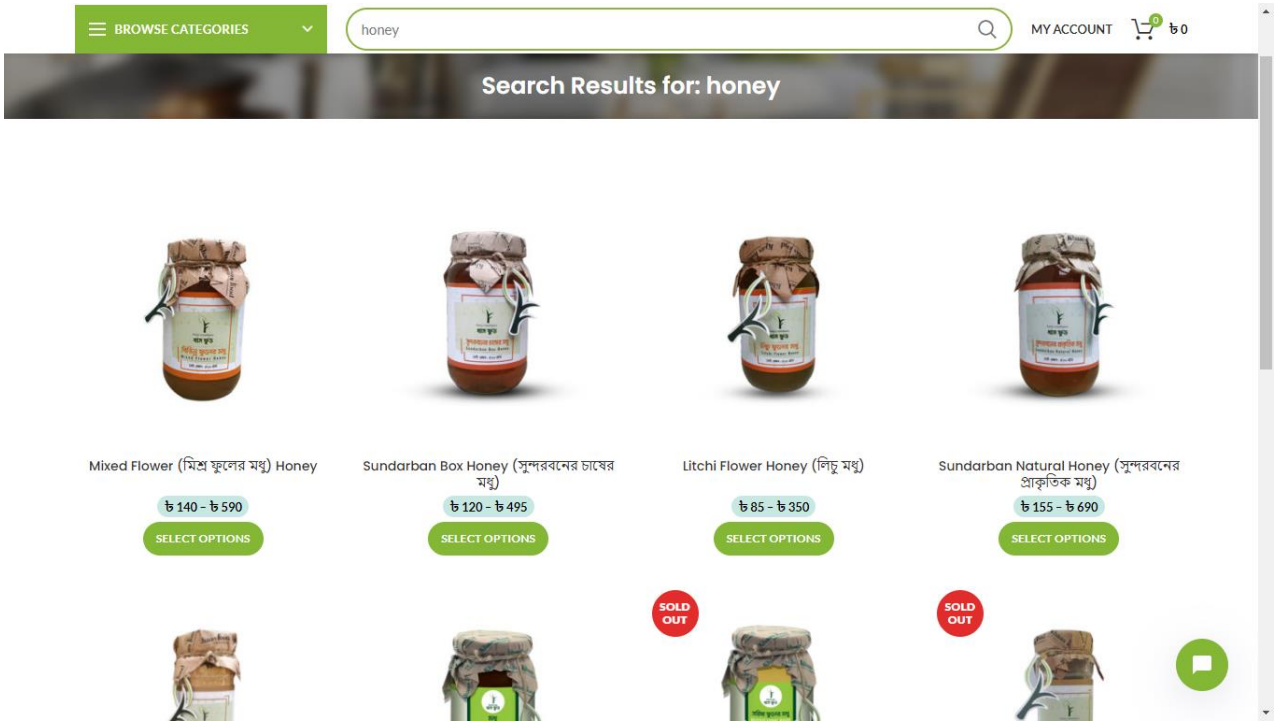
Customer – ***About Us***



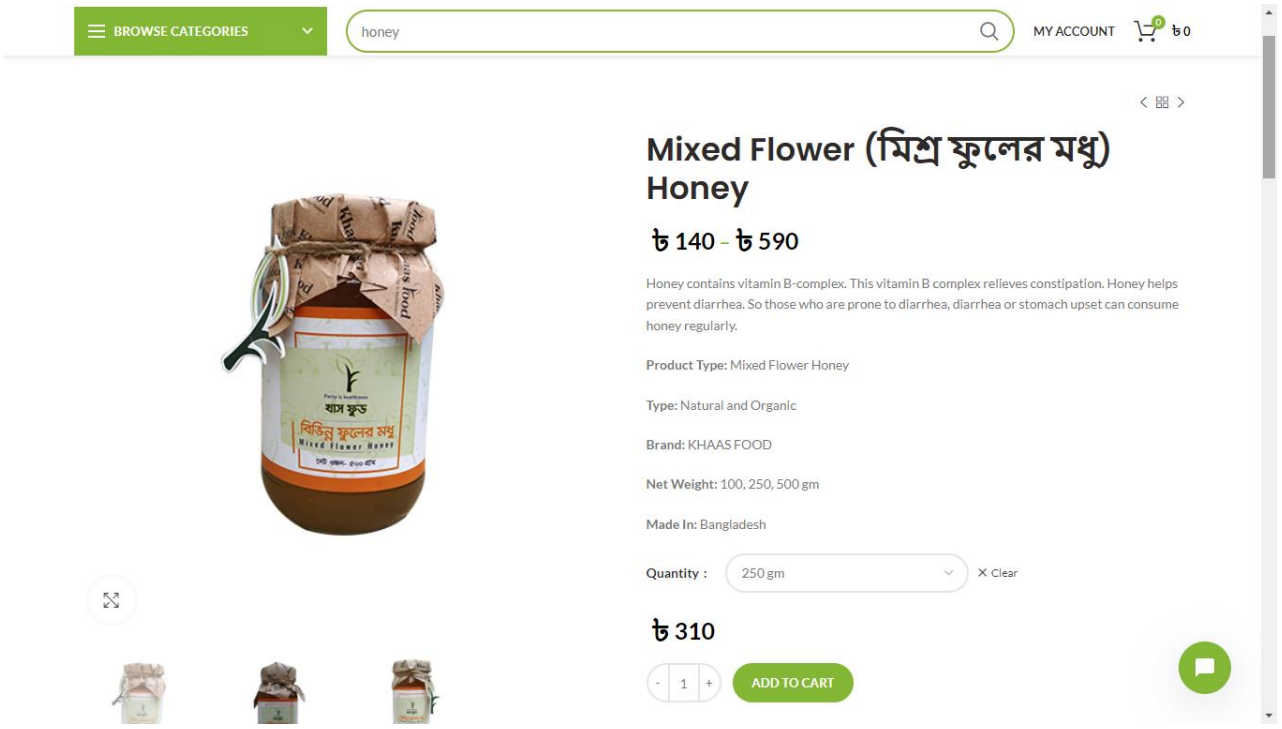
Customer – ***Browse By Category***



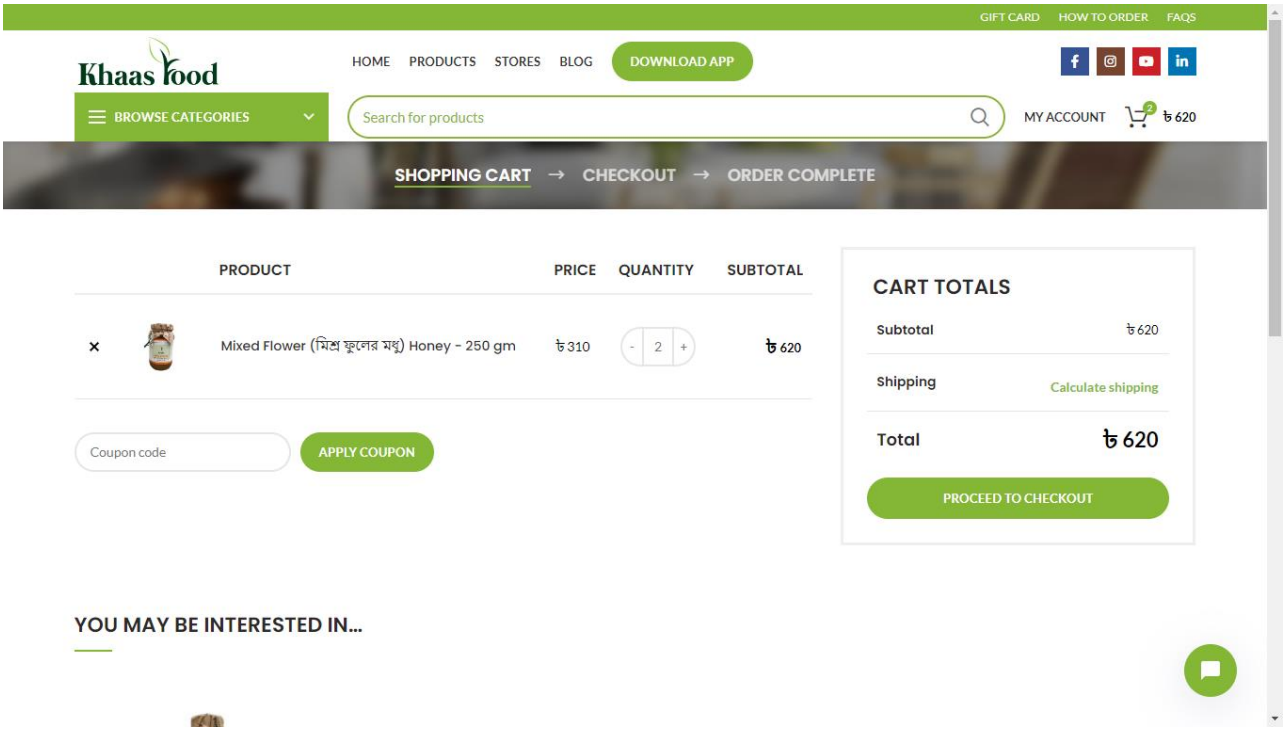
Customer – ***Search Product***



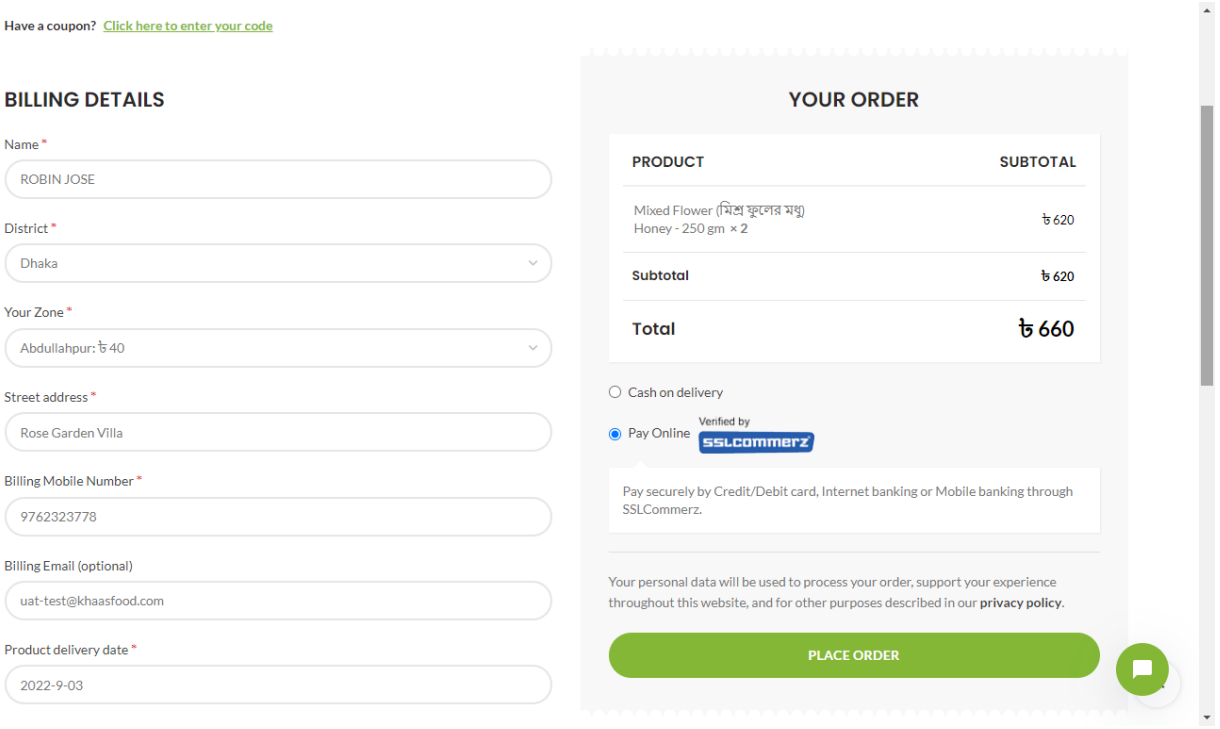
Customer – ***View Product Description***



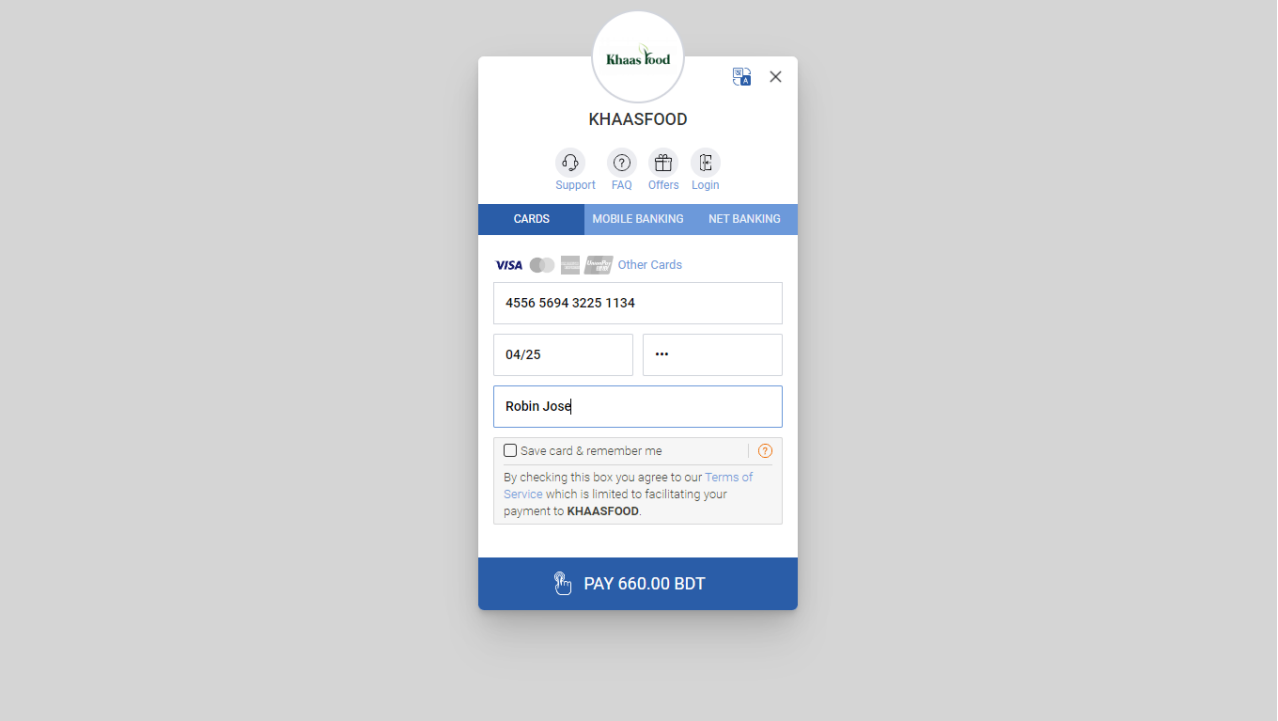
Customer – ***View Cart***



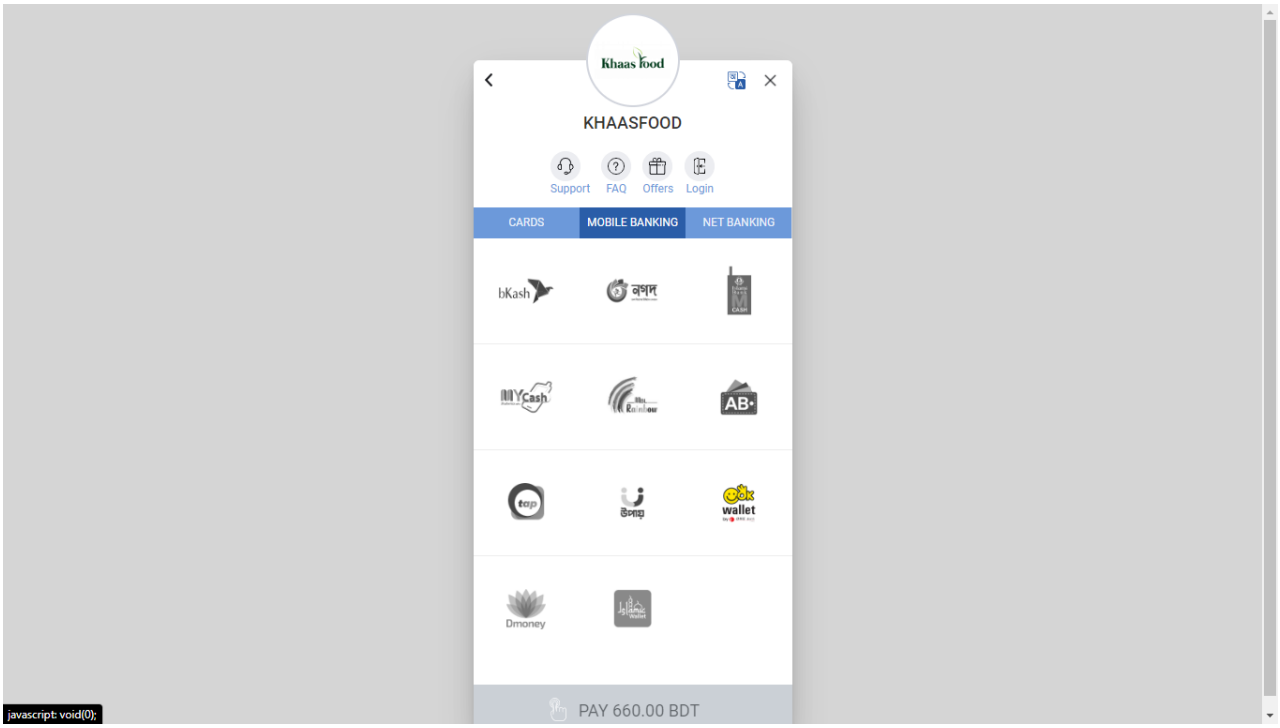
Customer – ***Checkout Page***



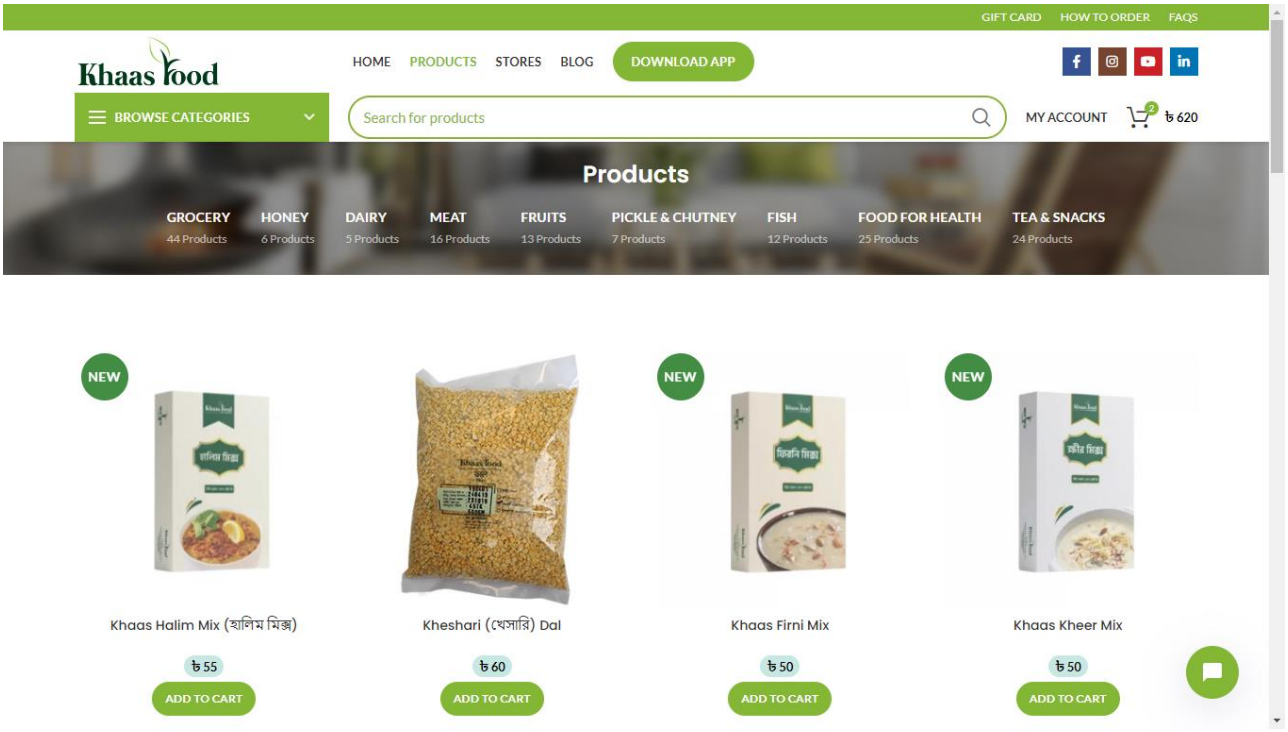
Customer – ***Payment Page***



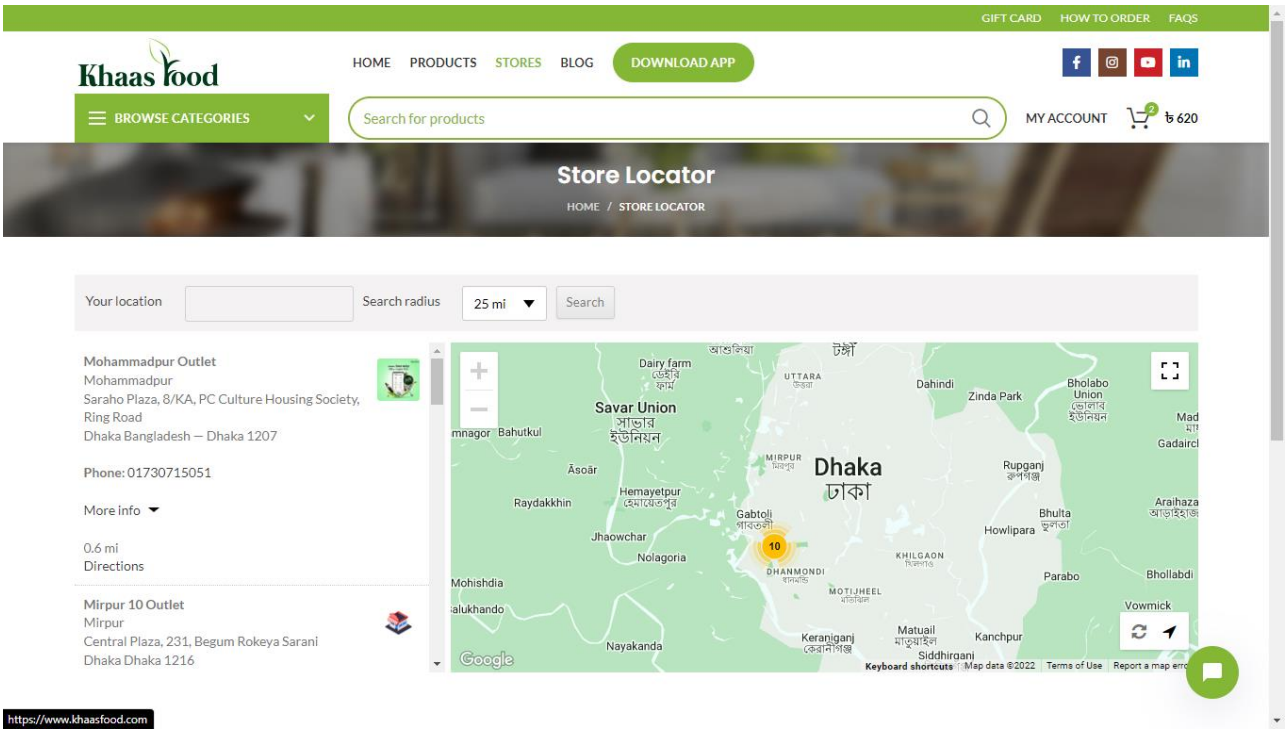
Customer – ***Payment Other Options***



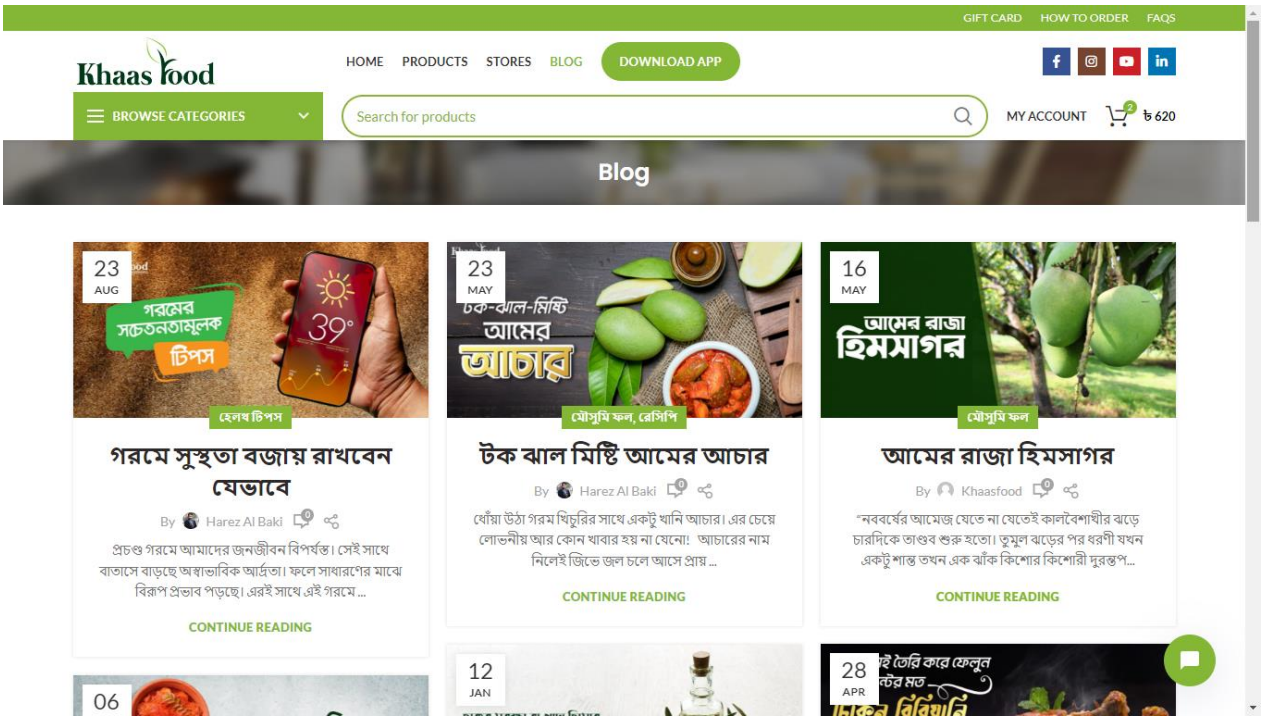
Customer – ***Products Tab***

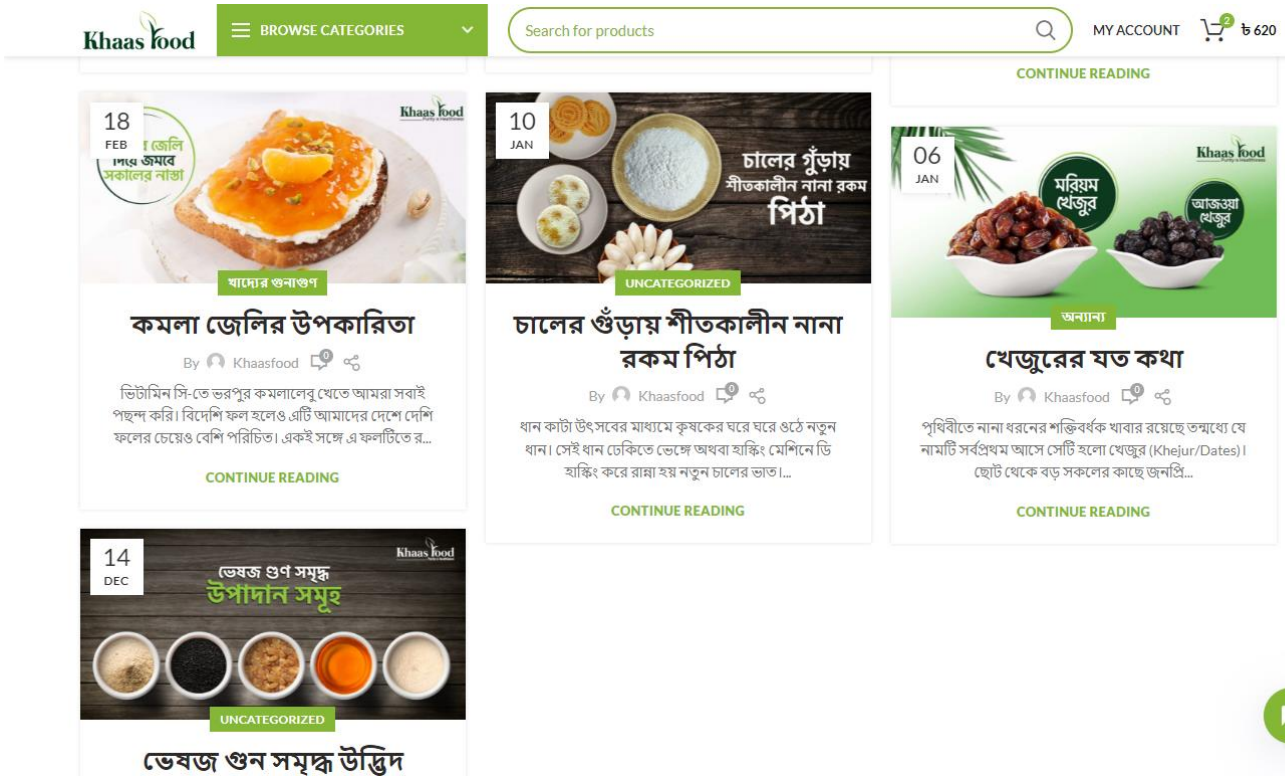


Customer – ***Store Locator***

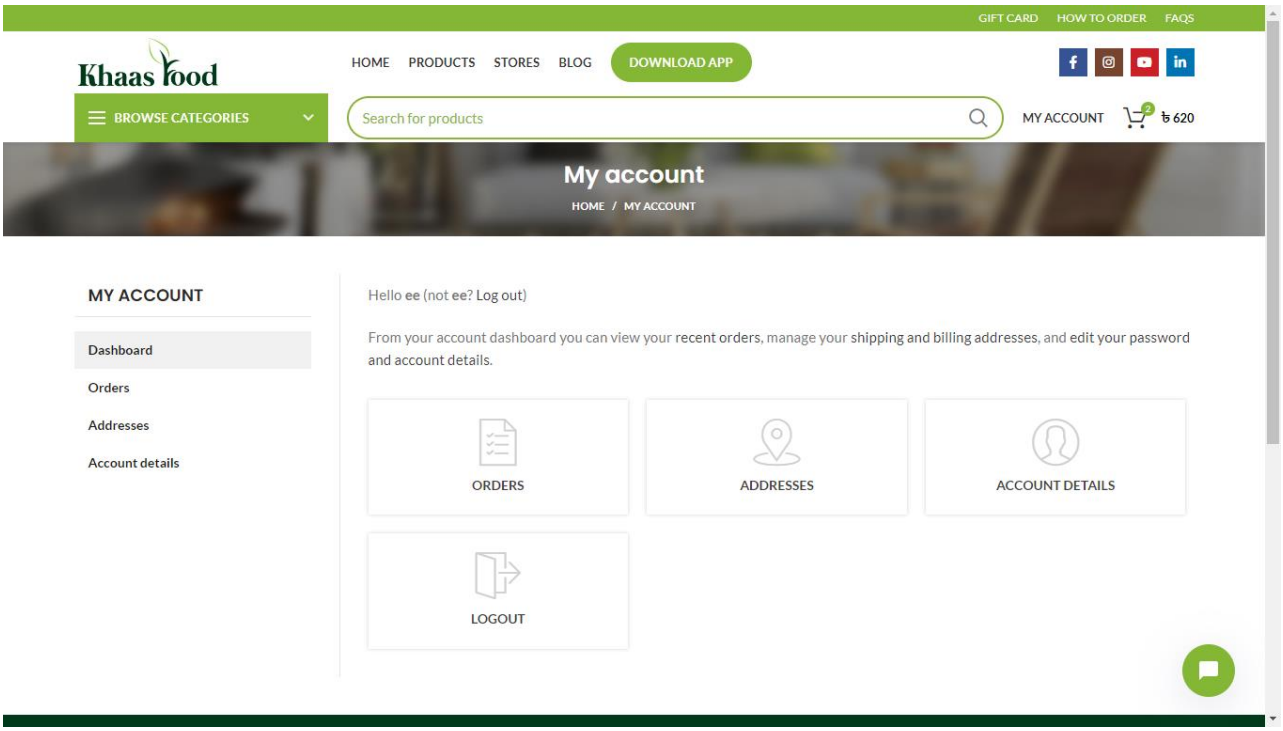


Customer – ***Blog Tab***

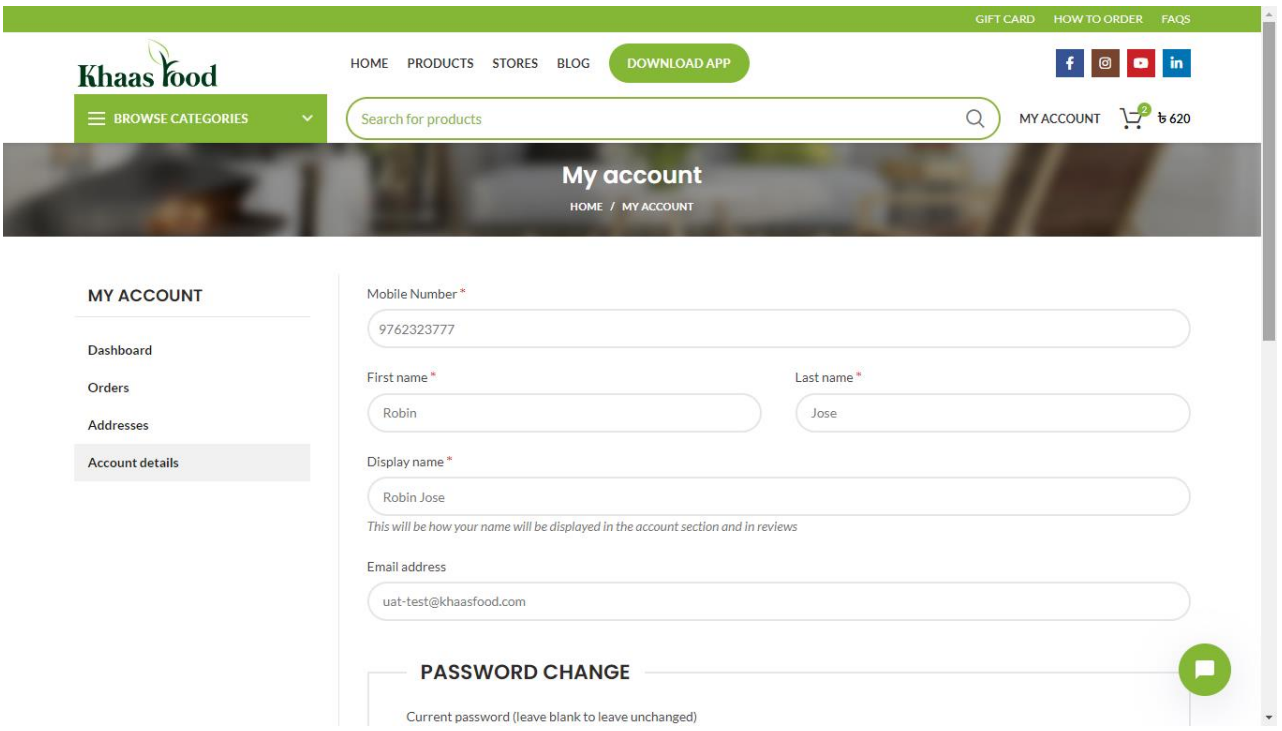




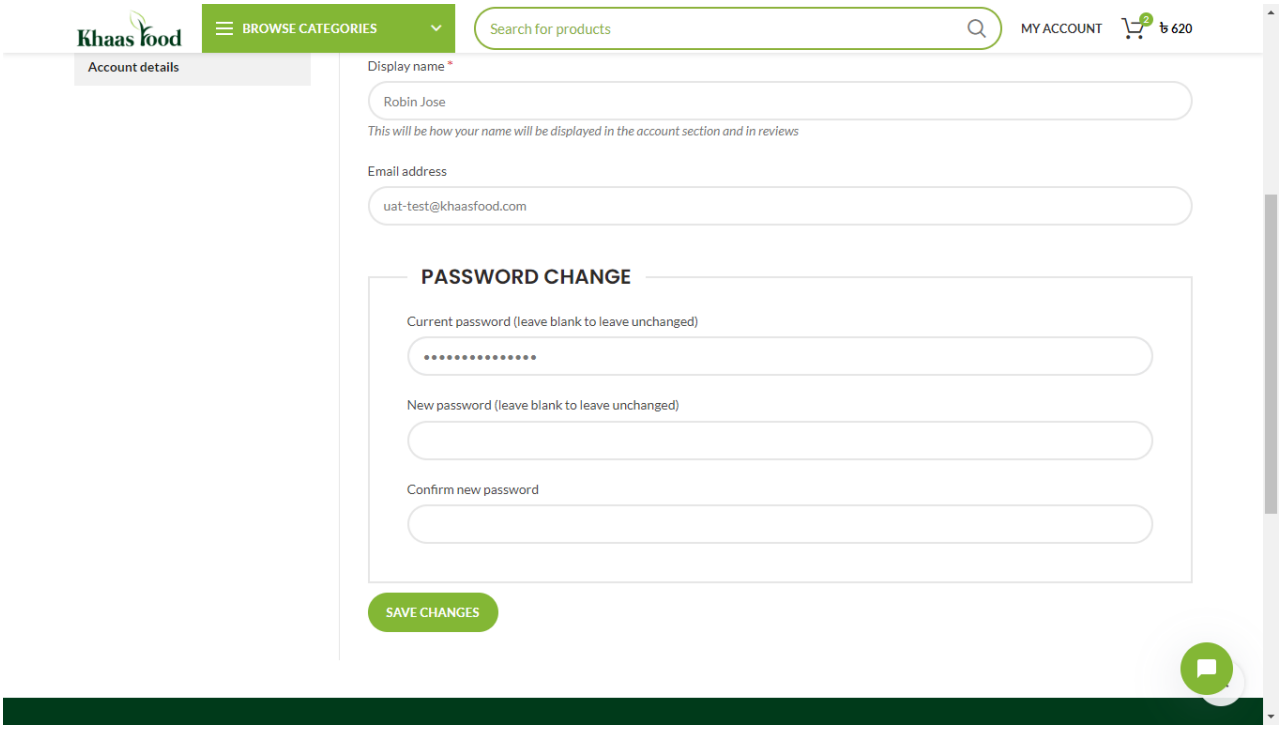
Customer – ***My Account***



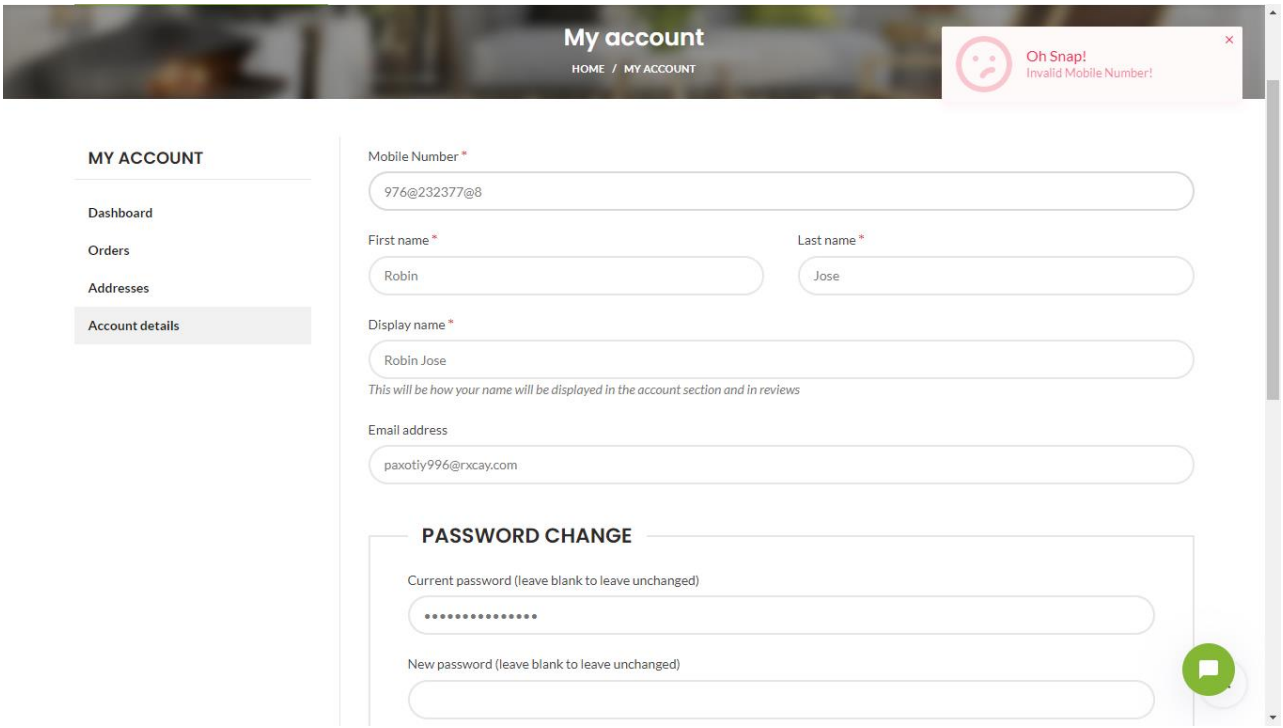
Customer – ***Account Details***



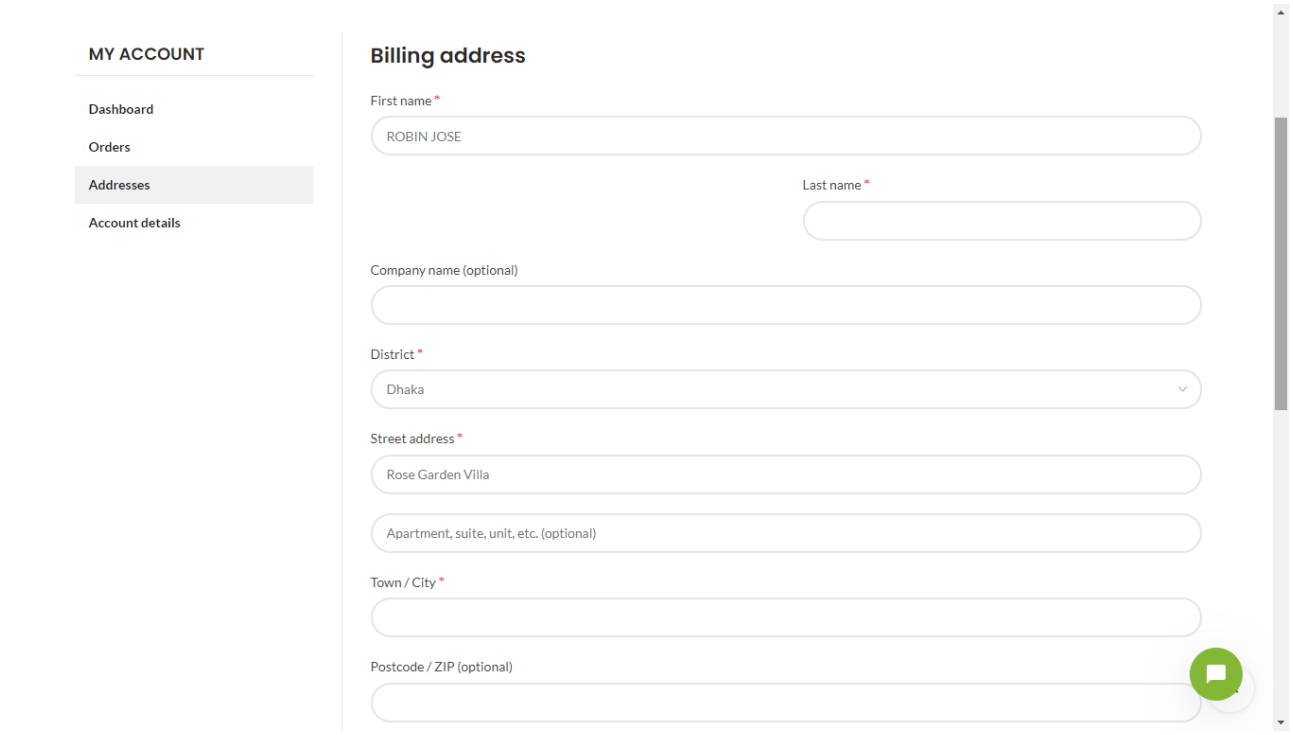
Customer – *Change Password*



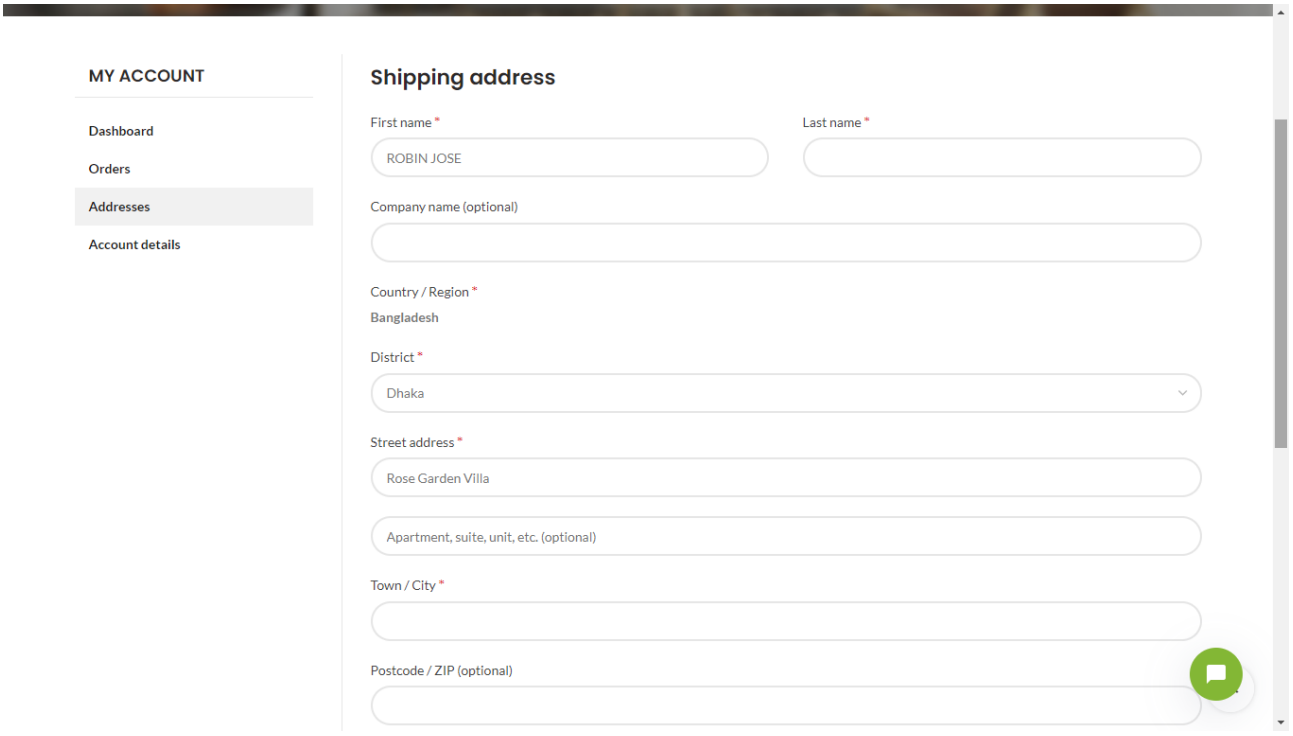
Customer – *Invalid Mobile Number Validation*



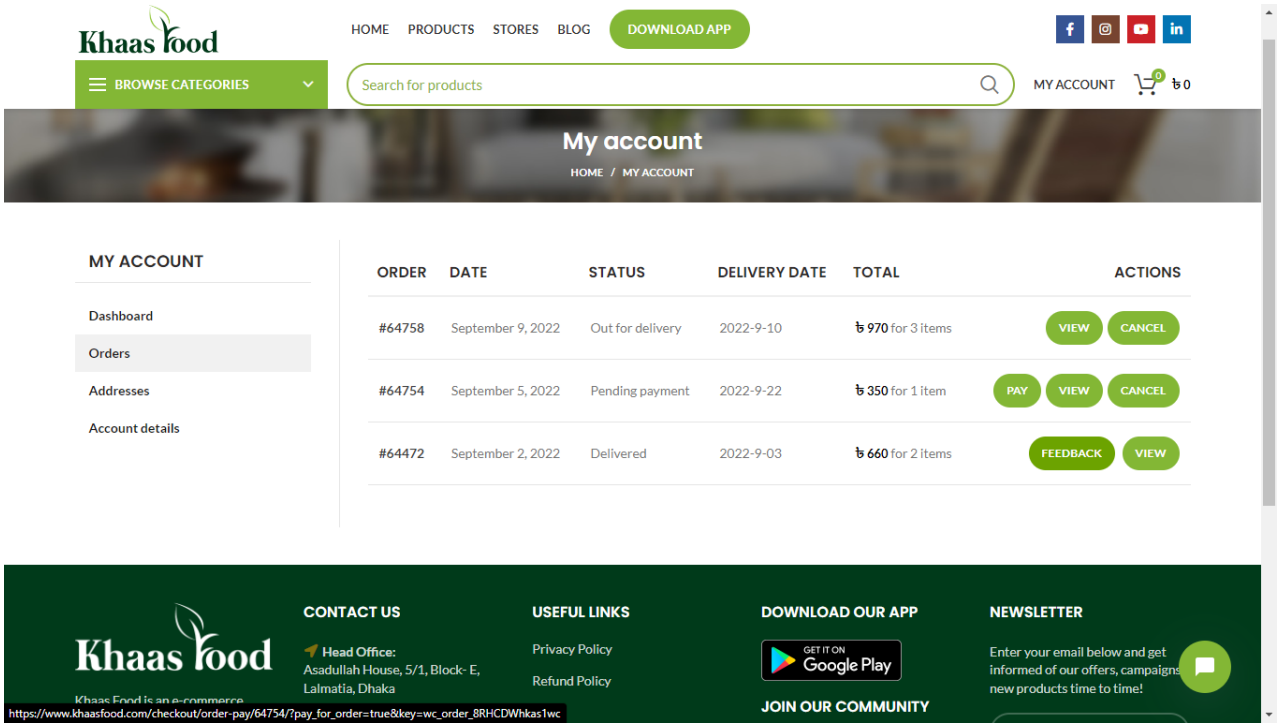
Customer – ***Billing Address Update***



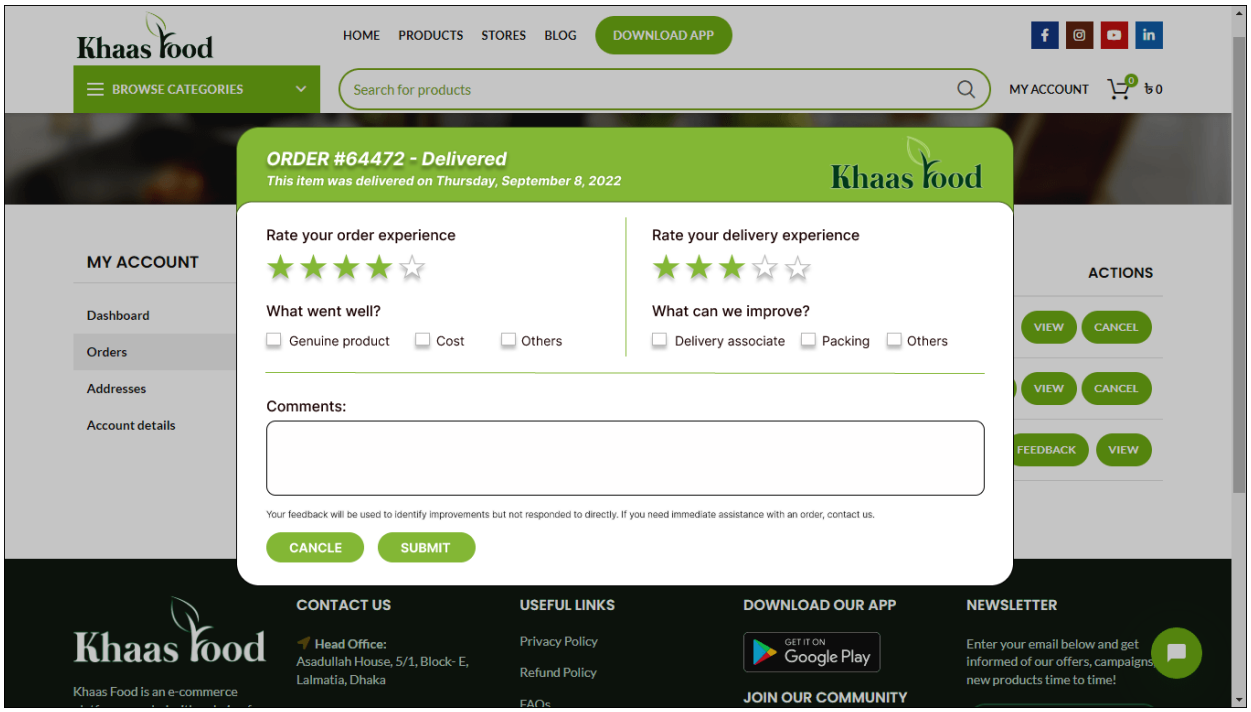
Customer – ***Updated Shipping Address***



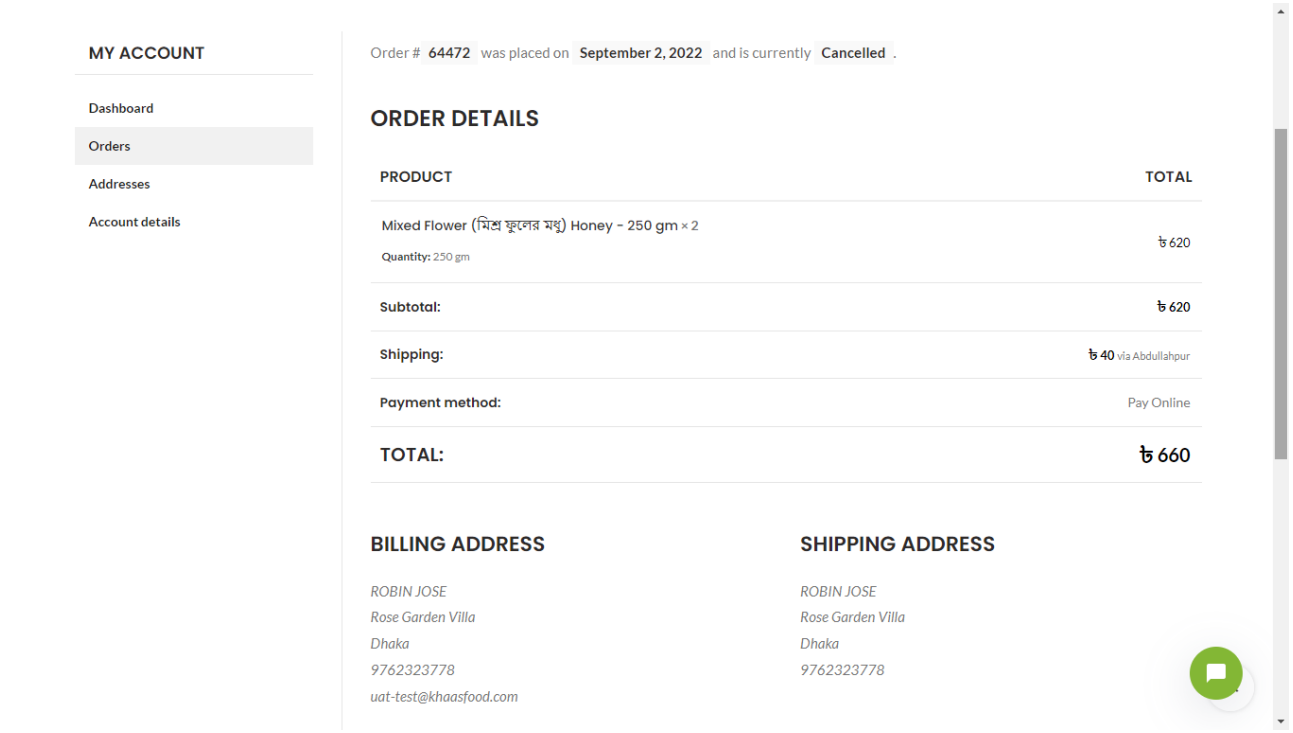
Customer – ***View Order History***



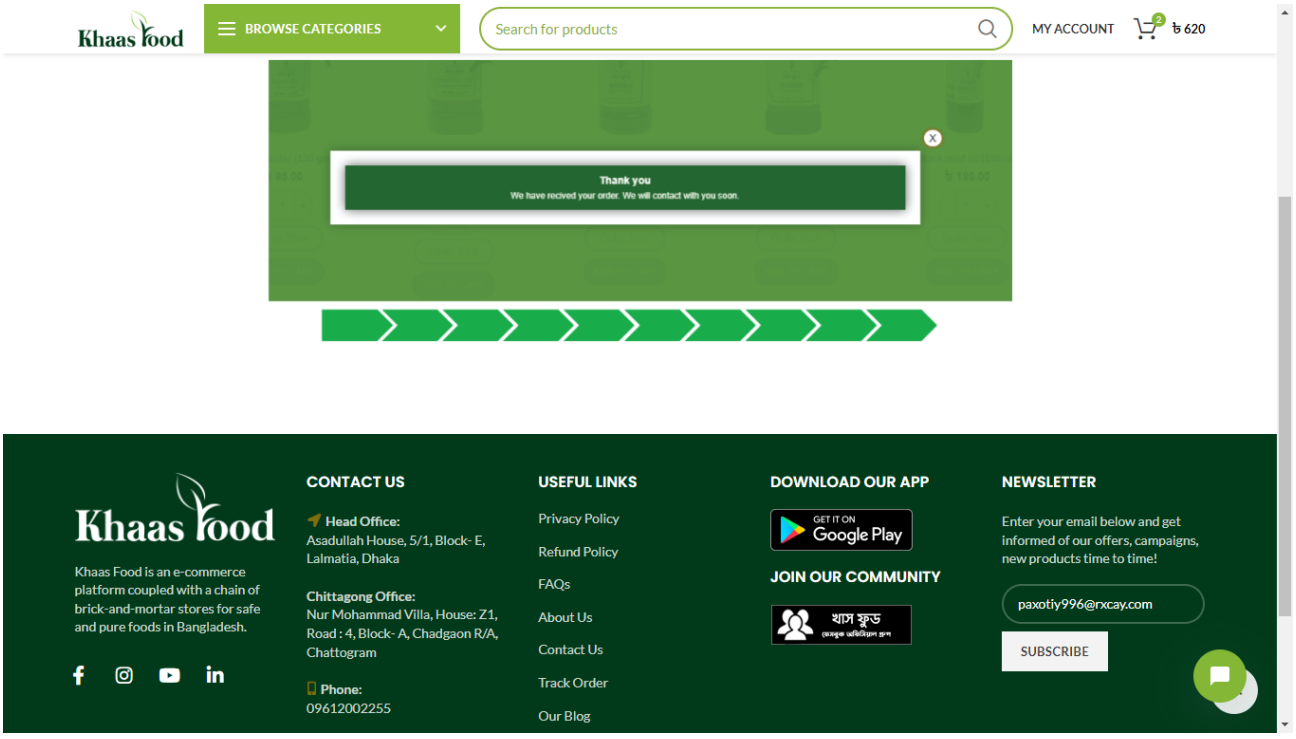
Customer – ***Feedback***



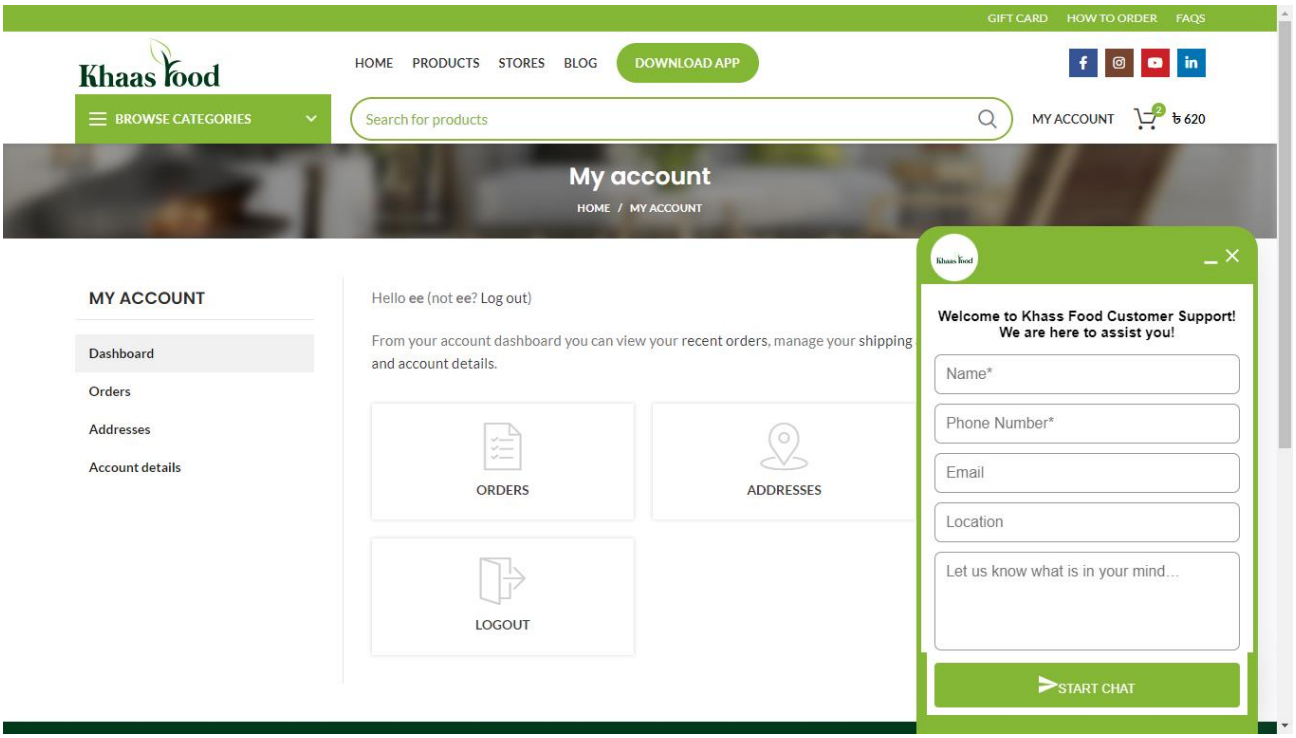
Customer – ***View Order Details***



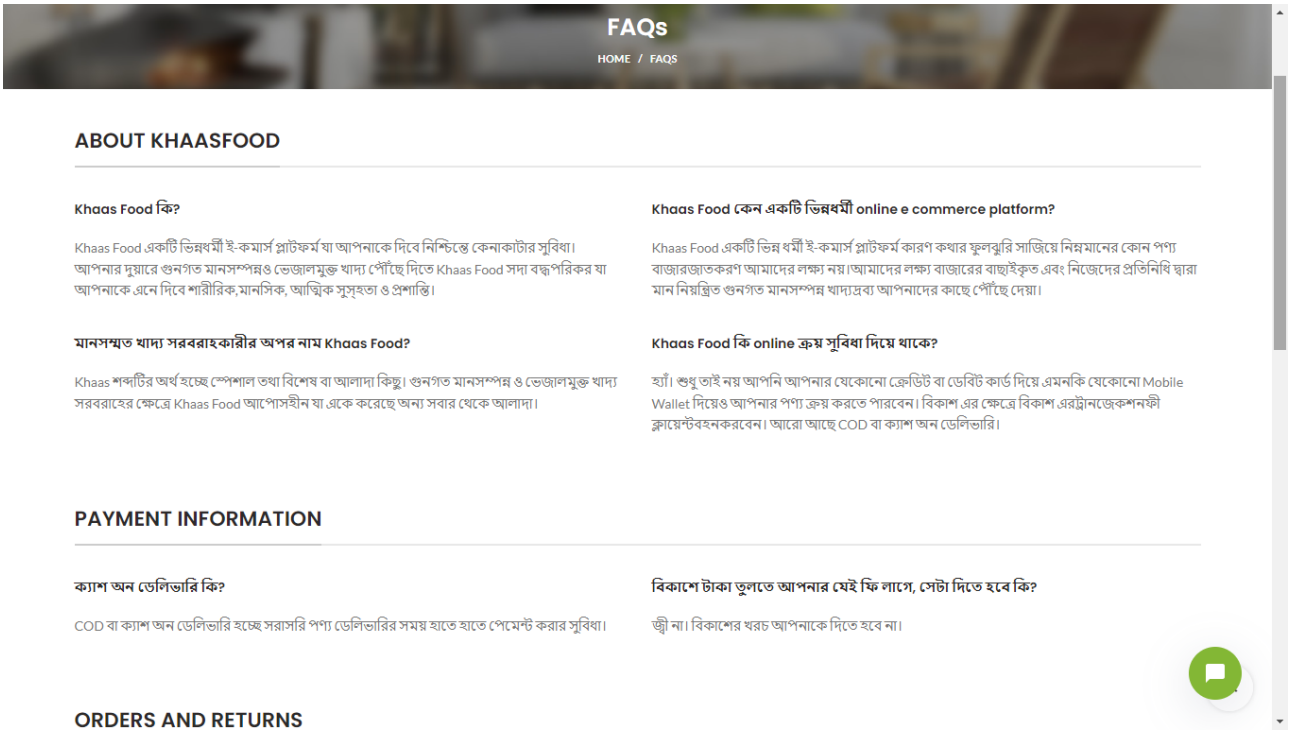
Customer – ***Order Placed Successfully***



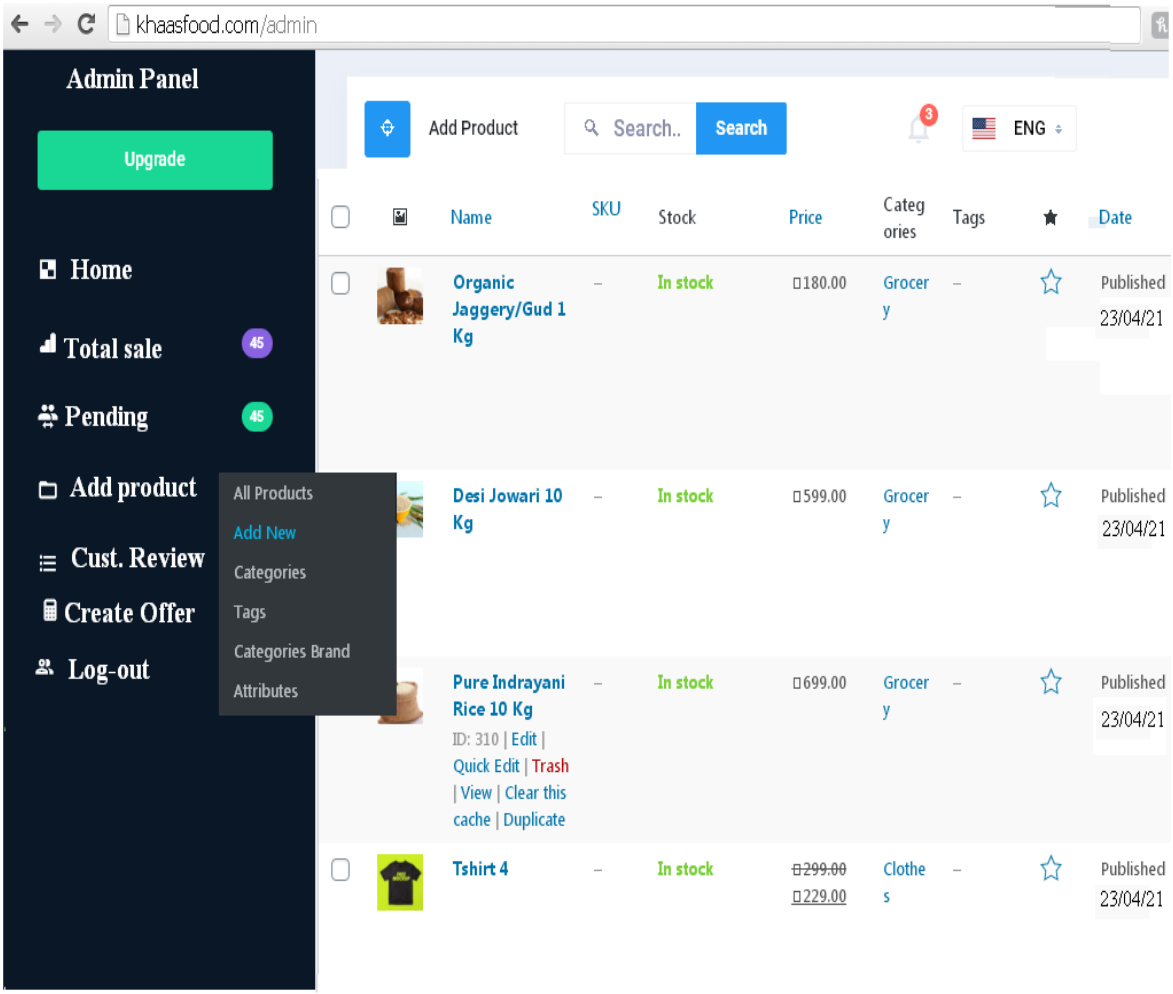
Customer – ***Chatbot***



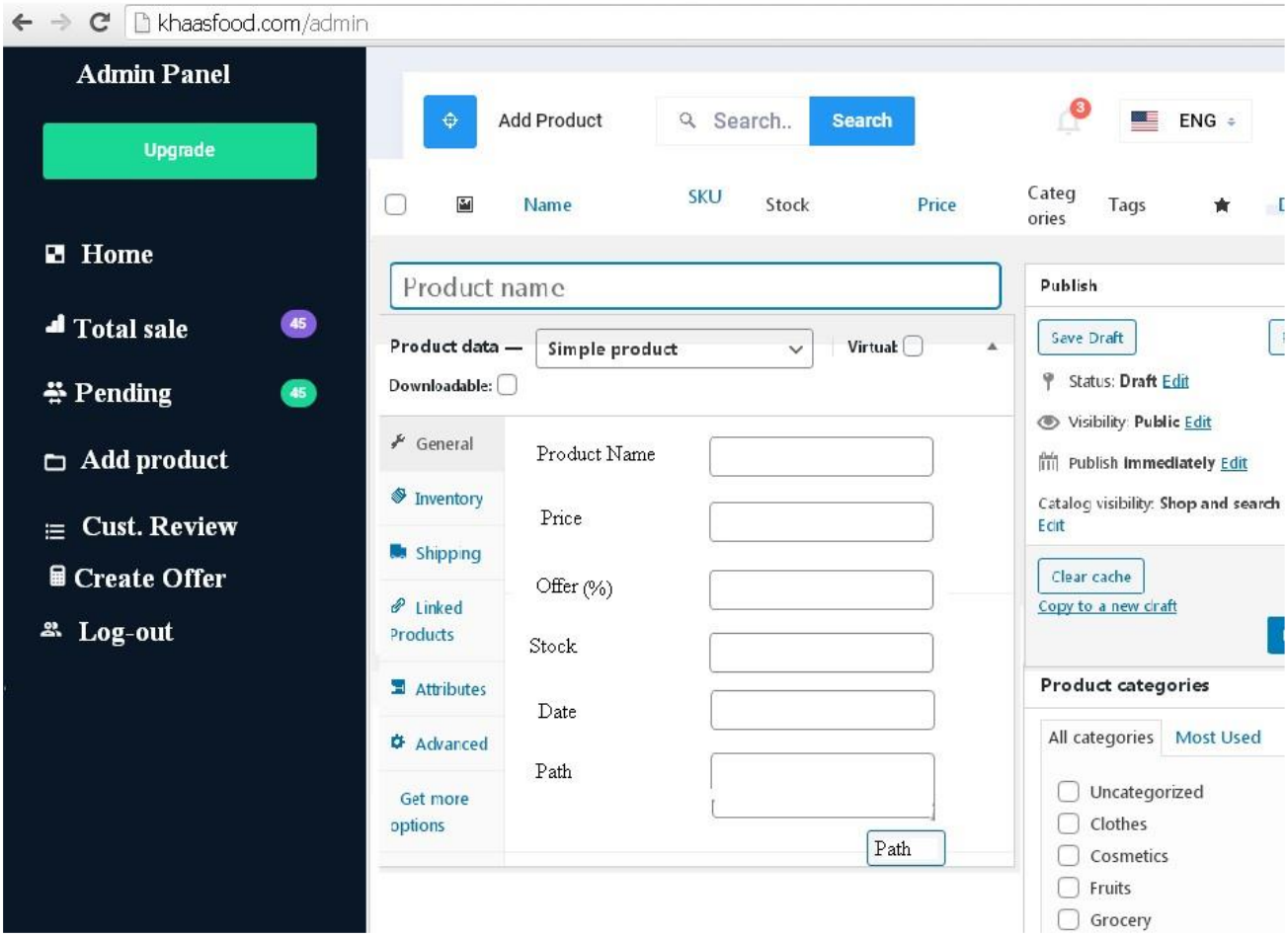
Customer - ***FAQ***



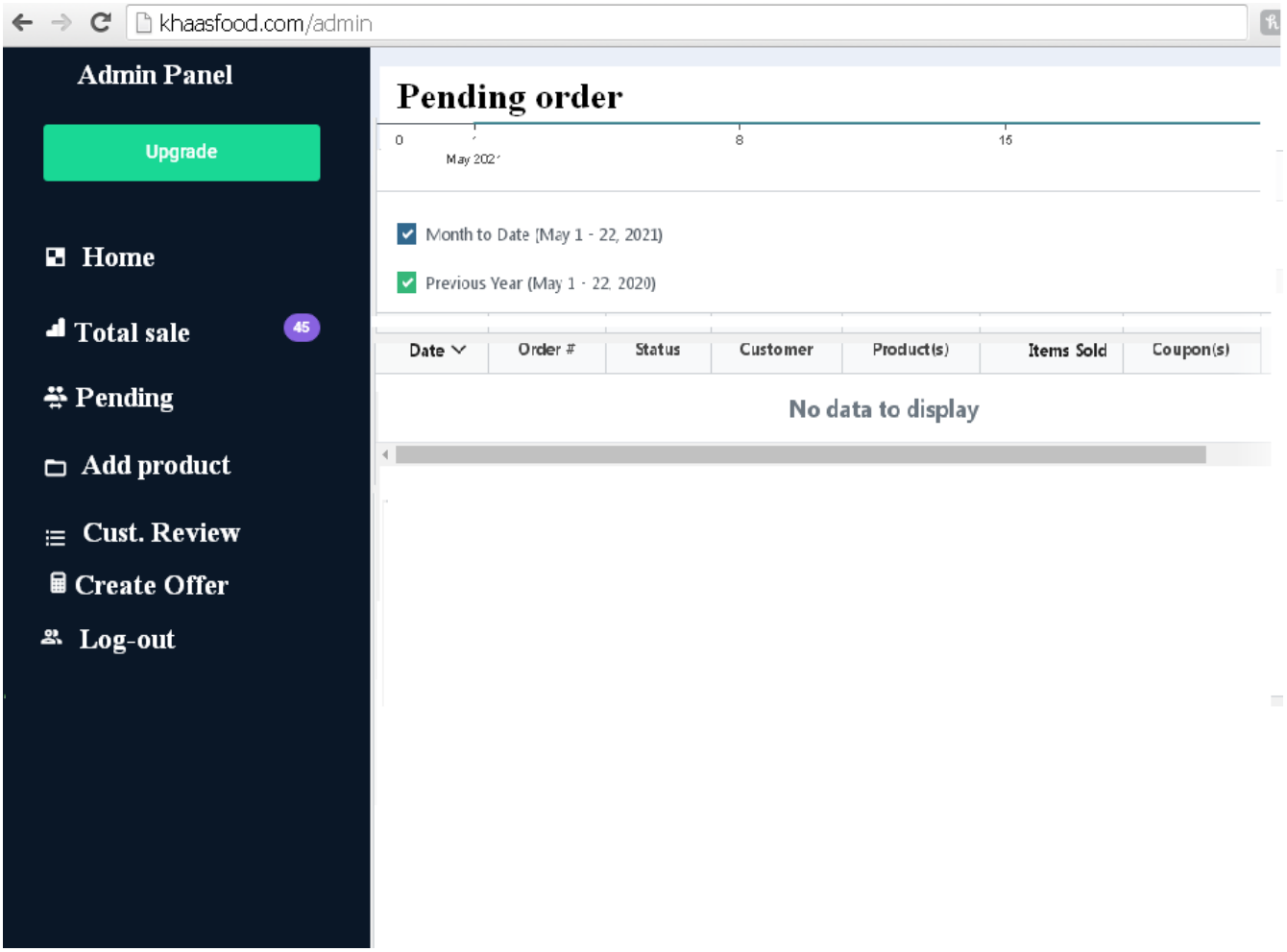
Admin – ***Dashboard***



Admin – Manage Product



Admin – Assign Order



4. Coding

4.1 Algorithms

1) As we click on application it open on web browser and home screen will be displayed.

2) Above at the top, there will be tabs displayed such as user, admin and about us.

3) Now, we have to do the login of the admin by entering username and password and then click on login button.

4) Once the login of admin is done, welcome page is displayed.

5) At the top of welcome page, we can view and search products based on categories.

6) User can view product and add then to cart.

7) Added items can be review and proceeded for checkout.

8) In checkout module user can use online payment or COD

9) After order is placed Admin assigns the order to a delivery partner

10) Delivery partner delivers the order and collects payment if order type is COD

11) Customer gives the order feedback

12) Admin will review the feedback and make changes as required

13) At last, we have to click on the logout button

4.2 Coding Snippets

- **build.gradle**

```
plugins {
   id 'org.springframework.boot' version '2.7.1'
   id 'io.spring.dependency-management' version '1.0.12.RELEASE'
   id 'com.gorylenko.gradle-git-properties' version '2.4.1'
   id 'java'
   id 'maven-publish'
}

group = 'com.ust.ptec.wit'
version = System.getenv("WIT_PROJECT_VERSION") ?: "0.0.1-SNAPSHOT"

sourceCompatibility = JavaVersion.VERSION_17
targetCompatibility = JavaVersion.VERSION_17

configurations {
   compileOnly {
      extendsFrom annotationProcessor
   }
}

repositories {
   mavenCentral()
}

publishing {
   publications {
      bootJava(MavenPublication) {
         artifact tasks.named("bootJar")
      }
   }
   repositories {
      maven {
         url "https://gitlab.glt.com/api/v4/projects/1355/packages/maven"
         name "GitLab"
         credentials(HttpHeaderCredentials) {
            name = 'Job-Token'
            value = System.getenv("CI_JOB_TOKEN")
         }
         authentication {
            header(HttpHeaderAuthentication)
         }
      }
   }
}

ext {
   set('mainClassName', 'com.ust.ptec.wit.WorkServiceApplication')
   set('buildNumber', version = System.getenv("CI_PIPELINE_ID") ?: "1")
}

dependencies {
   implementation 'org.springframework.boot:spring-boot-starter-actuator'
   implementation 'org.springframework.boot:spring-boot-starter-cache'
   implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
   implementation 'org.springframework.boot:spring-boot-starter-security'
```

```
    implementation 'org.springframework.boot:spring-boot-starter-validation'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation group: 'com.itextpdf', name: 'itextpdf', version: '5.0.6'
    implementation 'com.google.guava:guava:31.1-jre'
    implementation 'io.jsonwebtoken:jjwt-api:0.11.5'
    runtimeOnly 'io.jsonwebtoken:jjwt-impl:0.11.5'
    runtimeOnly 'io.jsonwebtoken:jjwt-jackson:0.11.5'
    implementation 'org.springdoc:springdoc-openapi-ui:1.6.9'
    implementation 'org.flywaydb:flyway-core'
    implementation 'org.mapstruct:mapstruct:1.5.2.Final'
    implementation 'org.springframework.boot:spring-boot-starter-mail:2.7.2'
    implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'
    implementation 'com.google.code.gson:gson:2.9.1'
    annotationProcessor 'org.projectlombok:lombok'
    annotationProcessor 'org.mapstruct:mapstruct-processor:1.5.2.Final'
    compileOnly 'org.projectlombok:lombok'
    runtimeOnly 'org.postgresql:postgresql'
    runtimeOnly 'com.github.ben-manes.caffeine:caffeine'
    developmentOnly 'org.springframework.boot:spring-boot-devtools'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testImplementation 'org.springframework.security:spring-security-test'
}

compileJava {
    options.compilerArgs = [
        '-Xlint:unchecked',
        '-Xlint:deprecation',
        '-Amapstruct.suppressGeneratorTimestamp=true',
        '-Amapstruct.suppressGeneratorVersionInfoComment=true',
        '-Amapstruct.defaultComponentModel=spring'
    ]
}

compileTestJava {
    options.compilerArgs = [
        '-Xlint:unchecked',
        '-Xlint:deprecation'
    ]
}

bootJar {
    enabled = true
    archiveFileName = project.name + '-' + project.version + '+' + project.buildNumber +
'.jar'
}

bootRun {
    mainClass.set(project.mainClassName)
    jvmArgs = [
        "-Xms256m",
        "-Xmx512m",
        "-Dspring.jmx.enabled=false",
        "-Dspring.main.lazy-initialization=true",
        "-Dapp.log-vol=/apps/wit/log",
        "-Dapp.data-vol=/apps/wit/data",
        "-Dlogging.config=classpath:logback-standard.xml",
        "-Dspring.profiles.active=local"
    ]
}
```

- **JpaUserDetailsServiceImpl.java**

```
@Component
public class JpaUserDetailsServiceImpl implements UserDetailsService {

    @Autowired
    private UserCredentialRepository userCredentialRepository;

    public UserDetails loadUserByUsername(final String username) throws
UsernameNotFoundException {
        final Optional<UserCredential> optUserCred =
this.userCredentialRepository.findByLoginId(username);
        if (optUserCred.isEmpty()) {
            throw new UsernameNotFoundException("User not found [" + username + "]");
        }
        final UserCredential userCred = optUserCred.get();
        return new User(username, userCred.getPassword(), Collections.emptyList());
    }

}
```

- **JwtTokenManager.java**

```
@Component
public class JwtAuthFilter extends OncePerRequestFilter {

    private static final String HEADER = "Authorization";

    private static final String PREFIX = "Bearer ";

    @Autowired
    private JwtTokenManager jwtTokenManager;

    protected void doFilterInternal(final @Nonnull HttpServletRequest request, final
@Nonnull HttpServletResponse response,
        final @Nonnull FilterChain chain) throws ServletException, IOException {
        final Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();
        if ((authentication != null) && authentication.isAuthenticated()) {
            chain.doFilter(request, response);
            return;
        }
        final String authenticationHeader = request.getHeader(HEADER);
        if ((authenticationHeader == null) || (!authenticationHeader.startsWith(PREFIX)))
{
            chain.doFilter(request, response);
            return;
        }
        final String jwtToken = authenticationHeader.replace(PREFIX, "").trim();
        if (!this.jwtTokenManager.validateToken(jwtToken)) {
            SecurityContextHolder.clearContext();
            response.setStatus(HttpServletResponse.SC_FORBIDDEN);
            response.sendError(HttpServletResponse.SC_FORBIDDEN, "Invalid token");
            return;
        }
        final JwtAuthenticationToken authToken = new JwtAuthenticationToken(jwtToken,
Collections.emptyList());
        authToken.setAuthenticated(true);
        SecurityContextHolder.getContext().setAuthentication(authToken);
        chain.doFilter(request, response);
    }

}
```

- **WorkflowLogServiceImpl.java**

```
@Slf4j
@Service
public class WorkflowLogServiceImpl implements WorkflowLogService {

    @Autowired
    WITRequestService witRequestService;

    @Autowired
    WorkflowLogMapper workflowLogMapper;

    @Autowired
    WorkflowLogRepository workflowLogRepository;

    @Override
    public Optional<WorkflowLogDTO> findRequestByRequestId(String requestId) {
        return this.workflowLogRepository.findByRequestId(requestId).map(request ->
workflowLogMapper.toDTO(request));
    }

    @Override
    public Optional<WorkflowLogDTO> save(final WorkflowLogDTO workflowLogDTO) {

        String username = getCurrentUsername();
//        TODO: Fetch current user role based on username
        User user = null;

        Optional<WITRequestDTO> witRequestDTO =
witRequestService.findRequestByRequestId(workflowLogDTO.getRequestId());
        if (witRequestDTO.isEmpty()) {
            return Optional.empty();
        }

        WITRequestDTO request = witRequestDTO.get();
        Map<RequestWorkflowConditionalStatusType, RequestStatusRule<RequestStatusType>>
ruleMap = createRequestAndWorkflowRules(username, request.getStatus().name(),
workflowLogDTO.getWorkflowStatus().name());
        RequestStatusType requestStatusType =
Stream.of(RequestWorkflowConditionalStatusType.values())
                .filter(x -> ruleMap.get(x).userRole.get())
                .filter(x -> ruleMap.get(x).requestStatus.get())
                .filter(x -> ruleMap.get(x).workflowStatus.get())
                .map(x -> ruleMap.get(x).updatedRequestStatus.get())
                .findFirst()
                .orElseThrow(() -> new
UnsupportedOperationException(MessageFormat.format("This operation is not allowed. user-
role: {0}, request-status: {1}", username, request.getStatus().name())));

        request.setStatus(requestStatusType);
        WorkflowLog workflowLog =
this.workflowLogRepository.save(this.workflowLogMapper.toModel(workflowLogDTO));
        this.witRequestService.updateByRequestId(workflowLogDTO.getRequestId(), request);

//        TODO: Send Notification
        sendWorkflowNotification(user);
        return this.findRequestByRequestId(workflowLog.getRequestId());
    }

    private void sendWorkflowNotification(User user) {

    }

    public Map<RequestWorkflowConditionalStatusType,
RequestStatusRule<RequestStatusType>> createRequestAndWorkflowRules(String userRole,
String requestStatus, String workflowStatus) {
```

```
        Map<RequestWorkflowConditionalStatusType, RequestStatusRule<RequestStatusType>>
ruleMap = new HashMap<>();

ruleMap.put(RequestWorkflowConditionalStatusType.PENDING_BU_APPROVAL_WORKFLOW_APPROVED,
pendingBuApprovalAndWorkflowApproved(userRole, requestStatus, workflowStatus));

ruleMap.put(RequestWorkflowConditionalStatusType.PENDING_BU_APPROVAL_WORKFLOW_REJECTED,
pendingBuApprovalAndWorkflowRejected(userRole, requestStatus, workflowStatus));

ruleMap.put(RequestWorkflowConditionalStatusType.PENDING_FI_APPROVAL_WORKFLOW_APPROVED,
pendingFiApprovalAndWorkflowApproved(userRole, requestStatus, workflowStatus));

ruleMap.put(RequestWorkflowConditionalStatusType.PENDING_FI_APPROVAL_WORKFLOW_REJECTED,
pendingFiApprovalAndWorkflowRejected(userRole, requestStatus, workflowStatus));

        return ruleMap;
    }

    private RequestStatusRule<RequestStatusType>
pendingBuApprovalAndWorkflowApproved(String userRole, String requestStatus, String
workflowStatus) {
        return createRule(
                () -> userRole.equals("BU Manager"),
                () -> requestStatus.equals(RequestStatusType.PENDING_BU_APPROVAL.name()),
                () -> workflowStatus.equals(WorkflowStatusType.APPROVED.name()),
                () -> RequestStatusType.PENDING_FI_APPROVAL
        );
    }

    private RequestStatusRule<RequestStatusType>
pendingBuApprovalAndWorkflowRejected(String userRole, String requestStatus, String
workflowStatus) {
        return createRule(
                () -> userRole.equals("BU Manager"),
                () -> requestStatus.equals(RequestStatusType.PENDING_BU_APPROVAL.name()),
                () -> workflowStatus.equals(WorkflowStatusType.REJECTED.name()),
                () -> RequestStatusType.REJECTED
        );
    }

    private RequestStatusRule<RequestStatusType>
pendingFiApprovalAndWorkflowApproved(String userRole, String requestStatus, String
workflowStatus) {
        return createRule(
                () -> userRole.equals("FI Manager"),
                () -> requestStatus.equals(RequestStatusType.PENDING_FI_APPROVAL.name()),
                () -> workflowStatus.equals(WorkflowStatusType.APPROVED.name()),
                () -> RequestStatusType.REQUESTED
        );
    }

    private RequestStatusRule<RequestStatusType>
pendingFiApprovalAndWorkflowRejected(String userRole, String requestStatus, String
workflowStatus) {
        return createRule(
                () -> userRole.equals("FI Manager"),
                () -> requestStatus.equals(RequestStatusType.PENDING_FI_APPROVAL.name()),
                () -> workflowStatus.equals(WorkflowStatusType.REJECTED.name()),
                () -> RequestStatusType.REJECTED
        );
    }

    private RequestStatusRule<RequestStatusType> createRule(Supplier<Boolean> userRole,
                                                             Supplier<Boolean>
requestStatus,
```

```
                                                        Supplier<Boolean>
workflowStatus,
                                                        Supplier<RequestStatusType>
process) {
        return new RequestStatusRule<>(userRole, requestStatus, workflowStatus, process);
    }

    private String getCurrentUsername() {
        Object principal =
SecurityContextHolder.getContext().getAuthentication().getPrincipal();
        return principal instanceof UserDetails
                ? ((UserDetails) principal).getUsername()
                : principal.toString();
    }
}
```

- **OrderRequestServiceImpl.java**

```
@Slf4j
@Service
public class OrderRequestServiceImpl implements WITRequestService {

    @Autowired
    private ApplicationEventPublisher eventPublisher;

    @Autowired
    FileExportService fileExportService;

    @Autowired
    WITRequestMapper witRequestMapper;

    @Autowired
    WITRequestRepository witRequestRepository;

    @Autowired
    EmailService emailService;

    @Override
    public Optional<WITRequestDTO> save(final WITRequestDTO witRequestDTO) {
        WITRequest witRequest =
this.witRequestRepository.save(this.witRequestMapper.toModel(witRequestDTO));
        this.eventPublisher.publishEvent(new
EntityCreateEvent(SecurityContextHolder.getContext().getAuthentication(), witRequest));
        return this.findRequestByRequestId(witRequest.getRequestId());
    }

    @Override
    public Page<WITRequestDTO> findRequestsByFilter(final Optional<String> search, final
String[] sortBy, final String sortDirection, final Integer page, final Integer pageSize)
{
        Pageable pageable = PageRequest.of(page, pageSize,
Sort.Direction.fromString(sortDirection), sortBy);
        Page<WITRequest> requestDetail = search.isPresent()
                ? this.witRequestRepository.findAll(pageable, search.get())
                : this.witRequestRepository.findAll(pageable);
        return requestDetail.map(witRequest -> this.witRequestMapper.toDTO(witRequest));
    }

    @Override
    public Optional<WITRequestDTO> findRequestByRequestId(String requestId) {
        return this.witRequestRepository.findByRequestId(requestId).map(request ->
this.witRequestMapper.toDTO(request));
    }

    @Override
```

```
    public Optional<WITRequestDTO> deleteByRequestId(String requestId) {
        return this.witRequestRepository.findByRequestId(requestId)
                .map(request -> {
                    if (!request.getStatus().equals(RequestStatusType.DRAFT))
                        throw new UnsupportedDeleteOperationException("Draft request can
only be deleted");
                    request.setActive(false);
                    this.witRequestRepository.save(request);
                    return Optional.of(this.witRequestMapper.toDTO(request));
                }).orElseGet(Optional::empty);
    }

    @Override
    public Optional<WITRequestDTO> updateByRequestId(final String requestId, final
WITRequestDTO witRequestDTO) {
        return this.witRequestRepository.findByRequestId(requestId).map(request -> {
            this.witRequestMapper.updateWITRequestFromDto(witRequestDTO, request);
            this.witRequestRepository.save(request);
            return Optional.of(this.witRequestMapper.toDTO(request));
        }).orElseGet(Optional::empty);
    }

    @Override
    public Resource getWitRequestPdf(String witRequestId) {
        Optional<WITRequestDTO> witRequestDTO =
witRequestRepository.findByRequestId(witRequestId)
                .map(request -> Optional.of(this.witRequestMapper.toDTO(request)))
                .orElseThrow(()->new
NoDataFoundException(WITConstants.NO_DATA_FOUND_MESSAGE + " Wit Request ID
:"+witRequestId));
        return new ByteArrayResource(fileExportService.getWitRequestPdf(witRequestDTO,
witRequestId));
    }

    @Override
    public Optional<RequestDashboardDTO> fetchRequestDashboardDetails() {
        List<WITRequest> witRequests = witRequestRepository.findAll();
        return witRequests.isEmpty() ? Optional.empty()
:Optional.of(getRequestDashboard(witRequests));

    }

    private RequestDashboardDTO getRequestDashboard(List<WITRequest> witRequests){
        int totalRequests =witRequests.size();
        Map<RequestStatusType, String> map =
witRequests.stream().collect(Collectors.groupingBy(WITRequest::getStatus,
Collectors.collectingAndThen(Collectors.counting(), count -> (String.format("%.2f%%",
((float)(count * 100) / totalRequests))))));
        return new RequestDashboardDTO(totalRequests, map.get(DRAFT),
map.get(PENDING_BU_APPROVAL), map.get(PENDING_FI_APPROVAL), map.get(REQUESTED),
map.get(PROCESSED), map.get(REJECTED));
    }

    public void exportPdfAndSendEmail(String requestId, Optional<UserDTO> userOpt) {
        Resource witRequestPdf = getWitRequestPdf(requestId);
        File file = new File( TEMP_FOLDER_PATH+ requestId + FILE_EXTENSION);
        try {
            Files.copy(witRequestPdf.getInputStream(), file.toPath(),
StandardCopyOption.REPLACE_EXISTING);
            Map<String, Object> properties = new HashMap<>();
            if (userOpt.isPresent()) {
                UserDTO user = userOpt.get();
                properties.put("name", user.getFirstName());
                properties.put("WIT", requestId);
                emailService.sendMail(new EmailDetailDTO(user.getEmail(), null, null,
Arrays.asList(user.getId()), properties, file.getAbsolutePath(), file.getName()));
```

```
            } else{
                logger.info("User details is not present");
            }
        } catch (Exception e) {
            logger.error("Error in Exporting File/Sending Email \n {}", e);
        } finally {
            deleteTempFile(file.toPath());
        }
    }

    private void deleteTempFile(Path path) {
        try {
            Files.delete(path);
        } catch (IOException e) {
            logger.error("Error while deleting temp file {}", e);
        }
    }
}
```

- RequestStatusRole.java

```
@NoArgsConstructor
@AllArgsConstructor
public class RequestStatusRule<T> {
    public Supplier<Boolean> userRole;
    public Supplier<Boolean> requestStatus;
    public Supplier<Boolean> workflowStatus;
    public Supplier<T> updatedRequestStatus;
}
```

- UserController.java

```
@RestController
@RequestMapping(path = USERS)
public class UserController {

    @Autowired
    private UserService userService;

    @GetMapping(path = USER_ID)
    @ApiResponses(value = {
            @ApiResponse(responseCode = "200", description = "Success", content =
@Content),
            @ApiResponse(responseCode = "400", description = "Invalid data/request",
content = @Content),
            @ApiResponse(responseCode = "401", description = "Auth information is
missing/invalid", content = @Content),
            @ApiResponse(responseCode = "403", description = "Operation denied", content
= @Content),
            @ApiResponse(responseCode = "404", description = "Not found", content =
@Content),
            @ApiResponse(responseCode = "500", description = "Internal error", content =
@Content)
    })
    @Operation(summary = "Get User By User ID", description = "Get User By User ID")
    public ResponseEntity<UserDTO> getByUserID(@PathVariable(name = "userId") Long
userid) {

        Optional<UserDTO> requestDTO = this.userService.findByUserID(userid);
        return requestDTO.isEmpty() ? ResponseEntity.notFound().build() :
ResponseEntity.ok(requestDTO.get());
    }
```

```
    @GetMapping
    @ApiResponses(value = {
            @ApiResponse(responseCode = "200", description = "Success", content =
@Content),
            @ApiResponse(responseCode = "400", description = "Invalid data/request",
content = @Content),
            @ApiResponse(responseCode = "401", description = "Auth information is
missing/invalid", content = @Content),
            @ApiResponse(responseCode = "403", description = "Operation denied", content
= @Content),
            @ApiResponse(responseCode = "404", description = "Not found", content =
@Content),
            @ApiResponse(responseCode = "500", description = "Internal error", content =
@Content)
    })
    @Operation(summary = "Get All Users", description = "Get All Users")
    public ResponseEntity<List<UserDTO>> getAll() {
        return ResponseEntity.ok(this.userService.findAll());
    }

    @PostMapping(produces = APPLICATION_JSON_VALUE)
    @ApiResponses(value = {
            @ApiResponse(responseCode = "201", description = "Created", content =
@Content),
            @ApiResponse(responseCode = "400", description = "Invalid data/request",
content = @Content),
            @ApiResponse(responseCode = "401", description = "Auth information is
missing/invalid", content = @Content),
            @ApiResponse(responseCode = "403", description = "Operation denied", content
= @Content),
            @ApiResponse(responseCode = "404", description = "Not found", content =
@Content),
            @ApiResponse(responseCode = "500", description = "Internal error", content =
@Content)
    })
    @Operation(summary = "Create Users", description = "Create Users")
    public ResponseEntity<Map<String, String>> createUsers(@Validated @RequestBody
List<UserDTO> users) {

        Optional<List<String>> userDtoOpt = Optional.empty();
        if(!users.isEmpty())
            userDtoOpt  = userService.save(users);

        Map<String, String> responseMap = new HashMap<>();
        String empIds =
userDtoOpt.get().stream().map(String::valueOf).collect(Collectors.joining(", "));
        responseMap.put("empIds",empIds);

        return userDtoOpt.isEmpty() ?
                ResponseEntity.notFound().build() :
                ResponseEntity.ok(responseMap);

    }

    @PutMapping(path = USER_ID, produces = APPLICATION_JSON_VALUE)
    @ApiResponses(value = {
            @ApiResponse(responseCode = "200", description = "Success", content =
@Content),
            @ApiResponse(responseCode = "400", description = "Invalid data/request",
content = @Content),
            @ApiResponse(responseCode = "401", description = "Auth information is
missing/invalid", content = @Content),
            @ApiResponse(responseCode = "403", description = "Operation denied", content
= @Content),
            @ApiResponse(responseCode = "404", description = "Not found", content =
```

```
@Content),
            @ApiResponse(responseCode = "500", description = "Internal error", content =
@Content)
    })
    @Operation(summary = "Edit User by User ID", description = "Edit User by User ID")
    public ResponseEntity editUser(@PathVariable(value = "userId") String userId,
                                                    @Validated @RequestBody UserDTO
userDTO) {


        if(Optional.ofNullable(userDTO.getEmployeeId()).isPresent())
                        return ResponseEntity.badRequest().body("Message : Employee ID
cannot be part of edit request");

        Optional<UserDTO> editedUserDto = userService.editUser(Long.parseLong(userId),
userDTO);

        return editedUserDto.isEmpty() ?
                ResponseEntity.notFound().build() :
                ResponseEntity.ok().body(editedUserDto);
    }

    @DeleteMapping(path = USER_ID)
    @ApiResponses(value = {
            @ApiResponse(responseCode = "200", description = "Success", content =
@Content),
            @ApiResponse(responseCode = "400", description = "Invalid data/request",
content = @Content),
            @ApiResponse(responseCode = "401", description = "Auth information is
missing/invalid", content = @Content),
            @ApiResponse(responseCode = "403", description = "Operation denied", content
= @Content),
            @ApiResponse(responseCode = "404", description = "Not found", content =
@Content),
            @ApiResponse(responseCode = "500", description = "Internal error", content =
@Content)
    })
    @Operation(summary = "Soft delete User by User ID", description = "Soft delete User
by User ID")
    public ResponseEntity deleteUserById(@PathVariable(name = "userId") String userId) {
        Optional<UserDTO> userDTO =
this.userService.deleteByUserId(Long.parseLong(userId));
        return userDTO.isEmpty() ? ResponseEntity.notFound().build() :
ResponseEntity.ok().build();
    }

}
```

5. Testing

5.1 Test Strategy

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

5.2 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code own should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive.

Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique pathof a business process performs accurately to the documented specifications. Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

5.3 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results:

All the test cases mentioned above passed successfully. No defects encountered.

White Box Testing:

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing:

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

System Test:

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and owns, emphasizing pre-driven process links and integration point

5.4 Test Cases

Test Case Id: *UAT-Khass-Admin-0056*

Testcase Id	Summery	Prediction	Execution Step	Expected Result	Actual Result	Pass/ Fail
1	Verify the site should be open successfully	1.Insert the link into URL field	1.click on the search icon	The site should be open successfully	The site opened successfully	Pass
2	Verify user login	Insert login credential Username Password	Click on login button	Login successfully y	Logged in successfully	Pass
3	Search Item	Insert search item in search bar	click on search button	Matching products should be displayed	Matching products should be displayed	Pass
4	Select Item	Select an item	Select the item	Item desc should be displayed	Item desc should be displayed	Pass
5	Verify Payment	Select Checkout option	Select online payment	Payment successful	Payment successful	Pass

6	Cancel Order	Click on cancel order	Select cancel order	Cancel order successful	Order canceled	Pass
7	Verify Customer logout	Click on logout option	Select logout button	Logout successful	Logout successful	Pass
8	Verify admin login	Insert admin data username and password	Click on admin button	Login successfully	Logged in successfully	Pass
9	add new category	Admin add new category	Click on ok button	Successfully added category	New Category added	Pass
10	add product	Admin add product	Click on ok button	Admin add the product	Add product	Pass
11	Add Delivery Partner	Admin add delivery partner	Click on ok button	Admin add the delivery partner	Add delivery partner	Pass
12	View the all booking	Admin view the all booking	Click on ok button	Admin view the all booking	View booking	Pass
13	Assign Order	Admin assign order to DP	Click on ok button	Admin assigned order	Order Assigned	Pass

6. Limitations of Proposed System

- Currently the system is only active in 2 cites (Dhaka, Chittagong)
- More languages option should be provided
- EMI currently not available in the system
- Automating the order assign process needed
- Live tracking of delivery partner is not available

7. Proposed Enhancements

- Presence Across Nation (PAN) expansion currently only in Dhaka, Chittagong
- Multilingual website should majorly support (বাংলা | English | हिन्दी)
- Integrating EMI option in payment
- Automating the order assign process
- Live tracking of delivery partner
- WhatsApp API integration for order updates

8. Conclusions

1. This in turn translated to development of an application that met the set objectives and the user requirements also.

2. Hence, most of the set objectives were met with the implementation of the application as it facilitated constant inflow of information from stakeholders that helped incoming up with a good design for the application.

9. Bibliography

Book References

- Spring Boot 2 - Reference Book
- PostgreSQL DBA Digest

Web References

- https://github.com/spring-projects/spring-boot
- https://www.baeldung.com/spring-boot
- https://spring.io/projects/spring-boot/
- https://www.javatpoint.com/java-tutorial
- https://www.stackoverflow.com/