

GustoGrove

A Comprehensive Spice Mart

Mini Project Report

Submitted by

ALEENA THOMAS

Reg. No.: AJC19MCA-I008

In Partial fulfillment for the Award of the Degree of

INTEGRATED MASTER OF COMPUTER APPLICATIONS

(INMCA)

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY



AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC with 'A' grade. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

2023-2024

DEPARTMENT OF COMPUTER APPLICATIONS
AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPPALLY



CERTIFICATE

This is to certify that the Project report, “**GustoGrove**” is the bonafide work of **STUDENT NAME (Regno: AJC19MCA-I008)** in partial fulfillment of the requirements for the award of the Degree of Integrated Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2023-24.

Mr.Jobin T J

Internal Guide

Ms.Meera Rose Mathew

Coordinator

Rev. Fr. Dr. Rubin Thottupurathu Jose

Head of the Department

DECLARATION

I hereby declare that the project report **“GustoGrove”** is a bona fide work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2023-2024.

Date: 25-10-2023

ALEENA THOMAS

KANJIRAPPALLY

Reg: AJC19MCA-I008

ACKNOWLEDGEMENT

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our Manager **Rev. Fr. Dr. Mathew Paikatt** and Principal **Dr. Lillykutty Jacob** for providing good faculty for guidance.

I owe a great depth of gratitude towards our Head of the Department **Rev.Fr.Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the project coordinator **Ms.Meera Rose Mathew** for his valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also express sincere gratitude to my guide **Mr.Jobin T J** for her inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions, and encouragement to make this venture a success.

ALEENA THOMAS

ABSTRACT

The "GustoGrove- Building a Comprehensive Spice Mart" project aims to create an all-inclusive online platform dedicated to providing a wide range of spices for customers. The e-commerce website offers an immersive and user-friendly experience, catering to the needs of spice enthusiasts looking for high-quality and diverse spice options. This project combines the convenience of online shopping with comprehensive spice information, user reviews, and an intuitive interface to enhance the spice buying experience.

The website features a visually captivating home page, welcoming visitors with enticing visuals of various spices. A robust search functionality and suggested search terms assists users in quickly finding their desired spices. The product listing section showcases an extensive collection of spices, and enabling seamless navigation and exploration.

Each spice is presented with detailed information, including descriptions, prices, and high-resolution images, allowing customers to make informed choices.

The shopping cart functionality allows customers to review their selected spices, adjust quantities, and calculate the total cost. The streamlined checkout process ensures a hassle-free transaction, providing various payment options for customer convenience.

The website also offers user accounts with personalized profiles, enabling customers to track their order history, view delivery status. Engaging content such as blogs or recipes related to spices enhances customer engagement and provides additional value.

Contact and customer support features are available, including a dedicated contact form, FAQs section, and potentially a live chat or chatbot functionality for immediate assistance. The about us section presents the spice mart's background, sourcing practices, commitment to quality, and team expertise, establishing trust and credibility among customers. Provide personalized recommendations for customers.

The comprehensive spice mart e-commerce website aims to deliver a seamless shopping experience, presenting an extensive range of spices, easy navigation, detailed information, and customer-centric features.

CONTENT

SL. NO	TOPIC	PAGE NO
1	INTRODUCTION	1
1.1	PROJECT OVERVIEW	2
1.2	PROJECT SPECIFICATION	2
2	SYSTEM STUDY	3
2.1	INTRODUCTION	4
2.2	EXISTING SYSTEM	5
2.3	DRAWBACKS OF EXISTING SYSTEM	5
2.4	PROPOSED SYSTEM	6
2.5	ADVANTAGES OF PROPOSED SYSTEM	6
3	REQUIREMENT ANALYSIS	8
3.1	FEASIBILITY STUDY	9
3.1.1	ECONOMICAL FEASIBILITY	9
3.1.2	TECHNICAL FEASIBILITY	9
3.1.3	BEHAVIORAL FEASIBILITY	10
3.1.4	FEASIBILITY STUDY QUESTIONNAIRE	10
3.2	SYSTEM SPECIFICATION	12
3.2.1	HARDWARE SPECIFICATION	12
3.2.2	SOFTWARE SPECIFICATION	12
3.3	SOFTWARE DESCRIPTION	12
3.3.1	DJANGO	13
3.3.2	SQLITE	13
4	SYSTEM DESIGN	14
4.1	INTRODUCTION	15
4.2	UML DIAGRAM	15
4.2.1	USE CASE DIAGRAM	16
4.2.2	SEQUENCE DIAGRAM	18
4.2.3	STATE CHART DIAGRAM	20
4.2.4	ACTIVITY DIAGRAM	21
4.2.5	CLASS DIAGRAM	23
4.2.6	OBJECT DIAGRAM	25
4.2.7	COMPONENT DIAGRAM	26

4.2.8	DEPLOYMENT DIAGRAM	27
4.3	USER INTERFACE DESIGN USING FIGMA	28
4.4	DATABASE DESIGN	31
5	SYSTEM TESTING	39
5.1	INTRODUCTION	40
5.2	TEST PLAN	40
5.2.1	UNIT TESTING	41
5.2.2	INTEGRATION TESTING	41
5.2.3	VALIDATION TESTING	42
5.2.4	USER ACCEPTANCE TESTING	42
5.2.5	AUTOMATION TESTING	42
5.2.6	SELENIUM TESTING	42
6	IMPLEMENTATION	50
6.1	INTRODUCTION	51
6.2	IMPLEMENTATION PROCEDURE	51
6.2.1	USER TRAINING	52
6.2.2	TRAINING ON APPLICATION SOFTWARE	52
6.2.3	SYSTEM MAINTENANCE	52
7	CONCLUSION & FUTURE SCOPE	53
7.1	CONCLUSION	54
7.2	FUTURE SCOPE	54
8	BIBLIOGRAPHY	56
9	APPENDIX	58
9.1	SAMPLE CODE	59
9.2	SCREEN SHOTS	66

List of Abbreviation

IDE - Integrated Development Environment

HTML - Hyper Text Markup Language.

CSS - Cascading Style Sheet

SQL - Structured Query Language

UML - Unified Modelling Language

JS - JavaScript

AJAX - Asynchronous JavaScript and XML Environment

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

GustoGrove is an ambitious project aimed at redefining the online spice shopping experience. This innovative e-commerce platform seeks to create a dynamic and user-centric marketplace for spice enthusiasts. With a robust focus on seamless database integration, real-time transactions, and interactivity, GustoGrove is designed to cater to both customers and administrators effectively. Administrators have the power to add, delete, and edit apparel within the system, among other functions, ensuring inventory management is a breeze. Users are provided with secure registration and profile management tools, enabling a personalized shopping experience. The platform offers a wide selection of spices for users to explore, with the added convenience of a wish cart for secure and efficient checkouts.

This user-friendly system not only enhances convenience but also fosters business growth. Users can effortlessly track their orders and take advantage of interactive web interfaces for product searches, detailed product descriptions, and the creation of wish lists to accumulate a multitude of items before secure checkout. Additionally, the system supports real-time electronic transactions, making sales and purchases more efficient. The automation of report generation eliminates the need for manual labor, allowing for frequent analysis of user and product data to facilitate informed decision-making. In essence, GustoGrove aims to create a revolutionary online spice marketplace, seamlessly integrating database-driven operations to serve a growing community of spice enthusiasts.

1.2 PROJECT SPECIFICATION

A cutting-edge e-commerce platform called GustoGrove is intended to transform the experience of buying spices online. Customers may buy spices that are customized to their interests using this complete system, which provides a 24/7 shopping experience. The main goal of this application is to make it easier for users to effortlessly buy their favorite spice goods by enabling online virtual shopping. The project follows a strong database integration strategy, and all sales and buy operations are conducted electronically, interactively, and in real-time.

The system includes 3 modules. They are:

1. Customer Module

Customers play a central role in the application, interacting with the platform by browsing the product catalog and placing orders based on their preferences. Searching options empower customers to tailor their spice selections to meet their specific needs. Individual

user interfaces allow customers to track order history. User profiles with editing capabilities are provided, enabling customers to manage their personal information and preferences effectively. The system also offers a shopping cart and wishlist functionality for customers, enabling them to add products to their cart for immediate purchase or to their wishlist for consideration.

2. Seller Module

Sellers hold authority over product management and have the ability to add, update, and maintain spice products within the catalog. They are equipped to perform thorough quality checks on spice products to ensure they meet the highest standards. Sellers also have access to a powerful search and filter feature that enables them to swiftly locate and manage their spice products. Additionally, sellers have access to detailed statistics, displaying the number of orders per product and the number of unique customers who have purchased their products. A notification feature alerts sellers when their product stock falls below a specified threshold, such as when stock is less than 5 units.

3. Admin Module

The system is under the general authority of administrators, who also have a unique login. Administrators have control over adding, managing, and creating new user roles as well as user permissions. The admin module offers improved administrative control, including user registration, customer management, and responsibilities like seller. By creating and allocating login credentials for moderators and designers, administrators prevent the formation of unauthorized login credentials and guarantee that only authorized users have access to the system.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

The 'GustoGrove - Comprehensive Spice E-commerce Management System' is designed to offer a cutting-edge web-based application that makes it easier to browse, investigate, and choose spice items. This system is carefully constructed to improve the user's capacity for effective search and discovery of spices, elevating the experience of online spice purchasing.

2.2 EXISTING SYSTEM

The current system requires all spice transactions to be done manually, which takes time. Users must physically visit spice stores or rely on suggestions from friends and family when choosing spices. Users are unable to track or buy spices online, and there is no online interactivity. It is vital to interact with intermediaries in real-world stores. Through the provision of an online, user-friendly platform, the planned GustoGrove system seeks to change this procedure. Without the need for in-person visits, users can easily browse, buy, pay for, and track their spice purchases, improving the whole spice purchasing experience. GustoGrove enables current online ease while bridging the gap between conventional offline spice purchasing and it.

2.2.1 NATURAL SYSTEM STUDIED

In the traditional way of spice shopping, people buy spices by physically going to stores, asking for recommendations from others, and interacting directly with store staff. There is no online option to see product details, make online payments, or track orders. GustoGrove plans to change this by providing an easy online platform where customers can browse, buy, and track spice products, making spice shopping more convenient.

2.3 DRAWBACKS OF EXISTING SYSTEM

- **Non-Interactive Environment:** The foremost issue with the existing system is its non-interactive nature. It lacks the features that allow users to interact with the spice marketplace in a dynamic and user-friendly manner.
- **Traditional User Interfaces:** The use of traditional user interfaces in the current system results in continuous post-backs to the server for each user action. This process involves making a call to the server, receiving a response, and refreshing the entire web form to display the result. This approach introduces delays in displaying information, which can be frustrating for users.

- **Limited Product Descriptions:** Product descriptions in the existing system are often limited, making it challenging for users to identify and select the desired spice products accurately. Users may lack comprehensive information about the spices available.
- **Geographical Limitations:** The existing system limits users to purchasing spices within the physical reach of their local shops. This restriction can be problematic for users who seek spices not readily available in their immediate vicinity.
- **Dependency on Shop Assistants:** Users often have to depend on shop assistants in the existing system. They may need to wait in line or for the shop assistant's availability to obtain assistance with their spice purchases, leading to inconvenience and potential delays.

2.4 PROPOSED SYSTEM

The proposed system for GustoGrove is meticulously designed to address the limitations of the existing system, creating an environment that is more user-friendly, visually appealing, and conducive to business growth, while prioritizing customer convenience. The primary objective of the proposed system is to develop an interface that facilitates a hassle-free interaction for customers, enabling them to track their spice orders and related updates efficiently. In the proposed system, users initiate their interaction by selecting spice products from the catalog and can even customize their choices based on available options. Each user gains access to a personal interface that allows them to monitor the status of their spice orders. To enhance user convenience, the system offers an easy-to-navigate web interface where users can search for spice products, access comprehensive product descriptions, and add products to their wish list. This empowers users to explore a multitude of spice options and securely checkout from their shopping cart. The system significantly reduces manual labor by enabling the swift generation of reports and the frequent analysis of user and product data, which, in turn, aids in informed decision-making.

2.5 ADVANTAGES OF PROPOSED SYSTEM

The proposed system, GustoGrove, offers numerous advantages over the existing spice shopping system, addressing the limitations and introducing innovative features:

- **Interactive and Attractive Web Interface:** GustoGrove is an AJAX-enabled website with interactive and attractive web pages. The integration of AJAX controls eliminates the need for annoying post-backs, enhancing the user experience. Users can seamlessly browse and

explore spice products with interactive features, view complete product specifications, multiple images, and read customer reviews, making spice selection more informed and enjoyable.

- **Wishlist and Convenient Checkout:** GustoGrove incorporates a wishlist feature that allows users to easily add or remove products from their shopping cart by dragging items to or from the wishlist. During the checkout process, users are provided with multiple shipping and billing options, adding convenience and flexibility to the purchasing experience.
- **Efficient and Comprehensive Search:** By making the application AJAX-enabled, GustoGrove eliminates unnecessary delays in the shopping process. Users can perform exhaustive searches with ease. The search engine lists a range of spice products based on search terms, and users can further refine their search results using various parameters, ensuring they find the exact spices they desire.
- **Enhanced Product Information:** The proposed system offers detailed product information, including specifications, images, and customer reviews. This not only provides users with a comprehensive understanding of the spices but also empowers them to make well-informed purchasing decisions.
- **Convenient Filtering Options:** Users can easily filter product lists based on their specific requirements and preferences, streamlining the search and selection process. This feature ensures that users can quickly find the spices that meet their unique needs.
- **Faster and More User-Centric:** With AJAX integration, GustoGrove significantly speeds up the user's interaction with the system, eliminating unnecessary delays and enhancing the overall user experience. This approach allows for a more efficient and enjoyable spice shopping journey.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

The viability of carrying out a project effectively is referred to as feasibility. It is evaluated using a feasibility study, which ascertains whether the suggested approach to fulfilling requirements is practical and doable in the context of software. This analysis takes into account variables such resource accessibility, software development costs, possible post-completion advantages to the firm, and ongoing maintenance costs. A report that suggests whether to move further with the requirements engineering and system development process is the result of the feasibility study.

3.1.1 Economical Feasibility

Economic feasibility is one of the most important aspects in the planning of a SHP- project. Even though it necessitates significant recurring expenditure, a SHP project generates income throughout the course of its existence. Therefore, conducting economic and financial analysis is quite important.

The e-commerce spices store is financially feasible by offering high-quality spices at competitive prices, attracting a broad customer base, and managing costs efficiently. Strategic marketing, secure payment options, and streamlined shipping contribute to profitability and customer satisfaction, ensuring long-term growth and success. Some queries are raised during the investigation includes the following:

- The costs conduct a full system investigation?
 - The proposed system is developed as part of project work, there is no manual cost to spend for the proposed system.
- The cost of the hardware and software?
 - All the resources are already available.

3.1.2 Technical Feasibility

Technical feasibility refers to the assessment of whether a proposed project, product, or solution can be successfully implemented using the available technology and resources. It involves evaluating whether the technology required for the project is accessible, reliable, and capable of fulfilling the project's objectives.

The technical feasibility of an e-commerce spices store involves evaluating the practicality of creating a user-friendly website with efficient search and checkout features. Security, scalability, and mobile responsiveness are crucial for a seamless shopping experience. Integration with

reliable payment gateways and inventory management ensures smooth transactions.

- Is the project feasible within the limits of current technology.
 - Yes
- Technical issues raised during the investigation are:
 - Nothing
- Can the technology be easily applied to current problems?
 - Yes
- Does the technology have the capacity to handle the solution?
 - Yes

3.1.3 Behavioral Feasibility

Operational feasibility is the determination of whether a proposed project or system can be successfully incorporated into the workflow, processes, and business operations already in place. It looks at whether the organization is able and willing to accept and use the new system without encountering too much disruption or resistance.

The e-commerce spices store is operationally feasible if it effectively addresses user needs and business concerns related to selling spices online. It must provide a seamless and efficient shopping experience for customers, allowing them to find and purchase spices easily. The system should be user-friendly and intuitive, requiring minimal training for users to adapt to the platform. Any anticipated difficulties in meeting user requirements should be identified and resolved appropriately.

The proposed system includes the following questions:

- Is there sufficient support for the users?
 - Yes
- Will the proposed system cause harm?
 - No

3.1.4 Feasibility Study Questionnaire

1. Can you provide an overview of your physical spice store?

The functioning of a physical spice store involves several key aspects that contribute to its successful operation and customer satisfaction. The spice store procures spices from various suppliers and distributors. The sourcing process involves selecting high-quality spices from different regions and ensuring they meet the store's quality standards.

2. Who is your target audience for the online store?

Our target audience for the online store includes home cooks, culinary enthusiasts, and individuals who appreciate the use of authentic and exotic spices in their cooking.

3. Which spices are the best-sellers in your physical store?

Our best-selling spices are cardamom, cinnamon, and pepper. Customers often inquire about these spices.

4. Have you noticed any specific customer preferences or trends in spice purchases?

Yes, we have observed an increasing demand for organic and ethically sourced spices.

5. Do you have more local customers visiting the physical store, or do you receive online orders from customers outside your local area?

Currently, our physical store attracts more local customers, but we believe the online store will help us reach customers from different regions

6. Do you have local suppliers for certain spices, and do you plan to highlight their sourcing on the online store?

Yes, we source some spices locally, and we would like to showcase this information to customers.

7. What are the main challenges you face in running the physical store, and how do you plan to address them in the online store?

One challenge is limited store hours. The online store will provide 24/7 accessibility, resolving this issue.

8. What is the typical or average amount that customers tend to spend on their purchases at your spice store?

However, the amount can vary based on the specific spices they choose, packaging options, and any ongoing promotions or discounts we offer. Some customers prefer to buy small quantities of specific spices, while others may purchase larger quantities or spice blends, leading to variations in the average spend. We regularly monitor customer purchase patterns to understand their preferences and tailor our offerings accordingly.

9. How is the billing process conducted at your spice store?

The billing process involves scanning or manually entering the selected spices' details and quantities into the system. The system calculates the total cost, including taxes, and generates a receipt for the customer. Various payment methods are accepted for completing the transaction.

10. How do you handle day-to-day changes in spice prices at your store?

To manage day-to-day price fluctuations, we regularly monitor spice prices from our suppliers and the market. If there are any changes, we update the prices in our system and ensure that our staff is informed. This helps us maintain accurate pricing for our customers and adapt to market dynamics effectively.

3.1 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor - Intel core i3

RAM - 4 G B

Hard disk - 1 T B

3.2.2 Software Specification

Front End - HTML, CSS

Back End - SQLite

Client on PC - Windows 7 and above.

Technologies used - DJANGO, JS, HTML5, AJAX, JQuery, PHP, CSS

3.3 SOFTWARE DESCRIPTION

3.3.1 HTML

HTML (Hypertext Markup Language) is the most basic building block of the Web. HTML is a fairly simple language made up of elements, which can be applied to pieces of text to give them different meaning in a document, structure a document into logical sections, and embed

content such as images and videos into a page. In the proposed system, HTML 5 is used to carry out the design part of the webpage. HTML validations are also used.

3.3.2 CSS

While HTML is used to define the structure and semantics of your content, CSS is used to style it and lay it out. For example, you can use CSS to alter the font, colour, size, and spacing of your content, split it into multiple columns, or add animations and other decorative features. In the proposed system CSS 3 is used. Animation using CSS is also used to give the website a unique look & feel.

3.3.3 JQuery

JQuery is a lightweight, "write less, do more", JavaScript library. The purpose of jQuery is to make it much easier to use JavaScript on your website. jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code. jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation. The jQuery library contains the following features: HTML/DOM manipulation, CSS manipulation, HTML event methods, Effects and animations, AJAX.

3.3.4 SQLite

SQLite is a widely used open-source, self-contained, serverless, and zero-configuration relational database management system (RDBMS). Unlike traditional RDBMS systems such as MySQL, PostgreSQL, or Oracle, SQLite is a self-contained library that provides a lightweight and efficient means of managing relational databases directly within applications.

3.3.5 JavaScript

JavaScript was initially created to “make web pages alive”. The programs in this language are called scripts. They can be written right in a web page’s HTML and run automatically as the page loads. In the proposed system loading screen, different popup animations are implemented using Java Script.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

Planning is the first step in the development phase of any proposed product system. Planning is creative process. Good planning is the key to an effective system. The term "design" is defined as follows, "A process where different techniques and principles are applied to define the process or a system detailed enough to be physically implemented". It can be defined as a process the application of various techniques and principles to define a device, process or process a detailed enough system to physically implement it. Software design is technical the core of software engineering and is applied independently of development paradigm used. Architectural details required for construction are developed during system planning system or product. As with any system approach, this software has gone through the best possible design phase that refines all levels of efficiency, effectiveness and precision. The design phase is the transition from a user-oriented document to a developer's document or database staff. System design goes through two development phases: Logical and Physical design.

4.2 UML DIAGRAM

UML is a standard language used to define, visualize, design and document artifacts of software systems. UML was created by the Object Management Group (OMG) and the draft UML 1.0 specification was submitted to OMG in January 1997. UML stands for Unified Modeling Language. UML is different from other common ones programming languages such as C, Java, COBOL, etc. UML is a visual language that is used make program drawings. UML can be described as a general-purpose visual modeling language visualize, define, build and document a software system. Although UML is usually used to model software systems is not limited by this limit. It is also used for modeling also non-software systems. For example, process flow in a production unit, etc. UML is not a programming language, but the tools can be used to generate code in different languages using UML diagrams. UML has a direct relationship with object analysis and design. After some standards, UML has become an OMG standard. All elements, relationships are is used to create a complete UML diagram and the diagram represents the system. Visual effect of the UML diagram is the most important part of the whole process. All other elements are used to make it perfect.

UML includes the following nine diagrams:

- Class diagram
- Object diagram
- Use case diagram
- Sequence diagram
- Collaboration diagram
- Activity diagram
- State Chart diagram
- Deployment diagram
- Component diagram

4.2.1 USE CASE DIAGRAM

A use case diagram is a graphical representation of the interactions between elements of a system. A use case is a method used in systems analysis to identify, explain and organize a system standard. In this context, the term "system" refers to something that is developed or used, for example, a mail order product sales and service website. Use case diagrams are used UML (Unified Modeling Language), a standard notation for modeling real-world objects and systems. System objectives may be general post-planning, hardware design validation, testing and debugging the development product, creating online help, or perform a task aimed at consumer service. For example, use cases in product sales the environment includes product ordering, catalog updates, payment processing and the customer relationships. A use case diagram consists of four parts:

- **Boundary:** The boundary in a use case diagram defines the system's scope and its interaction with the surrounding environment. It helps distinguish the system from the outside world.
- **Actors:** Actors represent the individuals, roles, or entities that interact with the system. These actors are categorized based on their roles in the system.
- **Use Cases:** Use cases represent specific functionalities or tasks that actors perform within the system. These use cases describe the interactions between actors and the system.
- **Relationships:** Relationships and dependencies between actors and use cases are depicted in the diagram. They show how actors interact with specific functionalities and tasks within the system.

Use case diagrams are drawn to capture the functional requirements of a system.

After identifying the above items, we have to use the following guidelines to draw an

efficient use case diagram.

- The name of a use case is very important. The name should be chosen in such a way so that it can identify the functionalities performed.
- Give a suitable name for actors.
- Show relationships and dependencies clearly in the diagram.
- Do not try to include all types of relationships, as the main purpose of the diagram is to identify the requirements.
- Use notes whenever required to clarify some important points.

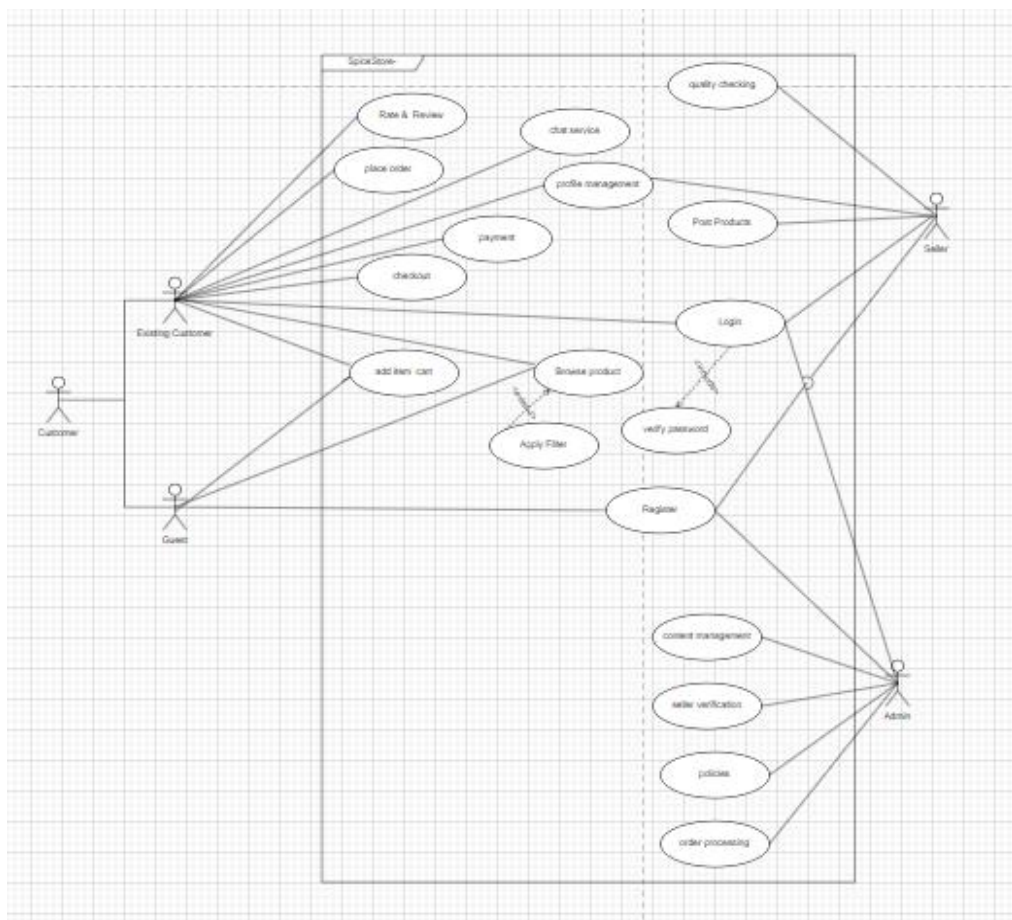


Fig.1

4.2.2 SEQUENCE DIAGRAM

A sequence diagram simply describes how objects interact with each other in a sequential order, ie the order in which these interactions occur. We can also use the terms event diagrams or event scenarios refer to the sequence scheme. Sequence diagrams describe how and in what order system functions objects. These charts are widely used by traders and software developers document and understand the requirements of new and existing systems.

Sequence Diagram Notations –

- i. **Actors** – An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the UML diagram. We use actors to depict various roles including human users and other external subjects. We represent an actor in a UML diagram using a stick person notation. We can have multiple actors in a sequence diagram.
- ii. **Lifelines** – A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram.
- iii. **Messages** – Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline. We represent messages using arrows. Lifelines and messages form the core of a sequence diagram.

Messages can be broadly classified into the following categories:

Synchronous messages

Asynchronous Messages

Create message

Delete Message

Self-Message

Reply Message

Found Message

Lost Message

- iv. Guards – To model conditions we use guards in UML. They are used when we need to restrict the flow of messages on the pretext of a condition being met. Guards play an important role in letting software developers know the constraints attached to a system or a particular process.

Uses of sequence diagrams –

- Used to model and visualize the logic behind a sophisticated function, operation or procedure.
- They are also used to show details of UML use case diagrams.
- Used to understand the detailed functionality of current or future systems.
- Visualize how messages and tasks move between objects or components in a system

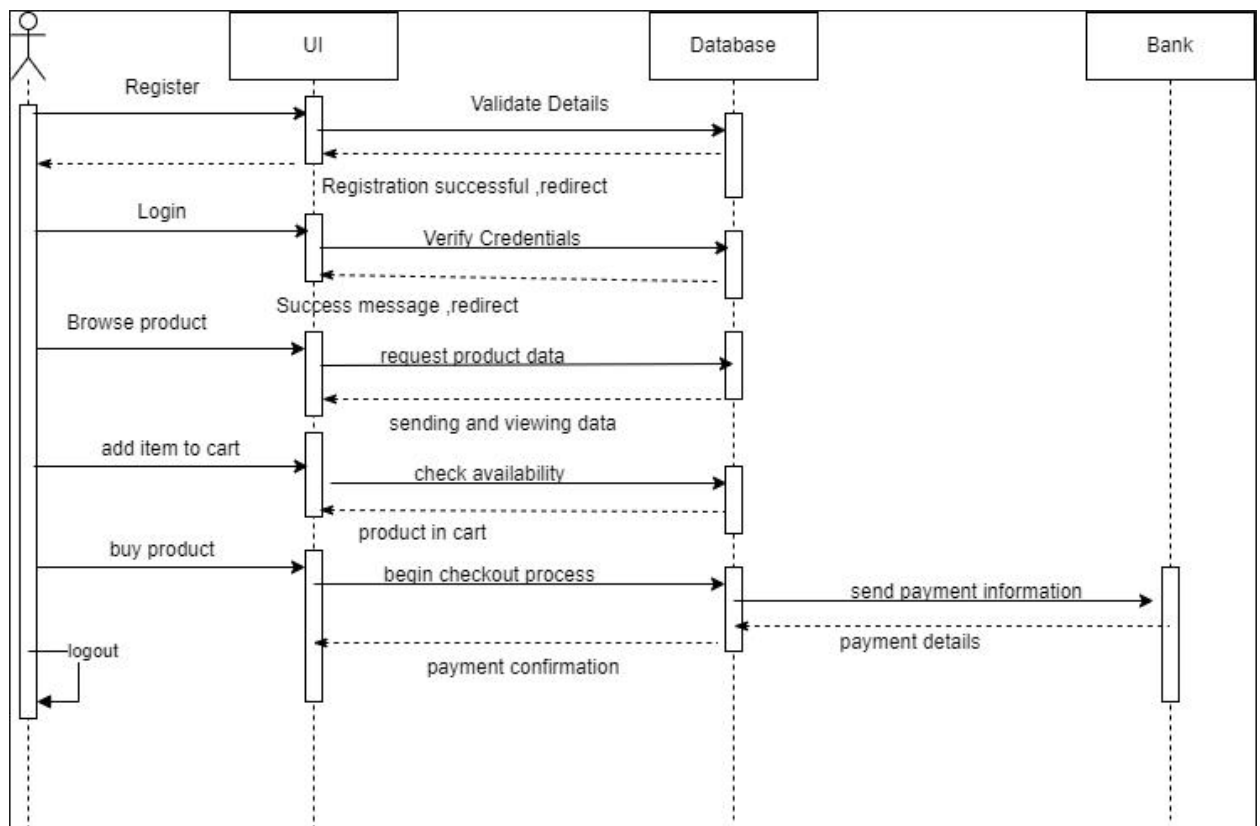


Fig.2

4.2.3 State Chart Diagram

The state machine diagram is also called the Statechart or State Transition diagram, which shows the order of states underwent by an object within the system. It models the behavior of a class, a subsystem, a package, and a complete system. It tends out to be an efficient way of modeling the interactions and collaborations in the external entities and the system. It models event-based systems to handle the state of an object. It also defines several distinct states of a component within the system. Each object/component has a specific state.

Notation of a State Machine Diagram

Following are the notations of a state machine diagram enlisted below:

- a) Initial state: It defines the initial state (beginning) of a system, and it is represented by a black filled circle.
- b) Final state: It represents the final state (end) of a system. It is denoted by a filled circle present within a circle.
- c) Decision box: It is of diamond shape that represents the decisions to be made on the basis of an evaluated guard.
- d) Transition: It is represented by an arrow labeled with an event due to which the change has ensued.
- e) State box: It depicts the conditions or circumstances of a particular object of a class at a specific point of time. A rectangle with round corners is used to represent the state box.

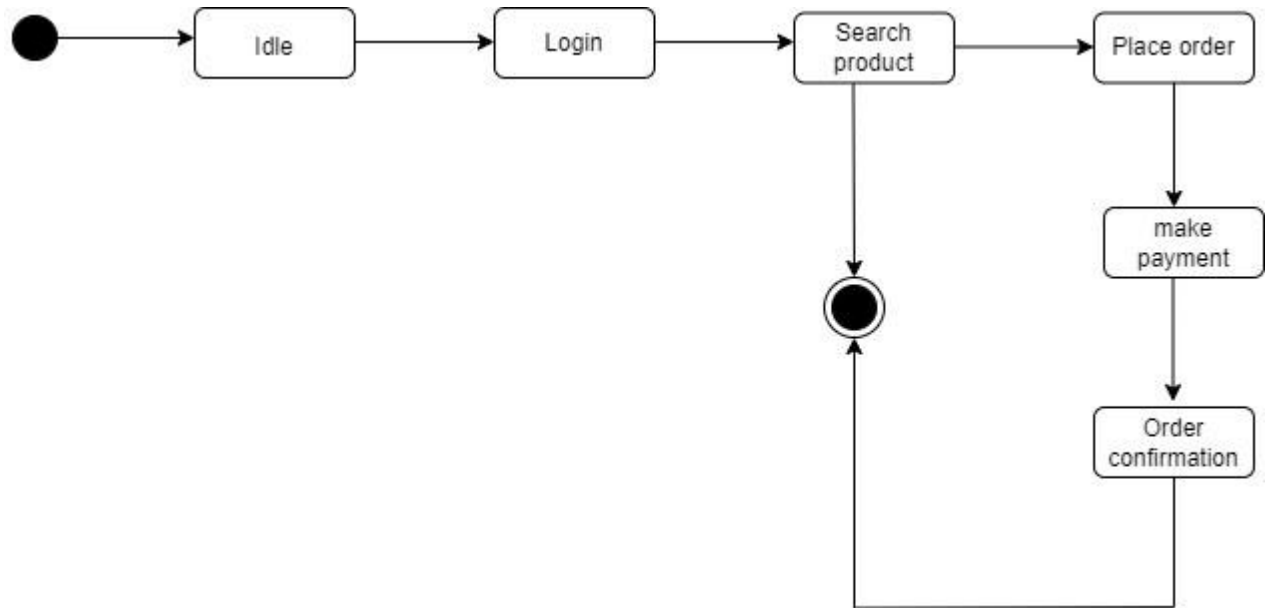


Fig.3

4.2.4 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.

Components of an Activity Diagram

Following are the component of an activity diagram:

- a. Activities:** The categorization of behavior into one or more actions is termed as an activity. In other words, it can be said that an activity is a network of nodes that are connected by edges. The edges depict the flow of execution. It may contain action nodes, control nodes, or object nodes. The control flow of activity is represented by control nodes and object nodes that illustrates the objects used within an activity. The activities are initiated at the initial node and are terminated at the final node.
- b. Activity partition /swimlane:** The swimlane is used to cluster all the related activities in one column or one row. It can be either vertical or horizontal. It used to add modularity to the

activity diagram. It is not necessary to incorporate swimlane in the activity diagram. But it is used to add more transparency to the activity diagram.

c. Forks: Forks and join nodes generate the concurrent flow inside the activity. A fork node consists of one inward edge and several outward edges. It is the same as that of various decision parameters. Whenever a data is received at an inward edge, it gets copied and split crossways various outward edges. It split a single inward flow into multiple parallel flows.

d. Join Nodes: Join nodes are the opposite of fork nodes. A Logical AND operation is performed on all of the inward edges as it synchronizes the flow of input across one single output (outward) edge.

Notation of an Activity diagram

Activity diagram constitutes following notations:

1. Initial State: It depicts the initial stage or beginning of the set of actions.
2. Final State: It is the stage where all the control flows and object flows end.
3. Decision Box: It makes sure that the control flow will follow only one path.
4. Action Box: It represents the set of actions that are to be performed

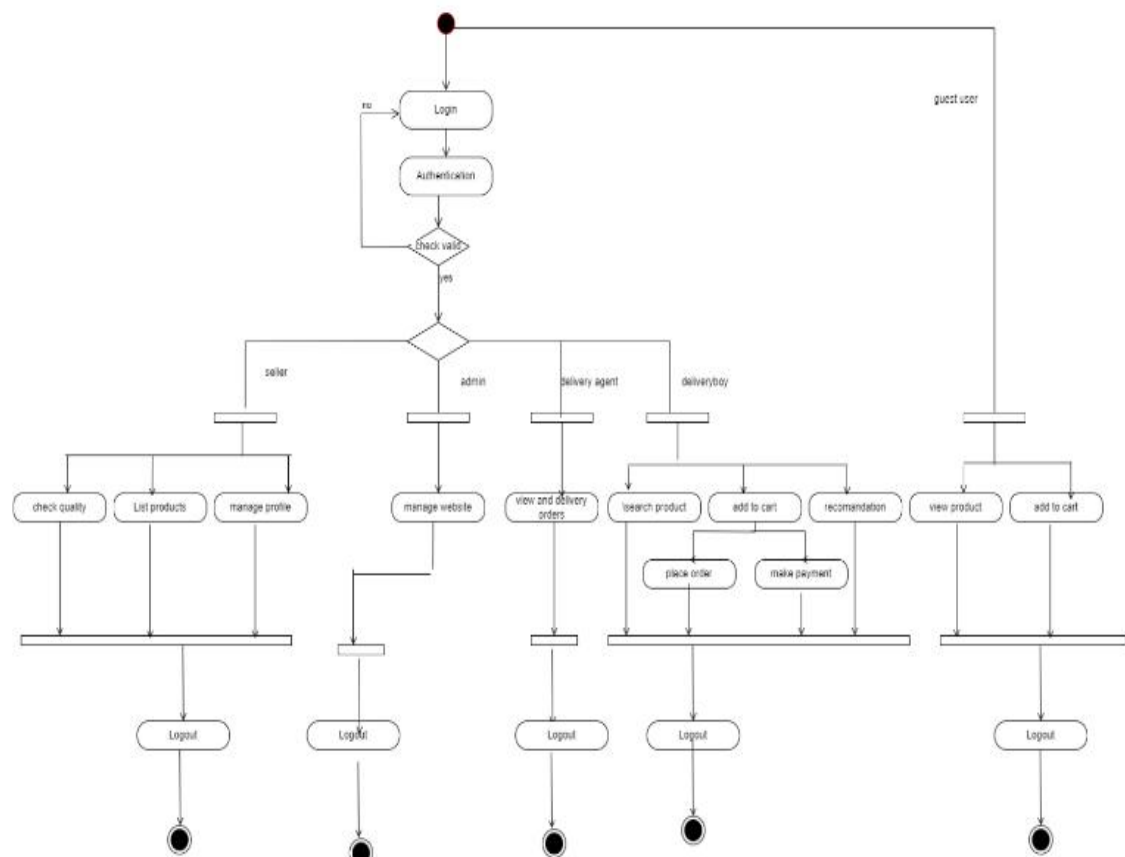


Fig. 4

4.2.5 Class Diagram

The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them. A class consists of its objects, and also it may inherit from other classes. A class diagram is used to visualize, describe, document various different aspects of the system, and also construct executable software code. It shows the attributes, classes, functions, and relationships to give an overview of the software system. It constitutes class names, attributes, and functions in a separate compartment that helps in software development. Since it is a collection of classes, interfaces, associations, collaborations, and constraints, it is termed as a structural diagram.

Components of a Class Diagram

The class diagram is made up of three sections:

Upper Section: The upper section encompasses the name of the class. A class is a representation of similar objects that shares the same relationships, attributes, operations, and semantics. Some of the following rules that should be taken into account while representing a class are given below:

- a. Capitalize the initial letter of the class name.
- b. Place the class name in the center of the upper section.
- c. A class name must be written in bold format.
- d. The name of the abstract class should be written in italics format.

Middle Section: The middle section constitutes the attributes, which describe the quality of class.

The attributes have the following characteristics:

- The attributes are written along with its visibility factors, which are public (+), private (-), protected (#), and package (~).
- The accessibility of an attribute class is illustrated by the visibility factors.
- A meaningful name should be assigned to the attribute, which will explain its usage inside the class.

Lower Section: The lower section contain methods or operations. The methods are represented in the form of a list, where each method is written in a single line. It demonstrates how a class interacts with data.

Relationships

In UML, relationships are of three types:

- **Dependency:** A dependency is a semantic relationship between two or more classes where a change in one class cause changes in another class. It forms a weaker relationship.
- **Generalization:** A generalization is a relationship between a parent class (superclass) and a child class (subclass). In this, the child class is inherited from the parent class.
- **Association:** It describes a static or physical connection between two or more objects. It depicts how many objects are there in the relationship.
- **Multiplicity:** It defines a specific range of allowable instances of attributes. In case if a range is not specified, one is considered as a default multiplicity.
- **Aggregation:** An aggregation is a subset of association, which represents has a relationship. It is more specific than association. It defines a part-whole or part-of relationship. In this kind of relationship, the child class can exist independently of its parent class.
- **Composition:** The composition is a subset of aggregation. It portrays the dependency between the parent and its child, which means if one part is deleted, then the other part also gets discarded. It represents a whole-part relationship.
- **Abstract Classes:** In the abstract class, no objects can be a direct entity of the abstract class. The abstract class can neither be declared nor be instantiated. It is used to find the functionalities across the classes. The notation of the abstract class is similar to that of class; the only difference is that the name of the class is written in italics. Since it does not involve any implementation for a given function, it is best to use the abstract class with multiple objects.

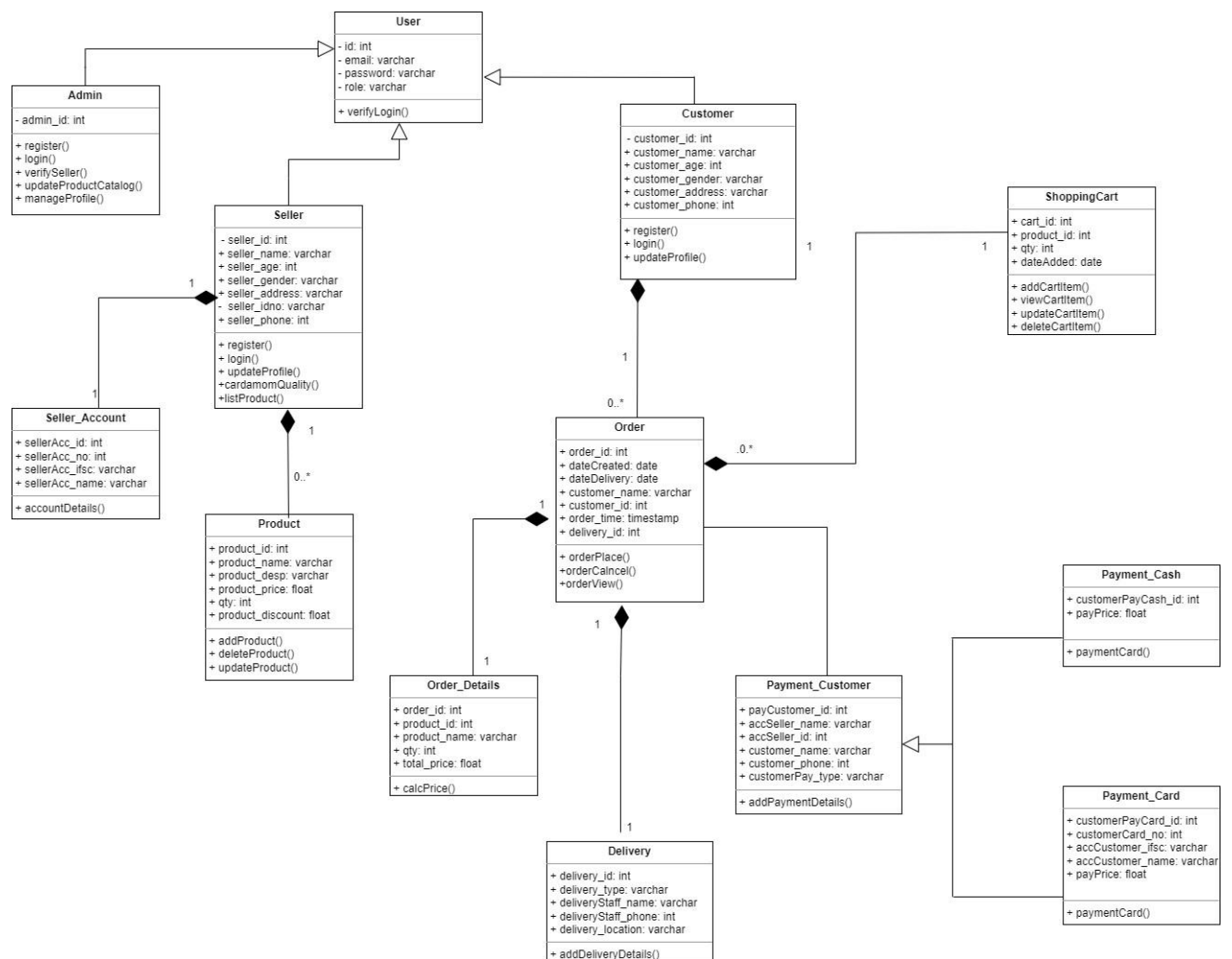


Fig.5

4.2.6 Object Diagram

Object diagrams are dependent on the class diagram as they are derived from the class diagram. It represents an instance of a class diagram. The objects help in portraying a static view of an object-oriented system at a specific instant. Both the object and class diagram are similar to some extent; the only difference is that the class diagram provides an abstract view of a system. It helps in visualizing a particular functionality of a system.

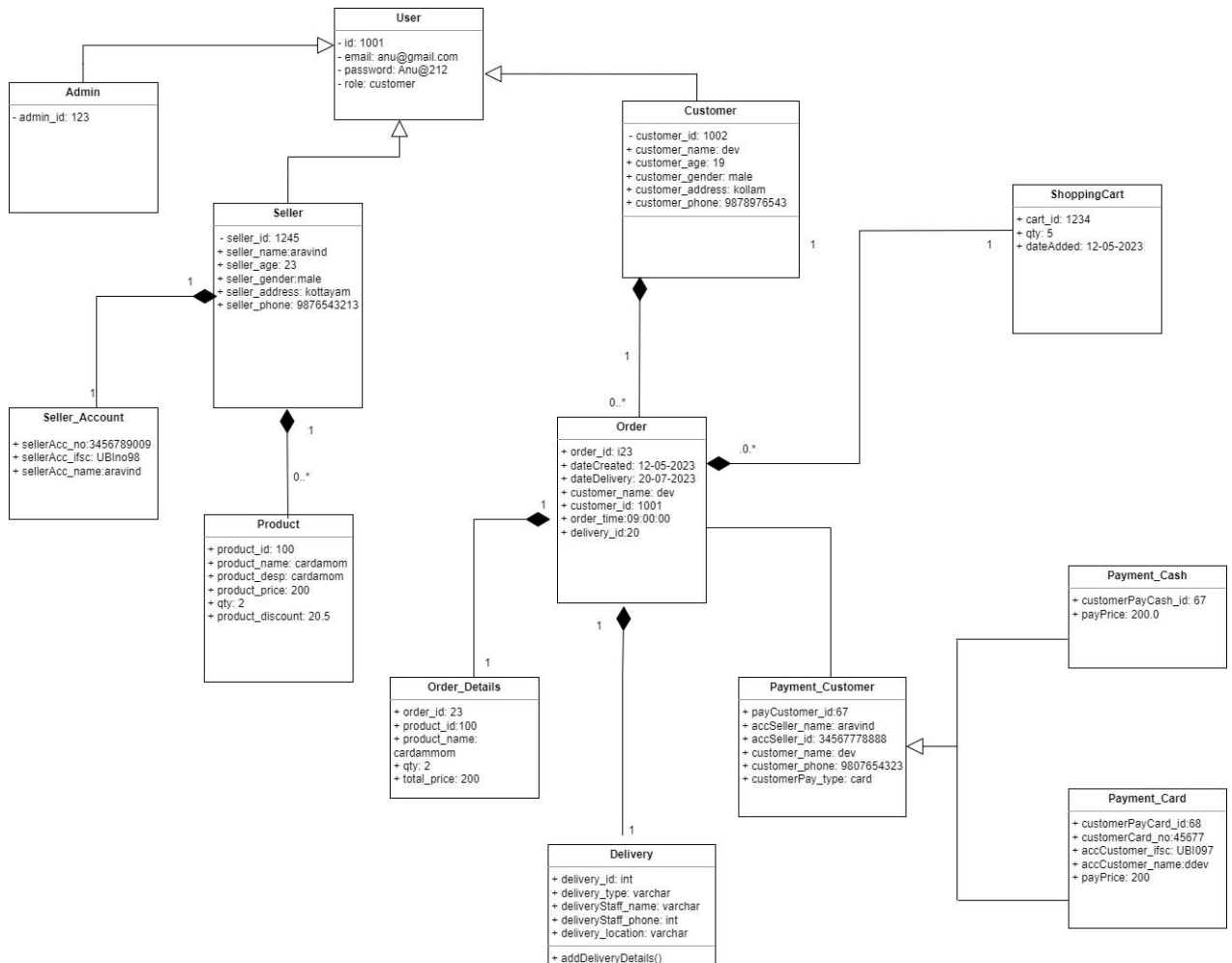


Fig .6

4.2.7 Component Diagram

A component diagram is used to break down a large object-oriented system into the smaller components, so as to make them more manageable. It models the physical view of a system such as executables, files, libraries, etc. that resides within the node. It visualizes the relationships as well as the organization between the components present in the system. It helps in forming an executable system. A component is a single unit of the system, which is replaceable and executable. The implementation details of a component are hidden, and it necessitates an interface to execute a function. It is like a black box whose behavior is explained by the provided and required interfaces.

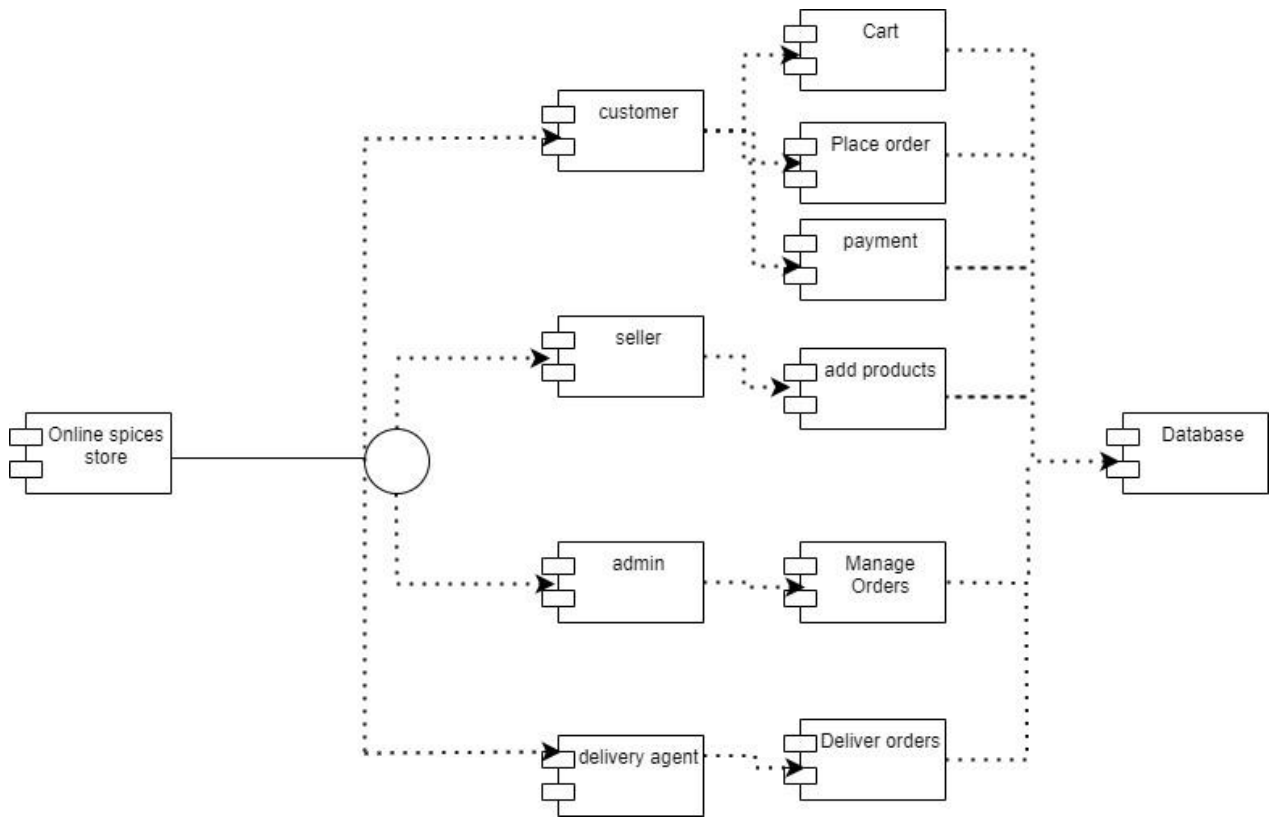


Fig. 7

4.2.8 Deployment Diagram

The deployment diagram visualizes the physical hardware on which the software will be deployed. It portrays the static deployment view of a system. It involves the nodes and their relationships. It ascertains how software is deployed on the hardware. It maps the software architecture created in design to the physical system architecture, where the software will be executed as a node. Since it involves many nodes, the relationship is shown by utilizing communication paths

Notation of Deployment diagram

The deployment diagram consists of the following notations:

1. A component
2. An artifact
3. An interface
4. A node

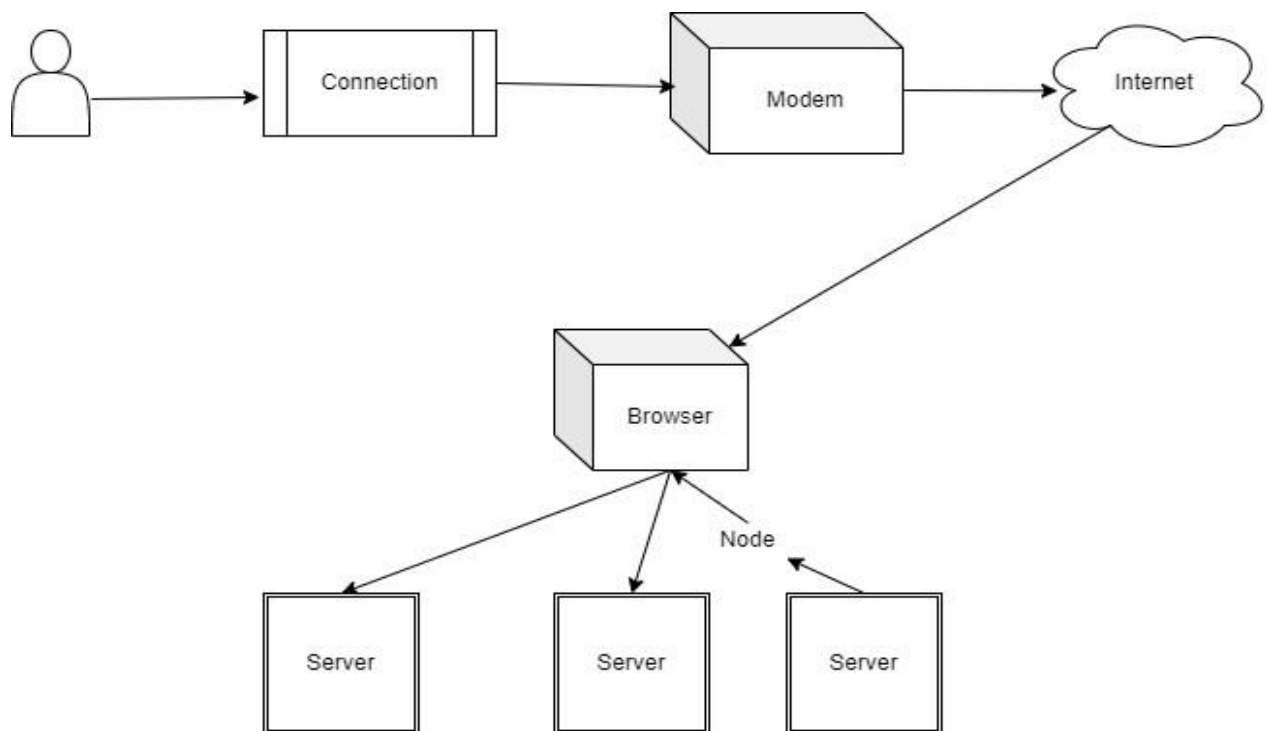


Fig. 7

4.3 USER INTERFACE DESIGN USING FIGMA

Form Name: Sign-In Form

Sign up

 Your email


 Password


[Forgot Password?](#)


Log in


Form Name: Sign Up form

Sign up

 Your Name

 Your Email

 Password

 Repeat your password

☐ I agree all statements in [Terms of service](#)

Register

Customer Profile page

Personal Details

First Name *

Enter your First Name

Last Name *

Enter your Last Name

Mobile number *

Enter your mobile no.

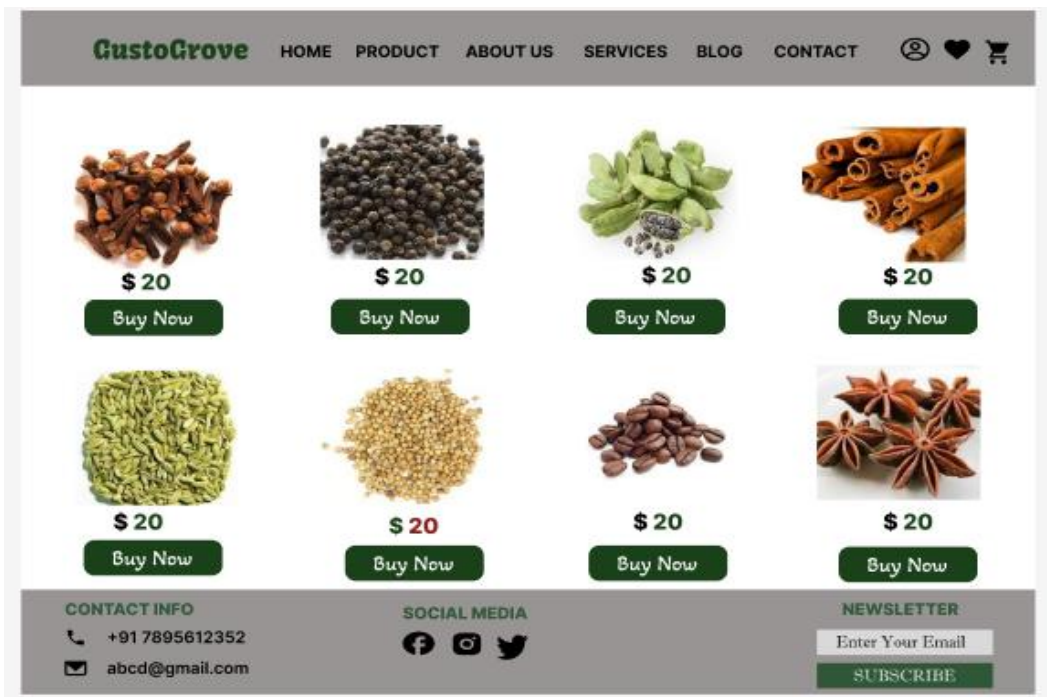
Email *

ria@gmail.com

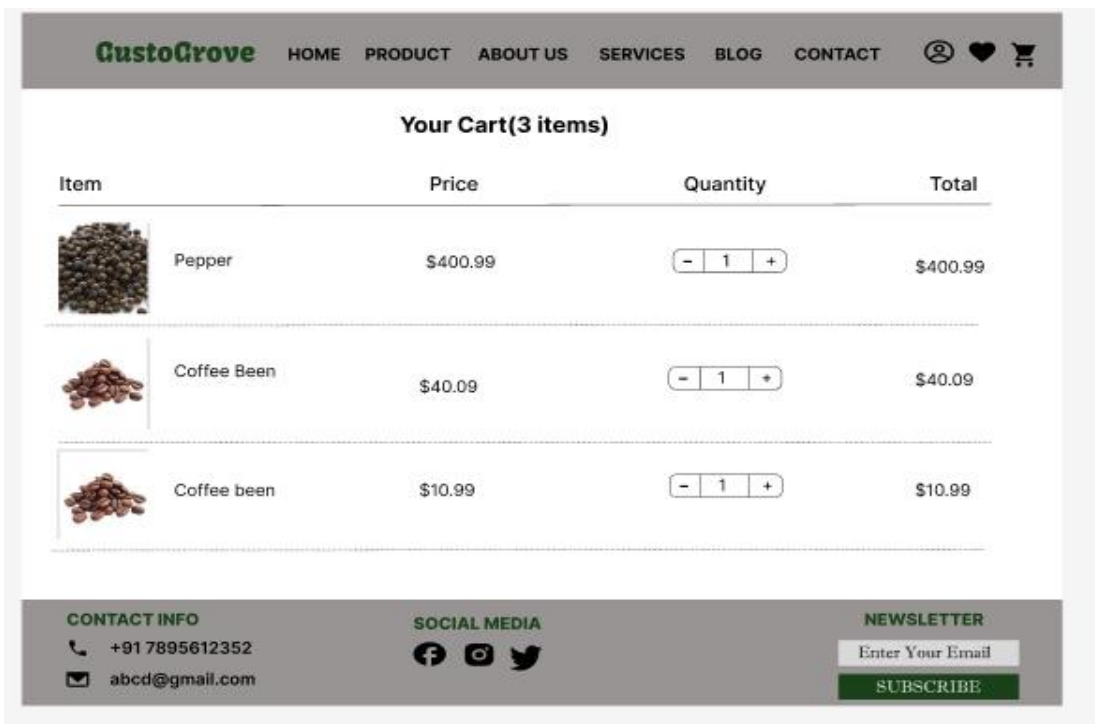
Save

Update

Shop page



Cart Page



4.4 DATABASE DESIGN

A database is a structured structure that stores information through which a user can access stored information efficiently and effectively. The information stored in a database is the purpose of a database and needs to be protected. Database design is a 2-step process. First, user needs are identified and a database that meets these needs is designed. This is known as Information Level Design, and it is done independently of any individual Database Management System (DBMS). Second, this Information Level Design is converted into a Design for the specific Database-as-a-Service (DAS) that will implement the system. This step, known as Physical Level Design, is concerned with the features of the specific Database Management System (DMS) that will be implemented. Database design runs in parallel to the system design.

4.4.1 Relational Database Management System (RDBMS)

A relational model represents the database as a collection of relations. Each relation resembles a table of values or file of records. In formal relational model terminology, a row is called a tuple, a column header is called an attribute and the table is called a relation. A relational database consists of a collection of tables, each of which is assigned a unique name. A row in a tale represents a set of related values.

Relations, Domains & Attributes

A table is a relation. The rows in a table are called tuples. A tuple is an ordered set of n elements. Columns are referred to as attributes. Relationships have been set between every table in the database. This ensures both Referential and Entity Relationship Integrity. A domain D is a set of atomic values. A common method of specifying a domain is to specify a data type from which the data values forming the domain are drawn. It is also useful to specify a name for the domain to help in interpreting its values. Every value in a relation is atomic, that is not decomposable.

- Key is used to build the associations between tables. Primary Key and Foreign Key are the two principal keys of utmost significance. With the use of these keys, relationships for entity integrity and referential integrity may be created.
- Entity Integrity forbids the use of null values for any Primary Key.
- It is required by referential integrity that no Primary Key have null values.
- Referential Integrity Each unique Foreign Key value must have a corresponding Primary Key

value in the same domain to maintain referential integrity. Super Key and Candidate Keys are additional keys.

4.4.2 Normalization

To minimize the impact of future changes on data structures, data are put together in the simplest possible way. Normalization is the formal method of arranging data structures in ways that reduce duplication and support integrity. The process of normalization involves dividing superfluous fields and dispersing a huge table into several smaller ones. Additionally, it is employed to prevent insertion, deletion, and update abnormalities. Two notions, keys and relationships, are used in the standard style of data modeling.

Each row in an a table is identified by a key. Keys come in two varieties: main keys and foreign keys. When identifying entries from the same table, a primary key is an element—or a set of elements—in the table. A column in a table that specifically identifies a record from another table is known as a foreign key. The third normal form has been applied to all tables' normalization. As the name suggests, it means returning things to their regular state. Through normalization, the application developer aims to create a data structure where names can be easily associated with the data by users, and where data is logically organized into appropriate tables and columns. By removing repeated groupings from data, normalization prevents data corruption.

- ✓ Normalize the data.
- ✓ Choose proper names for the tables and columns.
- ✓ Choose the proper name for the data.

First Normal Form

The First Normal Form states that the domain of an attribute must include only atomic values and that the value of any attribute in a tuple must be a single value from the domain of that attribute. In other words 1NF disallows “relations within relations” or “relations as attribute values within tuples”. The only attribute values permitted by 1NF are single atomic or indivisible values. The first step is to put the data into First Normal Form. This can be done by moving data into separate tables where the data is of similar type in each table. Each table is given a Primary Key or Foreign Key as per requirement of the project. In this we form new relations for each non-atomic attribute or nested relation. This eliminated repeating groups of data. A relation is said to be in first normal form if only if it satisfies the constraints that contain the primary key only.

Second Normal Form

According to Second Normal Form, for relations where primary key contains multiple attributes, no non-key attribute should be functionally dependent on a part of the primary key. In this we decompose and setup a new relation for each partial key with its dependent attributes. Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it. This step helps in taking out data that is only dependent on a part of the key. A relation is said to be in second normal form if and only if it satisfies all the first normal form conditions for the primary key and every non-primary key attributes of the relation is fully dependent on its primary key alone.

Third Normal Form

According to Third Normal Form, Relation should not have a non-key attribute functionally determined by another non-key attribute or by a set of non-key attributes. That is, there should be no transitive dependency on the primary key. In this we decompose and set up relation that includes the non-key attributes that functionally determines other non-key attributes. This step is taken to get rid of anything that does not depend entirely on the Primary Key. A relation is said to be in third normal form if only if it is in second normal form and more over the non key attributes of the relation should not be depend on other non-key attribute.

4.4.3 Sanitization

Data validation is an important part of web development, especially when working with forms the user first enters his personal information and then submits it to the database. Incorrect information was submitted format can cause DBMS security problems. Hackers often use SQL injection to inject malicious content SQL commands to the database. SQL injections can even destroy a database after it has been injected. Therefore, to ensure the security of the database the data entered by the user must be cleaned and filtered against hackers before sending Database. Data cleaning means the intentional, permanent removal or destruction of data a to the storage device so that it cannot be restored. Usually when data is deleted from memory the media is not actually deleted and can be recovered by an attacker device

4.4.4 Indexing

Indexing is used to quickly retrieve specific information from a database. There is a technique that uses data structures to optimize the lookup time of a database query. Indexing reduces their number drives needed to access certain data internally by creating an index table. The search key contains a copy of the primary key or candidate key of a database table. We usually keep The selected primary or candidate keys are sorted so that we can reduce the total number of queries time or search time. The index takes the search key as input and effectively returns the collection similar records. Indexing in a database is defined by its indexing attributes.

Different types of indexing methods are:

- Primary Indexing

Primary Index is an ordered file which is fixed length size with two fields. The first field is the same a primary key and second, filed is pointed to that specific data block. In the primary Index, there is always one to one relationship between the entries in the index table.

- Secondary Indexing

The secondary Index in DBMS can be generated by a field which has a unique value for each record, and it should be a candidate key. It is also known as a non-clustering index. This two-level database indexing technique is used to reduce the mapping size of the first level. For the first level, a large range of numbers is selected because of this; the mapping size always remains small.

- Multilevel Index

Multilevel Indexing in Database is created when a primary index does not fit in memory. In this type of indexing method, you can reduce the number of disk accesses to short any record and kept on a disk as a sequential file and create a sparse base on that file.

4.5 TABLE DESIGN

1.CustomUser

Primary key:Cust_ id

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Cust_id	INT	PK	Unique identifier for users.

2	email	VARCHAR(20)	NOT NULL	Email address of the user.
3	password	VARCHAR(20)	NOT NULL	User's password.
4	name	VARCHAR(20)	NOT NULL	Name of the user.
5	is_customer	BOOL		Indicates if the user is a customer.
6	is_seller	BOOL		Indicates if the user is a seller.
7	gstn	VARCHAR(15)		GSTN (Goods and Services Tax Number) field.

2.Customer_Profile

Primary key: Prof_id

Foreign Key: cust_id references table CustomUser,

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Prof_id	INT	PK	Unique identifier fo profiler .
2	Cust_id	INT	ForeignKey	Connects to the associated user.
3	first_name	VARCHAR(20)	NOT NULL	First name of the customer.
4	last_name	VARCHAR(20)	NOT NULL	First name of the customer
5	mobile_number	VARCHAR(20)	NOT NULL	Mobile number of the customer..

3.Product

Primary key: Pdt_id

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Pdt_id	INT	PK	Unique identifier fo products .
2	Product_name	VARCHAR(20)	NOTNULL	Name of the product.
3	description	VARCHAR(50)	NOT NULL	Description of the product.
4	image		NOT NULL	Image of the product
5	Quantity_value	INT	NOT NULL	Quantity value of the product.
6	Quantity_prefix	VARCHAR(5)	NOT NULL	Prefix for quantity (e.g., gms, kg).

7	stock	INT	DEFAULT-1	Current stock quantity of the product.
8	price	DECIMAL	NOT NULL	Price of the product
9	Date_added	DATE	NOTNULL	Date when the product was added.

3.Wishlist

Primary key: wishlist_id

Foreign Key: cust_id references table CustomUser,

Pdt_id references table Product,

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	wishlist_id	INT	PK	Unique identifier fo wishlist .
2	Cust_id	INT	ForeignKey	Connects to the associated user.
3	Pdt_id	INT	ForeignKey	Products in the user's wishlist.

4.Seller_Details

Primary key: seller-id

Foreign Key: cust_id references table CustomUser,

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	seller_id	INT	PK	Unique identifier fo wishlist .
2	Store_naame	VARCHAR(20)	NOT NULL	Name of the store
3	Phone-number	VARCHAR(12)	NOT NULL	Phone number of seller
4	pincode	VARCHAR(6)	NOT NULL	Pincode of the seller's location.
5	Pickup_address	VARCHAR(20)	NOT NULL	Pickup address for products
6	city	VARCHAR(20)	NOT NULL	City where the seller operates
7	state	VARCHAR(20)	NOT NULL	City where the seller operates
8	Account_number	VARCHAR(20)	NOT NULL	Account number for transactions.
9	bankname	VARCHAR(20)	NOT NULL	Account number for transactions.
10	Branch_name	VARCHAR(20)	NOT NULL	Branch of the bank for transactions.
11	Ifsc_code	VARCHAR(20)	NOT NULL	IFSC (Indian Financial System Code) of the bank.

12	Acc_name	VARCHAR(20)	NOT NULL	Name of the account holder.
13	Cust_id	INT	FK	Connects to the seller

5.Order

Primary key: order_id

Foreign Key: cust_id references table CustomUser,

Pdt_id references table Product,

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	order_id	INT	PK	Unique identifier fo wishlist .
2	Cust_id	INT	ForeignKey	Connects to the associated user.
3	Pdt_id	INT	ForeignKey	Products in the user's wishlist.
4	Total_price	DECIMAL	NOT NULL	Total price of the order.
5	Razor_pay_orderid	VARCHAR(20)	NOT NULL	Order ID provided by Razorpay.
6	Payment_status	VARCHAR(10)	NOT NULL	Payment status of the order.

5.CartItem

Primary key: cart_id

Foreign Key: cust_id references table CustomUser,

Pdt_id references table Product, order_id references table Order,

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	cart_id	INT	PK	Unique identifier fo cart .
2	Cust_id	INT	ForeignKey	Connects to the associated user.
3	Pdt_id	INT	ForeignKey	Products in the user's cart.
4	quantity	INT	NOT NULL	Quantity of the product in the cart..
5	Order_id	VARCHAR(20)	NOT NULL	Connects to the order if the item is in an order.
6	status	BOOL	NOT NULL	Indicates whether the item is active in the cart.

6.notification

Primary key: noti_id

cust_id references table CustomUser,

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	noti_id	INT	PK	Unique identifier fo notification .
2	message	VARCHAR(30)		Notification message.
3	Cust_id	INT	ForeignKey	Connects to the seller
4	read	BOOL	NOT NULL	Indicates if the notification has been read.
5	Created_at	Date	NOT NULL	Date and time when the notification was created.

CHAPTER 5

SYSTEM TESTING

5.1 INTRODUCTION

Software testing is the process of running software in a controlled manner to get an answer question - does the software work as intended? Software testing is often used together confirming and confirming the deadline. Validation is the checking or testing of items including ensure software compatibility and compliance with relevant specifications. Software testing is only one type of control, which also uses techniques such as audits, analyses, audits, and passages. Validation is the process of verifying that what is specified is what it is the user really wanted. Other activities often associated with software testing are static analysis and dynamic analysis. Static analysis examines software by looking for its source code problems and collecting metrics without running code. Dynamic analysis shows about the behavior of the software during its execution to provide information such as execution traces, temporal profiles and test scope data.

Testing is a set of activity that can be planned in advanced and conducted systematically. Testing begins at the module level and work towards the integration of entire computers-based system. Nothing is complete without testing, as its vital success of the system testing objectives, there are several rules that can serve as testing objectives. They are:

Testing runs a program to find a bug.

- A good test case is one that has a high probability of finding an undetected bug.
- A successful test is one that reveals an undetected bug.

It will become clear if the testing can be done according to the mentioned objectives software errors. Also, testing shows that the software feature appears to be working according to the specification, this performance requirement appears to be met.

5.2 TEST PLAN

A test plan is a document that outlines the required steps to complete various testing methodologies. It provides guidance on the activities that need to be performed during testing. Software developers create computer programs, documentation, and associated data structures. They are responsible for testing each component of the program to ensure it meets the intended purpose. To address issues with self-evaluation, an independent test group (ITG) is often established.

Testing objectives should be stated in quantifiable language, such as mean time to failure, cost to find and fix defects, remaining defect density or frequency of occurrence, and test work-hours per regression test.

The different levels of testing include:

- Unit testing
- Integration testing
- Data validation testing
- Output testing

5.2.1 Unit Testing

Unit testing is a software testing technique that focuses on verifying individual components or modules of the software design. The purpose of unit testing is to test the smallest unit of software design and ensure that it performs as intended. Unit testing is typically white-box focused, and multiple components can be tested simultaneously. The component-level design description is used as a guide during testing to identify critical control paths and potential faults within the module's perimeter.

During unit testing, the modular interface is tested to ensure that data enters and exits the software unit under test properly. The local data structure is inspected to ensure that data temporarily stored retains its integrity during each step of an algorithm's execution. Boundary conditions are tested to ensure that all statements in a module have been executed at least once, and all error handling paths are tested to ensure that the software can handle errors correctly.

Before any other testing can take place, it is essential to test data flow over a module interface. If data cannot enter and exit the system properly, all other tests are irrelevant. Another crucial duty during unit testing is the selective examination of execution pathways to anticipate potential errors and ensure that error handling paths are set up to reroute or halt work when an error occurs. Finally, boundary testing is conducted to ensure that the software operates correctly at its limits.

5.2.2 Integration Testing

Integration testing is a systematic approach that involves creating the program structure while simultaneously conducting tests to identify interface issues. The objective is to construct a program structure based on the design that uses unit-tested components. The entire program is then tested. Correcting errors in integration testing can be challenging due to the size of the overall program, which makes it difficult to isolate the causes of the errors. As soon as one set of errors is fixed, new ones may arise, and the process may continue in an apparently endless cycle. Once unit testing is complete for all modules in the system, they are integrated to check for any interface inconsistencies. Any discrepancies in program structures are resolved, and a unique program structure is developed.

5.2.3 Validation Testing or System Testing

The final stage of the testing process involves testing the entire software system as a whole, including all forms, code, modules, and class modules. This is commonly referred to as system testing or black box testing. The focus of black box testing is on testing the functional requirements of the software. A software engineer can use this approach to create input conditions that will fully test each program requirement. The main types of errors targeted by black box testing include incorrect or missing functions, interface errors, errors in data structure or external data access, performance errors, initialization errors, and termination errors.

5.2.4 Output Testing or User Acceptance Testing

User acceptance testing is performed to ensure that the system meets the business requirements and user needs. It is important to involve the end users during the development process to ensure that the software aligns with their needs and expectations. During user acceptance testing, the input and output screen designs are tested with different types of test data. The preparation of test data is critical to ensure comprehensive testing of the system. Any errors identified during testing are addressed and corrected, and the corrections are noted for future reference.

5.2.5 Automation Testing

Automation testing is a software testing approach that employs specialized automated testing software tools to execute a suite of test cases. Its primary purpose is to verify that the software or equipment operates precisely as intended. Automation testing identifies defects, bugs, and other issues that may arise during product development. While some types of testing, such as functional or regression testing, can be performed manually, there are numerous benefits to automating the process. Automation testing can be executed at any time of day and uses scripted sequences to evaluate the software. The results are reported, and this information can be compared to previous test runs. Automation developers typically write code in programming languages such as C#, JavaScript, and Ruby.

5.2.6 Selenium Testing

Selenium is an open-source automated testing framework used to verify web applications across different browsers and platforms. Selenium allows for the creation of test scripts in various programming languages such as Java, C#, and Python. Jason Huggins, an engineer at Thought Works, developed Selenium in 2004 while working on a web application that required frequent

testing. He created a JavaScript program called "JavaScriptTestRunner" to automate browser actions and improve testing efficiency. Selenium has since evolved and continues to be developed by a team of contributors.

In addition to Selenium, another popular tool used for automated testing is Cucumber. Cucumber is an open-source software testing framework that supports behavior-driven development (BDD). It allows for the creation of executable specifications in a human-readable format called Gherkin. One of the advantages of using Cucumber is its ability to bridge the gap between business stakeholders and technical teams. By using a common language, Cucumber facilitates effective communication and collaboration during the testing process. It promotes a shared understanding of the requirements and helps ensure that the developed software meets the intended business goals.

Cucumber can be integrated with Selenium to combine the benefits of both tools. Selenium is used for interacting with web browsers and automating browser actions, while Cucumber provides a structured framework for organizing and executing tests. This combination allows for the creation of end-to-end tests that verify the behavior of web applications across different browsers and platforms, using a business-readable and maintainable format.

Test Case 1 -Customer Login

Code

```
from datetime import datetime
from django.test import TestCase
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
class Hosttest(TestCase):

    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.implicitly_wait(10)
        self.live_server_url = 'http://127.0.0.1:8000/'
```

```
def tearDown(self):
    self.driver.quit()
def test_01_login_page(self):
    driver = self.driver
    driver.get(self.live_server_url)
    driver.maximize_window()
    time.sleep(1)
    login=driver.find_element(By.CSS_SELECTOR,"i.fa.fa-fw.fa-user.text-light.mr-1")
    login.click()
    login=driver.find_element(By.CSS_SELECTOR,"a.dropdown-item[href='/register/login/']")
    login.click()
    time.sleep(2)
    email = driver.find_element(By.CSS_SELECTOR, "input[name='email'][id='your_email']")
    email.send_keys("ria@gmail.com")
    password=driver.find_element(By.CSS_SELECTOR,
    "input[name='password'][id='your_pass']")
    password.send_keys("Ria@12")
    time.sleep(2)
```

Eg.Screenshot

```
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
There was an error managing chrome (error sending request for url (https://googlechromelabs.github.io/chrome-for-testing/known-good-versions-with-downloads.json): error trying to connect: dns error: No such host is known. (os error 11001)); using driver found in the cache

DevTools listening on ws://127.0.0.1:50166/devtools/browser/8893b415-c4d7-4799-8da3-2eeef44ee599
Test passed
```

Eg. Test Report

Test Case 1					
Project Name:GustoGrove					
Login Test Case					
Test Case ID: Test_1			Test Designed By: Aleena Thomas		
Test Priority(Low/Medium/High):			Test Designed Date: High		
Module Name:			Test Executed By : Mr.Jobin T J		
Test Title : Customer Login			Test Execution Date:		
Description: Verify login with valid email and password					
Pre-Condition :User has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigation to Login Page		Dashboard should be displayed	Login page displayed	Pass
2	Provide Valid Email	Ria@gmail.com	customer should be able to Login	cCstomer Logged in and navigated to home Dashboard with records	Pass
3	Provide Valid Password	Ria@12			
4	Click on Login button				
Post-Condition: customer is validated with database and successfully login into account. The Account session details are logged in database					

Test Case 2: Customer - Add to wishlist

Code

```

from datetime import datetime
from django.test import TestCase
from selenium import webdriver

```

```

from selenium.webdriver.common.keys import Keys
import time
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
class Hosttest(TestCase):
    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.implicitly_wait(10)
        self.live_server_url = 'http://127.0.0.1:8000/'
    def tearDown(self):
        self.driver.quit()
home=driver.find_element(By.CSS_SELECTOR,"a.nav-link[href='/']")
home.click()
time.sleep(2)
shop=driver.find_element(By.CSS_SELECTOR,"a.nav-link[href='/shop/']")
shop.click()
home2=driver.find_element(By.CSS_SELECTOR,"a.nav-link[href='/']")
home2.click()
time.sleep(2)
wish_view=driver.find_element(By.CSS_SELECTOR,"a.btn.btn-success i.fas.fa-heart")
wish_view.click()
time.sleep(2)

```

Screenshot

```

(envmt) D:\ProjectS9\project>py manage.py test
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
DevTools listening on ws://127.0.0.1:51011/devtools/browser/f76c2ff6-01d9-4172-ab2b-a3f9a9092e40
Test passed -wishlist

```

Test report

Test Case 2

Project Name:GustoGrove

Add to wishlist Test Case

Test Case ID: Test_2			Test Designed By: Aleena Thomas		
Test Priority(Low/Medium/High):			Test Designed Date: High		
Module Name: Add to wishlist			Test Executed By : Mr.Jobin T J		
Test Title : Customer cart			Test Execution Date:		
Description:The product is added to cart by the user					
Pre-Condition: User has valid email and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigation to Login Page		Dashboard should be displayed	Login page displayed	Pass
2	Provide Valid Email	Ria@gmail.com	customer should be able to Login	cCstomer Logged in and navigated to home Dashboard with records	Pass
3	Provide Valid Password	Ria@12			
4	Click on Login button				
5	The User navigates to the shop page and clicks on add to wishlist icon of a product		User should redirect to shop page and should be able to add an item to thewishlist	User navigated to shop page and added an item to wishlist	Pass
Post-Condition: User successfully added an item to the wishlist					

Test Case 3: Customer - Add to Cart

Code

```

from datetime import datetime
from django.test import TestCase
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains

```

```

from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
class Hosttest(TestCase):
    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.implicitly_wait(10)
        self.live_server_url = 'http://127.0.0.1:8000/'
    def tearDown(self):
        self.driver.quit()
home=driver.find_element(By.CSS_SELECTOR,"a.nav-link[href='/']")
home.click()
time.sleep(2)
shop=driver.find_element(By.CSS_SELECTOR,"a.nav-link[href='/shop/']")
shop.click()
time.sleep(2)
cart=driver.find_element(By.CSS_SELECTOR,"a.btn.btn-success.text-
white[href='/add_to_cart/3/']")
cart.click()
time.sleep(2)

```

Screenshot

```

(envmt) D:\ProjectS9\project>py manage.py test
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).

DevTools listening on ws://127.0.0.1:50610/devtools/browser/7510f2e5-a04b-41d1-b4aa-c0e1c162321e
Test passed

```

Test report

Test Case 3	
Project Name:GustoGrove	
Add to cart Test Case	
Test Case ID: Test_3	Test Designed By: Aleena Thomas
Test Priority(Low/Medium/High):	Test Designed Date: High
Module Name: Add to cart	Test Executed By : Mr.Jobin T J
Test Title : Customer cart	Test Execution Date:

Description: The product is added to cart by the user					
Pre-Condition: User has valid email and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigation to Login Page		Dashboard should be displayed	Login page displayed	Pass
2	Provide Valid Email	Ria@gmail.com	customer should be able to Login	cCstomer Logged in and navigated to home Dashboard with records	Pass
3	Provide Valid Password	Ria@12			
4	Click on Login button				
5	The User navigates to the shop page and clicks on add to cart button of a product		User should redirect to shop page and should be able to add an item to the cart	User navigated to shop page and added an item to cart	Pass
Post-Condition: User successfully added an item to the cart					

CHAPTER 6

IMPLEMENTATION

6.1 INTRODUCTION

Implementation is the phase of the project where the theoretical design is turned into work system. This can be considered the most important step in achieving a successful new system gain user confidence that the new system works and is efficient and accurate. It is mainly concerns user training and documentation. Conversion usually takes approx at the same time as user training or later. Implementation simply means calling a new one system design for implementation, which is the conversion of a new modified system design working.

At this stage, the user section carries the main workload, the most experience major upheaval and has the most significant impact on the current system. Poorly designed or controlled implementation can lead to confusion and chaos. It's a whole new system new, replace or modify an existing manual or automated system implementation is necessary to meet the needs of the organization. System implementation covers everything operations necessary to migrate from the old system to the new one. The system can only be implemented after extensive testing and found to work as intended. System staff will assess the feasibility of the system. Implementation requires thorough work in three main areas: training, system testing and migration. Implementation phase involves careful planning, study of the system and limits, and design of methods of achievement an exchange.

The implementation state involves the following tasks:

- Careful planning.
- Investigation of system and constraints.
- Design of methods to achieve the changeover.

6.1 IMPLEMENTATION PROCEDURES

Software deployment means the final installation of the package in its real environment, objectives and the functioning of the system in a satisfactory manner. In many organizations someone who doesn't use it subscribes to a software project. In the year In the early stages, people doubt the software, but we must make sure that the resistance does not build, because it is necessary to ensure that:

- The active user must be aware of the benefits of using the new system.
- Their confidence in the software is built up.
- Proper guidance is imparted to the user so that he is comfortable in using the application

Before going ahead and viewing the system, the user must know that for viewing the result,

the server program should be running in the server. If the server object is not up running on the server, the actual process won't take place.

6.1.1 User Training

User training is designed to prepare the user for testing and converting the system. To achieve the objective and benefits expected from computer-based system, it is essential for the people who will be involved to be confident of their role in the new system. As system becomes more complex, the need for training is more important. By user training the user comes to know how to enter data, respond to error messages, interrogate the database and call up routine that will produce reports and perform other necessary functions.

6.1.2 Training on the Application Software

After providing the necessary basic training on computer awareness, it is essential to provide training on the new application software to the user. This training should include the underlying philosophy of using the new system, such as the flow of screens, screen design, the type of help available on the screen, the types of errors that may occur while entering data, and the corresponding validation checks for each entry, and ways to correct the data entered. Additionally, the training should cover information specific to the user or group, which is necessary to use the system or part of the system effectively. It is important to note that this training may differ across different user groups and levels of hierarchy.

6.1.3 System Maintenance

The maintenance phase is a crucial aspect of the software development cycle, as it is the time when the software is actually put to use and performs its intended functions. Proper maintenance is essential to ensure that the system remains functional, reliable, and adaptable to changes in the system environment. Maintenance activities go beyond simply identifying and fixing errors or bugs in the system. It may involve updates to the software, modifications to its functionalities, and enhancements to its performance, among other things. In essence, software maintenance is an ongoing process that requires continuous monitoring, evaluation, and improvement of the system to meet changing user needs and requirements.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The GustoGrove project is a resounding success, representing a remarkable achievement in the realm of online spice shopping. This comprehensive e-commerce platform has been meticulously designed to provide spice enthusiasts with a seamless, user-friendly, and enriching shopping experience. The website is designed to help you find and choose spices effortlessly. You can search for spices, learn about them, and buy what you like.

The people who run the spice store can add, change, or remove spices as needed. You, as a customer, can create an account to keep track of your orders history . You can also put spices in a virtual shopping cart and buy them securely. User accounts, complete with personalized profiles, enable customers to easily track their order history and, enhancing the overall shopping experience. The project's visually captivating home page, featuring enticing visuals of various spices, welcomes visitors to a world of sensory delight. The robust search functionality and suggested search terms provide an intuitive and efficient way for users to explore an extensive collection of high-quality spices. The GustoGrove Spice Mart project is a testament to how technology can elevate the culinary journey. In a nutshell, GustoGrove Spice Mart makes shopping for spices easy, informative, and fun. It sets a new standard for buying spices online, promising you a convenient and trustworthy experience. It's the future of spice shopping, bringing the world of flavors to your fingertips with confidence and simplicity

7.2 FUTURE SCOPE

The GustoGrove Spice Mart system has a significant potential for further improvement and enhancements to provide an even more satisfying user experience. While the current system offers essential functionalities, introducing the following updates will make the web application an even more preferable choice over traditional in-store shopping:

- **Enhanced User Profiles:** Improve user profiles to include more personalization options, allowing customers to set preferences, save favorite spices, and receive tailored recommendations.
- **User Reviews and Ratings:** Allow users to submit reviews and ratings for spices they've purchased, enhancing trust and helping other customers make informed choices.

- **Interactive Cooking Guides:** Develop interactive cooking guides that suggest recipes based on the spices in a user's cart, offering step-by-step instructions and pairing suggestions.
- **Custom Spice Blending:** Allow users to create and order custom spice blends, tailored to their taste preferences, and have them delivered to their doorstep.
- **Spice Subscription Boxes:** Launch subscription boxes where customers can receive a curated selection of spices regularly, making spice discovery an ongoing culinary journey.
- **Real-Time Spice Alerts:** Enable users to set alerts for when specific spices are available, ensuring they never miss out on their favorite products.

These updates will not only make GustoGrove Spice Mart an even more attractive and user-friendly platform but also set new standards for online spice shopping, making it the preferred choice for spice enthusiasts seeking an enhanced, informative, and interactive culinary journey.

.

.

CHAPTER 8

BIBLIOGRAPHY

REFERENCES:

- Complete CSS Guide ,Maxine Sherrin and John Allsopp-O'ReillyMedia; September 2012
- PankajJalote, “Software engineering: a precise approach”

WEBSITES:

- www.w3schools.com
- <https://docs.djangoproject.com>
- www.bootstrap.com
- <https://www.tutorialspoint.com/django/index.htm>

CHAPTER 9

APPENDIX

9.1 Sample Code

Login page

```
{% load static %}
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}">
  <link rel="stylesheet" href="{% static 'css/templatemo.css' %}">
  <link rel="stylesheet" href="{% static '/css/custom.css' %}">
  <link rel="stylesheet"
href="https://fonts.googleapis.com/css2?family=Roboto:wght@100;200;300;400;500;700;900&display=s
wap">
  <link rel="stylesheet" href="{% static 'css/fontawesome.min.css' %}">
</head>
<body>
  <nav class="navbar navbar-expand-lg">
    <div class="container d-flex justify-content-between align-items-center">
      <div class="logo">
        <a class="navbar-brand" href="index.html">
          
        </a>
      </div>
      <button class="navbar-toggler border-0" type="button" data-bs-toggle="collapse" data-bs-
target="#templatemo_main_nav" aria-controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="align-self-center collapse navbar-collapse flex-fill d-lg-flex justify-content-lg-
between" id="templatemo_main_nav">
        <div class="flex-fill">
          <ul class="nav navbar-nav d-flex justify-content-between mx-lg-auto">
            <li class="nav-item">
              <a class="nav-link" href="index.html">Home</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="about.html">About</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="shop.html">Shop</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="contact.html">Contact</a>
            </li>
          </ul>
        </div>
        <div class="navbar align-self-center d-flex">
          <div class="d-lg-none flex-sm-fill mt-3 mb-4 col-7 col-sm-auto pr-3">
            <a class="nav-icon d-none d-lg-inline" href="#" data-bs-toggle="modal" data-bs-
target="#templatemo_search">
              <i class="fa fa-fw fa-search text-light mr-2"></i>
            </a>
          </div>
        </div>
      </div>
    </div>
  </nav>
```

```

        {% if user.is_authenticated %}
        <h4 class="text-light">{{user.name}}</h4>
        <div style="margin-left: 5px;" class="mt-2">
            <a class="nav-icon position-relative text-decoration-none" href="{% url 'logout' %}"
data-bs-toggle="tooltip" data-bs-placement="bottom" title="Logout">
                <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor" class="bi bi-box-arrow-right" viewBox="0 0 16 16">
                    <path fill-rule="evenodd" d="M10 12.5a.5.5 0 0 1-.5-.5h-8a.5.5 0 0 1-.5-.5v-9a.5.5 0
0 1 .5-.5h8a.5.5 0 0 1 .5.5v2a.5.5 0 0 1 0v-2A1.5 1.5 0 0 0 9.5 2h-8A1.5 1.5 0 0 0 3.5v9A1.5 1.5 0 0 0
1.5 14h8a1.5 1.5 0 0 0 1.5-1.5v-2a.5.5 0 0 1 0v2z"/>
                    <path fill-rule="evenodd" d="M15.854 8.354a.5.5 0 0 0-.708l-3-3a.5.5 0 0 0-.
.708.708L14.293 7.5H5.5a.5.5 0 0 0 1h8.793l-2.147 2.146a.5.5 0 0 0 .708.708l3-3z"/>
                </svg>
            </a>
        </div>
        {% endif %}
    </div>
</nav>
<div>
    {% if messages %}
    {% for message in messages %}
        <span id="email-error" class="error-message">{{ message }}</span>
    {% endfor %}
    {% endif %}
</div>
<div class="register d-flex justify-content-center " >
    <div class="form-container">
        <div class="image-holder"></div>
        <form method="post" onsubmit="return validateLoginForm()">
            {% csrf_token %}
            <h2 class="text-center"><strong>Login</strong> to your account.</h2>
            <div class="form-group">
                <input class="form-control" type="email" name="email" id="login-email"
placeholder="Email">
                <span id="login-email-error" class="error-message"></span>
            </div>
            <div class="form-group">
                <br>
                <input class="form-control" type="password" name="password" id="login-password"
placeholder="Password">
                <span id="login-password-error" class="error-message"></span>
            </div>
            {% if error_message %}
            <div class="d-flex justify-content-center"><p class="text-danger mt-2 fs-
6">{{ error_message }}</p></div>
            {% endif %}

            <div class="form-group"><button class="btn btn-primary btn-block w-100"
type="submit">Login</button></div>
            <a href="{% url 'password_reset' %}" class="already reset-link">Forgot Password? <span>

```

```

Reset here</span>.</a>
    </form>
  </div>
</div>
</body>
</html>

```

Cart.html

```

<!-- cart.html -->
<html>
  <head>
    {% block style %}
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css"
    integrity="sha384-
    rbsA2VBKQhggwzxH7pPCaAqO46MgnOM80zW1RWuH61DGLwZJEdK2Kadq2F9CUG65"
    crossorigin="anonymous">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.min.js" integrity="sha384-
    cuYeSxntonz0PPNIHhBs68uyIAVpIIIOZZ5JqeqvYYIcEL727kskC66kF92t6Xl2V"
    crossorigin="anonymous"></script>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
    awesome/5.15.3/css/all.min.css">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"
    integrity="sha384-kenU1KFdBLE4zVF0s0G1M5b4hcpyD9F7jL+jjXkk+Q2h455rYXK/7HAuoJI+0I4"
    crossorigin="anonymous"></script>
  </head>
  <body>
    {% block content %}
    {% if messages %}
      {% for message in messages %}
        <div class="alert alert-danger text-center">{{ message }}</div>
      {% endfor %}
    {% endif %}
    <div class="card">
      <div class="row">
        <div class="col-md-8 cart">
          <div class="title">
            <div class="row">
              <div class="col"><h4><b>Shopping Cart</b></h4></div>
              <div class="col align-self-center text-right text-muted">{{ total_items }} items</div>
            </div>
          </div>
          {% for item in cart_items %}
          <!-- Single item -->
          <div class="row border-top border-bottom">
            <div class="row main align-items-center">
              <div class="col-2"></div>
              <div class="col">
                <div class="row text-muted">{{ item.product.product_name }}</div>
              </div>
              <div class="col">
                <a class="text-decoration-none border" href="{% url 'decrease_item' item.id %}"></a>
                <a class="text-decoration-none" href="#" class="border">{{ item.quantity }}</a>
                <a class="text-decoration-none border" href="{% url 'increase_item' item.id %}">+</a>
              </div>
            </div>
          </div>
          </div>
        </div>
      </div>
    </div>
  </body>
</html>

```

```

        </div>
        <div class="col">₹ {{ item.product.price }}</div>
        <div class="col">
            <a href="{% url 'remove_from_cart' item.id %}" class="btn btn-danger"><i class="fas fa-trash-
alt"></i> Remove</a>
        </div>
    </div>
</div>
{% endfor %}
</body>
</html>

```

Views.py

```

def custom_login(request):
    if request.method == 'POST':
        email = request.POST.get('email')
        password = request.POST.get('password')
        if email and password:
            user = authenticate(request, email=email, password=password)
            if user is not None:
                auth_login(request, user)
                if user.is_customer:
                    return redirect('index') # Redirect to customer index page
                elif user.is_seller:
                    low_stock_notification_url = reverse('low_stock_notification', args=[user.id])
                    return redirect(low_stock_notification_url)
            else:
                error_message = "Invalid login credentials."
                return render(request, 'custlogin.html', {'error_message': error_message})
    return render(request, 'custlogin.html')

```

```

from django.shortcuts import render, redirect
from django.contrib.auth.decorators import login_required
from django.conf import settings
from django.contrib import messages
from .models import Customer_Profile
def index(request):

```

```

    user=request.user
    if user.is_anonymous:
        return render(request, 'index.html')
    elif user.is_seller==True:
        return redirect('seller_index')
    else:

```

```

        return render(request, 'index.html')
def contact(request):
    return render(request, 'contact.html')

```

```

def wishlist_view(request):
    if request.user.is_authenticated:

```

```
wishlist = Wishlist.objects.get_or_create(user=request.user)[0]
wishlist_products = wishlist.products.all()
return render(request, 'wishlist.html', {'wishlist_products': wishlist_products})
else:
    return render(request, 'login_required.html')

@login_required
def add_to_wishlist(request, product_id):
    product = get_object_or_404(Product, pk=product_id)
    wishlist, created = Wishlist.objects.get_or_create(user=request.user)
    wishlist.products.add(product)
    return JsonResponse({'message': 'Product added to wishlist'})

@login_required
def remove_from_wishlist(request, product_id):
    product = get_object_or_404(Product, pk=product_id)
    wishlist = get_object_or_404(Wishlist, user=request.user)
    wishlist.products.remove(product)
    success_message = f'{product.product_name} removed from your wishlist.'

    # You can pass the success_message to the template
    return redirect('wishlist')
def update_wishlist_quantity(request, product_id, new_quantity):
    try:
        new_quantity = int(new_quantity)
    except ValueError:
        return JsonResponse({'message': 'Invalid quantity'}, status=400)

    if new_quantity <= 0:
        return JsonResponse({'message': 'Quantity must be greater than zero'}, status=400)

    product = get_object_or_404(Product, pk=product_id)
    wishlist = get_object_or_404(Wishlist, user=request.user)

    # Update the quantity of the product in the wishlist
    item = wishlist.wishlist_items.filter(product=product).first()
    if item:
        item.quantity = new_quantity
        item.save()
        return JsonResponse({'message': 'Quantity updated successfully'})
    else:
        return JsonResponse({'message': 'Product not found in wishlist'}, status=404)

@never_cache
def seller_index(request):
    if request.user.is_seller:
        current_seller = request.user
        seller_products = Product.objects.filter(seller=current_seller)
        notification=Notification.objects.filter(seller_id=current_seller.id,read=False).count()
        product_data = []
        total_orders_all = 0
        unique_customers_all = 0
        total_amount_all = 0
        for product in seller_products:
            total_orders = Order.objects.filter(products=product).count()
            unique_customers = Order.objects.filter(products=product).values('user').distinct().count()
```

```
        total_amount = total_orders * product.price
        total_orders_all += total_orders
        unique_customers_all += unique_customers
        total_amount_all += total_amount

        product_data.append({
            'product': product,
            'total_orders': total_orders,
            'unique_customers': unique_customers,
            'total_amount': total_amount
        })

    context = {
        'seller_products': seller_products,
        'product_data': product_data,
        'total_orders_all': total_orders_all,
        'unique_customers_all': unique_customers_all,
        'total_amount_all': total_amount_all,
        'total_amount': total_amount,
        'unique_customers': unique_customers,
        'total_orders': total_orders,
        'notification': notification
    }

    return render(request, 'sellerindex.html', context)
else:
    return redirect('seller_reg_step')
@never_cache
def shop(request):
    supplier_products = Product.objects.all()
    paginator = Paginator(supplier_products, 6)
    page_number = request.GET.get('page')
    page = paginator.get_page(page_number)
    context = {
        'supplier_products': supplier_products,
        'page': page
    }
    return render(request, 'shop.html', context)

# @login_required
def product_detail(request, product_id):
    product = get_object_or_404(Product, pk=product_id)
    return render(request, 'shop-single.html', {'product': product})
@login_required(login_url='custom_login')
def seller_product_listing(request):
    if request.user.is_authenticated:
        current_seller = request.user
        seller_products = Product.objects.filter(seller=current_seller)

        context = {
            'seller_products': seller_products
        }
```

```
        return render(request, 'product_list.html', context)
    else:

        return redirect('seller_register')

@login_required(login_url='custom_login')
def add_product(request):
    if request.user.isSeller:
        if request.method == 'POST':
            product_name = request.POST.get('product_name')
            description = request.POST.get('description')
            image = request.FILES.get('image')
            quantity_value = request.POST.get('quantity')
            quantity_prefix = request.POST.get('quantity-prefix')
            price = request.POST.get('price')
            brand_name = request.POST.get('brand_name')
            stock = request.POST.get('stock')
            product = Product(
                product_name=product_name,
                description=description,
                image=image,
                quantity_value=quantity_value,
                quantity_prefix=quantity_prefix,
                price=price,
                brand_name=brand_name,
                stock=stock,
                seller=request.user
            )
            product.save()
            return redirect('seller_index')
        else:
            return render(request, 'add_product.html')
    else:

        return redirect('seller_register')
```

9.1 Screen Shots

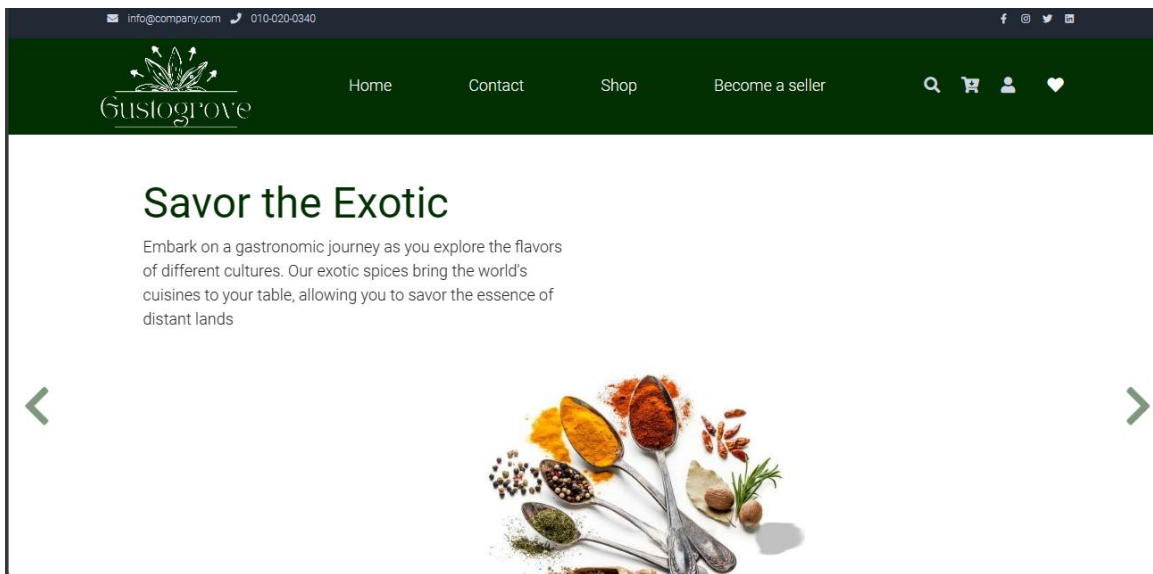


Fig . Index page

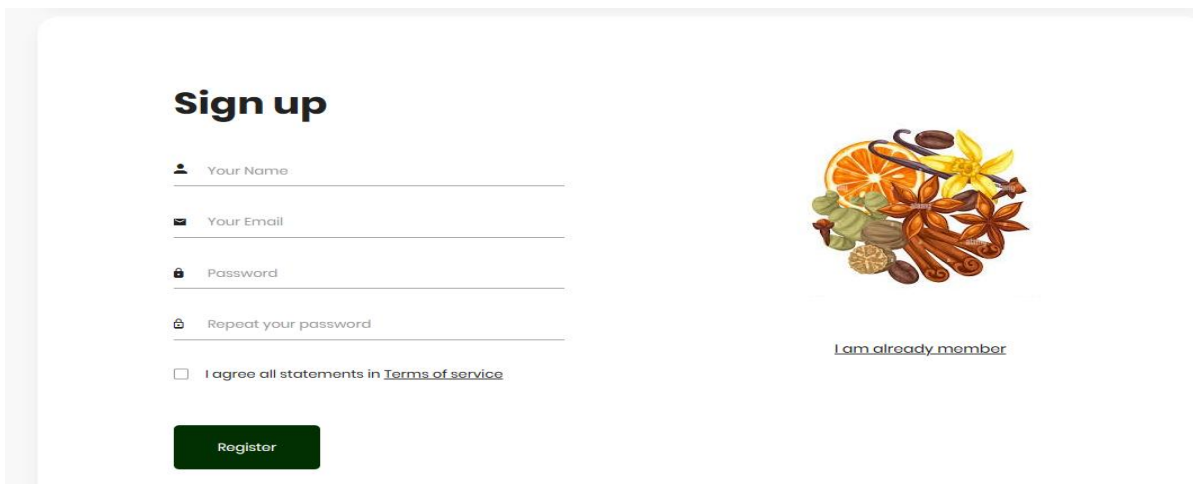


Fig .Customer Registration

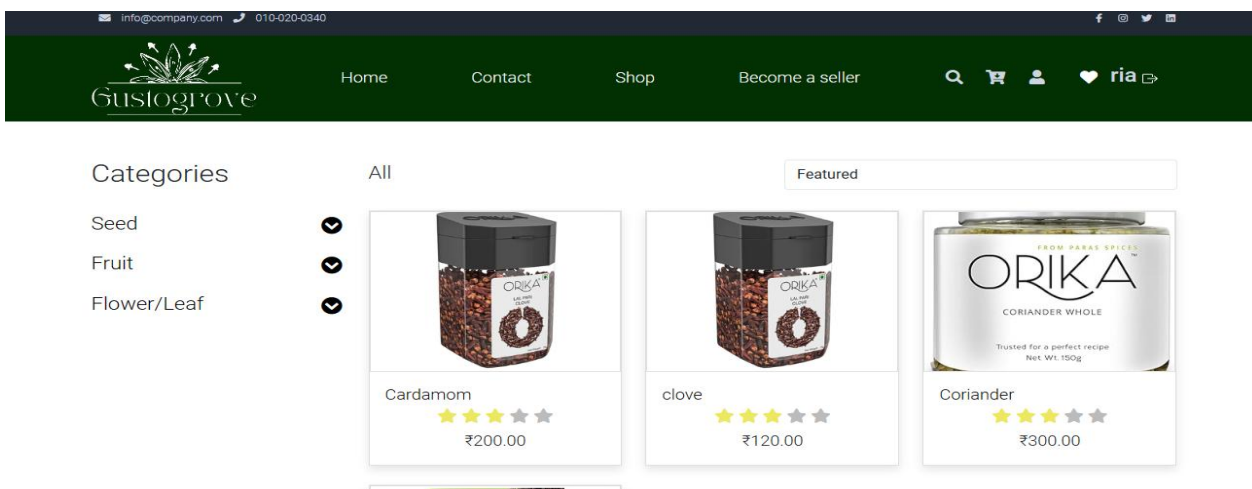


Fig .Shop page

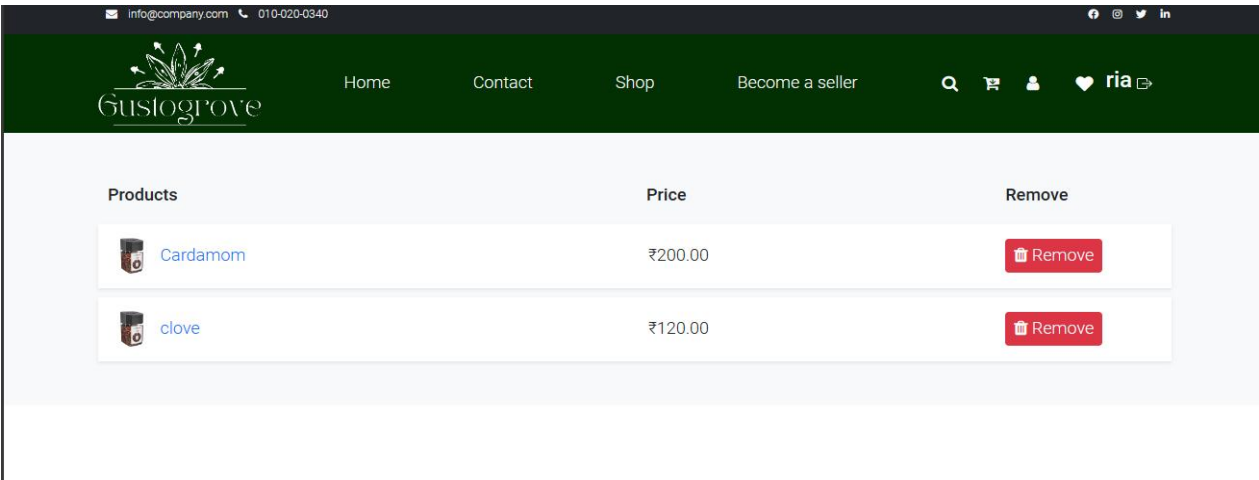


Fig. wishlist

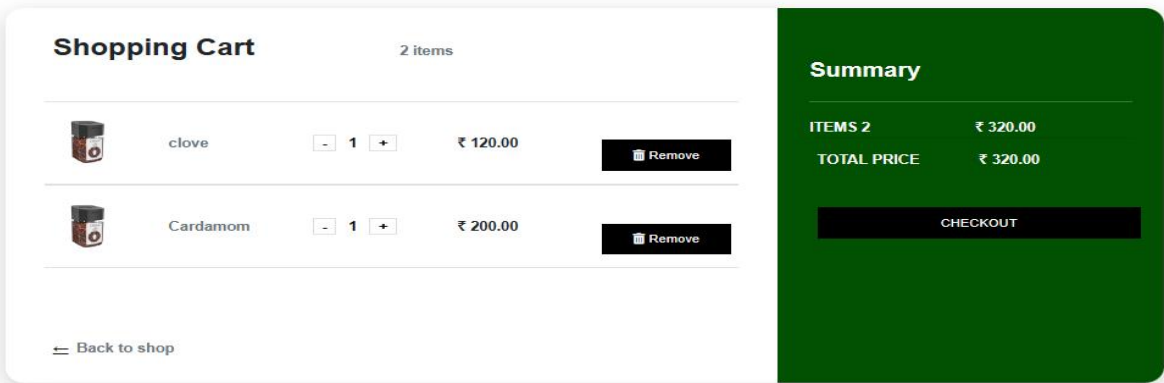


Fig .cart

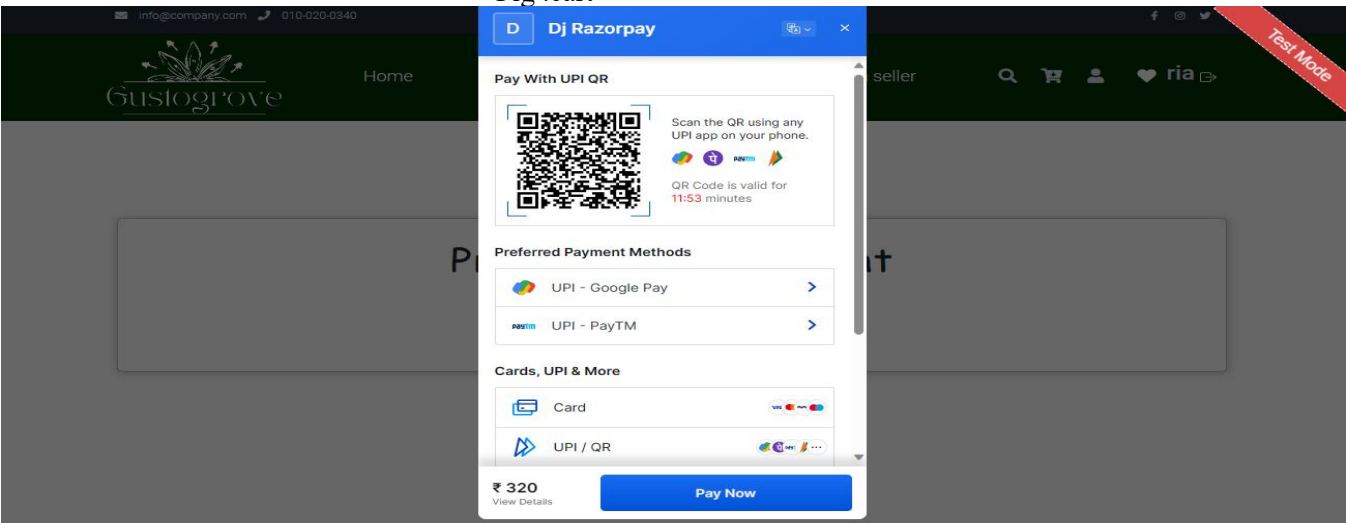


Fig.Payment

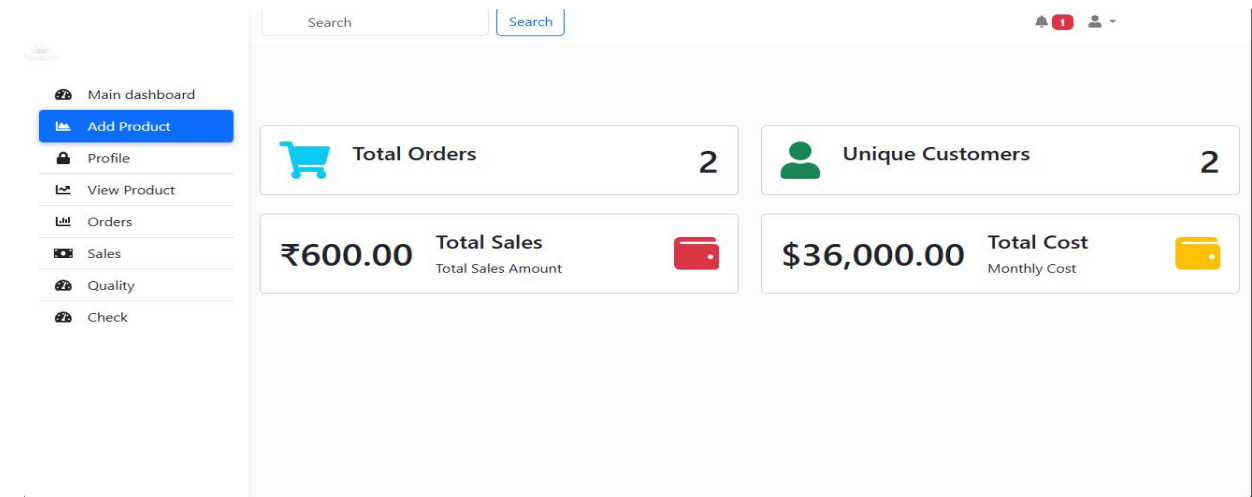


Fig.Seller index

image	product name	price	quantity	Instock	Date	Batch Id	
	Cardamom	₹200.00	500/gms	20	Sept. 18, 2023	P101	
	clove	₹120.00	60/gms	13	Sept. 18, 2023	P102	
	Coriander	₹300.00	100/gms	14	Oct. 1, 2023	P103	

Fig.View Product



Fig.orders





Image Classification

Select an Image

Choose File

No file chosen

Check

Fig.Quality check

