

CHAPTER-1

INTRODUCTION AND OVERVIEW

1. Introduction and Overview

1.1 Introduction

Introducing “**BizCart4all**”-**Comprehensive Seller Services**, a multi-vendor e-commerce platform designed to streamline online shopping and selling. Built using the MERN stack, BizCart4all offers robust features for admins, sellers, and buyers, ensuring a seamless and secure shopping experience. With dynamic product management, real-time chat support, and efficient order processing, our platform caters to all your e-commerce needs.

1.2 Problem Statement

The existing e-commerce landscape lacks a comprehensive solution that effectively caters to the needs of administrators, sellers, and buyers, resulting in a disjointed and inefficient shopping experience. Key challenges include inefficient database management causing slow performance, limited seller autonomy in managing storefronts, poor user interface design impacting user interactions, and security concerns with online transactions. Our project addresses these issues by developing a multi-vendor marketplace using MongoDB, React.js, Redux, and Tailwind CSS, revolutionizing e-commerce with a dynamic platform that fosters trust, enables seamless transactions, and enhances overall user experience.

1.3 Objectives

- To implement efficient database management strategies to ensure fast and reliable performance, enabling seamless handling of large volumes of product data.
- To provide sellers with autonomy in managing their storefronts, allowing them to customize listings, set prices, and manage inventory levels according to their unique offerings.
- To enhance security measures to ensure safe and secure online transactions, instilling trust and confidence among buyers.

1.4 Existing system

Current e-commerce platforms struggle to offer a unified and efficient experience for administrators, sellers, and buyers. These systems often suffer from slow database performance, restricted seller management capabilities, subpar user interfaces, and inadequate security measures, making it difficult to conduct seamless and secure online transactions.

1.5 Proposed system

We aim to develop a multi-vendor e-commerce platform using MongoDB, React.js, Redux, and Tailwind CSS. Our system will provide a seamless and secure shopping experience, with efficient database management, enhanced seller autonomy, and a user-friendly interface. The platform will feature real-time chat support, personalized vendor pages, and robust security measures to address the current shortcomings in the e-commerce landscape.

1.6 Features

1. Admin Capabilities:

- Add categories, view seller requests, and activate sellers based on provided data.
- Chat with sellers and handle payment requests efficiently.

2. Seller Capabilities:

- Add and update product information, including price, description, quantity, discount, and images.
- View added products and engage in real-time chat with buyers and admin.
- Manage shop information on a profile page and activate the shop upon admin approval.

3. Buyer Capabilities:

- View products, add to cart or Wishlist, and make secure payments.

- Search for products by categories or names, filter by prices, and browse products through personalized vendor pages and also chat with sellers.

4. Enhanced User Experience:

- Efficient database management for fast performance.
- Robust security measures for safe online transactions.
- User-friendly interface with a responsive design for a seamless shopping experience.

1.7 Languages and Software Tools Used

The user interface for this multi-vendor e-commerce platform was designed using React.js and Tailwind CSS. Redux was utilized for state management, and MongoDB served as the database for efficient data storage and retrieval. The entire system is built on the MERN stack, ensuring a robust and scalable solution for the platform's dynamic needs.

1.7.1 Technology Environment

1.7.1 MERN Stack

1.7.1.1 MongoDB

MongoDB is a leading NoSQL database that offers flexible, document-oriented storage, making it ideal for handling complex and unstructured data. It supports high performance, scalability, and indexing, which ensures efficient data retrieval and management. MongoDB's aggregation framework enhances data analysis capabilities, making it suitable for dynamic e-commerce environments.

1.7.1.2 Express.js

Express.js is a minimal and flexible Node.js web application framework that simplifies the development of robust and scalable server-side applications. It provides powerful routing and middleware capabilities, enabling the handling of HTTP requests and responses efficiently. Express.js is lightweight, making it an excellent choice for building RESTful APIs and managing server-side logic.

1.7.1.3 React.js

React.js is a popular JavaScript library for building dynamic and interactive user interfaces. It features a component-based architecture, allowing for the creation of reusable UI components. React's virtual DOM enhances performance by updating only the necessary parts of the UI, and React Hooks simplify state management and side effects in functional components.

1.7.1.4 Node.js

Node.js is a JavaScript runtime that enables server-side scripting and the development of scalable network applications. Its event-driven, non-blocking I/O model ensures efficient handling of asynchronous operations, and the extensive NPM ecosystem provides access to a wide range of libraries and tools, facilitating rapid development and deployment.

1.7.2 Additional Technologies

1.7.2.1 JavaScript

JavaScript (often shortened to JS) is a lightweight, interpreted, object-oriented language with first-class functions, and is best known as the scripting language for Web pages, but it's used in many non-browser environments as well. It is a prototype-based, multi-paradigm scripting language that is dynamic, and supports object-oriented, imperative, and functional programming styles. JavaScript runs on the client side of the web, which can be used to

design / program how the web pages behave on the occurrence of an event. JavaScript is an easy to learn and also powerful scripting language, widely used for controlling web page behaviour. Contrary to popular misconception, JavaScript is not "Interpreted Java". In a nutshell, JavaScript is a dynamic scripting language supporting prototype-based object construction. The basic syntax is intentionally similar to both Java and C++ to reduce the number of new concepts required to learn the language. Language constructs, such as if statements, for and while loops, and switch and try ... catch blocks function the same as in these languages (or nearly so). JavaScript can function as both a procedural and an object-oriented language. Objects are created programmatically in JavaScript, by attaching methods and properties to otherwise empty objects at run time, as opposed to the syntactic class definitions common in compiled languages like C++ and Java. Once an object has been constructed it can be used as a blueprint (or prototype) for creating similar objects.

1.7.2.2 Redux

Redux is a state management library that provides a predictable container for application state. It ensures a single source of truth by centralizing the state management and uses pure functions (reducers) to handle state changes. Redux supports middleware for handling asynchronous actions and integrates well with React to manage complex state interactions.

1.7.2.3 Tailwind CSS

Tailwind CSS is a utility-first CSS framework that enables the rapid development of custom user interfaces. It offers a comprehensive set of utility classes for building responsive designs directly in HTML. Tailwind CSS promotes a consistent design system, allows for extensive customization, and supports responsive design features.

1.7.3 Development Tools

1.7.3.1 Visual Studio Code

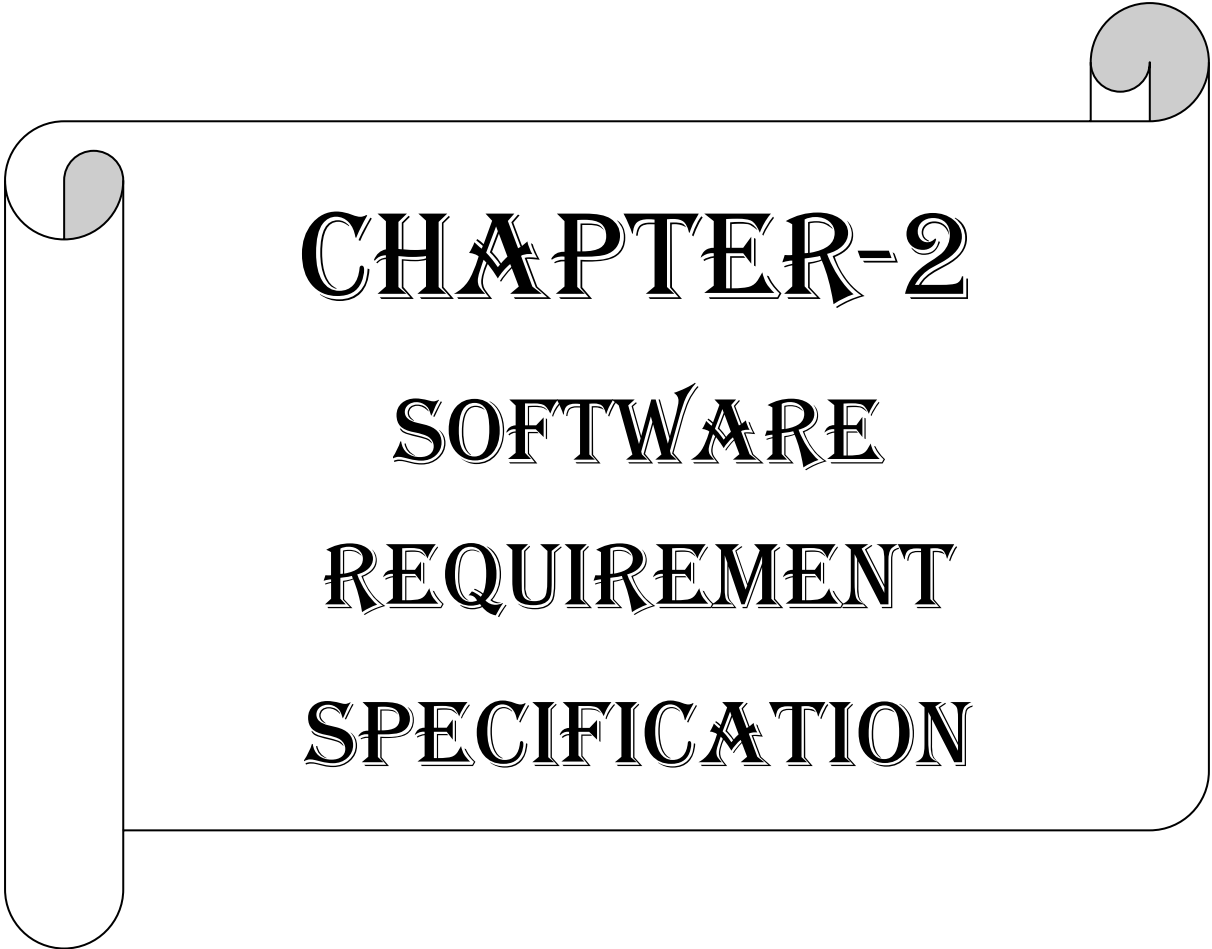
Visual Studio Code (VS Code) is a versatile and lightweight code editor that supports a wide range of programming languages and development tasks. It features a rich extension ecosystem, integrated terminal, advanced debugging tools, and built-in Git support, making it an essential tool for writing and managing code efficiently.

1.7.3.2 Git

Git is a distributed version-control system that tracks changes in source code and facilitates collaboration among developers. It supports branching and merging, enabling multiple developers to work on different features concurrently. Git maintains a history of changes, allowing for rollback and effective management of code revisions.

1.7.3.3 Postman

Postman is a collaborative API development tool that streamlines the process of designing, testing, and documenting APIs. It allows users to create and send API requests, analyse responses, and automate tests. Postman’s features include API documentation generation and team collaboration capabilities, which enhance API development workflows.

A decorative scroll graphic with a light gray background and a black outline. The scroll is unrolled in the center, revealing the chapter title. The top right corner of the scroll is curled up, and the bottom left corner is also curled up.

CHAPTER-2

SOFTWARE REQUIREMENT SPECIFICATION

2. Software Requirement Specification

2.1 Introduction

Requirement is defined as a condition needed by a user to solve a problem or achieve an objective; a condition or capability that must be met or possessed a system to satisfy a contract, a standard, a specification, or other formally imposed document. In software requirement of proposed system, that is, the capabilities that the system, which is yet to be developed, should have.

The software requirement specification (SRS) describes what the proposed software should do without describing how the software will do it. It has a document which completely describes external behaviour of the software. It is the first job of the software developer to study the system that needs to be developed and specifies the user requirement before going for the designing part.

2.2 Need for SRS

The origin of most software system is in the need of the client, who either wants to automate an existing manual system or describes a new software system. The software system itself is created by the developer. Finally, the complete system will be used by the end users. There are three major parties interested in new system; the client, the user and the developers. The requirements for the system that will satisfy the needs of the client, the users and the developers.

- SRS is the medium through which the client and the user needs are accurately specified.
- A SRS establishes the basis for agreement between the client and the supplier on what the software product will do.
- An SRS provides reference for validation of the final product.
- A high-quality SRS is pre-requisite to high quality software.
- A high-quality SRS reduces the development cost.

2.3 Purpose

As the world more competitive, you have to sharpen all your tools. Knowing what is on the customer mind is the more important thing we can do. What you want and what your customers want may not be same, but unless you give then what they want, you won't get what you want. The basic purpose of SRS is to bridge the communication gap between client and developer. It provides user friendly environment. This document contains information required for the developer of the software, client and user of the software. Another important purpose is helping the client understands their own needs. It also increases the customer of the product within the short period. Purpose of Time Table is to encapsulate the component you need to manage your daily routine in one package.

2.4 Project Scope

This multi-vendor e-commerce platform is designed to provide an intuitive and seamless experience for administrators, sellers, and buyers. The scope of the project includes efficient management of product listings, secure transactions, and real-time support across various user roles. It aims to streamline category management, enhance seller autonomy, and offer a user-friendly interface for buyers. Additionally, the platform supports responsive design and personalization, ensuring a dynamic and engaging shopping experience.

2.5 Overall Description

The main functions associated with the product are described in this section of SRS. The assumptions in this section result from interaction with the project stakeholders.

2.5.1 Product Perspective

- The multi-vendor e-commerce platform complements existing e-commerce tools by providing a comprehensive solution for managing products, transactions, and user interactions.
- It integrates various functionalities into a single system, offering a unified interface for administrators, sellers, and buyers.

- The platform ensures efficient operations with quick response times and streamlined handling of user actions.
- It features real-time chat support, secure payment processing, and a user-friendly design to optimize the shopping experience.
- The system is accessible and reliable across various devices and internet conditions, providing a consistent and engaging user experience.

2.5.2 Product Function

- The multi-vendor e-commerce platform provides a comprehensive solution for administrators, sellers, and buyers, facilitating seamless management of products, transactions, and communications.
- It integrates real-time chat support for instant assistance and efficient resolution of queries.
- The platform offers secure payment processing, ensuring safe and reliable transactions for all users.
- A user-friendly interface enhances the shopping experience, making it easy for users to navigate, search for products, and manage their accounts.
- The system supports responsive design, ensuring accessibility and functionality across various devices and screen sizes.

2.5.3 System Interface

- The platform features an intuitive and user-friendly interface, designed to simplify navigation for administrators, sellers, and buyers.
- Sellers can easily access and manage various functions such as product listings, transaction processing, and user communications.
- For buyers, the interface offers clear and accessible options for browsing products, adding items to the cart or Wishlist, and processing orders. The platform supports efficient searching, filtering, and sorting of products to enhance the shopping experience.

- Administrators can manage categories, review seller requests, and handle payment processing through a streamlined dashboard that provides all necessary tools in a single location.
- The frontend is built using React.js and Tailwind CSS, providing a responsive and visually appealing design that ensures a consistent experience across different devices and screen sizes.
- Data is stored and managed in MongoDB, ensuring efficient retrieval and updating of product information, user data, and transaction records.
- The system supports secure payment processing, real-time chat functionality, and a robust user authentication process to ensure a safe and reliable e-commerce environment.

2.6 Specific Requirement

2.6.1 External Interface Requirement

This application has striking feature that contain graphical user interface. So, any person who is authorized to access this system can easily understand what to do next when he is operating with it.

External interface requirement is divided into 3 types. They are:

- User Interface
- Hardware Interface Requirements
- Software Interface Requirements

2.6.1.1 User Interface

User interface is designed in user friendly manner. The labels are used on the view controller to direct the user of what are the controls are used for Textboxes, text area, dropdown list and buttons are used for input into the page. Date picker, auto complete textboxes are also used for inputting the information. The system is using swift for the user benefit. The frontend is built using React.js and Tailwind CSS for an engaging and responsive design.

2.6.1.2 Hardware Interface

- Desktop Computers: For accessing the platform via web browsers.
- Mobile Devices: For mobile access and usability.
- Simulators/Emulators: For testing and development purposes on various devices.

2.6.1.3 Software Interface

- Operating System: Microsoft Windows 10
- IDE: Visual Studio Code
- Frontend: React.js and Tailwind CSS
- Backend: MongoDB, Node.js, and Express.js

2.7 Functional Requirement

Functional Requirements essentially are the input/output behaviour specification for the software. Functional requirements are those that refer to the functionality of the system i.e. what services it will provide to the user. Non-functional requirement pertains to other information needed to produce the correct system and are detailed separately. The software must provide user interface to accept the valid details and display the information.

2.7.1 Module Description

The application consists of the following module,

❖ ADMIN

- Manage Categories: Admins can add, update, or remove product categories.
- Review Seller Requests: Admins can view and evaluate seller applications to decide on activation.
- Manage Payments: Admins can review payment requests from sellers and process payments.
- Chat Support: Admins can communicate with sellers and resolve issues as needed.

❖ SELLER

- **Profile Management:** Sellers can create and update their shop profile, which must be approved by the admin.
- **Add Products:** Sellers can add new products to existing categories, including details such as price, description, quantity, and images.
- **Update Products:** Sellers can modify product information and manage their inventory.
- **View Products:** Sellers can view a list of products they have added.
- **Chat with Buyers:** Sellers can communicate with buyers for queries and support.

❖ BUYER

- **Browse Products:** Buyers can view products, search by category or name, and filter by price.
- **Add to Cart/Wishlist:** Buyers can add items to their cart or Wishlist.
- **Secure Payments:** Buyers can make secure payments for their orders.
- **Order Tracking:** Buyers can view and sort their orders based on status.
- **Vendor Pages:** Buyers can browse products from specific vendors and place orders directly from vendor pages.

2.8 Performance Requirement

- **Speed:** The overall system should operate quickly to ensure a smooth user experience, with minimal delays in loading and processing.
- **User-Friendliness:** The system must be designed to be intuitive and easy to use, ensuring that users can navigate and interact with it effortlessly.
- **Backup System:** A robust backup system must be implemented to store and protect the database, ensuring data integrity and availability in case of system failures or data loss.

- **Security:** Access to the admin features must be restricted to authorized users who possess valid credentials, ensuring that only those with the proper authorization can use the administrative functions.

2.9 System Attribute

2.9.1 Adaptability

The system supports real-time interactions, allowing users, sellers, and admins to engage with the platform dynamically and adapt to changing requirements.

2.9.2 Correctness

The project ensures that all functionalities—such as product management, order processing, and user interactions—are executed accurately according to specified methods and requirements.

2.9.3 Maintainability

Evolve to meet the changing needs of users.

2.9.4 Dependability and Security

The platform includes security measures to protect user data and financial transactions. It is designed to avoid any physical or economic harm in the event of system failures.

2.9.5 Efficiency

Includes responsiveness, processing time and memory utilization.

2.9.6 Acceptability

The system is designed to be user-friendly and meet the needs of administrators, sellers, and buyers, ensuring a positive experience for all types of users.

2.9.7 Portability

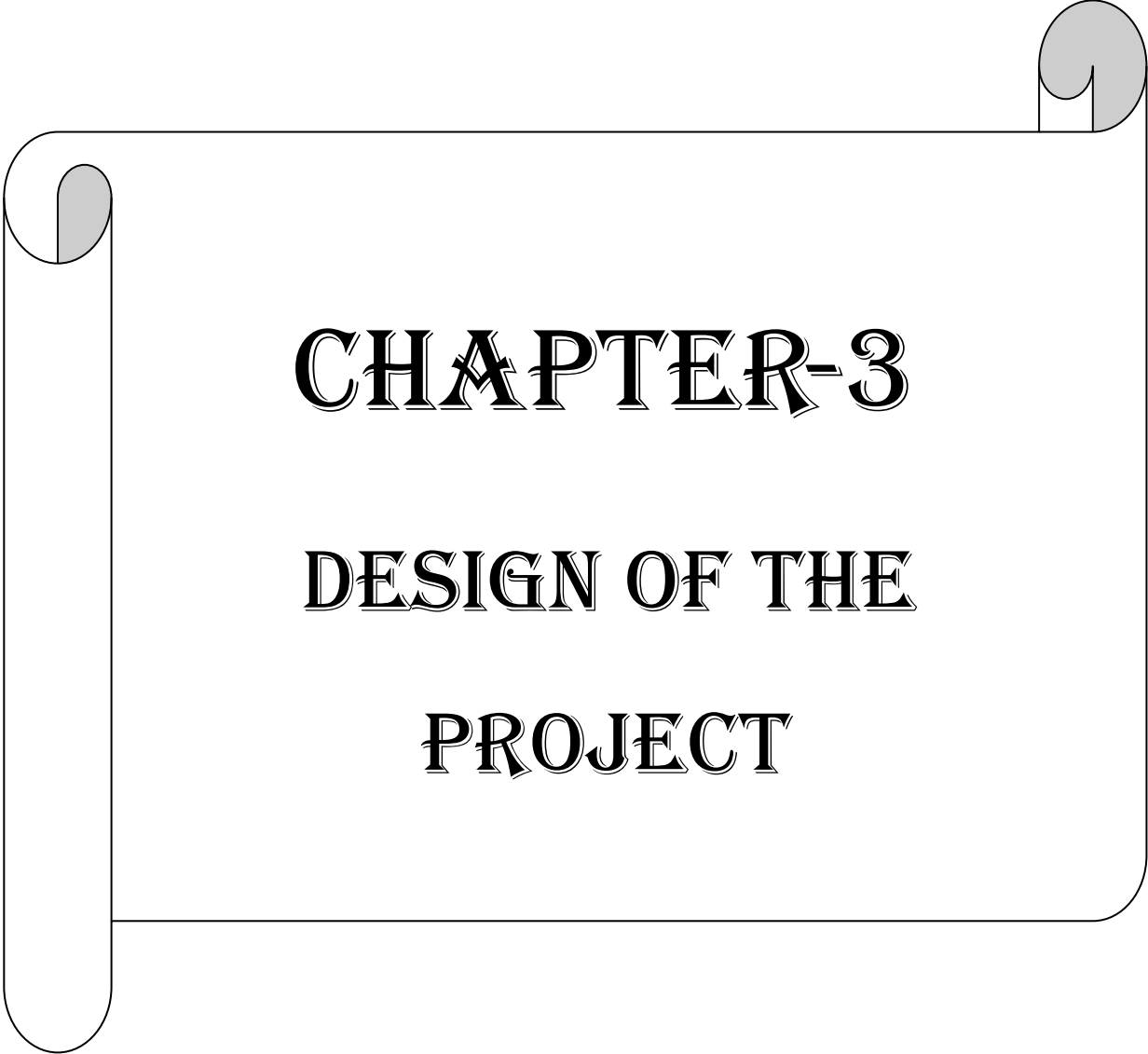
This software works in Android platform.

2.9.8 Testability

Checking for the working of the application for the required output is possible.

2.9.9 Designed Constraints

Software constraint: The software runs in Android

A decorative frame resembling a scroll or a piece of parchment. It has a large, light gray circular element at the top right corner and a smaller, similar element at the top left corner. The frame is outlined in black and contains the chapter title in a bold, black, serif font.

CHAPTER-3

DESIGN OF THE PROJECT

3.Design of the Project

3.1 System Design

3.1.1 Introduction

System design is the process or art of defining the architecture, components, and modules. Interface and the data for a system to satisfy specified requirements. System design is also called as top-level design, in which the focus is on deciding which modules is need for the system.

3.1.2 Overview

System design is therefore the process of defining and developing a system to satisfy specified requirements of the user. Until the 1990’s standardization of hardware and software resulted in the ability to build modular systems. The increasing the importance of the software running on the generic platform has enhanced the discipline of software engineering.

Object oriented analysis and design methods are becoming the most widely standard language used in the object-oriented analysis and design. It is widely used for modeling software system and it is increasingly used for big designing non software system and organizations.

3.1.3 Logical Design

The logical design of the system pertains to an abstract representation of data flows, inputs, and outputs of the system. This is often conducted via modeling, which involves a simplistic representation of an actual system. In the context of system design, modeling can undertake the following form, includes

- Data flow diagrams
- Entity relationship diagram

3.1.4 Physical Design

The physical design relates to actual input and output processes of the system. This is laid down in terms of how data is input the system, how it is verified. Physical design, in this context, does not refer to the tangible physical design of an information system.

To use an analogy, a personal computers physical design involves input via keyboard. Processing within the CPU, and output via monitor, printer etc. It would not concern the actual layout of the tangible hardware, which for a pc would be a monitor, CPU, motherboard, hard drive, modems, video/graphics cards, USB slots etc.

3.1.5 Scope and Overview

3.1.5.1 Scope

The scope of the project is from giving the constraint and accessing the data and represents it in graphical form. It helps the user to understand the data. To do this we find the solution using collected requirements and design this solution. Design is done in this phase. This phase is done in correct way because it effects the next phases also.

3.1.5.2 Overview

The software requirement specification step of a software development process yields specifications that are used in software engineering. If the system is semi-automated or user-centered, software design may involve user experience design yielding a story board to help determine those specifications.

If the software is completely automated, a software design may be as a flowchart or text describing a planned sequence of events. There are also semi standard method like Unified Modelling Language and Fundamental modeling concepts. In either case some documentation of the plan is usually the product of the design. A software design may be platform independent or platform specific, depending on the availability of the technology called for by the design.

3.2 System Overview

Mining the user information and represent it in the graphical form is the main functionality of a software. We develop an application which working on the concept of data mining and satisfies the user constraints.

3.2.1 Context Flow Datagram (CFD)

A context flow diagram defines the boundary between the system, or part of the system, and its environment, showing the entities that react with it. It represents all the external entities that may interact with the system. In context flow diagram, the entire system is treated as a single process and all input, output, sinks and source of the process are shown. Using CFD Design methodology DFD of the system is represented. The environment in which (the context) the software is used is depicted in CFD.

The CFD shows the external entity acting on the software. The CFD shows the external entity on the software. The software is shown here in CFD as a single process. The best CFD's are used to display how a system interoperates at a very high level, or how systems operate and interact logically.

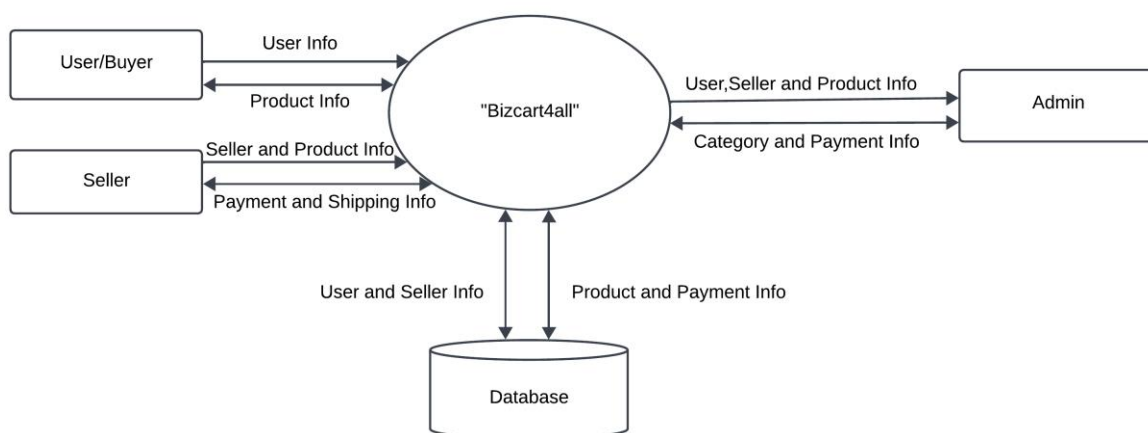


Fig.3.2.1.1 Control Flow Diagram

3.2.2 Data Flow Diagram (DFD)

Data flow diagram is also called Data Flow Graph or Bubble chart which is commonly used during problem analysis. A DFD shows the flow of data through a system as a function that transforms the inputs into desired outputs.

The process in DFD is represented by named circle and data flow are represented by named arrows entering or leaving the bubbles. A source or sink is typically outside the main system of study. The two main purposes of DFD are:

- To provide an indication of flow of data as they move through the system.
- To depict the function that transforms the flow.

The DFD provides additional information that is used during the analysis of the problem domain and serves as a basis for modelling functions. The DFD may be used to represent a system or software at any level of abstraction. In fact, DFD provides a mechanism for functional modelling as well as information flow modelling.

A level 0 DFD also called fundamental system model or a context model representing the entire software elements as a single bubble with inputs and output data indicated incoming and outgoing arrows respectively. Additional process and information flows are represented as level 0 DFD, which is partitioned to reveal more details. For example: a level 0 DFD might contain four or five bubbles with interconnecting arrows. Each of the processes represented level 1 is a sub function of overall system depicted in the context model.

The basic notation used to create a DFD makes it easy to analyse and understand. The DFD is a graphical tool that can be very valuable during software requirement analysis.

3.2.2.1 Notation for DFD


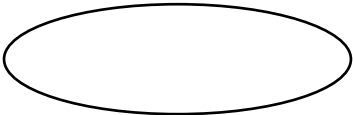
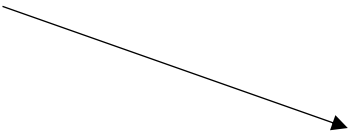
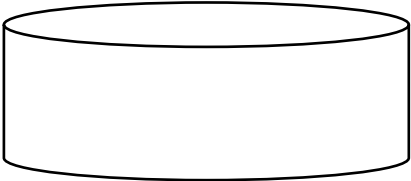

ENTITY: 	<p>External entities are outside the system, but they either supply input data into the system or use system output. These represented by rectangle. It is used for specifying from data where data comes and where it reaches.</p>
PROCESS: 	<p>A process shows a transformation or manipulation of data flows within the system. A process transforms incoming data flow into outgoing data flow.</p>
DATAFLOW: 	<p>A data flow shows flow of information from source to destination. A data flow is represented by a line, with arrow head showing the direction of flow.</p>
DATABASE: 	<p>Databases are outside the system, is used to store the data in backend.</p>
TABLES: 	<p>Tables are the part of databases. Used to store the data.</p>

Table.3.2.2.1 Notation for DFD

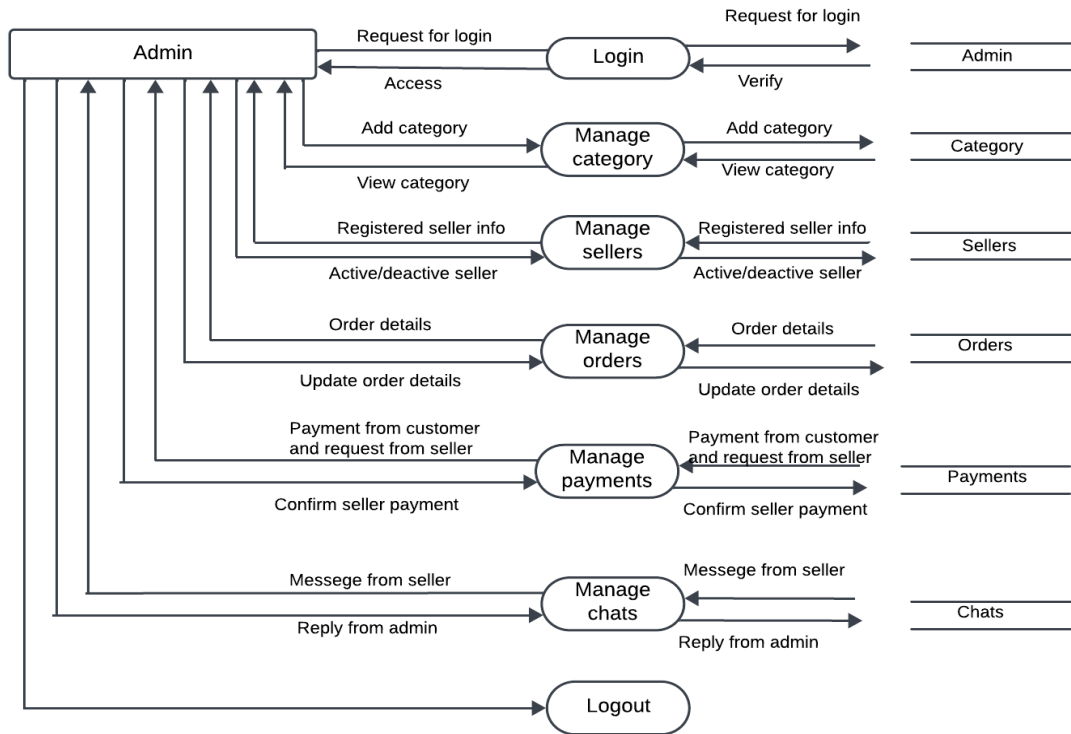


Fig.3.2.2.1 Admin Data Flow Diagram

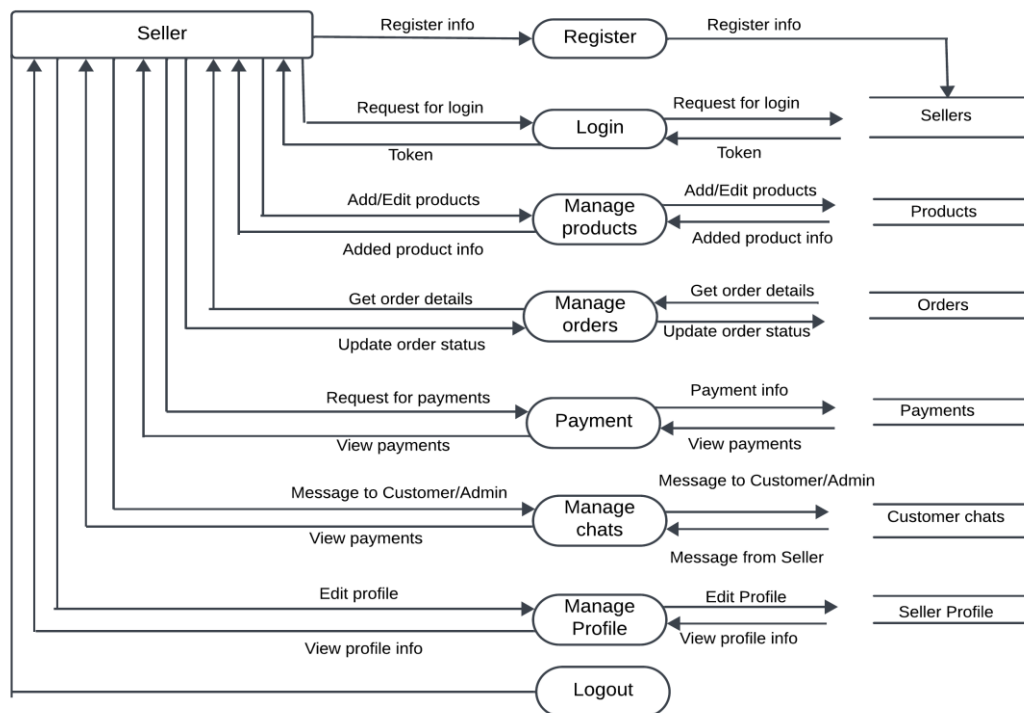


Fig.3.2.2.2 Seller Data Flow Diagram

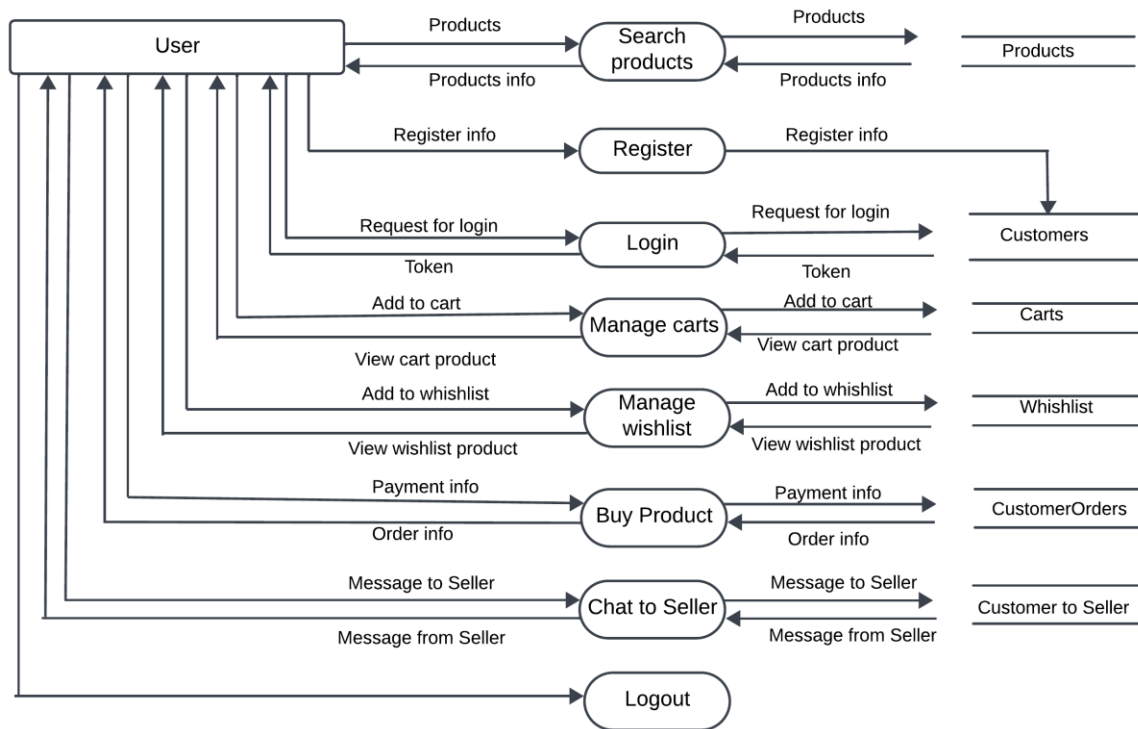


Fig.3.2.2.3 User Data Flow Diagram

3.2.3 Flow Chart

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task. A simple flowchart representing a process for dealing with a non-functioning lamp. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analysing, designing, documenting or managing a process or program in various fields.

Notation for Flowchart


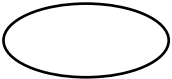

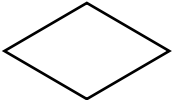
SYMBOL	PURPOSE	DESCRIPTION
	Flow line	Used to indicate flow of logic by connecting symbols.
	Terminal(start/stop)	Used to represent start and end of flowchart.
	Processing	Used for athematic operations and data manipulation.
	Decision	Used to represent the operation in which there are two alternatives, true and false.

Table.3.2.2.2 Notation for Flowchart

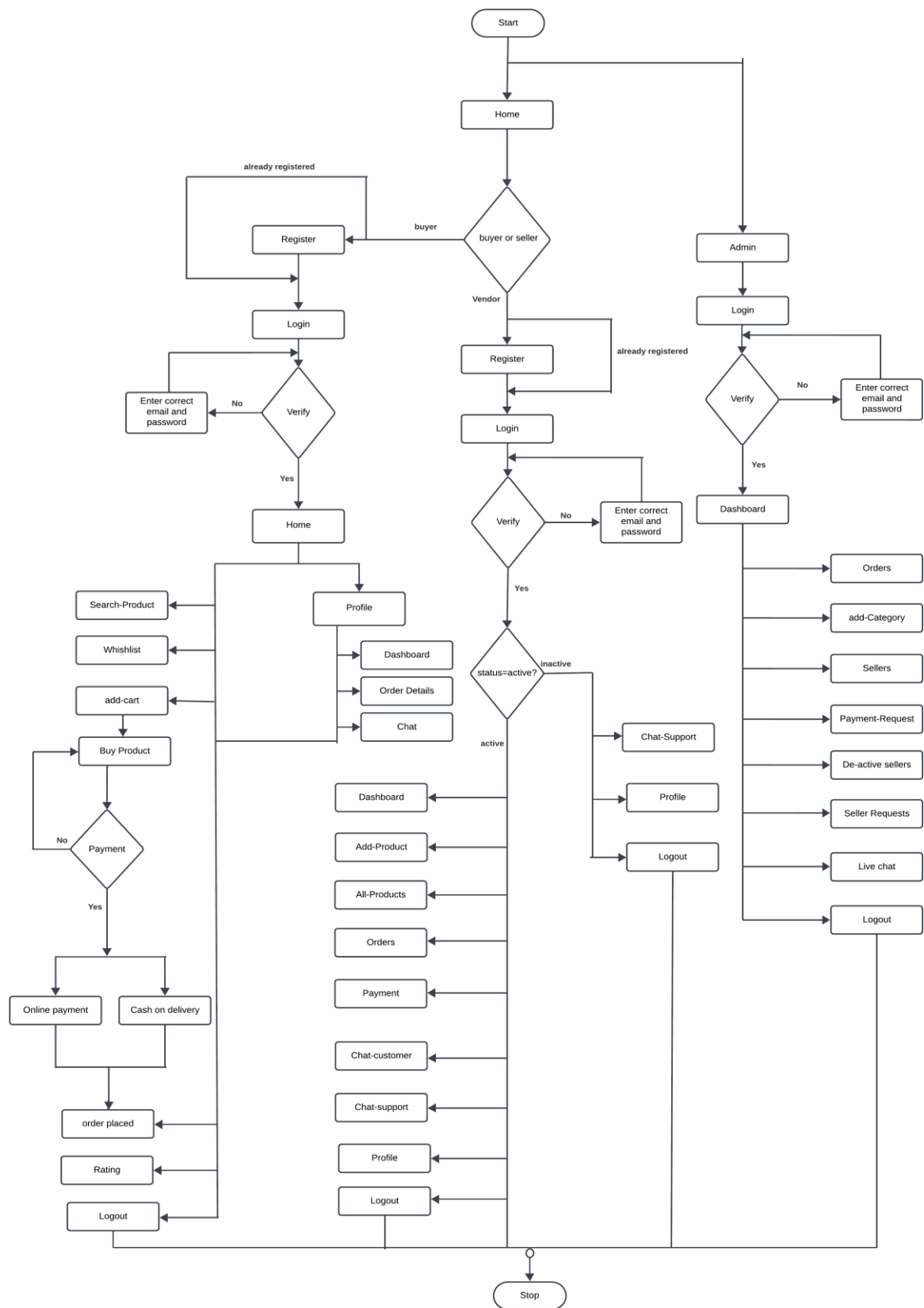



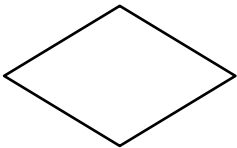
Fig.3.2.2.4 Flowchart

3.2.3 Entity Relationship (ER) Model

Entity Relationship model is popular high-level conceptual data model. This model and its variations are frequently used for the conceptual design of the database applications and many databases design tools employ its concepts. We described the basic data structuring concepts and constraints of the model and discuss their use in the design of conceptual; schemas for database application.

The basic units in an ER model are entities, their attributes, entity types and relationships between entity types.

- **Entities:** An entity represents an object defined within the information system about which you want to store information. An entity may be defined as a thing which is recognized as being capable of independent existence and which can be uniquely identified. An entity is an abstraction from the complexities of domain.
- **Entity Attributes:** Attributes are elementary pieces of information attached to entity. It is the entities where the number of attributes that can be used to describe them potentially vast, but the number that will actually apply to a given entity is relatively modest.
- **Relationships:** A relationship is a named connection or association between entities. A relationship captures how entities are related to one another.

Components	Description
	A rectangle or process symbol is generally used to represent entity.
	The diamond symbol is used to show relationships between entities.

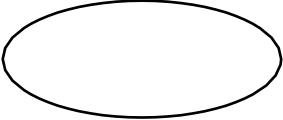
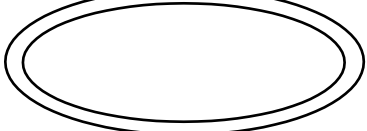
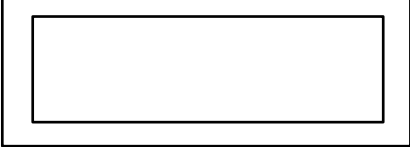
	<p>The terminator or oval symbol is used to define the attributes of the entity.</p>
	<p>The double oval symbol is used to define the multi valued attribute of the entity.</p>
	<p>Weak entity</p>

Table.3.2.3.1 Notation For E-R diagram

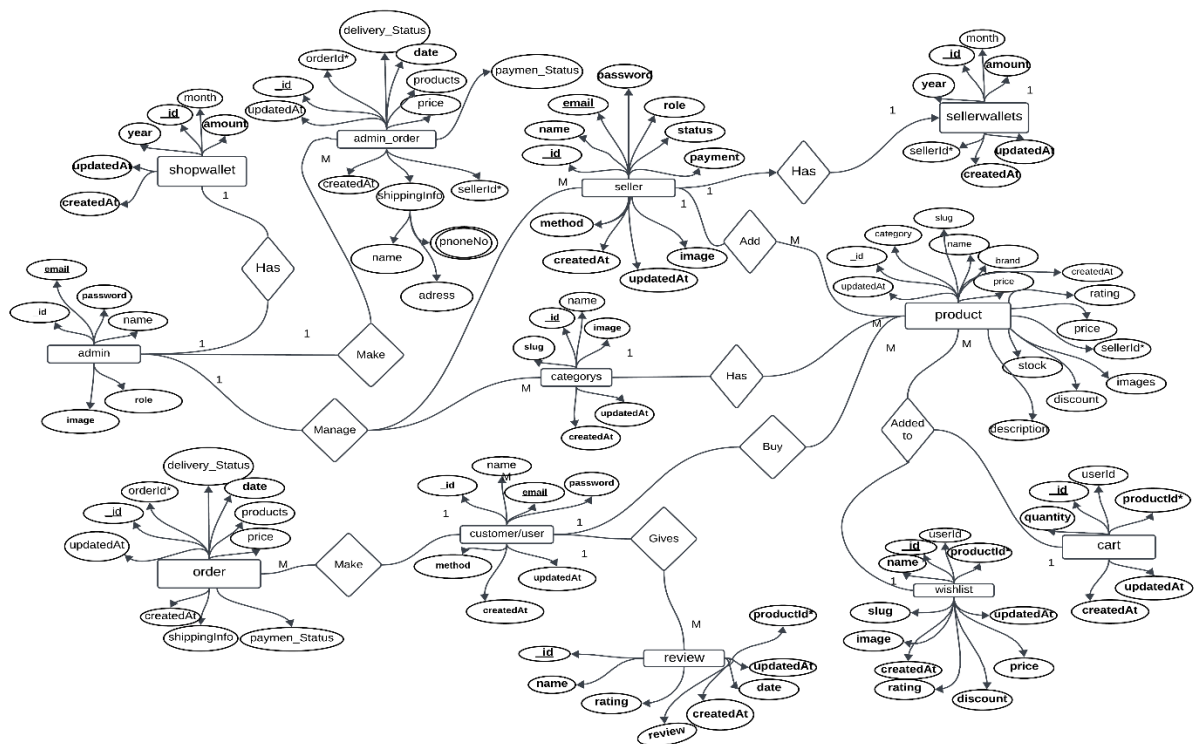


Fig.3.2.3.1 Entity Relationship Diagram

3.3 Database Design

3.3.1 Data Object Design

Database design is required to manage the large bodies of information. The management of data involves both the definition of structure of the storage of information and provisions of mechanism for the manipulation of information. In addition to the dB system must provide for the safety of information handled, despite the system crashes due to attempts at unauthorized access. For developing an efficient dB, we will have to fulfil certain condition such as:

- Control Redundancy
- Ease of use
- Data independence
- Accuracy and integrity

There are 6 major steps in designing process.

- Identify the table and relationship
- Identify the data that is need for each table and relationship.
- Resolve the relationship
- Verify the design
- Implement the design

3.3.2 Tables

The BizCart4all-Multivendor E-commerce Web Application uses following tables as fields:

adminModel

Field	Type	Required	Default
adminname	String	Yes	-
email	String	Yes	-
password	String	Yes	-
role	String	No	admin
createdAt	Date	No	Date. now
updatedAt	Date	No	Date. now

Table 3.3.2.1 Admin Model

sellerModel

Field	Type	Required	Default
name	String	Yes	-
email	String	Yes	-
password	String	Yes	-
role	String	No	seller
status	String	No	pending
payment	String	No	inactive
method	String	Yes	-
image	String	No	‘
shopinfo	Object	No	{}
createdAt	Date	No	Date. now
updatedAt	Date	No	Date. now

Table 3.3.2.2 Seller Model

customerModel

Field	Type	Required	Default
name	String	Yes	-
email	String	Yes	-
password	String	Yes	-
createdAt	Date	No	Date.now
updatedAt	Date	No	Date.now

Table 3.3.2.3 Customer Model

categoryModel

Field	Type	Required	Default
name	String	Yes	-
image	String	Yes	-
slug	String	Yes	-
createdAt	Date	No	Date.now
updatedAt	Date	No	Date.now

Table 3.3.2.4 Category Model

productModel

Field	Type	Required	Default
sellerId	Schema.ObjectId	Yes	-
name	String	Yes	-
slug	String	Yes	-
category	String	Yes	-
brand	String	Yes	-
price	Number	Yes	-
stock	Number	Yes	-
discount	Number	Yes	-
description	String	Yes	-
shopName	String	Yes	-
images	Array	Yes	-
rating	Number	No	0
createdAt	Date	No	Date.now
updatedAt	Date	No	Date.now

Table 3.3.2.5 Product Model

wishlistModel

Field	Type	Required	Default
userId	Schema.ObjectId	Yes	-
productId	String	Yes	-
name	String	Yes	-
price	Number	Yes	-
slug	String	Yes	-
discount	Number	Yes	-
image	String	Yes	-
rating	Number	No	0
createdAt	Date	No	Date.now
updatedAt	Date	No	Date.now

Table 3.3.2.6 Wishlist Model

cartModel

Field	Type	Required	Default
userId	Schema.ObjectId	Yes	-
productId	Schema.ObjectId	Yes	-
quantity	Number	Yes	-
createdAt	Date	No	Date.now
updatedAt	Date	No	Date.now

Table 3.3.2.7 Cart Model

stripeModel

Field	Type	Required	Default
sellerId	Schema.ObjectId	Yes	-
stripeId	String	Yes	-
code	String	Yes	-
createdAt	Date	No	Date.now
updatedAt	Date	No	Date.now

Table 3.3.2.8 Stripe Model

customerOrder

Field	Type	Required	Default
customerId	Schema.ObjectId	Yes	-
products	Array	Yes	-
price	Number	Yes	-
payment_status	String	Yes	-
shippingInfo	Object	Yes	-
delivery_status	String	Yes	-
date	String	Yes	-
createdAt	Date	No	Date.now
updatedAt	Date	No	Date.now

Table 3.3.2.9 CustomerOrder Model

A decorative scroll frame with a light gray background and a black outline. The frame has rounded corners and a vertical scroll on the left side. The text is centered within the frame.

CHAPTER-4

SYSTEM TESTING

4. TESTING

4.1 Introduction

Testing is the process of detecting errors. Testing performs a very special role for quality assurance and for ensuring the reliability of the software. Its basic function is to detect defects in software. Implementation is the process which tells the reliability, efficiency and flexibility. The result of testing is used later on during maintenance also. Testing is done to check whether the proposed system works as requested by the client. During testing is also check whether for the given input the expected output obtained or not. It also helps us in the rectification of errors in the system. There are different types of testing such as unit testing, integration testing, system testing. We used system testing and unit testing for testing the system.

4.1.1 Test Strategy

The testing strategy used will focus on reliability and performance of the product. This includes the reporting client hardware, software, management.

4.2 Types of Testing

“Program testing can be used to show the presence of bugs, but never to show their absence!” Clearly, the success of testing is revealing errors in program depend critically on the test cases.

The two basic approaches are

- 1.Black box or functional testing
- 2.White box or structural testing

4.2.1 Black box or functional testing

Black box testing is also known as functional testing. A software testing technique whereby the internal workings of the item being tested are not known by the tester. For example, in a black box test on software design the tester only known's the inputs and what the expected outcomes should be and not how the program arrives at those outputs. The testers ever

examine the programming code and do not need any further knowledge of the program other than its specifications.

4.2.2 box or structural testing

White box testing includes analysing data flow, control flow, information flow, coding practices and exception and error handling within the system, to test the intended and unintended software behaviour. White box testing can be performed to validate whether code implementation follows intended design, to validate implemented security functionality, and to uncover exploitable vulnerabilities. White box testing requires access to source code.

Though white box testing can be performed any time in the life cycle after the code is developed; it is a good practice to perform White box testing during the unit testing phase.

4.3 Different Levels of testing

During testing process different levels of testing are used. Each level of testing aiming to test different aspects of the system. The three different levels of testing are,

4.3.1 Unit testing

Unit testing focuses verification efforts on the smallest unit of software i.e, the module. Using detailed design and the process specification testing is done to uncover errors within the boundary of the module. All modules must be successful in the unit test before the start of the integration testing begins. Here different modules are tested against the specification produced during design for the module. It is necessary to verify the code written during the coding phase.

4.3.2 Integration testing

After the unit testing, we have to perform the integration testing. The goal here is to see if modules can be integrated properly, the emphasis being on testing interfaces between modules. This testing activity can be considered as testing the design and hence emphasis on testing module interaction. During this testing many unit tested modules are combined into sub systems which are then tested.

The following are the types of Integration Testing:

- Top-Down Testing
- Bottom-Up Testing

Top-Down Testing

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinated to the main program module is incorporated into the structure in either a depth first or breadth first manner.

Bottom-Up Testing

This method begins the construction and testing the modules at the lowest level in the program structure. Since the modules are integrated from the bottom-up processing required for modules subordinated to a given level is always available.

The bottom-up integration strategies may be implemented with the following steps:

- The low-level modules are combined into clusters that perform a specific software sub-function.
- A driver (i.e.) the control program for testing is written to coordinate test case.
- Input and output.
- The cluster is tested.
- Drivers are modulated and clusters are combined moving upward in the program structure.

4.3.3 System testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test.

4.3.4 Acceptance testing

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

4.4 Software Test Report

Test Case

Sl. No	Test Case	Expected Result	Obtained Result
1	Empty field in required Textbox	Enter some values in textbox	Successful

Table. 4.4.1 Empty Textbox Validation for Buyer

The screenshot displays the BizCart4all registration interface. At the top, there is a navigation bar with contact information (bizcart4all@gmail.com, +91 7019115360) and social media links. Below this is a header with the BizCart4all logo and navigation links (HOME, VENDORS, SHOP, ABOUT US, CONTACT US). A green sidebar on the left contains a category menu. The main content area features a registration form with fields for Name, Email, and Password. The Email field has a validation error message: "Please fill in this field." To the right of the form is an illustration of a person signing up on a large screen. Below the form, there is a link to "Login" for existing users.

Fig.4.4.1 Empty Text Box Validation for Buyer

Sl. No	Test Case	Expected Result	Obtained Result
1	Empty field in required Textbox	Enter some values in textbox	Successful

Table. 4.4.2 Empty Textbox Validation for Seller

The screenshot shows the 'Profile' page for a seller named 'Seller10'. The page has a sidebar with navigation links: Dashboard, Add Product, All Product, Orders, Payments, Chat-Customer, Chat-Support, Profile (selected), and Logout. The main content area is titled 'Status: pending' and 'Payment Account: Click Active'. Below this, there is a form with the following fields and validation messages:

- Shop Name: Shop Name is required
- Phone Number: Phone Number is required
- GST Number: GST Number is required
- Shop Address: Address is required
- State: State Name is required
- District: District is required

At the bottom of the form is a 'Submit' button.

Fig.4.4.2 Empty Text Box Validation for Seller

Sl. No	Test Case	Expected Result	Obtained Result
2	Email validation	Valid Email	Successful

Table. 4.4.3 Email Validation

The screenshot shows the BizCart4all website interface. At the top, there is a header with the logo, navigation links (HOME, VENDORS, SHOP, ABOUT US, CONTACT US), and a search bar. Below the header, there is a 'Login' section with the following fields and buttons:

- Email: pratham@gmail.com
- Password: (masked with asterisks)
- LOGIN button
- Forgot Password? link
- Don't Have An Account? Register link
- LOGIN AS A SELLER button
- REGISTER AS A SELLER button

On the left side of the login section, there is an illustration of a person sitting at a desk with a laptop, a lightbulb, and gears.

Fig 4.4.3 Email Validation

Sl. No	Test Case	Expected Result	Obtained Result
3	Password	Password didn't match	Successful

Table. 4.4.4 Password Validation

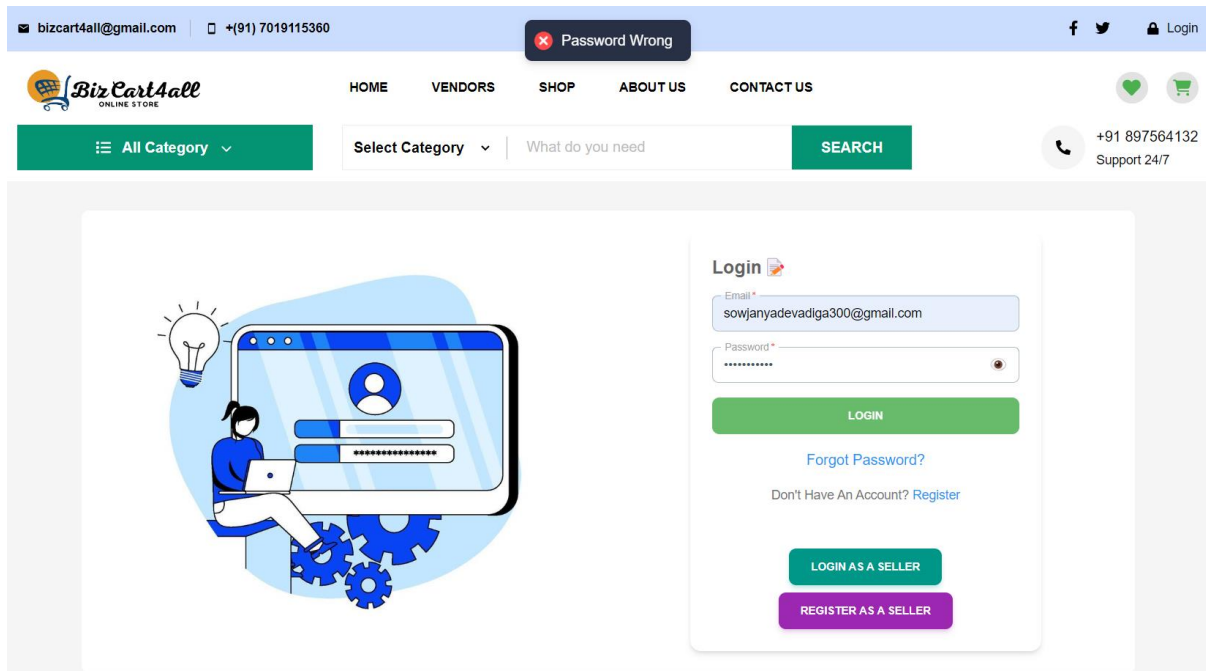


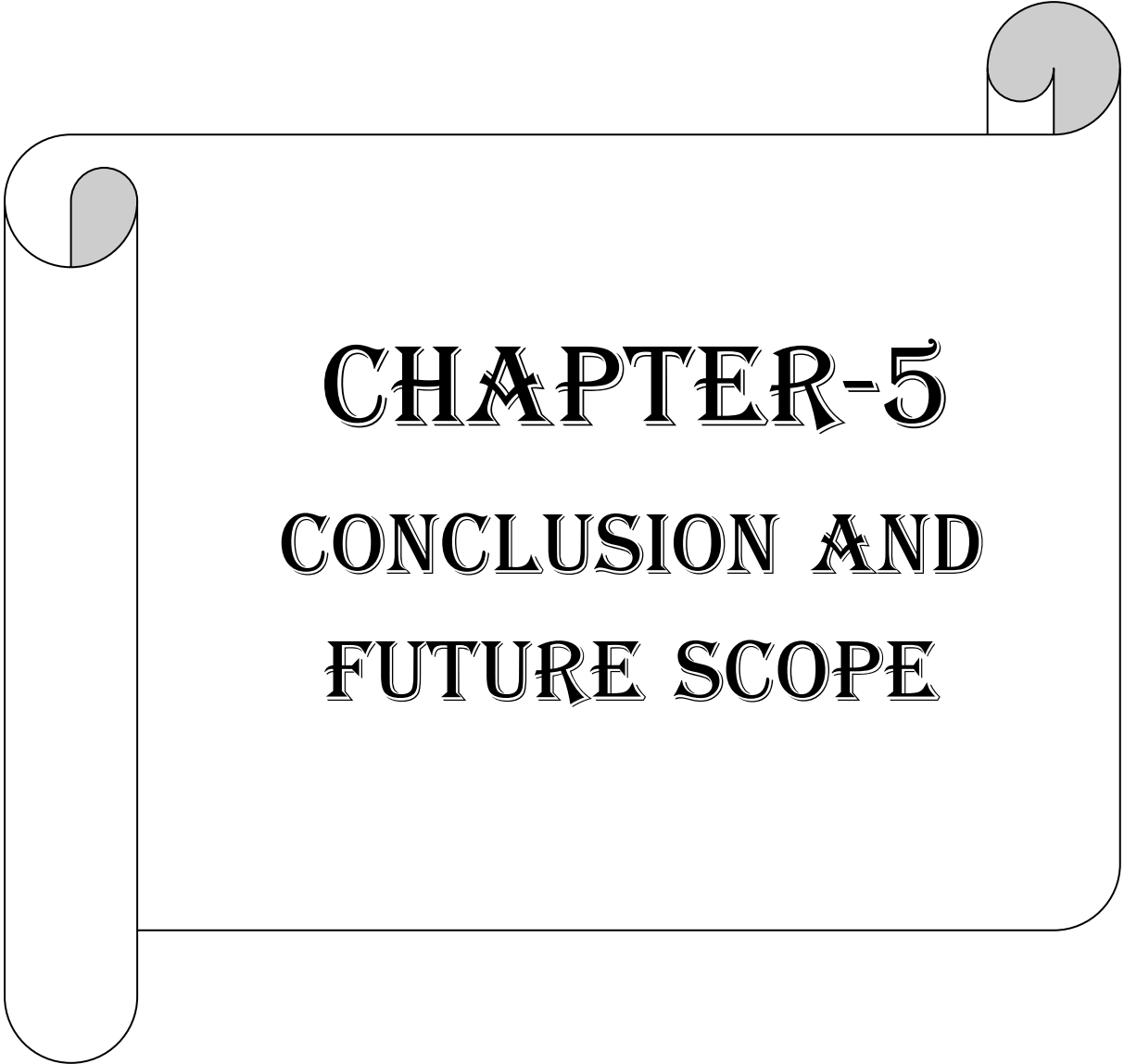
Fig 4.4.4 Password Validation

Sl. No	Test Case	Expected Result	Obtained Result
4	Password length less than 6 character	Must be minimum 6 character	Successful

Table. 4.4.5 Password Length Validation

The screenshot displays the registration interface of the BizCart4all website. At the top, a navigation bar includes contact information (email: bizcart4all@gmail.com, phone: +91 7019115360) and social media links. Below this, a header section features the BizCart4all logo, a menu (HOME, VENDORS, SHOP, ABOUT US, CONTACT US), and a search bar. The main content area is titled 'WELCOME TO BIZCART4ALL' and contains a 'Register' form. The form fields are: Name (filled with 'Sneha'), Email (filled with 'sneha@gmail.com'), and Password (masked with '*****'). A red error message below the password field states: 'Password must contain at least one uppercase letter, one special character, one number, and be at least 6 characters long'. A green 'Register' button is positioned below the form. To the right of the form is an illustration of a person signing up on a tablet. At the bottom of the form, there is a link to 'Login' for existing users.

Fig 4.4.5 Password Length and Pattern Validation

A decorative scrollwork frame surrounds the chapter title. It features a large, light gray scroll on the left side and a smaller, light gray scroll on the top right corner.

CHAPTER-5

CONCLUSION AND FUTURE SCOPE

6. CONCLUSION & FUTURE SCOPE

6.1 CONCLUSION

“BizCart4all”- Comprehensive Seller Services revolutionizes the multi-vendor e-commerce experience by addressing key challenges like inefficient database management and limited seller autonomy. It offers a user-friendly and secure platform for administrators, sellers, and buyers. With features such as advanced search and filtering options, robust security measures, and a responsive interface built with React.js, MongoDB, Redux, and Tailwind CSS, BizCart4all ensures a seamless and efficient shopping experience.

6.2 FUTURE ENHANCEMENT

To enhance the user experience on our e-commerce platform, we plan to integrate advanced image recognition technology to improve product listings, making it easier for sellers to upload accurate and visually appealing product images. Additionally, we aim to implement voice command functionality to provide users with a more seamless and hands-free interaction experience. To further personalize the shopping experience, we will introduce AI-driven recommendations that will suggest products based on user preferences and browsing history, ensuring a more tailored and engaging shopping journey.

A decorative border resembling a scroll, with a vertical strip on the left and a horizontal strip at the top, both featuring rounded ends and a grey shaded area.

REFERENCES

REFERENCES

- [1]. [Fashion Trade Show & Expo in Hong Kong | Global Sources \(2024\)](#)
- [2]. [Multi-vendor eCommerce Marketplace Development Guide - CSSChopper](#)
- [3]. [Top 15 Multi-Vendor Marketplace Platforms for Ecommerce 2024 \(cloudways.com\)](#)
- [4]. <https://tailwindcss.com>
- [5]. <https://legacy.reactjs.org/docs/getting-started.html>
- [6]. <https://www.w3schools.com/REACT/DEFAULT.ASP>
- [7]. <https://www.mongodb.com/docs/atlas/>
- [8]. Anti-Hacker Tool Kit 4th Edition by Mike Schema
- [9]. Software Engineering – A Practitioners approach 7th edition by Roger S. Pressman.



APPENDIX

Buyer

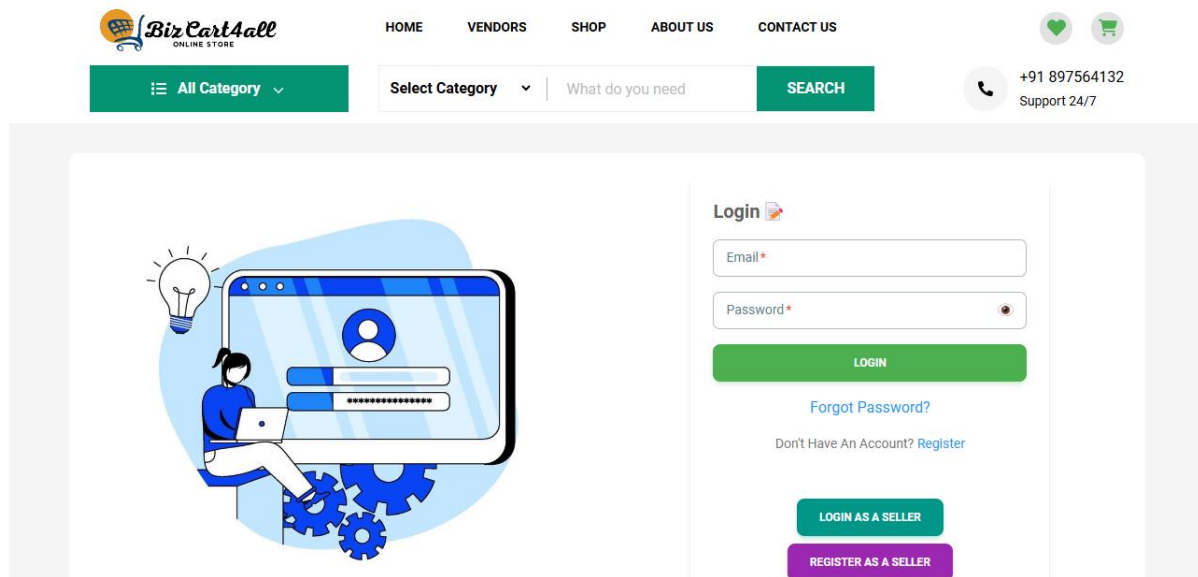


Fig.1 Buyer Login Page

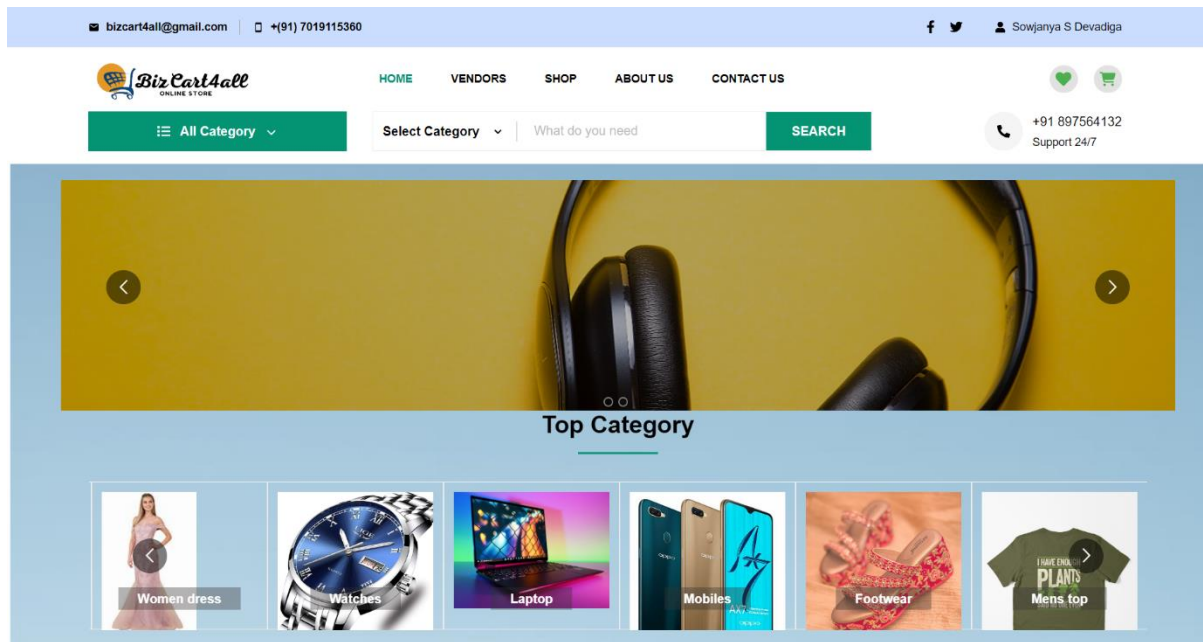


Fig. 2 Home Page

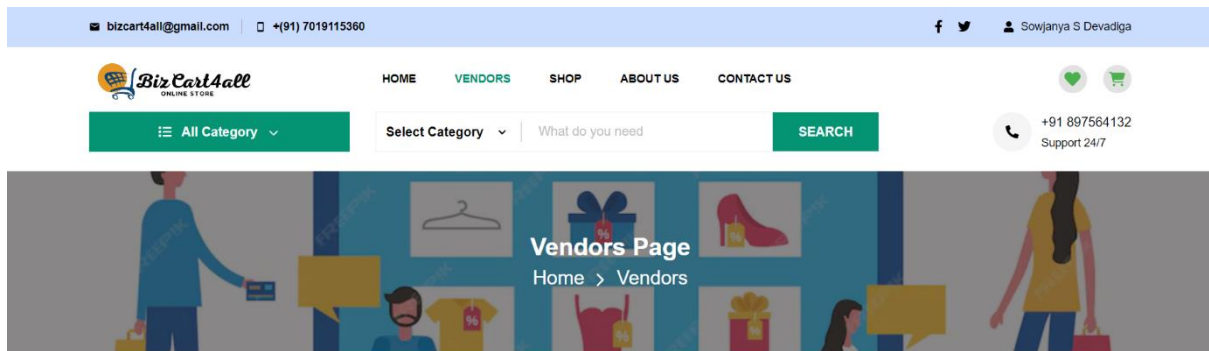


Fig. 3 Vendor Page

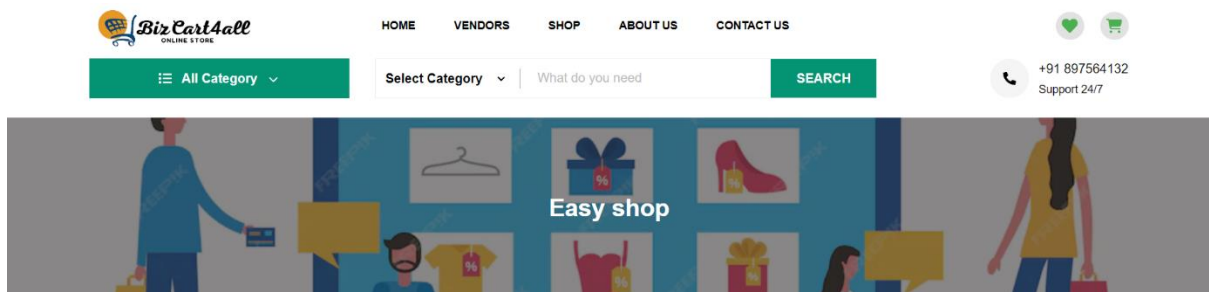


Fig. 4 Vendor Based Product Page

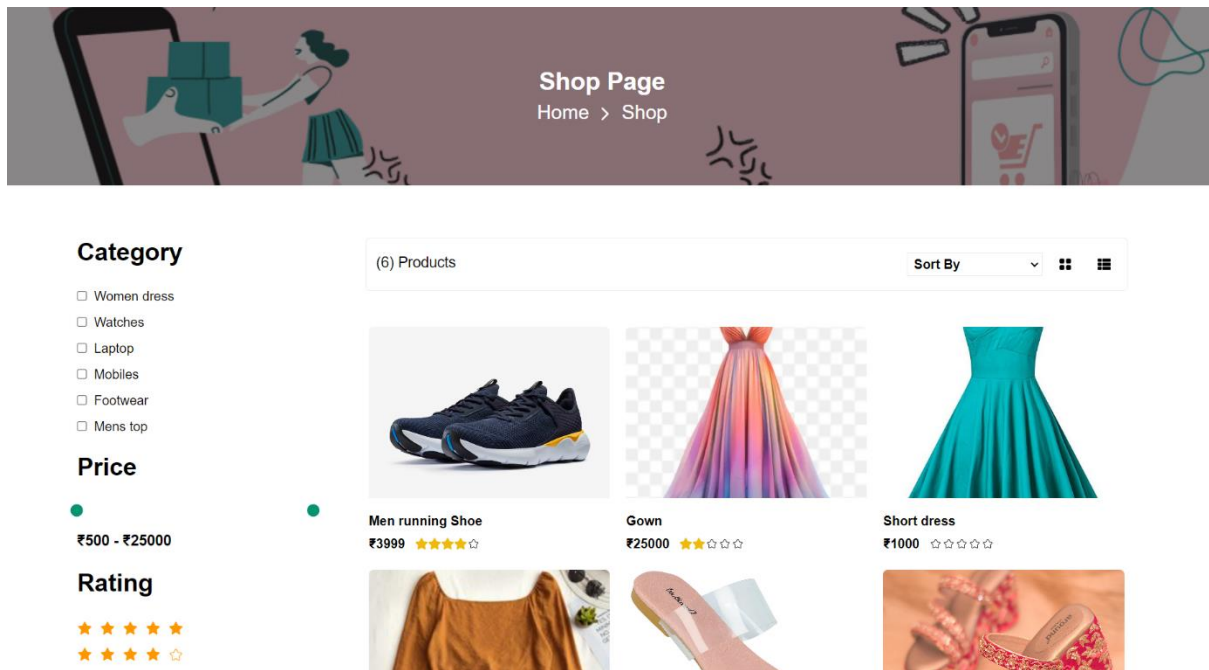


Fig. 5 Shop Page

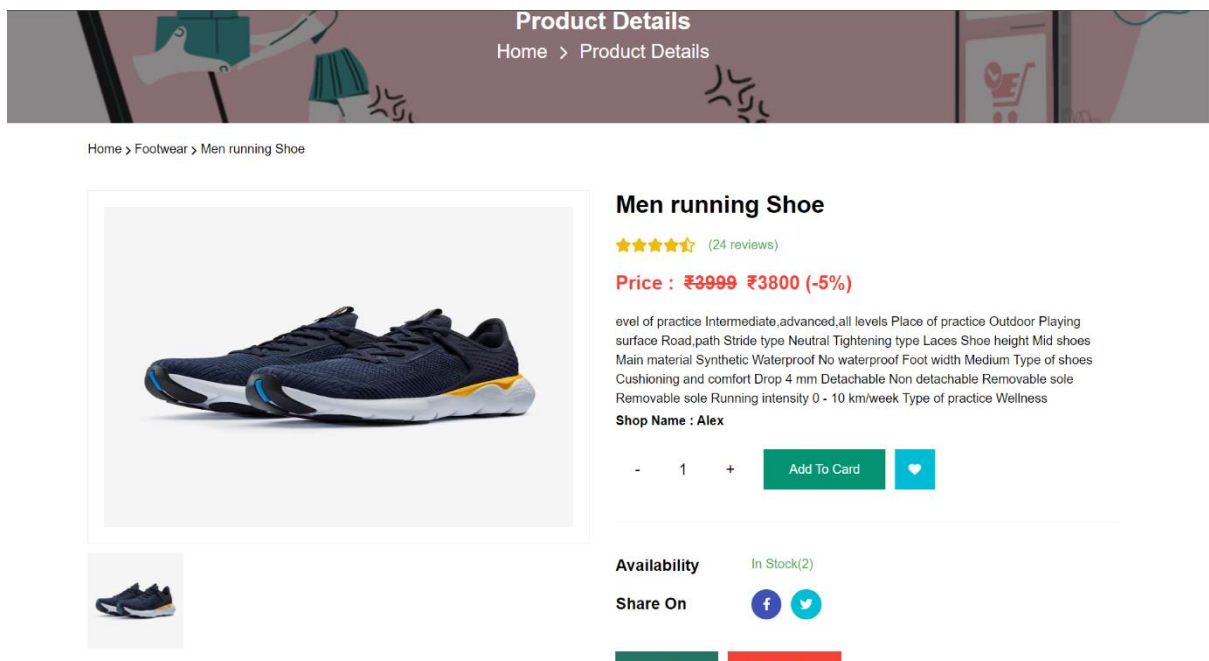
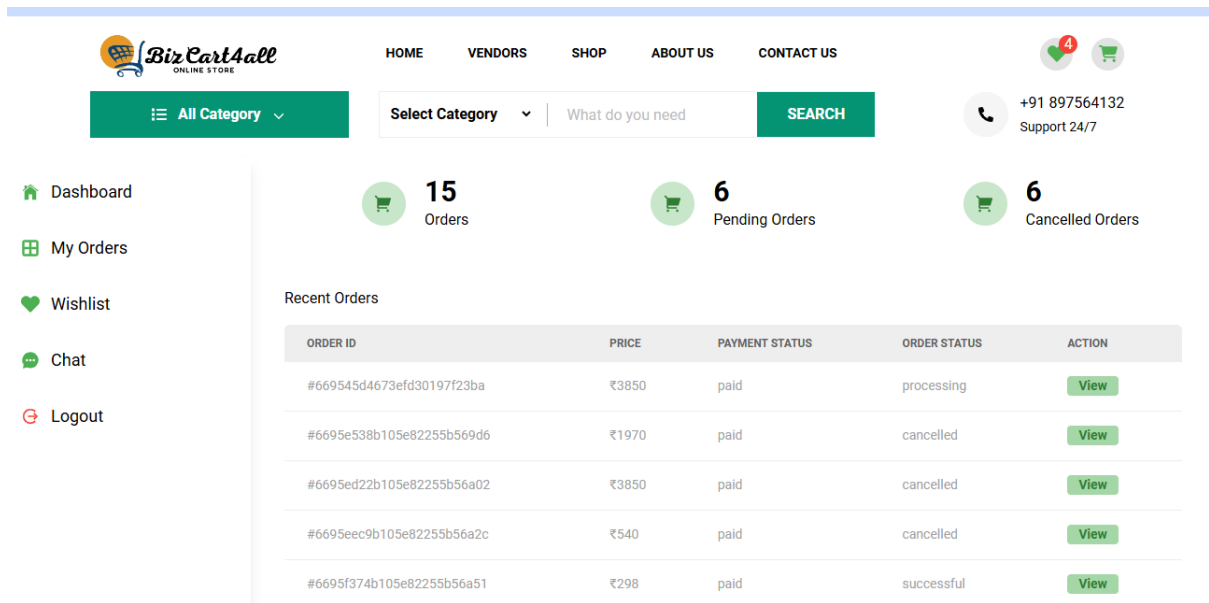


Fig. 6 Product Details Page



The screenshot shows the user dashboard of the BizCart4all website. The top navigation bar includes links for HOME, VENDORS, SHOP, ABOUT US, and CONTACT US. A search bar is present with a dropdown for 'Select Category' and a 'SEARCH' button. A sidebar on the left contains links for Dashboard, My Orders, Wishlist, Chat, and Logout. The main content area displays three order statistics: 15 Orders, 6 Pending Orders, and 6 Cancelled Orders. Below these is a 'Recent Orders' table with columns for ORDER ID, PRICE, PAYMENT STATUS, ORDER STATUS, and ACTION.

ORDER ID	PRICE	PAYMENT STATUS	ORDER STATUS	ACTION
#669545d4673efd30197f23ba	₹3850	paid	processing	View
#6695e538b105e82255b569d6	₹1970	paid	cancelled	View
#6695ed22b105e82255b56a02	₹3850	paid	cancelled	View
#6695ec9b105e82255b56a2c	₹540	paid	cancelled	View
#6695f374b105e82255b56a51	₹298	paid	successful	View

Fig. 7 User Dashboard

Seller

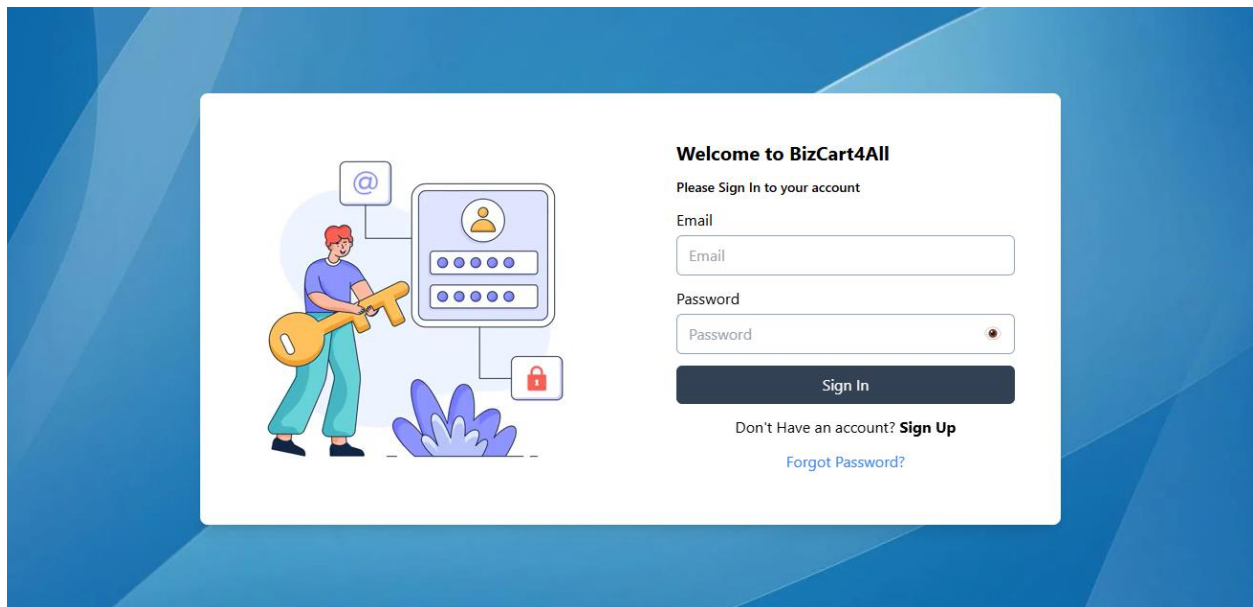


Fig. 8 Seller Login Page

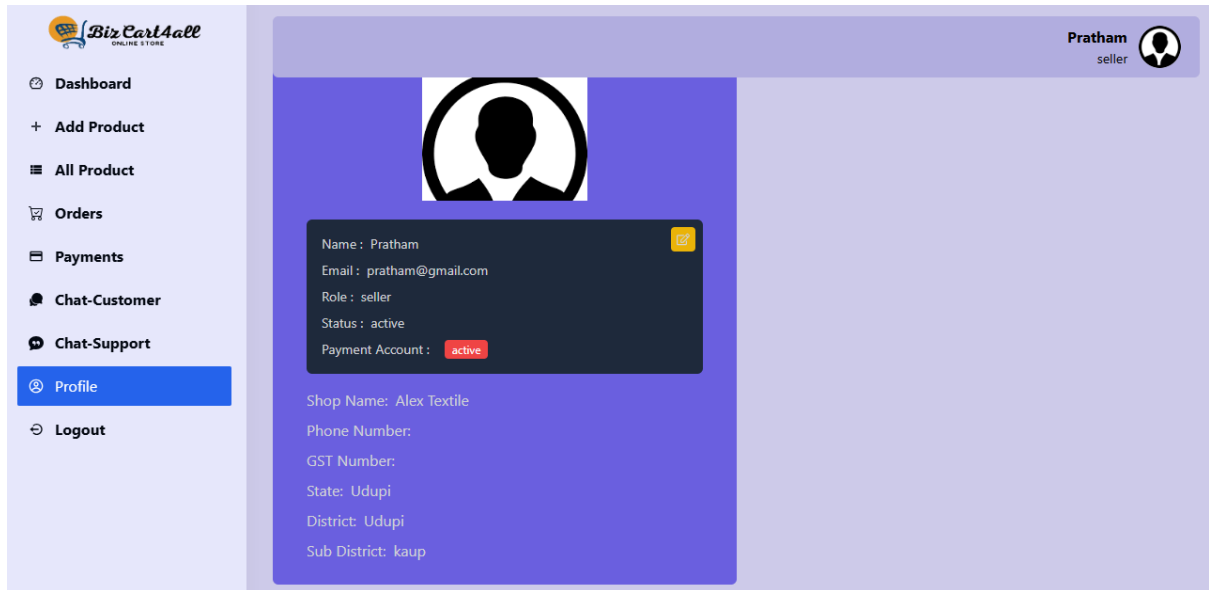


Fig. 9 Seller Profile Page



Fig. 10 Seller Dashboard

BizCart4all
ONLINE STORE

Pratham
seller

Add Product

Product Name:

Product Brand:

Category:

Product Stock:

Price:

Discount:

Description:

Fig. 11 Add Product

BizCart4all
ONLINE STORE

Pratham
seller

All Products

5






NO	IMAGE	NAME	CATEGORY	BRAND	PRICE	DISCOUNT	STOCK	ACTION
1		Men running Sho...	Footwear	Kalenji	₹3999	%5	2	   

Fig. 12 View and Update Products

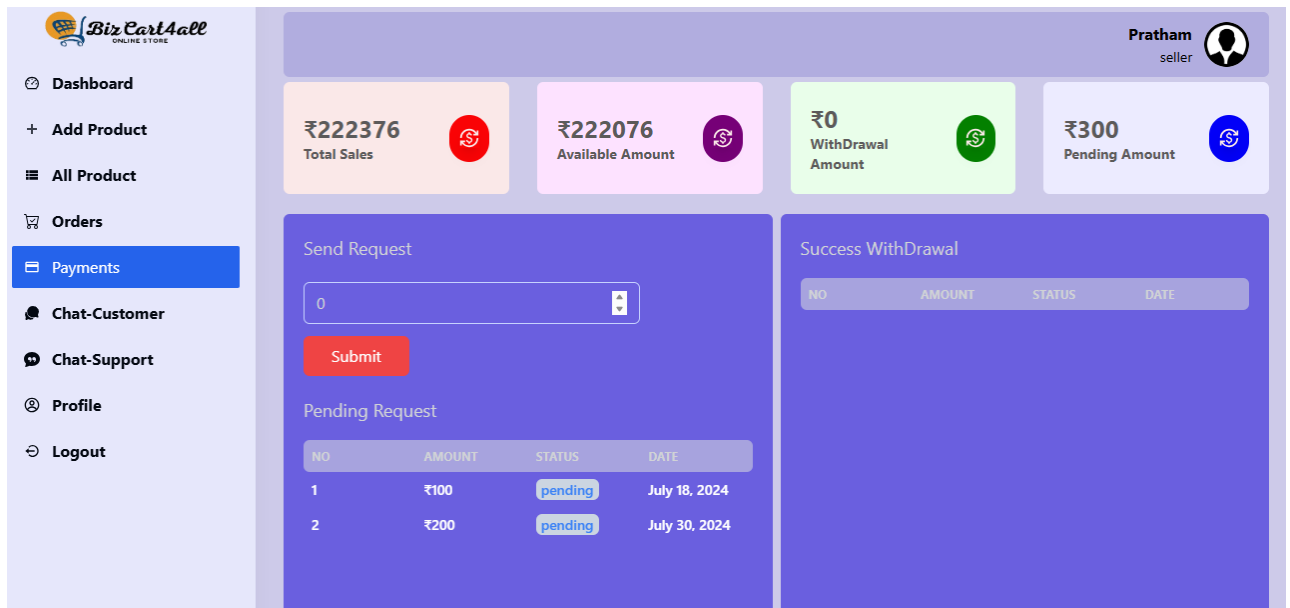


Fig .13 Send Payment Request

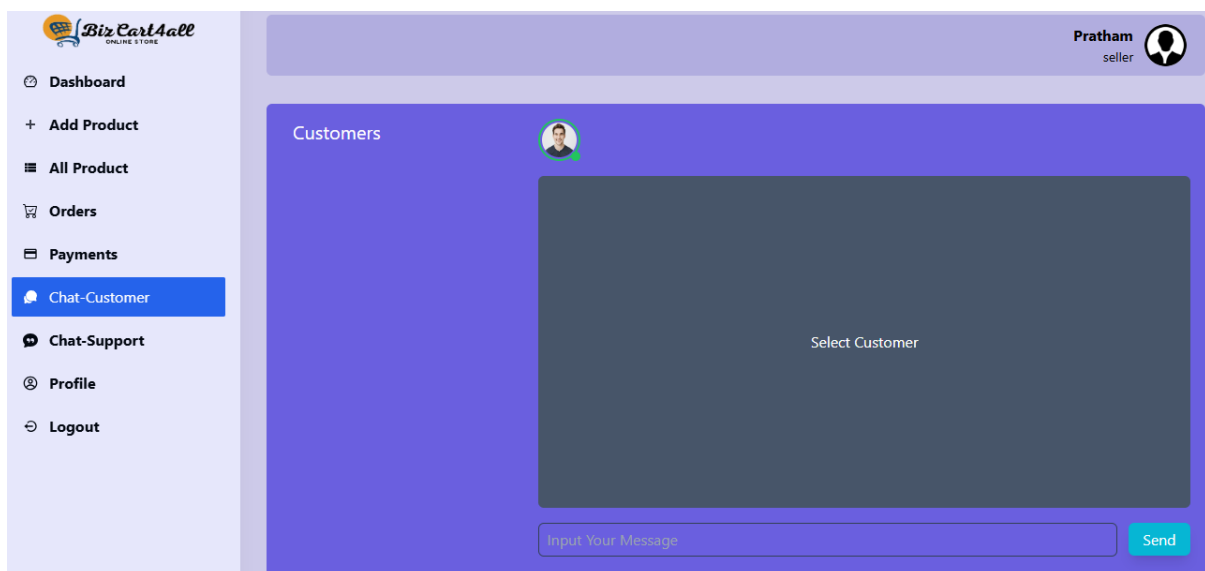


Fig. 14 Chat to Customer

Admin

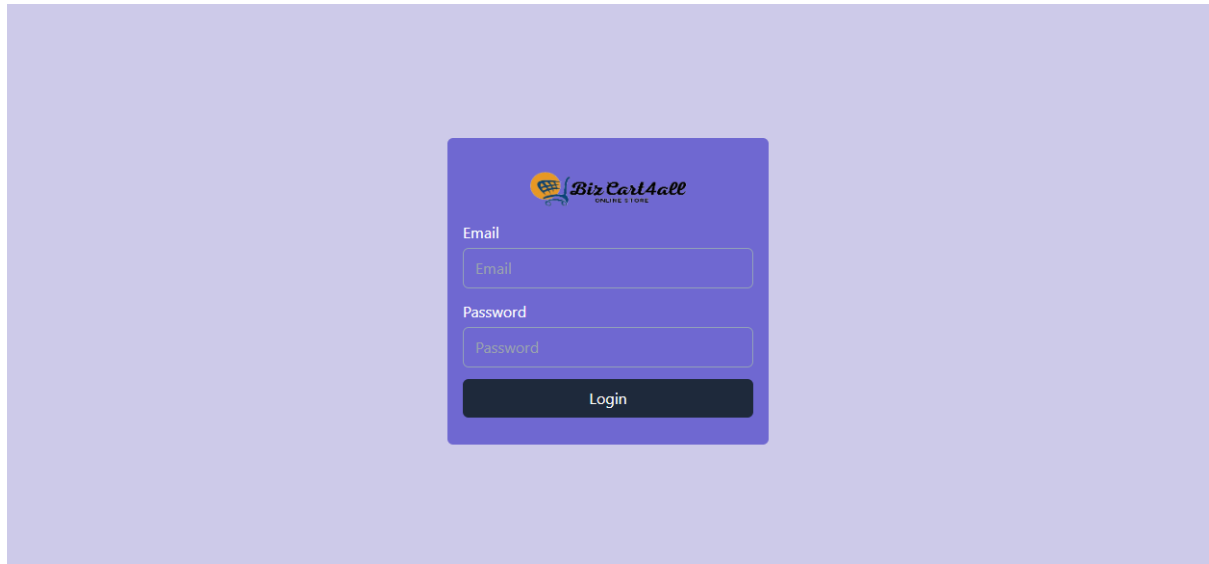


Fig. 15 Admin Login

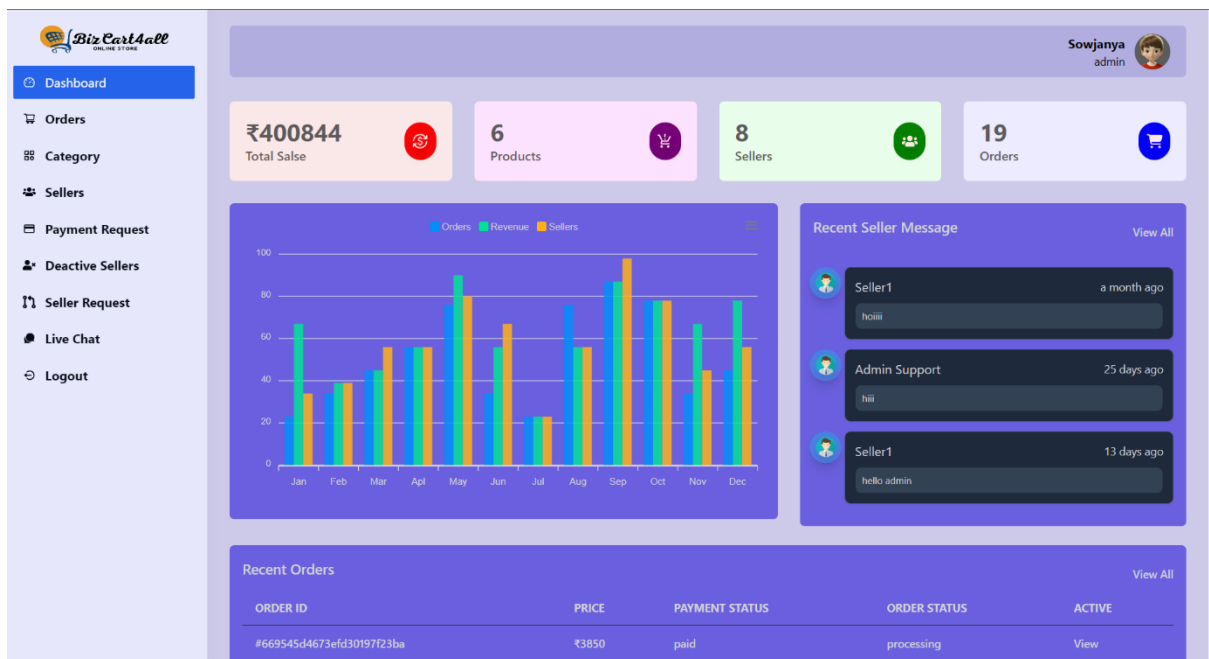


Fig .16 Admin Dashboard

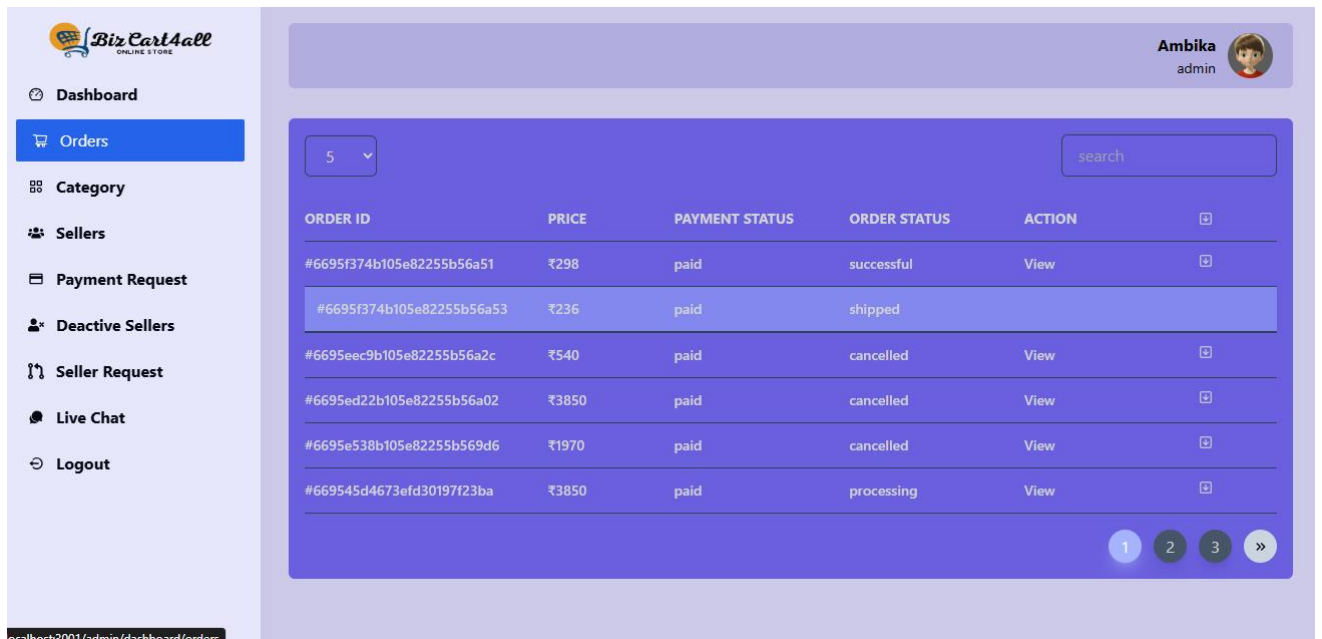


Fig. 17 Admin Orders

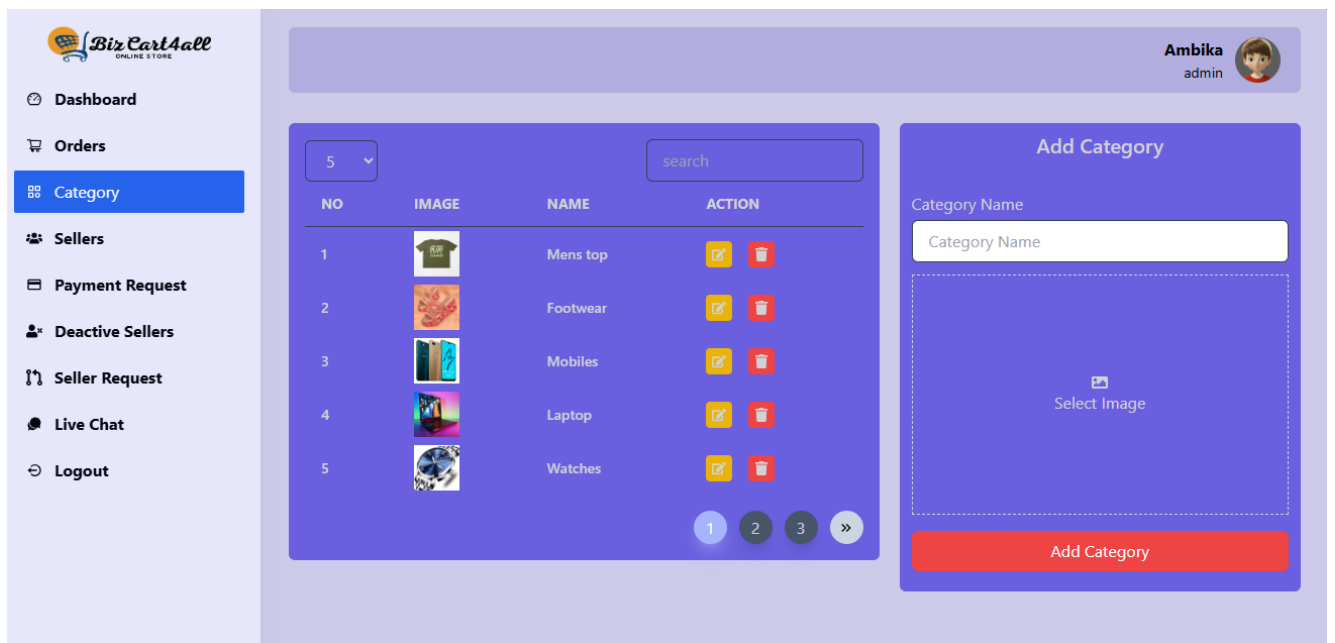


Fig. 18 Add Category

Seller Request

5

NO	NAME	EMAIL	PAYMENT STATUS	STATUS	ACTION
1	Seller10	seller10@gmail.com	inactive	pending	
2	seller6	seller7@gmail.com	inactive	pending	
3	seller6	seller6@gmail.com	inactive	pending	

1 2 3 >>

localhost:3001/admin/dashboard

Fig .19 Seller Request Page

Sellers

- Seller1
- Pratham
- seller5
- seller6
- seller6
- Sudeep
- Seller10

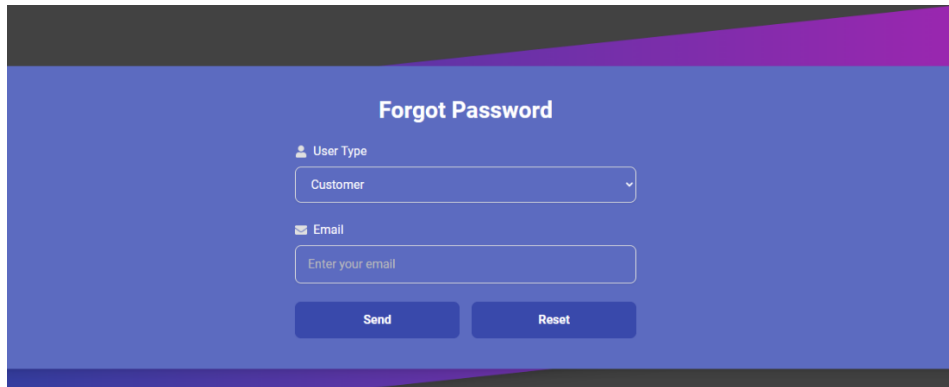
Seller1

hoiiii

hello admin

hiii

Fig. 20 Live Chat Admin to Seller



The image shows a 'Forgot Password' form on a blue background. At the top, the title 'Forgot Password' is centered. Below it, there is a 'User Type' dropdown menu with 'Customer' selected. Underneath is an 'Email' input field with the placeholder text 'Enter your email'. At the bottom, there are two buttons: 'Send' and 'Reset'.

Fig .21 Forgot Password

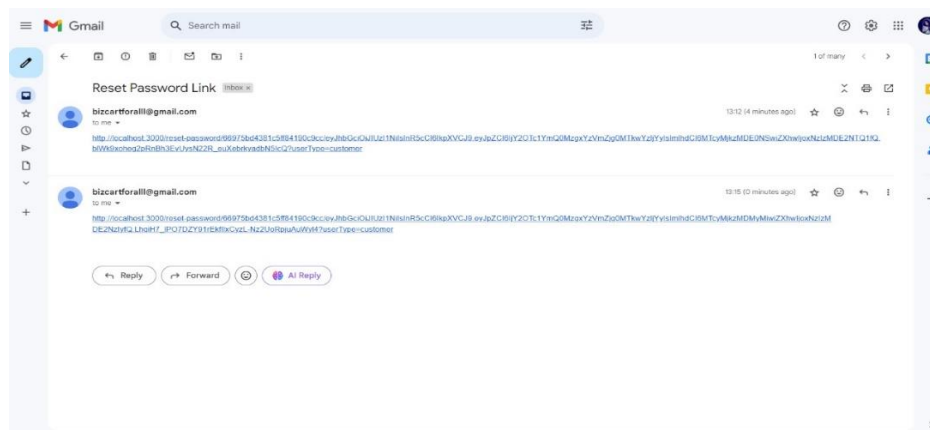
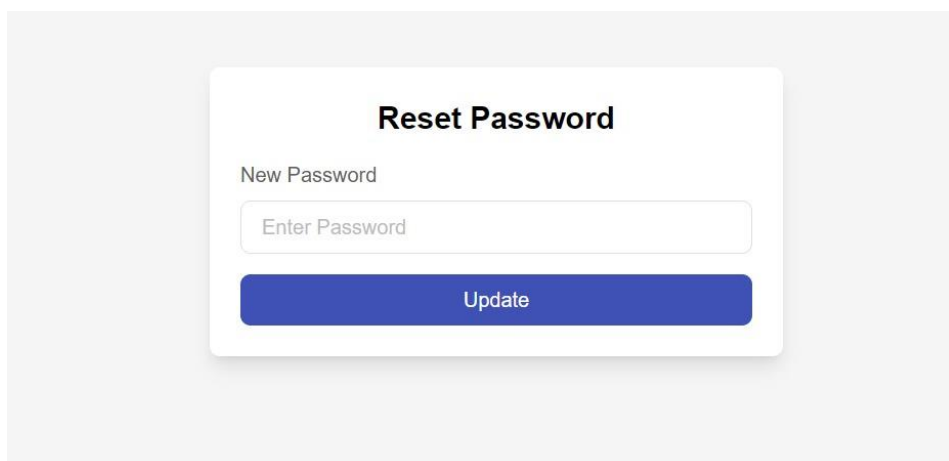


Fig 21.1 Forgot Password Link Sent to Email



The image shows a 'Reset Password' form on a light gray background. The title 'Reset Password' is centered at the top. Below it, there is a 'New Password' label and an input field with the placeholder text 'Enter Password'. At the bottom, there is a blue 'Update' button.

Fig. 21.2 Reset Password