** GCUF

GOVERNMENT COLLEGE UNIVERSITY FAISALABAD

FINAL REPORT

ExpenseHub

(Expense Management System)

**Submitted By**

**Shiza Malik**

**2020-GCUF-3215/459875**



**DEPARTMENT OF COMPUTER SCIENCE**

**Govt. College of Science Samanabad, Faisalabad**

**2024**

# DECLARATION

I hereby declare that the content of the project, **“ShopSmart Wiki”** are product of my own research and no part has been copied from any published source (except the references, some standard mathematical or genetic models / equations / protocols etc.). I further declare that this work has not been submitted for award of any other diploma/degree. The university can take action if the above statement is found inaccurate at any stage.

**Signature**

**AnzalaAdnan**

**2020-GCUF-3215/459875** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**CERTIFICATE BY SUPERVISORY COMMITTEE**

We certify that the contents and the form of project submitted by **Shiza Malik Roll # 3215/459875**, have been found satisfactory and recommend it to be processed for evaluation by the External Examiner(s) for the award of degree.

**Supervisors:**

**Faiza Batool Signature** : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Internal Panel**

**Member 1: Name Signature** : \_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Member 2: Name Signature** : \_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Member 3: Name Signature** : \_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Project Coordinator Head of Department**

**Principal**

# DEDICATION

***To My Beloved Parents***

&

***Teachers***

**ACKNOWLEDGEMENT**

All praise to ALMIGHTY ALLAH, the most merciful and the most compassionate and his Holy Prophet .MUHAMMAD'(Peace) be upon him) the most perfect and exalted among and even bon) on the surface of earth, who is, forever a torch of guidance and knowledge for the humanity as a whole.

The work presented in this manuscript was accomplished under the inspiring guidance, gorgeous assistance, constructive criticism and enlightened supervision of **Miss. Faiza Batool** Department of Computer Science in Govt. College of Science Samanabad, Faisalabad for his skillful guidance, constructive criticism, masterly advice, valuable suggestions and sympathetic behavior for the completion of this manuscript

We feel highly privileged to take this opportunity to express one heartiest gratitude and deep sense of indebt to our worthy supervisory committee, **Miss. Faiza Batool** of Computer Science Govt. College of Science Samanabad, Faisalabad under whose kind and scholastic guidance, keen interest and constant encouragement.

Words are very important to convey thoughts and thanks, the words are impossible to find thank our Father and whole family for their prayers and encouragement for us and for our work.

Finally, we are apologize if we have caused anger of offence to anybody and the errors that remain in the manuscript are mine alone.

Anzala Adnan

# Abstract

The **ShopSmart Wiki** project represents a comprehensive e-commerce platform designed to provide a practical learning experience in developing interactive online shopping systems. This project integrates modern client-side technologies, such as **React.js** and **HTML**, with server-side operations using **Node.js** and Express to create a fully functional, secure, and scalable e-commerce application.

The primary objective of the ShopSmart Wiki is to educate students on building and managing e-commerce applications from scratch. The client-side, developed with React.js, HTML, and CSS, offers a dynamic and responsive user interface, ensuring a seamless shopping experience. The server-side, utilizing Node.js, handles critical operations such as database management, session handling, transaction processing, and the integration of secure payment systems.

Key features of the ShopSmart Wiki include a dynamic product catalog, user registration and authentication, shopping cart management, and real-time updates. The system also supports essential functionalities like calculating shipping costs, sales tax, and order processing. Future enhancements may include local shop integration, improved delivery systems, and additional user engagement features like chat support.

This documentation provides a detailed analysis of the project’s architecture, functional and non-functional requirements, and potential future developments. By emphasizing practical application and user experience, the project aims to bridge theoretical knowledge with real-world e-commerce solutions.

Table of Contents

[DECLARATION ii](#_Toc168799162)

[DEDICATION iv](#_Toc168799163)

[ACKNOWLEDGEMENT v](#_Toc168799164)

[Chapter 1 1](#_Toc168799165)

[INTRODUCTION TO THE PROBLEM 1](#_Toc168799166)

[1.3 Background Study 4](#_Toc168799167)

[1.4 Purpose 6](#_Toc168799168)

[1.5 Scope 7](#_Toc168799169)

[1.6 Objective 7](#_Toc168799170)

[1.7 Intended Audience and Reading Suggestions 7](#_Toc168799171)

[1.8 Documentation Conventions 9](#_Toc168799172)

[Chapter 2 10](#_Toc168799173)

[SOFTWARE REQUIREMENT SPECIFICATION 10](#_Toc168799174)

[2.1 User Registration Requirements 10](#_Toc168799175)

[2.2 External Interface Requirements 11](#_Toc168799176)

[2.3 Other Non-Functional Requirements 18](#_Toc168799177)

[2.2.1 Performance Requirements 18](#_Toc168799178)

[2.1.2 Safety Requirements 18](#_Toc168799179)

[2.4 Architecture Diagram 21](#_Toc168799180)

[2.5 Entity-Relationship Diagram (ERD) 22](#_Toc168799181)

[2.6 Data Flow Diagram (DFD) 24](#_Toc168799182)

[2.7 Class Diagram 27](#_Toc168799182)

[2.8 Object Diagram 29](#_Toc168799183)

[2.9 Sequence Diagram 31](#_Toc168799184)

[2.10 Activity Diagram 31](#_Toc168799185)

[2.11 Collaboration Diagram 32](#_Toc168799186)

[2.12 State Transition Diagram 33](#_Toc168799187)

[Chapter 3 35](#_Toc168799188)

[ANALYSIS (USE CASE MODEL) 35](#_Toc168799189)

[3.1 Identifying Actors and Use Cases using Textual Analysis 35](#_Toc168799190)

[3.2 Forming Use Case Diagram with Candidate and Use Cases 36](#_Toc168799191)

[3.3 Describe the Events Flow for Use Case 37](#_Toc168799192)

[Chapter 4 40](#_Toc168799193)

[DESIGN 40](#_Toc168799194)

[4.1 Overall Description 40](#_Toc168799195)

[4.1.1 Product Perspective 41](#_Toc168799196)

[4.1.2 Product Features 41](#_Toc168799196)

[4.1.3 Design and Implementation Constraints 42](#_Toc168799197)

[4.1.4 Assumption and Dependencies 42](#_Toc168799198)

[4.2 System Features 43](#_Toc168799199)

[4.2.1 Order Us Page 43](#_Toc168799200)

[4.2.2 Contact Us Page 43](#_Toc168799201)

[4.2.3 About Us Page 43](#_Toc168799202)

[4.2.4 Login Page 44](#_Toc168799203)

[4.2.5 Sign up 44](#_Toc168799204)

[4.2.6 Registration 44](#_Toc168799206)

[4.2.7 Login 44](#_Toc168799206)

[4.2.8 Reset Password Flow Chart 45](#_Toc168799207)

[4.2.9 Edit Profile Flow Chart 46](#_Toc168799208)

[4.2.10 Changes to Cart 46](#_Toc168799209)

[4.2.11 Search Product Flow Chart 47](#_Toc168799210)

[4.2.12 Add Product to Cart and Checkout Flow Chart 47](#_Toc168799211)

[4.2.13 Payment 48](#_Toc168799212)

[4.2.14 Logout 48](#_Toc168799213)

[4.2.15 Report Generation 48](#_Toc168799214)

[4.2.16 Admin Dashboard 49](#_Toc168799215)

[4.2.17 Dashboard List 51](#_Toc168799216)

[4.2.18 Order 51](#_Toc168799217)

[4.2.19 View Order Detail 51](#_Toc168799219)

[4.2.20 Products 52](#_Toc168799221)

[4.2.21 Add Product 52](#_Toc168799222)

[4.2.22 User 52](#_Toc168799224)

[4.2.23 Add User 53](#_Toc168799225)

[4.2.24 Admin List 53](#_Toc168799227)

[4.2.25 Add Admin 53](#_Toc168799229)

[Chapter 5 54](#_Toc168799231)

[IMPLEMENTATION 54](#_Toc168799232)

[5.1 Component Diagram 54](#_Toc168799233)

[5.2 Deployment Diagram 55](#_Toc168799234)

[5.3 Database Architecture (1- Tier, 2-Tier, 3- Tier Architecture) 56](#_Toc168799235)

[Chapter 6 58](#_Toc168799236)

[TESTING (SOFTWARE QUALITY ATTRIBUTES) 58](#_Toc168799237)

[6.1 Test Case Specification 58](#_Toc168799238)

[6.2 Black Box Test Cases 58](#_Toc168799239)

[6.3 White Box Testing 60](#_Toc168799240)

[Chapter 7 61](#_Toc168799241)

[TOOL & TECNOLOGIES 61](#_Toc168799242)

[7.1 Programming Languages 61](#_Toc168799243)

[7.2 Operating Environment 62](#_Toc168799244)

[Appendix A: User Documentation 70](#_Toc168799242)

**LIST OF FIGURE**

|  |  |  |
| --- | --- | --- |
| **Figure no. No** | **Statement** | **Page No** |
| Figure no. 2.1 | MVC Diagram | 21 |
| Figure no. 2.2 | ER-Diagram | 23 |
| Figure no. 2.3 | DFD (L1) | 26 |
| Figure no. 2.4 | DFD (L2) | 27 |
| Figure no. 2.5 | Class Diagram | 28 |
| Figure no. 2.6 | Object Diagram | 29 |
| Figure no. 2.7 | Sequence Diagram | 30 |
| Figure no. 2.8 | Activity Diagram | 31 |
| Figure no. 2.9 | Collaboration Diagram | 32 |
| Figure no. 2.10 | State Transition Diagram | 33 |
| Figure no. 3.1 | ERD Site Flow | 36 |
| Figure no. 4.1 | Constraints and Implementation Constraints | 41 |
| Figure no. 4.2 | Login Flow Chart | 44 |
| Figure no. 4.3 | Reset Password Flow Chart | 44 |
| Figure no. 4.4 | Edit Profile Flow Chart | 45 |
| Figure no. 4.5 | Search Product Flow Chart | 46 |
| Figure no. 4.6 | Add Product Cart and Checkout Flow Chart | 46 |
| Figure no. 5.1 | Component Diagram | 53 |
| Figure no. 5.2 | Deployment Diagram | 54 |
| Figure no. 5.3 | Database Architecture Diagram | 55 |
| Figure no. 7.1 | VS Code | 60 |
| Figure no. 7.2 | Data Flow Diagram L0 | 64 |
| Figure no. 7.3 | Data Flow Diagram L1 | 65 |

# Chapter 1

# INTRODUCTION TO THE PROBLEM

**1.1 Introduction**

The comprehensive documentation of "ShopSmart Wiki" a Full Stack e-commerce project developed using the MERN Stack (MongoDB, Express.js, React.js, Node.js). This documentation aims to provide you with a detailed understanding of the project, its purpose, objectives, scope, and more. Whether you are a computer science student seeking to explore application design or a developer interested in full-stack web development, this documentation is designed to cater to your needs.

E-commerce (Electronic Commerce) is process of doing business through computer networks. The primary goal of an e-commerce site is to sell goods and services online. Online shopping is a form of electronic shopping store where the buyer is directly online to the seller’s computer usually via the internet. A person sitting on his chair in front of a computer can access all the facilities of the Internet to buy or sell the products. Online Shopping System helps in buying of goods, products and services online by choosing the listed products from website (E-Commerce site). The Shopping cart is mainly useful for who haven’t time to go to shopping. Shopping cart is a very important feature used in e-commerce to assist people making purchases online. The sale and purchase transaction is completed electronically and interactively in real-time. User can login into E-Commerce website, once he logged in then automatically one shopping cart will be created, once user select an item it will add to cart. In case user thinks the selected item is not useful for him, then he can delete that item form the cart. Report generation feature is provided using Crystal Reports to generate different kinds of reports like bar graphs, pie charts and table type charts etc.

Electronic commerce, commonly known as E-commerce, refers to the buying and selling of products or services conducted over electronic systems, with the internet being the most prevalent medium. "ShopSmart Wiki" is a software application specifically crafted to facilitate the learning process for computer science students interested in application design using client-side languages like React.js and HTML, along with the server-side language Node.js Express.

**1.2 Overview**

E-commerce is fast gaining ground as an accepted and used business paradigm. More and more business houses are implementing web sites providing functionality for performing commercial transactions over the web. It is reasonable to say that the process of shopping on the web is becoming commonplace. The objective of this project is to develop a general purpose ShopSmart Wiki where product like clothes can be bought from the comfort of home through the Internet. However, for implementation purposes, this paper will deal with an online shopping for clothes. An online store is a virtual store on the Internet where customers can browse the catalog and select products of interest. The selected items may be collected in a shopping cart. At checkout time, the items in the shopping cart will be presented as an order. At that time, more information will be needed to complete the transaction. Usually, the customer will be asked to fill or select a billing address, a shipping address, a shipping option, and payment information such as credit card number. An e-mail notification is sent to the customer as soon as the order is placed.

The proposed system helps in building a website to buy, sell products or goods online using internet connection. Unlike traditional commerce that is carried out physically with effort of a person to go and get products, E-commerce has made it easier for human to reduce physical work and to save time. The basic concept of the application is to allow the customer to shop virtually using the Internet and allow customers to buy the items and articles of their desire from the store. E-commerce is fast gaining ground as an accepted and used business paradigm.

More and more business houses are implementing web site providing functionality for performing commercial transactions over the web. E-commerce websites provides consumers with less expensive products and services by allowing them to shop in many places and conduct quick comparisons. E-Commerce which was started in early 1990’s has taken a great leap in the world of computers, but the fact that has hindered the growth of e-commerce is security. Customer selected some items, but in his credit or debit cart haven’t that much balance, then he was logout from the website, the selected items are stored at cart with specific users with his allotted carts, after some days he bought those items then automatically deleted from the cart.

ShopSmart Wiki System is the Simple shopping Solution. In day to day life, we will need to buy lots of goods or products from a shop. Customer can login and get various information about product and can purchase the suitable product. It may be food items, electronic items, house hold items etc. Customer can pay online, so security is must therefore E-commerce website provide secure transactions. Now a days, it is really hard to get some time to go out and get them by ourselves due to busy life style or lots of works. In order to solve this, B2C E-Commerce websites have been started. Using these websites, we can buy goods or products online just by visiting the website and ordering the item online by making payments online. After sale E-commerce website also provide after sales service in which customer problem is solved.

Electronic commerce refers to the buying and selling of products or services over electronic systems such as the Internet and other computer networks. The Shopping Cart is very important feature used in e-commerce to assist people making purchases products online. It also includes the entire online process of developing, marketing, selling, delivering, servicing and paying for products and services. In order to purchase a shopping cart is provided to the user. The amount of trade conducted electronically has grown extraordinarily with widespread Internet usage. The use of commerce is conducted in this way, spurring and drawing on innovations in electronic funds transfer, supply chain management, Internet marketing, online transaction processing, electronic data interchange (EDI), inventory management systems, and automated data collection systems. The E-commerce site will let customers to view and order products online from any part of the world.

The main advantage of e-commerce over traditional commerce is the user can browse online shops, compare prices and order merchandise sitting at home on their PC. Secure registration and profile management facilities for Customers. Shopping Cart feature allows online shopping customers to “place” items in the cart. It Decreases the cost of creating, processing, distributing, storing and retrieving paper-based information. Expands the marketplace to national and international markets. Upon “checkout” the software calculates as total for the order including shipping and handling postage, packing and taxes, if applicable. Reduces the time between the outlay of capital and the receipt of products and services. Customers should be able to mail the Shop about the items they would like to see in the Shop.

System helps in building a website to buy, sell products or goods online using internet connection. Enables consumers to shop or do other transactions 24 hours a day, all year round from almost any location. It can be accessed over the Internet.

Purchasing of goods online, user can choose different products based on categories online payments, delivery services and hence covering the disadvantages of the existing system and making the buying easier and helping the vendors to reach wider market. It Provides consumers with more choices. Customer can purchase Products Online.

This existing system of buying goods has several disadvantages. It requires lots of time to travel to the particular shop to buy the goods. It is having lots of manual work. Since everyone is leading busy life now a days, time means a lot to everyone. Also there are expenses for travelling from house to shop. It is less user-friendly. In current system user must go to shop and order products. It is difficult to identify the required product. More over the shop from where we would like to buy something may not be open. Hence we have to adjust our time with the shopkeeper’s time or vendor’s time. In current e commerce system user have to go shop to view the description of the product. It is unable to generate different kinds of report.

## 1.3 Background Study

The concept of electronic commerce, commonly known as E-commerce, emerged with the rise of the internet and digital technologies. Traditional commerce, which involved physical transactions at brick-and-mortar stores, began to transition to the digital realm in the 1990s. With the advent of secure online payment systems and advancements in internet connectivity, E-commerce quickly gained popularity and transformed the way people buy and sell goods and services.

E-commerce eliminates the geographical barriers that were once associated with traditional retail, allowing businesses to reach a global customer base. Consumers can browse through a vast array of products and services from the comfort of their homes, and businesses can operate 24/7, providing customers with the convenience of shopping at any time.

The rise of E-commerce has also given birth to various business models, such as Business-to-Consumer (B2C), Business-to-Business (B2B), Consumer-to-Consumer (C2C), and more. B2C E-commerce involves businesses selling directly to individual consumers, while B2B E-commerce involves businesses selling products or services to other businesses. C2C E-commerce platforms enable consumers to sell products or services to each other through online marketplaces.

The growth of E-commerce has been fueled by technological advancements, increased internet penetration, and the widespread adoption of mobile devices. Mobile commerce, or m-commerce, has become increasingly significant as consumers prefer to shop on their smartphones and tablets. As E-commerce continues to evolve, new trends and innovations shape the industry. Personalization, artificial intelligence, virtual reality, and Omni channel experiences are becoming integral to delivering exceptional customer experiences in the digital landscape.

The success of E-commerce platforms depends not only on their user-friendly interfaces but also on robust backend systems that handle inventory management, order processing, payment gateways, and security measures to protect sensitive customer data. In the realm of education, computer science students play a vital role in shaping the future of E-commerce. Learning application design and development skills is crucial for aspiring developers to create cutting-edge E-commerce platforms that cater to the ever-changing needs of consumers and businesses alike.

The objective of this project is to develop a general-purpose ShopSmart Wiki where any product (such as books, CDs, computers, mobile phones, electronic items, and home appliances) can be bought from the comfort of home through the Internet. However, for implementation purposes, this paper will deal with an online E-commerce store. An online store is a virtual store on the Internet where customers can browse the catalog and select products of interest. The selected items may be collected in a shopping cart. At checkout time, the items in the shopping cart will be presented as an order. At that time, more information will be needed to complete the transaction. Usually, the customer will be asked to fill or select a billing address, a shipping address, a shipping option, and payment information such as a credit card number. An email notification is sent to the customer as soon as the order is placed.

Project planning is part of project management, which relates to the use of schedules such as Gantt charts to plan and subsequently report progress within the project environment. Initially, the project scope is defined and the appropriate methods for completing the project are determined. Following this step, the durations for the various tasks necessary to complete the work are listed and grouped into a work breakdown structure. The logical dependencies between tasks are defined using an activity network diagram that enables identification of the critical path. Float or slack time in the schedule can be calculated using project management software. Then the necessary resources can be estimated and costs for each activity can be allocated to each resource, giving the total project cost. At this stage, the project plan may be optimized to achieve the appropriate balance between resource usage and project duration to comply with the project objectives. Once established and agreed, the plan becomes what is known as the baseline. Progress will be measured against the baseline throughout the life of the project

E-commerce is fast gaining ground as an accepted and used business paradigm. More and

more business houses are implementing web sites providing functionality for performing

commercial transactions over the web. It is reasonable to say that the process of shopping

on the web is becoming commonplace.

The objective of this project is to develop a general-purpose ShopSwift where any

product (such as books, CDs, computers, mobile phones, electronic items, and home

appliances) can be bought from the comfort of home through the Internet. However, for

implementation purposes, this paper will deal with an online E-commerce store.

An online store is a virtual store on the Internet where customers can browse the catalog

and select products of interest. The selected items may be collected in a shopping cart. At

checkout time, the items in the shopping cart will be presented as an order. At that time,

more information will be needed to complete the transaction.

E-commerce is fast gaining ground as an accepted and used business paradigm. More and

more business houses are implementing web sites providing functionality for performing

commercial transactions over the web. It is reasonable to say that the process of shopping

on the web is becoming commonplace.

The objective of this project is to develop a general-purpose ShopSwift where any

product (such as books, CDs, computers, mobile phones, electronic items, and home

appliances) can be bought from the comfort of home through the Internet. However, for

implementation purposes, this paper will deal with an online E-commerce store.

An online store is a virtual store on the Internet where customers can browse the catalog

and select products of interest. The selected items may be collected in a shopping cart. At

checkout time, the items in the shopping cart will be presented as an order. At that time,

more information will be needed to complete the transaction.

E-commerce is fast gaining ground as an accepted and used business paradigm. More and

more business houses are implementing web sites providing functionality for performing

commercial transactions over the web. It is reasonable to say that the process of shopping

on the web is becoming commonplace.

The objective of this project is to develop a general-purpose ShopSwift where any

product (such as books, CDs, computers, mobile phones, electronic items, and home

appliances) can be bought from the comfort of home through the Internet. However, for

implementation purposes, this paper will deal with an online E-commerce store.

An online store is a virtual store on the Internet where customers can browse the catalog

and select products of interest. The selected items may be collected in a shopping cart. At

checkout time, the items in the shopping cart will be presented as an order. At that time,

more information will be needed to complete the transaction.

E-commerce is fast gaining ground as an accepted and used business paradigm. More and

more business houses are implementing web sites providing functionality for performing

commercial transactions over the web. It is reasonable to say that the process of shopping

on the web is becoming commonplace.

The objective of this project is to develop a general-purpose ShopSwift where any

product (such as books, CDs, computers, mobile phones, electronic items, and home

appliances) can be bought from the comfort of home through the Internet. However, for

implementation purposes, this paper will deal with an online E-commerce store.

An online store is a virtual store on the Internet where customers can browse the catalog

and select products of interest. The selected items may be collected in a shopping cart. At

checkout time, the items in the shopping cart will be presented as an order. At that time,

more information will be needed to complete the transaction.

E-commerce is fast gaining ground as an accepted and used business paradigm. More and more business houses are implementing web sites providing functionality for performing commercial transactions over the web. It is reasonable to say that the process of shopping on the web is becoming commonplace. The objective of this project is to develop a general-purpose ShopSmart Wiki where any product (such as books, CDs, computers, mobile phones, electronic items, and home appliances) can be bought from the comfort of home through the Internet. However, for implementation purposes, this paper will deal with an online E-commerce store. An online store is a virtual store on the Internet where customers can browse the catalog and select products of interest. The selected items may be collected in a shopping cart. At checkout time, the items in the shopping cart will be presented as an order. At that time, more in for usually, the customer will be asked to fill or select a billing address, a shipping address, shipping option, and payment information such as a credit card number. An email notification is sent to the customer as soon as the order is placed motion will be needed to complete the transaction.

## 1.4 Purpose

The primary purpose of "ShopSmart Wiki" is to serve as an educational tool for students, providing them with valuable insights into the world of E-commerce and empowering them to design interactive E-commerce applications from scratch. By integrating client-side languages such as React.js and HTML with server-side Node.js Express, students can grasp the fundamental concepts of web application development.

Automation of product manipulation.

•Buying products.

•To manage information of different types of items.

•Consistently update information of all the item.

•Managing security by providing authorized email & password.

Manages database efficiently

* Automation of product manipulation.
* Buying products.
* To manage information of different types of items.
* Consistently update information of all the item.
* Managing security by providing authorized email & password. Manages database efficiently

## 1.5 Scope

The "ShopSmart Wiki" project has a broad scope, offering a comprehensive e-commerce platform designed to be available 24x7 for customers to browse, shop, and place orders at their convenience. In the future, the system could potentially be implemented by local shopkeepers, allowing them to provide an online interface for customers, leading to more efficient deliveries and enhanced customer experiences.

## 1.6 Objective

The motivation behind developing the "ShopSmart Wiki" shopping-cart application stems from a passion for online shopping and a desire to create a cost-effective, custom-designed E-commerce platform. Additionally, the project seeks to showcase the power and dynamism of Node.js and React programming languages in web design and applications. Beyond its educational aspect, "ShopSmart Wiki" provides a platform for visualizing code on a user interface, making it an invaluable resource for computer science students and developers alike.

## 1.7 Intended Audience and Reading Suggestions

This documentation is primarily intended for computer science students, web developers, and anyone interested in full-stack web development and E-commerce applications. If you are new to the MERN Stack or web development in general, it is recommended to start from the beginning and proceed sequentially. For experienced developers, you can jump directly to specific sections of interest.

|  |  |
| --- | --- |
| **Agency Goals** | **Project Objectives** |
| A well-defined goal will help everyone on the team be on the same page and focus on the primary objectives. These principles can help you achieve tremendous success from your E-commerce website. | An objective is a measurable milestone that you must complete to meet one of your goals. It is something you must do to achieve your end goal. For example, if your goal is to increase online sales by 10%, one of your objectives might be to increase website traffic by 20%. If you can reach that objective, you are well on achieving your goal. |
| When setting website goals and objectives for your E-commerce, it is essential to understand the difference between outputs and outcomes. The significant difference between the two is that outputs are deliverables of the project. In contrast, outcomes are the results of achieving those deliverables, that is, the impact these outputs bring to the world.  Outputs are the things you will produce to meet your objectives. In other words, they are the things you must do to meet your goals. For example, if you start an online clothing store and aim to increase sales, one of your objectives is to hire a new E-commerce manager. In this case, the objective is to find the right E-commerce manager for the job. Likewise, the outcome will be to get good E-commerce management and an efficient and orderly online sales process. | Setting specific and measurable website goals for your E-commerce is crucial because this will help you get everyone on the team to agree on what we want to achieve. It will also make it easier to monitor the success of your goals and help you see if you are moving in the right direction, if you are on the right track, if you need to make some adjustments, etc. By setting specific and actionable goals, you can progress toward achieving your objectives without getting lost in the bigger picture. |
| **Output-based deliverables and products**  Outputs are often tangible deliverables such as products, services, or data. For example, when you set goals and objectives for your E-commerce, one of your objectives could be to build a new E-commerce website. This objective would be an output-based deliverable. At the same time, you could have another objective: increase your sales by a certain percentage. This objective would be an outcome-based result. | There are many different ways of setting goals and objectives for your E-commerce. However, the most effective way to measure success is to use the **SMART** framework. **SMART** is **specific**, **measurable**, **achievable**, **relevant**, and **time**-**bound**.  **Specific**  What are your goals and objectives? What are you trying to achieve? What are the outcomes you want to measure? Be as specific as possible when setting goals and objectives.  **Metrics**  What metrics will you use to measure success? What are the outcomes you want to measure? How can you tell if your goals and **objectives** were successful? How can you use data and insights to measure progress? If you want to answer these questions, perhaps you need to use Google Analytics as well. |

## 1.8 Documentation Conventions

To support users in navigating and utilizing the application effectively, "ShopSmart Wiki" provides comprehensive user documentation, including step-by-step guides and FAQs.

# Chapter 2

# SOFTWARE REQUIREMENT SPECIFICATION

The analysis model is a crucial phase in the software development process, as it helps in understanding the system's requirements, interactions, and relationships between different components. In this chapter, we present various analysis diagrams that offer a comprehensive view of the "ShopSmart Wiki" e-commerce platform.

# 2.1 User Registration Requirements

* Build value around registration: Before users create an account you need to explain the reason for creating an account.
* Eliminate as many fields as possible: Only get data from user which are most important and remove or make all other fields as "Optional" which are not necessary about user.
* Group fields logically: Some sites gather a lot of information in registration form like shipping address, billing address etc, So many users want to deliver their personal information in certain order and we need an order for user registration. Break up long registration processes into steps and finally bread down all the processes into a well steps.
* Make your security and privacy policies clear as day: Make sure that the user who is creating an account on our website is secure, and include a clear link of our privacy policy.
* Make password recovery easy: If a user forgets his password or loses their password in some cases, so we want a password recovery link in our both registration and login form.
* Given the option to refresh captcha: In order to secure website from robots usage. So we want a captcha code for users to enter it while creating an account.
* Give a Stripe method: Before users enter into their accounts while in registration they need to confirm their account on PayPal or any other bank account.
* User Login System: It is necessary for every user to login into their accounts before they move to purchase any product from our website. We want a user email and password in login form for the users.
* Admin Panel: We want an admin panel in which we can control all the information/data about our website like we can insert a new data from admin panel and delete that data about product which are not available in our shop/market. And our admin panel must be secure in which only two admin can access to our admin area and edit / delete the data. And we have user’s information in our admin area So that we can easily check the user’s detail.

# 2.2 External Interface Requirements

The success of an e-commerce platform heavily relies on its user interfaces. The "ShopSmart Wiki" system prioritizes an intuitive and visually appealing design to provide a seamless shopping experience for its users. The user interfaces have been carefully crafted to cater to users of all levels of technical expertise.

**Homepage**: The homepage serves as the virtual storefront, welcoming users with an aesthetically pleasing layout. It showcases featured products, latest promotions, and trending items to capture users' attention. Clear navigation options guide users to explore different product categories and make informed choices.

**Registration and Login Page**: The registration process is straightforward and encourages new users to create accounts efficiently. Required fields are kept to a minimum to simplify the process. Existing users can log in securely by entering their unique user ID and password. The login page includes user-friendly error messages to help users troubleshoot login issues effectively.

**Product Listings**: The product listings display an extensive collection of products with high-quality images, detailed descriptions, prices, and customer reviews. Users can sort and filter products based on various criteria, such as price range, brand, popularity, and customer ratings. The use of visual cues and icons enhances the user experience and ensures easy navigation through the vast array of products.

**Shopping Cart**: The shopping cart offers a user-friendly interface where users can view the products they have selected. Quantities can be adjusted, and products can be added or removed as needed. The cart also displays the total cost of the order and any applicable discounts, making it easy for users to keep track of their purchases.

**Checkout**: The checkout process is designed to be straightforward and secure. Users are guided through the payment steps, presented with various payment options, and asked to provide delivery details. A summary of the order, including the final cost and selected items, is shown for review before the user confirms the purchase.

**User Profile**: The user profile section allows customers to manage their account information, view their order history, and track the status of their shipments. Users can update personal details, including shipping addresses and payment methods, to ensure a smooth shopping experience in the future.

**Order Confirmation and Invoice**: After completing a purchase, customers receive an order confirmation via email, providing them with a sense of security and assurance. The email includes a detailed invoice with the order number, product information, payment summary, and shipping details. This feature enables users to keep track of their purchases and acts as a reference for future inquiries or returns.

The user interfaces of the "ShopSmart Wiki" system aim to instill confidence in users, encouraging them to explore the platform and make purchases with ease. The emphasis on user experience ensures that both tech-savvy and novice users can navigate the system comfortably, fostering customer loyalty and satisfaction.

**Terms & Conditions**

* **Privacy Policy**: The privacy policy, which will set out how to use End user information, place at [Privacy Policy Link].
* **Prohibitions**: Display to user as “You must not misuse this Website. You will not: commit or encourage a criminal offense; transmit or distribute a virus, Trojan, worm, logic bomb or any other material which is malicious, technologically harmful, Breaching this provision would constitute a criminal offense.”
* **Intellectual Property, Software and Content**: The intellectual property rights in all s/w and content (including all graphics) made available to user on or through this system remains the property of [site URL].
* **Terms of Sale**: By placing an order, offering to purchase a product on and subject to the following terms and conditions. All orders will subject to availability and confirmation of the order price
* **Contract**: When user place an order, send an acknowledgement through e-mail confirming Receipt of order to user.
  + Pricing and Availability. Whilst try and ensure that all details, descriptions and prices which appear on the system, are accurate, errors may occur.
  + Payment receiving user’s order carry out a standard authorization check on user payment Card to ensure there are sufficient funds to fulfill the transaction.
* **Complaints:** Operate a complaints handling procedure which will use to try to resolve disputes when they first arise, let user to complaints or any comments anything about system.
* **Entire Agreement:** The above Terms of Service constitute the entire agreement of the parties and supersede any and all preceding and contemporaneous agreements between user and [site URL]. Any waiver of any provision of the Terms of Service will be effective only if in writing and signed by a Director of [site URL].

The "ShopSmart Wiki" system operates on standard hardware components commonly found in computing setups.

The system's hardware interfaces include:

**User Devices**: The system is accessible through various user devices, including desktop computers, laptops, tablets, and smartphones. The responsive design ensures that the user interface adapts seamlessly to different screen sizes and resolutions. Users can conveniently access "ShopSmart Wiki" from their preferred devices, enabling a flexible shopping experience.

**Servers**: The "ShopSmart Wiki" system relies on powerful servers to host the application and manage the database. These servers are equipped with adequate processing power, storage capacity, and memory to handle multiple concurrent user requests efficiently. High-performance servers contribute to a smooth user experience by ensuring quick response times and minimal downtime.

**Network Infrastructure**: A robust network infrastructure with stable internet connectivity is essential for the seamless communication between user devices and the server. To avoid disruptions in the shopping process, "ShopSmart Wiki" employs redundant network connections and load balancers, guaranteeing consistent and uninterrupted access to the platform.

**Data Storage Solutions**: The system utilizes reliable data storage solutions, such as cloud-based storage or on-premises databases, to store product information, user profiles, and order details securely. These storage solutions offer scalability, data redundancy, and data backup options to protect against data loss and ensure data availability.

The hardware interfaces of "ShopSmart Wiki" are carefully chosen to ensure optimal system performance, accessibility, and reliability. By leveraging the power of modern hardware technologies, the platform can accommodate a large user base, handle significant data loads, and provide a seamless shopping experience for customers.

**Specification**

**Analysis**

The potential objectives of this project website had been analysed for multiple times with and without outsiders. It was to make sure that the final product of this project could maximize the usefulness to help potential users. Firstly, it started with doing researches on the Internet about current trends of existing E-Commerce websites and what have not been done. By doing so, it led to many discoveries of new ideas and improvements on existing features. Then, the main impact audience of this project were set, which were all the E-Commerce users (buyers and sellers). After objectives had been set, literature reviews were done to study the background of related field which was E-Commerce website. It resulted in deeper understanding.

**Design**

The design of the website was started from scratch, from creating an empty project to a fully designed website. Every page of the design needed to be designed and arranged accordingly so that a good user experience (UX) would be accomplished. Besides, one of the main highlights of the website was a 2-D game. It was designed in a way that user to control a character to shop in a 2-D map terrain. This was one of the main features that could show the uniqueness of the website and drag more attentions from users.

**Verification Plan**

Verification plan was a repetitive step where the testing on newly built features were tested and verified for its expected output or usage. All the features would be tested after done coding to make sure the functionality. The testing was done by the developer himself and some college mates.

The "ShopSmart Wiki" system employs several software components to support its functionality and ensure seamless operation. Key software interfaces include:

**Operating System**: "ShopSmart Wiki" is designed to run on various operating systems, including Windows, macOS, and Linux. The compatibility with multiple operating systems allows users to access the platform using their preferred devices and operating environments.

**Web Browsers**: The user interfaces of "ShopSmart Wiki" are optimized to be compatible with popular web browsers, including Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge. Ensuring compatibility with diverse web browsers expands the reach of the platform to a broader audience.

**Database Management System**: The system utilizes a database management system (DBMS) to store and manage product information, user profiles, and order details efficiently. MongoDB, a NoSQL database, is preferred for its flexibility and scalability, enabling seamless data retrieval and storage.

**Server-Side Technologies**: The server-side of "ShopSmart Wiki" is developed using Node.js and Express.js. Node.js offers a fast, event-driven architecture that ensures efficient handling of server-side requests, while Express.js simplifies the development of RESTful APIs and facilitates smooth communication between the frontend and backend components.

**Client-Side Technologies**: The frontend of "ShopSmart Wiki" is built using React.js, a popular JavaScript library for building interactive user interfaces. React.js enables the creation of dynamic and responsive UI components, providing a smooth and engaging user experience.

**Payment Gateways and APIs**: "ShopSmart Wiki" integrates with payment gateways and third-party APIs to facilitate secure online transactions. These APIs handle payment processing and provide additional services, such as address verification and fraud detection, ensuring safe and reliable payment methods for customers.

The use of these software interfaces ensures the seamless integration and interoperability of "ShopSmart Wiki" components. By leveraging modern software technologies, the platform delivers a fast, responsive, and secure shopping experience to users, enhancing overall customer satisfaction.

Communication interfaces are essential for facilitating data exchange and connectivity within the "ShopSmart Wiki" system. Key communications interfaces include:

**HTTP/HTTPS**: Communication between user devices and the server occurs using the Hypertext Transfer Protocol (HTTP) or its secure counterpart, HTTPS. HTTPS ensures data encryption during transmission, safeguarding sensitive information such as login credentials and payment details.

**APIs**: "ShopSmart Wiki" may integrate with third-party APIs to enhance its functionality. Payment gateway APIs facilitate secure online transactions, while location-based APIs enable accurate shipping and delivery estimates. Integration with social media APIs allows users to share products and promotions, promoting organic marketing and engagement.

**Email Services**: The system utilizes email services to communicate with users effectively. Order confirmations, shipping notifications, and promotional offers are sent to users' registered email addresses, providing timely updates and enhancing customer engagement.

**Real-Time Notifications**: To keep users informed about the status of their orders and relevant updates, "ShopSmart Wiki" may implement real-time notification systems. Users receive notifications through web browser push notifications or mobile app alerts, ensuring that they stay informed about their orders' progress.

Communication interfaces play a crucial role in providing timely updates and seamless interactions between the "ShopSmart Wiki" system and its users. By leveraging these interfaces, "ShopSmart Wiki" ensures effective data transmission, secure communication, and efficient information sharing, all contributing to a positive and satisfactory user experience.

The system architecture and interfaces of "ShopSmart Wiki" are thoughtfully designed to provide a robust and user-friendly e-commerce platform. The intuitive user interfaces cater to diverse users, while the hardware, software, and communication interfaces work cohesively to deliver a seamless shopping experience. By prioritizing efficiency, security, and user satisfaction, "ShopSmart Wiki" aims to become a reliable and preferred destination for online shopping for users worldwide.

**Shipping policy and Delivery Time-frame**

* Shipping policy will easily accessible. Don’t require visitors to log in or create accounts, before they can view our shipping policy. Visitors to site will need to know the shipping terms before they place an order. An unpleasant surprise at the checkout will almost certainly lead to a lost sale.
* Accessibility of shipping policy from all pages. Place a link within website’s header or sidebar to do the trick.
* Provide all shipping details at the checkout. Tell customers how much each shipping option will cost and how long it will take. Provide full details on international shipping, if applicable.
* No overcharge for shipping. It should be clear that shipping is absolutely free.
* Provide detailed delivery time-frames. List delivery time-frames for each shipping option by delivery region.
* Provide tracking information. Email the shipment’s tracking number to customer as soon as it is available. Being able to track his order, customer will know at any given time exactly what the status of the shipment is.

Responsibility for lost shipments. Whenever if a shipment is lost, take responsibility and re-ship it. It may not be our fault, and it probably won’t be, but customer has placed an order with our business and will rightly hold us responsible for it.

# 2.3 Other Non-Functional Requirements

Non-functional requirements define system properties and constraints that are equally essential as functional requirements for the success of an e-commerce platform like "ShopSmart Wiki." In this chapter, we explore various non-functional requirements that contribute to the overall efficiency, effectiveness, and user satisfaction of the system.

## 2.2.1 Performance Requirements

Performance requirements are critical for ensuring that "ShopSmart Wiki" operates smoothly and provides a seamless shopping experience for users. These requirements encompass aspects such as response times, throughput, and scalability.

**Response Times**: The "ShopSmart Wiki" system aims to deliver rapid response times to user interactions. For example, the time taken to load product listings, display search results, and process transactions should be minimal. The system should prioritize efficient data retrieval and rendering, resulting in near-instantaneous response times to keep users engaged.

**Throughput**: The platform's throughput refers to the number of concurrent transactions it can handle. "ShopSmart Wiki" is designed to accommodate a large number of users simultaneously, ensuring smooth operations during peak traffic periods, such as flash sales or seasonal promotions. Adequate server resources and load balancing mechanisms are employed to maintain consistent and high throughput.

**Scalability**: The "ShopSmart Wiki" system is designed with scalability in mind. As the user base grows, the platform should seamlessly scale to accommodate increased demand. Elastic computing resources and horizontal scaling ensure that the system can handle a higher volume of transactions and users without sacrificing performance.

## 2.1.2 Safety Requirements

Safety requirements are essential for maintaining the security and integrity of the "ShopSmart Wiki" system and protecting user data and transactions from potential threats.

**Data Security**: "ShopSmart Wiki" employs robust data encryption techniques to safeguard user data, including personal information and payment details. Secure Socket Layer (SSL) and Transport Layer Security (TLS) protocols are used to encrypt data during transmission, preventing unauthorized access and ensuring the confidentiality of sensitive information.

**Payment Security**: Payment processing is handled securely through reputable and trusted third-party payment gateways. These gateways comply with Payment Card Industry Data Security Standard (PCI DSS) requirements, providing an additional layer of security for financial transactions.

**User Authentication**: The platform uses secure authentication mechanisms, such as password hashing and salting, to protect user credentials. Multi-factor authentication (MFA) may be implemented to further enhance account security.

**Secure** **APIs**: If the system integrates with third-party APIs, stringent security measures are applied to prevent unauthorized access and data breaches.

2.1.3 Security Requirements

Security is of paramount importance in any E-commerce application to protect user data, transactions, and sensitive information from unauthorized access and potential threats. The "ShopSmart Wiki" project incorporates a comprehensive set of security requirements to ensure a safe and secure online shopping environment for both users and administrators. These security measures encompass various aspects of the application, including authentication, data protection, and system integrity.

1. **User Authentication and Authorization:** The "ShopSmart Wiki" implements robust user authentication mechanisms to verify the identity of users during login and registration. It employs secure password storage practices, such as salting and hashing, to protect user credentials from unauthorized access. Role-based access control ensures that users are granted appropriate permissions based on their roles, preventing unauthorized actions.
2. **Secure Payment Processing:** To ensure the safety of financial transactions, the "ShopSmart Wiki" integrates with reputable and secure payment gateways. It implements encryption protocols such as SSL/TLS to secure the transmission of payment information between the user's browser and the server, safeguarding against man-in-the-middle attacks.
3. **Protection against Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF):** The application incorporates input validation and sanitization to prevent XSS attacks that could inject malicious scripts into web pages. Additionally, CSRF tokens are utilized to mitigate CSRF attacks, ensuring that requests from legitimate users are distinguished from forged requests.
4. **Secure Session Management:** The "ShopSmart Wiki" implements secure session management techniques, such as using HTTP-only and secure cookies, to prevent session hijacking and session fixation attacks. It sets appropriate session timeouts to automatically log out inactive users, enhancing security.
5. **Data Encryption and Privacy:** Sensitive data, such as user details and payment information, are encrypted using encryption algorithms to protect against data breaches. The "ShopSmart Wiki" complies with data privacy regulations and implements privacy settings to give users control over their personal information.
6. **Preventing SQL Injection:** The application adopts parameterized queries and prepared statements to defend against SQL injection attacks, ensuring that malicious SQL commands cannot be executed.
7. **Regular Security Audits and Penetration Testing:** The "ShopSmart Wiki" undergoes regular security audits and penetration testing to identify vulnerabilities and weaknesses in the application. Any identified issues are promptly addressed to maintain a secure environment.
8. **Error Handling and Logging:** Detailed error handling and logging mechanisms are implemented to capture and log errors securely. This aids in identifying potential security breaches and helps in debugging and troubleshooting.
9. **Secure File Uploads:** The "ShopSmart Wiki" validates and restricts file uploads to prevent malicious files from being uploaded to the server. File uploads are stored in secure locations, and access controls are enforced to limit file access.
10. **Security Patches and Updates:** The project team diligently applies security patches and updates for all software components, including frameworks and libraries, to keep the application protected against known vulnerabilities.
11. **Compliance with Industry Standards:** The "ShopSmart Wiki" complies with industry standards, such as Payment Card Industry Data Security Standard (PCI DSS), to ensure the secure handling of payment card data.

By incorporating these security requirements, the "ShopSmart Wiki" project ensures a trustworthy and secure platform for users to shop and transact online. These measures demonstrate a commitment to safeguarding user information and fostering a secure online shopping experience, making the "ShopSmart Wiki" a reliable and trusted destination for customers.

### **2.4 Architecture Diagram**

• The technologies which are used for building E-commerce are MongoDB, ExpressJS, ReactJS, NodeJS. This stack is popularly called MERN stack.

• The architectural work flow of MERN stack is as follows:

• In NodeJS, the request is processed by ExpressJS and this makes request to the database.

• Here, MongoDB will get the data and returns it to ExpressJS

• ExpressJS will now send the data to NodeJS and then the data is received by ReactJS, which is responsible for displaying the result. The main architecture that lies under this entire flow is the Model View Controller, which is popularly called MVC architecture.

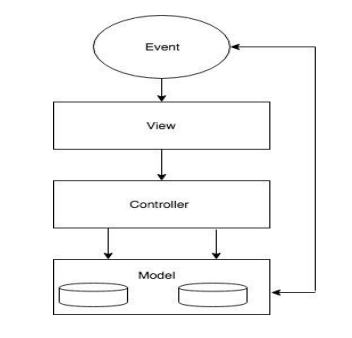


Figure 2.1 MVC Diagram

• **Model**: Model manages application data and it responds to the requests made from views, also listens to the instructions from controller.

• **View**: View is the portion which we see in the web application. View is responsible to display the data to user.

• **Controller**: Controller is very important because it controls the interactions between Model and Views. Controller responds to user input and performs interactions on the models. This also validates input.

When the user sends a request to through ReactJS then that request is firstly accessed by the NodeJS threading is done in the NodeJS and then it is sent to the ExpressJS.

* ExpressJS sends the request to MongoDB.
* MongoDB processes the request and send back to ExpressJS.
* NodeJS retrieves data from ExpressJS.
* Now, AngularJS receives data from NodeJS and updates the views and users can see the updated data.

### **2.5 Entity-Relationship Diagram (ERD)**

The Entity-Relationship Diagram (ERD) depicts the relationships between different entities in the database, representing the data model of the "ShopSmart Wiki" system.

Key Entities in the ERD include:

**Customer**: Represents customer data, including customer ID, name, email, and address.

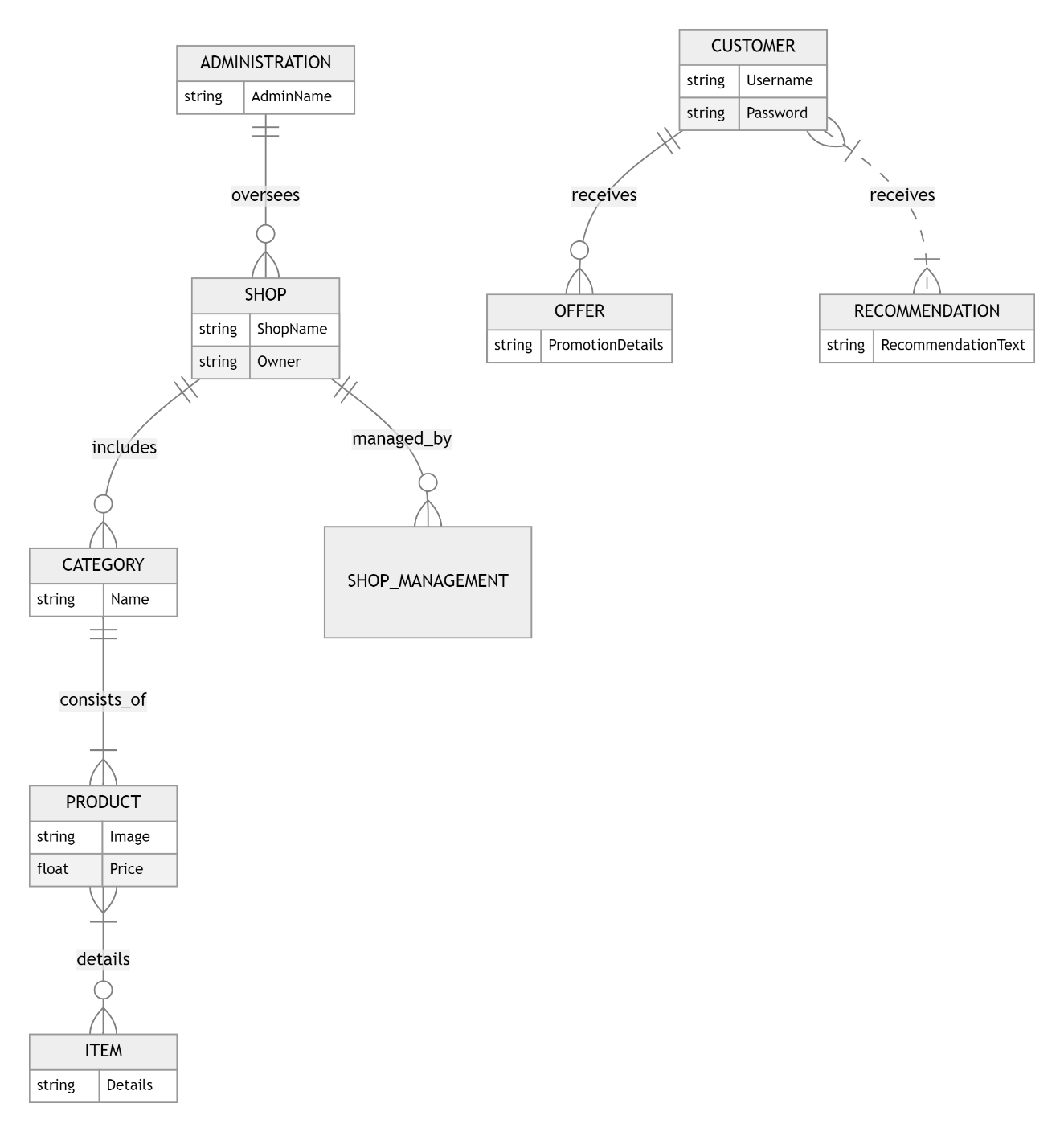
**Product**: Represents product information, such as product ID, name, price, and category.

**Order**: Represents order data, including order ID, customer ID, order date, and order status.

**Category**: Represents product categories and their relationships to products.

The relationships between entities are depicted using various cardinalities, such as one-to-many, many-to-many, and one-to-one.

In conclusion, the analysis model of the "ShopSmart Wiki" e-commerce platform encompasses various diagrams, each providing unique insights into the system's architecture, interactions, and functionality. The use case diagram outlines the platform's functional requirements, while the class diagram showcases the static structure of the system. Object, sequence, activity, collaboration, and state transition diagrams illustrate specific use cases and dynamic interactions within the system. Lastly, the Entity-Relationship Diagram offers a comprehensive view of the data model. Together, these analysis diagrams play a vital role in understanding, designing, and implementing a robust and efficient e-commerce platform like "ShopSmart Wiki."



**Figure 2.2 ER-Diagram**

### **2.6 Data Flow Diagram (DFD)**

A Data Flow Diagram (DFD) is a graphical representation of the flow of data within a system, illustrating how data moves between different processes, data stores, and external entities. For the "ShopSmart Wiki" project, the DFD can help depict the data movement between various components of the application.

**Level 0 DFD:**

At the highest level, the Level 0 DFD provides an overview of the major processes and data stores in the "ShopSmart Wiki" system. It illustrates the interactions between external entities and the system as well. Let's outline the main components of the Level 0 DFD for the "ShopSmart Wiki" project:

1. **External Entities:**
   * Customer: Represents the users of the "ShopSmart Wiki" who browse, search, and place orders.
   * Administrator: Represents the privileged user responsible for managing the store, products, and orders.
2. **Processes:**
   * Browse Products: Process where customers and guests can view the list of available products.
   * Search Products: Process where customers can search for specific products based on keywords or filters.
   * Manage Products: Process where the administrator can add, update, and delete products from the system.
   * Manage Orders: Process where the administrator can handle customer orders and update order statuses.
   * Process Order: Process where customers can place an order for the products in their shopping cart.
3. **Data Stores:**
   * Product Database: Stores all information related to products, such as product details, prices, and quantities.
   * Order Database: Stores information about customer orders, including order details and payment status.
4. **Data Flow:**
   * The data flow between processes represents the movement of data within the system. For example, the data flow from the "Browse Products" process to the "Product Database" indicates that the process retrieves product information from the database.

**Level 1 DFD:**

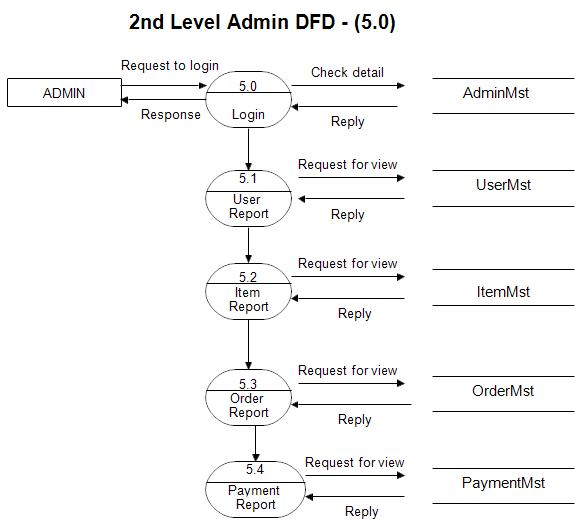
The Level 1 DFD provides a more detailed view of the individual processes from the Level 0 DFD. It decomposes each process into sub-processes, highlighting the data flows between them and the data stores. Let's explore some of the Level 1 DFD components for the "ShopSmart Wiki" project:

1. **Process Order:**
   * Authenticate User: Sub-process responsible for verifying the user's credentials during login or registration.
   * Add to Cart: Sub-process where authenticated users can add products to their shopping cart.
   * Remove from Cart: Sub-process where authenticated users can remove products from their shopping cart.
   * Process Payment: Sub-process that handles payment processing for customer orders.
   * Update Order Status: Sub-process that updates the order status based on payment completion.
2. **Browse Products:**
   * Retrieve Product List: Sub-process that retrieves the list of available products from the "Product Database."
   * Display Product List: Sub-process that displays the product list to the customer for browsing.
3. **Search Products:**
   * Process Search Query: Sub-process that handles customer search queries and filters products accordingly.
   * Display Search Results: Sub-process that displays the search results to the customer.
4. **Manage Products:**
   * Add New Product: Sub-process that allows the administrator to add new products to the "Product Database."
   * Update Product Details: Sub-process that enables the administrator to update product information.
   * Delete Product: Sub-process that allows the administrator to remove products from the system.
5. **Manage Orders:**
   * View Order List: Sub-process that displays the list of customer orders to the administrator.
   * Handle Customer Inquiries: Sub-process where the administrator can address customer inquiries related to orders.

The Level 1 DFD further breaks down the major processes into more manageable sub-processes, providing a detailed view of the data flows and interactions within the "ShopSmart Wiki" system. This comprehensive visualization aids in understanding the data processing and the overall flow of information in the application, which serves as a valuable foundation for the subsequent design and development phases.



**Figure 2.3 DFD (L1)**



**Figure 2.4 DFD (L2)**

**2.7 Class Diagram**

The Class Diagram provides a static view of the "ShopSmart Wiki" system, illustrating the classes, attributes, and relationships between objects.

The main classes in the "ShopSmart Wiki" system include:

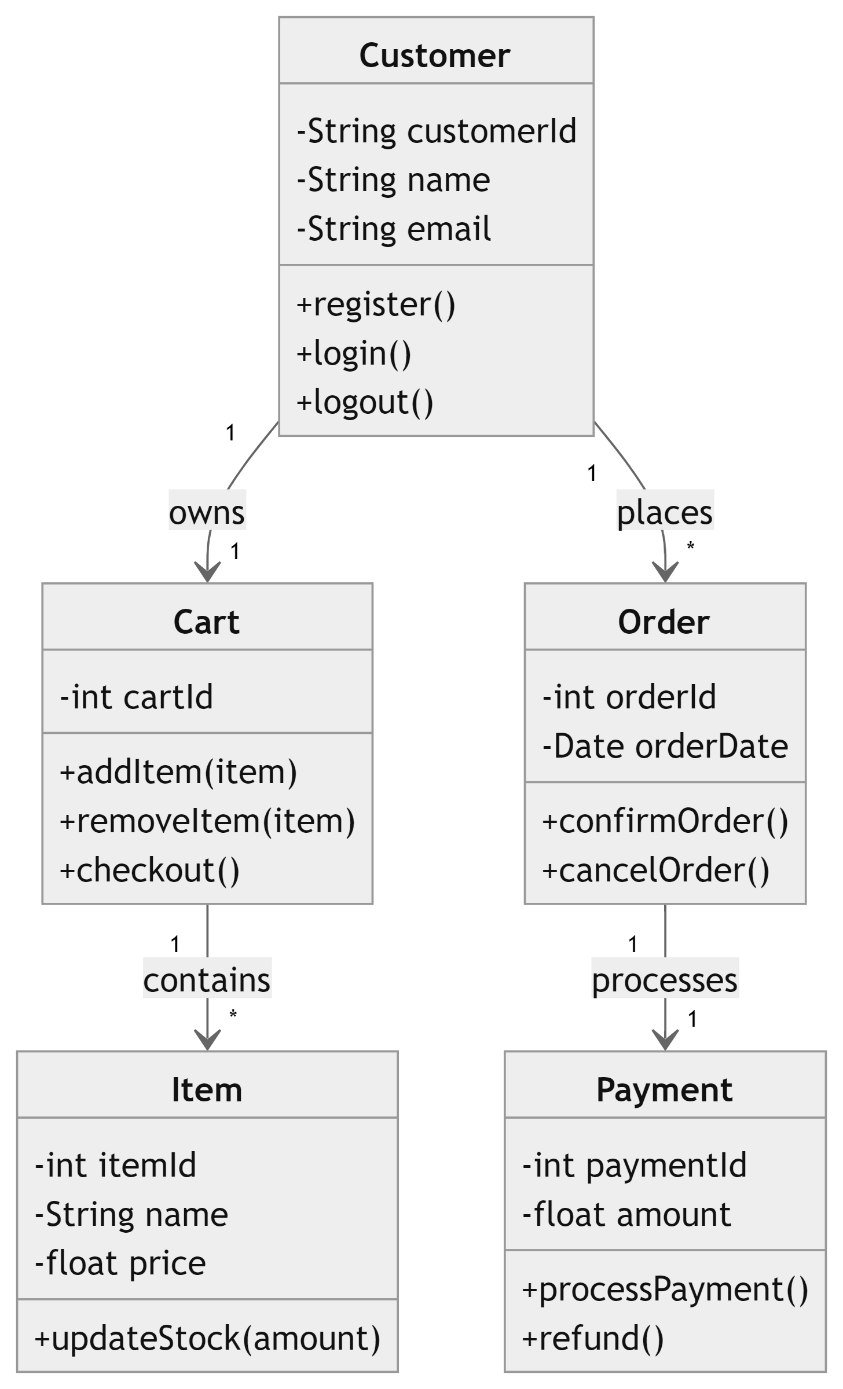
**Customer**: Represents the registered users on the platform and stores their personal information.

**Product**: Represents the products available for purchase and contains attributes like name, price, and category.

**Shopping Cart**: Represents the shopping cart where the Customer can add products before checkout.

**Order**: Represents the customer's order, including the list of purchased products and payment details.

**Administrator**: Represents the system administrator and stores administrative information.



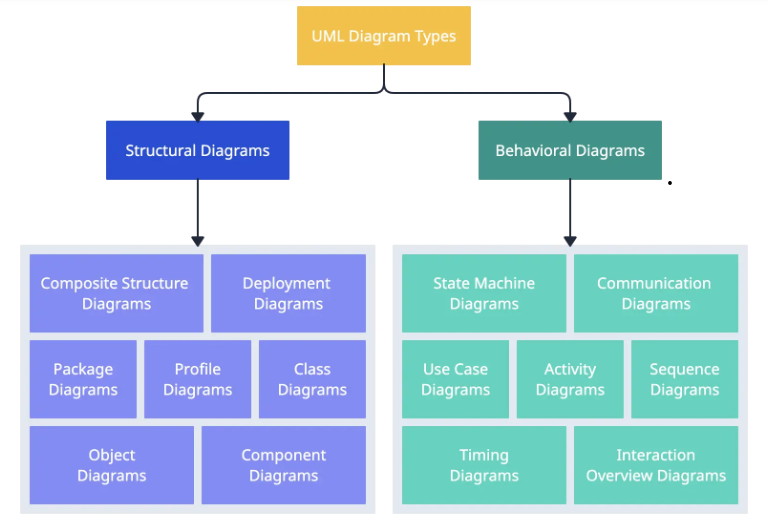
**Figure 2.5 Class Diagram**

The relationships between classes are represented by associations, such as the relationship between Customer and Order (one-to-many), Customer and Shopping Cart (one-to-one), and Product and Category (many-to-one).

### **2.8 Object Diagram**

The Object Diagram is a specific instance of the Class Diagram, showing objects and their relationships at a particular point in time.

* Example Object Diagram:
* Customer (object):
* Name: John Doe
* Email: john.doe@example.com
* Shopping Cart (object):
* Products: [Product 1, Product 2]
* Total Cost: $150
* Order (object):
* Order ID: 123456
* Customer: John Doe
* Products: [Product 1, Product 2]
* Total Cost: $150

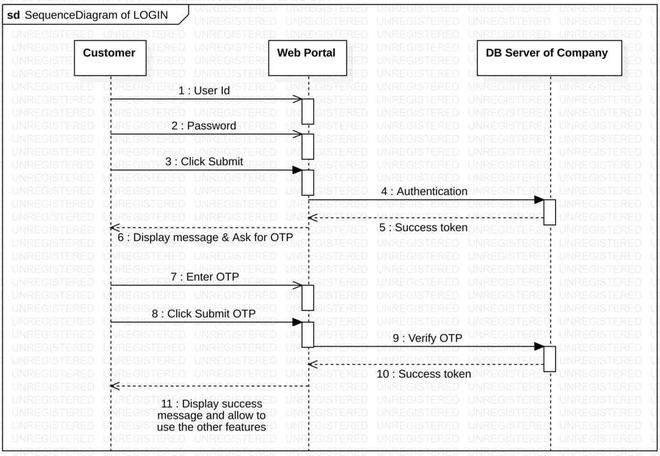


**Figure 2.6 Object Diagram**

### **2.9 Sequence Diagram**

The Sequence Diagram illustrates the interactions between objects and components over time, showcasing the flow of messages and actions during specific use cases.

* Example Sequence Diagram (Order Processing):
* Customer selects products and proceeds to checkout.
* The system calculates the total cost and presents the payment options.
* Customer selects the payment method (e.g., Cash on Delivery).
* The system generates an order confirmation and notifies the Customer.

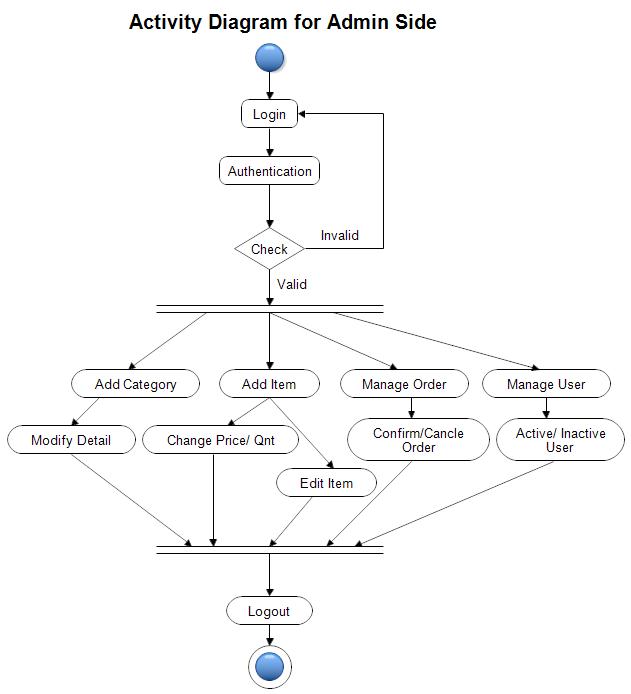


**Figure 2.7 Sequence Diagram**

### **2.10 Activity Diagram**

The Activity Diagram represents the flow of activities or processes within the "ShopSmart Wiki" system.

* Example Activity Diagram (Customer Registration):
* Customer starts the registration process.
* The system prompts the Customer to enter personal information.
* The Customer provides the required details.
* The system verifies the information and creates the Customer's account.
* The registration process is complete.

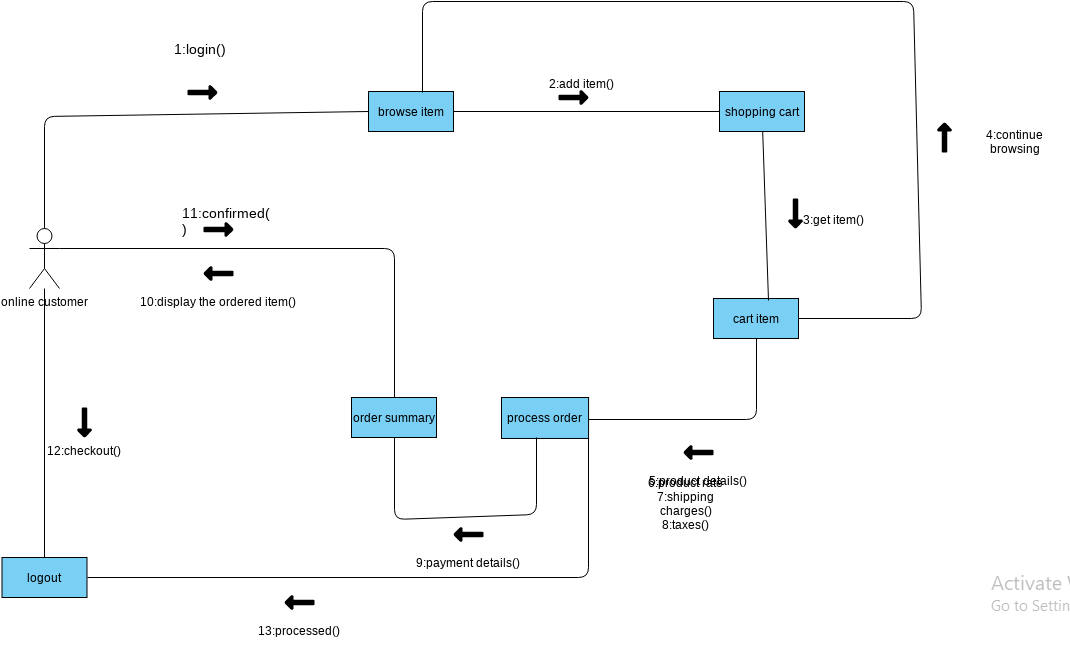


**Figure 2.8 Activity Diagram**

### **2.11 Collaboration Diagram**

The Collaboration Diagram shows how objects and components collaborate to achieve specific functionalities.

* Example Collaboration Diagram (Add to Cart):
* Customer selects a product and adds it to the Shopping Cart.
* The system updates the Shopping Cart with the selected product.
* The Shopping Cart sends a message to update the total cost.
* The system updates the total cost in the Shopping Cart.
* The Customer confirms the action, and the Shopping Cart is updated with the new product.

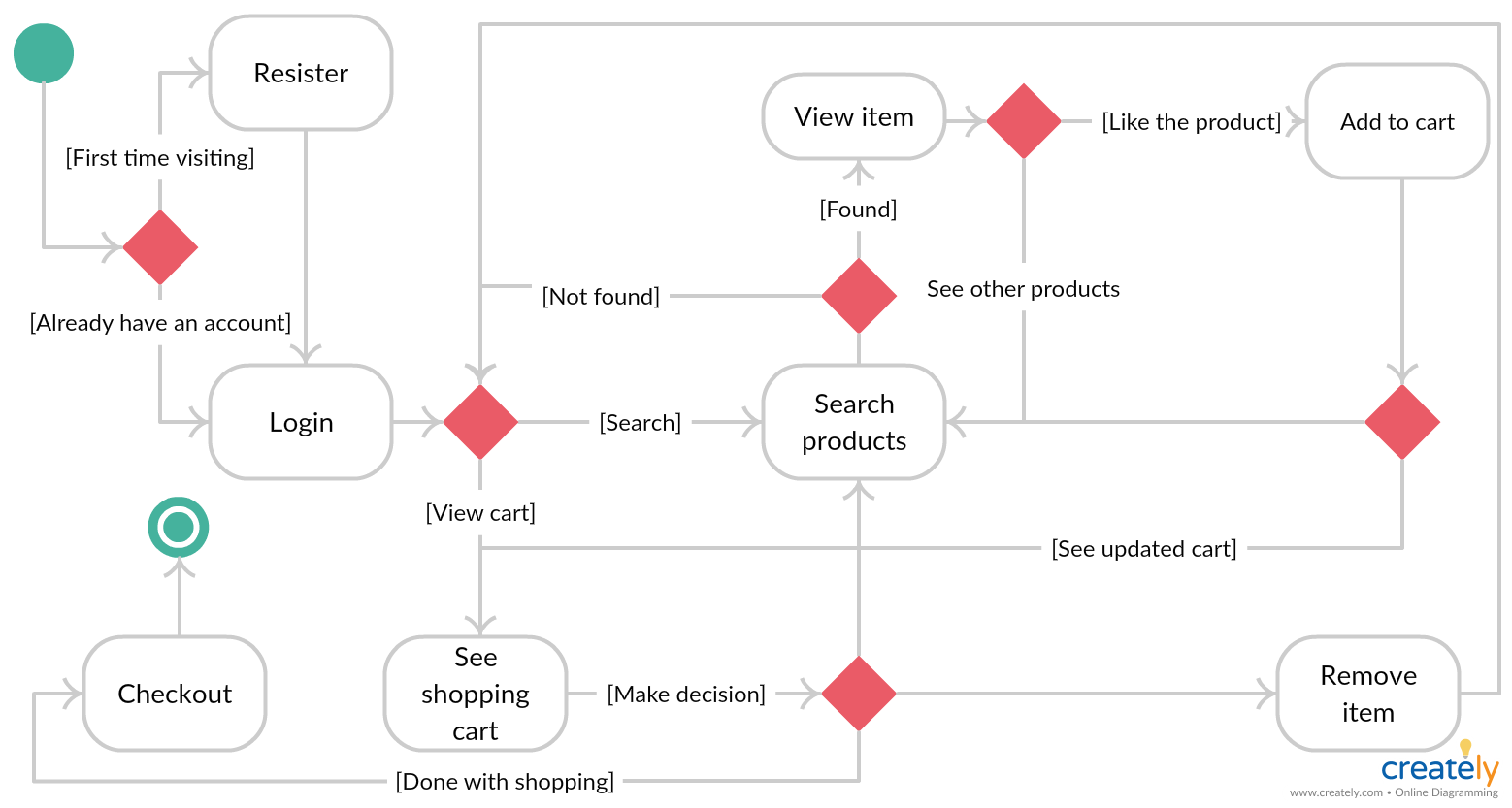


**Figure 2.9 Collaboration Diagram**

### **2.12 State Transition Diagram**

The State Transition Diagram illustrates how objects change states based on specific events or conditions.

* Example State Transition Diagram (Order Status):
* Initial State: New Order
* Events: Order Placed, Order Confirmed, Order Shipped, Order Delivered, Order Cancelled
* The system transitions the order from one state to another based on the events triggered during the order processing.



**Figure 2.10 State Transition Diagram**

# Chapter 3

# ANALYSIS (USE CASE MODEL)

### **3.1 Identifying Actors and Use Cases using Textual Analysis**

In this phase of the analysis, we identify the key actors and use cases for the "ShopSmart Wiki" application through textual analysis. Actors represent the different roles or users that interact with the system, while use cases represent the various functionalities or actions that the system can perform.

Key Actors:

Guest: A guest is a non-registered user who visits the "ShopSmart Wiki" without logging in. They have limited access to certain functionalities such as browsing products and viewing product details.

Customer: The customer is a registered user who logs in to the "ShopSmart Wiki" using their credentials. They have access to additional features such as adding items to the cart, placing orders, and managing their profile.

Administrator: The administrator is a privileged user responsible for managing the "ShopSmart Wiki" platform. They have access to administrative functionalities such as adding and updating products, managing categories, and handling customer inquiries.

**Use Cases:**

Browse Products: Both guests and customers can browse through the list of products available in the "ShopSmart Wiki." This use case allows users to view product images, descriptions, and prices.

Search Products: Customers can search for specific products using keywords or filters. This use case enables users to quickly find products based on their preferences.

View Product Details: Users can view detailed information about a specific product, including specifications, customer reviews, and availability.

Register Account: A guest can create a customer account by providing the required details, such as name, email, and password.

Login: Customers can log in to their accounts using their registered email and password, granting them access to personalized features.

Add to Cart: Authenticated users can add products to their shopping cart for future purchase.

Remove from Cart: Customers can remove items from their shopping cart if they change their minds or no longer wish to purchase a particular product.

Place Order: Customers can place an order for the products in their shopping cart, initiating the payment and delivery process.

Manage Profile: Customers can view and update their profile information, such as delivery address, contact details, and payment methods.

Manage Products (Administrator): The administrator can add new products to the store, update product details, and manage product categories.

Manage Orders (Administrator): The administrator can view and manage customer orders, update order statuses, and handle order-related inquiries.

### **3.2 Forming Use Case Diagram with Candidate and Use Cases**

In this section, we provide a comprehensive overview of the functional modules and requirements of the "ShopSmart Wiki" system. These modules are designed to enhance the shopping experience for customers and streamline the overall process.

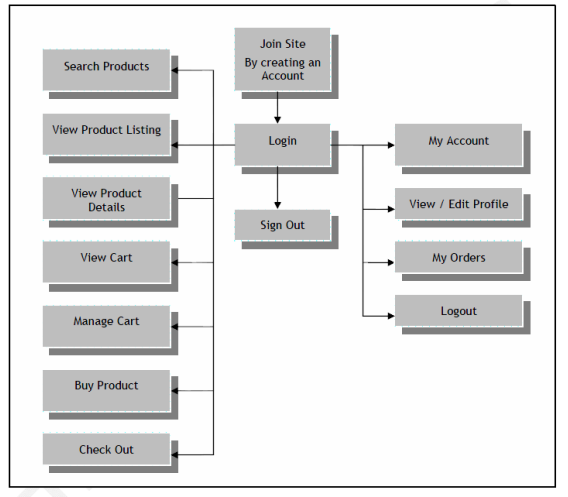


Figure 3.1 ERD Site Flow

### **3.3 Describe the Events Flow for Use Case**

**Browse Products:**

1. The user selects the "Browse Products" option from the main menu.
2. The system displays a list of products available in the store.
3. The user can scroll through the product list, view product images, descriptions, and prices.
4. The user can select a specific product to view its detailed information.
5. The use case ends.

**Search Products:**

1. The customer enters keywords or filters in the search bar and submits the search query.
2. The system processes the search query and displays relevant search results.
3. The customer can select a product from the search results to view its detailed information.
4. The use case ends.

**View Product Details:**

1. The user selects a product from the list or search results.
2. The system displays the detailed information of the selected product, including specifications and customer reviews.
3. The use case ends.

**Register Account:**

1. The guest selects the "Register" option from the main menu.
2. The system presents a registration form.
3. The guest fills in the required details, such as name, email, and password.
4. The guest submits the registration form.
5. The system validates the input and creates a new customer account.
6. The use case ends.

**Login:**

1. The customer selects the "Login" option from the main menu.
2. The system presents a login form.
3. The customer enters their registered email and password.
4. The customer submits the login form.
5. The system verifies the credentials and grants access to the customer account.
6. The use case ends.

**Add to Cart:**

1. The customer views a product and selects the "Add to Cart" option.
2. The system adds the selected product to the customer's shopping cart.
3. The use case ends.

**Remove from Cart:**

The customer views their shopping cart and selects the "Remove" option for a specific product.

1. The system removes the selected product from the customer's shopping cart.
2. The use case ends.

**Place Order:**

1. The customer selects the "Checkout" option from the shopping cart.
2. The system displays the order summary and payment options.
3. The customer selects a payment method and provides the necessary details.
4. The customer confirms the order.
5. The system processes the payment and updates the order status.
6. The use case ends.

**Manage Profile:**

1. The customer selects the "Profile" option from the main menu.
2. The system displays the customer's profile information.
3. The customer can update their profile details if needed.
4. The customer saves the changes.
5. The use case ends.

**Manage Products (Administrator):**

1. The administrator selects the "Manage Products" option from the admin panel.
2. The system displays a list of existing products and options to add new products.
3. The administrator can add new products, update product details, or delete products as needed.
4. The use case ends.

**Manage Orders (Administrator):**

1. The administrator selects the "Manage Orders" option from the admin panel.
2. The system displays a list of customer orders and their details.
3. The administrator can update the status of orders and handle customer inquiries related to orders.
4. The use case ends.

By describing the event flows for the identified use cases, we gain a clear understanding of how users interact with the "ShopSmart Wiki" application, ensuring a smooth and intuitive user experience.

# Chapter 4

# DESIGN

# 4.1 Overall Description

In this context, the "ShopSmart Wiki" project, developed using React.js and HTML, aims to provide computer science students with hands-on experience in building a comprehensive and functional web application. By offering a practical approach to application design, this project empowers students to understand the intricate interplay of client-side and server-side languages, along with database management, to create a seamless and user-centric E-commerce experience. As the E-commerce industry continues to thrive, projects like "ShopSmart Wiki" contribute to the ongoing evolution of digital commerce, ensuring that the future generation of developers is well-equipped to contribute to this dynamic and transformative sector.

E-commerce is fast gaining ground as an accepted and used business paradigm. More and more business houses are implementing web sites providing functionality for performing commercial transactions over the web. It is reasonable to say that the process of shopping on the web is becoming commonplace.

E-Commerce, a seemingly simple term had come a long way since 1991. Back in the days where technologies were still not that advance, never had anyone ever thought that E-commerce could bring the world into a new retail era where everything is almost just one click away. Essentially, E-Commerce was an abbreviation of Electronic Commerce and it was the activity of buying and selling items/products over the Internet. Technology, especially the World Wide Web (WWW) played an important role in E-Commerce whereby WWW allowed people to use E-Commerce services anytime. E-commerce became possible in 1991 when the Internet was opened to commercial use. Since that date thousands of businesses have taken up residence at web sites. (Kenneth C. Laudon, Carol G. Traver, 2008). It also overcame the geographical barriers indirectly so that people could buy and sell products from anywhere.

### 4.1.1 Product Perspective

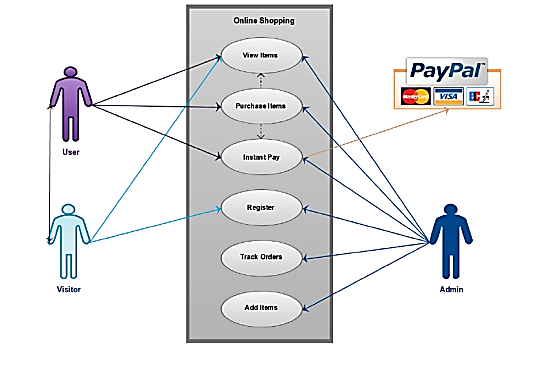
The "ShopSmart Wiki" (SS) application exists within the context of a dynamic e-commerce environment, where vendors can establish online shops to showcase and sell their products. Customers, on the other hand, have the opportunity to browse through these virtual shops and make online purchases conveniently. The system administrator plays a pivotal role in overseeing the entire operation, including approving new shop requests and maintaining a categorized list of shops. "ShopSmart Wiki" aims to provide a seamless and user-friendly experience for all stakeholders involved.

After peeking into the history and problems of E-Commerce, it was a must to perform reviews on other similar E-Commerce websites. Reviews about the strength and weaknesses of different features, functionalities allowed users to identify the problems and insufficiencies of the systems and provide solutions. A well-developed website should consist of a bunch of different useful features that ease users while they are using it. Hence, reviews on some key features of the E-Commerce website were involved in this chapter.

4.1.2 Product Features

This page consists of product details. This page appears same for both visitors and users. The products are arranged. Their description is shown when click the name of the product to see more details. After the name it has price and also add to cart icon. After clicking on add to cart icon, user goes to the next page for check out.

### 4.1.3 Design and Implementation Constraints

****

**Figure 4.1 Constraints and Implementation Constraints**

The "ShopSmart Wiki" application operates under certain constraints and assumptions:

Internet Dependence: The application is heavily reliant on internet connectivity to facilitate online shopping and data transactions.

Security Considerations: Ensuring the security of user data and financial transactions is of utmost importance to build trust with customers.

Vendor Accountability: Vendors are responsible for managing their shops and product inventory.

User Authentication: User accounts and authentication mechanisms are implemented to enhance security and personalize the user experience.

### 4.1.4 Assumption and Dependencies

The successful functioning of the "ShopSmart Wiki" application relies on the following dependencies:

Technology Stack: The MERN Stack, encompassing MongoDB, Express.js, React.js, and Node.js, forms the foundation of the application's development.

Internet Infrastructure: A stable and reliable internet connection is essential for seamless user interaction and online transactions.

User Base: The application's success depends on attracting and retaining a sizable user base of vendors and customers.

Security Protocols: Implementing robust security measures, including encryption and secure data storage, ensures user privacy and trust.

## 4.2 System Features

### 4.2.1 Order Us Page

Welcome to our Order Us page! Placing an order with us is simple and convenient. Begin by browsing our wide selection of products, and when you find something you like, click "Add to Cart." Once you've added all desired items, proceed to the checkout page. Here, you'll be prompted to enter your shipping information, choose a payment method, and review your order details. Make sure everything is correct, then click "Place Order" to complete your purchase. You will receive a confirmation email with your order details and tracking information.

### 4.2.2 Contact Us Page

Visitors and Registered users can contact website owners or administrators from here. First user or visitor should fill the form by putting name, email, subject and the message to deliver the administrator. After putting, user successfully delivered message to administrators.

### 4.2.3 About Us Page

About us page serves as an essential resource for users seeking to understand and utilize about admin. It provides detailed information about the admin and how it works or where he located. By centralizing all relevant information in one accessible location, this page helps users find answers quickly, resolve issues independently, and make informed decisions. Whether you are setting up for the first time, looking to optimize your usage, or need support with specific features, our Documentation page is your go-to guide for all things related to our offerings. Its importance lies in empowering you with the knowledge and tools needed to succeed, enhancing your overall experience with our website.

### 4.2.4 Login Page

Login page for both users and administrators. Put the email in email field and after that enter the password of your account. After clicking on sign in button, user if put wrong details then not getting login and when user put correct details then login the account. Same like user, administrators login page works same.

### 4.2.5 Sign up

## Welcome to our Sign Up page! Creating an account with us is quick and easy. Simply fill in your username, email address, and create a secure password to get started. By signing up, you'll enjoy a personalized shopping experience, faster checkout, and access to exclusive deals and promotions. Plus, you can easily track your orders and manage your preferences. Rest assured, your information is safe with us, and we are committed to protecting your privacy. Join our community today and start enjoying the many benefits of being a member!

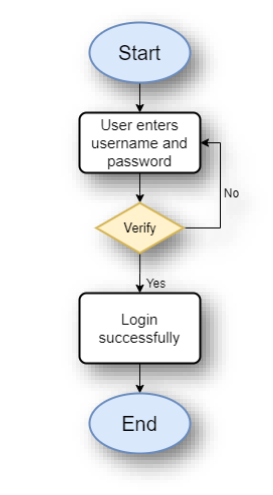
4.2.6 Registration

The "ShopSmart Wiki" system requires customers to register before they can access the shopping cart and make purchases. During the registration process, customers provide necessary information, such as their name, email address, and password, to create a unique user ID. Registration is a one-time process that enables personalized interactions and a secure login for future visits. By requiring registration, the system can maintain customer profiles, track order history, and offer tailored promotions or discounts based on user preferences.

### 4.2.7 Login

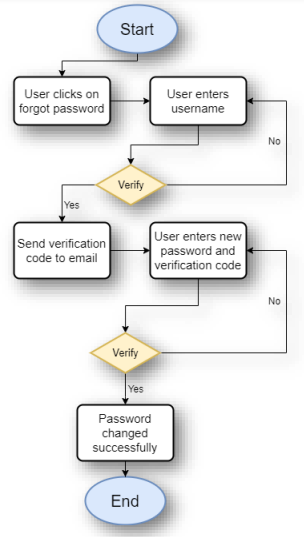
Once registered, customers can log in to the "ShopSmart Wiki" system by entering their valid user ID and password. The login process serves as an authentication step to ensure only authorized users can access the system's functionalities. Upon successful login, customers gain full access to their shopping cart, product details, and order history. The secure login mechanism safeguards user data and prevents unauthorized access, ensuring a safe and reliable shopping experience.

The flow of a user logging in. It was a simple login flow that followed the general step. User could only log in with username and password but not email. Methods were called to verify the user’s input and display whether or not he or she login successfully.

****

**Figure 4.2 Login Flow Chart**

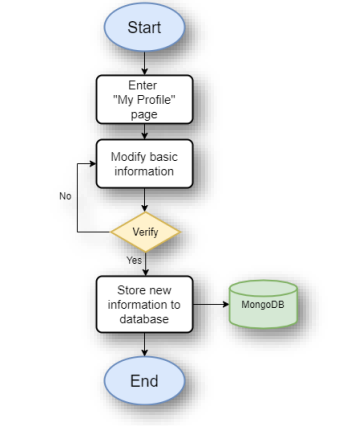
### 4.2.8 Reset Password Flow Chart

****

**Figure 4.3 Reset Password Flow Chart**

The picture above showed the steps of resetting the password of a user. The user was required to enter his or her username so that a function would be called to check if the username was existed or not.

### 4.2.9 Edit Profile Flow Chart

****

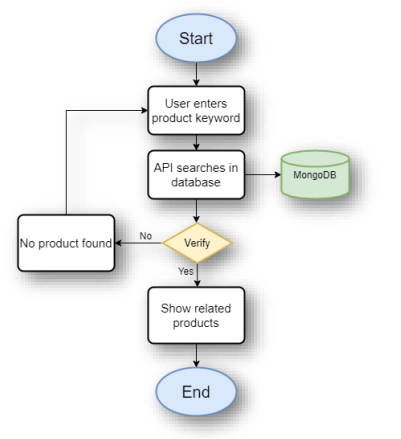
**Figure 4.4 Edit Profile Flow Chart**

The Figure no. above showed the steps of editing the profile information of a user. After the user clicked into the “My Profile” page, he or she could modify some basic information such as profile picture and display name of his or her account. After verification, newly modified data would be stored in the online database.

### 4.2.10 Changes to Cart

After logging in or registering, customers have the flexibility to make changes to their shopping cart. They can easily add products to the cart, remove items they no longer wish to purchase, and update quantities as needed. The dynamic cart management feature allows users to review their selected products and make adjustments before proceeding to checkout.

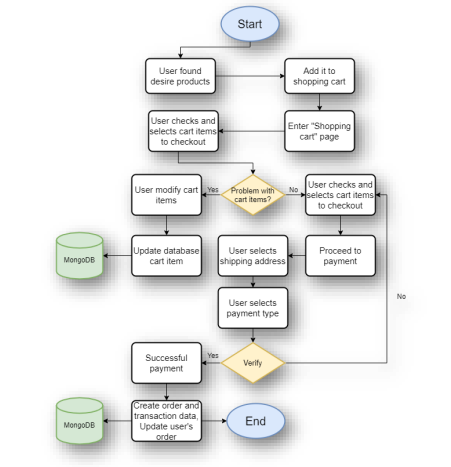
### 4.2.11 Search Product Flow Chart

****

**Figure 4.5 Search Product Flow Chart**

The picture above depicted the flow of searching for a product on this website. When the user enters a searching term in the search bar provided, an API method would be called to perform searching in the online database for matching product names.

### 4.2.12 Add Product to Cart and Checkout Flow Chart

****

**Figure 4.6 Add Product to Cart and Checkout Flow Chart**

The Figure above showed the flow of a user adding products into their shopping cart and proceeding to checkout. Firstly, users found desire products and add them to the shopping cart. The user was allowed to select or deselect items in the cart before checking out. If the user modified the amount in the shopping cart, data in the online database would be updated on the spot. If the user clicked checkout with selected items, it would proceed to select the shipping address. Next step, the user would be required to select payment type but for now the website only offers the “stripe” payment type. After payment verification successful, order and transaction data would be created in the database and the user’s data would be updated too.

### 4.2.13 Payment

Currently, the "ShopSmart Wiki" system supports cash payments for product purchases. Customers can choose cash on delivery (COD) as their preferred payment method during checkout. However, the system is designed to accommodate future enhancements, including the integration of credit cards, debit cards, and other online payment options. By offering various payment methods, "ShopSmart Wiki" aims to cater to a broader range of customers, accommodating different payment preferences and enhancing convenience.

### 4.2.14 Logout

Upon completing their shopping or browsing activities, customers have the option to log out of the "ShopSmart Wiki" system. Logging out ensures the security of their accounts and protects sensitive information from unauthorized access. This feature encourages responsible use of the system and allows users to maintain control over their online presence within the platform.

### 4.2.15 Report Generation

After customers place their orders, the "ShopSmart Wiki" system generates an order summary for each transaction. One copy of the bill is sent to the customer's provided email address for their reference and records. This facilitates easy access to order details and simplifies the tracking of past purchases. Simultaneously, a copy of the bill is stored in the system's database for future record-keeping and order management. The report generation feature enhances transparency, facilitates order tracking, and enables efficient customer service.

These system features are strategically designed to provide a seamless and secure shopping experience for customers using the "ShopSmart Wiki" platform. From registration and login to cart management, payment options, and order reporting, each module contributes to the overall efficiency and user satisfaction within the e-commerce system. As "ShopSmart Wiki" continues to evolve, additional features may be implemented to further enhance its functionality and meet customer needs effectively, facilitates order tracking, and enables efficient customer service.

### 4.2.16 Admin Dashboard

Creating an admin dashboard that allows for adding, updating, viewing, and deleting products, users, orders, and admins typically involves building a web application with a backend and a frontend. Below, I'll outline a general approach to developing such a dashboard. Keep in mind that this is a high-level overview, and the actual implementation may vary depending on your preferred programming language, framework, and technology stack.

1. Backend Development: Choose a backend technology stack that suits your needs. Common choices include Node.js with Express, Python with Django or Flask, Ruby with Ruby on Rails, or Java with Spring Boot.

a. Data Models: Define the data models for products, users, orders, and admins. Each of these entities will have specific attributes and relationships with other entities. For example:

* + Product: name, description, price, category, etc.
  + User: name, email, password, role, etc.
  + Order: order details, user ID (foreign key), products, etc.
  + Admin: name, email, password, etc.

b. API Endpoints: Create API endpoints to handle CRUD (Create, Read, Update, Delete) operations for each entity. For example:

* + POST /api/products: Add a new product.
  + GET /api/products: Fetch a list of all products.
  + PUT /api/products/:id: Update a specific product.
  + DELETE /api/products/:id: Delete a specific product.
  + Similarly, create endpoints for users, orders, and admins.

c. Authentication and Authorization: Implement authentication and authorization mechanisms to secure the dashboard. Only authenticated admins should have access to perform CRUD operations. You can use JSON Web Tokens (JWT) or sessions for authentication.

1. Frontend Development: Choose a frontend technology stack like React, Angular, or Vue.js.

a. Dashboard Layout: Design the dashboard layout with sections for products, users, orders, and admins. Each section should have tables, forms, or cards to display and interact with the data.

b. API Integration: Utilize the backend API endpoints you created to fetch, add, update, and delete data from the frontend. Use appropriate HTTP methods (GET, POST, PUT, DELETE) to perform the corresponding actions.

c. User Interface: Implement user-friendly interfaces for adding, updating, and deleting products, users, orders, and admins. Use form validations to ensure data integrity.

d. Authentication Handling: Implement the login functionality for admins to access the dashboard. After successful authentication, store the JWT or session token and include it in subsequent API requests to authenticate with the backend.

1. Database: Choose an appropriate database system (e.g., MySQL, PostgreSQL, MongoDB) to store the data. Create the necessary tables/collections to store products, users, orders, and admins.
2. Security Considerations: Security is essential for an admin dashboard. Ensure that you sanitize user inputs to prevent SQL injection and other vulnerabilities. Also, implement role-based access control to restrict certain actions to authorized users only.
3. Testing: Thoroughly test the dashboard to identify and fix any bugs or issues. Use unit tests, integration tests, and end-to-end tests to verify the functionality of the application.
4. Deployment: Deploy your backend and frontend to a web server or cloud platform (e.g., AWS, Heroku, Azure) to make the admin dashboard accessible to authorized users.

Remember that this is just an outline, and the actual implementation may involve more specific details based on your project requirements and the technologies you choose. Additionally, always prioritize security and data integrity when developing an admin dashboard, especially when dealing with sensitive data.

### 4.2.17 Dashboard List

We can control all the information/data about our website like we can insert a new data from admin panel and delete that data about product which are not available in our shop/market. And our admin panel must be secure in which only two admin can access to our admin area and edit / delete the data. And we have user’s information in our admin area So that we can easily check the user’s detail. Admin dashboard that allows for adding, updating, viewing, and deleting products, users, orders, and admins typically involves building a web application with a backend and a frontend.

### 4.2.18 Order

## The "Order ID" field uniquely identifies each order placed by customers. This field is crucial for tracking and managing individual orders throughout the fulfillment process. The "Customer Username" field associates each order with the specific user who placed it, allowing for personalized customer service and account management. The “Ordered Date” field associates each order date, when was it ordered. The "Order Status" field provides real-time updates on the progress of each order, indicating stages such as pending, processing, shipped, and completed. The “Amount” field associates the amount of product. These fields collectively ensure efficient order management and a seamless customer experience.

### 4.2.19 View Order Detail

## We have order id which is rare id give to order to get information about order. The status in it will tell us about the order, it shows order delivered or active or cancelled.

### 4.2.20 Products

Welcome to our Products page! Here, you will find a diverse selection of high-quality items carefully curated to meet all your needs and preferences. From the latest tech gadgets and stylish apparel to home essentials and unique gifts, our extensive range ensures there's something for everyone. Each product is accompanied by detailed descriptions, and high-resolution images to help you make informed decisions. Our commitment to quality and customer satisfaction means that every item is sourced from trusted brands and thoroughly tested for excellence. Explore our collection and discover the perfect products to enhance your lifestyle today!

### 4.2.21 Add Product

## Welcome to the Add Product page for administrators! This page is designed to streamline the process of adding new products to our inventory. To add a product, simply fill out the form with all required details, including the product name, description, price, and category. Upload high-quality images to showcase the product effectively. You can also specify inventory levels, shipping options, and any promotional tags. Ensure all information is accurate and comprehensive to provide customers with the best shopping experience. Once you've reviewed everything, click "Save" to add the product to our catalog. This page is your gateway to keeping our offerings fresh and up-to-date.

### 4.2.22 User

Welcome to the User Management page for administrators! This page provides a comprehensive overview of all users who have signed up on our website. As an administrator, you can view detailed information about each user, including their registration date and account status (active or inactive). You have the authority to add new users manually or remove existing ones as needed. This page also allows you to update user details and manage their access levels. By maintaining accurate and up-to-date user records, you can ensure a smooth and secure experience for all members of our community. Thank you for managing our user base with diligence and care!

### 4.2.23 Add User

## Welcome to the Add User page for administrators! This page allows you to easily add new users to our platform. To create a new user account, simply fill in the required fields: username, email address, and a secure password. Ensure that the username is unique and the email address is valid. After entering the information, you can set the user's initial status (active or inactive) and assign any relevant roles or permissions. Once all details are correctly filled out, click "Save" to add the new user to our system. This streamlined process helps you efficiently manage user registrations and maintain an up-to-date user database.

### 4.2.24 Admin List

## Welcome to the Admin List page! This page provides a detailed overview of all administrators managing our platform. Each entry includes essential information such as the admin's username, email address, role, and status (active or inactive). As a super administrator, you have the authority to add new admins, update existing admin details, and remove admins if necessary. This page ensures transparency and efficient management of administrative privileges, helping maintain a secure and well-organized system. By keeping the admin list current and accurate, you contribute to the effective governance and smooth operation of our platform.

### 4.2.25 Add Admin

## Welcome to the Add Admin page! This page allows you to efficiently add new administrators to our platform. To create a new admin account, simply fill in the required fields: username, email address, and a secure password. You can also assign a specific role or permission level to the new admin, ensuring they have the appropriate access needed for their responsibilities. Once all details are accurately filled out, click "Save" to add the new admin to our system. This streamlined process helps you maintain a robust and well-managed administrative team.

# Chapter 5

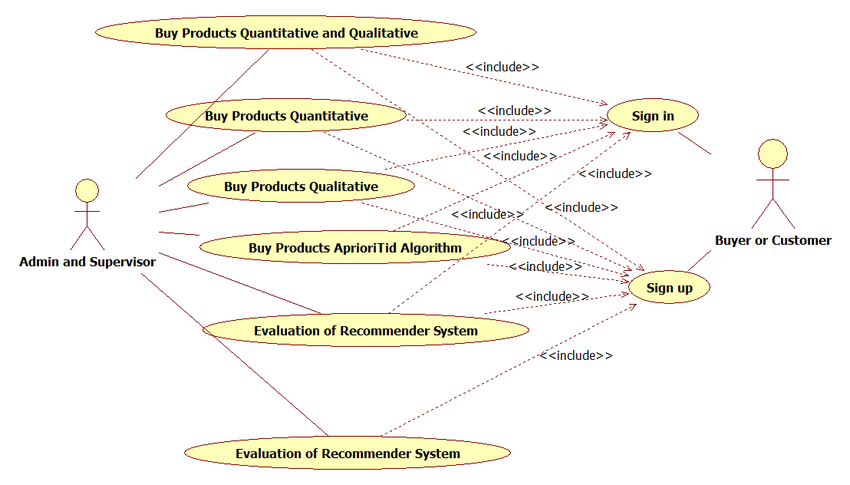
# IMPLEMENTATION

Our e-commerce website leverages the MERN stack, which comprises MongoDB, Express.js, React.js, and Node.js. This stack is chosen for its ability to handle dynamic content, provide a responsive user interface, and support a scalable backend. The implementation process is broken down into several key components, each addressing a critical aspect of the application, from database design to frontend development and API integration.

### **5.1 Component Diagram**

A Component Diagram is a type of structural diagram that shows the organization and dependencies of the components in a system. It provides a high-level view of the system's architecture and illustrates how the various components interact with each other.

In the context of the "ShopSmart Wiki" project, the Component Diagram would showcase the major software components and their relationships. The diagram would represent the key modules and libraries used in the frontend (React.js) and backend (Node.js and Express.js) of the application. Additionally, it would illustrate how these components communicate with each other to achieve the overall functionality of the system.

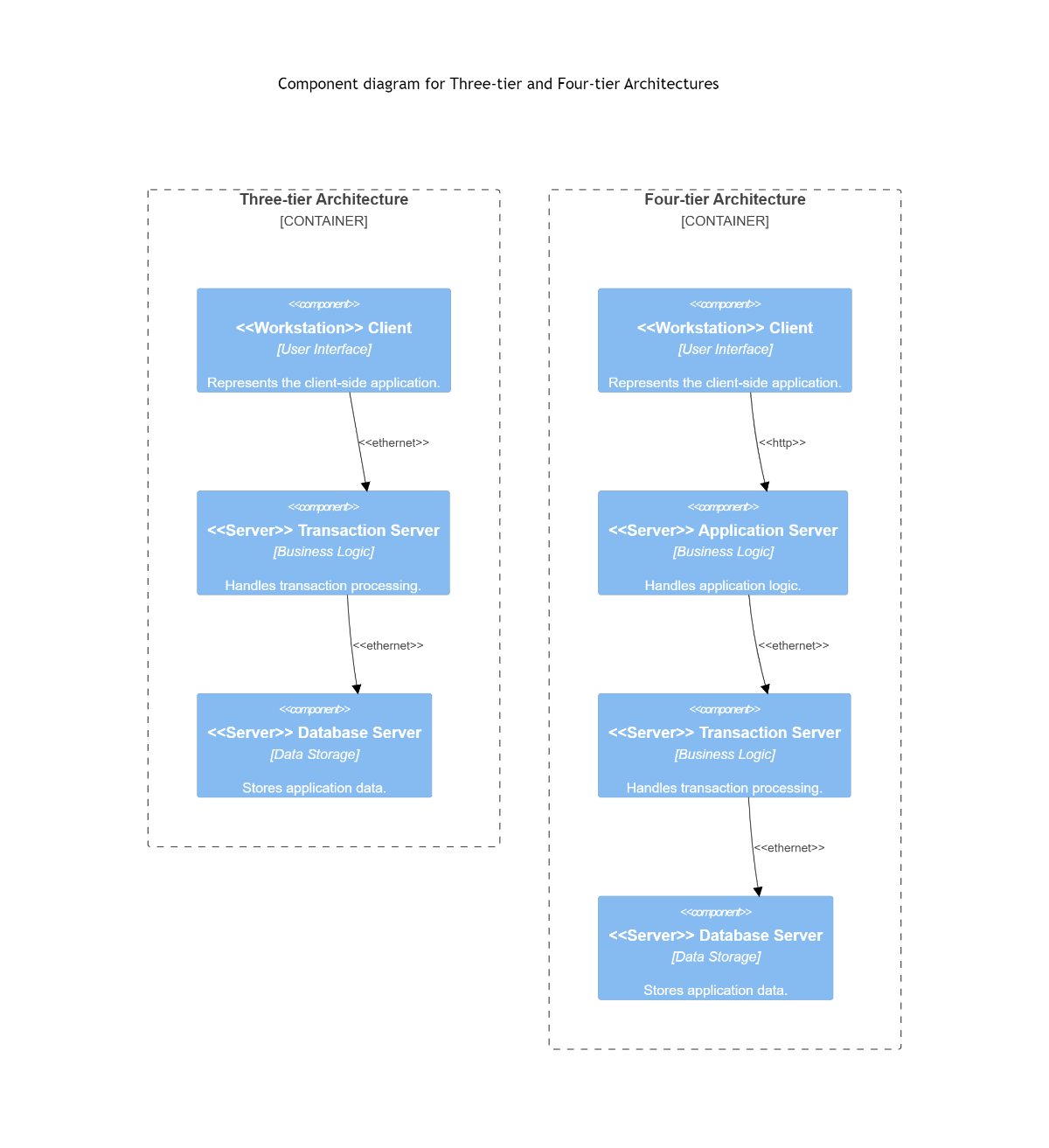
****

**Figure 5.1 Component Diagram**

### **5.2 Deployment Diagram**

A Deployment Diagram is a type of structural diagram that models the physical deployment of software components on hardware nodes. It illustrates the distribution and configuration of the software components across different nodes in the system.

In the "ShopSmart Wiki" project, the Deployment Diagram would depict how the frontend (React.js) and backend (Node.js and Express.js) components are deployed on servers or cloud-based platforms. It would show the relationship between the software components and the hardware nodes they run on. Additionally, it would indicate the connections and communication channels between the frontend and backend components.



**Figure 5.2 Deployment Diagram**

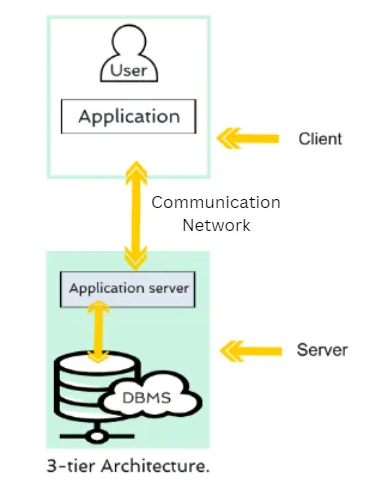
### **5.3 Database Architecture (1- Tier, 2-Tier, 3- Tier Architecture)**

Database Architecture refers to the arrangement and configuration of databases and their management within an application. It outlines how data is stored, accessed, and managed by the application.

In the context of the "ShopSmart Wiki" project, the Database Architecture is likely a 3-Tier Architecture. This means that the application follows a three-layered approach in handling data:

1. **Presentation Layer:** The frontend (React.js) represents the Presentation Layer, which handles user interactions and displays information from the application.
2. **Application Layer:** The backend (Node.js and Express.js) represents the Application Layer, which contains the business logic and processes data requests from the frontend. It mediates between the Presentation Layer and the Data Layer.
3. **Data Layer:** The database (MongoDB) represents the Data Layer, where data related to products, user accounts, and orders are stored. The backend interacts with the database to retrieve, update, and store data.

This 3-Tier Database Architecture provides a clear separation between the different layers, enabling modularity and scalability. It ensures that data management is centralized in the database, while the application logic and user interface are managed in their respective layers. This separation of concerns makes the application easier to maintain and expand in the future.



**Figure 5.3 Database Architecture Diagram**

# Chapter 6

# TESTING (SOFTWARE QUALITY ATTRIBUTES)

### **6.1 Test Case Specification**

Test Case Specification involves creating detailed test cases that describe the specific inputs, actions, and expected outcomes for testing various functionalities of the "ShopSmart Wiki" application. Each test case is designed to validate a specific aspect of the software to ensure that it meets the required quality attributes and functions as expected.

### **6.2 Black Box Test Cases**

Black Box Testing is a testing technique where the tester has no knowledge of the internal code or structure of the software. The focus is on testing the software's functionality based on its specifications and requirements. In the context of the "ShopSmart Wiki" project, several Black Box Test Cases are performed:

6.2.1 BVA or Boundary Value Analysis

Boundary Value Analysis is a test design technique that involves testing values at the boundaries of input domains. The objective is to ensure that the application handles boundary values correctly. Test cases are designed using values just above, at, and just below the boundary limits.

Example Test Case:

* Verify if the system correctly handles the minimum and maximum quantity of products that a customer can order (e.g., 1 and 100). Test cases may include ordering one product, ordering 100 products, and trying to order 0 or negative quantities.

6.2.2 Equivalence Class Partitioning

Equivalence Class Partitioning is a test design technique used to group similar test cases into classes to reduce the number of test cases required while ensuring adequate test coverage. It involves selecting representative values from each equivalence class.

Example Test Case:

* Verify if the system handles valid and invalid user logins correctly. Test cases may include valid login credentials, invalid usernames, and invalid passwords.

6.2.3 State Transition Testing

State Transition Testing is a test design technique used to verify if the application transitions between different states correctly. It focuses on testing the application's behavior when moving from one state to another.

Example Test Case:

* Verify if the system transitions correctly from the "Browse Products" state to the "View Product Details" state when a product is selected. Test cases may include transitioning from different states based on user interactions.

6.2.4 Decision Table Testing

Decision Table Testing is a test design technique used to validate combinations of inputs and their corresponding outcomes. It helps identify all possible scenarios and ensures that the application handles each scenario appropriately.

Example Test Case:

* Verify if the system calculates the correct total amount for an order with different combinations of products, quantities, and prices. Test cases may include different combinations of products and quantities to cover all possible scenarios.

6.2.5 Graph-Based Testing

Graph-Based Testing is a test design technique where the software's behavior is modeled as a graph, and test cases are designed based on the graph's paths to achieve maximum test coverage.

Example Test Case:

* Verify if the system correctly handles different navigation paths within the application, such as browsing products, adding to the cart, and placing orders. Test cases may include navigating through different paths and verifying the expected outcomes.

### **6.3 White Box Testing**

White Box Testing is a testing technique where the tester has access to the internal code and structure of the software. The focus is on ensuring that all possible code paths are executed and that the code functions as expected. In the "ShopSmart Wiki" project, the following White Box Testing techniques are employed:

6.3.1 Statement Coverage

Statement Coverage measures the percentage of executable statements in the code that are executed during testing. It ensures that each statement is tested at least once.

6.3.2 Branch Coverage

Branch Coverage measures the percentage of branches (decision points) in the code that are executed during testing. It ensures that all possible branches in the code are tested.

6.3.3 Path Coverage

Path Coverage measures the percentage of unique paths through the code that are executed during testing. It ensures that all possible paths in the code are tested.

By using a combination of Black Box and White Box Testing techniques, the "ShopSmart Wiki" project can ensure comprehensive testing coverage, identify potential defects, and deliver a high-quality, reliable, and efficient software product to its users.

# Chapter 7

# TOOL & TECNOLOGIES

The development and implementation of the "ShopSmart Wiki" e-commerce platform require a variety of tools and technologies to ensure efficiency, scalability, and security. In this chapter, we explore the programming languages, development frameworks, and operating systems employed in building and operating the system.

### 7.1 Programming Languages

**JavaScript:** JavaScript is the core programming language used extensively in the "ShopSmart Wiki" platform. It is employed for both frontend and backend development. On the frontend, JavaScript, along with React.js library, enables the creation of interactive and dynamic user interfaces. On the server-side, Node.js, a runtime environment for executing JavaScript, is utilized to develop the backend components, allowing for efficient handling of server-side requests and event-driven programming.

**HTML & CSS:** HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets) are fundamental technologies for building web pages and defining their appearance. The "ShopSmart Wiki" user interfaces are designed using HTML to structure content and CSS for layout and styling, ensuring a visually appealing and responsive user experience.

Development Frameworks

**React.js:** React.js is a widely used JavaScript library for building user interfaces. It allows for the creation of reusable UI components, facilitating the development of a dynamic and responsive frontend for "ShopSmart Wiki" React.js enables efficient rendering and updating of components, leading to a smooth and interactive user experience.

**Node.js:** Node.js serves as the backend framework for "ShopSmart Wiki" It offers a scalable and non-blocking I/O architecture, allowing the system to handle a large number of concurrent user requests efficiently. Node.js provides an event-driven model that aligns well with the asynchronous nature of web applications, making it ideal for building real-time and high-performance server-side applications.

**Express.js:** Express.js is a lightweight and flexible web application framework for Node.js. It simplifies the development of RESTful APIs, routing, and middleware integration. Express.js streamlines the backend development process in "ShopSmart Wiki" by providing a robust and organized structure.

### 7.2 Operating Environment

The "ShopSmart Wiki" platform runs on Windows-based operating systems. While Linux is a popular choice for web servers and hosting environments, Windows provides a suitable environment for hosting Node.js applications like "ShopSmart Wiki" Windows Server editions are well-known for their stability and security features, making them a viable option for deploying web applications.

MongoDB: MongoDB, a NoSQL database, is chosen as the database management system for "ShopSmart Wiki" It offers flexibility in handling unstructured data and scalability to manage a vast amount of product information, user profiles, and order details. MongoDB's JSON-like document format enables efficient data retrieval and storage, making it suitable for e-commerce applications with rapidly changing data.

Kickstarting a project with proper development tools is very important. It provides developers effective features to create, debug, maintain and support during the development process. In this project, the software used was Visual Studio Code 2019 and it was used to edit and manage codes written in different languages. Besides, the built-in terminal of Visual Studio code was also used to test and debug the project.



**Figure 7.1 VS code**

Minimum requirements for hardware:

- Window 7 SP1 (with latest Windows Update)

- 1.8 GHz Processor

- 2GB RAM

- 800 MB up to 210 GB of memory

The deployment of the "ShopSmart Wiki" project on Vercel is a straightforward and efficient process, leveraging the platform's seamless integration with the MERN stack and Next.js framework. Vercel's simplicity, scalability, and performance-oriented features make it an ideal choice for hosting and serving web applications.

Before deploying the "ShopSmart Wiki" the project is thoroughly prepared for production. The frontend, built with React.js and Next.js, undergoes optimization to create a production-ready build. This build includes all the necessary assets, while minimizing file sizes to enhance loading speed and overall user experience.

Similarly, the backend, developed using Node.js and Express.js, is configured to handle server-side operations and API requests. The connection to the MongoDB database is established, allowing seamless storage and retrieval of product data, user information, and order details.

With the frontend and backend components ready, the deployment process begins on Vercel:

Version Control Integration: The "ShopSmart Wiki" project is integrated with a version control system, such as Git. Vercel seamlessly connects to the Git repository, which allows for automated deployments triggered by new code commits.

Continuous Deployment: Vercel's continuous deployment feature ensures that the project remains up-to-date with the latest changes in the codebase. Whenever new code is pushed to the Git repository, Vercel automatically initiates a build process.

Build and Optimization: During the deployment process, Vercel generates optimized builds of the frontend and backend. The frontend assets are minified, bundled, and served with cache control headers, reducing load times for end-users.

Serverless Architecture: Vercel's serverless architecture enables automatic scaling of the application based on demand. This ensures that the "ShopSmart Wiki" can handle fluctuations in traffic and remains responsive, even during peak usage.

Global Content Delivery Network (CDN): Vercel leverages its global CDN to distribute the "ShopSmart Wiki" to edge locations across the world. The CDN caches static assets and serves them from the nearest edge location to users, reducing latency and improving performance.

Security: Vercel prioritizes security, providing HTTPS encryption by default for all deployments. This ensures that user data, login credentials, and payment information are protected during transmission.

Monitoring and Analytics: Vercel offers monitoring and analytics tools to track the performance of the "ShopSmart Wik" Developers can gain insights into website traffic, user behavior, and error tracking, enabling continuous improvements.

Next, the backend, built on Node.js and Express.js, is configured to handle server-side operations and API requests. The connection to the MongoDB database is established to store and retrieve product data, user information, and order details.

With the frontend and backend ready, the "ShopSmart Wiki" project is deployed on Vercel with a few simple steps. The seamless integration of Vercel with Git allows for effortless continuous deployment. Any updates pushed to the Git repository trigger an automatic build and deployment process, ensuring that the latest version of the project is always live.

Vercel's server less architecture ensures optimal resource utilization, automatically scaling the application based on demand. This scalability guarantees that the "ShopSmart Wiki" can handle varying traffic levels without any performance degradation.

Furthermore, Vercel's global content delivery network (CDN) ensures low latency and fast loading times for users across the globe. The CDN caches static assets and serves them from edge locations closest to the users, minimizing the time taken to load the web application.

Vercel's robust security features protect the "ShopSmart Wiki" from potential threats and vulnerabilities. With HTTPS enabled by default, user data and payment information are encrypted, ensuring a secure shopping environment.

In conclusion, deploying the "ShopSmart Wiki" project on Vercel empowers it with the capabilities of a world-class serverless platform. The seamless integration of the MERN stack with Vercel unleashes the full potential of this E-commerce application, providing users with a smooth and engaging shopping experience. The combination of React.js, Next.js, Node.js, Express.js, MongoDB, and Vercel creates a powerful synergy that makes the "ShopSmart Wiki" a standout project in the realm of modern web development and E-commerce. Git: Git, a distributed version control system, is used for collaborative development and version management of the "ShopSmart Wiki" source code. It allows multiple developers to work on the project simultaneously, track changes, and merge code efficiently.

Webpack: Webpack is employed to bundle and optimize frontend assets, such as JavaScript, CSS, and images. It helps reduce loading times and enhances the performance of the "ShopSmart Wiki" web application.

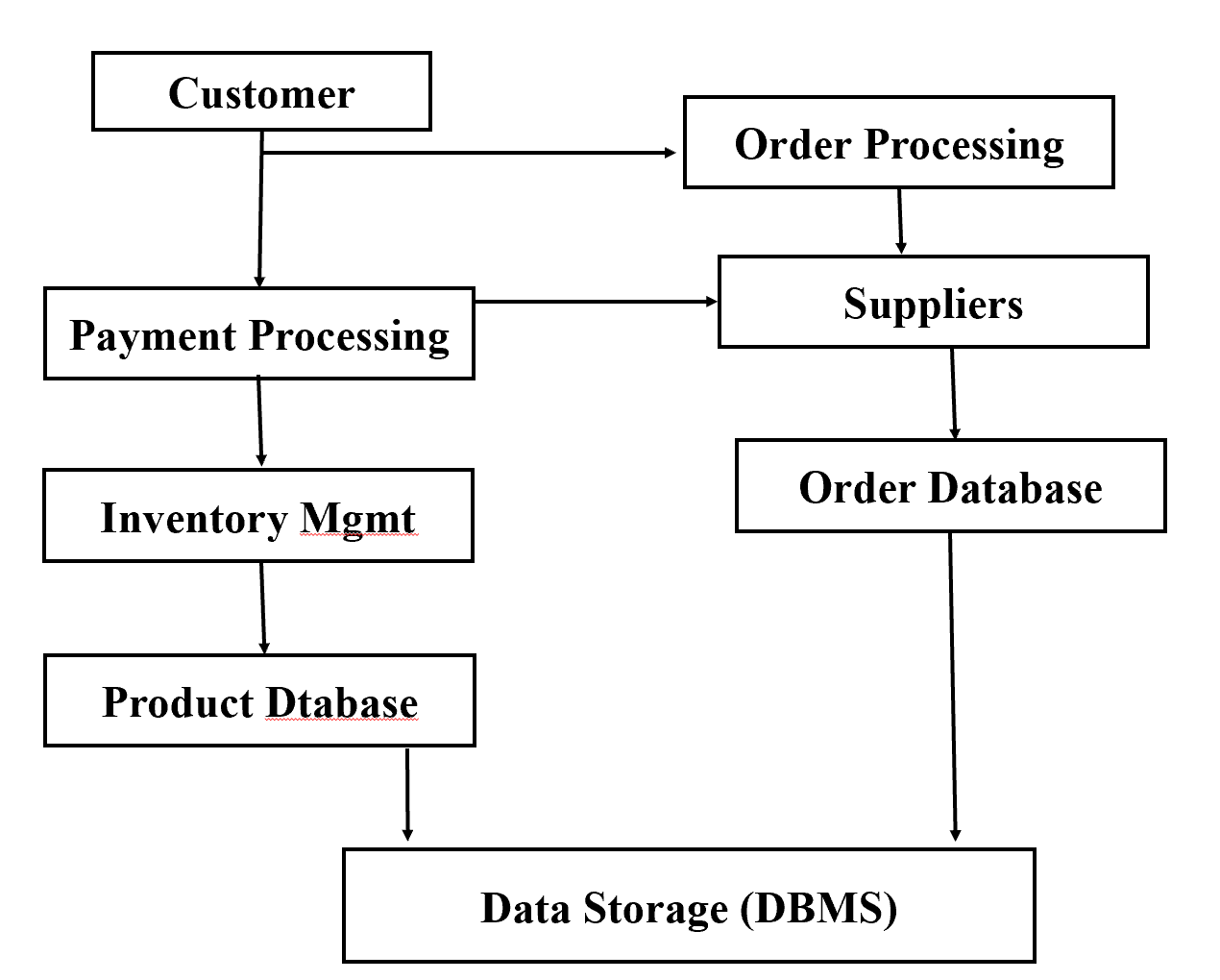
**Table 7.1 Use Cases**

|  |  |  |
| --- | --- | --- |
| **Use Case** | **Actor** | **Goal** |
| Register as a User | Customer | To create an account on the platform. |
| Login | Customer | To log in to their account on the platform. |
| Browse Products | Customer | To browse the products on the platform. |
| Add Product to Cart | Customer | To add a product to their shopping cart. |
| Checkout | Customer | To checkout and complete their purchase. |
| Track Order | Customer | To track the status of their order. |
| Manage Account Settings | Customer | To manage their account settings, such as their shipping address and payment information. |
| Contact Customer Support | Customer | To contact customer support for help with any issues they may have. |

A data flow diagram (DFD) is a graphical representation of the flow of data through a system. It is a tool for visualizing and documenting the information that is processed by a system. DFDs are used in software engineering, systems analysis, and business process modeling. Level 0 DFDs, also known as context diagrams, are the simplest type of DFD. They show the entire system as a single process, with its relationship to external entities. External entities are anything that interacts with the system, such as users, other systems, or databases. Level 1 DFDs are a more detailed view of the system. They break down the single process in the level 0 DFD into sub-processes. Each sub-process is represented as a separate box on the DFD. The data flows between the sub-processes are also shown.

**Level 0:**

The customer and supplier are external entities that interact with the system. The order processing process handles customer orders, the inventory management process manages the product inventory, and the payment processing process handles customer payments. The customer database, product database, and order database store data about customers, products, and orders.

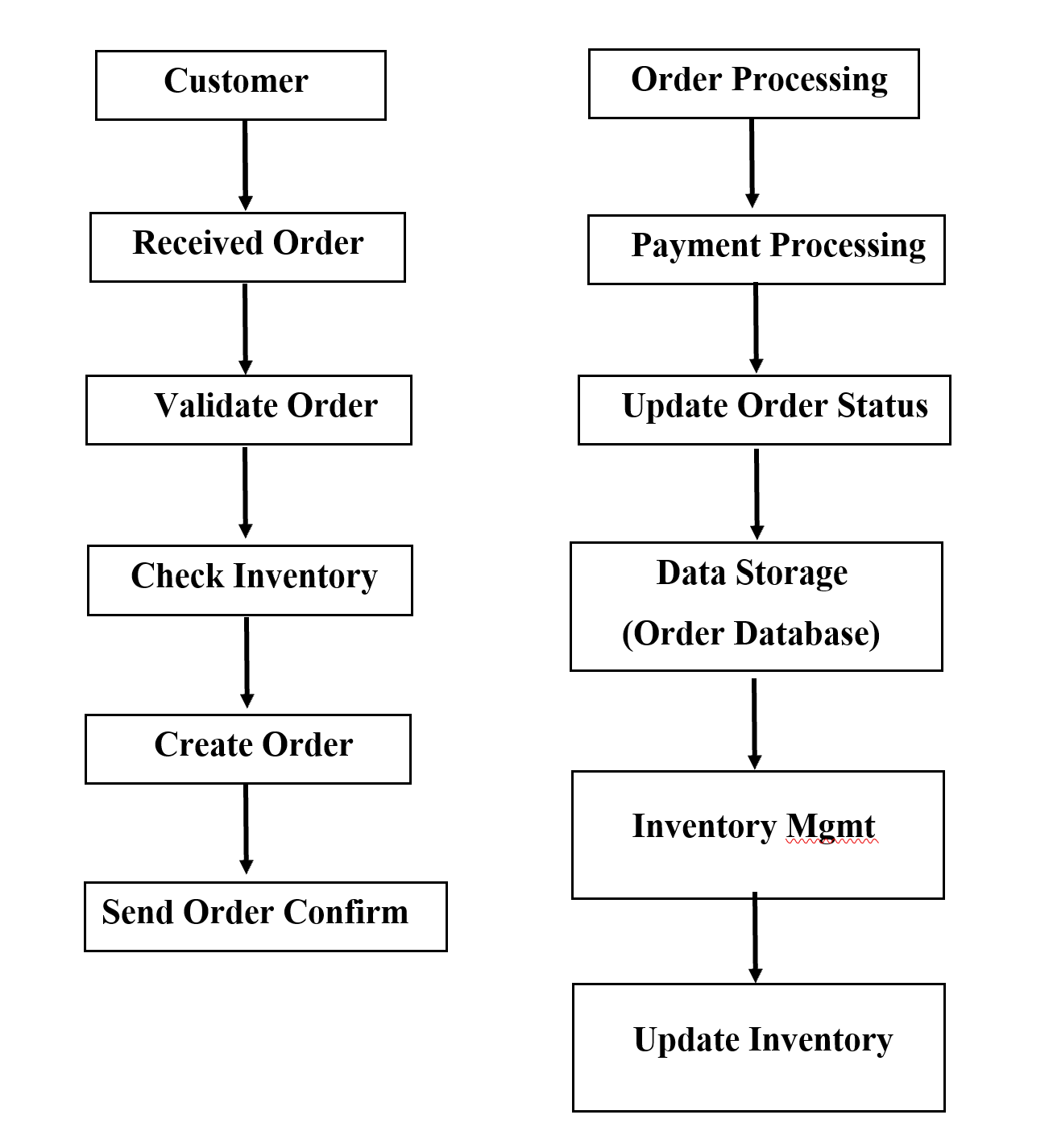


**Fig 7.2 Data Flow Diagram L0**

In this representation, the "Data Storage (DBMS)" entity is shown separately, indicating that it houses various databases like the "Customer Database," "Product Database," and "Order Database." Each entity is connected with arrows depicting the flow of information or processes between them.

**Level 1:**

The receive order process receives customer orders. The validate order process validates customer orders. The check inventory process checks the product inventory to see if the products ordered are in stock. The create order process creates a new order in the order database. The send order confirmation process sends an order confirmation to the customer.



**Fig 7.3 Data Flow Diagram L1**

* The "Customer" is an external entity that interacts with the "Order Processing" system.
* The "Order Processing" system has several main sub-processes:
* "Receive Order": Receives the order details from the customer.
* "Validate Order": Validates the received order.
* "Check Inventory": Checks the availability of products in the inventory.
* "Create Order": Creates the order based on validated details and available inventory.
* "Send Order Confirmation": Sends the order confirmation to the customer.
* The "Payment Processing" sub-process handles payment-related tasks.
* The "Update Order Status" sub-process updates the order status after successful payment processing.
* The "Data Storage (Order Database)" is a data store that holds order-related information.
* The "Inventory Management" sub-process handles inventory-related tasks and updates the inventory accordingly.

System implementation for the Ishamsha store website project involves the actual development and deployment of the website according to the design specifications. It is the process of transforming the conceptual and logical design into a fully functional website that can be accessed by users. Here is an introduction to system implementation for the Ishamsha store website project:

Development Environment Setup

Set up the development environment, including the necessary software tools, frameworks, and libraries for web development.

Choose the appropriate programming languages, such as HTML, CSS, JavaScript, and backend languages like PHP, Python, or Node.js.

**Frontend Development**

Develop the user interface (UI) components of the website, focusing on creating visually appealing and responsive designs.

Implement the UI elements using HTML, CSS, and JavaScript, ensuring compatibility across different web browsers and devices.

**Backend Development**

Build the backend components that handle the business logic and data processing of the website.

Develop server-side functionalities, such as user authentication, database interactions, and API integrations.

Use appropriate programming languages and frameworks for backend development, ensuring robustness, security, and scalability.

**Database Setup**

Set up the database system (e.g., MySQL, PostgreSQL, MongoDB) and create the necessary database tables and relationships based on the database design. Implement database queries and interactions to store and retrieve data efficiently.

**Conclusion**

The "ShopSmart Wiki" project represents a significant milestone in the realm of web development and E-commerce. The project's implementation using the MERN stack (MongoDB, Express.js, React.js, and Node.js) and deployment on Vercel showcases a powerful combination of cutting-edge technologies, delivering a feature-rich and user-friendly online shopping platform. The global shift towards electronic commerce, commonly known as E-commerce, has revolutionized the way businesses and consumers interact in the digital age. The "ShopSmart Wiki" aligns perfectly with this trend, offering a convenient and efficient solution for buying and selling products or services over electronic systems, primarily the internet. Through this project, computer science students gain valuable hands-on experience in application design, starting from the fundamentals of React.js and HTML. They learn the essential concepts of user interface design, frontend development, and backend operations using Node.js and Express.js. This immersive learning experience equips students with practical skills that are highly sought after in the ever-evolving world of web development.

Moreover, the "ShopSmart Wiki" project's deployment on Vercel showcases the seamless integration of modern technologies with a serverless and scalable deployment platform. Vercel's simplicity, security, and global content delivery network (CDN) ensure that the application runs efficiently and delivers an exceptional user experience. The platform's continuous deployment feature allows for rapid updates and improvements, keeping the " ShopSmart Wiki" at the forefront of E-commerce innovation.

The use of Next.js, a powerful framework built on top of React.js, enhances the frontend development, enabling highly interactive and dynamic user interfaces. The integration with MongoDB, a flexible NoSQL database, empowers the backend to handle vast amounts of data efficiently, ensuring a smooth and seamless online shopping experience.

The "ShopSmart Wiki" project not only serves as an educational tool for aspiring web developers but also embodies the potential for real-world applications. As businesses worldwide recognize the immense value of digital commerce, projects like the "ShopSmart Wiki" offer a blueprint for creating robust, scalable, and secure E-commerce platforms.

In conclusion, the "ShopSmart Wiki" project epitomizes the intersection of technology and commerce, where innovation and user-centric design come together to create a seamless online shopping experience. As the world continues to embrace electronic commerce, this project stands as a testament to the boundless possibilities of web development and the transformative impact it can have on businesses and consumers alike. By empowering computer science students with the skills and knowledge to build such applications, the " ShopSmart Wiki " contributes to the future of E-commerce and inspires the next generation of web developers to shape the digital landscape. The "ShopSmart Wiki" e-commerce platform leverages a well-chosen set of tools and technologies to provide a robust, scalable, and efficient system. The combination of JavaScript, React.js, Node.js, and Express.js facilitates the development of a responsive and dynamic frontend and a high-performance backend. The usage of MongoDB as the database management system ensures efficient data storage and retrieval. The choice of Linux as the operating system, along with other supporting tools like Git, Webpack, Nginx, and Docker, contributes to the reliability, security, and maintainability of the "ShopSmart Wiki" platform. The selection of these tools and technologies aligns with the system's goals of delivering an exceptional user experience and creating a seamless and reliable e-commerce platform for users worldwide.

**Appendix A: User documentation**

The "ShopSmart Wiki" User Documentation is a comprehensive and detailed guide that aims to empower end-users with the necessary information and instructions to navigate the platform effectively and make the most out of their online shopping experience. It provides step-by-step guidance, illustrations, and explanations to ensure users can easily use the application and its features. The User Documentation includes the following comprehensive sections:

**1. Introduction**

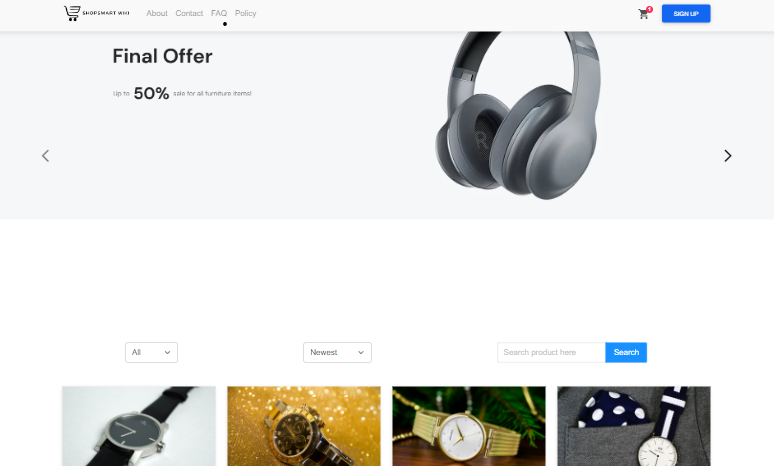
* Introduces the "ShopSmart Wiki" application, providing an overview of its purpose and objectives.
* Outlines the benefits and advantages of using the platform for online shopping.
* Defines the target audience of the documentation, highlighting that it is designed for end-users, customers, and shoppers.

**2. Getting Started**

* Guides users through the process of accessing the " ShopSmart Wiki " application from various devices, including computers, smartphones, and tablets.
* Provides detailed instructions on creating a new user account for new users, including setting up login credentials.
* Explains the login process for existing users, with instructions on how to reset passwords if needed.

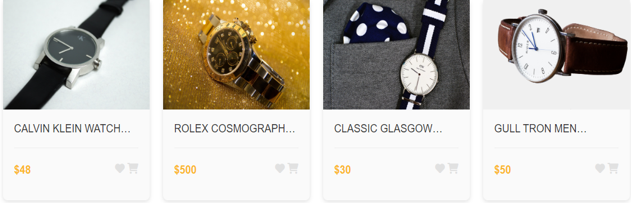
**3. User Interface**

* Offers an in-depth tour of the application's user interface, providing visual aids and screenshots to aid understanding.
* Demonstrates the layout, navigation menus, and main components of the user interface.
* Explains how users can easily navigate through different sections, product categories, and search for specific items.



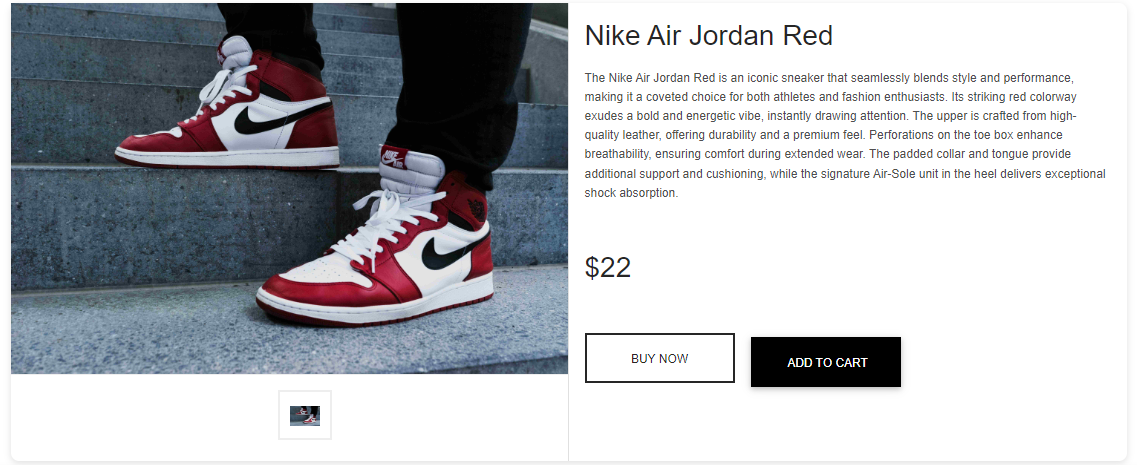
**4. Browsing Products**

* Provides detailed instructions on how to browse and search for products using various filters, sorting options, and search bar functionalities.
* Includes visual demonstrations of product categories and subcategories for efficient exploration.
* Shows users how to apply filters based on price, brand, popularity, and other attributes to find products matching their preferences.



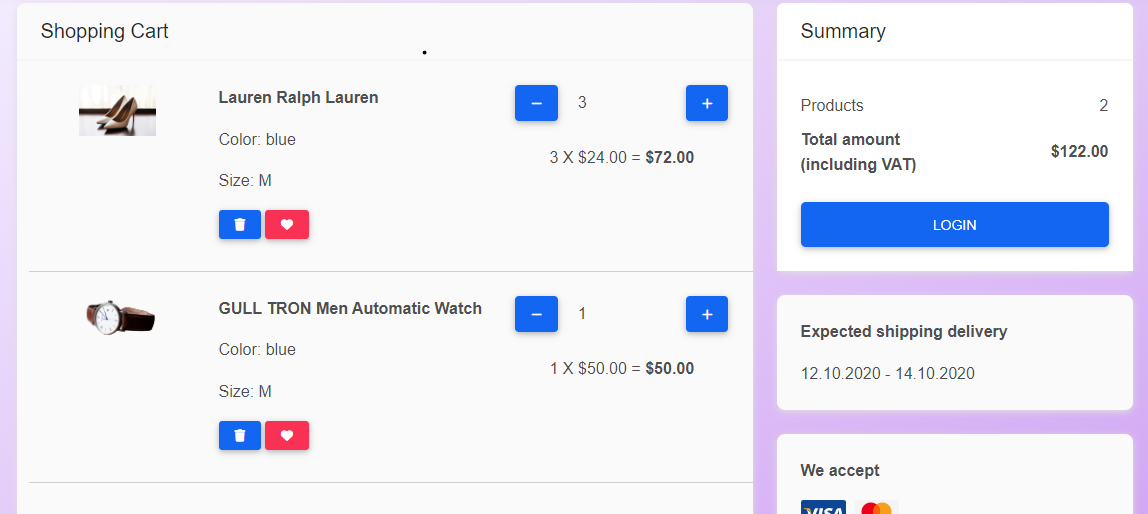
**5. Product Details**

* Offers a comprehensive guide on how to view detailed information about a particular product.
* Provides clear explanations of product attributes, specifications, images, and user reviews to aid purchasing decisions.
* Demonstrates how to access related products and product recommendations based on user preferences.



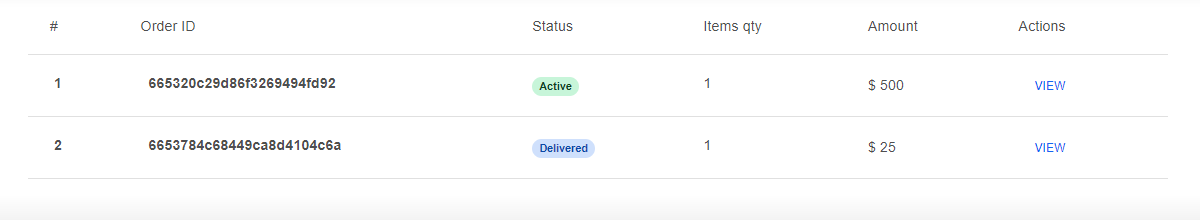
**6. Adding to Cart**

* Guides users through the process of adding products to their shopping cart for future purchase.
* Demonstrates how to adjust product quantities, remove items from the cart, and save items for later purchase.
* Provides information on the cart's features, such as displaying total order value and applying discounts or promotions.

****

**7. Placing Orders**

* Walks users through the seamless process of placing orders for selected products in the shopping cart.
* Provides detailed instructions on inputting shipping and payment details securely.
* Assures users of the platform's secure checkout process and data encryption.

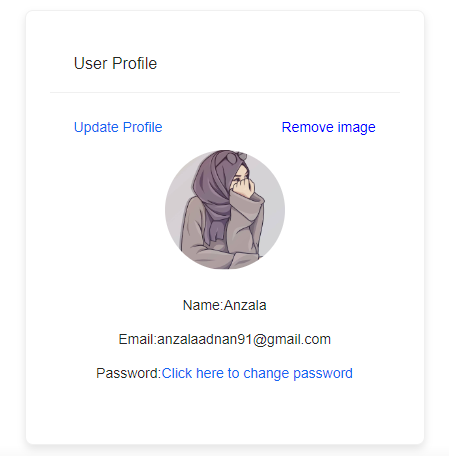
****

**8. Order Management**

* Explains how users can easily access and track their orders after completing a purchase.
* Provides information on order status updates, shipment tracking, and delivery estimates.
* Includes instructions on how to request order cancellations or modifications if required.

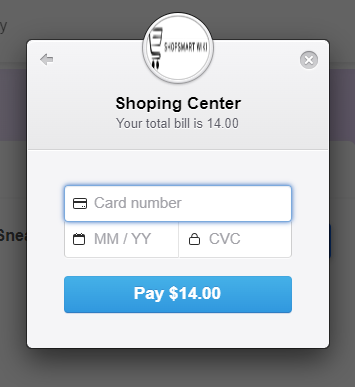
**9. Managing User Profile**

* Instructs users on how to update and manage their profile information, including names, addresses, contact details, and payment methods.
* Demonstrates how to adjust account settings, communication preferences, and notification preferences.
* Provides guidance on managing saved addresses and credit/debit card details for quicker and convenient checkouts.



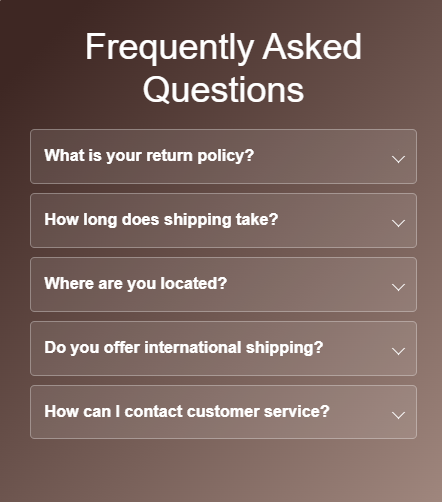
**10. Payment and Security**

* Provides an overview of the available payment methods, including credit cards, debit cards, and secure online payment gateways.
* Explains the application's robust security measures, such as Secure Socket Layer (SSL) encryption and data protection protocols.
* Assures users of the platform's commitment to safeguarding personal and financial information.

****

**11. Frequently Asked Questions (FAQs)**

* Addresses common questions and concerns that users may have while using the platform.
* Offers concise answers to frequently asked queries related to account management, product browsing, and order processing.
* Serves as a quick reference for users seeking immediate solutions to common issues.



**12. Troubleshooting**

* Provides troubleshooting tips and solutions for potential issues that users may encounter during their shopping experience.
* Includes step-by-step instructions on resolving login problems, clearing cache and cookies, and dealing with checkout errors.
* Reiterates the availability of customer support channels for further assistance with more complex issues.

**13. Glossary**

* Presents a comprehensive glossary of key terms and jargon specific to online shopping and the application.
* Helps users understand the meanings of technical terms, product attributes, and industry-related terminology.

**14. Conclusion**

* Concludes the User Documentation with a summary of the key takeaways and benefits of using the " ShopSmart Wiki " application.
* Encourages users to explore the platform confidently and assures them of the support available for a smooth shopping journey.

The "ShopSmart Wiki" User Documentation serves as a user-friendly and valuable resource, designed to empower users with knowledge and confidence in using the platform. Its detailed and comprehensive nature ensures that users can make informed decisions, navigate the application with ease, and enjoy a seamless, convenient, and enjoyable online shopping experience.