SANIWAREFIX

ONLINE SANITARYWARE STORE

Mini Project Report

Submitted by

MEENU ANNA CHERIAN

Reg. No.: AJC22MCA-2060

In Partial fulfillment for the Award of the Degree of

MASTER OF COMPUTER APPLICATION
(MCA TWO YEAR)
[Accredited by NBA]

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

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

2023-2024

DEPARTMENT OF COMPUTER APPLICATIONS

AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

CERTIFICATE

This is to certify that the Project report, "**SANIWAREFIX"** is the bonafide work of **MEENU ANNA CHERIAN (Regno: AJC22MCA-2060)** in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2023-24.

Mr.Binumon Joseph
Internal Guide

Ms. Meera Rose Mathew
Coordinator

Rev. Fr. Dr. Rubin Thottupurathu Jose
Head of the Department

DECLARATION

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

Date:	**MEENU ANNA CHERIAN**
KANJIRAPPALLY	**Reg: AJC22MCA-2060**

ACKNOWLEDGEMENT

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

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

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

MEENU ANNA CHERIAN

ABSTRACT

.

The Online Sanitary Ware Store and Repairing Services Project is a web based platform designed to provide users with a seamless and convenient shopping experience for a wide range of sanitary ware products. The project aim is to replicate the functionalities of a physical sanitary ware shop while leveraging the advantages of an Online platform. User can browse through an extensive catalog of sanitary products, view detailed product descriptions, prices. The platform ensure secure payment gateways. Admins can manage the products inventory, orders and user account efficienty. Stock manager can add, update and delete stocks and also view stocks.

Modules:

There are three modules:

1.Admin.

2.Stock manager

3.Customer.

CONTENT

List of Abbreviation

IDE	Integrated Development Environment
HTML	Hyper Text Markup Language.
CSS	Cascading Style Sheet
SQLite	Relational Database Management System
UML	Unified Modeling Language
AJAX	Asynchronous JavaScript and XML
JS	Java Scrip

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

The Online Sanitary Ware Store and Repairing Services Project is a web based platform designed to provide users with a seamless and convenient shopping experience for a wide range of sanitary ware products. The project aim is to replicate the functionalities of a physical sanitary ware shop while leveraging the advantages of an Online platform. User can browse through an extensive catalog of sanitary products, view detailed product descriptions, prices and customer reviews. The platform ensure secure payment gateways. Stock manager can add, update stocks and also view stocks.

1.2 PROJECT SPECIFICATION

This is a website in which user can buy different kinds of sanitary ware products as per their need and desires.

1.Admin

Manage the whole website.

View user details.

Manage stock manager details.

View stocks.

2.Stock manager

View Stocks.

Add the stock.

Update the stocks.

3.User

Create and edit the account.

Search, view and buy products.

Payment.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

The Online Sanitary Ware Store and Repairing Services is a pioneering web-based platform redefining how users interact with a diverse array of sanitary products. Seamlessly merging the convenience of online shopping with the comprehensive experience of a physical store, our platform aims to deliver an unparalleled shopping journey. Users can explore an extensive catalog featuring detailed product descriptions, prices, and a secure payment gateway, ensuring a hassle-free and secure purchasing process. With distinct modules catering to administrators, stock managers, and customers, our platform prioritizes efficiency. Administrators oversee the site's operations, manage user details, and monitor stock manager activities for smooth functioning. Stock managers maintain an up-to-date inventory, adding, updating, and deleting stocks effortlessly. For customers, account creation, seamless browsing, and secure transactions define their experience. We aim to set new standards in sanitary ware retail by providing a user-centric, reliable, and convenient online platform.

2.2 EXISTING SYSTEM

The existing sanitaryware products selling system operates within the constraints of traditional manual processes, leading to limited geographical reach, higher overhead costs, and reduced operational flexibility. The reliance on brick-and-mortar stores restricts accessibility and convenience for a broader audience, and the system faces challenges in adapting to evolving customer preferences. Seasonal fluctuations impact business, and the emergence of competitive online platforms poses a threat.

The existing sanitaryware products selling system operates within the constraints of traditional manual processes, leading to limited geographical reach, higher overhead costs, and reduced operational flexibility. The reliance on brick-and-mortar stores restricts accessibility and convenience for a broader audience, and the system faces challenges in adapting to evolving customer preferences. Seasonal fluctuations impact business, and the emergence of competitive online platforms poses a threat.

2.2.1 NATURAL SYSTEM STUDIED

The platform, much like an ecosystem, comprises modules (Admin, Stock Manager, Customer) that interact seamlessly to ensure smooth operations. Emulating nature's efficiency, the platform should evolve to meet user needs, optimize processes, and minimize waste. Incorporating redun dancies for resilience, integrating feedback mechanisms for self-improvement, and promoting sustainable practices (such as efficient inventory management and eco-friendly options) can create a robust and user-friendly online environment akin to the harmony found in natural systems.

2.2.2 DESIGNED SYSTEM STUDIED

The specialty of my designed e-commerce system for sanitary ware lies in its replication of a physical sanitary ware shop within an online platform. It blends convenience with the comprehend give functionalities of a brick-and-mortar store. This system offers users a seamless shopping experience by allowing them to browse through a wide array of sanitary products with detailed with detailed descriptions and prices, mirroring the in-store experience. The emphasis on secure payment gateways ensures trust and safety during transactions. Moreover, the platform's modularity with distinct modules (Admin, Stock Manager, Customer) allows efficient management of inventory, orders, and user accounts, fostering a smooth and organized operational structure. This specialization bridges the gap between traditional shopping and online convenience, providing users with the familiarity of a physical store while leveraging the advantages of an e-commerce platform.

2.3 DRAWBACKS OF EXISTING SYSTEM

- People Need to walk from shop to shop
- Additional fuel cost
- Expensive when product directly purchased from shop
- Additional fuel cost
- Less choice
- Timely Based System
- Less User

2.4 PROPOSED SYSTEM

Online Sanitaryware Products and Repairing Service, an e-commerce platform that aims to simplify the process of purchasing high- quality sanitaryware products online. With a vast selection of sanitaryware products, the platform allows users to quickly find the products they need. The user interface is designed to be user-friendly, allowing for easy navigation and hassle-free transactions. Moreover, the system provides stock manager with a platform to showcase their products and engage with potential customers.

2.5 ADVANTAGES OF PROPOSED SYSTEM

- Convenience: The proposed system offers a high level of convenience to customers. They can access the system anytime and anywhere with an internet connection, allowing them to browse, select, and purchase sanitaryware products at their own convenience. This eliminates the need for physical travel, saving time and effort.

- Expanded product availability: The proposed system can provide a broader range of sanitary options compared to traditional offline sanitaryware store. This expanded product availability gives customers more choices and opportunities to find specific sanitaryware products they are looking for.

- Detailed product information: The proposed system can provide detailed product information for each sanitaryware products.

- Efficient search options: The proposed system can include advanced search and filtering options to facilitate products selection. Customers can criteria such as product type, dimensions, or specific functionalities to refine their search and locate the precise sanitary products they seek. This robust search capability not only saves valuable time but also ensures customers find products that precisely match their unique requirements, enhancing their overall shopping experience.

CHAPTER 3
REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

This examination is a basic organize in figuring out if a project will accomplish the objectives of the association in connection to the assets, work, and time put into it. It helps the designer in assessing the project's conceivable focal points and conceivable outcomes for long run. To discover in case a proposed framework is doable and beneficial of advance examination, a possibility consider must be conducted. The proposed system's impact on the association, capability to fulfill client requests, and resource-effectiveness are all assessed as portion of the possibility think about. Hence, an achievability think about is regularly performed some time recently a modern application is given the go-ahead to be created. The specialized, financial, and operational reasonability of the extend are as it were many of the components that were carefully taken into consideration all through the assessment, concurring to the possibility consider paper.

3.1.1 Economical Feasibility

- Increased Revenue: - The organization can generate revenue through the sale of sanitaryware product. The diversified income stream can contribute to the organization’s financial growth.
- Cost Efficiency: - Operating an online platform can often be more cost efficient than maintaining physical Store.
- Wide Market Research: - An online platform can reach a broader audience, potentially even on a global scale. This expanded market can lead to increased sales and revenue opportunities for the organizations.
- Cost Savings: - User can potentially and competitive prices online and avoid the cost associated with travelling to physical stores.
- Time Savings: - The Platform allows users to quickly search for products and services, reducing the time spent shopping and scheduling repair services.
- Conveniences: - User can purchase 24/7 eliminating the need for physical store visits.

3.1.2 Technical Feasibility

Access whether the required technologies (eg: Augmented reality) are available and feasible to implement within the project budget and timeline. This study evaluates whether the proposed project is technically achievable given the chosen technologies, resources and expertise. In the context of online sanitary ware products and repairing services,technical feasibility involves accessing whether the proposed technology such as augment reality images processing can be effectively implemented within the project

3.1.3 Behavioral Feasibility

In order to ensure the success of this system, two important questions have been considered:
(1) whether there is enough support for users, and
(2) whether the system will cause any harm.
These questions were thoroughly evaluated to ensure that the system would be beneficial upon implementation. Additionally, all behavioral factors were taken into account during the feasibility study to ensure that the project is behaviorally feasible. Overall, the proposed system is expected to achieve its objectives with a high level of success.

3.1.4 Feasibility Study Questionnaire

1.Project Overview?

The Online Sanitary Ware Store and Repairing Service project is a web-based platform aimed at providing users with a seamless and convenient shopping experience for a wide range of sanitary ware products.

2.To what extend the system is proposed for?

The system is proposed to cover various aspects of an online sanitary ware store and related service including product management, user management, stock management. It aim to provide end to end solutions for customers looking to purchase sanitary ware products

3.Specify the viewers/public which is to be involved in the system?

Customers: Individuals looking to purchase sanitary ware products and services.
Admins: Responsible for managing the entire platform.
Stock Manager: Responsible for managing and update the product inventory.

4.List the Modules included in your system?

Admin ,Stock Manager ,User (Customer) .

5.Who owns the system?

Administrator

6.System is related to which firm/industry/organization?

The system is related to the sanitary ware industry and maybe associated with a specific firm or organization involved in the sale and maintenance of sanitary products.

7.Details of person that you have contacted for data collection

Xaviour Antony (Jaquar, Pathanamthitta)

Questionnaire to collect details about the project?

1.How is task allotted to handle the functioning of your electronic shop?

Task is divided among different sections like Accountant, cashier, staffs, stock manager, cleaners.

2.How is it made user friendly?

If a customer visit the shop for first time his credentials are feeded on to system so that they get e bill, get notified on offers .

3.How is stock managed?

Reports are kept for each sold products and will refill out of stock products.

4.Does staff help out customers to find a relevant product?

Staff will just help to know about systems specification and they donot encourage to buy a particular product ,according to marketing strategy

5.How do you attract customers?

By keeping offers.

6.How is payment system managed?

Either as cash or online transaction (g pay phone pay)

7.What are the extra expenditures?

Maintenance cost, Fuel cost, Electricity bill.

8.Is it government organized system?

It is organized by both public and private sector.

9.Is there any factors that affect sales?

Yes, public holidays, harthals and mainly area where shop is situated Is a factor for natural disaster like flood

10.How the customers details are stored?

Store in the shop register.

3.2 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor	-	Intel core i7
RAM	-	1 6 G B
Hard disk	-	1 T B

3.2.2 Software Specification

Front End	-	HTML, CSS
Back End	-	Django
Database	-	SQLite
Client on PC	-	Windows 7 and above.
Technologies used	-	JS, HTML5, AJAX, J Query, PHP, CSS

3.3 SOFTWARE DESCRIPTION

3.3.1 Django

Django, a leading open-source web framework, epitomizes the pinnacle of modern web development with its efficiency and versatility. Built on Python, Django follows the Model-View-Controller architecture, prioritizing simplicity and flexibility. Noteworthy features include its Object-Relational Mapping system, streamlined URL routing, and a user-friendly template engine. The built-in admin interface simplifies data management, while robust security measures and middleware support enhance application integrity. Django scales effortlessly, accommodating growing datasets and traffic demands. Its REST Framework extension further extends its utility to API development. Supported by an active community and extensive documentation, Django stands as a powerful toolkit, seamlessly shaping the development of dynamic and secure web applications for diverse purposes, from content management systems to Restful API. In essence, Django empowers developers with a comprehensive and adaptable framework for crafting sophisticated and salable web solutions.

3.3.2 SQLite

SQLite, a lightweight and server less relational database management system, is celebrated for its simplicity, portability, and efficiency. Operating as a self-contained, single-file database, SQLite requires minimal setup and eliminates the need for a separate server process. This design makes it a preferred choice for embedded systems, mobile applications, and small to medium-scale projects. With zero configuration and a server less architecture, SQLite streamlines the database management process. It supports ACID transactions, ensuring data integrity and reliability, even in the face of system interruptions. The dynamic typing feature allows flexible data storage, accommodating various data types within the same column.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

Any designed system or product's development starts with the design phase. An efficient system depends on well-executed design, which is a creative process. It entails utilizing a variety of approaches and concepts to define a process or system in enough depth to allow for its actual execution. Regardless of the development model chosen, the design phase is critical in software engineering. It strives to produce the architectural detail needed to build a system or product and serves as the technical backbone of the software engineering process. This program has through a thorough design phase that optimizes every aspect of effectiveness, performance, and accuracy. A user-oriented document is converted into a document for programmers or database employees during the design process.

4.2 UML DIAGRAM

Software engineering uses the Unified Modeling Language (UML), a standardized visual language, to model, develop, and document software systems. UML diagrams provide a concise and organized approach to represent different facets of a software system, serving as a common communication tool for developers, stakeholders, and designers. There are many different varieties of UML diagrams, including class diagrams, sequence diagrams, use case diagrams, and more. Each is designed to communicate a particular piece of knowledge about the design, operation, and interactions of the system. UML diagrams are essential to the software development process because they help with the visualization, analysis, and design of complex systems, which in turn makes the process more effective and efficient.

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

4.2.1 USE CASE DIAGRAM

A use case diagram may be a graphical delineation that appears how clients and other outside on-screen characters associated with a system's inside components. A utilize case diagram's essential work is to perceive, layout, and orchestrate a system's utilitarian needs as seen through the eyes ofits clients. The Unified Modeling Language (UML), a standard language for modeling actual things and systems, is frequently used to construct use case diagrams. Use cases can be utilized to achieve an assortment of framework objectives, counting setting fundamental prerequisites, confirming equipment plans, testing and investigating program, creating online offer assistance references, or performing client bolster obligations. Customer support, product obtaining, catalogue overhauling, and payment processing are as it were a couple of illustrations of use cases within the setting of item deals.

The system boundaries, actors, use cases, and their connections together make up a use case diagram. The system boundary establishes the system's boundaries in reference to its surroundings.Actors are often defined depending on the roles they play and reflect the people or systems that interact with the system. The precise activities or behaviors that actors carry out within or close tothe system are known as use cases. Finally, the graphic shows the connections between actors anduse cases as well as the use cases themselves.

Use case diagrams are graphical representations used to capture the functional requirements of a system. When drawing a use case diagram, it is important to follow these guidelines to ensure an efficient and effective diagram:

- Choose descriptive names for use cases that accurately reflect the functionalities.
- Assign appropriate names to actors to help identify their roles in the system.
- Ensure that relationships and dependencies are clearly depicted in the diagram.
- Avoid including every possible relationship, as the main goal is to identify the essentialrequirements.
- Use notes when necessary to clarify important points.

By following these guidelines, we can create a clear and concise use case diagram that accurately represents the functional requirements of the system.

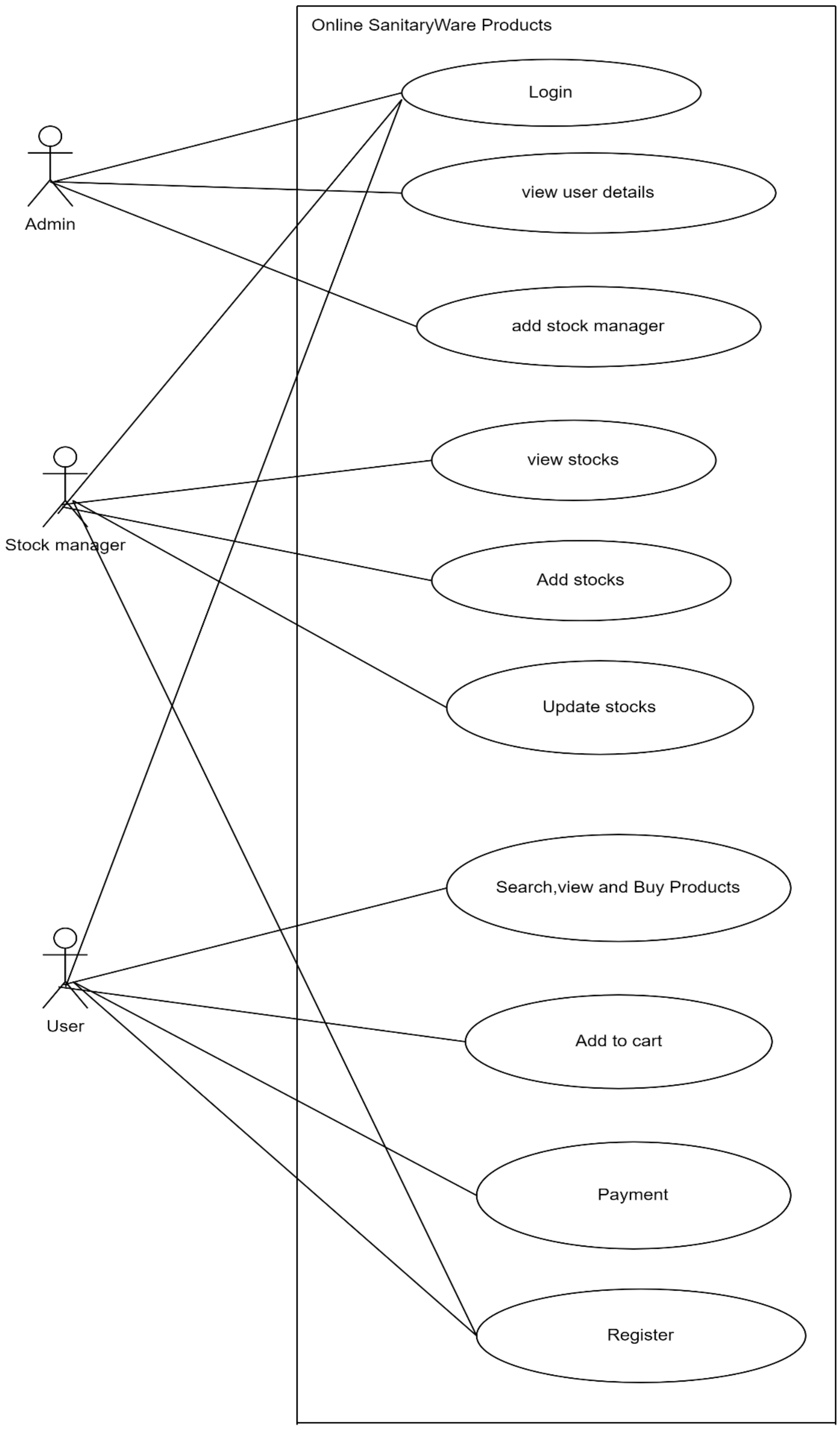


Fig 1: Use Case diagram

4.2.2 SEQUENCE DIAGRAM

The chronological order of interactions between various system components is shown in a sequence diagram, a form of interaction diagram. It demonstrates how several things communicate with one another over the course of a series of messages. These images are sometimes referred to as event scenarios or event scenarios diagrams. In software engineering, sequence diagrams are frequently used to describe and comprehend the needs of both new and old systems. They support the visualization of object control relationships and the detection of systemic issues.
Sequence Diagram Notations –

i. Actors – In UML, a role that interacts with the system and its objects is represented by an actor. Actors frequently exist outside of the system that the UML diagram is intended to portray. Actors can play a variety of roles, including those of external topics or human users. A stick person notation is used in UML diagrams to represent actors. Depending on the situation that is being modelled, a sequence diagram may have more than one actor.

ii. Lifelines – A lifeline in a sequence diagram is a vertical dashed line that represents the lifespan of an object participating in the interaction. Each lifeline represents an individual participant in the sequence of events and is labeled with the name of the participant. The lifeline shows the timeline of events for the participant and is drawn as a vertical line extending from the participant's activation point to its deactivation point.

iii. Messages - Messages are a key component of sequence diagrams, representing the interactions and communication between objects or components in a system. They can be categorized into synchronous and asynchronous messages, create and delete messages, self-messages, reply messages, found messages, and lost messages. Guards are also used to model conditions and restrictions on message flow.

iv. Guards- Guards in UML are used to model conditions and are employed to restrict the flow of messages when a certain condition is met. This feature is essential for letting software developers know about any constraints or limitations associated with a system or a particular process.

Uses of sequence diagram –

- Modeling and visualizing the logic of complex functions, operations, or procedures.
- Showing details of UML use case diagrams.
- Understanding the detailed functionality of current or future systems.
- Visualizing how messages and tasks move between objects or components in a system.
- Overall, sequence diagrams are useful for representing the flow of interactions between objects in a system, and can help both businesspeople and software engineers better understand and communicate system requirements and behavior.

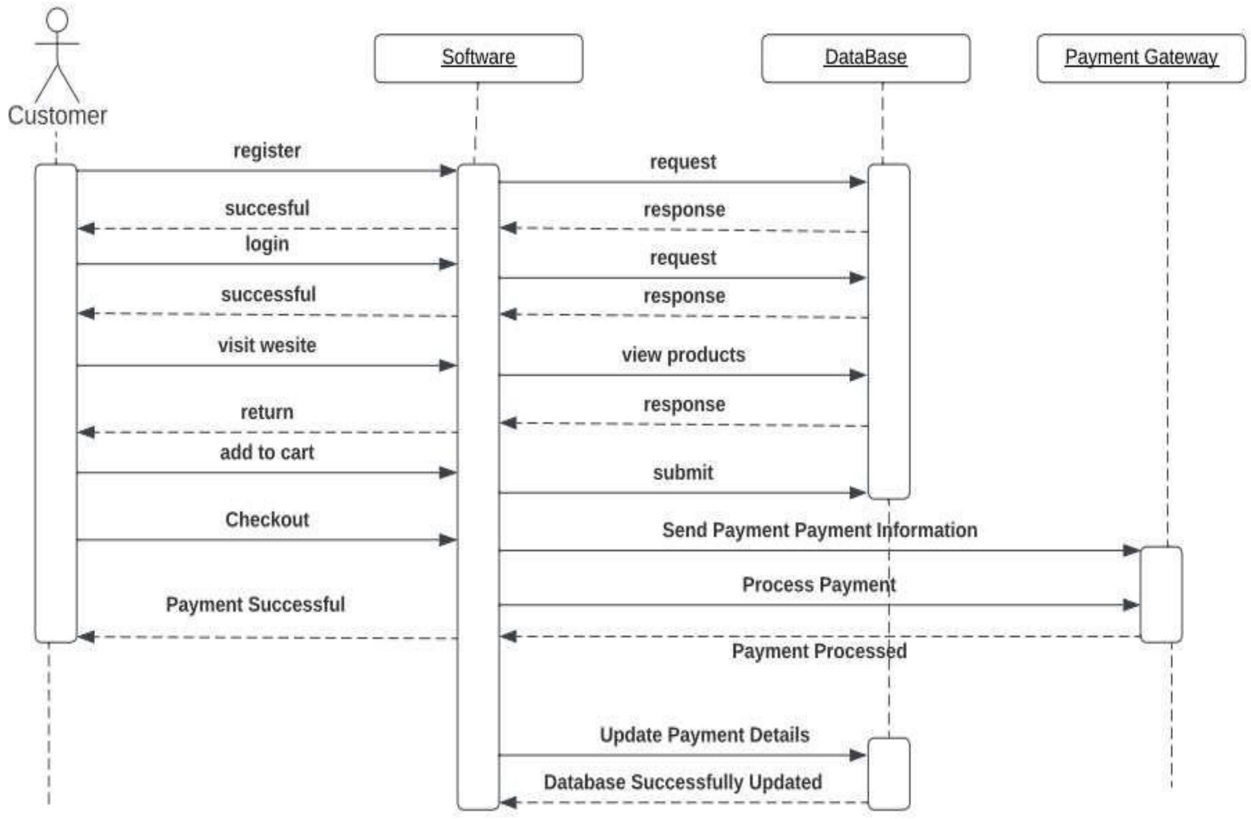


Fig 2: Sequence diagram

4.2.3 State Chart Diagram

A state diagram is a visual representation, often created using the Unified Modeling Language (UML), that shows the different states that an object can exist in and how it can transition between those states. It is also referred to as a state machine diagram or state chart diagram.

The State Chart Diagram is a behavioral diagram in UML that describes the behavior of a system or object over time. It includes various elements such as:

- Initial State - This state represents the starting point of the system or object and is denoted by a solid black circle.
- State - This element describes the current state of the system or object at a specific point in time and is represented by a rectangle with rounded corners.
- Transition - This element shows the movement of the system or object from one state to another and is represented by an arrow.
- Event and Action - An event is a trigger that causes a transition to occur, and an action is the behavior or effect of the transition.
- Signal - A message or trigger caused by an event that is sent to a state, causing a transition to occur.
- Final State - The State Chart Diagram ends with a Final State element, which is represented by a solid black circle with a dot inside. It indicates that the behavior of the system or object has completed.

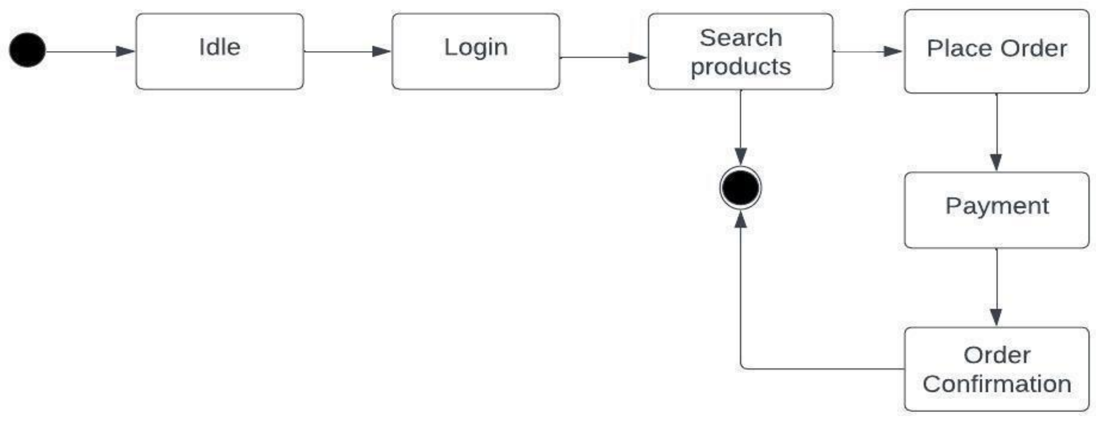


Fig 3: State Chart diagram

4.2.4 Activity Diagram

An activity diagram is a visual representation of a workflow that shows how one activity leads to another. An activity is referred to as a system operation, and one operation leads to another in the control flow. A flow can be parallel, concurrent, or branched, and activity diagrams use various functions such as branching, joining, etc., to manage all types of flow control. Activity diagrams are a type of behavior diagram that shows the behavior of a system. They show the flow of controlfrom the start point to the end point and show the different decision paths that exist during the execution of the activity.

The key components of an activity diagram are:

- Initial node - A starting point of the activity diagram, denoted by a black circle.
- Activity - A task or action performed by the system or entity, represented by a rectanglewith rounded corners.
- Control flow - It represents the sequence of activities or actions performed by the system orentity, represented by an arrow.
- Decision node - A decision or branching point in the activity flow, denoted by a diamondshape.
- Merge node - Used to merge multiple branches of the activity flow into a single flow,represented by a diamond shape with a plus sign inside.
- Fork node - Used to split the activity flow into multiple parallel flows, represented by a solidblack circle with multiple arrows.
- Join node - Used to join multiple parallel flows back into a single flow, represented by asolid black circle with multiple arrows pointing towards it.
- Final node - The end point of the activity diagram, denoted by a black circle with a dotinside.
- Object flow - Represents the flow of objects or data between activities, represented by adashed arrow.

Activity diagrams are useful in clarifying complex processes, identifying potential issues, and communicating process flows to stakeholders and project team members.

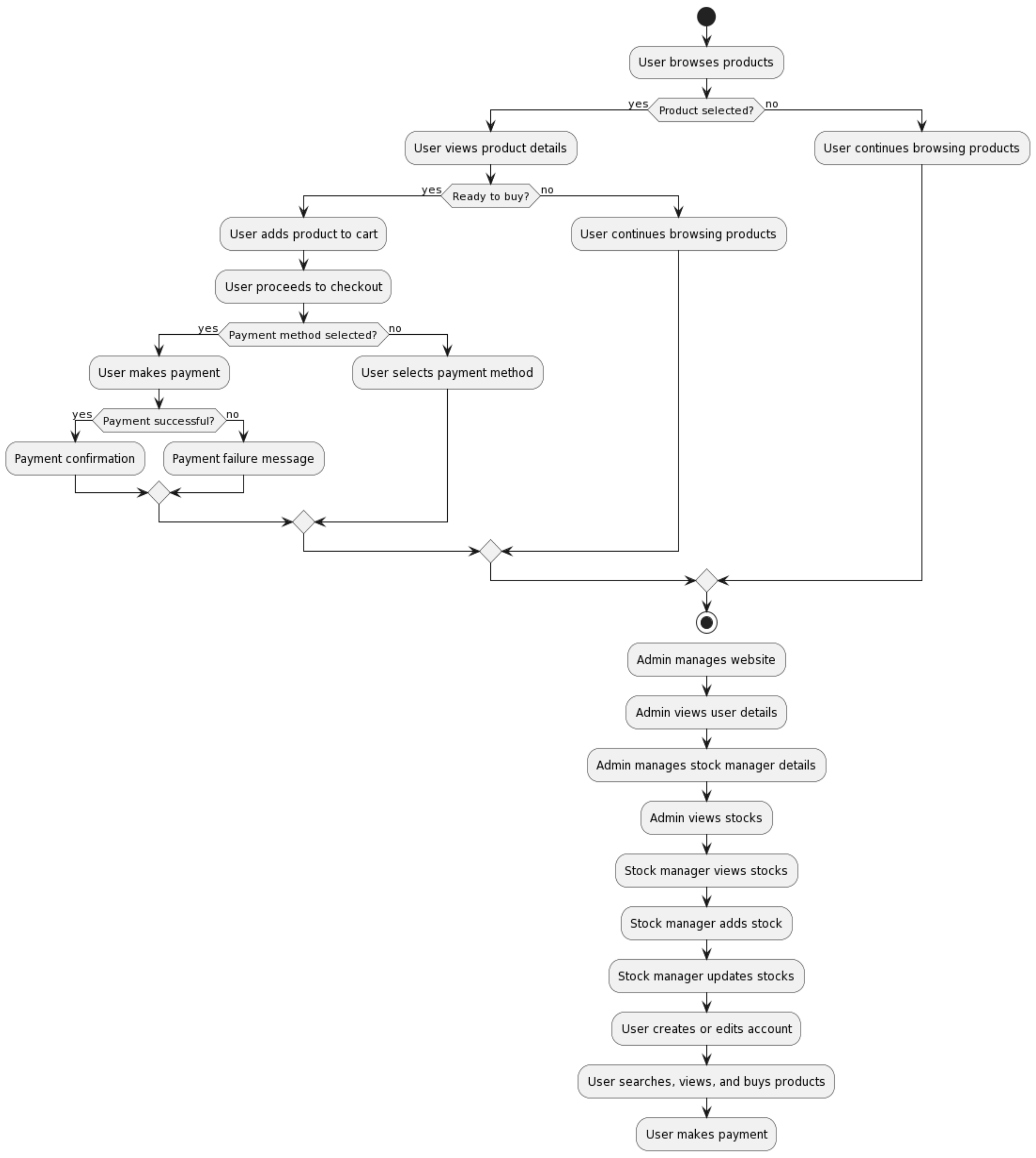

User browses products
Product selected?
yes
no
User views product details
User continues browsing products
Ready to buy?
yes
no
User adds product to cart
User continues browsing products
User proceeds to checkout
Payment method selected?
yes
no
User makes payment
User selects payment method
Payment successful?
yes
no
Payment confirmation
Payment failure message
Admin manages website
Admin views user details
Admin manages stock manager details
Admin views stocks
Stock manager views stocks
Stock manager adds stock
Stock manager updates stocks
User creates or edits account
User searches, views, and buys products
User makes payment

Fig 4: Activity diagram.

4.2.5 Class Diagram

The class diagram is a fundamental component of object-oriented modeling and serves as the primary means of conceptual modeling for the structure of an application. Additionally, class diagrams can be used for detailed modeling that can be translated into programming code. They can also be employed for data modeling purposes Class diagrams are a crucial component of UML used to represent classes, objects, interfaces, and their relationships and attributes in a system. Some important components of a class diagram are:

- Class: It is a blueprint or template for creating objects and is represented as a rectangle withthe class name, attributes, and methods.
- Interface: It is a collection of abstract methods that specify a contract between a class and the outside world. It is represented as a circle with the interface name inside.
- Object: It is an instance of a class with state and behavior. It is represented as a rectangle with the object name inside.
- Association: It is a relationship between two classes that represents a connection or link andis represented as a line with optional directionality, multiplicity, and role names.
- Aggregation: It is a part-whole relationship where the whole (aggregator) is composed of parts (aggregates) and is represented as a diamond shape on the aggregator side.
- Composition: It is a stronger form of aggregation where the parts cannot exist without the whole and is represented as a filled diamond shape on the aggregator side.
- Inheritance: It is a relationship between a superclass and its subclasses that represents an "is-a" relationship and is represented as a line with an open arrowhead pointing from the subclass to the superclass.
- Dependency: It is a relationship where a change in one class may affect the other class andis represented as a dashed line with an arrowhead pointing from the dependent class to the independent class.
- Multiplicity: It represents the number of instances of a class that can be associated with another class and is represented as a range of values near the association or aggregation line.

Class diagrams are essential in designing and modeling object-oriented software systems as they provide visual representation of the system's structure, its functionality, and relationships.

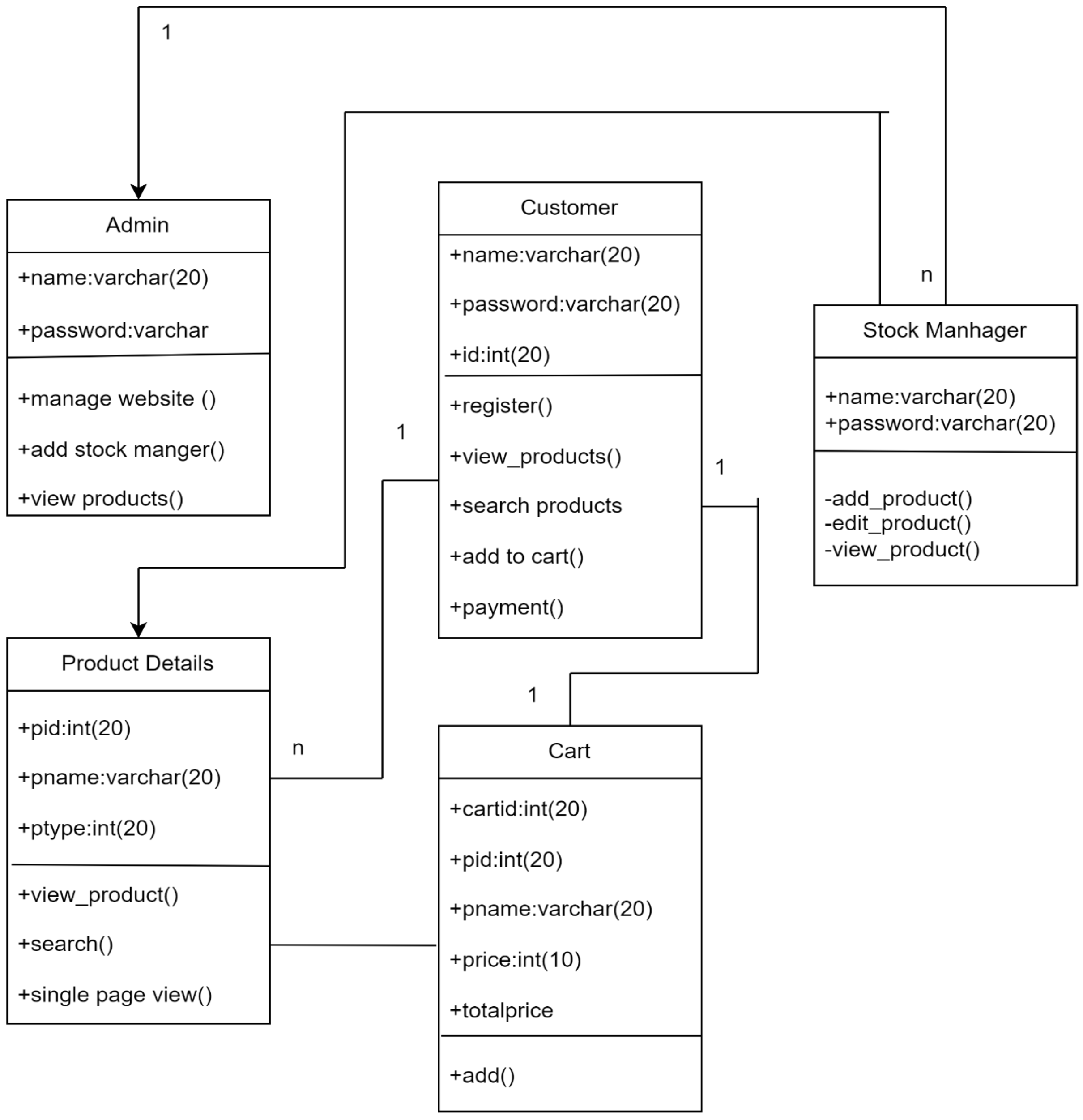


Fig 5: Class diagram

4.2.6 Object Diagram

Class diagrams and object diagrams are closely related in object-oriented modeling. The object diagrams are instances of class diagrams, which represent a snapshot of the system at a given moment in time. Both types of diagrams use the same concepts and notation to represent the structure of a system. While the class diagrams are used to model the structure of the system, includingits classes, attributes, and methods, object diagrams represent a group of objects and their connections at a specific point in time.

An object diagram is type of structural diagram in UML that shows instances of classes and their relationships. The main components of an object diagram include:

- Object: An object is an instance of a class that represents a specific entity in the system. It is represented as a rectangle with the object name inside.
- Class: A class is a blueprint or template for creating objects that defines its attributes and methods. It is represented as a rectangle with three compartments for the class name, attributes, and methods.
- Link: A link is a relationship between two objects that represents a connection or association. It is represented as a line connecting two objects with optional labels.
- Attribute: An attribute is a property or characteristic of an object that describes its state. It is represented as a name-value pair inside the object rectangle.
- Value: A value is a specific instance or setting of an attribute. It is represented as a value inside the attribute name-value pair.
- Operation: An operation is a behavior or action that an object can perform. It is representedas a method name inside the class rectangle.
- Multiplicity: Multiplicity represents the number of instances of a class that can be associatedwith another class. It is represented as a range of values (e.g. 0..1, 1..*, etc.) near the link between objjects.
- Object diagrams help to visualize the relationships between objects and their attributes in a system.They are useful for understanding the behavior of a system at a specific point in time and for identifying potential issues or inefficiencies in the system.

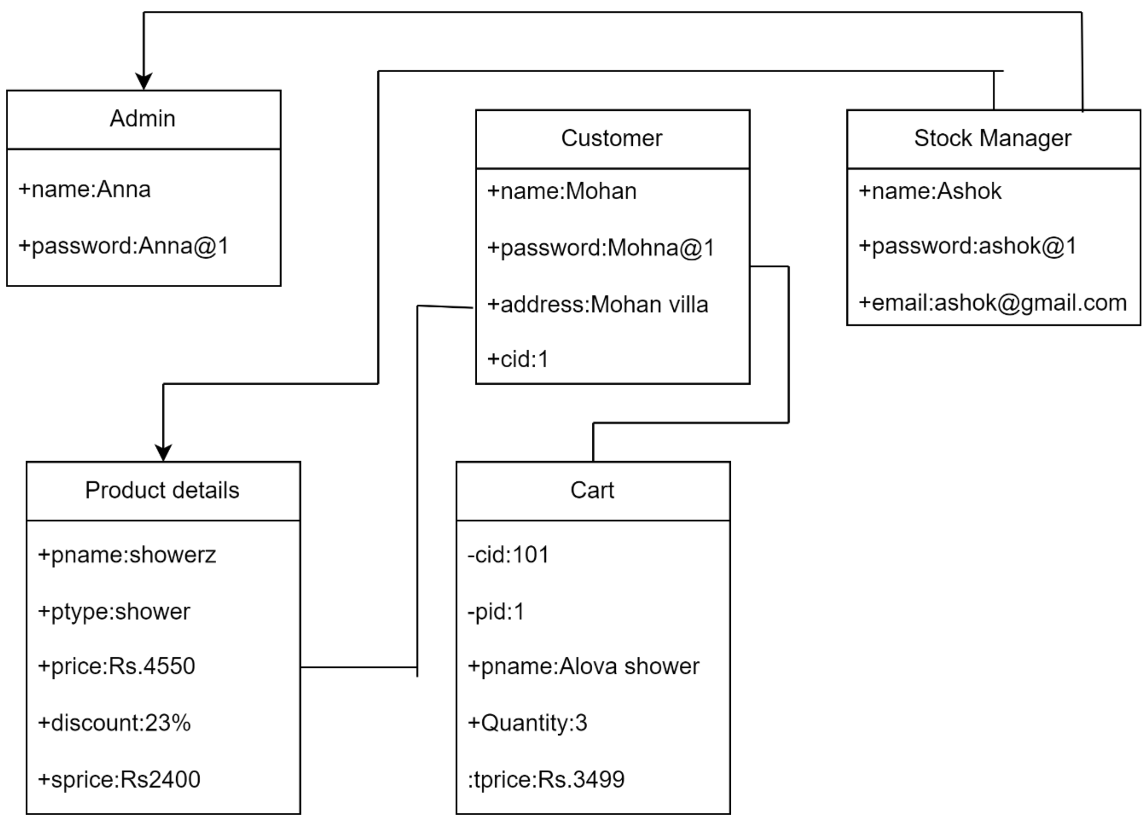


Fig 6: Object diagram

4.2.7 Component Diagram

A component diagram in UML illustrates how various components are interconnected to create larger components or software systems. It is an effective tool for representing the structure of complex systems with multiple components. By using component diagrams, developers can easily visualize the internal structure of a software system and understand how different components worktogether to accomplish a specific task.

Its key components include:

- Component: A modular and encapsulated unit of functionality in a system that offers interfaces to interact with other components. It is represented as a rectangle with the component name inside.
- Interface: A contract between a component and its environment or other components, specifying a set of methods that can be used by other components. It is represented as a circle with the interface name inside.

- Port: A point of interaction between a component and its environment or other components. It is represented as a small square on the boundary of a component.
- Connector: A link between two components that enables communication or data exchange. It is represented as a line with optional adornments and labels.
- Dependency: A relationship between two components where one component depends on another for its implementation or functionality. It is represented as a dashed line with an arrowhead pointing from the dependent component to the independent component.
- Association: A relationship between two components that represents a connection or link. It is represented as a line connecting two components with optional directionality, multiplicity, and role names.
- Provided/Required Interface: A provided interface is an interface that a component offers to other components, while a required interface is an interface that a component needs from other components to function properly. These are represented by lollipops and half-circles respectively.

Component diagrams are useful for modeling the architecture of a software system, and can help identify potential issues and improvements in the design. They can also be used to communicate the structure and behavior of a system to stakeholders, such as developers and project managers.

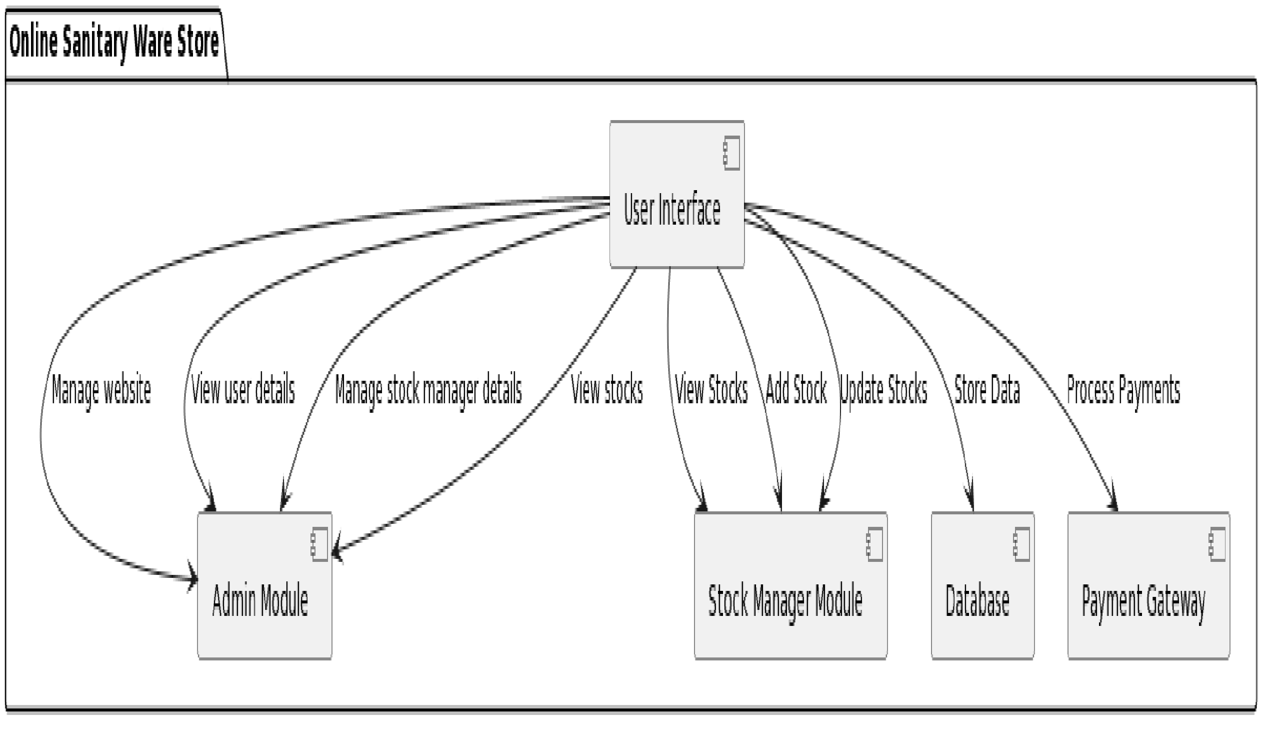


Fig 7: Component diagram

4.2.8 Deployment Diagram

A deployment diagram is a type of UML diagram that focuses on the physical hardware used to deploy software. It provides a static view of a system's deployment and involves nodes and their relationships. The deployment diagram maps the software architecture to the physical system architecture, showing how the software will be executed on nodes. Communication paths are used to illustrate the relationships between the nodes. Unlike other UML diagram types, which focus on the logical components of a system, the deployment diagram emphasizes the hardware topology.

The key components of a deployment diagram are:

- Node - A node is a physical or virtual machine on which a component or artifact is deployed.It is represented by a box with the node's name inside.
- Component - A component is a software element that performs a specific function or provides a specific service. It is represented by a rectangle with the component's name inside.
- Artifact - An artifact is a physical piece of data that is used or produced by a component. Itis represented by a rectangle with the artifact's name inside.
- Deployment Specification - A deployment specification describes how a component or artifact is deployed on a node. It includes information about the location, version, and configuration parameters of the component or artifact.
- Association - An association is a relationship between a node and a component or artifact that represents a deployment dependency. It is represented by a line connecting the two components with optional directionality, multiplicity, and role names.
- Communication Path - A communication path represents the connection between nodes, such as a network connection or communication channel. It is represented by a line with optional labels and adornments.

Deployment diagrams help in visualizing the physical architecture of a system and identifying any potential issues or bottlenecks in the deployment process. They also aid in planning the deploymentstrategy and optimizing the use of hardware resources.

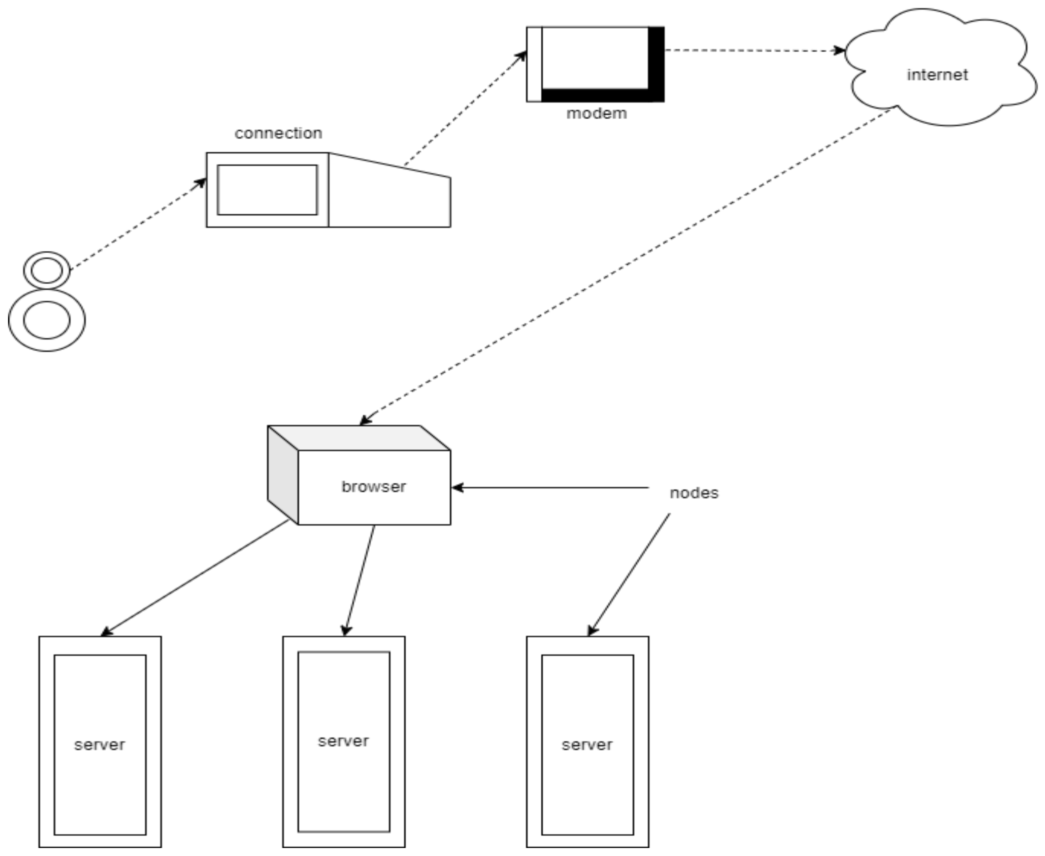


Fig 8: Deployment diagram

4.2.9 Collaboration Diagram

A collaboration diagram is a diagram that is used to represent the relationships between objects in a system. It is similar to a sequence diagram in that it represents the same information, but it does so in a different way. Instead of showing the flow of messages between objects, it depicts the structure of the objects in the system. This is because collaboration diagrams are based on object- oriented programming, where objects have various attributes and are connected to each other. Thus, collaboration diagrams are a visual representation of the object architecture in a system.

A component diagram includes the following components:

- Objects: Objects are represented by symbols with their name and class underlined, separated by a colon. In a collaboration diagram, objects are used to represent a class instance and specify its name and class. It is not necessary for every class to have an object representation, and a single class may have multiple objects. Objects are created first, andtheir class is specified afterwards. Naming objects is important to differentiate them from one another.

- Actors: Actors play a key role in the collaboration diagram as they invoke the interaction.Each actor has its own name and role. In the diagram, one actor initiates the use case.
- Links: Links are instances of association that connect objects and actors. They represent the relationship between objects through which messages are sent. Links are represented by solid lines and help objects to navigate to other objects.
- Messages: Messages represent communication between objects that carry information andare identified by a sequence number. They are represented by labeled arrows placed near the link and sent from the sender to the receiver. The direction must be navigable in that specific direction, and the receiver must understand the message.

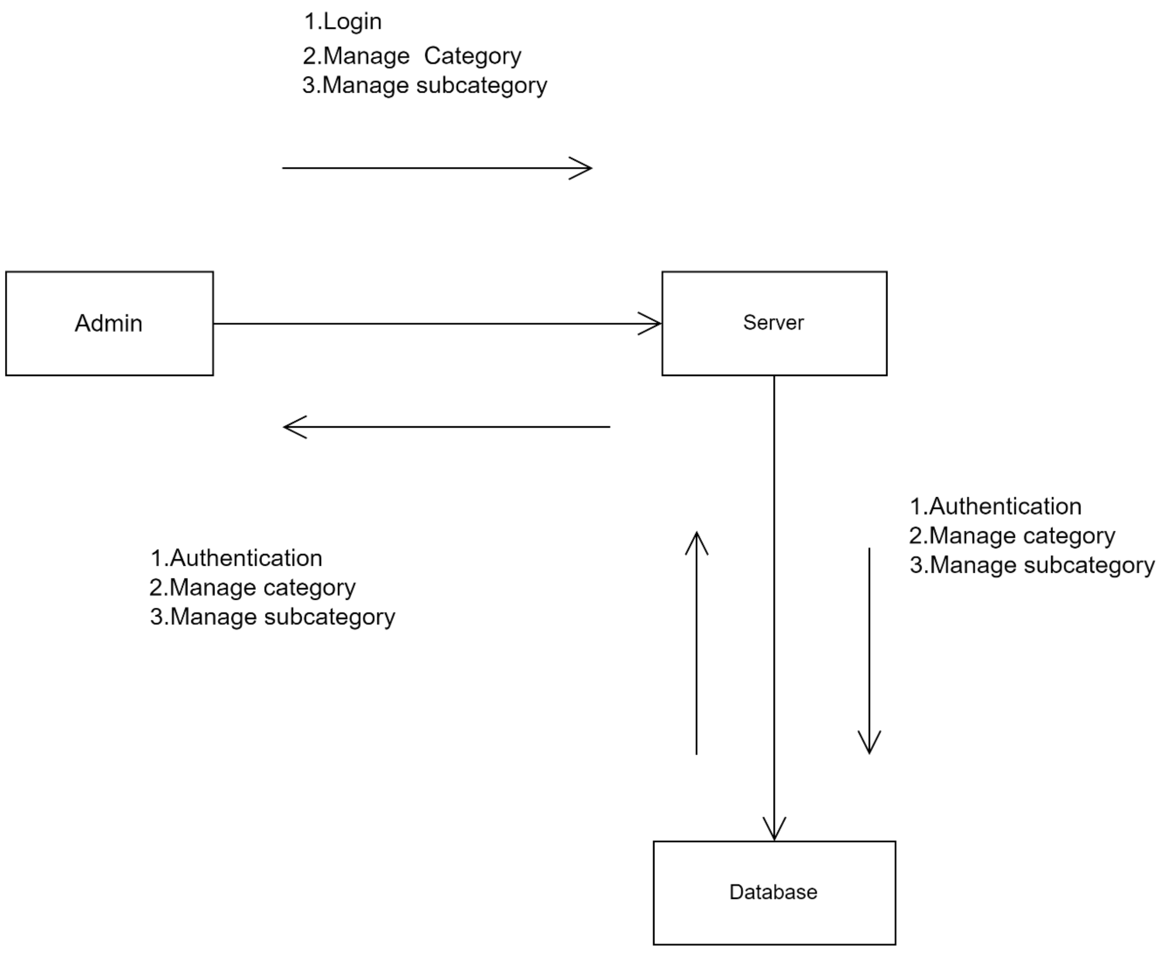
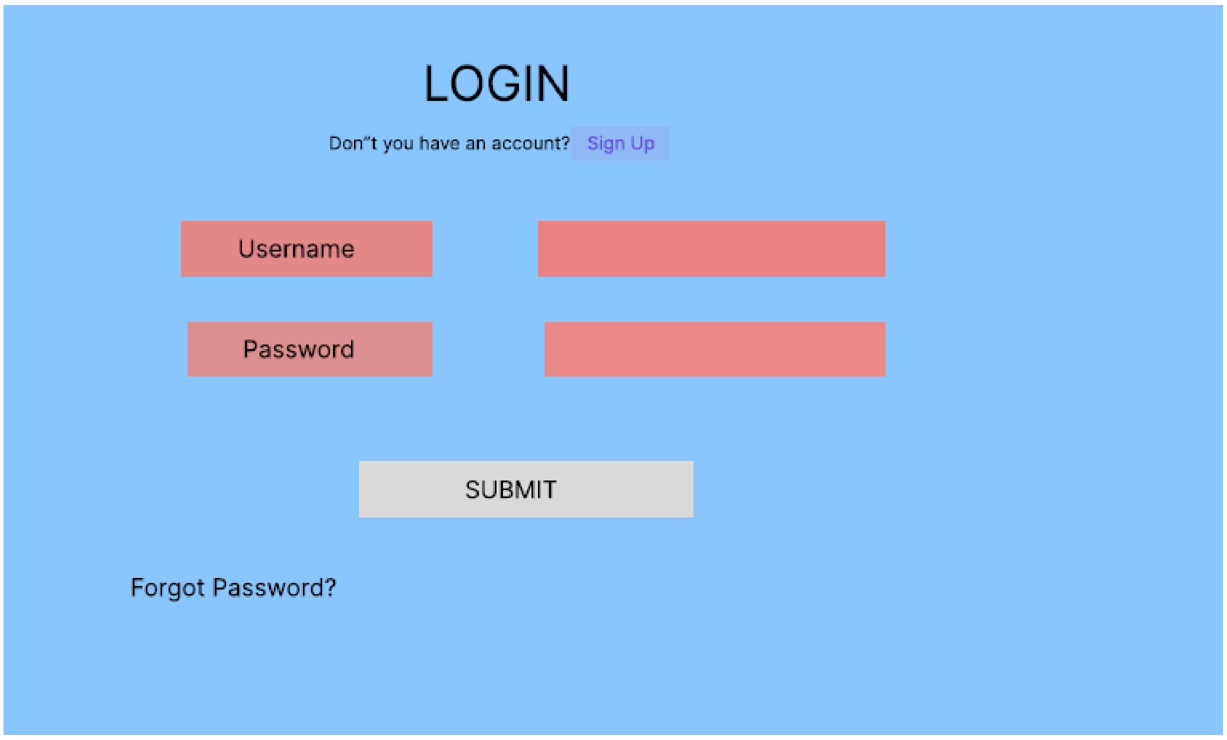


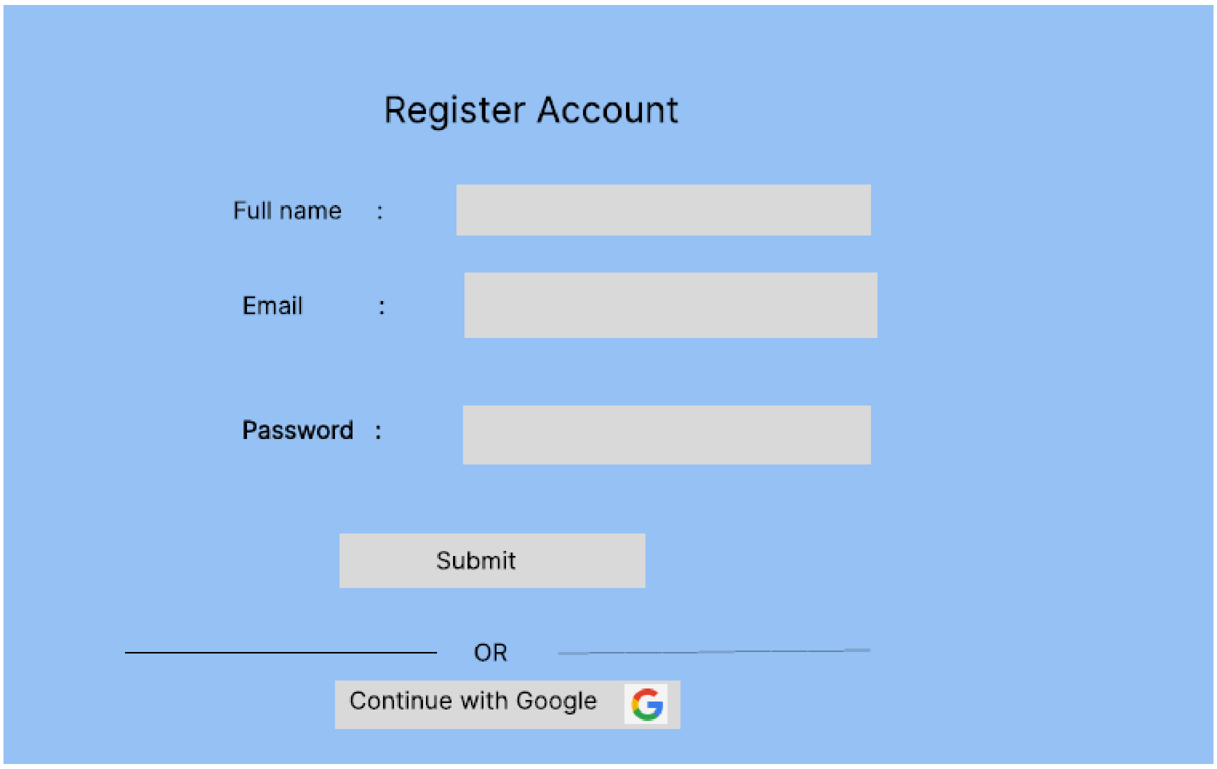
Fig 9:Colloboration diagram.

4.3 USER INTERFACE DESIGN USING FIGMA

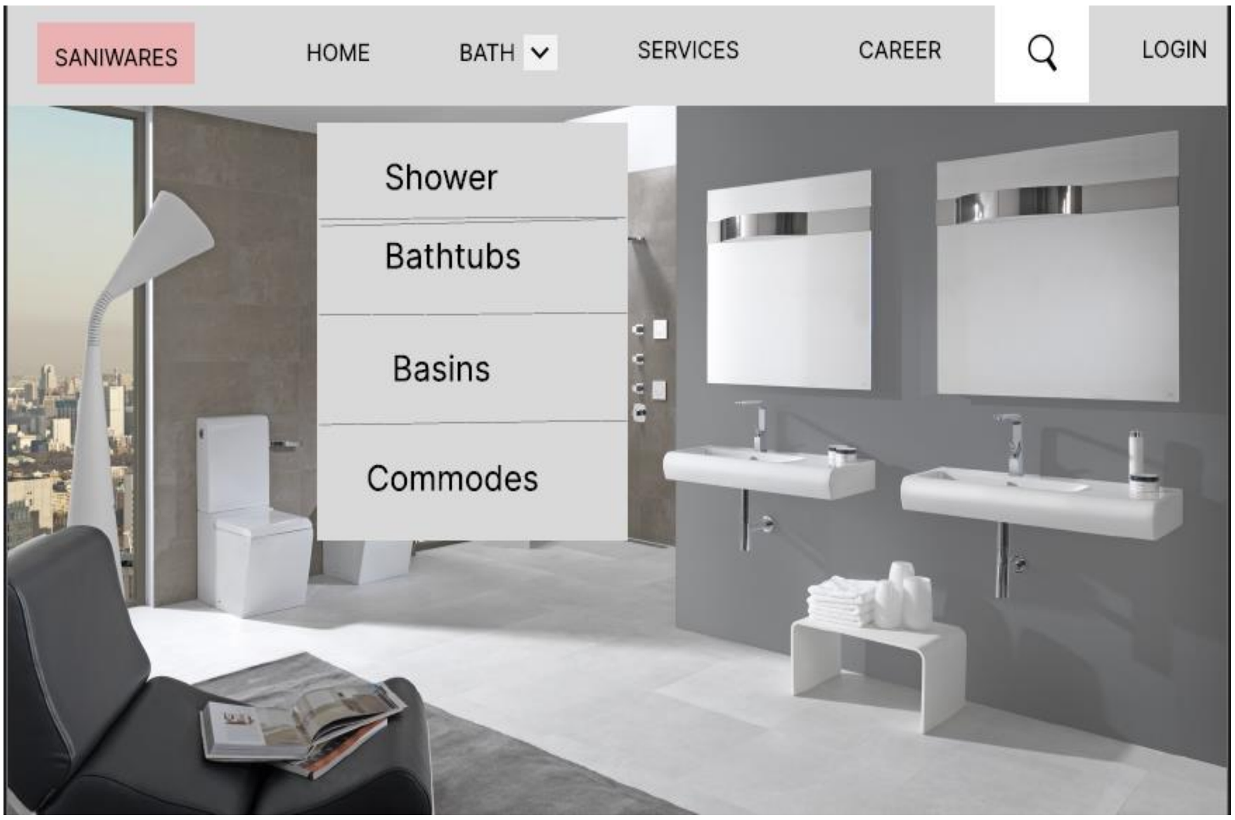
Form Name: Login form page



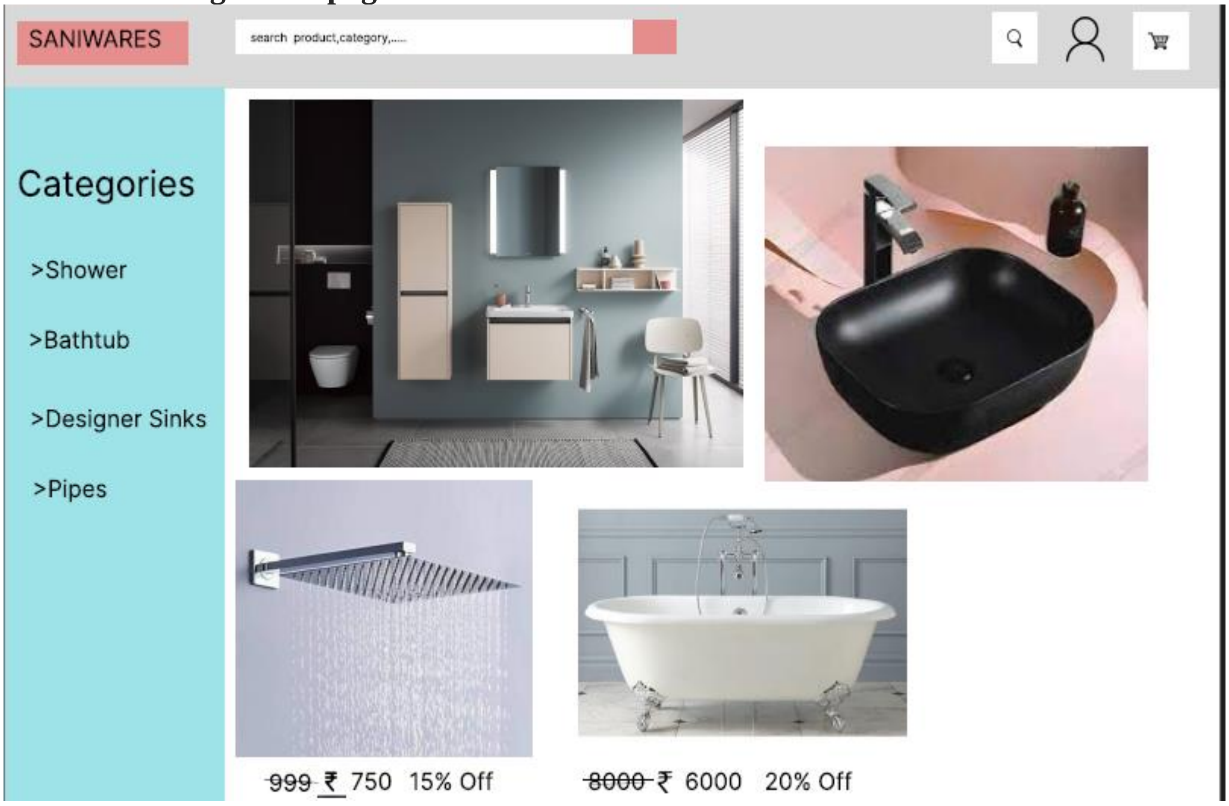
Form Name: User Registration Form



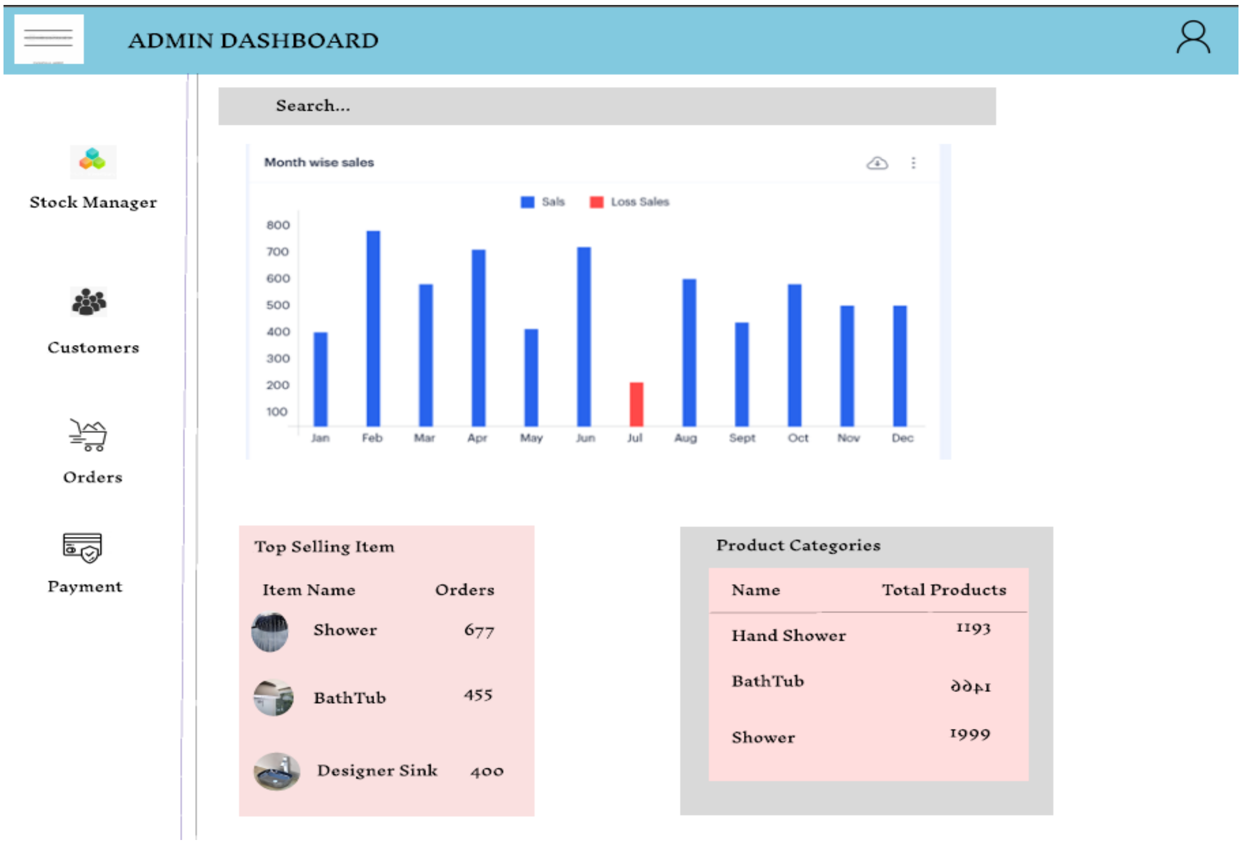
Form Name:Home page



Form Name:Login user page



Form Name:Admin Dashboard



Top Selling Item

Item Name	Orders
Shower	677
BathTub	455
Designer Sink	400

Product Categories

Name	Total Products
Hand Shower	1193
BathTub	1499
Shower	1999

4.4 DATABASE DESIGN

A database is a collection of data that has been organized to make it simple to manage and update. Information security could be one of the main aims of any database. The database design process is divided into two steps. The goal of the first step is to gather user requirements to create a database that as clearly as possible satisfies user needs. It is known as information-level design and is carried out without the aid of any DBMS. An information-level design is changed to a specific DBMS design that will be used to develop the system in the following stage. The physical-level design phase is where the characteristics of the specific DBMS are considered.

4.4.1 Relational Database Management System (RDBMS)

A Relational Database Management System (RDBMS) is a critical software application for organizing and managing data in a structured manner. It stores data in tables with rows and columns, where each row represents a record, and each column is a specific attribute or field. RDBMS systems like MySQL, Oracle, or Microsoft SQL Server ensure data integrity, enforce relationships between tables, and allow for efficient data retrieval and manipulation through Structured Query Language (SQL). These systems are widely used in various applications, such as e-commerce websites, financial systems, and inventory management, due to their robustness, scalability, and ability to handle complex data structures, making them essential for modern data-driven environments.

A Relational Database Management System (RDBMS) is a cornerstone of modern data management. It structures data into tables, where each row represents a unique entry, and each column corresponds to a specific attribute, ensuring a logical and organized storage system. RDBMS systems employ complex algorithms for data retrieval and manipulation, guaranteeing data consistency and adherence to predefined relationships between tables. Popular RDBMS software, including PostgreSQL, MySQL, and Microsoft SQL Server, provides robust features for transactions, data security, and scalability. These systems are integral to a myriad of applications, from healthcare records and customer databases to inventory control and financial platforms, supporting the efficient storage, retrieval, and analysis of vast datasets, making them essential tools in today's data-driven world.

4.4.2 Normalization

Normalization is a database design technique used in relational database management systems (RDBMS) to eliminate data redundancy and improve data integrity. It involves organizing data in a way that reduces data duplication and ensures that relationships between tables are defined and maintained.

The primary goals of normalization are:

Minimizing Data Redundancy: By breaking down data into separate tables and ensuring that each piece of data is stored only once, normalization helps reduce the chances of data inconsistencies or errors.

Ensuring Data Integrity: Normalization enforces the rules of referential integrity, which means that relationships between tables are well-defined and maintained. This ensures that data remains accurate and consistent.

Normalization typically involves dividing a database into multiple related tables and using primary keys and foreign keys to establish relationships between these tables. There are several normal forms, from First Normal Form (1NF) to Fifth Normal Form (5NF), each with specific rules and requirements. The level of normalization achieved depends on the specific needs of the database and the trade-off between data redundancy and query performance.

By applying normalization principles, designers can create efficient and reliable database structures that support data integrity and ease data maintenance and manipulation.

There are several normal forms (NF) that define specific rules and requirements for achieving progressively higher levels of normalization. Here are the most common normal forms, from First Normal Form (1NF) to Fifth Normal Form (5NF):

First Normal Form (1NF):

- Each table has a primary key.
- All columns contain atomic (indivisible) values.
- There are no repeating groups or arrays in columns.

Second Normal Form (2NF):

- The table is in 1NF.
- All non-key attributes are fully functionally dependent on the entire primary key. In other words, all non-key attributes must be dependent on the entire primary key, not just part of it.

Third Normal Form (3NF):

- The table is in 2NF.
- There is no transitive dependency, meaning that non-key attributes are not dependent on other non-key attributes.

Boyce-Codd Normal Form (BCNF):

- A stricter version of 3NF.
- It enforces that every non-trivial functional dependency involves a super key.

Fourth Normal Form (4NF):

- Addresses multi-valued dependencies.
- Eliminates any multi-valued dependencies within the data.

Fifth Normal Form (5NF):

- Also known as Project-Join Normal Form (PJ/NF).
- Handles cases where data can be derived by joining multiple tables.

4.4.3 Sanitization

In Python Django, sanitization refers to the process of cleaning and validating data to ensure that it is safe and free from malicious content before it is used or stored in a database. Sanitization is a crucial security practice to prevent various forms of attacks, such as cross-site scripting (XSS) and SQL injection, which can compromise the security and integrity of a web application.

4.4.4 Indexing

Indexing in Django is a fundamental database optimization technique. It involves creating data structures, or indexes, on specific fields to expedite data retrieval. These indexes act as signposts that enable the database to swiftly locate and fetch relevant data, especially in tables with substantial amounts of information. Django offers automatic index creation for primary keys, unique fields, and foreign keys. Additionally, developers can define custom indexes for fields frequently used in filtering, sorting, or searching. The choice of proper indexing plays a pivotal role in improving query performance, resulting in more responsive and scalable web applications. It's essential to consider application-specific query patterns and create indexes accordingly, as well as to understand the capabilities and limitations of the chosen database backend. Effective indexing is a cornerstone of efficient database operations in Django, contributing to enhanced application performance.

4.5 TABLE DESIGN

1.Tbl_users_reg

Eg.Primary key: **id**

No.	Field Name	Datatype (Size)	Key Constraints	Description of the field
1.	id	AutoField	Primary Key	Unique identifier for user
2.	username	CharField(150)	Unique	User’s username
3.	Password	CharField	Not Null	User’s password
4.	Email	EmailField	Unique	Users’email
5.	First_name	CharField(30)	Not Null	User’s first name
6.	Last_name	CharField(30)	Not Null	User’s last name
7.	role	CharField(100)	Default:””	Role or user type

2.Tbl_Product

Eg.Primary key: **product_id**

No.	Field name	Datatype (Size)	Key Constraints	Description of the field
1	product_id	Primary Key	Primary Key	Unique identifier for the product
2	product_name	CharField(255)	Not Null	Name of the product.
3	category	CharField(100)	Not Null	Category of the product.
4	subcategory	CharField(100)	Not Null	Subcategory of the product.
5	quantity	PositiveIntegerField	Not Null	Quantity available for the product
6	description	TextField	Not Null	Description of the product
7.	price	DecimalField(10,2)	Not Null	Price of the product.
8.	discount	DecimalField(5,2)	Default	Discount percentage for the product
9.	sale_price	DecimalField(10,2)	Editable: False	Calculated sale price of the product
10.	status	CharField(200)	Choices	Status of the product
11.	product_image	ImageField		Image field to upload product images.

3.Tbl_UserProfile

Eg.Primary key: **id**

Eg.Foreign key: **User_id** references table **Tbl_users_reg**

No.	Field name	Datatype (Size)	Key Constraints	Description of the field.
1.	Id	BigAutoField	Primary key	Unique Identification.
2.	User_id	OneToOneField	Foreign key	Reference to reg table
3.	Image	ImageField	Not Null	Image field for user profile picture
4.	First_name	CharField(255)	Not Null	User’s first name
5.	Last_name	CharField(255)	Not Null	User’s last name
6.	Phone_number	CharField(15)	Not Null	User’s phone number
7.	Address	TextField	Not Null	User’s address
8.	Pincode	CharField(10)	Not Null	Pincode of User’s location
9.	City	CharField(50)	Not Null	City of user’s location
10.	state	CharField(50)	Not Null	State of user’s location

4.Tbl_CartItem

Eg.Primary key: **id**

Eg.Foreign key: **cart,product** references table **Tbl_cart,Tbl_product.**

No.	Field Name	Datatype (Size)	Key Constraints	Description of the field.
1.	CartItem_id	BigAutoField	Primary key	CartItem id
2.	cart	ManyToMany	Foreign key	Reference to associated Cart1.
3.	Product	PositiveIntegerField	Foreign key	Reference to associated Product.
4.	quantity	PositiveIntegerField	Not Null	Quantity of the product in cart

5.Tbl_Cart

Eg.Primary key: **id**

Eg.Foreign key: **user,products** references table **Tbl_users_login,Tbl_CartItem.**

No.	Field Name	Datatype (Size)	Key Constraints	Description of the field
1.	cartid	BigAutoField	Primary key	Cart id
2.	user	OneToOne Field	Not null	Unique identifier
3.	products	ManyToMany	Through=’CartItem’	Reference associated products

6.Tbl_Order

Eg.Primary key: Order_**id**

Eg.Foreign key: **user_id,products** references table **Tbl_users_login,Tbl_OrderItems**

No.	Field Name	Datatype (Size)	Key Constraints	Description of the field
1.	Order_id	BigAutoField	Primary key	Unique identification
2.	User_id	OneToOneField	Foreign key	Reference to reg table
3.	products	ManyToMany Field	through='OrderItem'	Reference associated Product
4.	total_amount	Decimal(10,2)	Not Null	Total amount of the order
5.	payment_id	CharField(100)	Not Null	Identifier for the payment
6.	payment_status	BooleanField	Default=False	Status of payment(True/False)
7.	created_at	DateTimeField	auto_now_add	Date and time of order creation.

7.Tbl_OrderItem

Eg.Primary key: **OrderItem_id**

Eg.Foreign key: **order,product** references table **Tbl_Order,Tbl_Product**

No.	Field Name	Datatype (Size)	Key Constraints	Description of the Field
1.	Orderitem_id	CharField	Primary key	Unique Identification
1.	Order	ForeignKey	Foreign key	Reference to associated Order.
2.	Product	ForeignKey	Foreign key	Reference to associated Product
3.	Quantity	PositiveIntegerField	Foreign key	Quantity of the product in the order
4.	Item_total	DecimalField(10,2)	Not Null	Total cost of the items in the order

CHAPTER 5
SYSTEM TESTING

5.1 INTRODUCTION

Software testing is a way to check if the computer program works like it's supposed to. We use testing to make sure the software does what it is supposed to do. Validation means checking or testing things like software to make sure they meet the requirements and standards they are supposed to follow. Software testing is a way to check if a program works well. It goes along with other methods like checking and walking through the program. Validation means making sure that what the user wanted is what they got. There are several rules that can serve as testing objectives.

They are:

Testing is a process of executing a program with the intent of finding an error.

• A good test case is one that has high possibility of finding an undiscovered error.

• A successful test is one that uncovers an undiscovered error.

If a test works well and follows its goals, it can find mistakes in the software. The test showed that

the computer program is working like it's supposed to and is doing well.

There are three ways to test program.

• For correctness

• For implementation efficiency

• For computational complexity

5.2 TEST PLAN

A test plan suggests several required steps that need be taken to complete various testing methodologies. The activity that is to be taken is outlined in the test plan. A computer program, its documentation, and associated data structures are all created by software developers. It is always the responsibility of the software developers to test each of the program's separate components to make sure it fulfills the purpose for which it was intended. To solve the inherent issues with allowing the builder evaluate what they have developed, there is an independent test group (ITG). Testing's precise goals should be laid forth in quantifiable language. So that the mean time to failure, the cost to find and fix the defects, remaining defect density or frequency of occurrence and test work-hours per regression test all should be stated within the test.

The levels of testing include:

• Unit testing

• Integration Testing

• Data validation Testing & Output Testing

5.2.1 Unit Testing

Unit testing concentrates verification efforts on the software component or module, which is the smallest unit of software design. The component level design description is used as a guide when testing crucial control paths to find faults inside the module's perimeter. The level of test complexity and the untested area determined for unit testing. Unit testing is white-box focused, and numerous components may be tested simultaneously. To guarantee that data enters and exits the software unit under test properly, the modular interface is tested. To make sure that data temporarily stored retains its integrity during each step of an algorithm's execution, the local data structure is inspected. Boundary conditions are tested to ensure that all statements in a module have been executed at least once. Finally, all error handling paths are tested.Before starting any other test, tests of data flow over a module interface are necessary. All other tests are irrelevant if data cannot enter and depart the system properly. An important duty during the unit test is the selective examination of execution pathways. Error circumstances must be foreseen in good design, and error handling paths must be put up to cleanly reroute or halt work when an error does arise. The final step of unit testing is boundary testing. Software frequently fails at its limits.In the Sell-Soft System, unit testing was carried out by treating each module as a distinct entity and subjecting them to a variety of test inputs. The internal logic of the modules had some issues, which were fixed. Each module is tested and run separately after coding. All unused code was eliminated, and it was confirmed that every module was functional and produced the desired outcome

5.2.2 Integration Testing

Integration testing is a methodical approach for creating the program's structure while also carrying out tests to find interface issues. The goal is to construct a program structure that has been determined by design using unit tested components. The program is tested. Correction is challenging since the size of the overall program makes it challenging to isolate the causes. As soon as these mistakes are fixed, new ones arise, and the process repeats itself in an apparently unending cycle. All the modules were integrated after unit testing was completed in the system to check for an interface inconsistency. A distinctive program structure also developed when discrepancies in program structures.

5.2.3 Validation Testing or System Testing

The testing process comes to an end here. This involved testing the entire system in its entirety, including all forms, code, modules, and class modules. Popular names for this type of testing include system tests and black box testing. The functional requirements of the software are the main emphasis of the black box testing approach. That example, using Black Box testing, a software engineer can create sets of input conditions that will fully test every program requirement. The following sorts of problems are targeted by black box testing: erroneous or missing functions, interface errors, data structure or external data access errors, performance errors, initialization errors, and termination errors.

5.2.4 Output Testing or User Acceptance Testing

The system considered is tested for user acceptance; here it should satisfy the firm's need. The software should keep in touch with perspective system; user at the time of developing and making changes whenever required.

This done with respect to the following points:

• Input Screen Designs,

• Output Screen Designs,

The above testing is done taking various kinds of test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested using thatest data. While testing the system by which test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future.

5.2.5 Automation Testing

Automation Testing is a software testing technique that performs using special automated testing software tools to execute a test case suite. Essentially, it's a test to double-check that the equipment or software does exactly what it was designed to do. It tests for bugs, defects, and any other issues that can arise with product development. Although some types of testing, such as regression or functional testing can be done manually, there are greater benefits of doing it automatically. Automation testing can be run at any time of the day. It uses scripted sequences to examine the software. Automation developers generally write in the following programming languages: C#, JavaScript, and Ruby.

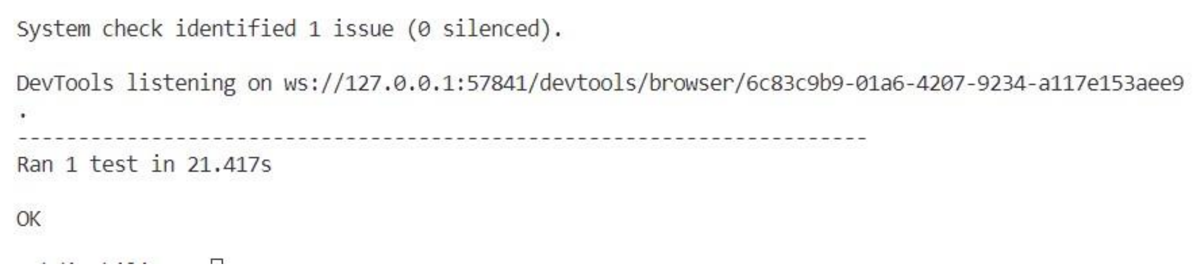
5.2.6 Selenium Testing

Selenium is an open-source automated testing framework used to verify web applications across different browsers and platforms. Selenium allows for the creation of test scripts in various programming languages such as Java, C#, and Python. Jason Huggins, an engineer at Thought Works, developed Selenium in 2004 while working on a web application that required frequent testing. He created a JavaScript program called "JavaScriptTestRunner" to automate browser actions and improve testing efficiency. Selenium has since evolved and continues to be developed by a team of contributors. In addition to Selenium, another popular tool used for automated testing is Cucumber. Cucumber is an open-source software testing framework that supports behavior-driven development (BDD). It allows for the creation of executable specifications in a human-readable format called Gherkin. One of the advantages of using Cucumber is its ability to bridge the gap between business stakeholders and technical teams. By using a common language, Cucumber facilitates effective communication and collaboration during the testing process. It promotes a shared understanding of the requirements and helps ensure that the developed software meets the intended business goals. Cucumber can be integrated with Selenium to combine the benefits of both tools. Selenium is used for interacting with web browsers and automating browser actions, while Cucumber provides a structured framework for organizing and executing tests. This combination allows for the creation of end-to-end tests that verify the behavior of web applications across different browsers and platforms, using a business-readable and maintainable format.

Test Case 1

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import unittest
import time
class LoginTest(unittest.TestCase):
    def setUp(self):
        self.driver = webdriver.Chrome()  # Initialize your WebDriver
        self.driver.get("http://127.0.0.1:3000/login/")  # Replace with your login page URL
    def test_valid_login(self):
        username = "Mohan1"
        password = "Mohan@1"
        # Find username and password fields
        username_field = WebDriverWait(self.driver, 30).until(
            EC.presence_of_element_located((By.ID, "username"))  # Update with your username
field locator
        )
        password_field = WebDriverWait(self.driver, 30).until(
            EC.presence_of_element_located((By.ID, "password"))  # Update with your password
field locator
        )
        # Enter username and password
        username_field.send_keys(username)
        password_field.send_keys(password)
        # Trigger change events after entering username and password
        self.driver.execute_script("arguments[0].dispatchEvent(new Event('change'))",
username_field)
        self.driver.execute_script("arguments[0].dispatchEvent(new Event('change'))",
password_field)
        # Find and click the login button
        login_button = WebDriverWait(self.driver, 30).until(
            EC.element_to_be_clickable((By.ID, "login-button"))  # Update with your login button locator
        )
        login_button.click()
        # Add assertions to verify successful login
        expected_url = "/home/"
        self.assertIn(expected_url, self.driver.current_url)  # Update with assertion to check if expected_url is present in the current URL
    def tearDown(self):
        self.driver.quit()
if __name__ == "__main__":
    unittest.main()
```

Eg.Screenshot



```
System check identified 1 issue (0 silenced).

DevTools listening on ws://127.0.0.1:57841/devtools/browser/6c83c9b9-01a6-4207-9234-a117e153aee9
.
----------------------------------------------------------------------
Ran 1 test in 21.417s

OK
```

Eg.Test Report

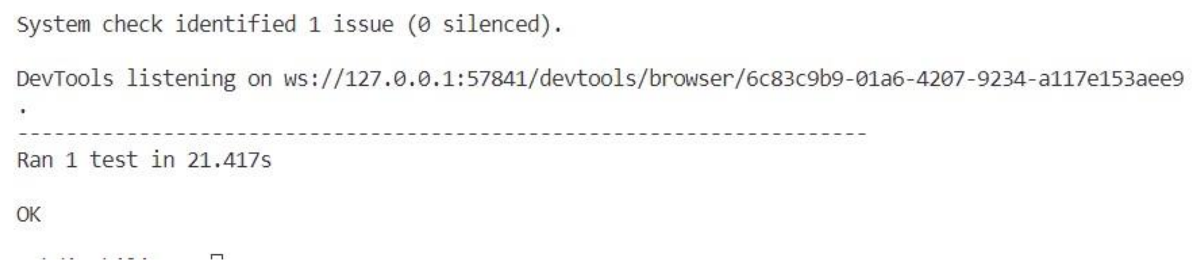
Test Case 1	
Project Name:	
Login Test Case	
Test Case ID: Test_1	**Test Designed By:** Meenu Anna Cherian
Test Priority(Low/Medium/High):High	**Test Designed Date:** 3/12/23
Module Name: Login Page	**Test Executed By :** Mr.Binumon Joseph
Test Title : Verify login with email and password	**Test Execution Date:** 4/12/23
Description: Testing the login page	
Pre-Condition : User has valid username and password	

Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/ Fail)
1	Navigation to login page		Login page should be displayed	Login page displayed	Pass
2	Provide valid username	Username:Mohan1	User should be able to login	User logged in and navigated To dashboard	Pass
3	Provide valid password	Mohan@1			
4	Click on Login button				
Post-Condition: User is validated with database and successfully login into account.The Account session details are logged in the database					

Test Case 2:

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
# Set up the WebDriver (assuming Chrome here)
driver = webdriver.Chrome()
try:
    # Open the login page
    driver.get("http://127.0.0.1:3000/login/")  # Change the URL accordingly
    # Fill out the login form with sample data
    username_input = driver.find_element(By.NAME, "username")
    password_input = driver.find_element(By.NAME, "password")
    username_input.send_keys("ashok@gmail.com")
    password_input.send_keys("ashok")
    # Submit the login form
    driver.find_element(By.ID, "login-button").click()
    # Navigate to the add product page
    driver.get("http://127.0.0.1:3000/addproduct/")  # Change the URL accordingly)
    # Fill out the form with sample data
    driver.find_element(By.ID, "product-name").send_keys("dishower")
    driver.find_element(By.ID, "category-name").send_keys("shower")
    driver.find_element(By.ID, "subcategory-name").send_keys("Standalone shower")
    driver.find_element(By.ID, "quantity").send_keys("12")
    driver.find_element(By.ID, "description").send_keys("This is a test product description.")
    driver.find_element(By.ID, "price").send_keys("100")
    driver.find_element(By.ID, "discount").send_keys("10")
    # Submit the form
    driver.find_element(By.ID, "add-product-button").click()
    # Open the view product page
    driver.get("http://127.0.0.1:3000/viewproduct/")  # Change the URL accordingly
if __name__ == "__main__":
    unittest.main()

finally:
    # Close the browser window
    driver.quit()
```

Screenshot



```
System check identified 1 issue (0 silenced).

DevTools listening on ws://127.0.0.1:57841/devtools/browser/6c83c9b9-01a6-4207-9234-a117e153aee9
.
----------------------------------------------------------------------
Ran 1 test in 21.417s

OK
```

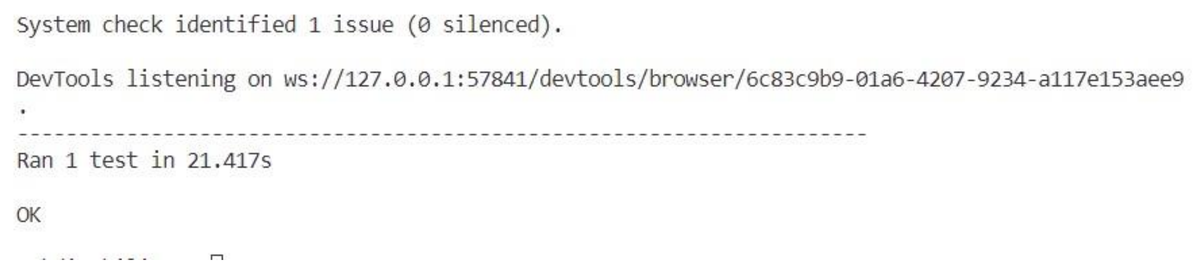
Test Case 2	
Project Name:	
Add Product	
Test Case ID: Test_2	**Test Designed By:** Meenu Anna Cherian
Test Priority(Low/Medium/High):High	**Test Designed Date: 4/12/23**
Module Name: Add Product	**Test Executed By : Mr.Binumon Joseph**
Test Title : Adding products to website.	**Test Execution Date: 4/12/23**

Description: Adding Product To website from vendor side					
Pre-Condition : User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigation to login page		Login page should be displayed	Vendor page displayed	Pass
2	Navigate to add product page		Vendor should able to add values	Vendor entered the value and added to the table.	Pass
3	Enter the values for each items	showerzz			
4	Added new product to the user page		User should be able to view the new product	User Viewed Newly added product	Pass
Post-Condition: Vendor logged in and successfully added product to the website.					

Test Case 3:

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import unittest

class LoginAndAddToCartTest(unittest.TestCase):
    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.get("http://127.0.0.1:3000/login/")

    def test_login_and_add_to_cart(self):
        username = "Mohan1"
        password = "Mohan@1"

        # Find username and password fields, and perform login
        username_field = WebDriverWait(self.driver, 30).until(
            EC.presence_of_element_located((By.ID, "username"))
        )
        password_field = WebDriverWait(self.driver, 30).until(
            EC.presence_of_element_located((By.ID, "password"))
        )

        username_field.send_keys(username)
        password_field.send_keys(password)

        # Trigger change events after entering username and password
        self.driver.execute_script("arguments[0].dispatchEvent(new      Event('change'))",
username_field)
        self.driver.execute_script("arguments[0].dispatchEvent(new      Event('change'))",
password_field)

        login_button = WebDriverWait(self.driver, 30).until(
            EC.element_to_be_clickable((By.ID, "login-button"))
        )
        login_button.click()

        # Wait for the login process to complete and then move to the target page
        WebDriverWait(self.driver, 30).until(

            EC.url_matches("http://127.0.0.1:3000/home/")
        )
        # Navigate to the 'http://127.0.0.1:3000/shower/built-in/' page
        self.driver.get("http://127.0.0.1:3000/shower/built-in/")

    def tearDown(self):
        self.driver.quit()

if __name__ == "_main_":
    unittest.main()
```

Screenshot



```
System check identified 1 issue (0 silenced).

DevTools listening on ws://127.0.0.1:57841/devtools/browser/6c83c9b9-01a6-4207-9234-a117e153aee9
.
----------------------------------------------------------------------
Ran 1 test in 21.417s

OK
```

Eg.Test report

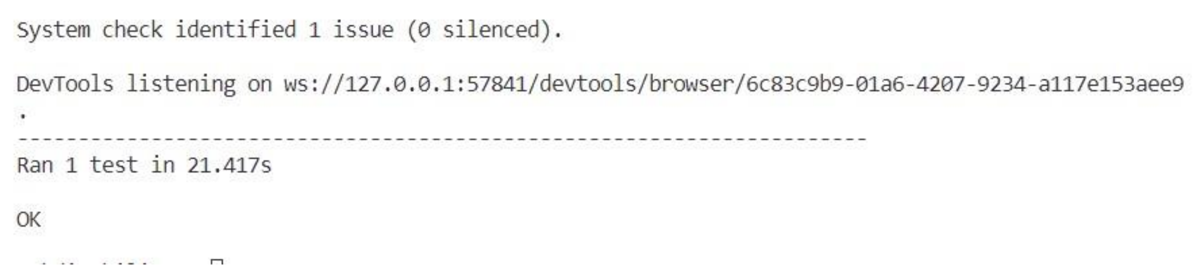
Test Case 3

Project Name:					
Product listing					
Test Case ID: Test_3			**Test Designed By:** Meenu Anna Cherian		
Test Priority(Low/Medium/High):High			**Test Designed Date: 4/12/23**		
Module Name: Product listing			**Test Executed By : Mr.Binumon Joseph**		
Test Title : Product listing for the customer			**Test Execution Date: 4/12/23**		
Description: Product should be displayed to the customer					
Pre-Condition : User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigation to login page		Login page should be displayed	Vendor page displayed	Pass
2	Navigate to home page		Home page should be seen.	Home page should be seen	Pass
3	Navigate to the product listing page.	showers			
4	Viewing products in the product listing page		User should be able to view the products	User Viewed Newly added products.	Pass
Post-Condition: Vendor logged in and successfully viewing product to the website.					

Test Case 4:

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import unittest
class LoginAndAddToCartTest(unittest.TestCase):
  def setUp(self):
    self.driver = webdriver.Chrome()
    self.driver.get("http://127.0.0.1:3000/login/")
  def test_login_and_add_to_cart(self):
    username = "Mohan1"
    password = "Mohan@1"
    # Find username and password fields, and perform login
    username_field = WebDriverWait(self.driver, 30).until(
      EC.presence_of_element_located((By.ID, "username"))
    )
    password_field = WebDriverWait(self.driver, 30).until(
      EC.presence_of_element_located((By.ID, "password"))
    )
    username_field.send_keys(username)
    password_field.send_keys(password)
    # Trigger change events after entering username and password
    self.driver.execute_script("arguments[0].dispatchEvent(new Event('change'))",
username_field)
    self.driver.execute_script("arguments[0].dispatchEvent(new Event('change'))",
password_field)
    login_button = WebDriverWait(self.driver, 30).until(
      EC.element_to_be_clickable((By.ID, "login-button"))
    )
    login_button.click()
    # Wait for the login process to complete and then move to the target page
    WebDriverWait(self.driver, 30).until(
      EC.url_matches("http://127.0.0.1:3000/home/")
    )
    # Navigate to the 'http://127.0.0.1:3000/shower/built-in/' page
    self.driver.get("http://127.0.0.1:3000/shower/built-in/")
    # Wait for the 'Add to Cart' button and click it
      add_to_cart_button = WebDriverWait(self.driver, 30).until(
       EC.element_to_be_clickable((By.ID, "add-to-cart-button"))
    # )
    # add_to_cart_button.click()
    # # Wait for the 'http://127.0.0.1:3000/cart/' page to load
     WebDriverWait(self.driver, 30).until(
       EC.url_matches("http://127.0.0.1:3000/cart/")
    # )
    # # Interact with the 'product-name' element on the cart page
      product_name_element = WebDriverWait(self.driver, 30).until(
       EC.presence_of_element_located((By.ID, "Showerzz"))
    # )
    # # Assuming you want to input some text in the 'product-name' field
     product_name_element.send_keys("showerzz")

    # # You can perform further actions/assertions as needed
  def tearDown(self):
    self.driver.quit()
if __name__ == "_main_":
  unittest.main()
```

Screenshot



```
System check identified 1 issue (0 silenced).

DevTools listening on ws://127.0.0.1:57841/devtools/browser/6c83c9b9-01a6-4207-9234-a117e153aee9
.
----------------------------------------------------------------------
Ran 1 test in 21.417s

OK
```

Test Case 4

Project Name:					
Add To Cart					
Test Case ID: Test_4			**Test Designed By:** Meenu Anna Cherian		
Test Priority(Low/Medium/High):High			**Test Designed Date:** 7/12/23		
Module Name: Add to cart			**Test Executed By :** Mr.Binumon Joseph		
Test Title : Product add to the cart			**Test Execution Date:** 7/12/23		
Description: Testing the add to cart					
Pre-Condition : User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigation to login page		Login page should be displayed	Login page displayed	Pass
2	Select category option	shower	Product listing page should be displayed	Product listing page displayed	Pass
3	Select subcategory option	Built-in-showers			
4	Add product to the cart page	Cart button	User should be able to view the carted items	User Viewed all carted items in the website	Pass
Post-Condition: Customer logged in and successfully adding product to the cart.					

CHAPTER 6
IMPLEMENTATION

6.1 INTRODUCTION

Implementation is the stage of the project where the theoretical design is turned into a working system. It can be the most crucial stage in achieving a successful new system gaining the users confidence that the new system will work and will be effective and accurate. It is primarily concerned with user training and documentation. Conversion usually takes place about the same time the user is being trained or later. Implementation simply means convening a new system design into operation, which is the process of converting a new revised system design into an operational one.

At this stage the main work load, the greatest upheaval and the major impact on the existing system shifts to the user department. If the implementation is not carefully planned or controlled, it can create chaos and confusion. Implementation includes all those activities that take place to convert from the existing system to the new system. The new system may be a totally new, replacing an existing manual or automated system or it may be a modification to an existing system. Proper implementation is essential to provide a reliable system to meet organization requirements. The process of putting the developed system in actual use is called system implementation. This includes all those activities that take place to convert from the old system to the new system. The system can be implemented only after through testing is done and if it is found to be working according to the specifications. The system personnel check the feasibility of the system. The more complex the system being implemented, the more involved will be the system analysis and design effort required to implement the three main aspects: education and training, system testing and changeover. The implementation state involves the following tasks:

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

6.2 IMPLEMENTATION PROCEDURES

The implementation procedures for this project involve a systematic and phased approach to bring the online crime reporting portal to fruition. This complex system, designed to cater to the needs of various stakeholders, requires careful planning and execution.

To commence the implementation, the project team will initiate a detailed requirements analysis phase. This stage involves comprehensive discussions with law enforcement officials, control room staff, prison wardens, and potential end-users to determine their specific needs and expectations. The results of this analysis will inform the development of a detailed system specification and design.The development phase will follow, involving the creation of the portal's architecture, databases, and user interfaces. It's during this stage that the functionalities outlined in the project's scope, including user management, crime reporting, and communication features, will be integrated into the system. The portal will be designed with a user-friendly interface to ensure ease of use for all categories of users.

Simultaneously, a dedicated team will work on the incorporation of advanced technologies, such as machine learning capabilities for crime trend analysis and predictive features. These technologies will be implemented carefully to ensure the portal's robustness and security.

Once the development phase is complete, rigorous testing will be conducted to identify and rectify any bugs or issues. The portal will undergo extensive quality assurance and user acceptance testing to ensure that it meets all the necessary requirements and is free of vulnerabilities.

Deployment of the system will be carried out in a controlled manner, ensuring minimal disruption to the existing processes, and allowing for gradual adaptation by the involved stakeholders. Comprehensive training sessions will be conducted to familiarize law enforcement personnel, control room staff, prison wardens, and registered users with the portal's functionality and features.

Post-deployment, the project team will continue to provide support and maintenance, addressing any issues that may arise and making necessary improvements as the system matures. Regular updates and security measures will be implemented to safeguard user data and maintain the portal's efficiency.

6.2.1 User Training

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

6.2.2 Training on the Application Software

After providing the necessary basic training on computer awareness the user will have to be trained on the new application software. This will give the underlying philosophy of the use of the new system such as the screen flow, screen design type of help on the screen, type of errors while entering the data, the corresponding validation check at each entry and the ways to correct the date entered. It should then cover information needed by the specific user/ group to use the system or part of the system while imparting the training of the program on the application. This training may be different across different user groups and across different levels of hierarchy.

6.2.3 System Maintenance

Maintenance is the enigma of system development. The maintenance phase of the software cycle is the time in which a software product performs useful work. After a system is successfully implemented, it should be maintained in a proper manner. System maintenance is an important aspect in the software development life cycle. The need for system maintenance is for it to make adaptable to the changes in the system environment. Software maintenance is of course, far more than “Finding Mistakes”.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The Online Sanitary Ware Store and Repairing Services Project is a comprehensive and user-centric web-based platform designed to revolutionize the sanitary ware shopping experience. By seamlessly combining the convenience of online shopping with the functionalities of a physical store, this project aims to provide users with an extensive catalog of sanitary products, detailed descriptions, and transparent pricing, ensuring a hassle-free browsing and purchasing journey. The platform prioritizes security through robust payment gateways, guaranteeing safe transactions for users.The project's modular structure, comprising of Admin, Stock Manager, and Customer modules, streamlines operations and enhances efficiency. The Admin module empowers administrators with complete oversight, allowing them to manage the website, view user details, and efficiently handle stock manager information and inventory. The Stock Manager module facilitates seamless stock management, enabling additions, updates, and deletions while ensuring accurate stock visibility. Meanwhile, the Customer module offers a user-friendly interface for account creation and management, product search, viewing, and purchasing. In conclusion, the project addresses the evolving needs of the sanitary ware industry by providing a robust, secure, and user-friendly online platform that mimics the experience of a physical store. Through its intuitive design and efficient modules, it offers a holistic solution for users seeking a diverse range of sanitary products while also streamlining backend operations for administrators and stock managers.

7.2 FUTURE SCOPE

The future scope of the Online Sanitary Ware Store and Repairing Services Project is promising and ripe with opportunities for expansion and enhancement. Moving forward, the platform could incorporate advanced features such as AI-powered product recommendations based on user preferences and browsing history, thereby personalizing the shopping experience. Implementing augmented reality (AR) or virtual reality (VR) tools could allow users to visualize how sanitary products would appear in their spaces, enhancing decision-making and customer satisfaction. Integration with IoT devices for smart sanitary solutions could also be explored, enabling remote monitoring and management of these products. Additionally, expanding the platform to include a wider array of related services like installation assistance, maintenance tutorials, and repair services could further establish it as a comprehensive solution in the sanitary ware industry. Furthermore, collaborating with manufacturers for exclusive deals or introducing a ratings and review system for products and services could foster trust and engagement among users, contributing to the platform's growth and success in the future. Overall, the project holds immense potential for evolution and expansion through technological advancements 2and strategic partnerships, catering to evolving consumer demands and industry trends.

CHAPTER 8
BIBLIOGRAPHY

REFERENCES:

- PankajJalote, “Software engineering: a precise approach”
- Gary B. Shelly, Harry J. Rosenblatt, “System Analysis and Design”, 2009
- Ken Schwaber, Mike Beedle, Agile Software Development with Scrum_,

  Pearson(2008)
- Roger S Pressman, “Software Engineering”
- IEEE Std 1016 Recommended Practice for Software Design Descriptions

WEBSITES:

- www.w3schools.com
- www.bootstrap.com
- https://chat.openai.com/chat
- www.jquery.com

CHAPTER 9

APPENDIX

9.1 Sample Code

Login.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login Page</title>
  <style>
    /* CSS for Login Form */
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f4;
    }
    .blue-header {
     background-color: #007bff; /* Blue color code */
     color: #fff; /* Text color */
     padding: 15px 0; /* Adjust padding as needed */
   }
    .login-container {
      max-width: 400px;
      margin: 0 auto;
      padding: 20px;
      background-color: #fff;
      border-radius: 5px;
      box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1);
      margin-top: 50px;
    }
    .login-container h2 {
      text-align: center;
      margin-bottom: 20px;
      color: #333;
    }
    .form-input {
      width: 100%;
      padding: 10px;
      margin-bottom: 10px;
      border: 1px solid #ccc;
      border-radius: 3px;
    }
    .login-button {
      width: 100%;
      background-color: #007bff; /* Blue color code */
      color: #fff;
      border: none;
      padding: 10px;
      border-radius: 3px;
      cursor: pointer;
   }
    .login-button:hover {
      background-color: #0056b3; /* Darker blue color code for hover effect */
    }
    .registration-link {
      text-align: center;
      margin-top: 10px;
    }
  </style>
  <link rel="stylesheet" href="{% static '/css/bootstrap.css' %}">
  <link rel="stylesheet" href="{% static '/vendors/linericon/style.css' %}">
  <link rel="stylesheet" href="{% static '/css/font-awesome.min.css' %}">
  <link rel="stylesheet" href="{% static '/css/themify-icons.css' %}">
  <link rel="stylesheet" href="{% static '/css/flaticon.css' %}">
  <link rel="stylesheet" href="{% static '/vendors/owl-carousel/owl.carousel.min.css' %}">
  <link rel="stylesheet" href="{% static '/vendors/lightbox/simpleLightbox.css' %}">
  <link rel="stylesheet" href="{% static '/vendors/nice-select/css/nice-select.css' %}">
  <link rel="stylesheet" href="{% static '/vendors/animate-css/animate.css' %}">
  <link rel="stylesheet" href="{% static '/vendors/jquery-ui/jquery-ui.css' %}">
```

```
  <link rel="stylesheet" href="{% static '/css/style.css' %}">
  <link rel="stylesheet" href="{% static '/css/responsive.css' %}">
</head>
<body>
  <!-- Header Area -->
  <header class="header_area blue-header">
    <div class="top_menu">
      <div class="container">
        <div class="row">
          <div class="col-lg-7">
            <div class="float-left">
              <!--<p>Phone: +01 256 25 235</p>
              <p>email: info@eiser.com</p>-->
            </div>
          </div>
          <div class="col-lg-5">
            <div class="float-right">
              <ul class="right_side">
                <li>
                  <!--<a href="cart.html">
                    gift card
                  </a>-->
                </li>
                <li>
                  <!--<a href="tracking.html">
                    track order
                  </a>-->
                </li>
                <li>
                  <!--<a href="contact.html">
                    Contact Us
                  </a>-->
                </li>
              </ul>
          </div>
         </div>
        </div>
       </div>
      </div>
      <div class="main_menu">
       <div class="container">
        <nav class="navbar navbar-expand-lg navbar-light w-100">
         <!-- Brand and toggle get grouped for better mobile display -->
         <a class="navbar-brand logo_h" href="index.html">
          <img src="{% static '/img/logo3.png' %}" alt="" />
         </a>
         <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent"
          aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle
navigation">
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
         </button>
         <!-- Collect the nav links, forms, and other content for toggling -->
         <!--<div class="collapse navbar-collapse offset w-100" id="navbarSupportedContent">
          <div class="row w-100 mr-0">
           <div class="col-lg-7 pr-0">
            <ul class="nav navbar-nav center_nav pull-right">
             <li class="nav-item active">
              <a class="nav-link" href="#">Home</a>
             </li>
             <li class="nav-item submenu dropdown">
              <a href="#" class="nav-link dropdown-toggle" data-toggle="dropdown"
role="button" aria-haspopup="true"
               aria-expanded="false">Shop</a>
              <ul class="dropdown-menu">
               <li class="nav-item">
```

```
                                <a class="nav-link" href="#">Shop Category</a>
                              </li>
                              <li class="nav-item">
                                <a class="nav-link" href="#">Product Details</a>
                              </li>
                              <li class="nav-item">
                                <a class="nav-link" href="#">Product Checkout</a>
                              </li>
                              <li class="nav-item">
                                <a class="nav-link" href="#">Shopping Cart</a>
                              </li>
                            </ul>
                          </li>
                          <li class="nav-item submenu dropdown">
                            <a href="#" class="nav-link dropdown-toggle" data-toggle="dropdown"
role="button" aria-haspopup="true"
                              aria-expanded="false">Services</a>
                            <ul class="dropdown-menu">
                              <!--<li class="nav-item">
                                <a class="nav-link" href="#">Services</a>
                              </li>
                              <li class="nav-item">
                                <a class="nav-link" href="#">Blog Details</a>
                              </li>-->
                            <!--</ul>
                          </li>
                          <li class="nav-item submenu dropdown">
                            <a href="#" class="nav-link dropdown-toggle" data-toggle="dropdown"
role="button" aria-haspopup="true"
                              aria-expanded="false">Career</a>
                            <ul class="dropdown-menu">
                              <li class="nav-item">
                                <a class="nav-link" href="#">Apply now</a>
                              </li>
```

```
          <!--<li class="nav-item">
            <a class="nav-link" href="#">Elements</a>
          </li>-->
        <!--</ul>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">About</a>
      </li>
    </ul>
  </div>
  <div class="col-lg-5 pr-0">
    <ul class="nav navbar-nav navbar-right right_nav pull-right">
      <li class="nav-item">
        <a href="#" class="icons">
          <i class="ti-search" aria-hidden="true"></i>
        </a>
      </li>
      <li class="nav-item">
        <a href="#" class="icons">
          <i class="ti-shopping-cart"></i>
        </a>
      </li>
      <li class="nav-item">
        <a href="login" class="icons">
          <i class="ti-user" aria-hidden="true"></i>
        </a>
      </li>-->
      <!--<li class="nav-item">
        <a href="{% url 'logout' %}" class="icons">
          <!--<i class="fa fa-power-off"></i>-->
            <!--<i class="ti-heart" aria-hidden="true"></i>
        </a>
      </li>-->
    <!--</ul>-->
```

```
                    </div>
                  </div>
                </div>
              </nav>
            </div>
          </div>
    </header>
    <!-- Login Form Content -->
    <div class="login-container">
        <h2>Login</h2>
        <form class="login-form" action="#" method="POST" onsubmit="return validateForm()">
            {% csrf_token %}
            <input class="form-input" type="text" name="username" id="username"
placeholder="Username" required>
            <span class="username-error" id="username-error"></span>
            <input class="form-input" type="password" name="password" id="password"
placeholder="Password" required>
            <span class="password-error" id="password-error"></span>
            <button class="login-button" type="submit">Login</button>
        </form>
        <!--<p class="forgot-password"><a href="#">Forgot your password?</a></p>-->
        <p class="registration-link">Don't have an account? <a href="{% url
'registration'%}">Register here</a></p>
        <p class="forgot-password"><a href="{% url 'password_reset' %}">Forgot
Password?</a></p>
    </div>
    <!-- JavaScript and other scripts -->
    <!-- Bootstrap JS and other script includes go here -->
</body>
</html>
```

Cart.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
```

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title> SaniwareFix Shopping Cart</title>
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" />
 <title>Eiser ecommerce</title>
 <link rel="stylesheet" href="{% static '/css/bootstrap.css' %}" />
 <link rel="stylesheet" href="{% static '/vendors/linericon/style.css' %}" />
 <link rel="stylesheet" href="{% static '/css/font-awesome.min.css' %}" />
 <link rel="stylesheet" href="{% static '/css/themify-icons.css' %}" />
 <link rel="stylesheet" href="{% static '/css/flaticon.css' %}" />
 <link rel="stylesheet" href="{% static '/vendors/owl-carousel/owl.carousel.min.css' %}" />
 <link rel="stylesheet" href="{% static '/vendors/lightbox/simpleLightbox.css' %}" />
 <link rel="stylesheet" href="{% static '/vendors/nice-select/css/nice-select.css' %}" />
 <link rel="stylesheet" href="{% static '/vendors/animate-css/animate.css' %}" />
 <link rel="stylesheet" href="{% static '/vendors/jquery-ui/jquery-ui.css' %}" />
 <!-- main css -->
 <link rel="stylesheet" href="{% static '/css/style.css' %}" />
 <link rel="stylesheet" href="{% static '/css/responsive.css' %}" />
  <style>
    .container {
      max-width: 800px;
      margin: 0 auto;
      padding: 20px;
      background-color: #f4f4f4;
      border-radius: 8px;
      box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  }
  .cart-header {
      text-align: center;
      margin-bottom: 20px;
  }
  .cart-item {
      display: flex;
      justify-content: space-between;
      align-items: center;
      padding: 10px;
      border-bottom: 1px solid #ddd;
  }
  .cart-item-details {
      display: flex;
      align-items: center;
  }
  .cart-item-name {
      font-weight: bold;
      margin-right: 10px;
      flex: 1;
      white-space: nowrap;
      overflow: hidden;
      text-overflow: ellipsis;
  }
  .remove-from-cart-btn,
  .quantity-btn {
      background-color: #ff6347;
      color: white;
      border: none;
      border-radius: 4px;
      padding: 5px 10px;
      cursor: pointer;
      transition: background-color 0.3s;
  }
  .remove-from-cart-btn:hover,
  .quantity-btn:hover {
      background-color: #e74c3c;
  }
  .cart-item-quantity {
      display: flex;
      align-items: center;
```

```
        margin-right: 10px;
    }
    .cart-item-price {
        font-weight: bold;
    }
    .continue-shopping-link {
        display: inline-block;
        margin-top: 20px;
        text-decoration: none;
        color: #3498db;
        font-weight: bold;
    }
    .continue-shopping-link:hover {
        text-decoration: underline;
    }
    .checkout-button {
        display: inline-block;
        background-color: #28a745;
        color: white;
        padding: 10px 20px;
        border-radius: 4px;
        margin-top: 20px;
        margin-left: 400px;
        text-decoration: none;
        font-weight: bold;
        transition: background-color 0.3s;
    }
    .checkout-button:hover {
        background-color: #218838;
    }
      .search-bar {
          float: right; /* Float the search bar to the right */
      }
      .search-bar input[type="text"] {
          width: 150px; /* Adjust the width as needed */
      }
      .search-bar button {
          background: none;
          border: none;
          cursor: pointer;
      }
      .search-bar {
          float: right; /* Float the search bar to the right */
      }
      .search-bar input[type="text"] {
          width: 150px; /* Adjust the width as needed */
      }
      .search-bar button {
          background: none;
          border: none;
          cursor: pointer;
      }
    .small-profile-image {
      width: 30px; /* Adjust width as needed */
      height: 30px; /* Adjust height as needed */
    }
    .default-profile-image {
      width: 30px; /* Set the desired width for the default image */
      height: 30px; /* Set the desired height for the default image */
   }
    .header_area {
      width: 100%; /* Full width */
      background-color: #112418; /* Background color */
      color: white; /* Text color */
      text-align: center; /* Align text within the header */
      padding: 10px 0; /* Adjust padding as needed */
      position: relative; /* Enable positioning */
      left: 50%; /* Shift the header to the right by 50% */
```

```
      transform: translateX(-50%); /* Center the header */
    }
/* Make header-menu items display horizontally */
.header-menu {
  list-style: none;
  padding: 0;
  margin: 0;
  display: flex;
  justify-content: flex-start; /* Align menu items to the right */
   /* Align items vertically */
}
.header-menu li {
  margin-right: 10px; /* Adjust the spacing as needed */
}
.header-menu a {
  text-decoration: none;
  color: #fff; /* Adjust the color as needed */
}
.main_menu {
  display: flex;
  justify-content: space-between; /* Distribute items evenly */
  align-items: center; /* Align items vertically */
}
.navbar-brand img {
  max-width: 200px; /* Adjust the width if necessary */
  height: auto; /* Maintain aspect ratio */
}
    </style>
</head>
<body>
  <!--================Header Menu Area =================-->
  <header class="header_area">
    <div class="top_menu">
      <div class="container">
    <div class="row">
      <!--<div class="col-lg-7">
        <div class="float-left">
          <p>Phone: +01 256 25 235</p>
          <p>email: info@eiser.com</p>
        </div>
      </div>-->
      <div class="col-lg-5">
        <div class="float-right">
          <ul class="right_side">
            <li>
              <!--<a href="cart.html">
                gift card
              </a>-->
            </li>
            <li>
              <!--<a href="tracking.html">
                track order
              </a>-->
            </li>
            <li>
              <!--<a href="contact.html">
                Contact Us
              </a>-->
            </li>
          </ul>
        </div>
      </div>
    </div>
  </div>
</div>
<div class="main_menu">
  <div class="container">
    <nav class="navbar navbar-expand-lg navbar-light w-100">
```

```
          <!-- Brand and toggle get grouped for better mobile display -->
          <a class="navbar-brand logo_h" href="index.html">
            <img src="{% static '/img/logo3.png' %}" alt="" />
          </a>
          <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent"
            aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle
navigation">
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </button>
          <!-- Collect the nav links, forms, and other content for toggling -->
          <div class="collapse navbar-collapse offset w-100" id="navbarSupportedContent">
            <div class="row w-100 mr-0">
              <div class="col-lg-7 pr-0">
                <ul class="nav navbar-nav center_nav pull-right">
                  <li class="nav-item active">
                    <a class="nav-link" href="{% url 'home' %}">Home</a>
                  </li>
                </ul>
              </div>
            </div>
          </div>
        </nav>
      </div>
    </div>
  </header>
<body>
  <div class="container">
    <div class="cart-header">
        <h1>Your Cart</h1>
    </div>
    <ul>
    {% for item in cart_items %}
    <li class="cart-item">
      <div class="cart-item-details">
        <div class="cart-item-name">{{ item.product.product_name }}</div>
        <form action="{% url 'remove-from-cart' item.product.product_id %}" method="post">
          {% csrf_token %}
          <button class="remove-from-cart-btn" type="submit">Remove</button>
        </form>
      </div>
      <div class="item-quantity">
        <form action="{% url 'increase-cart-item' item.product.product_id %}" method="post">
          {% csrf_token %}
          <button class="quantity-btn increase-quantity" type="submit">+</button>
        </form>
        <span class="cart-item-quantity" data-quantity="{{ item.quantity }}">{{ item.quantity
}}</span>
        <form action="{% url 'decrease-cart-item' item.product.product_id %}"
method="post">
          {% csrf_token %}
          <button class="quantity-btn decrease-quantity" type="submit">-</button>
        </form>
      </div>
      <div class="cart-item-price" data-price="{{ item.product.sale_price }}">
        &#8377;{{ item.product.sale_price }}
      </div>
    </li>
    {% endfor %}
  </ul>
  <p class="total-price-data"><span id="total-price"></span></p>
  <a class="continue-shopping-link" href="{% url 'home' %}">Continue Shopping</a>
  <a class="checkout-button" href="{% url 'checkout' %}">Checkout</a>
</div>
<script>
 document.addEventListener("DOMContentLoaded", function() {
    var showersDropdown = document.getElementById('showersDropdown');
    var basinsDropdown = document.getElementById('basinsDropdown');
    var showerSubCategories = document.getElementById('showerSubCategories');
    var basinsSubCategories = document.getElementById('basinsSubCategories');
    showersDropdown.addEventListener('click', function(event) {
      event.preventDefault();
      event.stopPropagation();
      basinsSubCategories.classList.remove('show');
      showerSubCategories.classList.toggle('show');
    });
    basinsDropdown.addEventListener('click', function(event) {
      event.preventDefault();
      event.stopPropagation();
      showerSubCategories.classList.remove('show');
      basinsSubCategories.classList.toggle('show');
    });
    document.addEventListener("click", function(e) {
      if (!e.target.closest('#showersDropdown') && !e.target.closest('#showerSubCategories')) {
        showerSubCategories.classList.remove('show');
      }
      if (!e.target.closest('#basinsDropdown') && !e.target.closest('#basinsSubCategories')) {
        basinsSubCategories.classList.remove('show');
      }
    });
  });
</script>
  <script>
   document.addEventListener("DOMContentLoaded", function () {
    // Your existing increase and decrease buttons logic
    const increaseButtons = document.querySelectorAll(".increase-quantity");
    const decreaseButtons = document.querySelectorAll(".decrease-quantity");
    increaseButtons.forEach((button) => {
      button.addEventListener("click", (event) => {
        event.preventDefault();
```

```
            const currentItem = event.target.closest(".cart-item");
            const quantityElement = currentItem.querySelector(".cart-item-quantity");
            const priceElement = currentItem.querySelector(".cart-item-price");
            const pricePerItem = parseFloat(priceElement.getAttribute("data-price"));
            let currentQuantity = parseInt(quantityElement.textContent);
            currentQuantity++;
            quantityElement.textContent = currentQuantity;
            const totalPrice = (pricePerItem * currentQuantity).toFixed(2);
            priceElement.textContent = "Rs" + totalPrice;
            updateTotalPrice();
            saveCartData(); // Save cart data to localStorage after each change
        });
    });
    decreaseButtons.forEach((button) => {
        button.addEventListener("click", (event) => {
            event.preventDefault();
            const currentItem = event.target.closest(".cart-item");
            const quantityElement = currentItem.querySelector(".cart-item-quantity");
            const priceElement = currentItem.querySelector(".cart-item-price");
            const pricePerItem = parseFloat(priceElement.getAttribute("data-price"));
            let currentQuantity = parseInt(quantityElement.textContent);
            if (currentQuantity > 1) {
                currentQuantity--;
                quantityElement.textContent = currentQuantity;
                const totalPrice = (pricePerItem * currentQuantity).toFixed(2);
                priceElement.textContent = "Rs" + totalPrice;
                updateTotalPrice();
                saveCartData(); // Save cart data to localStorage after each change
            }
        });
    });
    loadCartData();
    function loadCartData() {
        const itemElements = document.querySelectorAll(".cart-item");
        itemElements.forEach((itemElement) => {
            const productId = itemElement.dataset.productId;
            const quantity = parseInt(localStorage.getItem(productId)) || 1; // Default quantity set to 1
            updateCartItemQuantity(itemElement, quantity);
        });
        updateTotalPrice();
    }
    function updateCartItemQuantity(itemElement, quantity) {
        const quantityElement = itemElement.querySelector(".cart-item-quantity");
        const priceElement = itemElement.querySelector(".cart-item-price");
        const pricePerItem = parseFloat(priceElement.getAttribute("data-price"));
        quantityElement.textContent = quantity;
        const totalPrice = (pricePerItem * quantity).toFixed(2);
        priceElement.textContent = "Rs" + totalPrice;
    }
    function updateTotalPrice() {
        let total = 0;
        const itemElements = document.querySelectorAll(".cart-item");
        itemElements.forEach((itemElement) => {
            const price = parseFloat(itemElement.querySelector(".cart-item-
price").getAttribute("data-price"));
            const quantity = parseInt(itemElement.querySelector(".cart-item-quantity").textContent);
            total += price * quantity;
        });
        const totalPriceElement = document.getElementById("total-price");
        totalPriceElement.textContent = "Total Price Rs" + total.toFixed(2);
    }
    function saveCartData() {
        const itemElements = document.querySelectorAll(".cart-item");
        itemElements.forEach((itemElement) => {
            const productId = itemElement.dataset.productId;
            const quantity = parseInt(itemElement.querySelector(".cart-item-quantity").textContent);
            localStorage.setItem(productId, quantity);
        });
```

```
      }
    });
  </script>
</body>
</html>
```

Addproduct.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Online Sanitary Ware Shop Admin</title>
    <style>
        body {
            margin: 0;
            font-family: Arial, sans-serif;
        }
        .navbar {
            background-color: #333;
            color: white;
            padding: 15px 0;
        }
        .navbar-container {
            display: flex;
            justify-content: space-between;
            align-items: center;
            margin: 0 20px;
        }
        .profile {
            display: flex;
            align-items: center;
        }
```

```
 .profile-pic {
    width: 40px;
    height: 40px;
    border-radius: 50%;
    margin-right: 10px;
 }
 .welcome {
    font-weight: bold;
 }
 .navbar-menu a {
    color: white;
    text-decoration: none;
    margin: 0 15px;
 }
 .content {
    display: flex;
}
 .sidebar {
    width: 200px;
    background-color: #f4f4f4;
    padding: 20px;
}
 .sidebar h2 {
    margin-bottom: 20px;
 }
 .sidebar ul {
    list-style: none;
    padding: 0;
 }
 .sidebar li {
    margin-bottom: 10px;
    cursor: pointer;
}
 .sidebar li {
    margin-bottom: 10px;
    padding: 10px;
    border-radius: 5px;
    background-color: #ddd;
}
.sidebar li:nth-child(1) {
    background-color: #ff9999; /* Add Products */
}
.sidebar li:nth-child(2) {
    background-color: #66b3ff; /* View Products */
}
.sidebar li:nth-child(3) {
    background-color: #99ff99; /* Analytics */
}
.sidebar li:nth-child(4) {
    background-color: #ffcc99; /* My Orders */
}
.sidebar li:nth-child(5) {
    background-color: #ffcc00; /* Partners */
}
.sidebar li:nth-child(6) {
    background-color: #ccffcc; /* Calendar */
}
.sidebar li:nth-child(7) {
    background-color: #ffccff; /* Users */
}
.main-content {
    flex: 1;
    padding: 20px;
}
body {
    font-family: Arial, sans-serif;

    background-size: cover;
```

```
        background-repeat: no-repeat;
        margin: 0;
    }
    .header {
        background-color: #003300; /* Dark green color for the header */
        color: #fff;
        text-align: center;
        padding: 10px;
     }
    .footer {
        background-color: #003300; /* Dark green color for the footer */
        color: #fff;
        text-align: center;
        padding: 10px;
    }
    .form-container {
        background-color: #fff;
        border: 1px solid #ddd;
        padding: 20px;
        width: 40%;
        margin: 0 auto;
        box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
    }
    .form-group label {
        font-weight: bold;
    }
    .form-control {
        width: 100%;
        padding: 10px;
        margin-top: 5px;
        margin-bottom: 15px;
        border: 1px solid #ccc;
        border-radius: 4px;
        box-sizing: border-box;
```

```
            transition: border-color 0.15s ease-in-out, box-shadow 0.15s ease-in-out;
        }
        .btn-primary {
            background-color: #79bd82;
            color: #fff;
            border: none;
            padding: 10px 20px;
            border-radius: 4px;
            cursor: pointer;
        }
        .btn-primary:hover {
            background-color: #27472b;
        }
        .sidebar ul li a {
            text-decoration: none; /* Remove underline from anchor elements */
            color: inherit; /* Use the default text color for the links */
       }
    </style>
</head>
<body>
    <nav class="navbar">
        <div class="navbar-container">
            <div class="profile">
                <!--<img src="profile-pic.jpg" alt="Profile Picture" class="profile-pic">-->
                <span class="welcome">Welcome, Stock Manager</span>
            </div>
        </div>
    </nav>
    <div class="content">
        <div class="sidebar">
            <h2>Options</h2>
            <ul>
                <li><a href="{% url 'addproduct' %}">Add Products</a></li>
                <li>View Products</li>
                <li>Analytics</li>
                <li>My Orders</li>
                <li>Partners</li>
                <li>Calendar</li>
                <li>Users</li>
                <li>Sales</li>
                <li>Orders Received</li>
                <li>Add Product Here</li>
                <li>View Your Products</li>
            </ul>
        </div>
        <div class="main-content">
            <!-- Content for orders received, add product here, view your products -->
            <div class="form-container" id="product-form">
                <form action="{% url 'addproduct' %}" method="post" enctype="multipart/form-data"
onsubmit="return validateProductForm();">
                    {% csrf_token %}
                    <div class="form-group">
                        <label for="product-name">Product Name</label>
                        <input type="text" id="product-name" name="product-name" class="form-control"
required oninput="validateProductName(this)">
                        <span id="product-name-error" class="error-text"></span>
                    </div>
                    <div class="form-group">
                        <label for="category-name">Category Name</label>
                        <select id="category-name" name="category-name" class="form-control" required
onchange="updateSubcategoryOptions()">
                            <option value="">Select</option>
                            <option value="shower">Shower</option>
                            <option value="basins">Basins</option>
                            <!--<option value="seed">Seed</option>
                            <option value="accessories">Accessories</option>
                            <!-- Add more categories as needed -->
                        </select>
```

```
          </div>
          <div class="form-group">
            <label for="subcategory-name">Subcategory Name</label>
            <select id="subcategory-name" name="subcategory-name" class="form-control"
required>
              <!-- Subcategory options will be added dynamically here -->
            </select>
          </div>
          <div class="form-group">
            <label for="quantity">Quantity</label>
            <input type="number" id="quantity" name="quantity" class="form-control"
required>
          </div>
          <div class "form-group">
            <label for="description">Description</label>
            <textarea id="description" name="description" class="form-control"
required></textarea>
          </div>
          <div class="form-group">
            <label for="price">Product Price</label>
            <input type="number" id="price" name="price" class="form-control" required
oninput="validatePrice(this); calculateSalePrice(document.getElementById('discount'))">
            <span id="price-error" class="error-text"></span>
          </div>
          <div class="form-group">
            <label for="discount">Discount (%)</label>
            <input type="number" id="discount" name="discount" class="form-control"
oninput="calculateSalePrice(this)">
          </div>
          <div class="form-group">
            <label for="sale-price">Sale Price</label>
            <input type="number" id="sale-price" name="sale-price" class="form-control"
readonly>
          </div>
```

```
                <div class="form-group">
                    <label for="status">Status</label>
                    <select id="status" name="status" class="form-control" required>
                        <option value="active">Active</option>
                        <option value="inactive">Inactive</option>
                    </select>
                </div>
                <div class="form-group">
                    <label for="product-image">Product Image</label>
                    <input type="file" id="product-image" name="product-image" class="form-
control" accept="image/*" required onchange="validateProductImage(this)">
                    <span id="product-image-error" class="error-text"></span>
                </div><!-- Rest of your form content -->
                <button type="submit" id="add-product-button" class="btn btn-primary">Add
Product</button>
                <button type="reset" class="btn btn-primary">Reset</button>
            </form>
        </div>

    </div>
  </div>
  <script>
    function updateSubcategoryOptions() {
        const categorySelect = document.getElementById('category-name');
        const subcategorySelect = document.getElementById('subcategory-name');
        // Define subcategory options for each category
        const subcategoryOptions = {
            shower: ['built-in showers', 'Standalone shower'],
            basins: ['Full Pedestal Basins','Semi Pedestal Basins'],
             // No subcategories
            // Add more categories and options as needed
        };
        const selectedCategory = categorySelect.value;
        // Clear existing options
        subcategorySelect.innerHTML = '';
        // Add new options based on the selected category
        for (const option of subcategoryOptions[selectedCategory] || []) {
            const optionElement = document.createElement('option');
            optionElement.value = option;
            optionElement.textContent = option;
            subcategorySelect.appendChild(optionElement);
        }
    }
     function calculateSalePrice(discountInput) {
    const priceInput = document.getElementById('price');
    const salePriceInput = document.getElementById('sale-price');
    const discount = discountInput.value || 0;
    const price = priceInput.value || 0;
    const discountedPrice = price - (price * (discount / 100));
    salePriceInput.value = discountedPrice.toFixed(2);
}
function validatePrice(input) {
    const price = input.value;
        const priceError = document.getElementById('price-error');
        // Check if the input is a positive number
        if (isNaN(price) || price <= 0) {
            priceError.textContent = 'Price must be a positive number.';
            input.classList.add('is-invalid');
            return false;
        } else {
            priceError.textContent = ''; // Clear the error message
            input.classList.remove('is-invalid');
            return true;
       }
            calculateSalePrice(document.getElementById('discount'));
        }
            updateSubcategoryOptions();
```

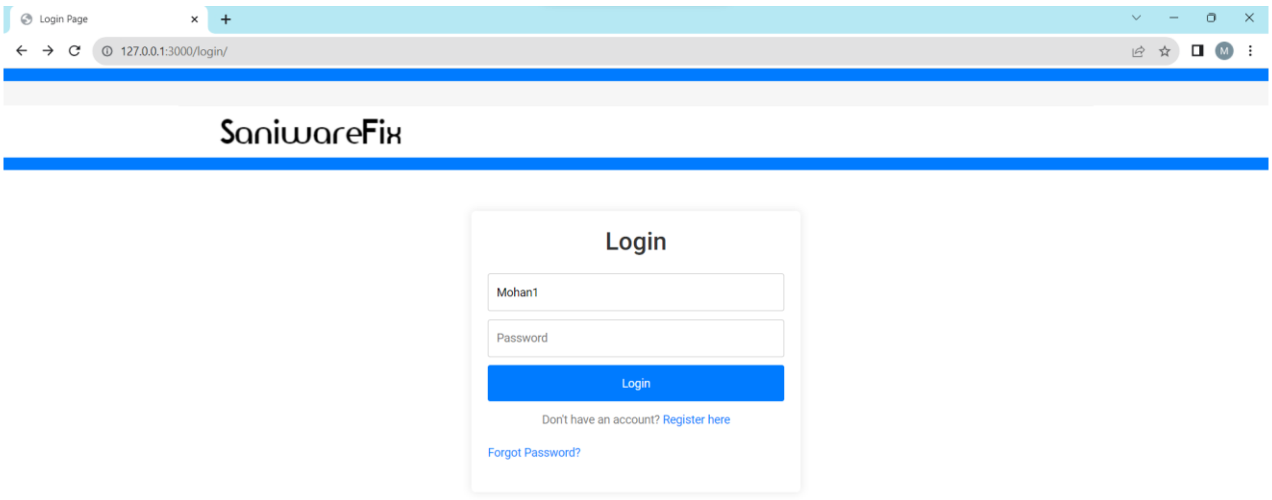
```
</script>
<script>
  function preventZeroAndNegative(event) {
    const input = event.target;
    const value = parseFloat(input.value.trim()
    if (value <= 0 || isNaN(value)) {
      input.value = ''; // Clear the input field if zero or a negative number is entered
    }
  }
  document.addEventListener('DOMContentLoaded', function() {
    const quantityInput = document.getElementById('quantity');
    const priceInput = document.getElementById('price');
    const discountInput = document.getElementById('discount');
    quantityInput.addEventListener('input', preventZeroAndNegative);
    priceInput.addEventListener('input', preventZeroAndNegative);
    discountInput.addEventListener('input', preventZeroAndNegative);
    // Clear stored values on page load
    localStorage.removeItem('quantity');
    localStorage.removeItem('price');
    localStorage.removeItem('discount');
  });
 </script>
 <script>
  document.addEventListener('DOMContentLoaded', function() {
    const productNameInput = document.getElementById('product-name');
    productNameInput.addEventListener('input', preventNegativeAndZero);
    function preventNegativeAndZero(event) {
      const input = event.target;
      const value = input.value.trim();
      if (value === '-' || value === '0') {
        input.value = ''; // Clear the input field if the value is '-' or '0'
      }
    }
  });
```

```
    </script>
    <script>
      document.addEventListener('DOMContentLoaded', function() {
        const descriptionInput = document.getElementById('description');
        descriptionInput.addEventListener('input', preventNumericInput);
        function preventNumericInput(event) {
          const input = event.target;
          const value = input.value.trim();
          if (/^\d+$/.test(value)) {
            input.value = ''; // Clear the input field if it contains only numeric characters
          }
        }
      });
    </script>
    <script>
      document.addEventListener('DOMContentLoaded', function() {
        // Retrieve all input fields
        const inputFields = document.querySelectorAll('input[type="text"], input[type="number"],
textarea');
        // Loop through each input field and clear its value
        inputFields.forEach(function(input) {
          input.value = ''; // Clear the value of the input field
        });
      });
    </script>
<script>
  document.addEventListener('DOMContentLoaded', function() {
    const form = document.getElementById('add-product-form'); // Replace 'add-product-form'
with your actual form ID
    form.addEventListener('submit', function(event) {
      const inputs = form.querySelectorAll('input[required], textarea[required], select[required]');
      let isEmpty = false;

      inputs.forEach(function(input) {
        if (!input.value.trim()) {
          isEmpty = true;
        }
      });

      if (isEmpty) {
        event.preventDefault(); // Prevent form submission if any required field is empty
        alert('Please fill in all required fields.');
        // You can also customize how you want to alert the user about the empty fields
      }
    });
  });
</script>
<script>
    document.addEventListener('DOMContentLoaded', function() {
      const form = document.getElementById('add-product-form'); // Replace 'add-product-form'
with your actual form ID
      form.addEventListener('submit', function(event) {
        const inputs = form.querySelectorAll('input[required], textarea[required], select[required]');
        let isEmpty = false;
        inputs.forEach(function(input) {
          if (input.type === 'number' && input.name === 'discount') {
            if (input.value.trim() === '') {
              isEmpty = true;
            }
            if (input.value.trim() !== '' && isNaN(parseFloat(input.value))) {
              isEmpty = true;
            }
          } else if (!input.value.trim()) {
            isEmpty = true;
          }
        });
        if (isEmpty) {
          event.preventDefault(); // Prevent form submission if any required field is empty or has an
```

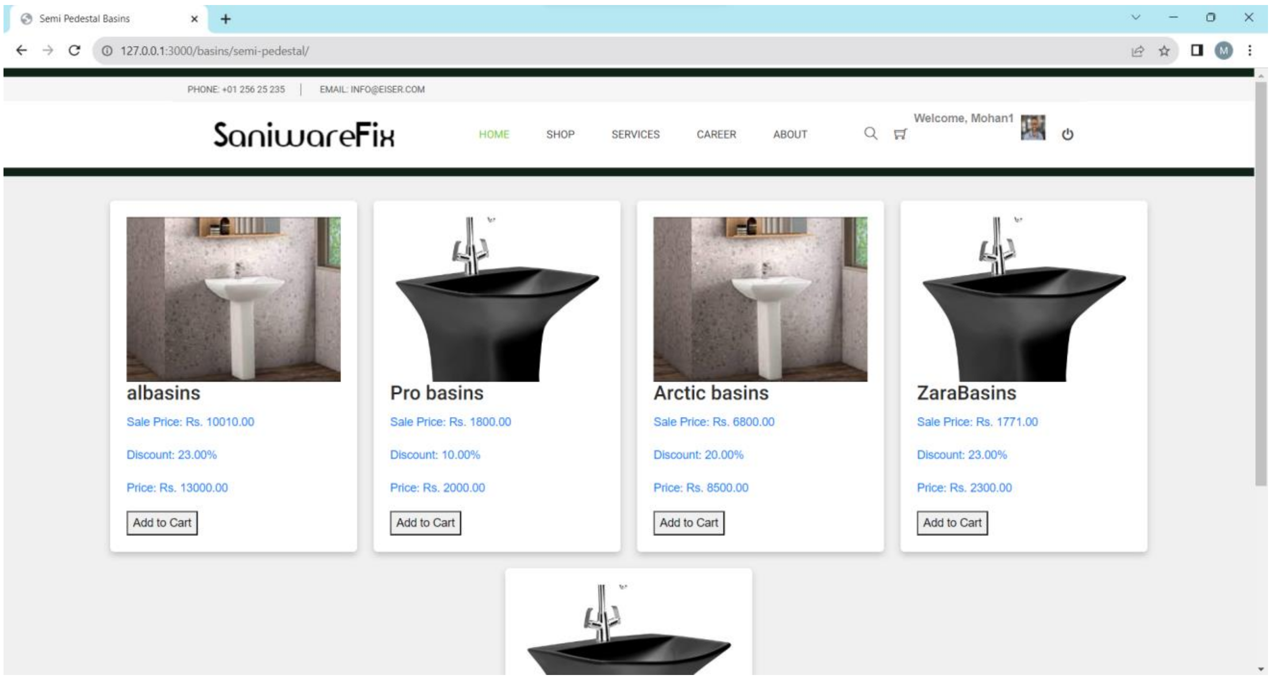
```
invalid discount value
          alert('Please fill in all required fields with valid values, including the discount.');
          // You can also customize how you want to alert the user about the empty fields or invalid
values
        }
      });
    });
  </script>
    </body>
</html>
```

9.2 Screen Shots

9.2.1 Userlogin page



9.2.2 Product View page



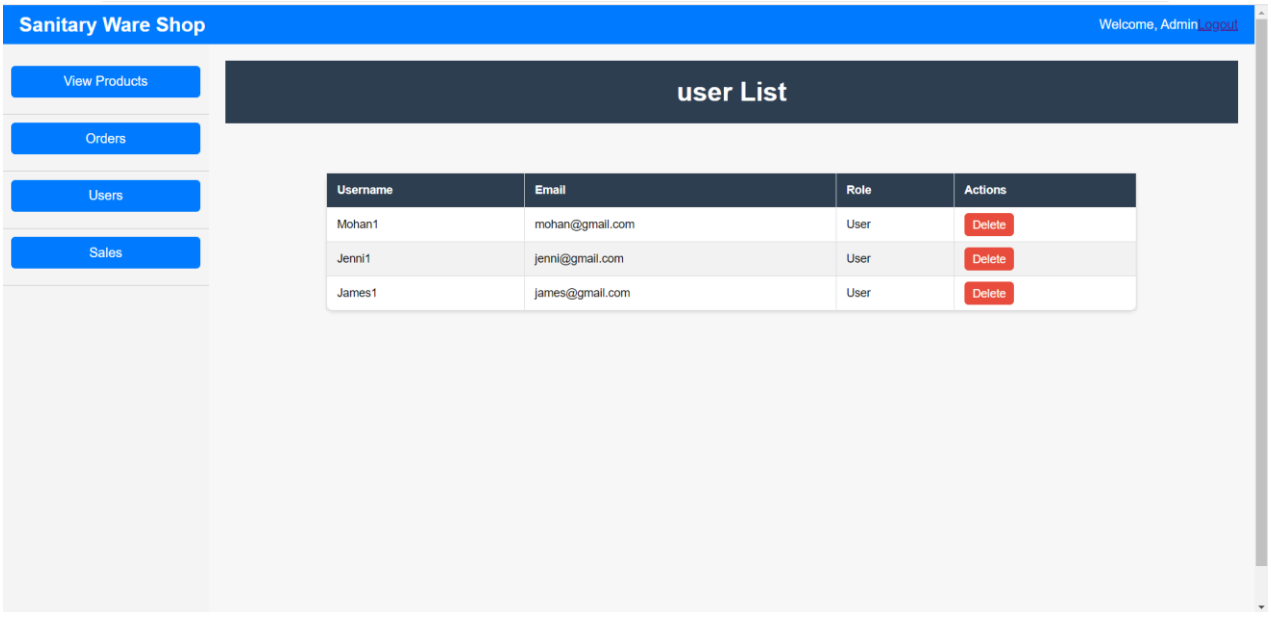
albasins
Sale Price: Rs. 10010.00
Discount: 23.00%
Price: Rs. 13000.00
Add to Cart

Pro basins
Sale Price: Rs. 1800.00
Discount: 10.00%
Price: Rs. 2000.00
Add to Cart

Arctic basins
Sale Price: Rs. 6800.00
Discount: 20.00%
Price: Rs. 8500.00
Add to Cart

ZaraBasins
Sale Price: Rs. 1771.00
Discount: 23.00%
Price: Rs. 2300.00
Add to Cart

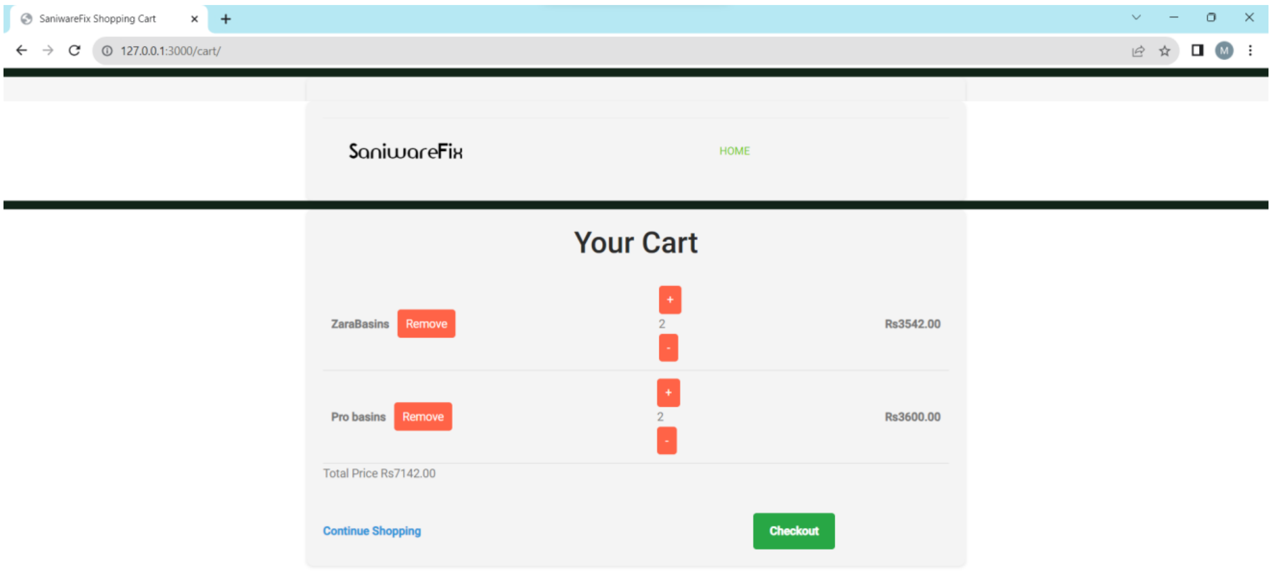
9.2.3 Admindashboard Page.



user List

Username	Email	Role	Actions
Mohan1	mohan@gmail.com	User	Delete
Jenni1	jenni@gmail.com	User	Delete
James1	james@gmail.com	User	Delete

9.2.4 Cart Page

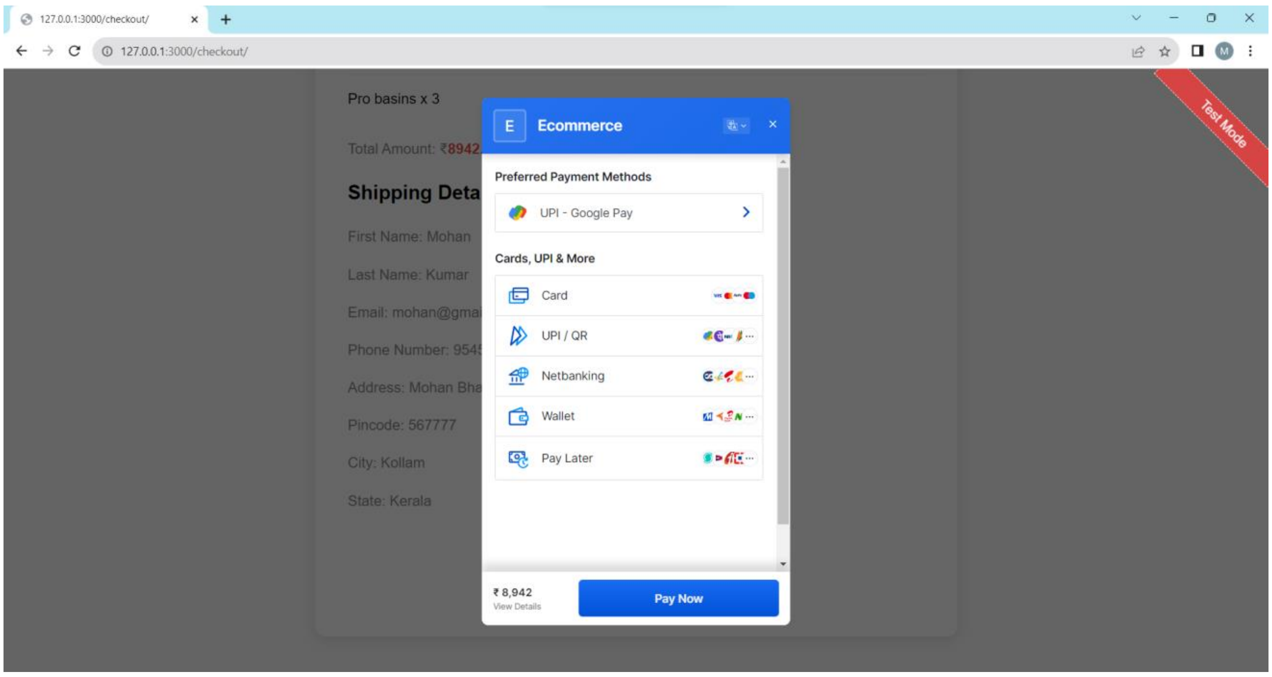


Your Cart

ZaraBasins	Remove	+ 2 -	Rs3542.00
Pro basins	Remove	+ 2 -	Rs3600.00

Total Price Rs7142.00

9.2.5 Payment page


127.0.0.1:3000/checkout/

Pro basins x 3

Total Amount: ₹8942

Shipping Deta

First Name: Mohan

Last Name: Kumar

Email: mohan@gmai

Phone Number: 954

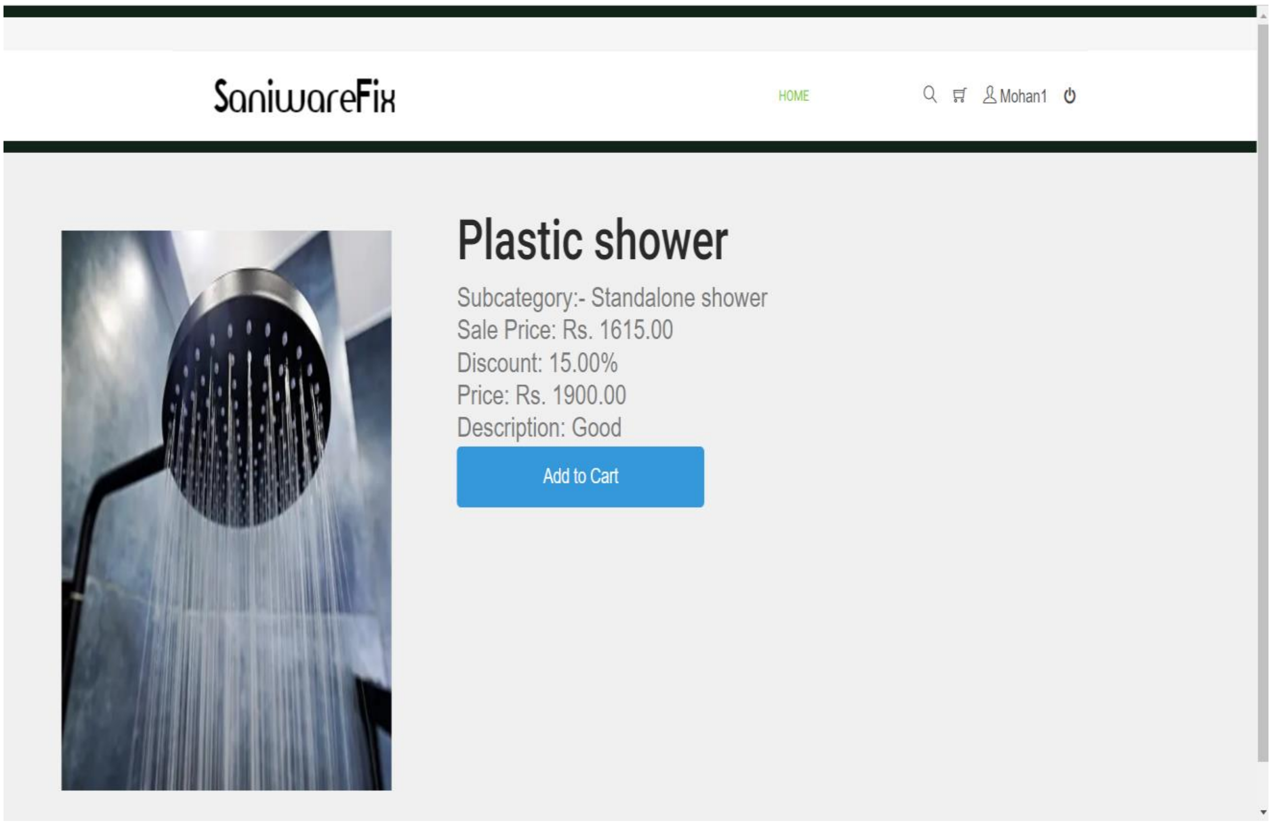
Address: Mohan Bha

Pincode: 567777

City: Kollam

State: Kerala

9.2.6



Plastic shower

Subcategory:- Standalone shower

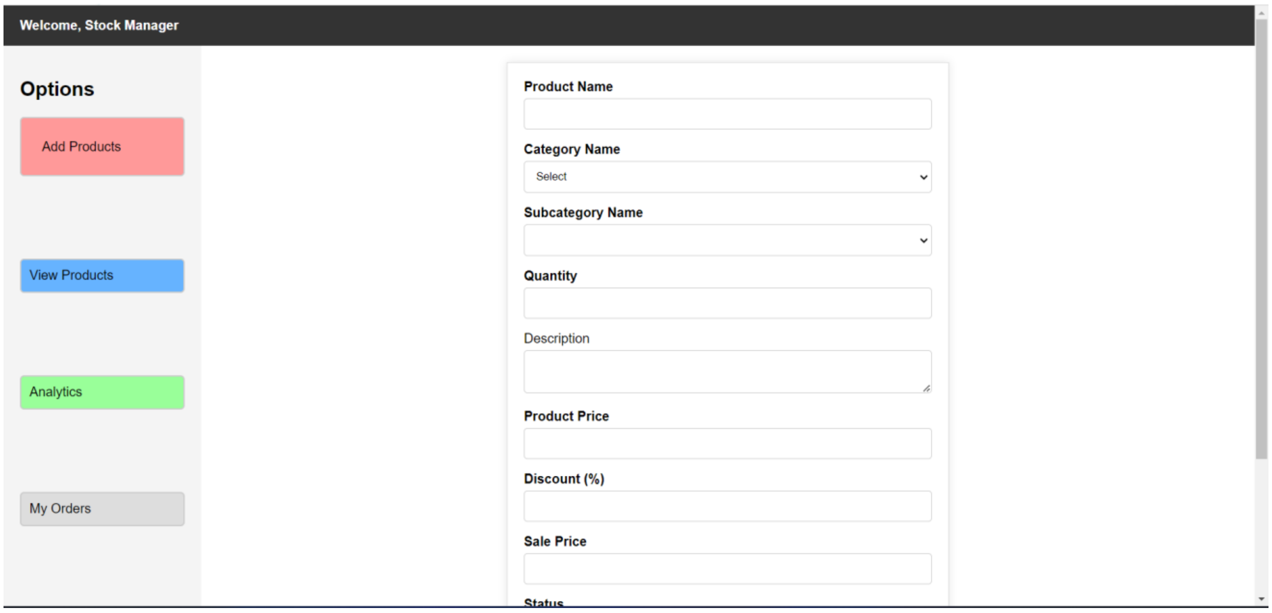
Sale Price: Rs. 1615.00

Discount: 15.00%

Price: Rs. 1900.00

Description: Good

9.2.7



Welcome, Stock Manager

Options

Add Products

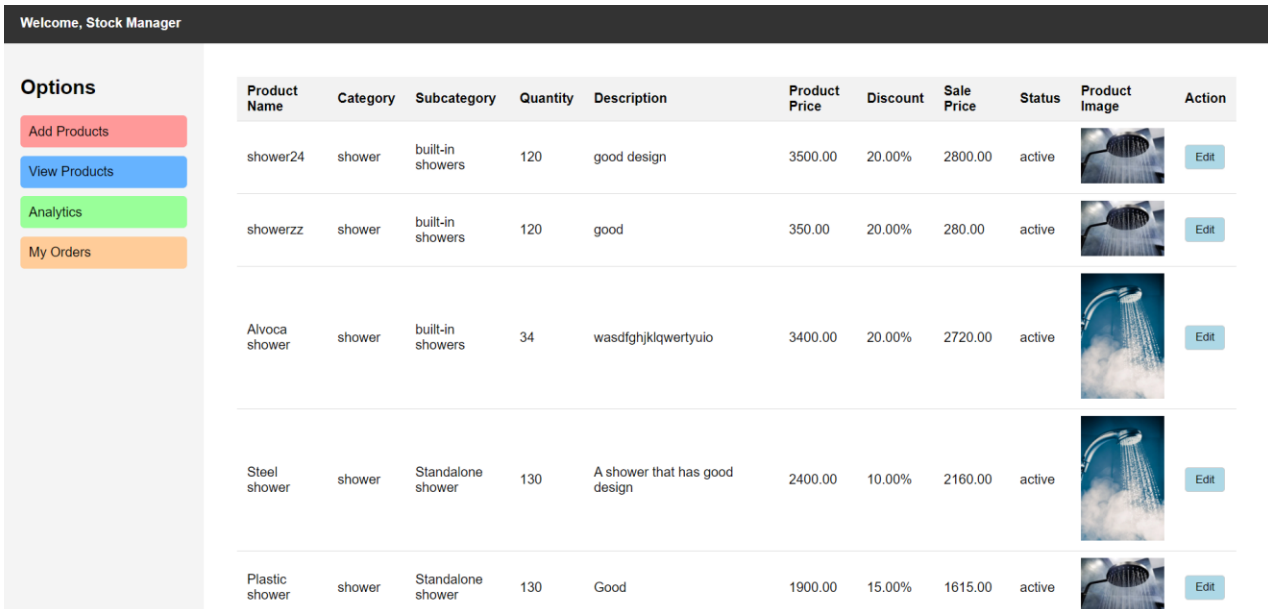
View Products

Analytics

My Orders

Product Name	
Category Name	Select
Subcategory Name	
Quantity	
Description	
Product Price	
Discount (%)	
Sale Price	
Status	

9.2.8



Welcome, Stock Manager

Options

Add Products

View Products

Analytics

My Orders

Product Name	Category	Subcategory	Quantity	Description	Product Price	Discount	Sale Price	Status	Product Image	Action
shower24	shower	built-in showers	120	good design	3500.00	20.00%	2800.00	active		Edit
showerzz	shower	built-in showers	120	good	350.00	20.00%	280.00	active		Edit
Alvoca shower	shower	built-in showers	34	wasdfghjklqwertyuio	3400.00	20.00%	2720.00	active		Edit
Steel shower	shower	Standalone shower	130	A shower that has good design	2400.00	10.00%	2160.00	active		Edit
Plastic shower	shower	Standalone shower	130	Good	1900.00	15.00%	1615.00	active		Edit