National Institute Of Electronics & Information Technology

रा.इ.सू.प्रौ.सं
NIELIT

Project Report

On

“Online Shopping Platform- Smartshop”

A Project

Submitted in partial fulfillment of the requirements for

The award of the

‘A’ Level

Under the Guidance of
Mr. Gagandeep Singh

Submitted by :
Dharmender
Reg. No. : 1061648
‘A’ Level

Annexure – II

**PERFORMA FOR A / B / C Level PROJECT CERTIFICATE
FROM PROJECT GUIDE /ACCREDITED INSTITUTE**

[For Direct as well as candidate from Accredited Institute]

This is to certify that the Project / Dissertation entitled, __________________ is a bonafide work done by Mr. / Ms. __________________________________ (NIELIT Registration No: __________________________________) in partial fulfillment of A Level / B Level / C Level examination and has been carried out under my direct supervision and guidance. This report or a similar report on the topic has not been submitted for any other examination and does not form a part of any other course undergone by the candidate.

____

Signature of Guide / Supervisor

	Name: ______________________
Place: _______________	**Designation**: ______________________
Date: ________________	**Address**: ______________________
	______________________
	______________________
	______________________

____

Signature of Center Manager
[**in case of a candidate from Accredited Institute**]

PERFORMA OF COVERING LETTER TO THE PROJECT REPORT

Controller of Examinations,
NIELIT,
Plot No. 3, PSP Pocket
Dwarka, Sector-8
New Delhi -110077

Sir,
I am submitting my ____________ Level Project for evaluation. Details of my Registration and postal address, etc is as under:

Regn. No: ________________________ **Level _______**

Name: ________________________

Father's Name: _______________________

Address:

(a) Residential Address: ___________________________________
____
____
____

Tele No: ____
(Country Code) (City Code) (Telephone number)

(b) Office Address: ____
____
____
____

Tele No: ____
(Country Code) (City-Code) (Telephone number)

Fax: ____
(Country-Code) (City-Code) (Telephone number)

E-mail Address (Please in block letters only): __________________________________

Contents

ACKNOWLEDGEMENT

The present work submitted by me would not be completed without the assistance of the National Institute of Electronics and Information Technology (NIELIT). I would like to acknowledge my supervisor Mr. Gagandeep Singh for his useful instructions for processing and generating useful results, and gave his valuable time throughout the period to complete this project.

SUBMITTED BY: -

Name : Dharmender

Reg. No.: 1061648

‘A’ Level

Signature :

DECLARATION

We hereby declare that the work is being presented by us in this project, entitled ***"*Online Shpping Platform - SmartShop*"*** in partial fulfillment of requirement for 'A' Level, and submitted at the "NIELIT Shimla" is an authentic piece of our own work carried out under the supervision of Mr. Gagandeep Singh.

SUBMITTED BY: -
Name : Dharmender
Reg. No.: 1061648
'A' Level
Signature :

GUIDE CERTIFICATE

This is to certify that this project entitled ***“Online Shpping Platform - SmartShop”*** submitted in fulfillment of the Diploma in A Level to the NIELIT Shimla , done by Dharmender (1061648) is an authentic work carried out by them at NIELIT Shimla Centre under my guidance. The matter embodied in this project work has not been submitted earlier for award of any degree or diploma to the best of my knowledge and belief.

Signature of the Students

Signature of the Guide

(Gagandeep Singh)

<u>CERTIFICATE</u>

This is to certify that the project entitled ***“Online Shpping Platform - SmartShop”*** carried out by Dharmender (1061648) for the partial fulfillment of the requirement for the ‘A’ Level is a bonafide record the work done by the candidates as certified by the candidate, under my guidance. To the best of my knowledge, this work has not been submitted for award of any other degree or diploma.

Project Guide

Director Incharge

Mr. Gangandeep Singh

5 Objective & Scope of the Project

The primary objective of "SmartShop: An Online Shopping Platform" is to design and develop a comprehensive e-commerce platform that provides a seamless and secure online shopping experience for customers. This platform allows customers to browse a wide range of products, add items to their carts, and place orders easily from any location. Businesses can take advantage of the platform to efficiently manage their inventory, orders, and customer data.

The platform is developed using Django, a high-level Python web framework known for its scalability, security, and simplicity. The backend utilizes SQLite as the default database, ensuring smooth and lightweight data handling during development and testing phases. The system is designed to accommodate various user roles, including administrators, customers, and suppliers, each with specific functionalities.

This section provides an overview of the system's objectives, features, and architecture, setting the foundation for subsequent sections that detail the development, implementation, and design strategies.

Key Objectives of the Platform:

1. **Enhanced User Experience:**
    - Provide customers with an intuitive and easy-to-navigate interface for browsing and purchasing products.
    - Incorporate advanced search and filtering options to allow users to find products quickly and efficiently.
2. **Vendor Empowerment:**
    - Develop tools for vendors to manage product listings, inventory, and orders effectively.
    - Offer sales insights and analytics to help vendors optimize their performance.
3. **Secure Transactions:**
    - Implement reliable and secure payment gateways to ensure customer trust.
    - Protect sensitive user data using advanced security measures like encryption and secure sessions.
4. **Scalable Architecture:**
    - Design a modular system that can accommodate increasing numbers of users, vendors, and products.
    - Ensure system stability and performance even under high traffic loads.
5. **Real-Time Updates:** Enable real-time updates for product availability, order status, and notifications without requiring page reloads.
6. **Multi-Device Accessibility:** Ensure full compatibility across various devices, including desktops, tablets, and smartphones, to enhance user reach.
7. **Cost-Effective Solution:** Provide an affordable e-commerce platform that is accessible to small and medium businesses, fostering their growth in the digital economy.

Scope of the Platform The platform is intended for small to medium-sized businesses looking to establish an online presence. It caters to:

1. **Customers**: Allowing them to browse and purchase products, track their orders, and manage their accounts.
2. **Administrators**: Providing tools to manage the product catalog, order processing, inventory, and user roles.
3. **Suppliers**: Enabling them to supply products and manage their partnerships with the platform.
4. **Scalability**: Supporting future features such as analytics dashboards, multi-language support, and integration with external APIs for payment gateways and delivery services.

Core Functionalities The SmartShop platform includes the following core functionalities:

1. **Product Management**: Administrators can add, edit, or delete products from the catalog. Products can be grouped into categories for better organization.
2. **Customer Management**: Administrators manage user registrations, profiles, and roles, ensuring secure access to the platform.
3. **Order Management**: Customers can place and track orders. Administrators oversee order fulfillment, including shipping and delivery updates.
4. **Shopping Cart Management**: Customers can add products to their shopping cart, update quantities, and proceed to checkout seamlessly.
5. **Supplier Management**: Suppliers can manage the products they provide and ensure timely restocking.
6. **Reports Generation**: The system generates various reports, such as sales summaries, customer activity, and product performance, to aid in business decision-making.

System Architecture The SmartShop platform is developed using the **Model-View-Controller (MVC)** design pattern, ensuring a clear separation of concerns and modularity:

1. **Model Layer**: Handles database operations and represents the platform's entities such as customers, products, and orders.
2. **View Layer**: Defines the user interface and interactions, presenting dynamic content based on user roles.
3. **Controller Layer**: Processes requests, manages business logic, and communicates between the model and view layers.

The platform's implementation also adheres to modern software engineering principles, ensuring flexibility, maintainability, and scalability.

Technological Stack

1. **Backend Framework**: Django (Python)
2. **Database**: SQLite (default database, can be upgraded to PostgreSQL or MySQL for production)
3. **Frontend Technologies**: HTML5, CSS3, JavaScript (with optional frameworks like Bootstrap for responsive design)
4. **Hosting**: Compatible with platforms like AWS, Heroku, and PythonAnywhere
5. **Version Control**: Git (repository hosting on GitHub)

This introduction highlights the purpose, scope, and architecture of the SmartShop platform, setting the stage for detailed discussions on system design, implementation, and functionality in subsequent sections.

6 Theoretical Background

The development of the SmartShop Online Shopping Platform is based on several theoretical concepts and established principles in the fields of e-commerce, web application development, and software engineering. This section offers a detailed exploration of the theoretical background, including the foundations of e-commerce, architectural patterns, and the technologies that support the platform.

6.1 E-Commerce Foundations

E-Commerce, or electronic commerce, is the buying and selling of goods and services over the Internet. It has revolutionized the way businesses operate, allowing global reach and enhanced customer engagement. The following principles form the theoretical basis for e-commerce systems like SmartShop:

1. **Digital Transactions**: Transactions occur digitally through secure payment gateways, ensuring seamless exchange of value.
2. **Product Catalog**: Online platforms use dynamic catalogs that allow administrators to manage and display products in real time.
3. **Customer-Centric Design**: Platforms are designed to prioritize user experience, incorporating search functionality, filters, and personalized recommendations.
4. **Scalability and Availability**: E-Commerce platforms must handle increasing traffic while ensuring uninterrupted service availability.

SmartShop utilizes these principles to create a user-friendly and scalable shopping experience for both customers and administrators.

6.2 Architectural Design Patterns

The platform uses the **Model-View-Controller (MVC)** design pattern, a widely adopted framework in the development of web applications.

1. **Model Layer**:

   The model layer represents the data structure of the system and encapsulates database operations. Entities such as **Product**, **Customer**, **Order**, and **Supplier** are defined in this layer, ensuring a clear representation of the business logic.

2. **View Layer**:

   The view layer focuses on the user interface (UI), providing dynamic web pages to customers and administrators. Technologies like **HTML**, **CSS**, and **JavaScript** are used to create responsive and interactive designs.

3. **Controller Layer**:

   The controller acts as an intermediary, processing user requests, managing business logic, and returning appropriate responses. It ensures a seamless interaction between the model and the view layers.

Using the MVC pattern, the system achieves modularity, making it easier to debug, maintain, and scale.

6.3 Database Design Concepts

The SmartShop platform follows the principles of relational database to ensure data integrity, consistency, and security. Key database concepts include:

1. **Normalization**:

   The database schema is normalized to reduce redundancy and improve query efficiency. For example, product details are stored in a separate table and are referenced by orders.

2. **Entity-Relationship (ER) Modeling**:

   ER diagrams are used to model relationships between entities such as **Customer**, **Product**, and **Order**, providing a conceptual blueprint of the structure of the database.

3. **Primary and Foreign Keys**:

   Primary keys uniquely identify each record, while foreign keys maintain relationships between tables. For example, the **Order** table references the **Customer** table via a foreign key.

4. **SQL Queries**:

   Structured Query Language (SQL) is used to interact with the database for CRUD (Create, Read, Update, Delete) operations.

6.4 Security Principles

The platform adheres to industry best practices to ensure security and data protection, including:

1. **Authentication and Authorization**:

   The system uses secure login mechanisms, ensuring only authorized users can access specific functionalities. Django's built-in authentication framework is leveraged to manage roles and permissions.

2. **Data Encryption**:

   Sensitive data, such as user passwords and payment information, is encrypted using secure hashing algorithms.

3. **Input Validation**:

   All user inputs are validated to prevent common vulnerabilities such as SQL injection and cross-site scripting (XSS).

4. **Secure Payment Integration**:

   Theoretical models for integrating secure payment gateways like **PayPal** or **Stripe** are considered, ensuring encrypted and seamless transaction processes.

6.5 Web Technologies

The SmartShop platform relies on several modern web technologies and frameworks, each serving specific purposes:

1. **Backend Framework – Django**:

   Django, a high-level Python framework, simplifies the development process by providing tools for database management, URL routing, and authentication. Its adherence to the **Don't Repeat Yourself (DRY)** principle ensures efficient code reuse.

2. **Frontend Technologies – HTML, CSS, JavaScript**:

   These technologies are used to create responsive and visually appealing web pages. CSS frameworks like **Bootstrap** enhance the platform's mobile compatibility.

3. **Database – SQLite**:

   SQLite is used during development to store light and efficient data. It can be migrated to a more robust database such as PostgreSQL for production environments.

4. **Version Control – Git**:

   Git is used to manage source code, enabling version control and collaborative development. GitHub serves as the hosting platform for the code repository.

6.6 Theoretical Models Supporting Online Shopping

1. **Consumer Behavior Models**:

   The platform is designed to meet the needs of customers, incorporating features that enhance decision-making, such as product comparisons, reviews, and order tracking.

2. **Payment and Delivery Models**:

   Integration with third-party APIs enables secure payment processing and real-time tracking of shipment and delivery status.

3. **Scalability Models**:

   The system is built with scalability in mind, using modular components and caching mechanisms to handle increased traffic as the user base grows.

7 Definition of Problem

In the modern digital world, online shopping platforms are transforming the retail sector by giving customers convenient access to a variety of products and services. However, despite the growing popularity of e-commerce, numerous small and medium-sized businesses find it difficult to create a strong online presence because they lack affordable, scalable, and user-friendly options. The SmartShop Online Shopping Platform aims to address these challenges by offering a comprehensive and efficient system that caters to both customer needs and administrative requirements.

7.1 Challenges Faced by Small and Medium-Sized Businesses (SMBs)

1. **Limited Technical Expertise**:

   Many SMBs lack the technical knowledge to set up and maintain an online platform. They require a solution that simplifies the process while ensuring scalability and flexibility.

2. **Cost Constraints**:

   Existing e-commerce platforms often have high setup and maintenance costs, making them inaccessible to small businesses.

3. **Inefficient Inventory Management**:

   Managing product categories, tracking stock levels, and ensuring timely restocking are significant challenges for businesses without a digital solution.

4. **Customer Engagement and Retention**:

   SMBs struggle to provide personalized experiences and seamless order tracking, resulting in lower customer satisfaction and retention rates.

5. **Integration with Payment and Delivery Services**:

   Businesses often face difficulties integrating secure payment gateways and logistics solutions, which are essential for a reliable online shopping experience.

7.2 Problems Faced by Customers

1. **Limited Accessibility**:

   Many existing platforms lack mobile-friendly interfaces, limiting accessibility for users who prefer shopping on mobile devices.

2. **Poor User Experience**:

   Complex navigation, slow loading times, and insufficient search and filter functionalities lead to a frustrating shopping experience.

3. **Lack of Transparency in Order Tracking**:

   Customers often face uncertainty regarding the status of their orders due to inadequate tracking systems.

4. **Payment Security Concerns**:

   Customers are hesitant to make online purchases due to fears of data breaches and insecure payment processes.

7.3 Gaps in Existing E-Commerce Platforms

While there are many established e-commerce platforms, they often fail to address the unique needs of small businesses and customers:

1. **Generic Solutions**:

   Most platforms offer generic solutions that lack customization options, making it difficult for businesses to tailor the platform to their specific needs.

2. **High Overheads**:

   Premium features often come at a high cost, making them inaccessible to smaller businesses with limited budgets.

3. **Inflexible Integration**:

   Many platforms have limited integration options for third-party services, such as custom payment gateways or regional logistics providers.

4. **Lack of Modular Architecture**:

   Non-modular platforms make it challenging to add or modify features as business requirements evolve.

5. **Scalability Issues**:

   Some platforms are not designed to handle high traffic, leading to performance issues as businesses grow.

7.4 Problem Statement

The primary problem addressed by the SmartShop Online Shopping Platform is the lack of an affordable, scalable, and user-friendly e-commerce solution that caters to the specific needs of small and medium-sized businesses while ensuring a seamless shopping experience for customers.

Specifically, the problem can be broken down into the following key aspects:

1. **Business Challenges**:

   - Lack of affordable and customizable e-commerce solutions for SMBs.
   - Inefficiencies in managing inventory, tracking orders, and handling suppliers.
   - Difficulty in providing secure and reliable payment options.

2. **Customer Challenges**:

   - Poor user experience due to limited search and filtering options.
   - Lack of transparency in order status and delivery tracking.

- Concerns about payment security and data privacy.

3. **Technical Challenges**:
   - Integration with third-party APIs for payments and logistics.
   - Ensuring scalability to handle increasing traffic and product catalogs.
   - Building a secure system that protects user data and prevents vulnerabilities.

7.5 Objectives of the SmartShop Platform

To address these challenges, the **SmartShop Online Shopping Platform** is designed with the following objectives:

1. **For Businesses**:
   - Provide an affordable and scalable e-commerce solution that is easy to set up and manage.
   - Enable efficient management of product catalogs, orders, inventory, and suppliers.
   - Offer customizable features to accommodate unique business requirements.
2. **For Customers**:
   - Ensure a seamless and user-friendly shopping experience with intuitive navigation and search capabilities.
   - Provide real-time order tracking and transparent delivery updates.
   - Implement secure payment gateways to build trust and confidence in the platform.
3. **Technical Objectives**:
   - Develop a modular architecture using the Django framework to ensure flexibility and scalability.
   - Ensure data security through encryption, secure authentication, and robust error handling.
   - Integrate third-party APIs for payment processing and logistics.

8 System analysis & design vis-a-vis user requirements

The system analysis and design phase is essential for the effective implementation of the SmartShop Online Shopping Platform. It focuses on understanding user requirements, analyzing the scope of the problem, and creating a system that meets the requirements of both customers and administrators. This section explains how user requirements are converted into technical solutions and how the platform's design aligns with these needs.

8.0.1 Understanding User Requirements

The SmartShop platform is built to cater to two primary user groups:

1. **Customers**

   Customers interact with the platform to browse products, place orders, and track deliveries. Their requirements include:

- **Easy Navigation:** A user-friendly interface with intuitive navigation and responsive design.
- **Search and Filtering Options:** Advanced search and filtering mechanisms to quickly locate desired products.
- **Secure Payment Options:** Integration of trusted payment gateways for seamless and safe transactions.
- **Order Tracking:** Real-time updates on order status and delivery progress.
- **Customer Support:** Quick and efficient resolution of queries or issues.

2. **Administrators (Business Owners)**

   Administrators manage the backend of the platform, including product catalogs, orders, inventory, and suppliers. Their requirements include:

   - **Inventory Management:** Tools to track stock levels, restock alerts, and supplier management.
   - **Order Management:** A dashboard to monitor and process customer orders efficiently.
   - **Analytics and Reporting:** Insights into sales performance, customer behavior, and inventory trends.
   - **Customizability:** The ability to add or modify features based on business needs.
   - **Scalability:** A platform that grows with the business and handles increased traffic seamlessly.

8.0.2 System Analysis

The system analysis phase identifies how the platform meets these user requirements through a structured approach:

Functional Requirements

1. **Product Management:**
   - Ability to add, update, and delete products within specific categories.
   - Upload product images, descriptions, and pricing information.
2. **Customer Management:**
   - User registration, login, and profile management.
   - Store and manage customer order history.
3. **Order and Delivery Management:**
   - Track orders from placement to delivery.
   - Provide customers with order updates via email or notifications.
4. **Payment Integration:** Support for multiple payment gateways such as PayPal, Stripe, or credit cards.
5. **Reporting and Analytics:**
   - Generate sales reports and product performance analytics.
   - Monitor website traffic and user activity.

Non-Functional Requirements

1. **Performance:** The system must handle up to 10,000 concurrent users without performance degradation.
2. **Security:**
   - Implementation of encryption for sensitive data like payment details.
   - Protection against common vulnerabilities like SQL injection and cross-site scripting (XSS).
3. **Scalability:** The system must scale horizontally to accommodate business growth.
4. **Availability:** Ensure 99.9% uptime to prevent business interruptions.

8.0.3 System Design

The system design phase involves creating a blueprint for how the platform's components will interact and function.

Architectural Design The SmartShop platform follows a modular, three-tier architecture:

1. **Presentation Layer (Frontend)**
   - The user interface is built using responsive web design techniques, ensuring compatibility with both desktop and mobile devices.
   - Frontend frameworks such as Bootstrap and JavaScript are used to enhance interactivity.
2. **Application Layer (Backend)**
   - The backend is developed using the Django framework, which ensures a modular and scalable architecture.
   - APIs are used to connect the frontend with backend services, enabling seamless data exchange.
3. **Data Layer (Database)**
   - A relational database, such as PostgreSQL, is used to store structured data, including product details, customer information, and order history.
   - The database schema is designed to handle relationships between entities such as products, customers, and orders efficiently.

Entity-Relationship (ER) Diagram The system's ER diagram ensures that all entities and their relationships are clearly defined:

- **Entities:**
  - `Admin`: Manages the platform and handles products, orders, and inventory.
  - `Product`: Represents items available for purchase, categorized for easy browsing.
  - `Customer`: Registers on the platform to place orders and track purchases.

  - `Order`: Contains details of products purchased, customer information, and order tracking.
  - `Supplier`: Supplies products to the admin for inventory.

- **Relationships:**
  - `Admin` manages `Products`.
  - `Customer` places an `Order` which contains `Products`.
  - `Order` has `Tracking Details`.
  - `Supplier` provides `Products` to the platform.

Data Flow Diagram (DFD) The DFD captures the flow of data within the system:

1. **Level 0:** Depicts the interaction between external users (customers and admins) and the system.
2. **Level 1:** Breaks down the processes such as product management, order management, and payment processing.

User Interface Design The user interface is designed with a focus on:

- **Intuitive Navigation:** Clear menus and logical page flows.
- **Aesthetic Appeal:** Use of modern design principles to enhance visual appeal.
- **Responsiveness:** Compatibility across various devices, including smartphones and tablets.

8.0.4 Addressing User Requirements

The system design ensures that all user requirements are met effectively:

1. **For Customers:**
   - **Requirement:** Easy navigation and advanced search options. **Solution:** A frontend with a responsive design and robust search filters.
   - **Requirement:** Secure payment options. **Solution:** Integration with trusted payment gateways and encrypted transactions.
2. **For Administrators:**
   - **Requirement:** Efficient inventory and order management. **Solution:** A backend dashboard for real-time stock tracking and order processing.
   - **Requirement:** Customizability. **Solution:** Modular design allows for adding or modifying features without disrupting the system.

9 System Planning (PERT Chart)

System planning plays a vital role in the development of the SmartShop Online Shopping Platform. It ensures that tasks are executed in a timely and organized manner to achieve the project's goals. A PERT (Program Evaluation and Review Technique) chart is used to visualize and organize the project's activities, their dependencies, and estimated completion times. This method helps in identifying critical tasks and planning the overall timeline effectively.

9.1 PERT Chart Overview

A PERT chart is a project management tool used to map out tasks, identify dependencies, and estimate timelines. It enables:

1. **Clear Visualization of Tasks:** Breaking the project into manageable components.
2. **Dependency Mapping:** Understanding the order in which tasks need to be completed.
3. **Critical Path Identification:** Determining the longest sequence of dependent tasks, which dictates the minimum project duration.
4. **Time Estimation:** Using optimistic, pessimistic, and most likely time estimates for better planning.

9.2 Key Activities for SmartShop Development

The activities required to develop the SmartShop platform are divided into distinct phases:

Phase 1: Requirement Analysis

- **Task 1:** Gather user and business requirements.
- **Task 2:** Conduct feasibility analysis.
- **Task 3:** Create a detailed system requirement specification (SRS) document.

Phase 2: System Design

- **Task 4:** Design database schema (ER diagram and relational model).
- **Task 5:** Design user interface prototypes for the frontend.
- **Task 6:** Develop system architecture (three-tier architecture design).

Phase 3: Development

- **Task 7:** Develop the frontend (HTML, CSS, JavaScript, Bootstrap).
- **Task 8:** Implement backend functionality using Django.
- **Task 9:** Set up the database and integrate with the backend.
- **Task 10:** Develop APIs for communication between frontend and backend.

Phase 4: Testing

- **Task 11:** Conduct unit testing of individual components.
- **Task 12:** Perform integration testing to ensure compatibility between components.
- **Task 13:** Conduct user acceptance testing (UAT) with feedback from stakeholders.

Table 1: Project task breakdown and dependencies

Task ID	Task Name	Dependency	Time Estimate (Days)
1	Gather user and business requirements	None	5
2	Conduct feasibility analysis	1	3
3	Create system requirement specification	2	4
4	Design database schema	3	6
5	Design UI prototypes	3	5
6	Develop system architecture	4, 5	5
7	Develop the frontend	6	8
8	Implement backend functionality	6	10
9	Set up the database and integrate backend	4, 8	7
10	Develop APIs	8	6
11	Conduct unit testing	7, 8, 9, 10	5
12	Perform integration testing	11	5
13	Conduct user acceptance testing	12	4
14	Deploy the system on a live server	13	3
15	Provide training and documentation	13	5
16	Plan maintenance and scalability	14	3

Phase 5: Deployment and Maintenance

- **Task 14:** Deploy the system on a live server.
- **Task 15:** Provide training and documentation for admins.
- **Task 16:** Plan regular maintenance and future scalability.

9.3 PERT Chart for SmartShop Development

The following PERT chart outlines the tasks, dependencies, and estimated durations for the SmartShop platform:

Tasks and Dependencies

Critical Path Identification The critical path represents the longest sequence of dependent tasks that determines the project's overall timeline.

- **Critical Path:**
  $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 8 \rightarrow 9 \rightarrow 11 \rightarrow 12 \rightarrow 13 \rightarrow 14 \rightarrow 16$
- **Critical Path Duration:** 51 days

9.4 PERT Chart Visualization

The PERT chart below visualizes the tasks and dependencies (an actual diagram can be drawn using tools like Lucidchart, Microsoft Project, or any PERT chart software):

1. Start → Task 1 → Task 2 → Task 3 → Task 4
2. Task 4 splits into Task 6 and Task 8, with Task 6 also leading to frontend design (Task 7).

3. Task 9 (Database Integration) depends on Tasks 4 and 8.
4. Testing (Tasks 11, 12, and 13) follows development and integration (Tasks 7, 8, 9, and 10).
5. Deployment and post-deployment tasks (Tasks 14, 15, and 16) conclude the project.

9.5 Time Estimation and Buffer Allocation

In PERT, each task's duration is estimated using three values:

1. **Optimistic Time (O):** The shortest possible time to complete a task.
2. **Pessimistic Time (P):** The longest possible time to complete a task.
3. **Most Likely Time (M):** The most realistic estimate.

The formula for the expected time is:
TE = $\frac{O+4M+P}{6}$
For instance, if Task 1 has O= 4, M=5M, and P=6:

TE=$\frac{4+4(5)+6}{6}$ = 5 days
Buffer time is allocated for critical tasks to account for unforeseen delays.

10 Methodology Adopted, System Implementation & Details of Hardware and Software Used

10.1 Methodology Adopted

To develop the SmartShop online shopping platform, the **Agile Software Development Methodology** was adopted. Agile is an iterative and incremental approach that emphasizes flexibility, collaboration, and continuous delivery of functional components.

Key Features of the Adopted Methodology:

1. **Iterative Development:** Dividing the project into multiple sprints, each delivering a working module.
2. **Stakeholder Involvement:** Regular feedback from stakeholders, including customers and admin users.
3. **Continuous Testing and Integration:** Ensuring system reliability by testing each module and integrating it into the main system seamlessly.
4. **Adaptability:** Adjusting the project plan based on feedback or changing requirements during the development cycle.

Workflow in Agile:

1. **Planning Phase:**
   - Requirements gathering and prioritization.
   - Creation of user stories for each feature.
2. **Design Phase:**

- Designing system architecture, database, and user interfaces.
- Using mockups and wireframes to visualize the platform's functionality.

3. **Development Phase:**
   - Implementing backend and frontend features in sprints.
   - Conducting code reviews to ensure code quality.
4. **Testing Phase:**
   - Unit testing for individual components.
   - Integration testing to check interaction between modules.
   - User Acceptance Testing (UAT) for final approval.
5. **Deployment and Maintenance:**
   - Deploying the system on a live server.
   - Addressing bugs and planning future updates based on user feedback.

5.6.2 System Implementation The implementation of SmartShop followed a structured approach, ensuring that each module is designed and developed independently and then integrated to form a cohesive system.

Major Components of System Implementation:

1. **Frontend Implementation:**
   - Developed a user-friendly interface for customers and administrators using HTML, CSS, Bootstrap, and JavaScript.
   - Implemented responsive design to ensure compatibility across devices.
2. **Backend Implementation:**
   - Used **Django**, a Python-based framework, to develop the backend.
   - Created models for managing products, orders, customers, and tracking details.
   - Developed RESTful APIs for communication between the frontend and backend.
3. **Database Integration:**
   - Designed a relational database schema using **PostgreSQL**.
   - Ensured efficient data storage and retrieval for products, customers, and orders.
4. **Authentication and Authorization:**
   - Implemented user authentication using Django's built-in authentication system.
   - Created separate roles for customers and administrators with appropriate permissions.
5. **Payment Gateway Integration:** Integrated a payment gateway (e.g., PayPal or Stripe) to process online transactions securely.
6. **Order Management and Tracking:** Enabled customers to view order history and track shipment status in real time.
7. **Testing and Debugging:**
   - Conducted thorough testing to identify and fix bugs.
   - Used tools like Selenium for automated testing and debugging.

Table 2: **Hardware Requirements for Development, Testing, and Deployment Environments**

Hardware Component	**Specifications**
Development Machines	Laptops/PCs with at least:
	- Processor: Intel i5 or higher
	- RAM: 8 GB or higher
	- Storage: 256 GB SSD
	- Operating System: Windows 10, macOS, or Linux
Testing Environment	Virtual Machines or dedicated testing machines
Deployment Server	Cloud-based server with the following:
	- Processor: Quad-core (or equivalent)
	- RAM: 16 GB
	- Storage: 512 GB SSD or more
	- Bandwidth: High-speed internet (100 Mbps or higher)

Table 3: **Software Tools and Their Purposes**

Software Tool	**Purpose**
Django	Backend development framework
PostgreSQL	Relational database management system
HTML, CSS, JavaScript	Frontend design and development
Bootstrap	Frontend framework for responsive design
Python	Programming language for backend development
Visual Studio Code	Code editor for development
Git	Version control for managing codebase
Docker	Containerization for deployment and testing
Selenium	Automated testing tool
Postman	API testing and debugging
Cloud Hosting Platform	Deployment environment (e.g., AWS, Google Cloud, or Heroku)

5.6.3 Details of Hardware Used The development and deployment of SmartShop required the following hardware resources:

5.6.4 Details of Software Used The following software tools were used during the SmartShop development lifecycle:

5.6.5 Integration of Hardware and Software

1. **Development Phase:**
   - Software was developed locally on developer machines.
   - Git was used for version control, ensuring collaboration among team members.
   - Docker containers were used to simulate the production environment during testing.
2. **Testing Phase:**
   - A dedicated virtual machine was used to test the system on different operating systems and browsers.

- Automated testing scripts were executed using Selenium to ensure reliability.

3. **Deployment Phase:**
    - The system was deployed on a cloud-based server with high scalability to handle user traffic.
    - Continuous Integration and Continuous Deployment (CI/CD) pipelines were set up for automatic updates and maintenance.

5.6.6 Challenges Encountered

1. **Integration Issues:** Initial difficulties in ensuring seamless communication between the frontend and backend were resolved using Django's REST framework.
2. **Database Optimization:** Indexing and query optimization techniques were applied to enhance database performance.
3. **Cross-Browser Compatibility:** Ensuring the platform worked uniformly across different browsers required extensive testing.

11 System Maintenance & Evaluation

11.1 System Maintenance

The maintenance of the SmartShop online shopping platform is an essential part of its lifecycle, ensuring that the system remains functional, secure, and up-to-date. Maintenance activities are organized and carried out to address issues that arise after deployment, enhance performance, and modify the system to meet changing user needs or technological progress.

Types of Maintenance Performed:

1. **Corrective Maintenance:**
    - Focused on fixing any bugs or errors identified after deployment.
    - Issues such as broken links, payment gateway glitches, or incorrect database queries were resolved promptly.
    - Continuous monitoring and feedback collection allowed for the identification of these problems.
2. **Adaptive Maintenance:** Adjustments made to keep the system compatible with external changes, such as:
    - Updates to the Django framework.
    - New versions of PostgreSQL or changes in APIs for third-party integrations (e.g., payment gateways).
    - Changes in web browsers or mobile operating systems.
3. **Perfective Maintenance:** Enhancements were implemented to improve the system's performance, usability, or functionality, such as:
    - Optimizing page loading speeds.

    - Enhancing the user interface to make navigation more intuitive.
    - Adding new features like product recommendations or discount codes based on user feedback.

4. **Preventive Maintenance:**
    - Proactive measures were taken to reduce the risk of future issues.
    - Regular server maintenance, database backups, and security audits were conducted.
    - Code refactoring was performed to improve maintainability and scalability.

Key Maintenance Tasks:

- Monitoring server uptime and response times using tools like New Relic or Google Cloud Monitoring.
- Performing regular security updates to protect against vulnerabilities.
- Backing up data daily to ensure minimal data loss in case of hardware failures.
- Updating the documentation to reflect any changes made to the system.

11.2 System Evaluation

System evaluation is an ongoing process used to assess whether the SmartShop platform meets its functional, technical, and performance requirements. Evaluation ensures that the system continues to serve its users effectively and delivers value to the business.

Evaluation Criteria:

1. **Performance Metrics:**
    - System response time, including page load speed and API response time.
    - Server uptime percentage (target: 99.9%).
    - Maximum concurrent user load without degradation in performance.

2. **Functionality Assessment:**
    - All system modules (product listing, order management, payment gateway, etc.) are tested periodically to ensure they operate as expected.
    - Verification of new features added during maintenance to check for alignment with user requirements.

3. **User Feedback:**
    - Regular surveys and feedback collection from customers and administrators to evaluate user satisfaction.
    - Support ticket analysis to identify recurring issues or areas needing improvement.

4. **Security Evaluation:**
    - Conducting penetration testing to identify and address vulnerabilities.
    - Monitoring for unauthorized access or unusual activities using tools like fail2ban or security monitoring platforms.

5. **Cost-Effectiveness:**
    - Evaluating the operational costs, including server hosting and maintenance personnel, against the revenue generated by the platform.
    - Assessing the cost-benefit ratio of implementing new features or upgrades.

Evaluation Techniques:

- **Log Analysis:**
    - Logs generated by the server and database are analyzed for errors or performance bottlenecks.
    - Tools like ELK Stack (Elasticsearch, Logstash, and Kibana) are used for visualization and pattern identification.
- **Key Performance Indicators (KPIs):**
    - Metrics such as conversion rate, average session duration, and bounce rate are tracked using Google Analytics.
    - Customer retention rates and the number of repeat purchases are monitored to gauge user engagement.
- **Testing Frameworks:** Automated testing tools like Selenium are used to periodically evaluate the functionality of critical system workflows, such as the checkout process or order tracking system.

Challenges in Maintenance and Evaluation

1. **Handling Evolving Requirements:** Users may request new features or changes that were not originally planned, requiring additional development and testing efforts.
2. **Scalability Management:** As user traffic grows, ensuring the system can handle increased loads without performance degradation becomes challenging.
3. **Security Threats:** New security vulnerabilities are discovered regularly, requiring continuous monitoring and updates to keep the system secure.
4. **Integration Issues:** Updates to third-party APIs (e.g., payment gateways) can occasionally cause disruptions that require immediate resolution.

Continuous Improvement System maintenance and evaluation are not one-time activities but rather ongoing processes that evolve with the system. Based on evaluation results, the following actions are taken:

- Prioritizing and implementing user-requested enhancements.
- Investing in better hosting infrastructure if performance metrics indicate a need for improved scalability.
- Training the team on emerging technologies and best practices to keep the system modern and competitive.

Future Considerations for Improvement:

- Implementing AI-based product recommendations to enhance user experience.
- Introducing multilingual support to cater to a global audience.
- Transitioning to a microservices architecture for better modularity and scalability.

12 Detailed Life Cycle of the Project

The SmartShop: An Online Shopping Platform was developed using a systematic life cycle model to make sure it functions well, operates efficiently, and is easy for users. The project life cycle consisted of the following phases:

12.1 Entity-Relationship Diagram (ERD):

The ERD describes the relationships among entities in the SmartShop platform.

- **Entities:**
  - Admin manages the **Online Shopping Website**.
  - The website contains **Product Categories** and **Products**.
  - Customers make **Orders**, which contain **Products**.
  - Each order has associated **Tracking Details**.
  - **Suppliers** provide products.
- **Key Relationships:**
  - Admin manages Online Shopping Website.
  - Customer makes Order contains Product.
  - Product belongs to Product Category.
  - Supplier supplies Product.

12.2 Data Flow Diagram (DFD):

1. **Level 0 - Context Diagram**

   This provides an overview of the system and its external entities.

   External Entities:

   - Customers: Place orders and track them.
   - Admins: Manage product categories and orders.
   - Suppliers: Add products to the system.

2. **Level 1 - High-Level Process View**

   This breaks down the core processes of the system.

   Processes:

   - Product Management: Admins manage product categories and details.

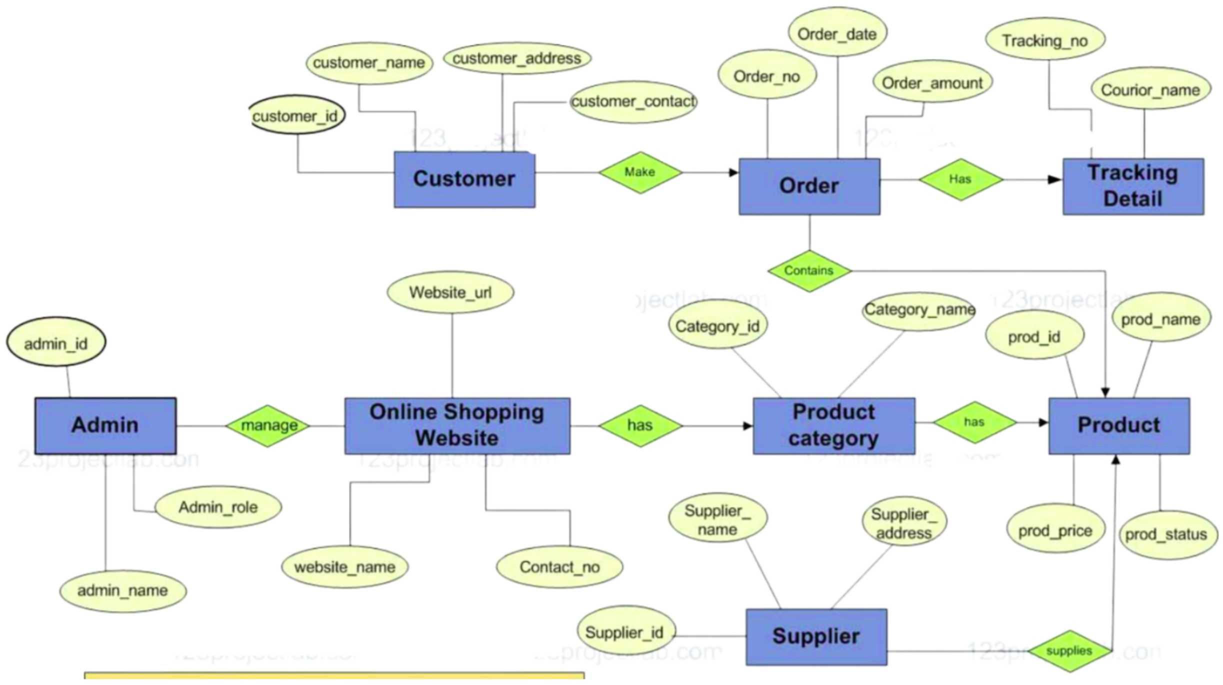


Figure 1: ER diagram for Smartshop online shopping platform.

- Order Management: Customers place orders, and the system updates tracking information.
- Supplier Management: Suppliers provide products and manage inventory.

3. **Level 2 - Detailed Process View**

   This elaborates on the internal workings of each major process.

   Processes:

   - Order Placement: Customers select products, verify payment, and confirm orders.
   - Inventory Update: The system updates stock levels based on supplier input.
   - Tracking Information: Order status is updated in real time and shared with customers.

12.3 Implementation Phase

The coding and development of the system were completed during this phase.

- **Frontend Development:**
  - Frameworks: HTML, CSS, JavaScript.
  - Features: Intuitive navigation and a responsive interface.
- **Backend Development:**
  - Framework: Django.

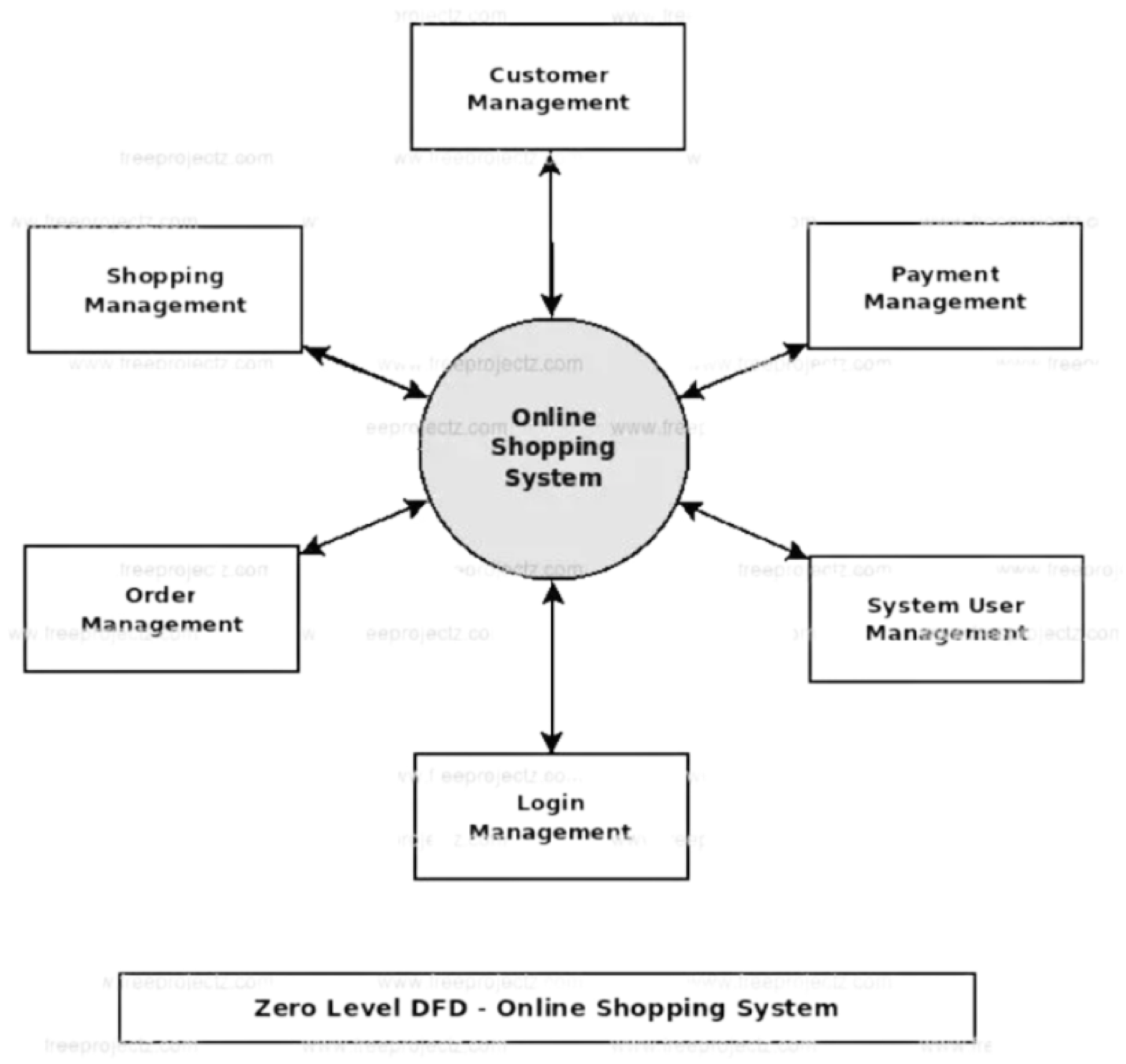


Figure 2: Level-0 DFD diagram for Smartshop online shopping platform.

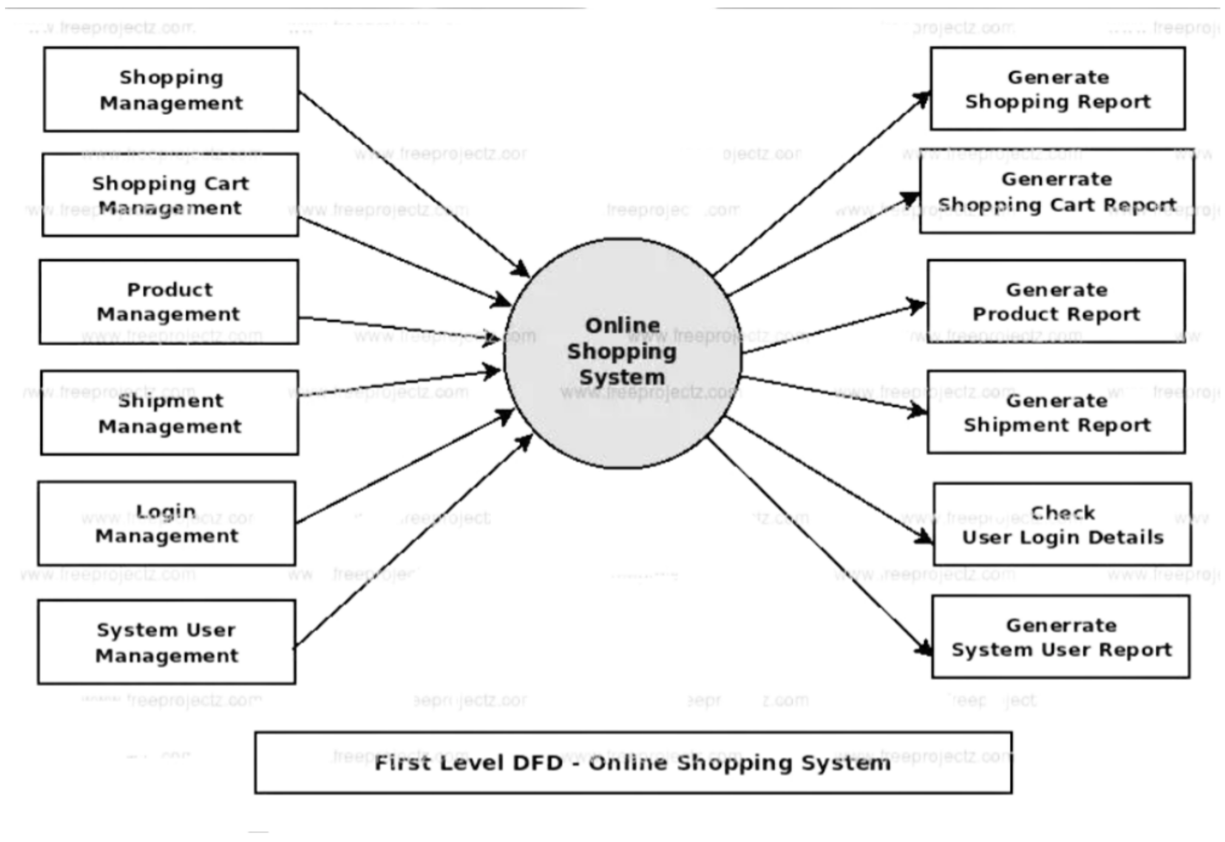


Figure 3: Level-1 DFD for Smartshop online shopping platform.

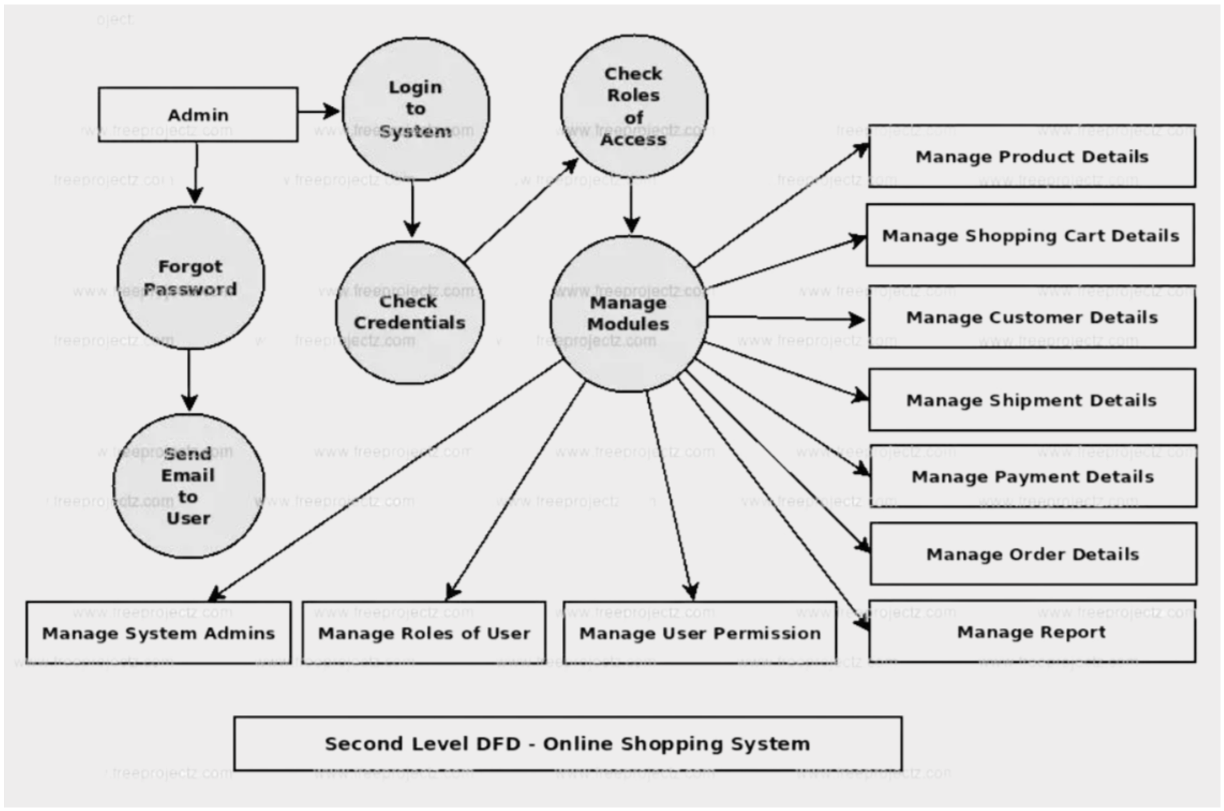


Figure 4: Level-2 DFD diagram for Smartshop online shopping platform.

    - Features: Secure authentication, order processing, and data management.

- **Database Development:**
    - Database: PostgreSQL.
    - Tables include entities such as Admin, Customer, Product, Order, Supplier, etc.

- **Third-Party Integration:**
    - Payment gateways.
    - Notification APIs for email confirmations and updates.

12.4 Testing Phase

Testing ensured that the system met the functional and non-functional requirements.

- **Testing Types Performed:**
    - **Unit Testing:** Verifying individual modules, such as login functionality.
    - **Integration Testing:** Checking seamless interactions between modules like order placement and payment confirmation.
    - **Load Testing:** Evaluating system performance under high user load.
    - **Security Testing:** Ensuring resistance to vulnerabilities like SQL injection.

- **Tools Used:**
    - Selenium for automated testing.
    - JIRA for bug tracking and resolution.

12.5 Deployment Phase

The system was deployed on AWS for scalability and reliability.
Steps:

- Hosting on AWS EC2 instances.
- Configuring a secure server with SSL encryption.
- Setting up a load balancer for high availability.

12.6 Maintenance Phase

Post-deployment, the system undergoes regular maintenance to ensure uninterrupted service.
Key Activities:

- Monitoring system performance and resolving issues promptly.
- Updating security protocols to protect user data.
- Implementing feature requests based on user feedback.

12.7 Input and Output Screen Design

Input Screens:

1. **Login/Register Page:**
   - Inputs: Email, password, role selection (Customer/Admin/Supplier).
   - Validation: Strong password, email format validation.
2. **Add Product Page (Admin):**
   -
   - Inputs: Product name, category, price, stock level.
   - Features: Real-time stock update validation.
3. **Shopping Cart and Checkout Page:** Users can review selected items, place order and confirm payment.

Output Screens:

1. **Product Listing Page:** Outputs: Product name, price, stock availability, and filters for category.
2. **Order Confirmation Page:** Outputs: Order details, estimated delivery date, payment success/failure message.
3. **Order Tracking Page:** Outputs: Real-time status updates (shipped, in transit, delivered).

4. Processes Involved

1. **Customer Journey:** Register/Login → Browse products → Add to cart → Checkout → Track orders.
2. **Admin Workflow:** Login → Add/Update/Delete products → Monitor orders → Manage inventory.
3. **Supplier Operations:** Login → Update stock levels → Notify admin of replenishments.

5. Methodology Used for Testing Testing Methodology:

1. **Unit Testing:** Tested individual components (e.g., user authentication, adding items to the cart).
2. **Integration Testing:** Verified seamless interaction between components (e.g., checkout and inventory updates).
3. **System Testing:** Ensured the system worked as a whole without errors.
4. **User Acceptance Testing (UAT):** Collected feedback from users and refined the platform accordingly.

6. Test Report and Documentation Key Features Tested:

- User authentication (successful login/logout, password reset).
- Product filtering and browsing (search by category, availability).
- Order placement and payment verification.
- Order tracking (real-time updates on order status).

Test Results:
Code Documentation:

- Frontend design (HTML/CSS/JavaScript) for responsiveness.
- Clean, modular, and well-commented code.
- Backend implementation (Django framework) with efficient database queries.

7. User/Operational Manual Features Overview:

- **Admin Panel:**
  - Add/update/delete products.
  - View/manage orders.
  - Monitor inventory levels.
- **Customer Features:**
  - Browse and filter products.
  - Place orders and make secure payments.
  - Track orders in real time.

Security Aspects:

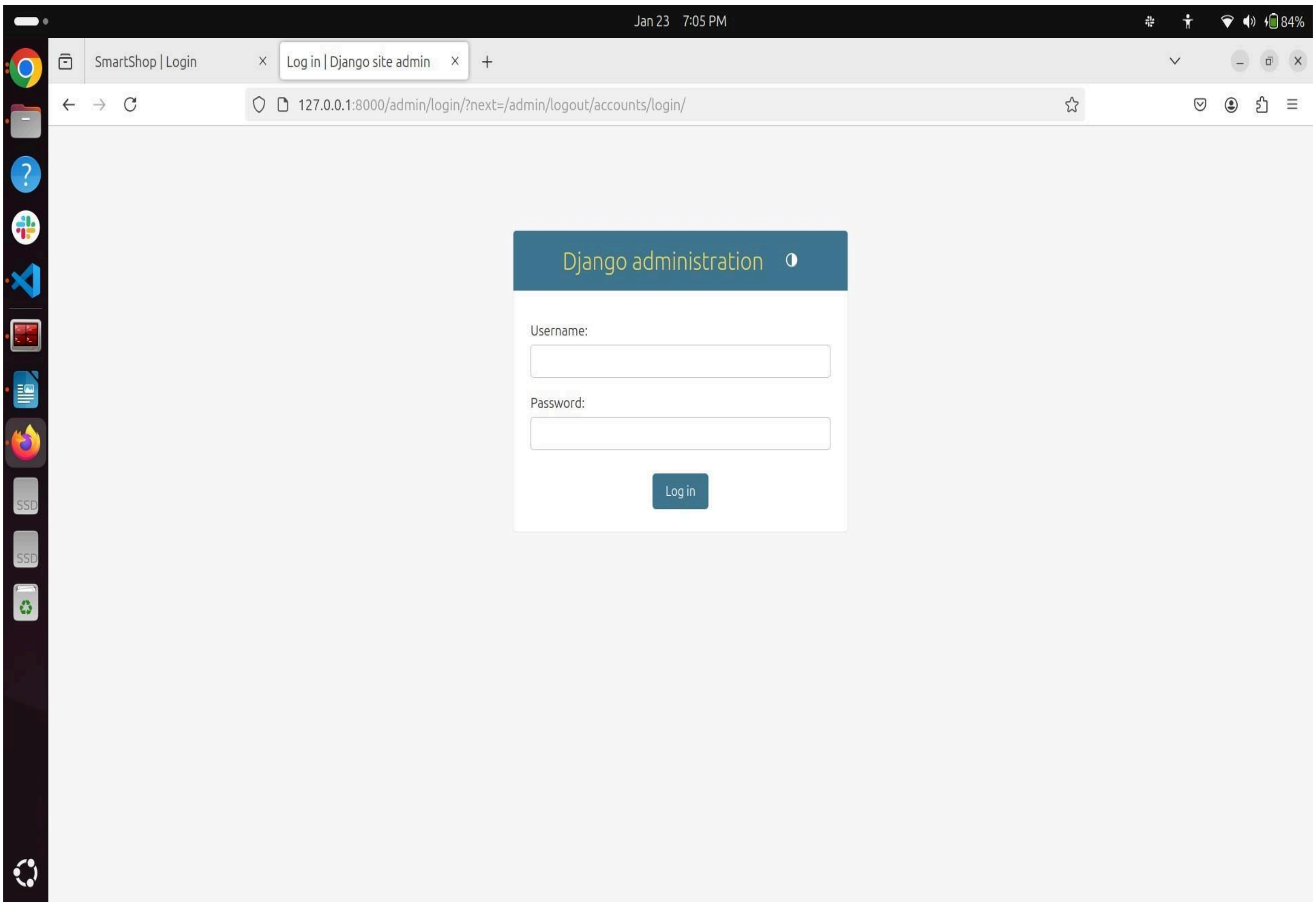
1. **Role-Based Access Control:** Admins, customers, and suppliers have distinct roles and permissions.
2. **Data Encryption:** Sensitive data, such as payment information, is encrypted to ensure security.
3. **Authentication:** Strong password policy for bot protection.

Backup Policy:

Weekly automated backups of the database to ensure minimal data loss.
Operational Controls:

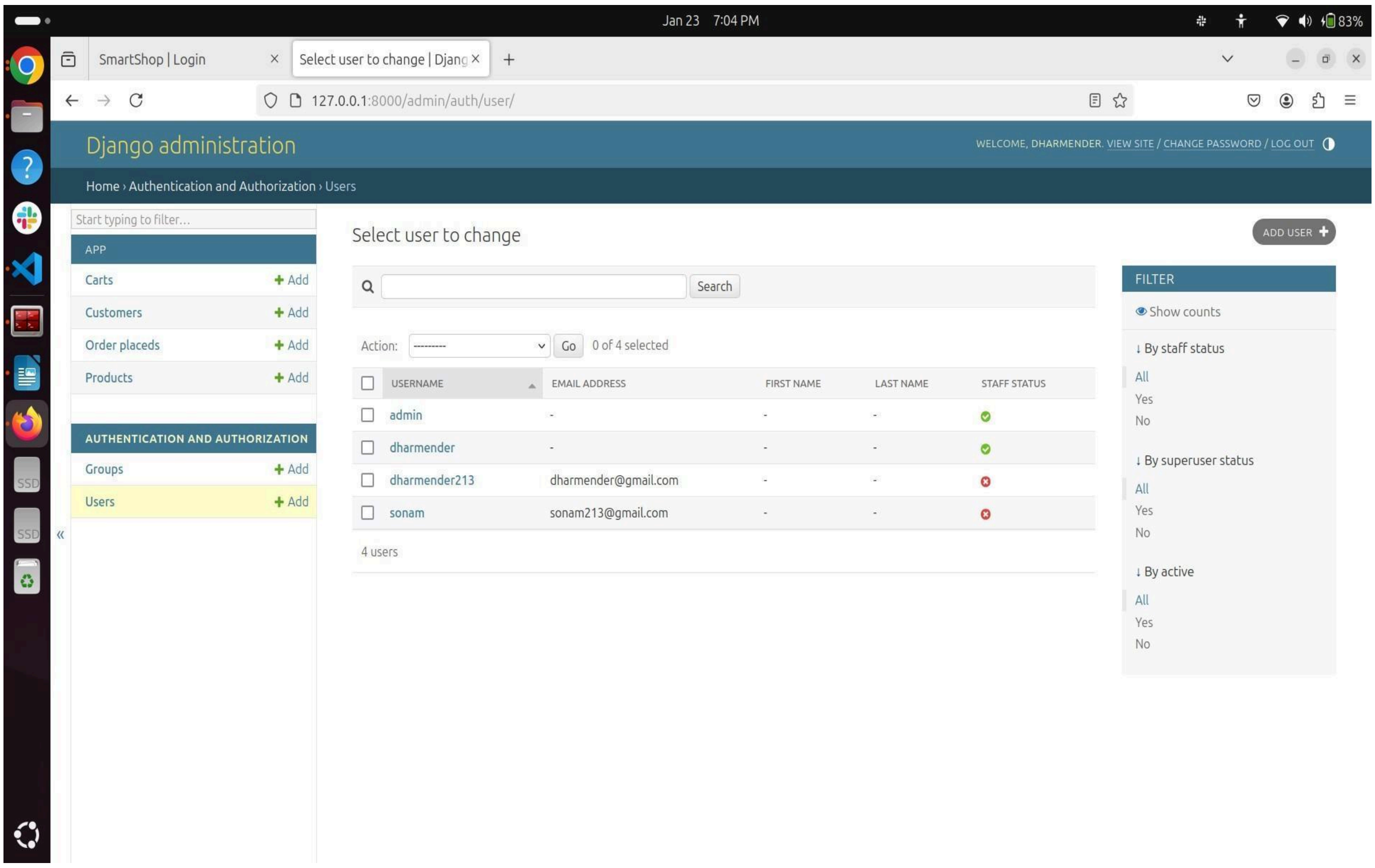
- Notifications to suppliers of low stock levels.
- Alerts for admins regarding unprocessed orders.

Screen shots


Jan 23 7:05 PM
84%
SmartShop | Login
Log in | Django site admin
127.0.0.1:8000/admin/login/?next=/admin/logout/accounts/login/
Django administration
Username:
Password:
Log in

Purpose : Admin Login Page

This image illustrates the login interface of the Django admin panel for the SmartShop project. It serves as a reference to check that the admin site is set up correctly and accessible during development, ensuring that the backend is configured properly for tasks such as managing users, products, and other database models.



Jan 23 7:04 PM

SmartShop | Login

Select user to change | Djang

127.0.0.1:8000/admin/auth/user/

Django administration

WELCOME, DHARMENDER. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Authentication and Authorization › Users

Start typing to filter...

APP	
Carts	+ Add
Customers	+ Add
Order placeds	+ Add
Products	+ Add

AUTHENTICATION AND AUTHORIZATION	
Groups	+ Add
Users	+ Add

Select user to change

ADD USER +

Search

Action: --------- Go 0 of 4 selected

USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
admin	-	-	-	✓
dharmender	-	-	-	✓
dharmender213	dharmender@gmail.com	-	-	✗
sonam	sonam213@gmail.com	-	-	✗

4 users

FILTER

Show counts

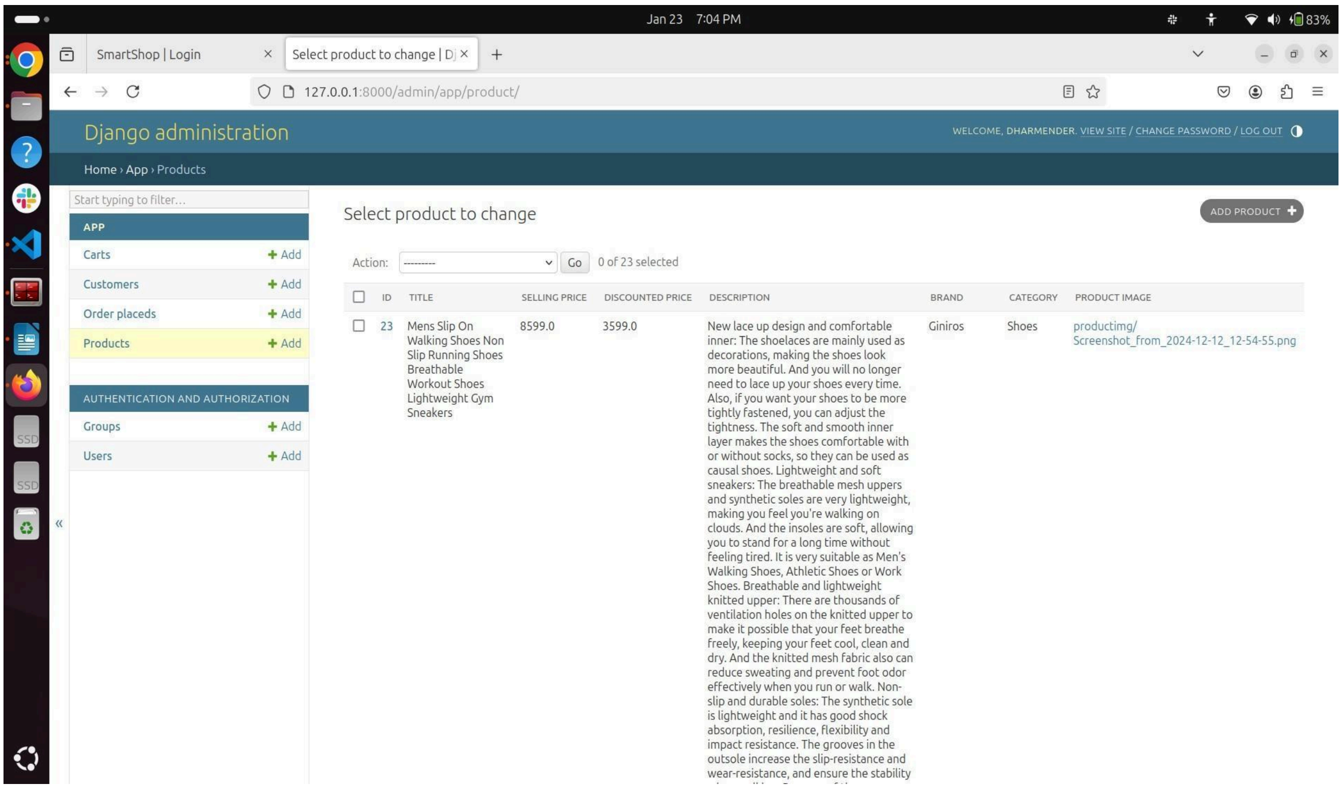
↓ By staff status
All
Yes
No

↓ By superuser status
All
Yes
No

↓ By active
All
Yes
No

Purpose : Django Admin Panel - User Management Page

This image illustrates the "User Management" section of the Django admin panel for the SmartShop project. It shows the interface used for managing user accounts, including information like usernames, email addresses, and staff status. Administrators can view, edit, or delete user accounts from this page, as well as add new users. Additionally, it features filtering options on the right side for sorting users by staff status, superuser status, or activity level, which helps streamline the management of authentication and authorization within the application.



Select product to change

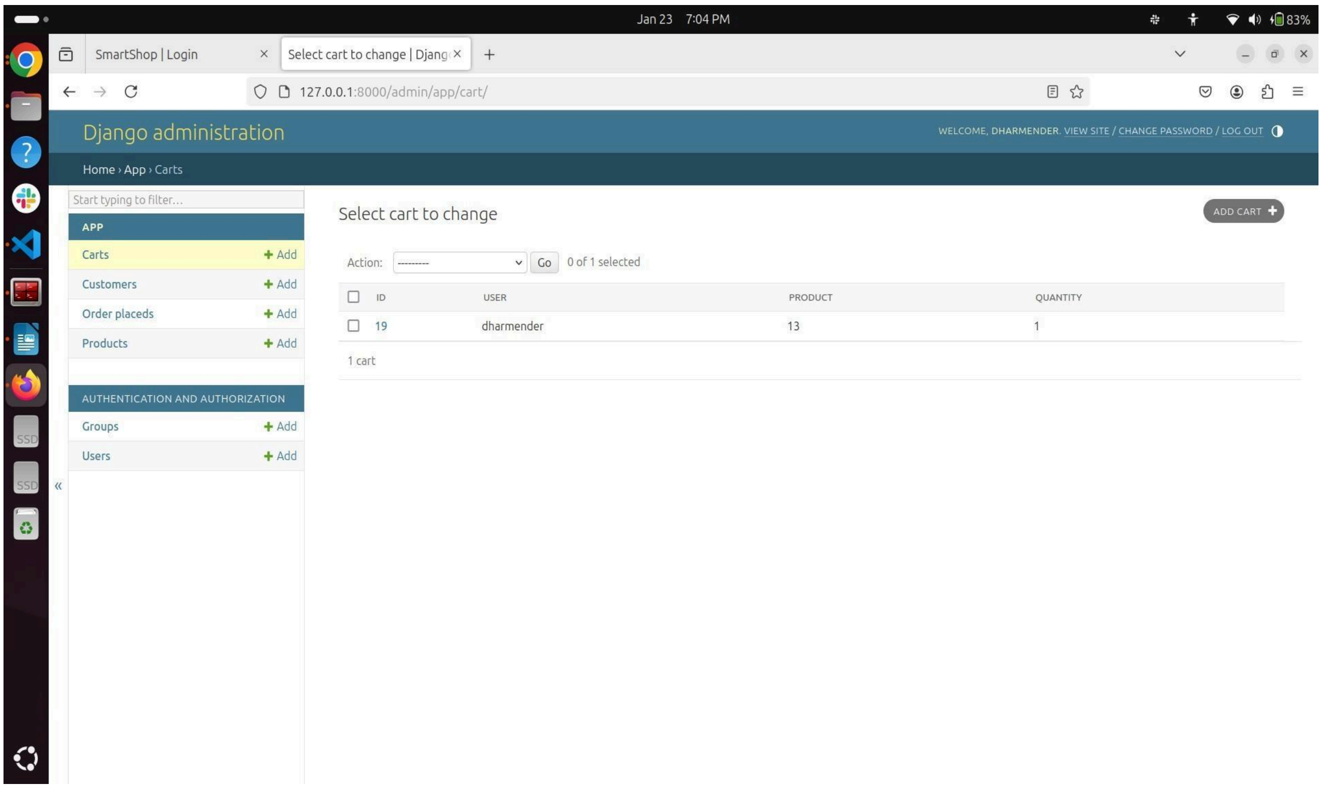
0 of 23 selected

ID	TITLE	SELLING PRICE	DISCOUNTED PRICE	DESCRIPTION	BRAND	CATEGORY	PRODUCT IMAGE
23	Mens Slip On Walking Shoes Non Slip Running Shoes Breathable Workout Shoes Lightweight Gym Sneakers	8599.0	3599.0	New lace up design and comfortable inner: The shoelaces are mainly used as decorations, making the shoes look more beautiful. And you will no longer need to lace up your shoes every time. Also, if you want your shoes to be more tightly fastened, you can adjust the tightness. The soft and smooth inner layer makes the shoes comfortable with or without socks, so they can be used as causal shoes. Lightweight and soft sneakers: The breathable mesh uppers and synthetic soles are very lightweight, making you feel you're walking on clouds. And the insoles are soft, allowing you to stand for a long time without feeling tired. It is very suitable as Men's Walking Shoes, Athletic Shoes or Work Shoes. Breathable and lightweight knitted upper: There are thousands of ventilation holes on the knitted upper to make it possible that your feet breathe freely, keeping your feet cool, clean and dry. And the knitted mesh fabric also can reduce sweating and prevent foot odor effectively when you run or walk. Non-slip and durable soles: The synthetic sole is lightweight and it has good shock absorption, resilience, flexibility and impact resistance. The grooves in the outsole increase the slip-resistance and wear-resistance, and ensure the stability	Giniros	Shoes	productimg/ Screenshot_from_2024-12-12_12-54-55.png

Purpose : Django Admin Panel - Product Management

This image represents the "Products" management section within the Django admin panel of the SmartShop platform. It showcases the interface where administrators can view, edit, and manage product details, including:

- **ID:** A unique identifier for each product.
- **Title:** The name of the product.
- **Selling Price and Discounted Price:** Displaying the original price and the discounted price.
- **Description:** A detailed product description highlighting features, materials, and usage.
- **Brand and Category:** Such as "Nike" under "Shoes."
- **Product Image:** A reference to the associated image file for the product.


Jan 23 7:04 PM
83%
SmartShop | Login
Select cart to change | Djang
127.0.0.1:8000/admin/app/cart/
Django administration
WELCOME, DHARMENDER. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › App › Carts
Start typing to filter...
APP
Carts + Add
Customers + Add
Order placeds + Add
Products + Add
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
ADD CART
Action: --------- Go 0 of 1 selected

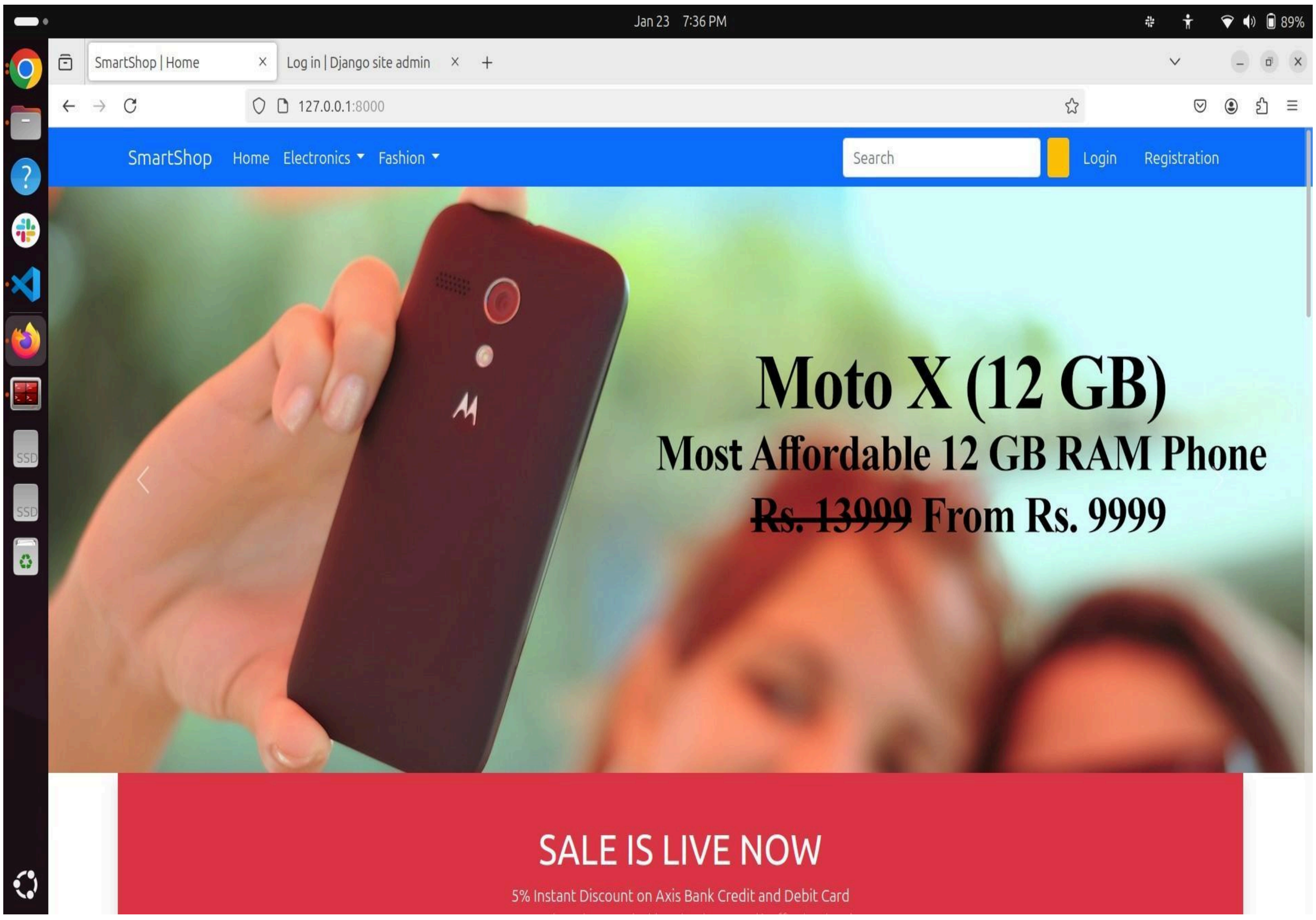
Select cart to change

ID	USER	PRODUCT	QUANTITY
19	dharmender	13	1

1 cart

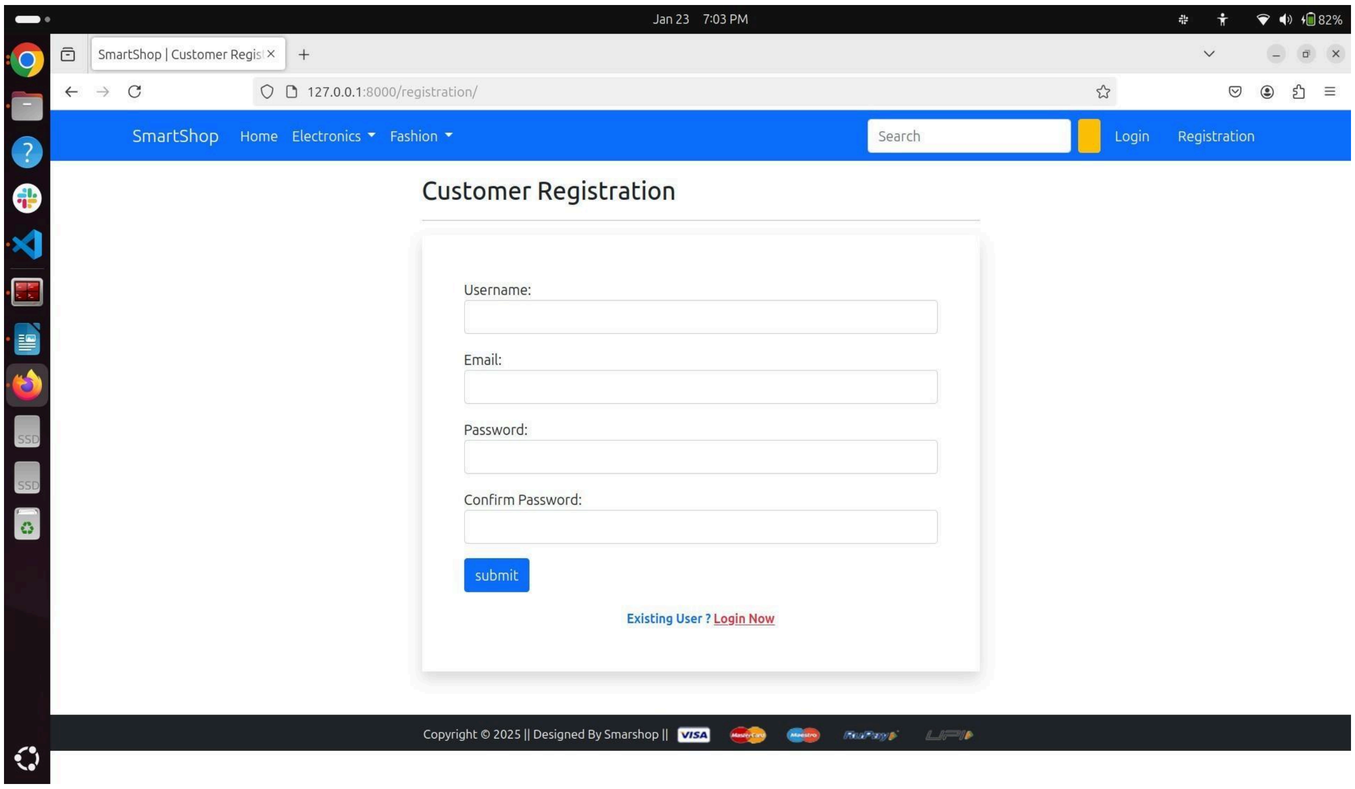
Purpose: Django Admin Panel - Cart Management

This image displays the "Carts" management section within the Django admin panel of the SmartShop platform. It serves several key functions: View Cart Entries: Lists all the cart entries made by users, including details such as the User (dharmender), the Product (ID 13), and the Quantity of the item in the cart.
Manage Carts: Enables administrators to add, edit, or delete cart entries, allowing direct oversight of user shopping cart activities.
Track User Activity: Provides visibility into the items users have added to their carts, supporting inventory monitoring and user-specific customization.


Jan 23 7:36 PM
89%
SmartShop | Home
Log in | Django site admin
127.0.0.1:8000
SmartShop
Home
Electronics
Fashion
Search
Login
Registration
Moto X (12 GB)
Most Affordable 12 GB RAM Phone
~~Rs. 13999~~ From Rs. 9999
SALE IS LIVE NOW
5% Instant Discount on Axis Bank Credit and Debit Card

Purpose: SmartShop Homepage - Product Promotion

This image shows the homepage of the SmartShop platform, highlighting the promotion of the Moto X (12 GB) smartphone. The page is designed to attract customer attention with a bold advertisement showcasing a significant discount, reducing the price from Rs. 13,999 to Rs. 9,999. The banner emphasizes the affordability and value of the product. The navigation bar at the top provides easy access to categories like "Electronics" and "Fashion," as well as options to log in or register. This layout demonstrates the platform's user-friendly design and its focus on promoting sales and product visibility.



Customer Registration

Username:

Email:

Password:

Confirm Password:

submit

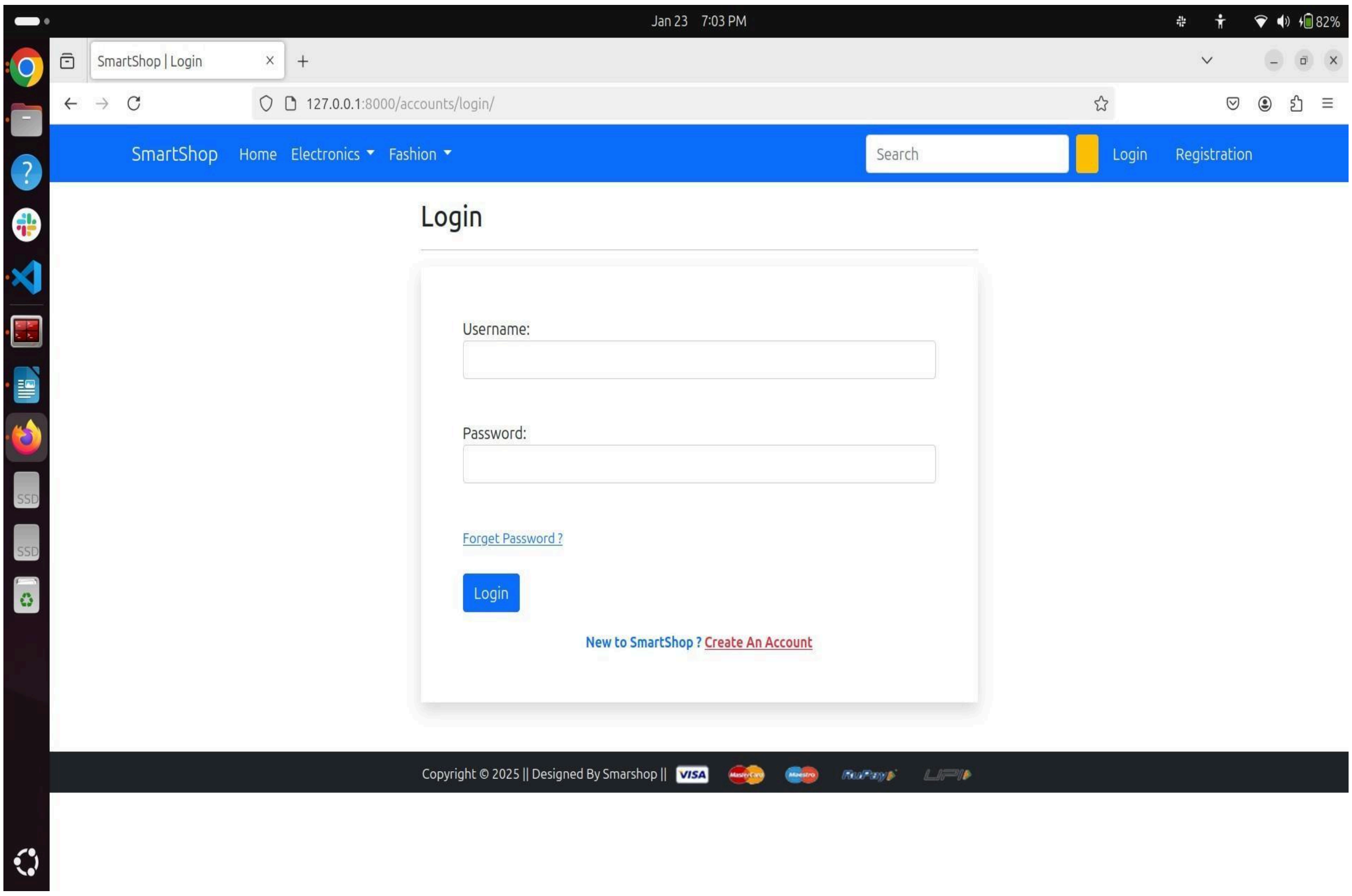
Existing User ? Login Now

Copyright © 2025 || Designed By Smarshop ||

Purpose: Customer Registration Page – SmartShop

This image shows the customer registration page of the SmartShop platform, emphasizing the following features:

- **User Registration:** Allows new users to create an account by providing essential details like **Username**, **Email**, **Password**, and confirming the password.
- **User Onboarding:** Offers an intuitive interface for registering new customers, facilitating access to personalized shopping experiences.
- **Navigation Links:** Provides a quick link for existing users to navigate to the login page using the "Login Now" option.
- **Brand Identity:** Features the SmartShop branding and a menu bar for easy navigation across different sections, such as **Electronics** and **Fashion**.
- **Security:** Ensures users input and confirm their password for added security during the account creation process.



Login

Username:

Password:

Forget Password ?

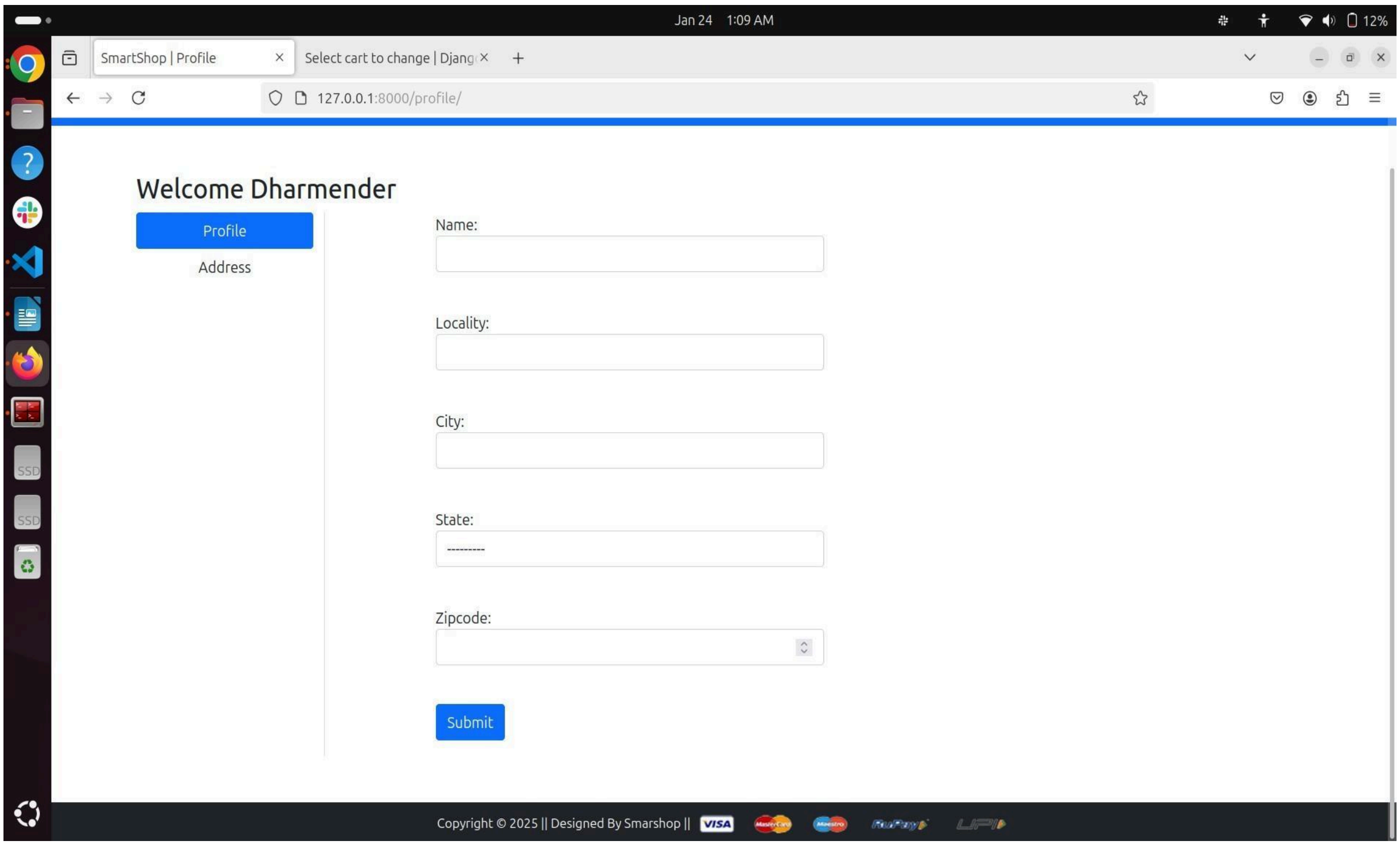
Login

New to SmartShop ? Create An Account

Copyright © 2025 || Designed By Smarshop ||

Purpose : SmartShop Customer Login Page

This image displays the login page of "SmartShop," an e-commerce website running on a local server (127.0.0.1:8000). The page includes fields for entering a username and password, a "Forgot Password?" link, and an option for new users to create an account. The top navigation bar provides access to different product categories, a search bar, and login/registration options. The footer contains copyright information and payment method icons. This page is likely part of a Django-based shopping website.


Jan 24 1:09 AM

SmartShop | Profile

Select cart to change | Djang

127.0.0.1:8000/profile/

Welcome Dharmender

Name:

Locality:

City:

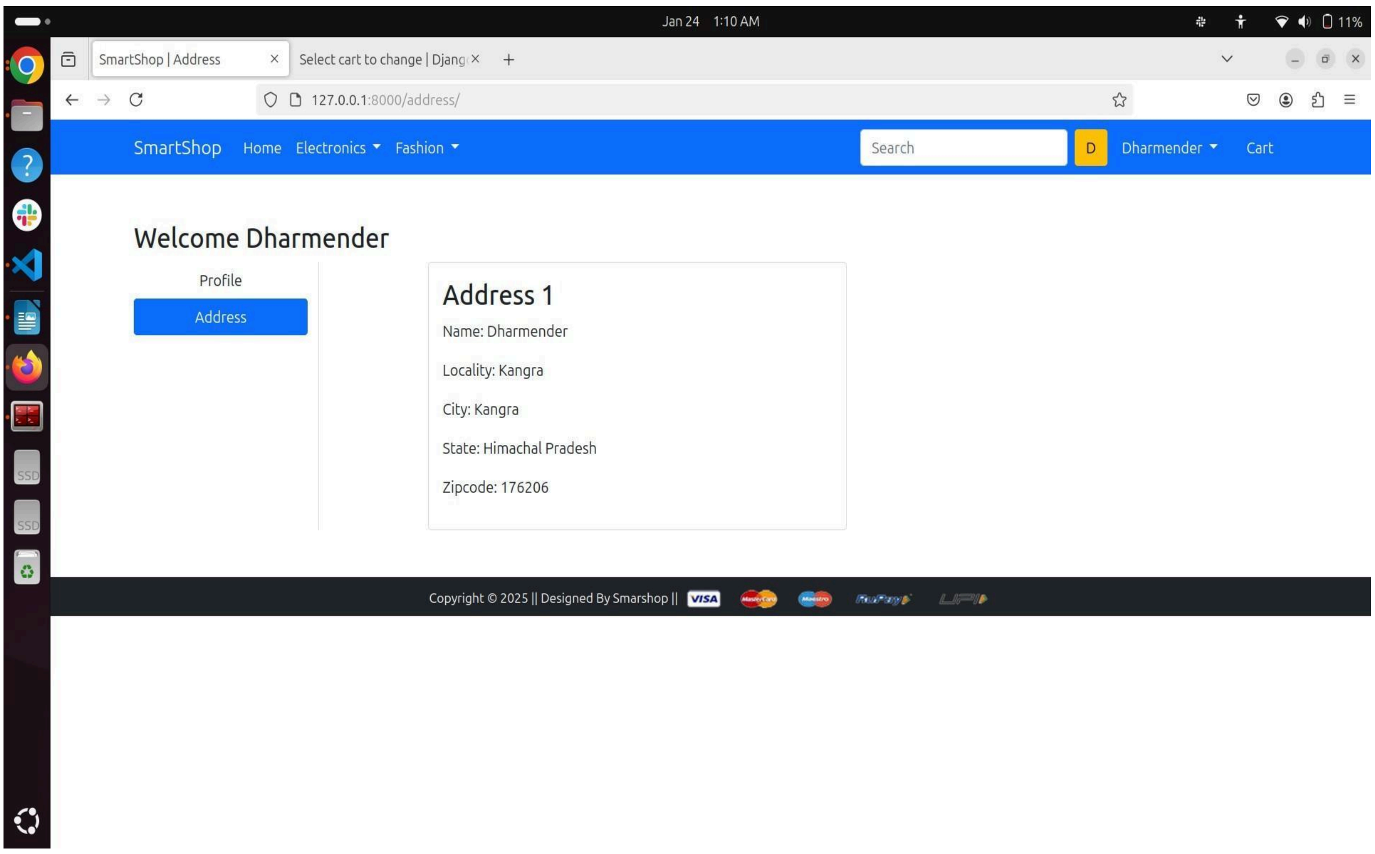
State: ---------

Zipcode:

Copyright © 2025 || Designed By Smarshop ||

Purpose : SmartShop User Profile Page

This image displays the user profile page of "SmartShop," an e-commerce website running on a local server (127.0.0.1:8000/profile/). After logging in, the user is welcomed by name and can manage their profile and address details. The form includes fields for the user's name, locality, city, state, and zip code, with a submit button to save changes. A sidebar allows switching between profile and address settings. The footer contains copyright information and supported payment methods. This page is designed to help users update their personal and shipping details.



Welcome Dharmender

Address 1

Name: Dharmender

Locality: Kangra

City: Kangra

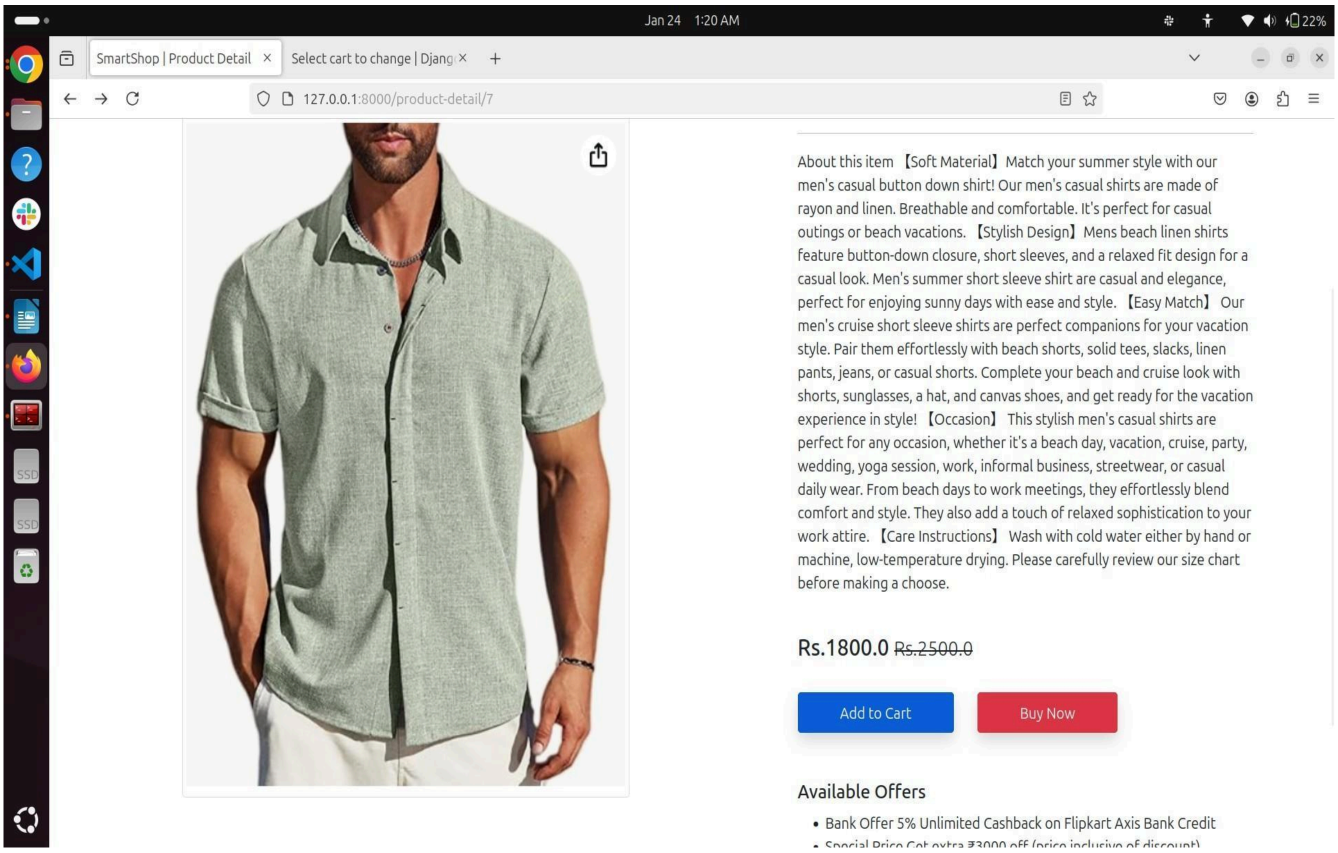
State: Himachal Pradesh

Zipcode: 176206

Copyright © 2025 || Designed By Smarshop ||

Purpose : SmartShop User Address Page

This image displays the **user address page** of "SmartShop," an e- commerce website running on a local server (127.0.0.1:8000/address/). After the user logs in and enters their profile details, this page shows the saved address information, including **name, locality, city, state, and zip code**. The sidebar allows navigation between the **Profile** and **Address** sections. The page ensures that the correct shipping details are stored for future orders. The footer contains copyright information and supported payment methods.


Jan 24 1:20 AM
22%
SmartShop | Product Detail
Select cart to change | Djang
127.0.0.1:8000/product-detail/7

About this item 【Soft Material】 Match your summer style with our men's casual button down shirt! Our men's casual shirts are made of rayon and linen. Breathable and comfortable. It's perfect for casual outings or beach vacations. 【Stylish Design】 Mens beach linen shirts feature button-down closure, short sleeves, and a relaxed fit design for a casual look. Men's summer short sleeve shirt are casual and elegance, perfect for enjoying sunny days with ease and style. 【Easy Match】 Our men's cruise short sleeve shirts are perfect companions for your vacation style. Pair them effortlessly with beach shorts, solid tees, slacks, linen pants, jeans, or casual shorts. Complete your beach and cruise look with shorts, sunglasses, a hat, and canvas shoes, and get ready for the vacation experience in style! 【Occasion】 This stylish men's casual shirts are perfect for any occasion, whether it's a beach day, vacation, cruise, party, wedding, yoga session, work, informal business, streetwear, or casual daily wear. From beach days to work meetings, they effortlessly blend comfort and style. They also add a touch of relaxed sophistication to your work attire. 【Care Instructions】 Wash with cold water either by hand or machine, low-temperature drying. Please carefully review our size chart before making a choose.

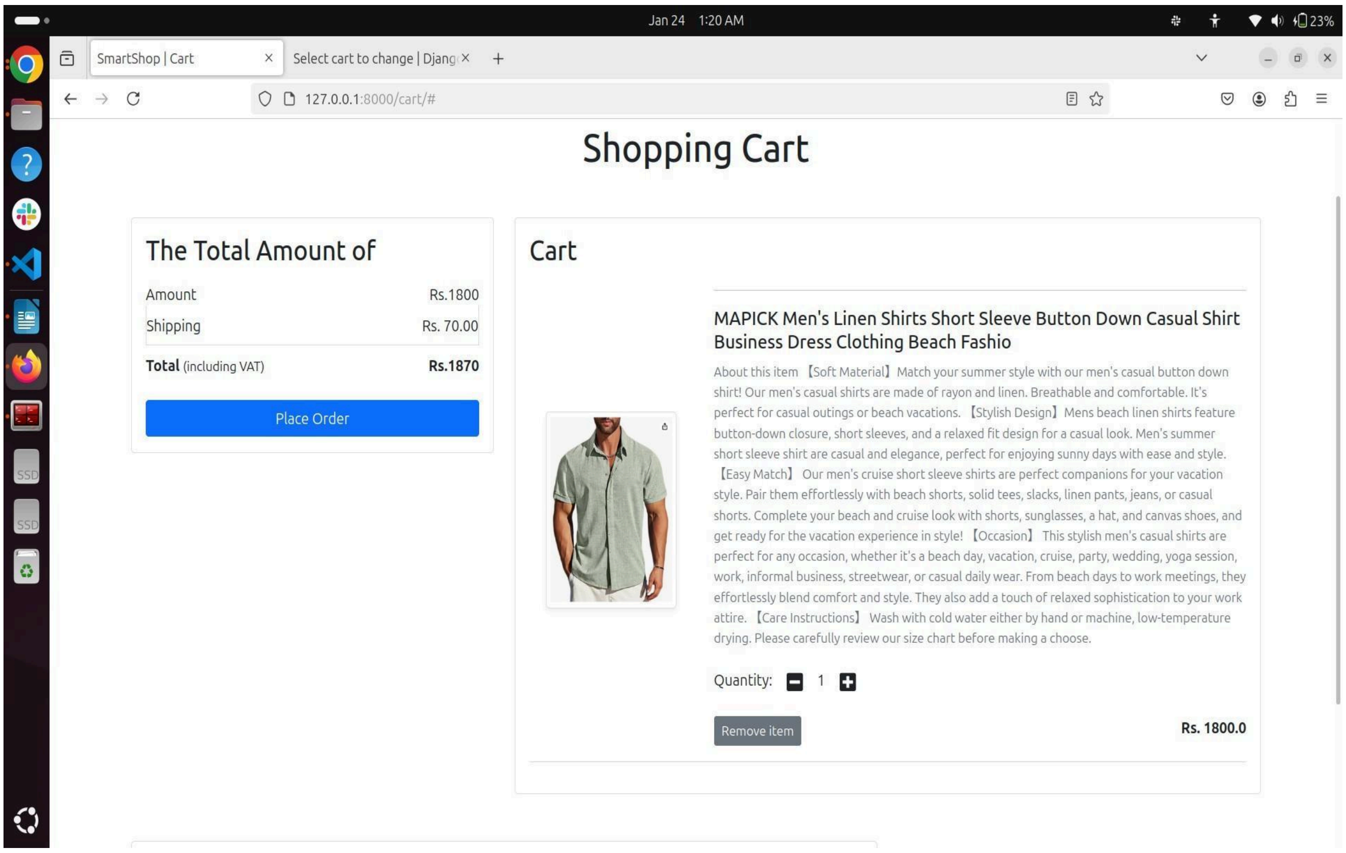
Rs.1800.0 ~~Rs.2500.0~~

Available Offers

- Bank Offer 5% Unlimited Cashback on Flipkart Axis Bank Credit

Purpose: SmartShop Product Detail Page

This image shows the **product detail page** of the SmartShop e-commerce website, displaying a men's casual button-down shirt. The page includes a **product image, description, price, and available offers**. Users have two options: **"Add to Cart"** and **"Buy Now"**. However, the **"Add to Cart"** button is only functional if the user is logged into their account. This ensures that products are correctly linked to a registered user for seamless order management.



Jan 24 1:20 AM

SmartShop | Cart

Select cart to change | Djang

127.0.0.1:8000/cart/#

Shopping Cart

The Total Amount of

Amount	Rs.1800
Shipping	Rs. 70.00
Total (including VAT)	**Rs.1870**

Place Order

Cart

MAPICK Men's Linen Shirts Short Sleeve Button Down Casual Shirt Business Dress Clothing Beach Fashio

About this item 【Soft Material】 Match your summer style with our men's casual button down shirt! Our men's casual shirts are made of rayon and linen. Breathable and comfortable. It's perfect for casual outings or beach vacations. 【Stylish Design】 Mens beach linen shirts feature button-down closure, short sleeves, and a relaxed fit design for a casual look. Men's summer short sleeve shirt are casual and elegance, perfect for enjoying sunny days with ease and style. 【Easy Match】 Our men's cruise short sleeve shirts are perfect companions for your vacation style. Pair them effortlessly with beach shorts, solid tees, slacks, linen pants, jeans, or casual shorts. Complete your beach and cruise look with shorts, sunglasses, a hat, and canvas shoes, and get ready for the vacation experience in style! 【Occasion】 This stylish men's casual shirts are perfect for any occasion, whether it's a beach day, vacation, cruise, party, wedding, yoga session, work, informal business, streetwear, or casual daily wear. From beach days to work meetings, they effortlessly blend comfort and style. They also add a touch of relaxed sophistication to your work attire. 【Care Instructions】 Wash with cold water either by hand or machine, low-temperature drying. Please carefully review our size chart before making a choose.

Quantity: 1

Remove item

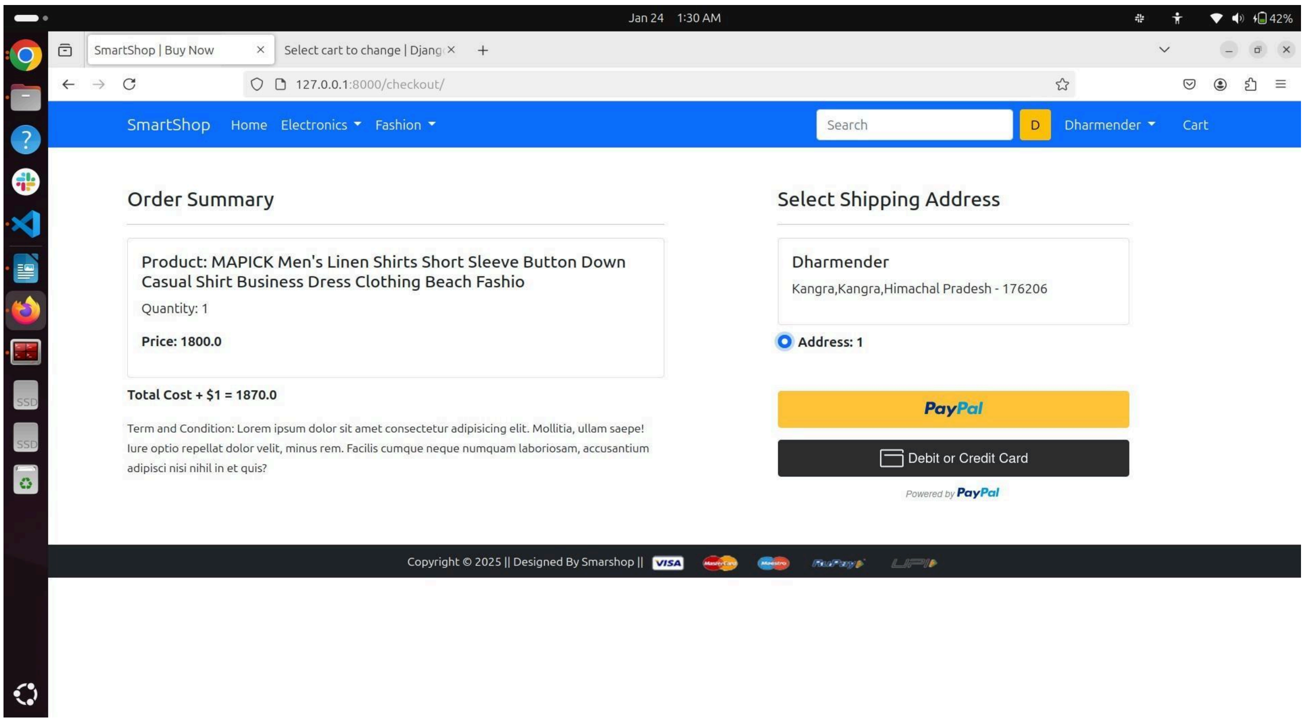
Rs. 1800.0

Purpose : Shopping Cart Page – SmartShop

This image illustrates the SmartShop shopping cart page, showing:

- Product details (image, description, price, and quantity).
- Options to update quantity or remove items.
- Order summary with total amount, including shipping and VAT.
- A "Place Order" button for proceeding to checkout.

It ensures users can review and finalize their purchases efficiently.


127.0.0.1:8000/checkout/

SmartShop Home Electronics Fashion Search D Dharmender Cart

Order Summary

Product: MAPICK Men's Linen Shirts Short Sleeve Button Down Casual Shirt Business Dress Clothing Beach Fashio

Quantity: 1

Price: 1800.0

Total Cost + $1 = 1870.0

Term and Condition: Lorem ipsum dolor sit amet consectetur adipisicing elit. Mollitia, ullam saepe! Iure optio repellat dolor velit, minus rem. Facilis cumque neque numquam laboriosam, accusantium adipisci nisi nihil in et quis?

Select Shipping Address

Dharmender

Kangra,Kangra,Himachal Pradesh - 176206

Address: 1

PayPal

Debit or Credit Card

Powered by PayPal

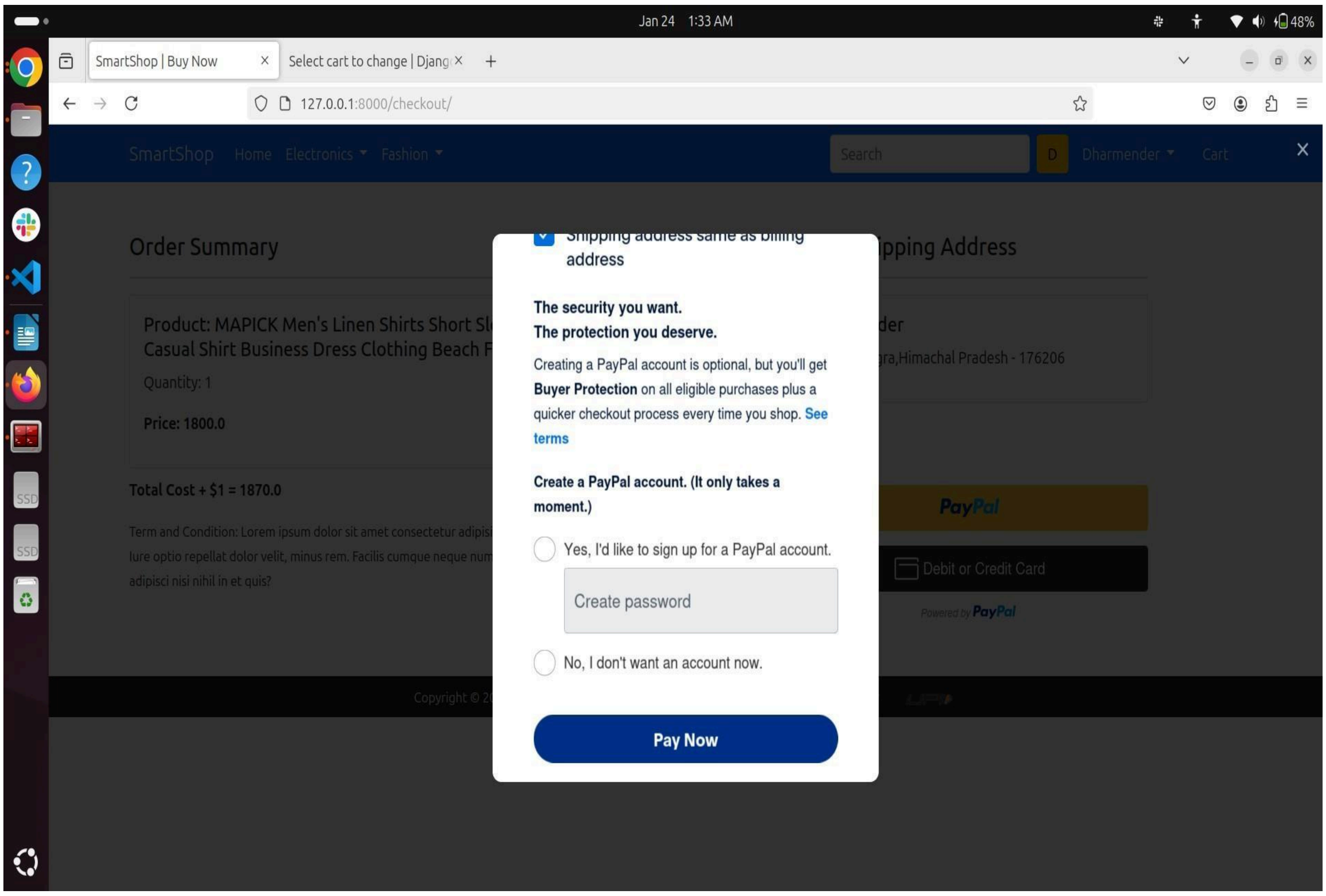
Copyright © 2025 || Designed By Smarshop ||

Purpose :Checkout Page – SmartShop

This image shows the SmartShop checkout page after clicking "Place Order." It includes:

- **Order Summary:** Product details (name, quantity, price, and total cost with shipping).
- **Shipping Address Selection:** Allows users to choose the delivery address.
- **Payment Options:** Provides options for PayPal or credit/debit card payment.

The page ensures a streamlined process for users to confirm order details and complete payment securely.



Jan 24 1:33 AM

SmartShop | Buy Now

Select cart to change | Djang

127.0.0.1:8000/checkout/

SmartShop Home Electronics Fashion Search Dharmender Cart

Order Summary

Product: MAPICK Men's Linen Shirts Short Sl
Casual Shirt Business Dress Clothing Beach F

Quantity: 1

Price: 1800.0

Total Cost + $1 = 1870.0

Term and Condition: Lorem ipsum dolor sit amet consectetur adipisi
Iure optio repellat dolor velit, minus rem. Facilis cumque neque num
adipisci nisi nihil in et quis?

ipping Address

der
ra,Himachal Pradesh - 176206

PayPal

Debit or Credit Card

Powered by PayPal

Copyright © 2

Shipping address same as billing address

The security you want.
The protection you deserve.

Creating a PayPal account is optional, but you'll get **Buyer Protection** on all eligible purchases plus a quicker checkout process every time you shop. See terms

Create a PayPal account. (It only takes a moment.)

Yes, I'd like to sign up for a PayPal account.

Create password

No, I don't want an account now.

Pay Now

Purpose : PayPal Payment Modal – SmartShop

This image displays the PayPal payment modal that appears after selecting the "PayPal" option on the checkout page. It provides:

- **Account Creation Option:** Users can choose to create a PayPal account or proceed without one.
- **Buyer Protection Details:** Highlights the benefits of using PayPal, such as buyer protection and faster checkout.
- **Payment Confirmation:** The "Pay Now" button allows users to finalize their purchase securely.

This modal ensures a seamless and secure payment experience for SmartShop customers using PayPal.

Sample Code

base.html ::

```
<!doctype html>
{% load static %}
<html lang="en">
<head>
<!-- Required meta tags -->
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<!-- Bootstrap CSS -->
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/css/bootstrap
.min.css"
rel="stylesheet" integrity="sha384-
giJF6kkoqNQ00vy+HMDP7azOuL0xtbfIcaT9wjKHr8RbDVddVHyTfAAsrekwK
mP1"
crossorigin="anonymous">
<!--Owl Carousel CSS-->
<link rel="stylesheet" href="{% static 'app/css/owl.carousel.min.css' %}">
<!--FontAwesome CSS-->
<link rel="stylesheet" href="{% static 'app/css/all.min.css' %}">
<!--Custom CSS-->
<link rel="stylesheet" href="{% static 'app/css/style.css' %}">
<title>SmartShop | {% block title %} {% endblock title %} </title>
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-primary">
<div class="container">
<a class="navbar-brand" href="/">SmartShop</a>
<button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-
target="#navbarSupportedContent"
aria-controls="navbarSupportedContent" aria-
expanded="false" aria-label="Toggle navigation">
<span class="navbar-toggler-icon"></span>
</button>
```

```
<div class="collapse navbar-collapse" id="navbarSupportedContent">
<ul class="navbar-nav me-auto mb-2 mb-lg-0">
<li class="nav-item">
<a class="nav-link active" aria-current="page" href="/">Home</a></li><li
class="nav-item dropdown">
<a class="nav-link dropdown-toggle text-white" href="#"
id="electronicsDropdown" role="button"
data-bs-toggle="dropdown" aria-expanded="false">
Electronics
</a>
<ul class="dropdown-menu" aria-labelledby="electronicsDropdown">
<li><a class="dropdown-item" href="{% url 'mobile' %}">Mobile</a></li>
<li><a class="dropdown-item" href="{% url 'laptop' %}">Laptop</a></li>
</ul>
</li>
<li class="nav-item dropdown">
<a class="nav-link dropdown-toggle text-white" href="#"
id="fashionDropdown" role="button"
data-bs-toggle="dropdown" aria-expanded="false">
Fashion
</a>
<ul class="dropdown-menu" aria-labelledby="fashionDropdown">
<li><a class="dropdown-item" href="topwear">Top Wear</a></li>
<li><a class="dropdown-item" href="bottomwear">Bottom Wear</a></li>
</ul>
</li>
</ul>
<form class="d-flex">
<input class="form-control me-2" type="search" placeholder="Search"
aria-label="Search">
<button class="btn btn-warning" type="submit">{{
request.user.username|first }}</button>
<!-- <button class="btn btn-warning" type="submit">S</button> -->
</form>
<div>
<ul class="navbar-nav me-auto mb-2 mb-lg-0">
{% if request.user.is_authenticated %}
```

```
<li class="nav-item dropdown mx-2">
<a class="nav-link dropdown-toggle text-white" href="#"
id="profileDropdown" role="button"
data-bs-toggle="dropdown" aria-expanded="false">
{{ request.user.username|capfirst }}
</a>
<ul class="dropdown-menu" aria-labelledby="profileDropdown">
<li><a class="dropdown-item" href="{% url 'profile' %}">Profile</a></li>
<li><a class="dropdown-item" href="{% url 'orders' %}">Orders</a></li>
<li><a class="dropdown-item" href="{% url 'changepassword' %}">Change
Password</a></li>
<li><a class="dropdown-item" href="{% url 'logout' %}">Logout</a></li>
<!-- <li><a class="dropdown-item" href="{% url 'login'
%}">Logout</a></li> -->
</ul>
</li>
<li class="nav-item mx-2">
<a href="{% url 'showcart' %}" class="nav-link text-white"><span
class="badge bg-
danger">4</span> Cart </a>
</li>
{% else %}
<li class="nav-item mx-2"><a href="{% url 'login' %}" class="nav-link
text-white">Login</a>
</li>
<li class="nav-item mx-2">
<a href="{% url 'customerregistration' %}" class="nav-link
text-white">Registration</a>
</li>{% endif %}
</ul>
</div>
</div>
</div>
</nav>
<!-- Start Footer -->
<footer class="container-fluid bg-dark text-center p-2 mt-5">
```

```
<small class="text-white">Copyright &copy; 2025 || Designed By Smarshop ||
</small>
<img src="{% static 'app/images/payment.png' %}" alt="" srcset=""
class="img-fluid"
height="2px">
</footer> <!-- End Footer -->
<!-- Jquery -->
<script src="https://code.jquery.com/jquery-3.5.1.min.js"
integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
crossorigin="anonymous"></script>
<!-- Bootstrap Bundle with Popper -->
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/js/bootstrap.b
undle.min.js"
integrity="sha384-ygbV9kiqUc6oa4msXn9868pTtWMgiQaeYH7/
t7LECLbyPA2x65Kgf80OJFdroafW"
crossorigin="anonymous"></script>
<script src="{% static 'app/js/owl.carousel.min.js' %}"></script>
<script src="{% static 'app/js/all.min.js' %}"></script>
<script src="{% static 'app/js/myscript.js' %}"></script>
</body>
</html>
```
—--

admin.py :

```
from django.contrib import admin
from .models import (Customer,
Product,
Cart,
OrderPlaced)
@admin.register(Customer)
class CustomerModelAdmin(admin.ModelAdmin):
list_display = ['id','user','name',
'locality','city','zipcode','state']
@admin.register(Product)class ProductModelAdmin(admin.ModelAdmin):
list_display = ['id','title','selling_price',
'discounted_price','description','brand',
'category','product_image']
@admin.register(Cart)
class CartModelAdmin(admin.ModelAdmin):
list_display = ['id','user','product','quantity']
@admin.register(OrderPlaced)
class OrderPlacedModelAdmin(admin.ModelAdmin):
list_display=
['id','user','customer','product','quantity','ordered_date','status']
```

—---

forms.py :

```
from django import forms
from django.contrib.auth.forms import UserCreationForm,
AuthenticationForm,UsernameField,
PasswordChangeForm, PasswordResetForm, SetPasswordForm
from django.contrib.auth.models import User
from django.utils.translation import gettext, gettext_lazy as _
from django.contrib.auth import password_validation
from .models import Customer
## Customer registration form
class CustomerRegistrationForm(UserCreationForm):
password1 =
forms.CharField(label='Password',widget=forms.PasswordInput(attrs={'class
':'form-
control'}))
password2 = forms.CharField(label='Confirm Password',widget=
forms.PasswordInput(attrs={'class':'form-control'}))
email =
forms.EmailField(widget=forms.EmailInput(attrs={'class':'form-control'}))
class Meta:
model = User
fields = ['username','email','password1','password2']
labels = {'email':'Email'}
```

```
widgets = {'username':forms.TextInput(attrs={'class':'form-control'}),
'email':forms.EmailInput(attrs={'class':'form-control'})}
## Customer login form
class LoginForm(AuthenticationForm):
username                                                              =
UsernameField(widget=forms.TextInput(attrs={'autofocus':True,'class':'form
-
control'}))
password =
forms.CharField(label=_("Password"),strip=False,widget=forms.PasswordIn
put(attrs={'autocomple
te':'current-password','class':'form-control'}))
## Customer password change form
class
MyPasswordChangeForm(PasswordChangeForm):old_password
=
forms.CharField(label=_("Old
Password"),strip=False,widget=forms.PasswordInput(attrs={'autocomplete':'
current-
password','autofocus':True,'class':'form-control'}))
new_password1 = forms.CharField(label=_("New
Password"),strip=False,widget=forms.PasswordInput(attrs={'autocomplete':'
new-
password','class':'form-control'}))
new_password2 = forms.CharField(label=_("Confirm New
Password"),strip=False,widget=forms.PasswordInput(attrs={'autocomplete':'
new-
password','class':'form-control'}))
class Meta:
model = User
fields = ['old_password','new_password1','new_password2']
## Customer password reset form
class MyPasswordResetForm(PasswordResetForm):
email
=forms.EmailField(label=_("Email"),max_length=254,widget=forms.EmailIn
put(attrs={'autocomplete ':'email','class':'form-control'}))
## Customer password set form
```

```
class MySetPasswordForm(SetPasswordForm):
new_password1 = forms.CharField(label=_("New
Password"),strip=False,widget=forms.PasswordInput(attrs={'autocomplete':'
new-
password','class':'form-control'}),
help_text=password_validation.password_validators_help_text_html())
new_password2 = forms.CharField(label=_("Confirm New
Password"),strip=False,widget=forms.PasswordInput(attrs={'autocomplete':'
new-
password','class':'form-control'}))
## Customer profile form
class CustomerProfileForm(forms.ModelForm):
class Meta:
model = Customer
fields = ['name','locality','city','state','zipcode']
widgets = {'name':forms.TextInput(attrs={'class':'form-control'}),
'locality':forms.TextInput(attrs={'class':'form-control'}),
'city':forms.TextInput(attrs={'class':'form-control'}),
'state':forms.Select(attrs={'class':'form-control'}),
'zipcode':forms.NumberInput(attrs={'class':'form-control'})}
```

Models.py :

```
from django.db import models
from django.contrib.auth.models import User
from django.core.validators import MaxValueValidator,MinValueValidator
STATE_CHOICES = (
('Andaman & Nicobar Islands','Andaman & Nicobar Islands'),
('Andhra Pradesh','Andhra Pradesh'),('Arunachal Pradesh','Arunachal
Pradesh'),
('Assam','Assam'),
('Bihar','Bihar'),
('Chandigarh','Chandigarh'),
('Chhattigarh','Chhattisgarh'),
('Dadra & Nagar Haveli','Dadra & Nagar Haveli'),
('Daman and Diu','Daman and Diu'),
('Delhi','Delhi'),
('Goa','Goa'),
('Haryana','Haryana'),
('Himachal Pradesh','Himachal Pradesh'),
('Jammu & Kashmir','Jammu & Kashmir'),
('Jharkhand','Jharkhand'),
('Karnataka','Karnataka'),
('Kerala','Kerala'),
('Lakshadweep','Lakshadweep'),
('Madhya Pradesh','Madhya Pradesh'),
('Maharashtra','Maharashtra'),
('Manipur','Manipur'),
('Meghalaya','Meghalaya'),('Mizoram','Mizoram'),
('Nagaland','Nagaland'),
('Odisha','Odisha'),
('Puducherry','Puducherry'),
('Punjab','Punjab'),
('Rajasthan','Rajasthan'),
('Sikkim','Sikkim'),
('Tamil Nadu','Tamil Nadu'),
('Telangana','Telangana'),
('Tripura','Tripura'),
('Uttrakhand','Uttrakhand'),
('Uttar Pradesh','Uttar Pradesh'),
('West Bengal','West Bengal'),
)
class Customer(models.Model):
user = models.ForeignKey(User, on_delete=models.CASCADE)
name = models.CharField(max_length=50)
locality = models.CharField(max_length=200)
city = models.CharField(max_length=50)
zipcode = models.IntegerField()
state = models.CharField(choices=STATE_CHOICES,max_length=50)
def ___str___(self):
return str(self.id)
CATEGORY_CHOICES = (
('M','Mobile'),('L','Laptop'),
('TW','Top Wear'),
('BW', 'Bottom Wear'),
('K','Kids'),
('S','Shoes'),
)
class Product(models.Model):
title = models.CharField(max_length=100)
selling_price = models.FloatField()
discounted_price = models.FloatField()
description = models.TextField()
brand = models.CharField(max_length=100)
category = models.CharField(choices=CATEGORY_CHOICES,
max_length=2)
product_image = models.ImageField(upload_to='productimg')
def ___str___(self):
return str(self.id)
class Cart(models.Model):
user = models.ForeignKey(User, on_delete=models.CASCADE)
product = models.ForeignKey(Product, on_delete=models.CASCADE)
quantity = models.PositiveIntegerField(default=1)
def ___str___(self):
return str(self.id)
@property
def total_cost(self):
return self.quantity * self.product.discounted_price
STATUS_CHOICES = (
('Accepted','Accepted'),('Packed','Packed'),
('On The Way','On The Way'),
('Delivered','Delivered'),
('Cancel','Cancel')
)
class OrderPlaced(models.Model):
user = models.ForeignKey(User, on_delete=models.CASCADE)
customer = models.ForeignKey(Customer, on_delete=models.CASCADE)
product = models.ForeignKey(Product, on_delete=models.CASCADE)
quantity = models.PositiveIntegerField(default=1)
```

```
ordered_date = models.DateTimeField(auto_now_add=True)
status                                                                    =
models.CharField(max_length=50,choices=STATE_CHOICES,default='Pendin
g')
@property
def total_cost(self):
return self.quantity * self.product.discounted_price
```

—--

Views.py:

```
from django.http import JsonResponse
from django.shortcuts import render, redirect
from django.views import View
from .models import Customer, Product, Cart, OrderPlaced
from .forms import CustomerRegistrationForm, CustomerProfileForm
from django.contrib import messages
from django.db.models import Q
from django.contrib.auth.models import User
from django.contrib.auth.decorators import login_required
from django.utils.decorators import method_decorator
## Product
class ProductView(View):
def get(self,request):
mobiles = Product.objects.filter(category='M')
laptops = Product.objects.filter(category='L')
topwears = Product.objects.filter(category='TW')
bottomwears = Product.objects.filter(category='BW')
kids = Product.objects.filter(category='K')
shoes = Product.objects.filter(category='S')
return render(request, 'app/home.html',
{'topwears':topwears, 'bottomwears':bottomwears, 'kids':kids,'shoes':shoes,
'mobiles': mobiles,'laptops':laptops})
## Product Detail
class ProductDetailView(View):
def get(self,request, pk):
product = Product.objects.get(pk=pk)
item_already_in_cart = False
```

```
if request.user.is_authenticated:
item_already_in_cart = Cart.objects.filter(Q(product=product.id) & Q(user=request.user)).exists()return render(request,'app/productdetail.html', {'product': product})
## Add to cart
def add_to_cart(request):
user = request.user
product_id = request.GET.get('prod_id')
product = Product.objects.get(id=product_id)
Cart(user=user, product=product).save()
return redirect('/cart')
## Show cart
def show_cart(request):
if request.user.is_authenticated:
user = request.usercart = Cart.objects.filter(user=user)
amount = 0.0
shipping_amount = 70.0
total_amount = 0.0
cart_product = [p for p in Cart.objects.all() if p.user == user]
if cart_product:
for p in cart_product:
tempamount = (p.quantity * p.product.discounted_price)
amount += tempamount
total_amount = amount + shipping_amount
return render(request, 'app/addtocart.html',
{'carts': cart, 'totalamount': total_amount, 'amount': amount})
else:
return render(request, 'app/emptycart.html')
## Plus cart
def plus_cart(request):
if request.method == 'GET':
prod_id = request.GET['prod_id']
c = Cart.objects.get(Q(product=prod_id) & Q(user=request.user))
c.quantity += 1
c.save()
amount = 0.0
shipping_amount = 70.0
```

```
cart_product = [p for p in Cart.objects.all() if p.user == request.user]
for p in cart_product:
tempamount = (p.quantity * p.product.discounted_price)
amount += tempamount
data = {
'quantity': c.quantity,
'amount': amount,
'totalamount': amount + shipping_amount
}
return JsonResponse(data)
## Minus cart
def minus_cart(request):
if request.method == 'GET':
prod_id = request.GET['prod_id']
c = Cart.objects.get(Q(product=prod_id) & Q(user=request.user))
c.quantity -= 1
c.save()amount = 0.0
shipping_amount = 70.0
cart_product = [p for p in Cart.objects.all() if p.user == request.user]
for p in cart_product:
tempamount = (p.quantity * p.product.discounted_price)
amount += tempamount
data = {
'quantity': c.quantity,'amount': amount,
'totalamount': amount + shipping_amount
}
return JsonResponse(data)
## Remove cart
def remove_cart(request):
if request.method == 'GET':
prod_id = request.GET['prod_id']
c = Cart.objects.get(Q(product=prod_id) & Q(user=request.user))
c.delete()
amount = 0.0
shipping_amount = 70.0
cart_product = [p for p in Cart.objects.all() if p.user == request.user]
for p in cart_product:
```

```
tempamount = (p.quantity * p.product.discounted_price)
amount += tempamount
data = {
'amount': amount,
'totalamount': amount + shipping_amount
}
return JsonResponse(data)
## Payment
@method_decorator(login_required, name='dispatch')
def payment_done(request):
user = request.user
custid = request.GET.get('custid')
customer = Customer.objects.get(id=custid)
cart = Cart.objects.filter(user=user)
for c in cart:
OrderPlaced(user=user, customer=customer, product=c.product,
quantity=c.quantity).save()
c.delete()
return redirect("orders")
## Buy Now
def buy_now(request):
return render(request, 'app/buynow.html')
## Address view
def address(request):
add = Customer.objects.filter(user=request.user)
return render(request, 'app/address.html',{'add':add,'active':'btn-primary'})
## Order view@login_required
def orders(request):
op = OrderPlaced.objects.filter(user=request.user)
return render(request, 'app/orders.html', {'order_placed':op})
## Change password
def change_password(request):return render(request,
'app/changepassword.html')
## Mobile
def mobile(request,data=None):
if data == None:
mobiles = Product.objects.filter(category='M')
```

```
elif data == 'Redmi' or data == 'Samsung':
mobiles = Product.objects.filter(category='M').filter(brand=data)
elif data == 'below':
mobiles =
Product.objects.filter(category='M').filter(discounted_price___lt=10000)
elif data == 'above':
mobiles =
Product.objects.filter(category='M').filter(discounted_price___gt=10000)
return render(request, 'app/mobile.html',{'mobiles':mobiles})
## Laptop view
def laptop(request, data=None):
if data == None:
laptops = Product.objects.filter(category='L')
elif data == 'Dell' or data == 'HP':
laptops = Product.objects.filter(category='L').filter(brand=data)
elif data == 'below':
laptops =
Product.objects.filter(category='L').filter(discounted_price___lt=50000)
elif data == 'above':
laptops =
Product.objects.filter(category='L').filter(discounted_price___gt=50000)
return render(request, 'app/laptop.html', {'laptops': laptops})
## Top Wear
class TopWearView(View):
def get(self, request, data=None):
if data is None:
topwears = Product.objects.filter(category='TW')
elif data in ['Nike', 'Adidas']:
topwears = Product.objects.filter(category='TW', brand=data)
elif data == 'below':
topwears = Product.objects.filter(category='TW',
discounted_price___lt=1000)
elif data == 'above':
topwears = Product.objects.filter(category='TW',
discounted_price___gt=1000)
return render(request, 'app/topwear.html', {'topwears': topwears})
## Bottom Wear
```

```
def bottomwear(request, data=None):if data == None:
bottomwears = Product.objects.filter(category='BW')
elif data == "Levis" or data == 'Wrangler':
bottomwears = Product.objects.filter(category='BW').filter(brand=data)
elif data == 'below':
bottomwears =
Product.objects.filter(category='BW').filter(discounted_price__lt=1500)
elif data == 'above':
bottomwears =
Product.objects.filter(category='BW').filter(discounted_price__gt=1500)
return render(request, 'app/bottomwear.html', {'bottomwears':
bottomwears})
## Shoes
def shoes(request, data=None):
if data == None:
shoes = Product.objects.filter(category='S')
elif data == 'Nike' or data == 'Adidas':
shoes = Product.objects.filter(category='S').filter(brand=data)
elif data == 'below':shoes =
Product.objects.filter(category='S').filter(discounted_price__lt=2000)
elif data == 'above':
shoes =
Product.objects.filter(category='S').filter(discounted_price__gt=2000)
return render(request, 'app/shoes.html', {'shoes': shoes})
## Customer registration
class CustomerRegistrationView(View):
def get(self,request):
form = CustomerRegistrationForm()
return render(request, 'app/customerregistration.html', {'form':form})
def post(self,request):
form = CustomerRegistrationForm(request.POST)
if form.is_valid():
messages.success(request, 'Congratulations!! Registered Successfully')
form.save()
return render(request, 'app/customerregistration.html', {'form':form})
## Checkout Page
@login_required
```

```
def checkout(request):
user = request.user
add = Customer.objects.filter(user=request.user)
cart_items = Cart.objects.filter(user=user)
amount = 0.0
shipping_amount = 70.0
total_amount = 0.0
cart_product = [p for p in Cart.objects.all() if p.user == request.user]
if cart_product:
for p in cart_product:
tempamount = (p.quantity * p.product.discounted_price)
amount += tempamount
total_amount = amount + shipping_amountreturn render(request,
'app/checkout.html',{'add':add,
'totalamount':total_amount,'cart_items':cart_items})
## Profile View
@method_decorator(login_required, name='dispatch')
class ProfileView(View):
def get(self,request):
form = CustomerProfileForm()
return render(request, 'app/profile.html', {'form':form,
'active':'btn-primary'})
def post(self,request):
form = CustomerProfileForm(request.POST)
if form.is_valid():
usr = request.user
name = form.cleaned_data['name']
locality = form.cleaned_data['locality']
city = form.cleaned_data['city']
state = form.cleaned_data['state']
zipcode = form.cleaned_data['zipcode']
reg =
Customer(user=usr,name=name,locality=locality,city=city,state=state,zipcod
e=zipcode)
reg.save()
messages.success(request, 'Congratulations!! Profile Updated Successfully')
```

```
return render(request, 'app/profile.html',
{'form':form,'active':'btn-primary'})
```

Urls.py :

```
from django.urls import path
from . import views
from django.conf import settings
from django.conf.urls.static import static
from django.contrib.auth import views as auth_views
from .forms import LoginForm, MyPasswordChangeForm,
MyPasswordResetForm,
MySetPasswordForm
urlpatterns = [
# Registration
path('registration/',views.CustomerRegistrationView.as_view(),
name='customerregistration'),
# Authentication
path('accounts/login/',
auth_views.LoginView.as_view(template_name='app/login.html',
authentication_form=LoginForm), name='login'),
path('logout/',
auth_views.LogoutView.as_view(next_page='/accounts/login/'),
name='logout'),
path('changepassword/',
auth_views.PasswordChangeView.as_view(template_name='app/changepass
word.html',
form_class=MyPasswordChangeForm,
success_url='/passwordchangedone/'), name='changepassword'),
path('passwordchangedone/',
auth_views.PasswordChangeDoneView.as_view(template_name='app/passw
ordchangedone.html'),
name='passwordchangedone'),path('password-reset/',
auth_views.PasswordResetView.as_view(template_name='app/password_re
set.html',
form_class=MyPasswordResetForm), name='password_reset'),
path('password-reset/done/',
auth_views.PasswordResetDoneView.as_view(template_name='app/passwor
d_reset_done.html'),
name='password_reset_done'),
path('password-reset-confirm/<uidb64>/<token>/',
auth_views.PasswordResetConfirmView.as_view(template_name='app/pass
word_reset_confirm.ht
ml', form_class=MySetPasswordForm), name='password_reset_confirm'),
path('password-reset-complete/',
auth_views.PasswordResetCompleteView.as_view(template_name='app/pas
sword_reset_complete
.html'), name='password_reset_complete'),
# Home and product details
path('', views.ProductView.as_view(), name='home'),
path('product-detail/<int:pk>',
views.ProductDetailView.as_view(), name='product-detail'),
# Cart operations
path('add-to-cart/', views.add_to_cart, name='add-to-cart'),
path('cart/', views.show_cart, name='showcart'),
path('pluscart/', views.plus_cart, name='pluscart'),
path('minuscart/', views.minus_cart, name='minuscart'),
path('removecart/', views.remove_cart, name='removecart'),
# Purchase and profile
path('buy/', views.buy_now, name='buy-now'),
path('profile/', views.ProfileView.as_view(), name='profile'),
path('address/', views.address, name='address'),
path('orders/', views.orders, name='orders'),
# Product categories
path('mobile/', views.mobile, name='mobile'),
path('mobile/<slug:data>', views.mobile, name='mobiledata'),
path('laptop/', views.laptop, name='laptop'),
path('laptop/<slug:data>', views.laptop, name='laptopdata'),
path('topwear/', views.TopWearView.as_view(), name='topwear'),
path('topwear/<slug:data>',views.TopWearView.as_view(),
name='topweardata'),
path('bottomwear/', views.bottomwear, name='bottomwear'),
path('bottomwear/<slug:data>',
```

References :

Books:

1. Django for Professionals: Production websites with Python & Django – William S. Vincent.
2. Two Scoops of Django 3.x: Best Practices for the Django Web Framework – Daniel Roy Greenfeld & Audrey Roy Greenfeld.
3. Python Crash Course: A Hands-On, Project-Based Introduction to Programming – Eric Matthes.

Websites:

1. Official Django Documentation: https://docs.djangoproject.com/
2. W3Schools Django Tutorial: https://www.w3schools.com/django/
3. Stack Overflow for troubleshooting and community discussions: https://stackoverflow.com/
4. Real Python tutorials for Django: https://realpython.com/tutorials/django/

Research Articles & Papers:

1. Kula, J. S., et al. "Efficient E-commerce Platforms: Django as a Backend Framework." Journal of Web Development (2020).
2. Kumar, R., et al. "Optimizing Online Shopping Platforms Using Modern Frameworks." International Journal of Computer Applications (2022).

Online Resources:

1. GitHub Django Projects Repository: https://github.com/topics/django
2. Mozilla Developer Network (MDN) Web Docs: https://developer.mozilla.org/

Standards and Guidelines:

1. OWASP Secure Coding Practices for Web Applications: https://owasp.org/