TY B.Tech Information Technology

VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY, PUNE

Green Software Development and Sustainability in IT

IT DEPARTMENT

By,

NAME	ROLL NO	PRN NO
Tanishq Chavan	333013	22111203

CLASS: Third Year Engineering DIVISION:TY-C

Table of Contents

1. Introductory Chapter.....
1.1 Introduction.....
1.2 Scope.....
2. Specific Requirements.....
2.1 Functional Requirements.....
2.1.1 Login / Registration
2.1.2 Add to Cart
2.1.3 Receive Order
2.1.4 Managing products
2.1.5 Managing accounts
2.2 Database Requirements.....
2.3 Non- Functional Requirements.....
2.3.1 Security
2.3.2 Maintainability
2.3.3 Reliability
2.3.4 Safety Requirements
2.4 Use Case
2.4.1 Buyer
2.4.2 Seller
3. Conclusion.....
4. References.....

1 Introduction

1.1 Introduction

An e-commerce website is like an online store where you can buy and sell things. It's a place on the internet where businesses and customers meet to do shopping electronically. Instead of going to a physical store, you use your computer or phone to browse through products, pick what you want, and then make a purchase online.

1.2 Scope

We want to create a website called the E-Commerce website that acts like an online store for Electronics. This website will let people buy their gadgets without going to a physical store. You can easily pick what you need from the website, and the products will be delivered to your doorstep. The main goals are to make shopping convenient, provide home delivery, and have tools to analyze sales for better business profits. The things the website must do (functional) and the way it should work (non-functional) are all written in this document.

2 Specific Requirements

2.1 Functional Requirement

2.1.1 Login / Signup

Users, whether new or returning, can easily register on our online platform with their credentials for login convenience, and once registered, they can log in by entering their credentials.

2.1.2 Add to Cart

Users have the flexibility to explore, add, or remove products in their cart, which serves as a convenient tool for tracking their selected items on our platform.

2.1.4 Managing products

Admin have the capability to add, edit, or delete products by efficiently managing their inventory.

2.1.5 Managing accounts

The administrator, seller, and user have the capability to modify their profile details seamlessly on the platform.

2.2 Database Requirements

1. Profile: Database of all the users accessing the system that includes their information like email id, password, phone number and other details

2. Order: It is the database of all the order the site or the administrator has

3. Shopping Cart: Contains the products that the customer is planning to purchase. Entities can be added to or removed from the cart.

4. Product Database: The database will contain information regarding product availability, stock etc.

2.3 Non-Functional Requirements

2.3.3 Security Requirement: Implement robust encryption and authentication measures to ensure the confidentiality and integrity of user data in the e-commerce platform.

2.3.2 Maintainability Requirement: The e-commerce system should have clear documentation and modular design to facilitate easy updates and modifications, ensuring long-term maintainability.

2.3.3 Reliability Requirement: The platform must have a robust infrastructure and failover mechanisms to guarantee high availability and reliability, minimizing downtime for users.

2.3.4 Safety Requirement: Ensure secure payment gateways and adherence to industry safety standards to safeguard user financial information and transactions on the e-commerce platform.

2.4 Technology Used

2.4.1 Frontend (HTML & CSS):

- Create HTML templates for different pages of your e-commerce website (e.g., home page, product listing page, product detail page, cart page, checkout page, etc.).
- Use CSS to style these HTML templates to ensure a visually appealing and responsive design.

2.4.2 Backend (Flask):

- Set up a Flask application to handle routing and request handling.
- Define routes for different functionalities such as displaying product listings, handling user authentication and authorization, managing shopping cart, processing orders, etc.
- Integrate Flask with MongoDB to perform database operations (CRUD operations) such as retrieving product information, storing user data, updating orders, etc.

2.4.3 Database (MongoDB):

- Design and create MongoDB collections to store data related to your e-commerce application (e.g., products, users, orders, etc.).
- Use PyMongo (the official MongoDB driver for Python) to interact with MongoDB from your Flask application. This includes querying the database to retrieve and manipulate data as needed by your application.

2.4.4 Advantage of HTML/CSS/Flask/MongoDB:

- Simplicity: This stack offers a straightforward setup for developers comfortable with Python. Flask provides a simple backend framework, and MongoDB allows flexible data modeling.

- Full-Stack with Python: Ideal for maintaining consistency with Python across frontend (Flask templates) and backend (Flask server) development.

- Flexibility: MongoDB's NoSQL design is adaptable for evolving data needs, suitable for smaller-scale projects or unique requirements.

-Scalability: MongoDB supports horizontal scaling, beneficial for accommodating growth in data volume and complexity.

3 Specific Requirements

3.1 Buyer:

Product Exploration:

- Search and Discovery: Buyers can search for specific electronic devices or explore categories like smartphones, laptops, cameras, and more.
- Feature Comparison: Compare specifications, features, and customer reviews to make informed decisions.

Order Placement:

- Cart Management: Add electronics to the shopping cart, manage quantities, and review the selected items before proceeding to checkout.
- Secure Checkout: Complete the purchase using secure payment methods, with options for different payment gateways.

Product Assistance:

- Product Details: Access detailed information about each electronic product, including technical specifications and compatibility.
- Customer Support: Engage with customer support for pre-purchase inquiries, technical assistance, and troubleshooting.

3.2 Seller:

Product Listing:

- Detailed Listings: Create comprehensive product listings with high-quality images, detailed specifications, and pricing information.
- Inventory Management: Keep track of stock levels, restock products, and manage inventory efficiently.

4. External Interface Specification

4.1 User-Friendly Look

- Easy to Fit Any Screen:
- Looks good on different devices, thanks to tools like Bootstrap or Flexbox.
- Easy for both users and admins to find their way around.

4.2 Safe Talking

- Keep Data Safe with HTTPS:
- Make sure the info going back and forth is encrypted.
- Confirm a secure connection with SSL certificates.
- Talking to Servers in a Simple Way:
- Design how the website talks to servers in an easy-to-understand way.
- Write down the rules for how to ask for things and what to expect back.

4.3 Talking to Different Machines

- Friends with Most Browsers:
- Works well with popular browsers like Chrome,Firefox,Safari, and Edge.
- Phone-Friendly Talking:
- Looks good and works well on phones and other small screens.

4.4 Safe Storage Space

- Storing Data with MongoDB: - Uses MongoDB to store info in a flexible way.
-Regularly saves a copy to avoid losing anything important.

5 Conclusion

The website combines the power of HTML and CSS for beautiful, interactive design, while Flask handles the backend with efficiency and flexibility. Together, they create a seamless user experience. Explore our site, built with these technologies, and see how we blend style with functionality to deliver a standout web experience."

This conclusion highlights the key technologies used (HTML, CSS, Flask) and emphasizes their role in creating an effective and engaging website. Feel free to modify it to better fit your specific project or website!