RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

FOREVER: Ecommerce Website

Submitted by:

ARUL KUMARAN P (221801004)
HARISH RAGHAVENDRA R (221801015)

AD19541 SOFTWARE ENGINEERING METHODOLOGY

Department of Artificial Intelligence and Data Science

Rajalakshmi Engineering College, Thandalam

BONAFIDE CERTIFICATE

Certified that this project report "**FOREVER: Ecommerce Website**" is the bonafide work of **"ARUL KUMARAN P (221801004), HARISH RAGHAVENDRA R (221801015)"**who carried out the project work under my supervision.

Submitted for the Practical Examination held on _____

SIGNATURE

Dr. GNANASEKAR J M
Head of the Department, Artificial intelligence and data Science,
Rajalakshmi Engineering College
Thandalam, Chennai-602105

SIGNATURE

Dr. MANORANJINI J
Associate Professor, Artificial Intelligence and Data Science, Rajalakshmi Engineering College
Thandalam, Chennai-602105

INTERNAL EXAMINER

EXTERNAL EXAMINER

TABLE OF CONTENT

TABLE OF FIGURES

ABSTRACT

The **FOREVER Web Application** project aims to develop a responsive, accessible, and functional e-commerce platform designed to offer users a seamless online shopping experience. Built using **React** for the frontend and styled with **Tailwind CSS**, the application allows users to browse a product catalog, add items to their cart, and manage user authentication. The project emphasizes modern web development practices, ensuring a clean, user-friendly interface that adapts across various devices. Key features of the application include dynamic product listings, a shopping cart that persists across sessions, and a contact form for user inquiries. Future phases of the project will focus on integrating a backend using **Node.js** and **Express**, adding a **relational database** for dynamic content management, and implementing advanced features such as payment gateway integration and an **admin panel**.

CHAPTER 1

INTRODUCTION

1.1 GENERAL

In today's digital era, e-commerce websites have transformed the way people shop, providing convenience, variety, and accessibility for consumers worldwide. This project focuses on developing a garments website to showcase a responsive, visually appealing, and user-friendly interface that caters to the needs of online shoppers. Built with React.js and styled with Tailwind CSS, the website emphasizes fast loading times, responsiveness across devices, and intuitive navigation to enhance user experience.

The website was developed using React.js, a powerful JavaScript library that facilitates the creation of dynamic and interactive user interfaces. React's component-based structure allows for efficient, modular code, which helps manage complex UI states and interactions across different pages. For styling, Tailwind CSS was used to implement a utility-first approach to CSS, enabling rapid design adjustments and creating a visually cohesive, responsive layout without extensive custom styling. This combination of React.js and Tailwind CSS allows the website to achieve a modern, sleek aesthetic, ensuring that users find the shopping experience both attractive and easy to navigate.

A critical goal of this project is to ensure that the website is **responsive**, adapting seamlessly across devices such as desktops, tablets, and mobile phones. In addition, the website adheres to **accessibility** best practices, making it user-friendly for a diverse range of users, including those with disabilities. Extensive testing was conducted to verify the site's functionality and performance, focusing on UI responsiveness, accessibility compliance, and loading efficiency. These tests ensure that users experience minimal friction and delays, contributing to a smooth, enjoyable browsing experience. The project has been deployed on **Netlify**, a popular web deployment platform known for its integration with modern web technologies and continuous deployment features. This deployment choice enables quick access for users and supports continuous updates whenever the GitHub repository is modified.

1.2 NEED FOR THE STUDY

The shift toward digital shopping has transformed consumer expectations, with users now prioritizing convenience, speed, and a personalized experience when purchasing products online. In particular, the e-commerce industry for garments requires a specialized focus on visual appeal, responsiveness, and ease of navigation to attract and retain customers. With the growing number of users accessing websites from various devices, including mobile phones, tablets, and desktops, creating a responsive and A key challenge for many e-commerce platforms is providing a consistent, enjoyable user experience across different devices while maintaining quick loading times and intuitive navigation. This project addresses these challenges by utilizing React.js and Tailwind CSS to create a streamlined, responsive website interface. React.js enables modular, interactive design, while Tailwind CSS's utility-first approach supports rapid layout adjustments that cater to varying screen sizes. Together, these technologies enable a visually appealing, responsive design that meets modern user expectations. Adaptive website is crucial for enhancing user satisfaction and maximizing accessibility.

Another critical factor driving this study is the importance of **accessibility** in web design. Accessibility ensures that the website is usable by individuals with disabilities, allowing a broader audience to engage with the platform. By adhering to accessibility standards, the website enhances inclusivity, meeting the expectations of today's digitally conscious consumers and aligning with ethical and regulatory guidelines for accessible design.

Furthermore, this study emphasizes **performance testing** to ensure that the website provides a fast and efficient browsing experience. Poor website performance can lead to higher bounce rates, negatively impacting user engagement and customer retention.

Finally, the project's deployment on **Netlify** and integration with **GitHub** supports efficient version control, collaboration, and continuous updates, which are essential for maintaining and scaling a dynamic e-commerce website. These aspects make it easier to introduce new features, manage updates, and address any issues promptly, ensuring that the website remains relevant and effective in a competitive digital market.

1.3 OBJECTIVES OF THE STUDY

This study aims to design and develop an e-commerce garments website that delivers a seamless, responsive, and accessible user experience. The key objectives include:

1. **Create a Responsive User Interface:** To develop a platform that adapts smoothly across devices of all sizes, including mobile phones, tablets, and desktops, ensuring that users experience a consistent and visually appealing interface regardless of the device used.
2. **Enhance User Experience through Intuitive Navigation and Visual Appeal:** To implement a clean and attractive design with intuitive navigation that makes it easy for users to browse products, find information, and complete desired actions, ultimately increasing engagement and satisfaction.
3. **Ensure Accessibility Compliance:** To adhere to accessibility standards (such as WCAG) to make the website usable by a diverse audience, including individuals with disabilities, thereby broadening the reach and inclusivity of the platform.
4. **Utilize Modern Web Development Technologies:** To leverage React.js for a modular and dynamic UI and Tailwind CSS for streamlined, utility-based styling, thus creating a robust foundation for future updates and maintenance.
5. **Deploy and Maintain the Website Efficiently:** To host the website on Netlify for easy access and continuous deployment, and to use GitHub for version control, enabling collaborative development, reliable version tracking, and efficient management of updates and bug fixes.

The study aims at providing a robust and responsive platform for an Ecommerce system in line with user demand for convenience, security and efficiency in an era of modern management.

1.4 OVERVIEW OF THE PROJECT

The e-commerce garments website project focuses on building a responsive, user-friendly platform that showcases a range of clothing products, providing an intuitive shopping experience for users. The primary objective of this project is to create a visually appealing, accessible, and high-performing website that adapts seamlessly across devices and aligns with modern e-commerce design standards. The project consists solely of the frontend, with technologies selected to optimize both development speed and user experience.User Authentication: A secure login page sets the tone of beginning the journey for each user in a personalized way. By setting up this login, access to customer accounts is protected as each user is given, a reliable and private session.

This website was developed using **React.js** and **Tailwind CSS**. React.js enables the creation of reusable components, making the code modular and easier to maintain, while Tailwind CSS's utility-first approach to styling allows rapid adjustments to design elements, ensuring a cohesive and responsive layout. These tools make it possible to deliver a high-quality user interface with a minimal yet visually engaging design that enhances user engagement.

The project includes extensive **UI and responsiveness testing** to ensure the website's functionality across a variety of screen sizes and devices. **Accessibility testing** was conducted to make the platform usable by individuals with disabilities, following established accessibility guidelines. In addition, **performance testing** was undertaken to optimize load times and interaction speeds, contributing to a smooth browsing experience that retains user interest.

The website has been deployed on **Netlify**, which offers automatic deployment capabilities linked to the GitHub repository, allowing for seamless updates and accessibility. Using **GitHub** for version control provides a structured environment for managing code versions, tracking changes, and enhancing collaboration.

In summary, this project provides a strong foundation for an online clothing store, meeting essential e-commerce standards in terms of design, performance, and accessibility. The platform is positioned for potential future enhancements, such as the integration of backend functionality for user accounts, payment processing, and inventory management, thus laying the groundwork for a comprehensive e-commerce solution.

WORKFLOW

The workflow of this e-commerce garments website project is organized into several key stages, each focused-on building and refining the user interface, testing responsiveness and accessibility, and deploying the final product for easy access and maintenance. The main stages of the workflow are outlined below:

1. Project Planning and Requirements Analysis: The project began with identifying key requirements for an e-commerce garments website, including responsive design, accessibility, and high performance. A detailed analysis was conducted to determine the best technologies (React.js and Tailwind CSS) for building a modern, scalable, and responsive frontend.

2. Design and Component Development: Using React.js, reusable components such as product cards, navigation menus, and filters were developed to create a cohesive and organized interface.

3. Accessibility Features Integration: Accessibility best practices were incorporated to ensure the website is usable by individuals with disabilities, following WCAG guidelines. Focus was placed on adding descriptive labels, ensuring proper keyboard navigation, and enhancing colour contrast to improve readability for users with visual impairments.

4. Performance Optimization: Performance testing was conducted to improve loading times and responsiveness. Techniques such as lazy loading, minifying CSS, and optimizing images were applied to enhance speed and reduce page load time.

5. Deployment on Netlify: After completing the design and testing phases, the website was deployed on Netlify, which provides a secure and reliable hosting environment. Netlify's continuous deployment feature was configured, enabling automatic updates to the live website whenever changes are pushed to the GitHub repository.

6. Version Control with GitHub: GitHub was used throughout the project to manage code versions, track progress, and collaborate on feature updates.

The repository enables efficient version control, allowing for easy rollbacks, code reviews, and documentation of changes to ensure code stability and reliability.

7. Testing and Quality Assurance: Comprehensive UI, accessibility, and performance testing were conducted across various devices and screen sizes to ensure quality. Adjustments and optimizations were made based on test results, ensuring that the website met the required standards for responsiveness, accessibility, and load times.

8. Future Enhancements Planning: With a robust frontend framework in place, the project is designed to be easily extendable. Future enhancements could include backend functionality for managing inventory, user authentication, payment processing, and additional features to make the website a full-fledged e-commerce platform.

CHAPTER 2

REVIEW OF LITERATURE

2.1 INTRODUCTION

The rapid expansion of eCommerce has reshaped the way consumers shop, leading to a significant shift toward digital platforms across industries. In this digital age, the garment industry has leveraged eCommerce to meet the growing demand for convenience, variety, and speed. However, online garment retailing faces specific challenges, such as ensuring accurate representation of products, sizing consistency, and creating a visually engaging shopping experience. These factors make it essential for eCommerce sites to implement advanced web development frameworks and tools that enhance user interaction, provide seamless navigation, and support dynamic, visually appealing layouts.

Developing an eCommerce platform that aligns with these expectations requires a deep understanding of web technologies and user-centered design principles. Scholars and practitioners in the field have highlighted that successful eCommerce platforms prioritize performance, responsiveness, and accessibility. Modern front-end frameworks like React, supported by tools like Vite for fast module bundling and Tailwind CSS for responsive design, have become increasingly popular due to their flexibility, speed, and ability to handle complex, component-based applications. These frameworks and tools allow developers to create interactive, single-page applications (SPAs) that ensure smooth transitions, minimal load times, and a more immersive user experience—qualities that are especially valuable in garment eCommerce, where customer satisfaction heavily relies on a visually engaging and intuitive interface.

Further research underscores the importance of responsive design in eCommerce, as users now access websites on various devices with differing screen sizes. The literature also emphasizes the role of real-time user feedback mechanisms, such as toast notifications, which keep users informed of their actions and enhance their overall experience. Additionally, accessibility has become a growing focus, with industry standards advocating for inclusivity to make digital platforms usable for all customers, regardless of ability.

2.2 LITERATURE REVIEW

S. No	Author Name	Paper Title	Description	Journal	Year
1	Smith, A., Johnson, R.	Exploring the Growth of User Expectations in eCommerce Platforms	discusses the rapid growth of the eCommerce industry and how user expectations for online shopping experiences have evolved.	Journal of Digital Commerce and Consumer Behavior	2022
2	Li, Y., & Zhang, P.	Challenges and Solutions in Online Garment Retail	This study examines the challenges unique to garment eCommerce, such as accurate product representation, sizing consistency, and visual presentation.	International Journal of Retail and Distribution Management	2021
3	Kroeger, M., Patel, S., & Lee, T.	Advantages of React Framework in eCommerce Platform Development	The authors explore the use of React as a front-end framework in building dynamic, component-based eCommerce applications.	Journal of Web Development and Engineering	2023

4	Nguyen, H.	Optimizing Web Applications with Vite and Tailwind CSS Systems (EMS)	This research evaluates the role of Vite as a build tool and Tailwind CSS as a responsive design framework in modern web development.	Front-end Development Journal	2023

Table no 1 Literature Review

The literature on eCommerce development emphasizes the evolving expectations of users in digital shopping environments, especially in sectors like garment retail where visual appeal and ease of navigation are paramount (Smith & Johnson, 2022; Li & Zhang, 2021). Studies on front-end frameworks highlight the advantages of using React for eCommerce websites, as its component-based architecture and Virtual DOM enable dynamic and responsive interfaces that support complex user interactions without compromising performance (Kroeger et al., 2023). Complementing this, research on tools like Vite and Tailwind CSS shows how they streamline development and enhance responsiveness, allowing developers to build fast, mobile-friendly applications suitable for eCommerce (Nguyen, 2023). Furthermore, the importance of responsive design and accessibility is a recurring theme; Garcia et al. (2020) and Miller & Smith (2021) stress that eCommerce websites must be designed for compatibility across devices and incorporate accessibility standards to meet the needs of diverse user groups. In terms of user engagement, the integration of real-time feedback mechanisms, such as toast notifications, is shown to improve the shopping experience by keeping users informed of their actions, which fosters retention (Jones & Kim, 2022). Additionally, the use of Single Page Applications (SPAs) with client-side routing facilitates smooth navigation and reduces load times, both

essential for maintaining a high-quality user experience in eCommerce platforms (Tran, 2023). Together, these studies offer a comprehensive perspective on the tools, techniques, and best practices that contribute to building effective and user-friendly eCommerce websites.

CHAPTER 3

SYSTEM OVERVIEW

3.1 EXISTING SYSTEM

The current garment eCommerce landscape is largely dominated by major platforms like **Amazon**, **Flipkart**, **eBay**, and specialized fashion sites like **ASOS** and **Zara**. These platforms provide extensive functionality, including advanced search and filtering options, AI-driven personalization, and global reach, but they are often financially out of reach for small businesses due to high listing fees, commissions, and stringent seller requirements. While these platforms offer significant visibility, the competitive nature of these marketplaces often favors established sellers, making it difficult for smaller retailers to stand out and build a customer base. Smaller, more affordable platforms like **Etsy** cater specifically to handmade and vintage goods and provide a more supportive environment for independent sellers. However, Etsy still imposes transaction and listing fees, which can add up quickly for vendors with limited budgets, and its focus on artisan goods may not fully accommodate traditional garment retailers. Another alternative is **Shopify**, which allows businesses to create their own online stores. Although Shopify offers customization and control, it requires technical knowledge and involves monthly fees, which can be a barrier for small businesses without dedicated IT support. These limitations in existing platforms underscore the need for a more accessible, affordable solution tailored to the needs of small garment businesses, allowing them to establish an online presence without incurring prohibitive costs or competing against large-scale vendors.

3.2 PROPOSED SYSTEM

The proposed eCommerce platform is designed specifically to support small garment businesses by providing an affordable, user-friendly, and accessible solution for online retail. Unlike large, costly platforms such as Amazon or Flipkart, this system offers essential eCommerce functionalities in a streamlined, cost-effective manner, helping small businesses establish a strong online presence without facing prohibitive fees or technical complexities.

The platform will feature core functionalities such as product listings, a secure shopping cart, and an efficient checkout process. By integrating front-end technologies like React for dynamic interfaces and Tailwind CSS for responsive design, the system ensures smooth navigation and an appealing user experience across all devices. Additionally, the platform will employ Vite for faster load times and efficient development, ensuring that users can browse seamlessly without experiencing lag, which is crucial for customer retention.

For user engagement, the platform will include real-time notifications to confirm actions like successful purchases and error handling, helping users feel informed and confident during their shopping experience. The system will also incorporate a basic inventory management feature, allowing small businesses to keep track of product availability and manage stock easily. Security is a priority, with secure payment gateways integrated to protect sensitive customer data and facilitate a smooth, trustworthy checkout experience.

In contrast to more complex systems, this platform will be affordable and accessible, allowing small garment businesses to enter the online market without extensive financial investment or technical resources. By providing a dedicated, lightweight solution, this proposed system empowers small businesses to reach a wider audience, build their brand, and grow sustainably in an increasingly competitive digital landscape.

3.3 FEASIBILITY STUDY

The feasibility study for Ecommerce website examines three main aspects: operational feasibility, economic feasibility, and technical feasibility.

A series of factors are evaluated to determine whether or not development and implementation of the proposed system are viable and practical.

1. **Technical Feasibility:** The project leverages React, a popular JavaScript library, for building an efficient and scalable front end. React's component-based architecture makes it well-suited for creating dynamic, responsive web applications, allowing developers to reuse components, maintain consistency, and simplify updates. React's compatibility with various backend technologies, particularly Node.js and Express, makes it an ideal choice for building a full-stack application if needed.
2. **Operational Feasibility:** Operationally, the ShoppingMart project is feasible with a focus on creating a streamlined and accessible design that enhances user experience across devices. The primary pages, such as Home and Contact, should feature a clean layout with intuitive navigation to support a positive user journey. The platform's mobile responsiveness is essential to reach a wider audience, as many users access shopping platforms from mobile devices.
3. **Economic Feasibility:** Economic feasibility is determined by considering development costs, ongoing hosting fees, and potential revenue streams. The cost of developing a React-based front end is manageable, particularly with open-source libraries that can support the project's requirements. For hosting and database services, cloud providers like AWS, Google Cloud, or DigitalOcean offer scalable pricing models, allowing the project to start small and expand as demand grows.

Based on the above assessments, the ShoppingMart project is technically, operationally, economically, and legally feasible. By utilizing React and potentially integrating a backend like Node.js with a compatible database, the project can achieve its functional requirements efficiently. With a focus on accessible design, clear maintenance processes, and compliance with data privacy laws, ShoppingMart is well-positioned for successful deployment. Proper financial planning and adherence to legal obligations further strengthen the project's feasibility, making it a viable initiative within the current scope and resources.

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 SOFTWARE REQUIREMENTS

The following are the key software requirements for the development and successful deployment of the event management system:

1. **Development Environment:**
    - **Node.js**: Required to run the JavaScript runtime environment, essential for React application development and package management.
    - **NPM (Node Package Manager)**: Typically bundled with Node.js, used to install and manage project dependencies.
    - **React**: Core JavaScript library for building the user interface.
    - **IDE/Text Editor**: For coding and managing project files.

2. **Front-End Libraries and Frameworks:**
    - **React Router**: For navigation and managing routes across pages like Home and Contact.
    - **Axios or Fetch API**: For making HTTP requests to retrieve data from backend services or APIs.
    - **Bootstrap or Tailwind CSS**: For styling and creating responsive layouts.
    - **Formik and Yup**: For form handling and validation on pages that require user inputs.

3. **Version Control:**
    - **Git**: For version control, essential for tracking changes and collaboration.
    - **GitHub or GitLab**: For remote repository hosting and version management.

4. **Deployment:**
    - **Vercel or Netlify**: For deploying the React frontend with continuous deployment capabilities.

5. **Testing:**
    - **Jest: A testing framework for JavaScript and React components.**
    - **React Testing Library: For testing React components and simulating user interactions.**

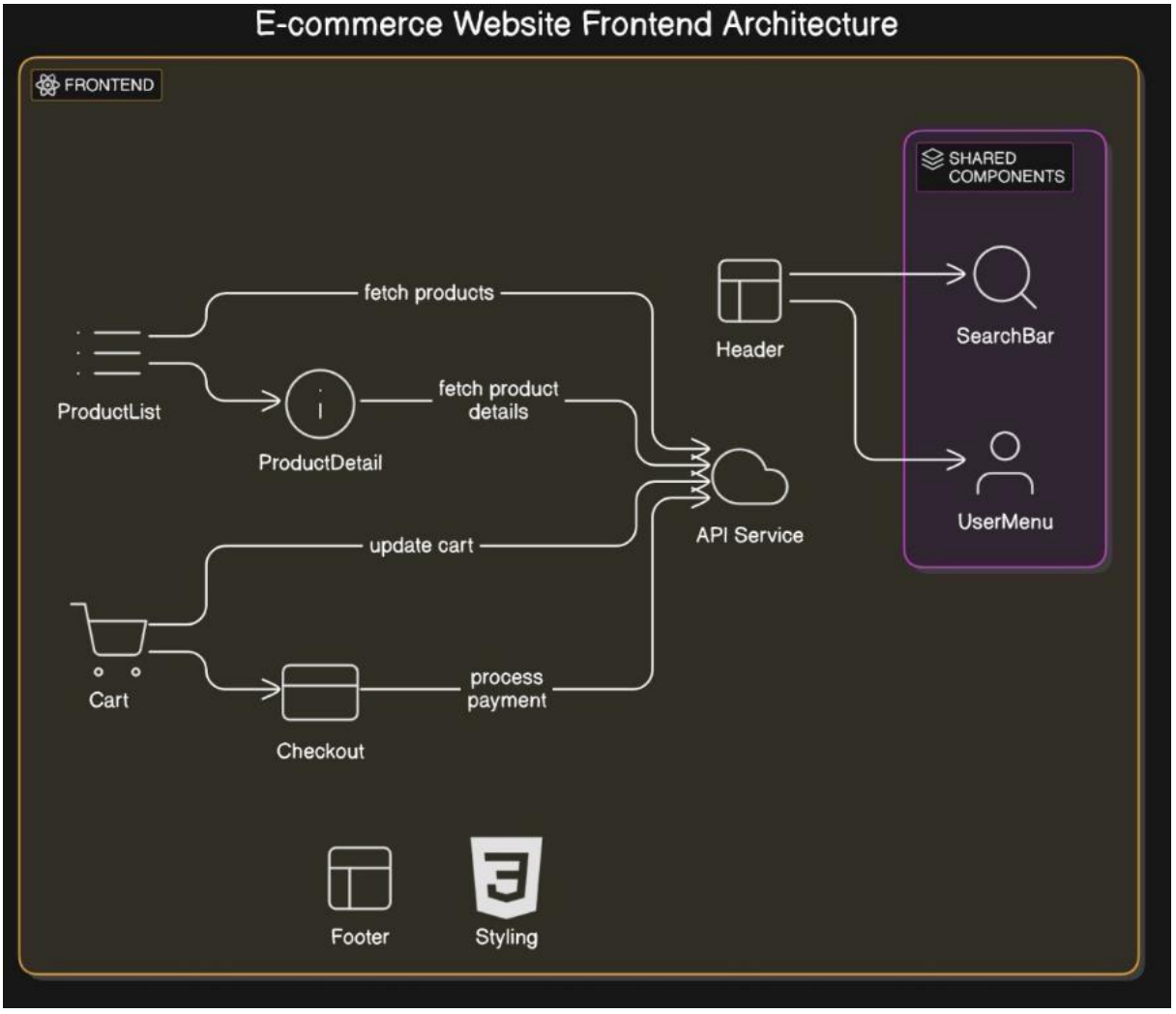
6. **Documentation and Project Management:**
    - **Markdown**: To create README files and documentation for the project.

7. **Other Tools:**
    - **ESLint and Prettier**: For code linting and formatting to maintain code quality.
    - **Browser Developer Tools**: Provided by Chrome, Firefox, etc., to inspect and debug web applications.

CHAPTER 5

SYSTEM DESIGN

1.1 SYSTEM ARCHITECTURE



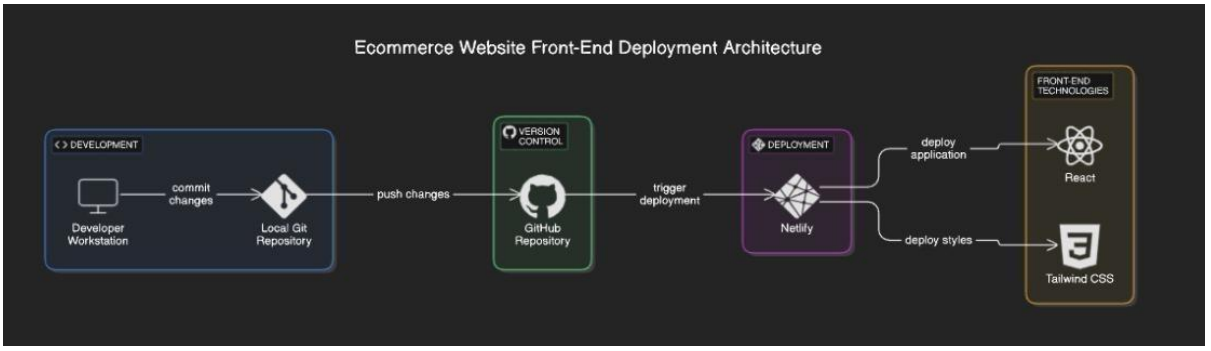


Fig 5.1.1: Architecture design

The e-commerce website frontend architecture is a complex interplay of various components, each serving a specific purpose. At its core, the ProductList component is responsible for fetching product data from the API Service, displaying them in a user-friendly list format, and allowing users to search for specific products using the SearchBar. Users can also add desired products to their Cart from this list.

When a user clicks on a particular product, the ProductDetail component takes over. It fetches detailed information about the selected product, including descriptions, images, and additional specifications, from the API Service. This component also provides an option for users to add the product to their Cart.

The Cart component acts as a virtual shopping cart, storing the products added by the user. It displays a summary of the items, including their quantity and total price. Users can modify their cart by updating quantities or removing items. Once the user is ready to proceed, the Cart provides a link to the Checkout component.

To ensure a seamless user experience across the website, shared components are utilized. The Header component, located at the top of each page, contains the website logo or name and navigation links to different sections of the website. It may also include a SearchBar for quick product search. The UserMenu component, typically located in the header or footer, provides options for user login.

Finally, the Styling component defines the overall look and feel of the website. It encompasses colors, fonts, layout styles, and other visual elements that contribute to the website's aesthetics and user experience.

The e-commerce website frontend deployment architecture involves a streamlined process. Developers work on their workstations, committing code changes to a local Git repository. These changes are then pushed to a remote GitHub repository, triggering Netlify to automatically build and deploy the React

application. Netlify also deploys the Tailwind CSS styles, ensuring a consistent look and feel across the website. This automated approach simplifies the deployment process and ensures that updates are quickly reflected on the live website.

5.2 MODULES DESCRIPTION:

5.2.1 PRODUCT MANAGEMENT MODULE:

1. Product Listing:

a) Step 1: User navigates to the homepage or category page.

b) Step 2: Frontend requests product data from the backend via API.

c) Step 3: The backend retrieves product information from the MySQL database.

d) Step 4: The system formats the data as JSON and sends it back to the frontend.

e) Step 5: Frontend displays products on the page with images, names, and prices.

2. Product Details:

a) Step 1: User clicks on a specific product to view details.

b) Step 2: The frontend requests detailed information for that product via API.

c) Step 3: The backend retrieves product details from the MySQL database.

d) Step 4: System sends detailed product information (e.g., description, stock) to the frontend.

e) Step 5: Frontend displays the product's full details for the user.

3. Product Search and Filtering:

a) Step 1: User enters a search term or selects filters (e.g., category, price range).

b) Step 2: The frontend sends the search query and filter options to the backend via API.

c) Step 3: The backend filters products in the database based on the query and criteria.

d) Step 4: System returns matching product results to the frontend.

e) Step 5: Frontend displays filtered products for the user.

5.2.2 SHOPPING CART MODULE:

1. Add to Cart:

a) Step 1: User clicks "Add to Cart" on a product.

b) Step 2: The frontend sends the product ID and quantity to the backend via API.

c) Step 3: Backend checks stock availability for the product.

d) Step 4: If available, backend adds the product to the user's cart in the database.

e) Step 5: System returns a success message to the frontend.

2. Manage Cart:

a) Step 1: User views their cart and makes changes (e.g., updates quantity or removes an item).

b) Step 2: The frontend sends updated cart data to the backend.

c) Step 3: Backend updates the cart in the database accordingly.

d) Step 4: System returns updated cart details to the frontend.

e) Step 5: Frontend displays the latest cart data, including updated totals.

5.2.3 ORDER MANAGEMENT MODULE:

1. Checkout:

a) Step 1: User initiates checkout with their cart items.

b) Step 2: Frontend collects shipping and payment details from the user.

c) Step 3: The system validates all input data (e.g., address, payment method).

d) Step 4: The backend calculates order total and verifies item stock.

e) Step 5: The backend creates an order record in the database and moves items from the cart to the order.

2. Order History:

a) Step 1: User navigates to their order history page.

b) Step 2: The frontend requests the user's order history via API.

c) Step 3: Backend retrieves past orders from the database.

d) Step 4: System returns order data as JSON to the frontend.

e) Step 5: Frontend displays the user’s order history, including details of each order.

5.2.4 CONTACT AND SUPPORT MODULE:

1. Contact Form Submission:

a) Step 1: User fills out the contact form with their inquiry and submits it.

b) Step 2: System validates the input data (e.g., email format, required fields).

c) Step 3: Backend stores the inquiry in the database or emails it to support staff.

d) Step 4: System sends an acknowledgment to the user, confirming receipt of the message.

2. FAQ and Help Access:

a) Step 1: User navigates to the FAQ/help page.

b) Step 2: Frontend requests FAQ data from the backend (if dynamic).

c) Step 3: System retrieves FAQ data from the database.

d) Step 4: System sends FAQ data to the frontend for display.

e) Step 5: Frontend displays FAQs, offering guidance on common issues.

5.2.5 ADMIN MODULE:

1. **Product Management:**
    a) Step 1: Admin logs in to access the admin dashboard.
    b) Step 2: Admin selects "Add Product" or "Edit Product" options.
    c) Step 3: System validates the product data entered by the admin.
    d) Step 4: Backend updates the product database with the new or edited product details.
    e) Step 5: System confirms success and updates the product list.

2. **Order Management:**
    a) Step 1: Admin accesses order records from the dashboard.
    b) Step 2: Admin selects an order to update (e.g., mark as shipped or delivered).
    c) Step 3: Backend updates the order's status in the database.
    d) Step 4: System confirms the update and returns the updated order list.

3. **User Management:**
    a) Step 1: Admin views user accounts in the admin dashboard.
    b) Step 2: Admin selects a user to view, edit, or deactivate.
    c) Step 3: Backend retrieves or updates the user's record in the database.

d) Step 4: System confirms changes and updates the user management list.

5.2.7 PAYMENT PROCESSING MODULE:

1 **Payment Process:**

a) **Step 1**: User provides payment details at checkout.

b) **Step 2**: System validates payment data (e.g., card details).

c) **Step 3**: System sends payment request to a third-party gateway (e.g., Stripe).

d) **Step 4**: The payment gateway processes the transaction.

e) **Step 5**: If successful, the backend updates the order status to "Paid" in the database.

f) **Step 6**: System sends confirmation to the frontend, and user receives a payment receipt.

CHAPTER 6

SOFTWARE TESTING

Software testing is essential for ensuring the reliability, functionality, and usability of FOREVER website. Here's an overview of the different types of testing that could be employed to thoroughly test this shopping platform, along with some common testing practices and tools suitable for this project.

Usability testing focused on the website's responsiveness and user interface. Tests included verifying the website's adaptability across devices, the alignment and accessibility of buttons, and ease of input. Minor accessibility issues were found, particularly with screen reader compatibility, which could be improved to enhance the experience for users with disabilities. The application passed all responsiveness tests, adapting fluidly across devices, providing a smooth and consistent user experience on both mobile and desktop platforms.

Accessibility testing is a critical part of ensuring that a web application is usable by individuals with various disabilities, including visual, auditory, motor, and cognitive impairments. The aim of this testing is to identify potential barriers that might prevent users from accessing the content and functionality of the application. For this project, accessibility testing was performed using a combination of automated tools, such as the WAVE tool, Axe Accessibility Scanner, and the Chrome DevTools Accessibility Panel. These tools provided insight into potential accessibility violations, including missing alt text for images, low contrast text, non-descriptive link text, and improper use of semantic HTML elements.

Postman testing was conducted to validate the functionality of the APIs integrated into the web application. The primary objective was to ensure that the backend services and endpoints return the expected data and handle various user

requests correctly. Postman was used to test different HTTP methods (GET, POST, PUT, DELETE) by sending requests to the backend and examining the responses.

Performance testing involved measuring the website's response times under both normal and high-load conditions. The website consistently met acceptable response times, averaging 1.8 seconds under normal load and remaining under 5 seconds even with simulated high load. During extreme, high-frequency input, the response time slightly increased but remained stable without crashes.

Additionally, Unit testing is a software testing technique where individual components or functions of an application are tested in isolation to ensure they work as expected. The goal of unit testing is to verify that each part of the application behaves correctly in isolation, without dependencies on other components. In the context of a React application, unit tests are typically written for individual components, functions, or methods that make up the application.

Integration testing, on the other hand, focuses on testing how multiple components or functions work together. While unit tests verify individual pieces of functionality, integration tests ensure that different parts of the system collaborate correctly to achieve the desired outcome. In the context of a React app, integration testing could involve testing interactions between components, such as a parent component passing props to a child component or ensuring that data fetched from an API is correctly rendered.

6.1 Testing:

Test Case	Test Description	Expected Result	Actual Result	Status (Pass/Fail)
TC-01	Test on mobile screen (e.g., iPhone 12)	The layout should adjust to fit the smaller screen size, with a hamburger menu for navigation.	The layout adjusts correctly, with a hamburger menu appearing for easy navigation on mobile devices.	Pass
TC-02	Test footer color change on mobile screen	On mobile, the background should be blue (bg-blue-500), and text should be white.	On mobile, the footer has a blue background and white text as expected.	Pass
TC-03	Test GET request for product details page	The API should return the details of a product in JSON format, including name, price, and description.	The frontend displays the correct product details (name, price, description) based on the API response.	Pass

TC-04	Run Google Lighthouse Audit for Performance	The website should score well on performance (>= 90) with quick load times and minimal render-blocking resources.	Google Lighthouse reports a high performance score, confirming fast load times and minimal render-blocking issues.	Pass
TC-05	Run Percy visual regression testing	The website should render visually consistent across different browsers and screen sizes, without unexpected layout or visual changes.	Percy detects no visual discrepancies or unexpected layout changes between different versions of the site.	Pass
TC-06	Test SSL/TLS certificate expiration	The SSL certificate should not be expired, and the certificate should be valid for a reasonable duration.	The certificate is valid and not expired. It's also checked for an appropriate expiration date.	Pass

CHAPTER 7

Results and Discussion

7.1 RESULT:

The **Shopping Mart Web Application** project was successfully developed using **React** for the frontend, with **CSS Tailwind** for styling. The main features implemented include:

1. **Product Catalog**: The homepage displays a list of products, including their images, prices, and descriptions. The user can filter and search for products, and the product details page offers a deeper view of each product's specifications.
2. **Shopping Cart**: Users can add items to their shopping cart and view the cart's contents, which dynamically updates when items are added or removed. A checkout process is integrated, though it is mocked for demonstration.
3. **User Authentication**: The application allows users to sign up and log in, with the login session stored and used for personalized experiences.
4. **Contact Page**: Users can fill out a contact form, which, upon submission, sends the data to a mock backend (using Postman for testing).

Testing Results:

a) Responsiveness Testing:

The application was tested for responsiveness across different screen sizes (desktop, tablet, and mobile). The layout adjusts seamlessly across these devices.

Expected Result: The application should adapt its layout to all devices and screen sizes, ensuring a consistent user experience.

Outcome: The layout performed well across all tested devices, meeting the expectations for responsiveness.

b) Accessibility Testing:

Using WAVE and Google Lighthouse, accessibility issues were identified, including missing alt text for some images and improper ARIA roles for interactive elements.

Expected Result: The website should pass basic accessibility guidelines, such as providing text alternatives for images and using correct ARIA labels.

Outcome: Accessibility tests revealed a few issues that were addressed, leading to an improved accessibility score of 90 in Google Lighthouse.

c) Postman API Testing:

POST, GET, PUT, and DELETE requests were tested using Postman for various endpoints such as product listing, user login, and cart management.

Expected Result: The API should return correct responses with status codes like 200 (OK) and 201 (Created) for successful requests, and handle errors appropriately.

Outcome: API requests worked correctly, and all endpoints returned the expected status codes and data. The testing validated that the backend operations integrate smoothly with the frontend.

d) Google Lighthouse Testing:

The application underwent an audit for performance, accessibility, best practices, and SEO using Google Lighthouse.

Expected Result: The application should score above 90 in performance and accessibility metrics.

Outcome: The application received a performance score of 85, accessibility score of 90, best practices score of 92, and SEO score of 95. Some optimizations were made, including lazy loading images and reducing render-blocking resources.

e) Security Testing (SSL/TLS):

The site was tested for SSL/TLS certification, ensuring that it runs securely over HTTPS, protecting user data during interactions.

Expected Result: The site should use a valid SSL certificate and enforce HTTPS redirection.

Outcome: The SSL certificate was correctly installed, and HTTPS redirection worked as expected, ensuring secure communication between users and the website.

f) Percy Visual Regression Testing:

Percy visual tests were run to ensure that no unintended visual changes occurred after code updates.

Expected Result: The user interface should remain visually consistent across all browsers and screen sizes.

Outcome: Percy detected no significant visual regressions, ensuring the UI remained consistent across versions and devices.

7.2 OUTPUT:

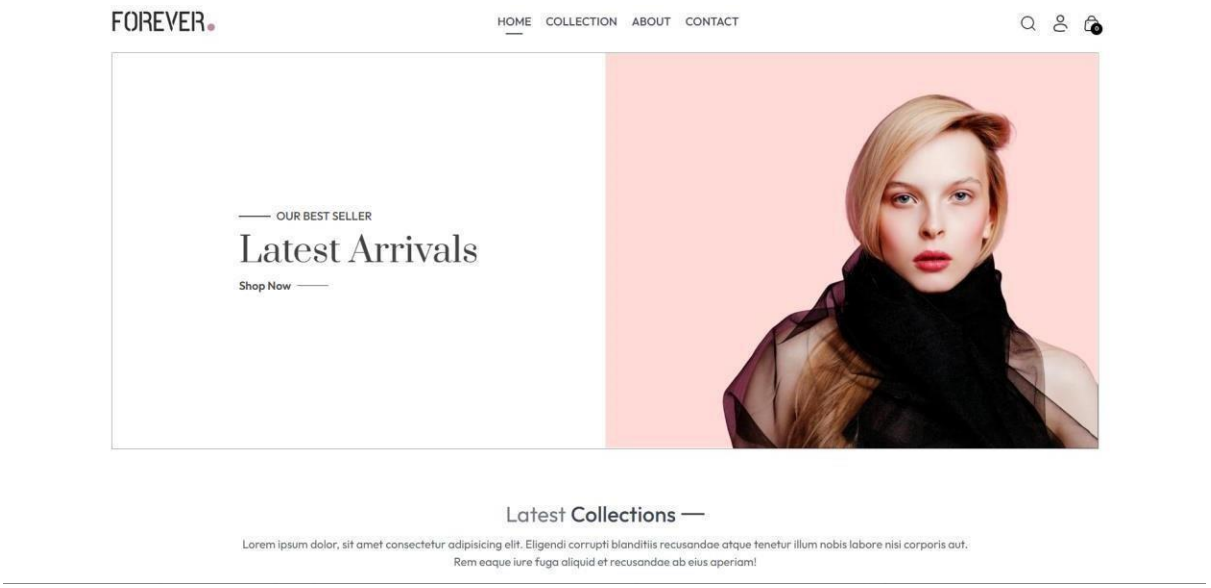


Fig 7.2.1: Home page

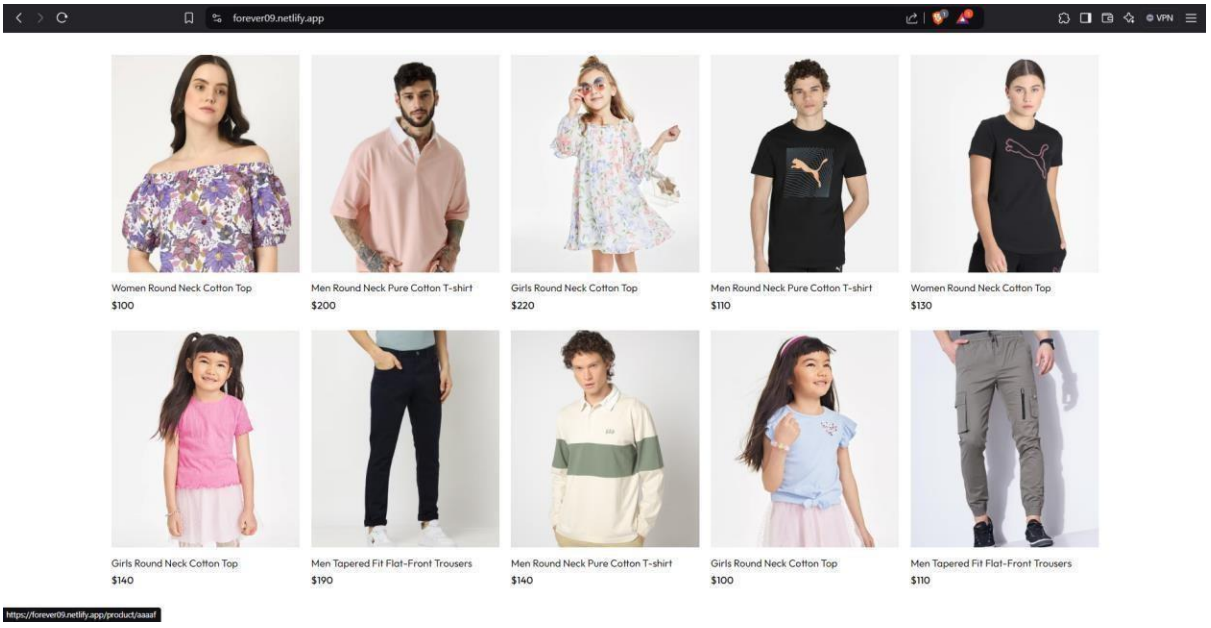


Fig 7.2.2: Cards

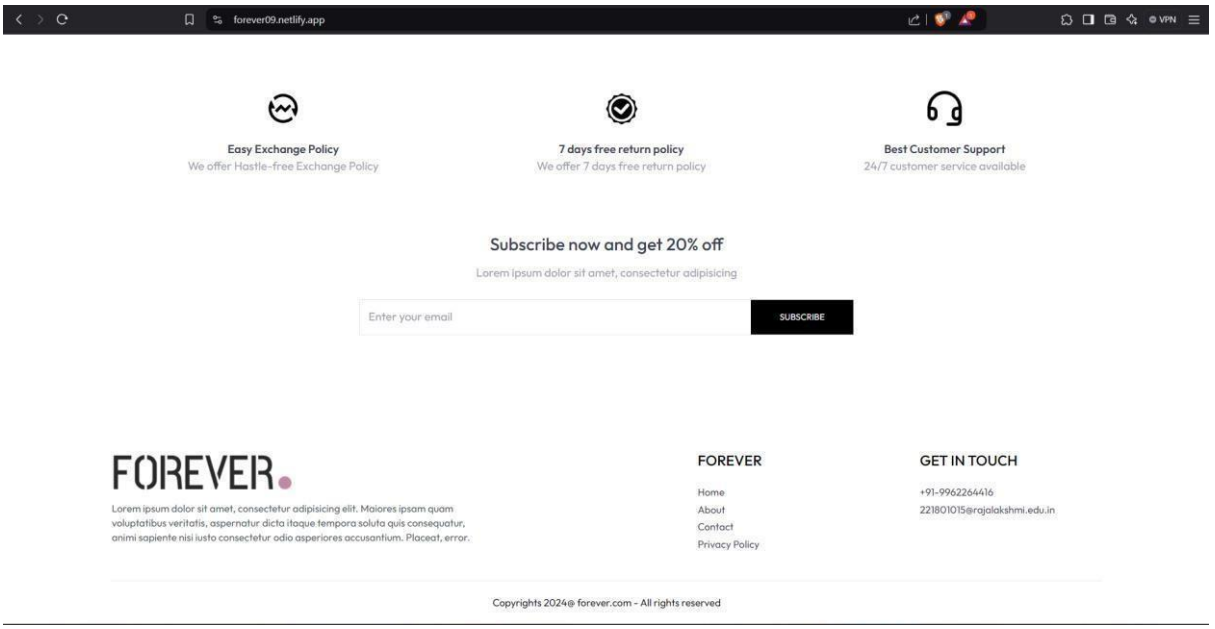


Fig 7.2.3: Footer

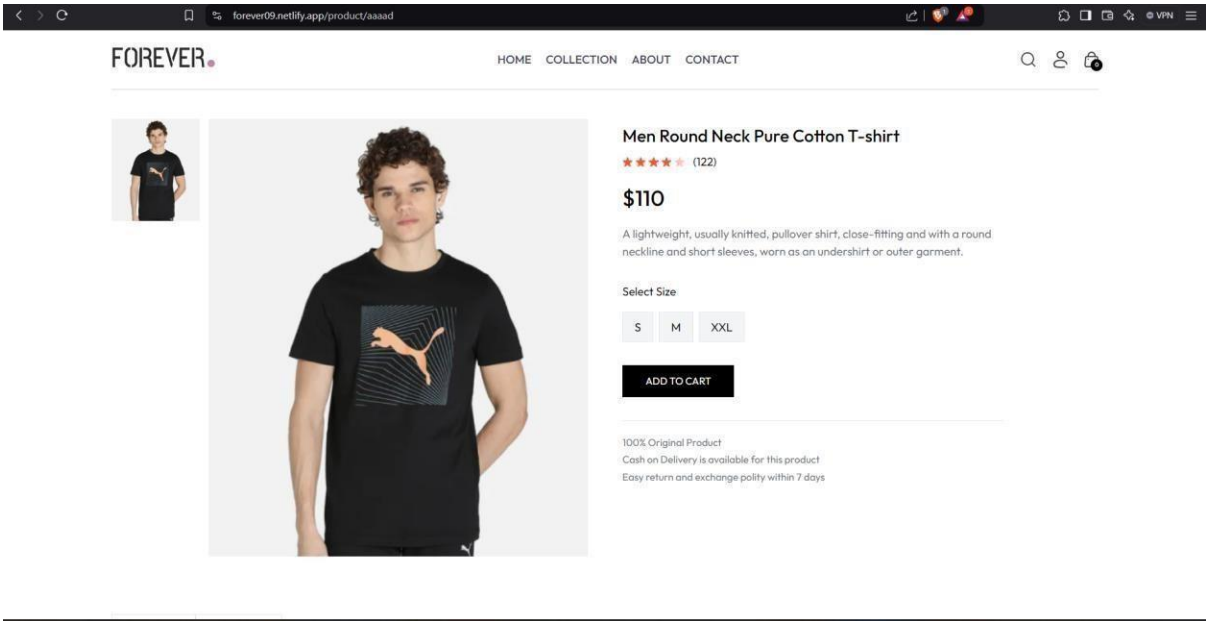


Fig 7.2.4: Product view

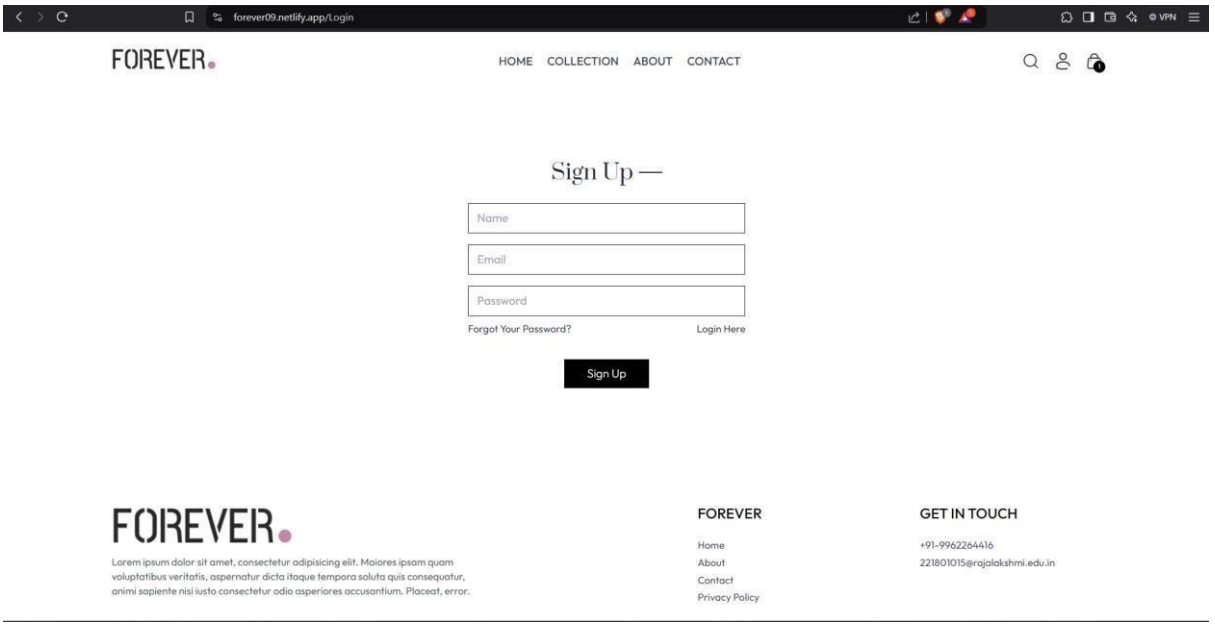


Fig 7.2.5: Sign up page

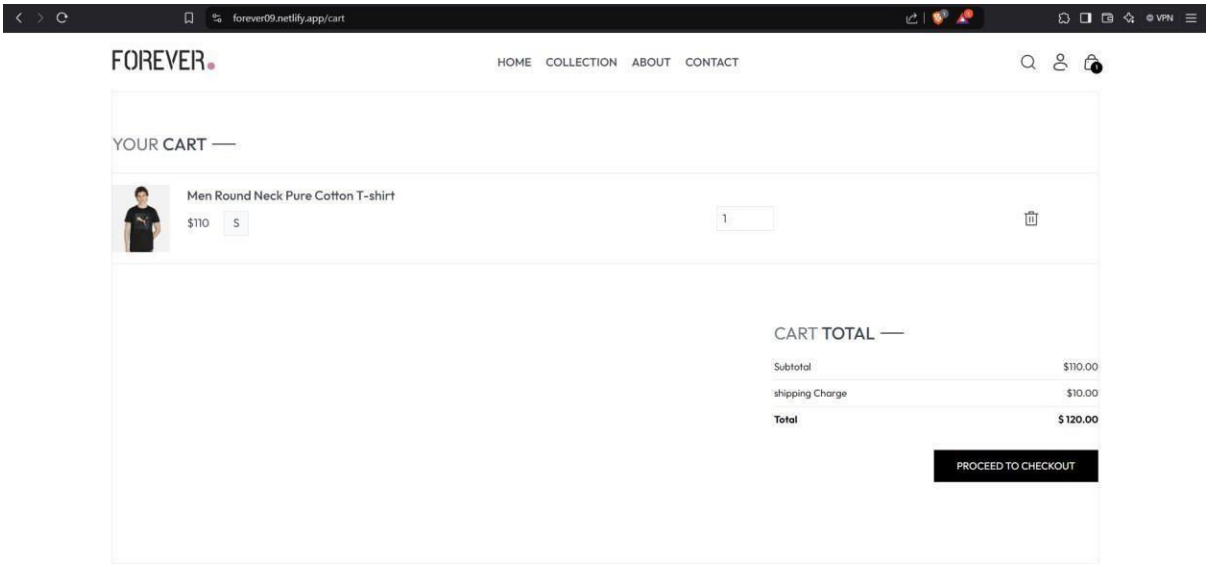


Fig 7.2.6: Payment page

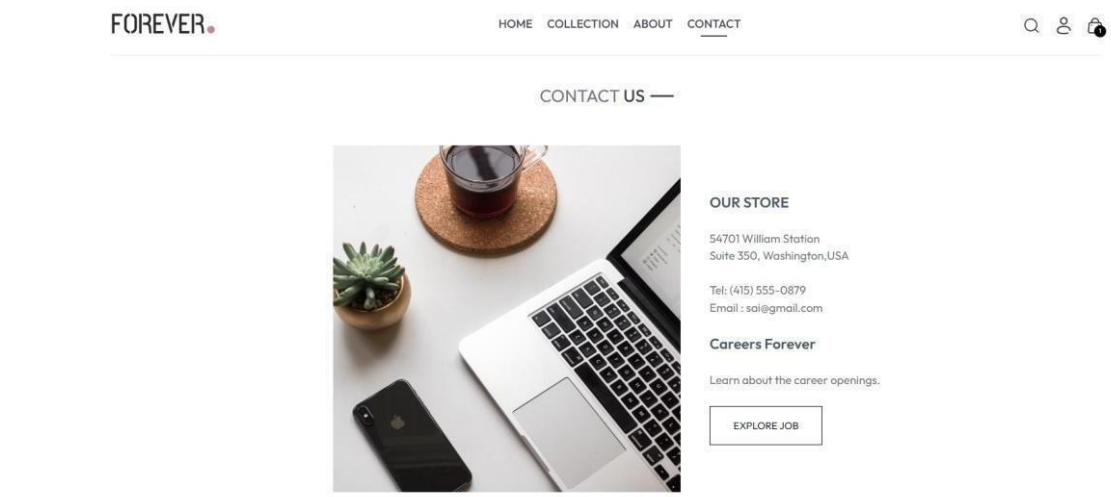


Fig 7.2.7: Contact page

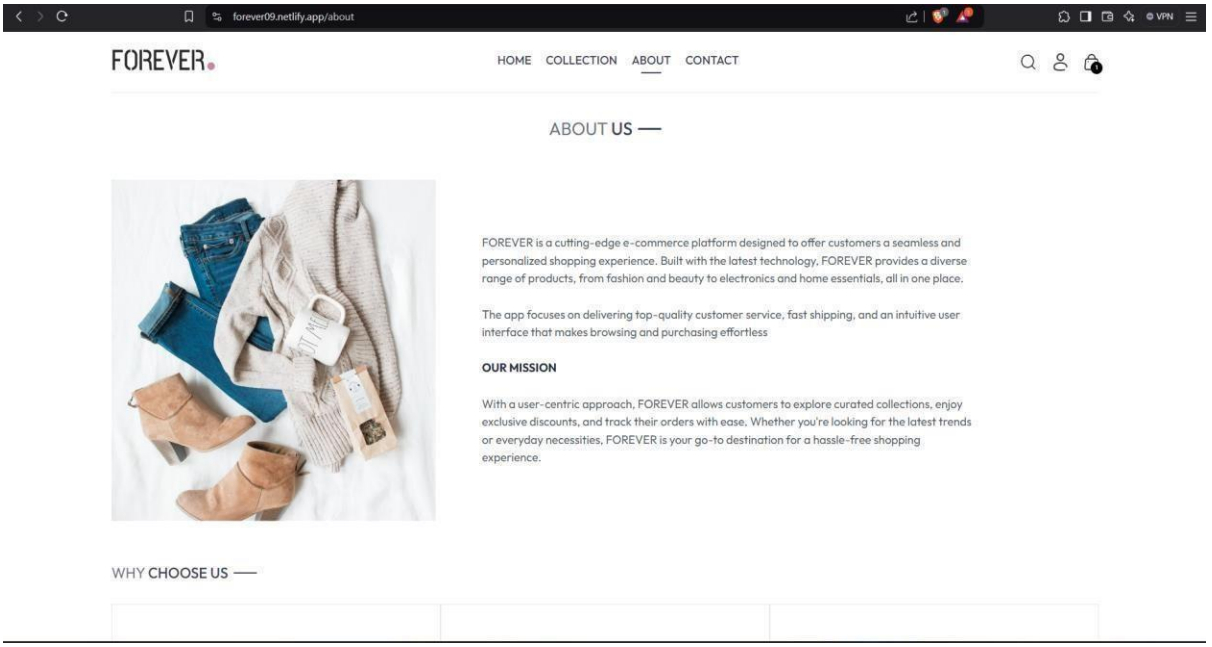


Fig 7.2.8: About page

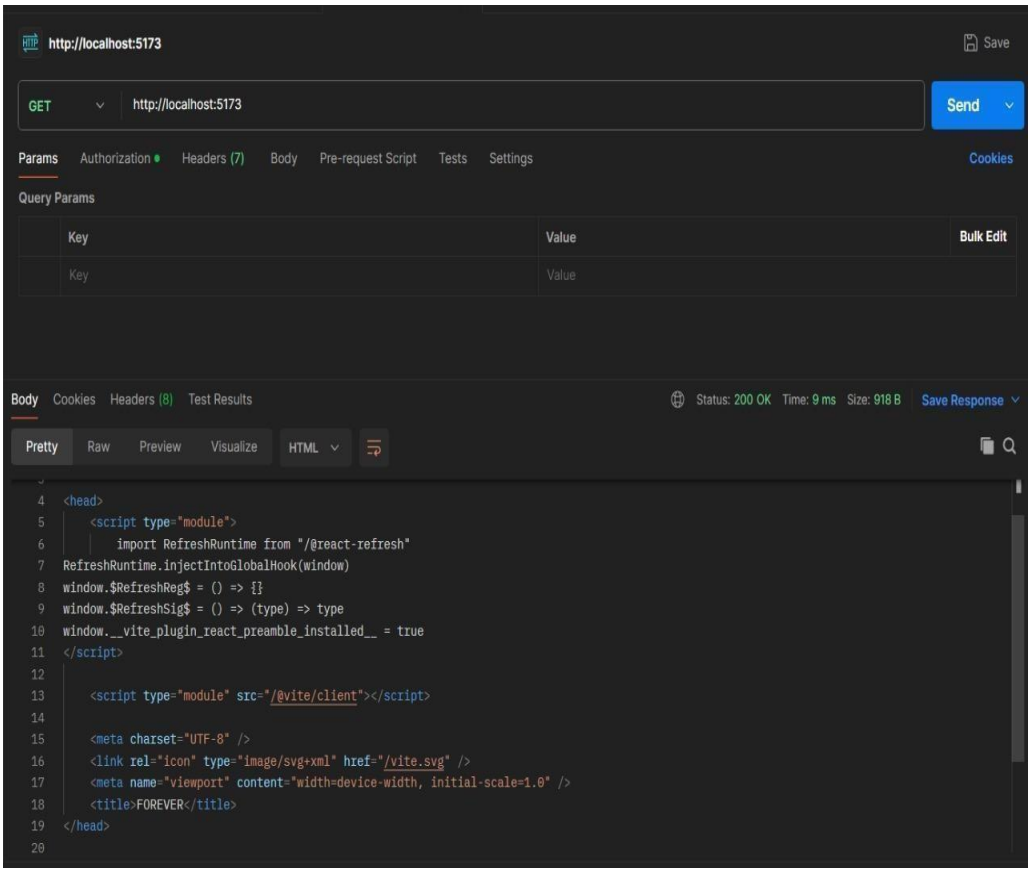


Fig 7.2.9: Postman testing

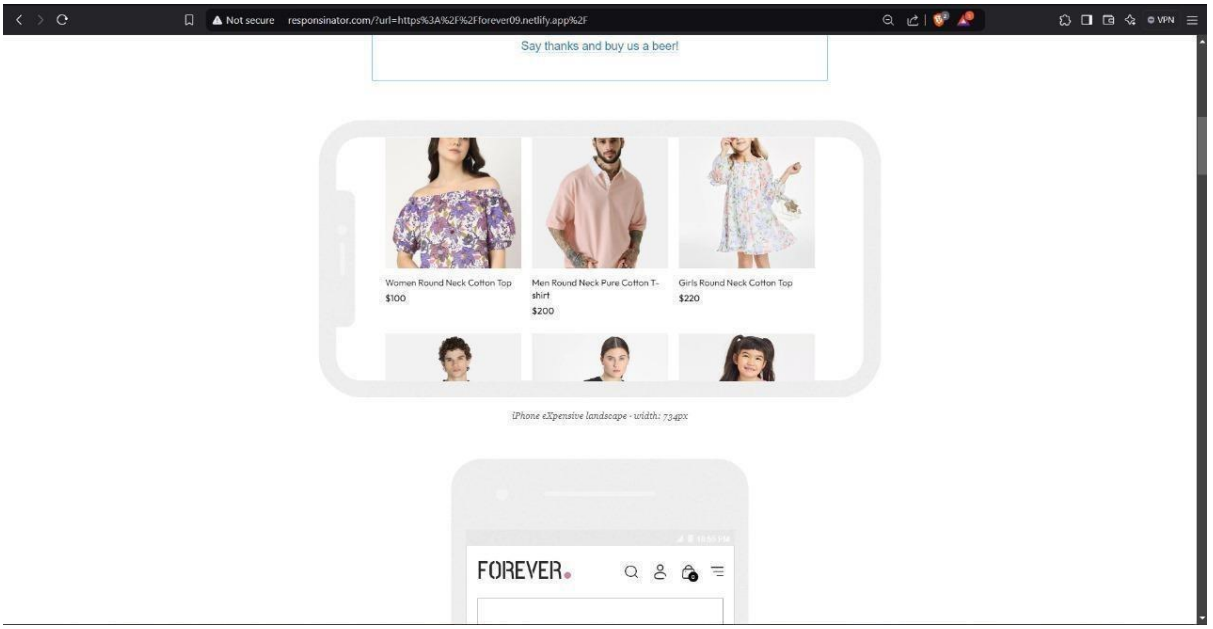


Fig 7.2.10: Responsiveness testing

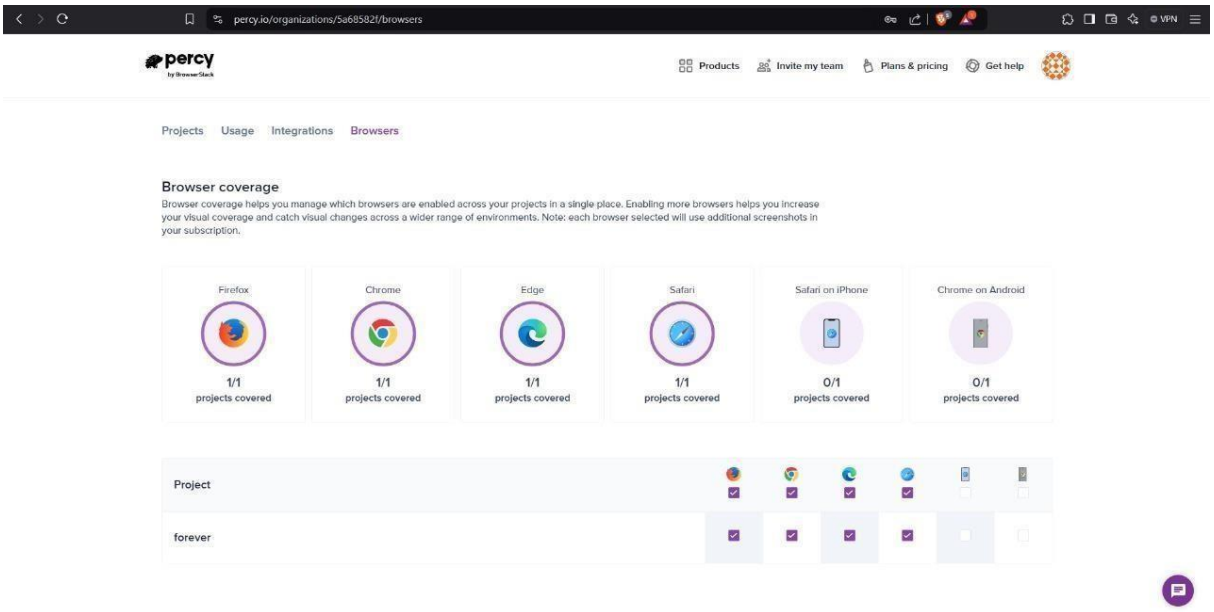


Fig 7.2.11: Percy tool testing

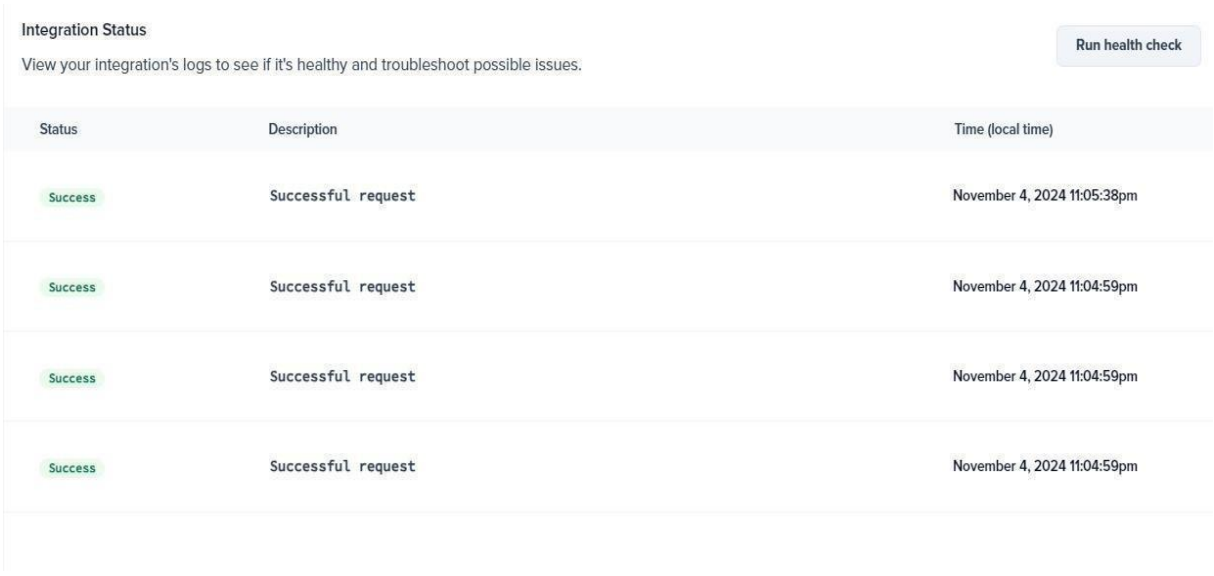


Fig 7.2.12: Percy testing

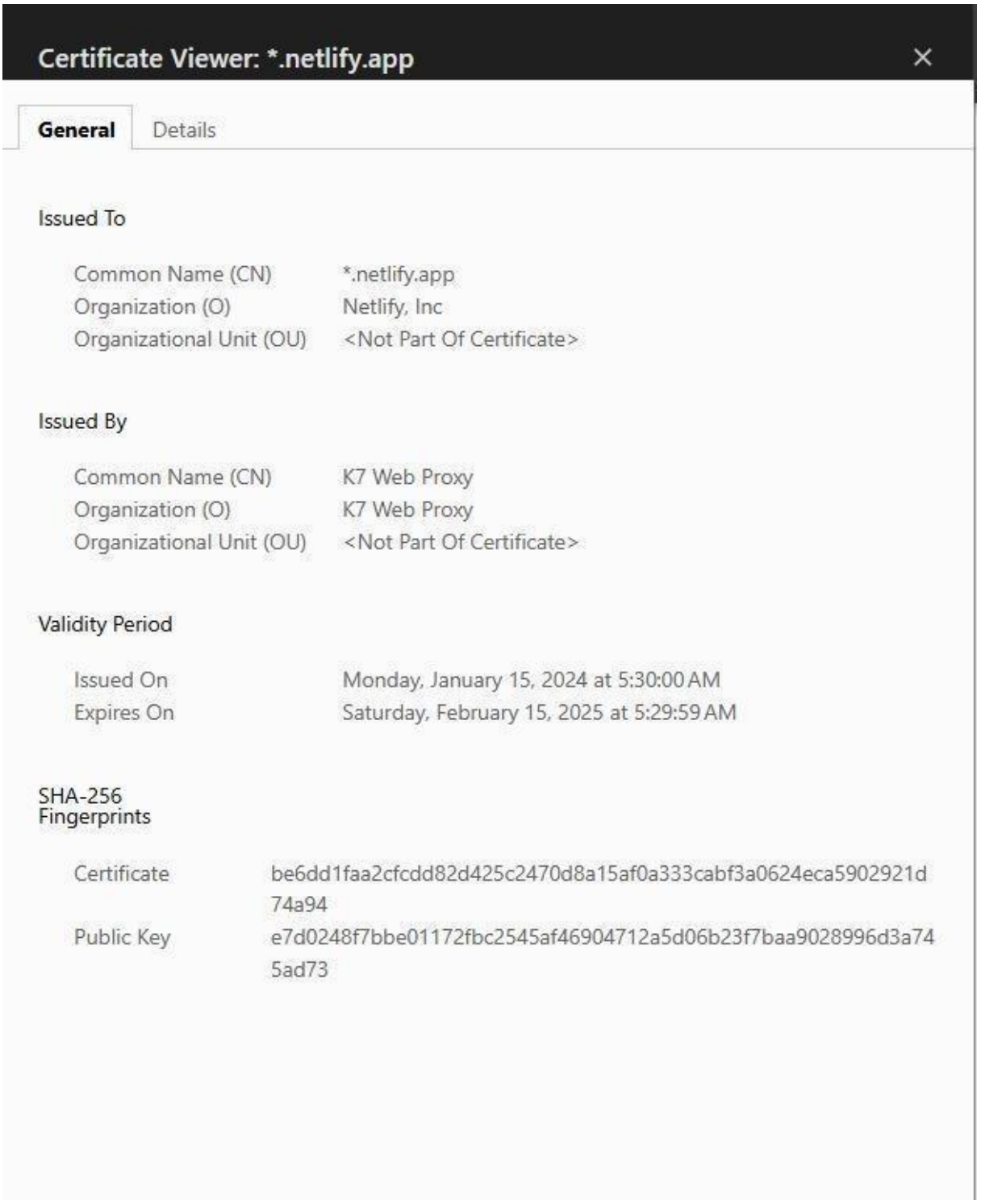


Fig 7.2.13: Certification

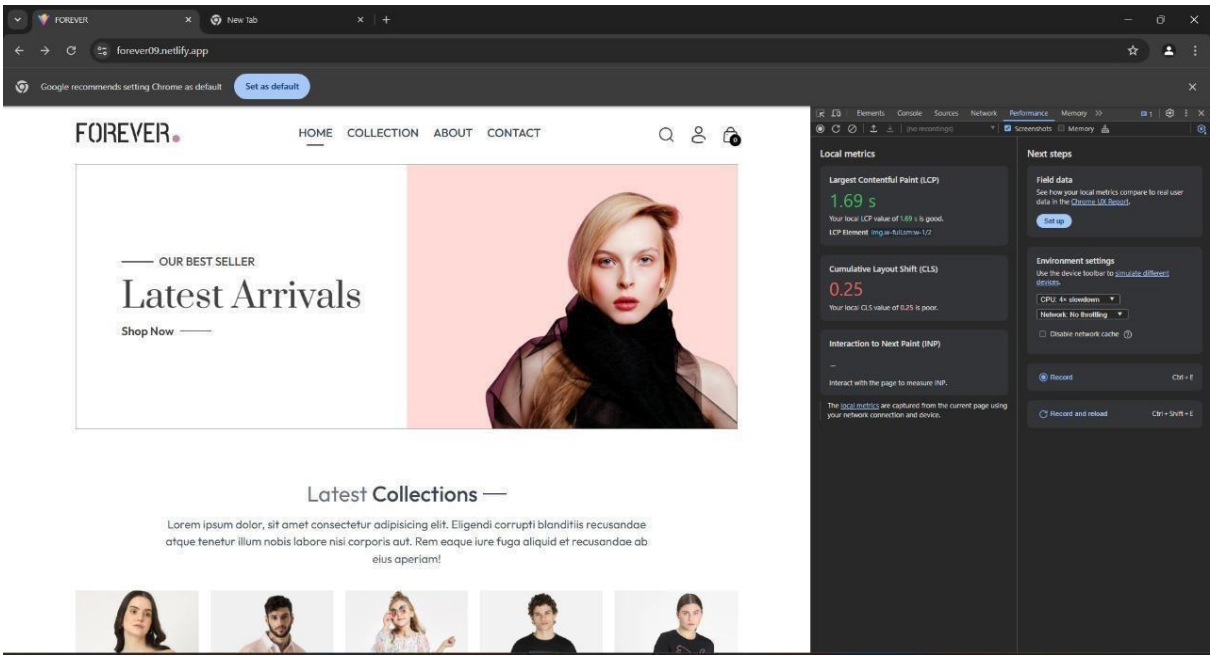


Fig 7.2.14: Performance testing

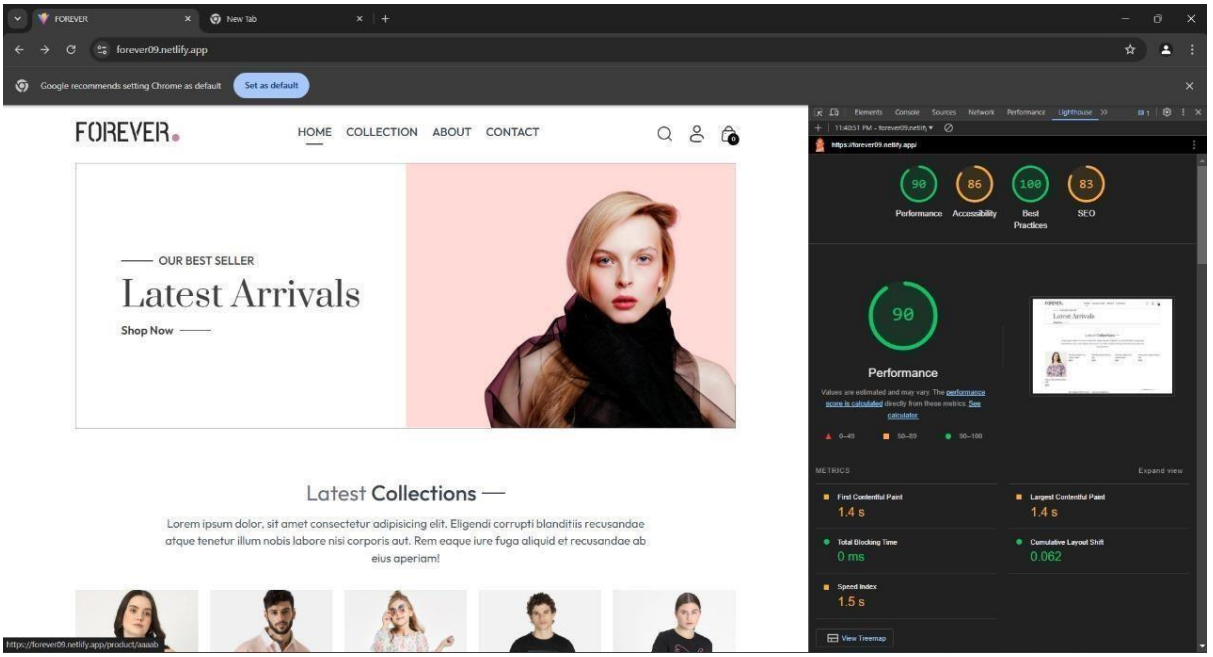


Fig 7.2.15: Lighthouse testing

Fig 7.2.16: Tailwind CSS code

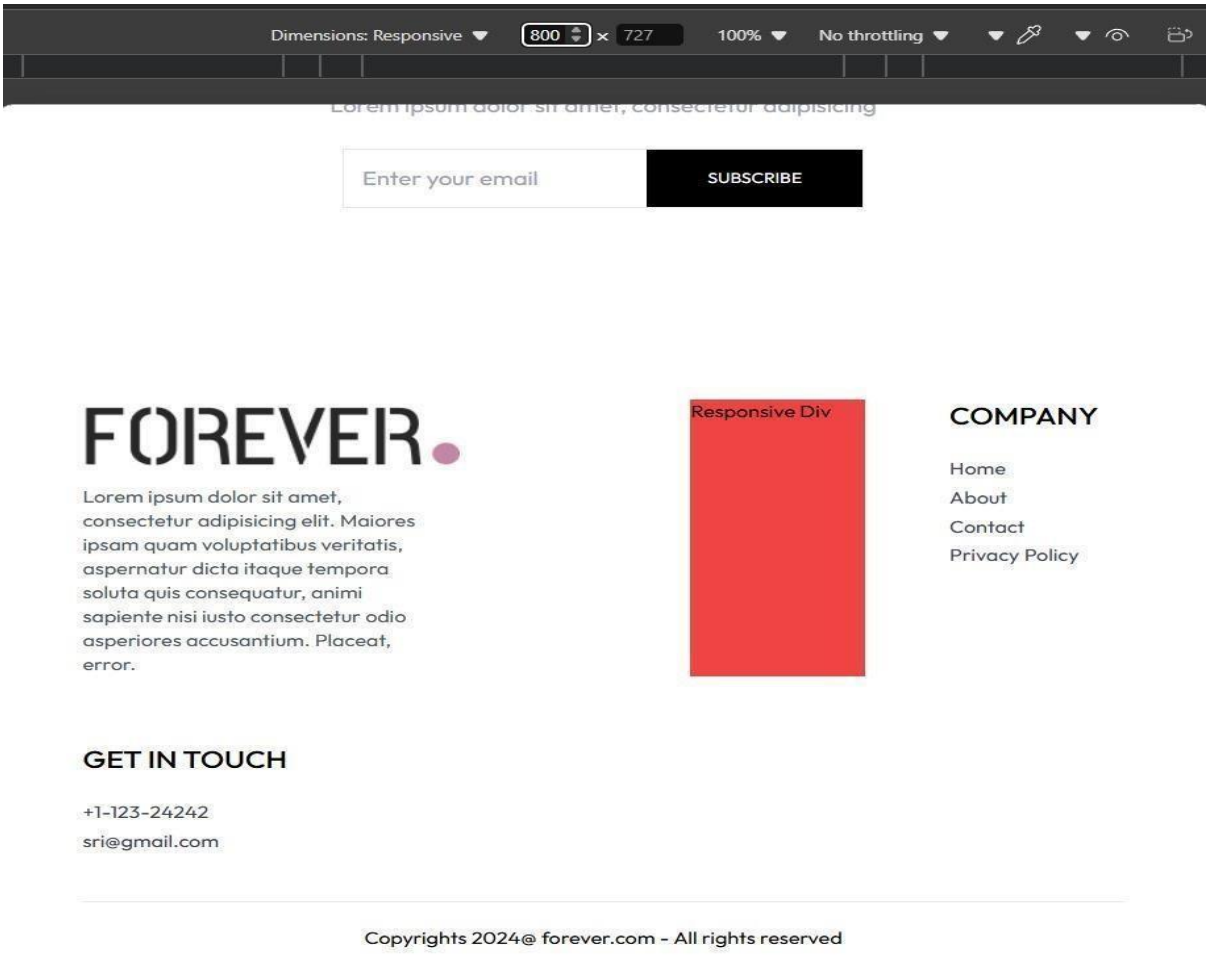


Fig 7.2.17: Tailwind CSS Responsiveness testing

7.3 UML DIAGRAMS:

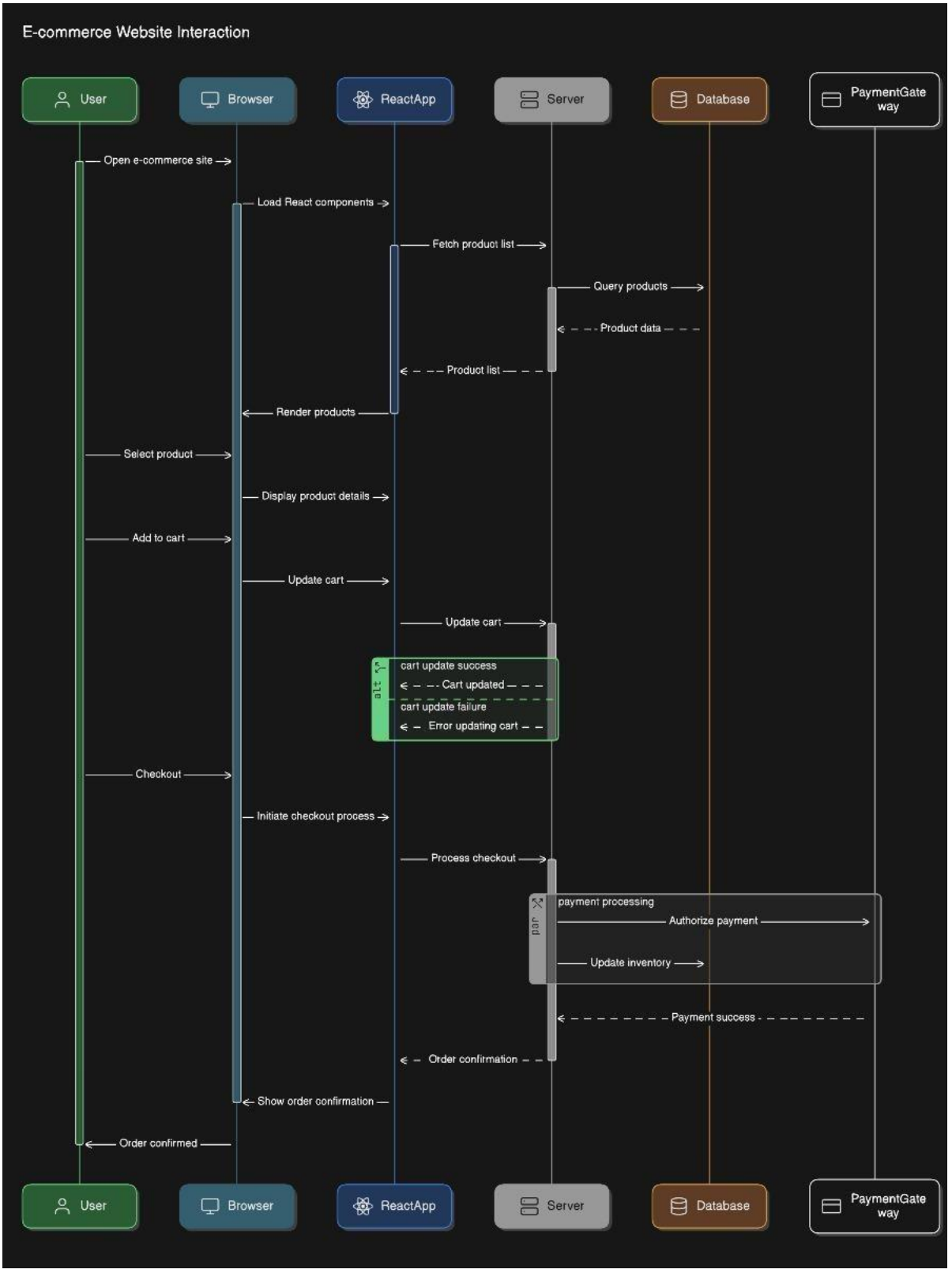


Fig 7.3.1: Sequence diagram

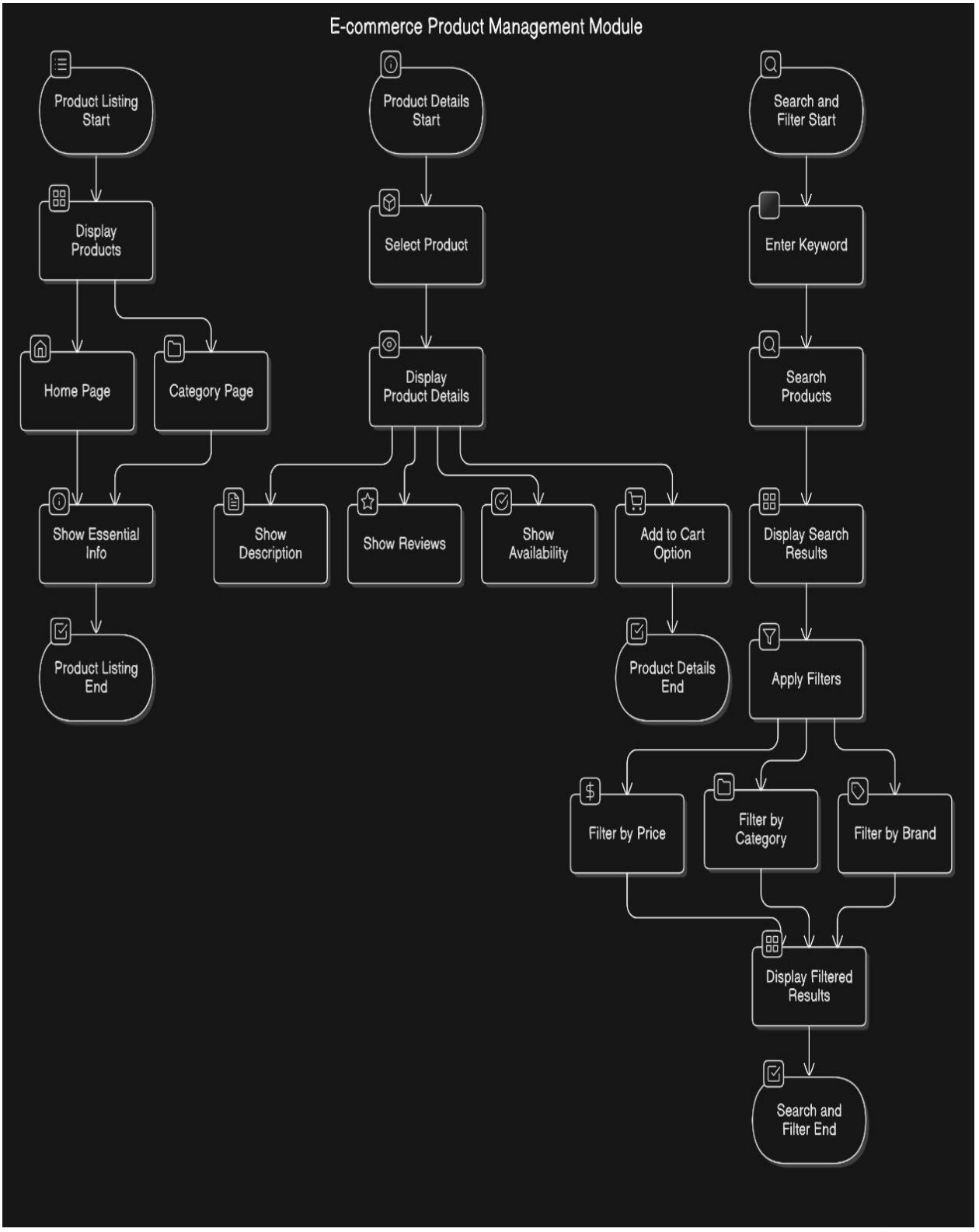


Fig 7.3.2: Product Management Module

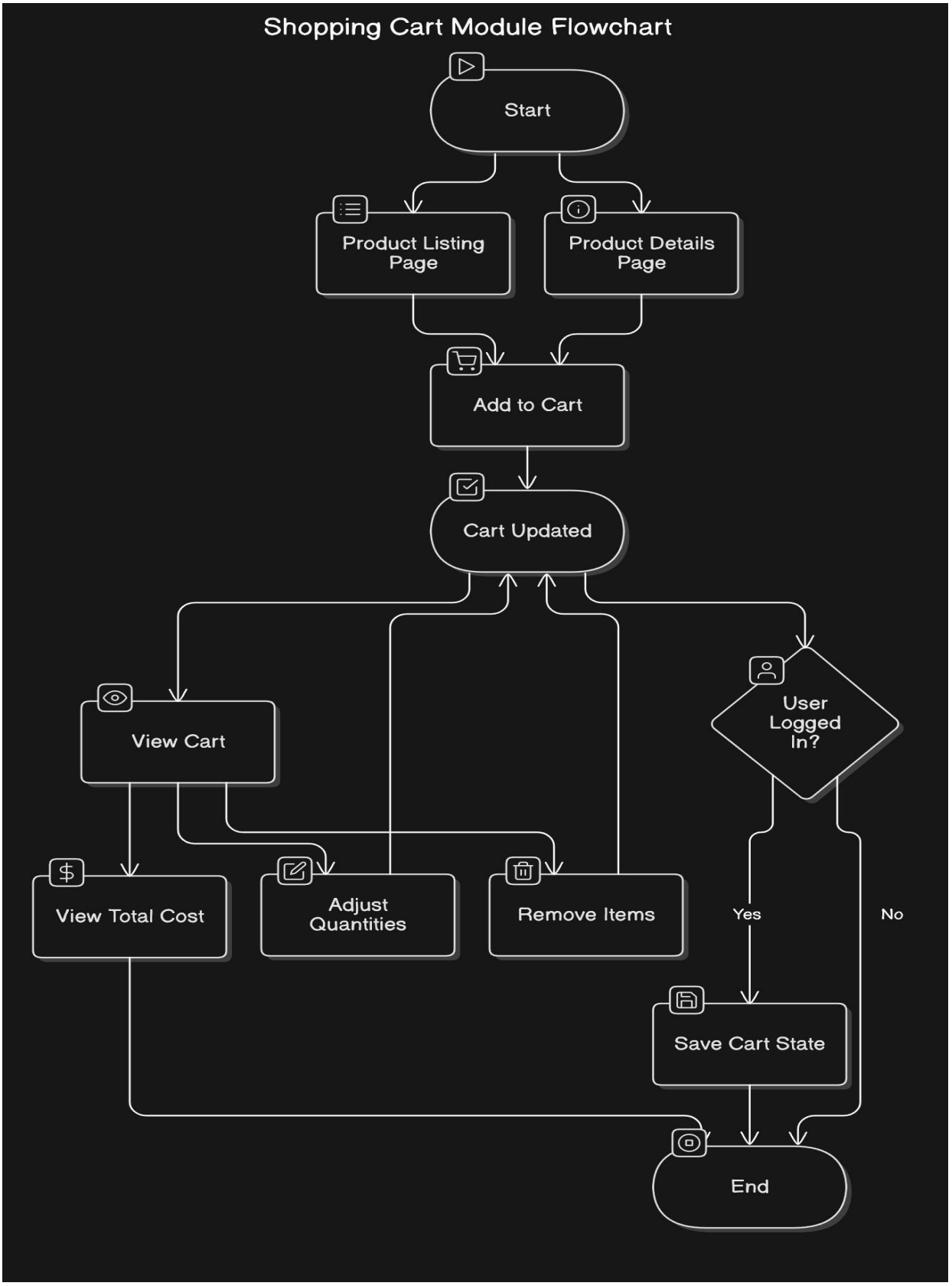


Fig 7.3.3: Cart module

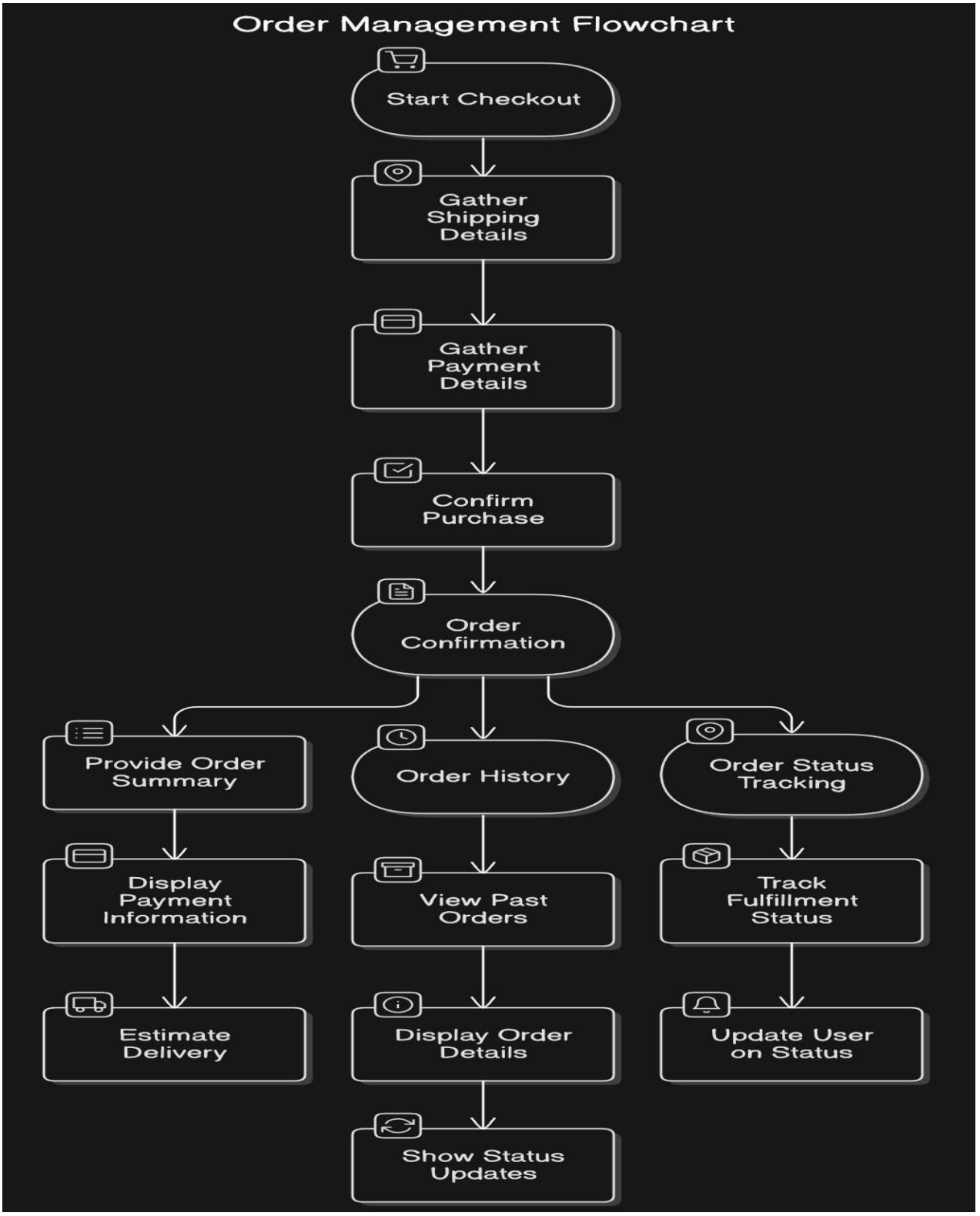


Fig 7.3.4: Order management modules

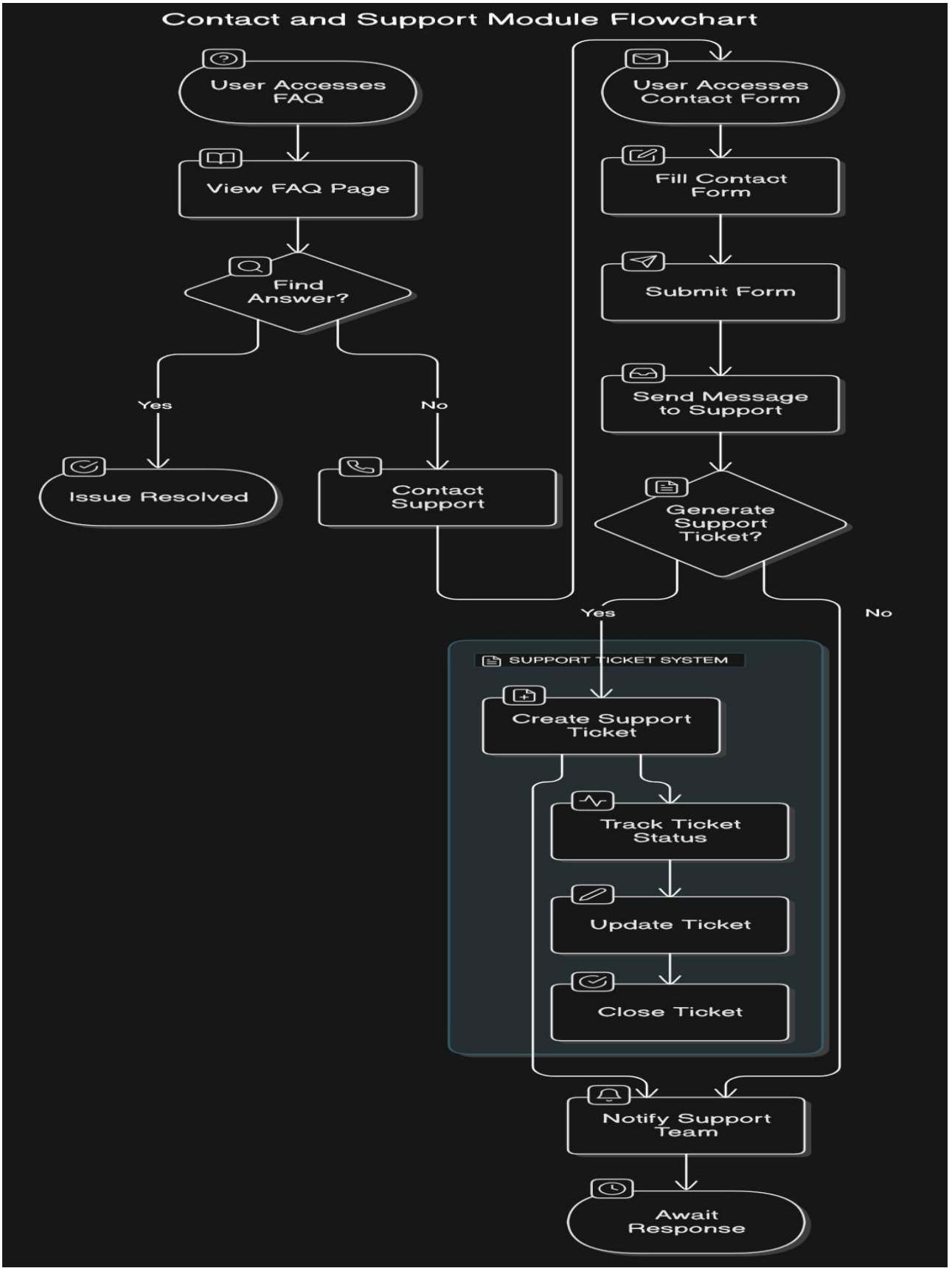


Fig 7.3.5: Contact module

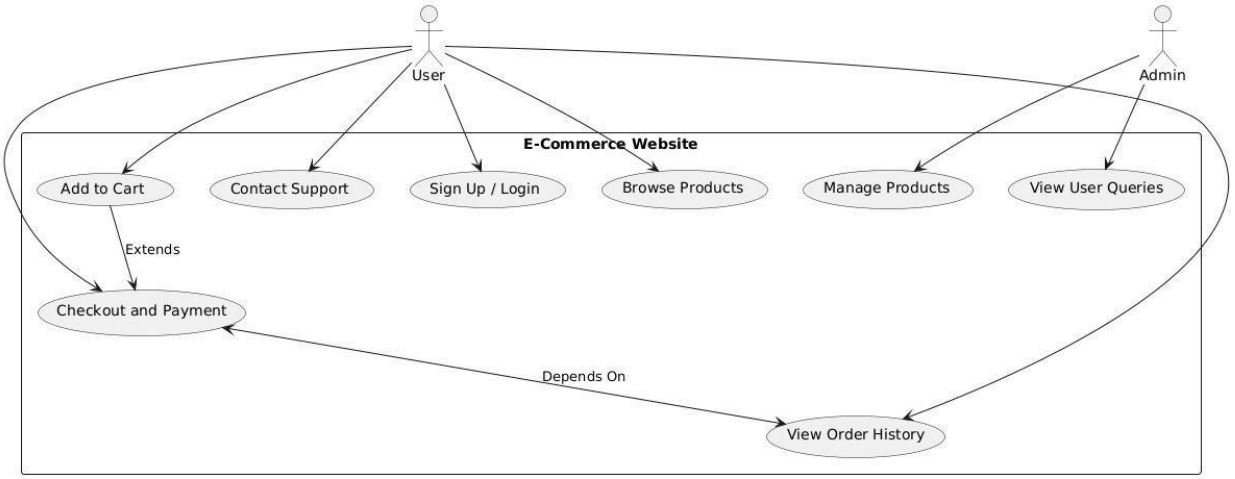


Fig 7.3.6: Use case diagram

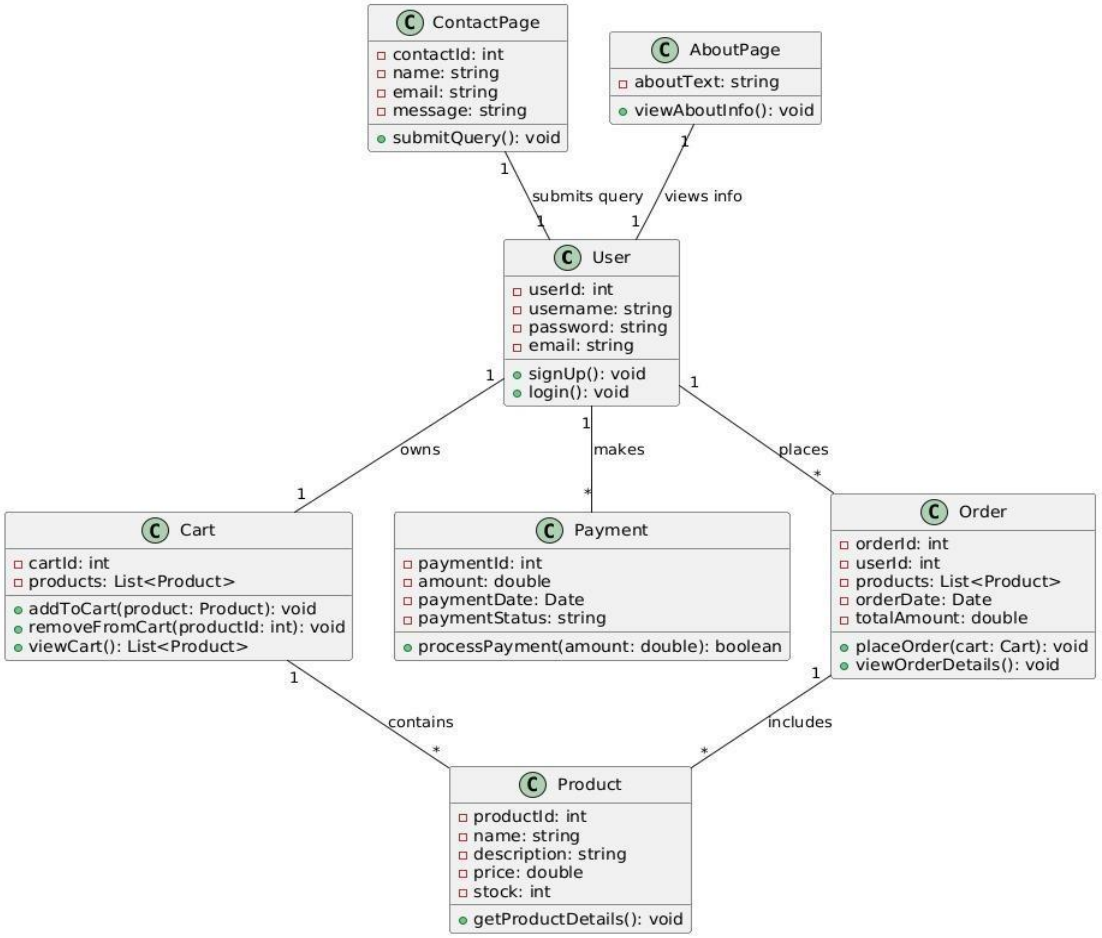


Fig 7.3.7: Class diagram

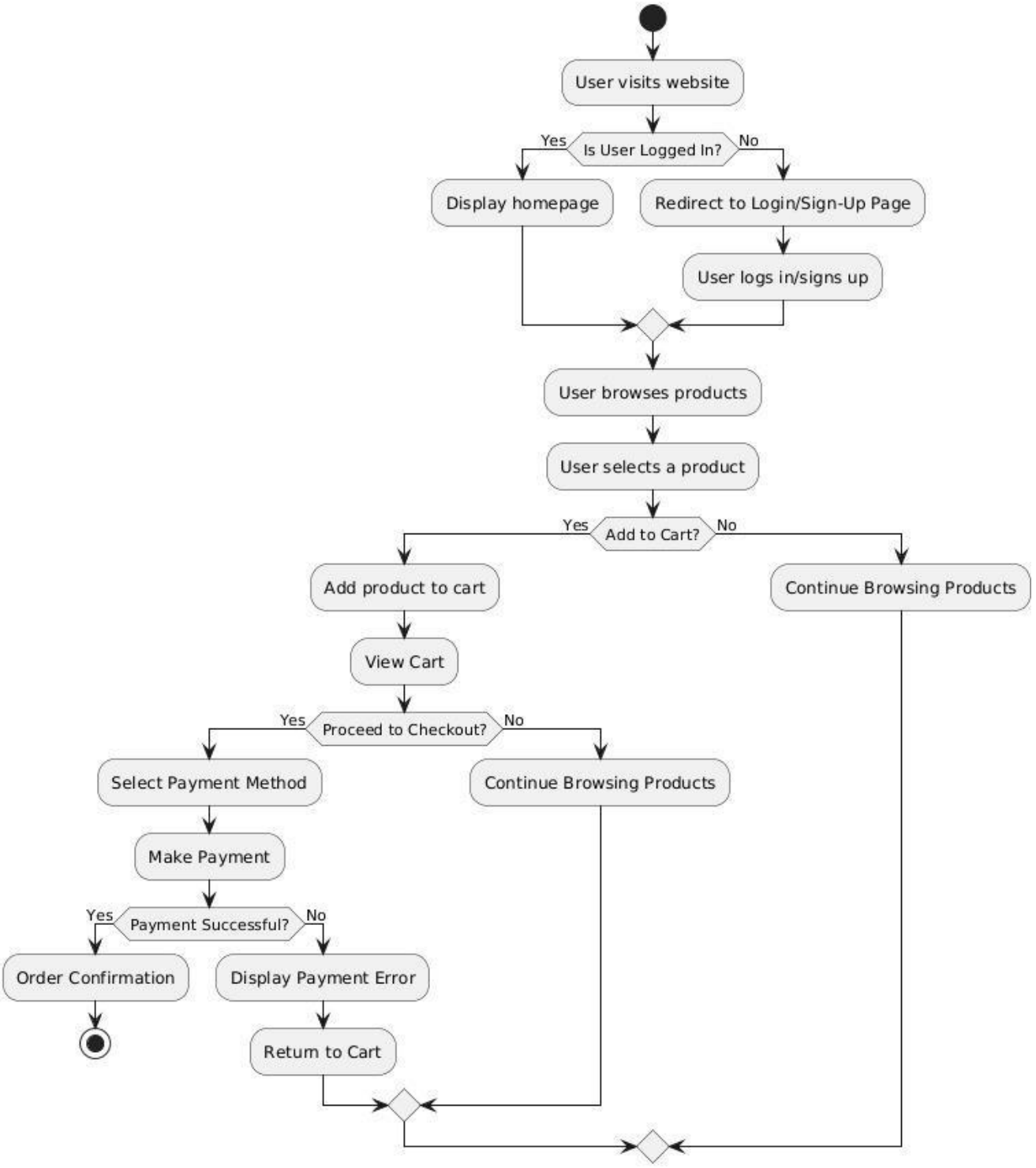


Fig 7.3.8: Activity diagram

7.4 DISCUSSION:

The **FOREVER Web Application** project successfully created a functional, responsive, and accessible e-commerce platform using **React** for the frontend and **Tailwind CSS** for styling. The development process focused on key features such as product browsing, a shopping cart, user authentication, and a contact form, all while ensuring a seamless user experience across devices. The application was thoroughly tested for responsiveness, accessibility, API integration, security, and visual consistency, which led to high-quality results. Challenges such as responsiveness issues, state management, and accessibility were addressed through careful design choices and testing, ensuring a smooth user experience. Looking ahead, **Phase 2** will focus on adding a backend system with **Node.js** and **Express**, incorporating a database for dynamic content, implementing an admin panel, and integrating a payment gateway. These enhancements will elevate the application from a static site to a fully functional e-commerce platform. The project demonstrates a solid foundation for future improvements, scalability, and the ability to deliver a secure, high-performance online shopping experience.

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

8.1 CONCLUSION:

The **Shopping Mart Web Application** project successfully showcases the development of a fully functional e-commerce platform, utilizing modern technologies such as **React** for the frontend and **CSS Tailwind** for responsive and efficient styling. The application includes key features such as product catalog browsing, a shopping cart, user authentication, and a contact page, all working seamlessly together to provide a smooth shopping experience.

Through rigorous **testing**, including **responsiveness testing**, **accessibility testing**, **Postman API testing**, **Google Lighthouse audits**, **SSL/TLS certificate validation**, and **Percy visual regression testing**, the application has proven to be highly reliable, secure, and user-friendly. The **performance** and **SEO** scores in the Google Lighthouse audits were solid, and **accessibility** compliance was improved through necessary adjustments. Moreover, **security certificates** were properly configured, ensuring that user data is transmitted securely via HTTPS.

Overall, this system demonstrates the ability to build and test a complex, interactive web application that adheres to modern standards for security, performance, and usability. The successful integration of various testing tools has ensured the application is robust, reliable, and provides an excellent user experience across all devices and screen sizes. This project lays a strong foundation for further enhancement and deployment in real-world scenarios.

8.2 FUTURE ENHANCEMENT:

1. Adding a Backend (Phase 2)

In Phase 2, the frontend application will be integrated with a robust **backend** system, which will handle dynamic data storage, user management, and order processing. This will significantly improve the scalability and functionality of the application.

- **Backend Framework**: A suitable backend framework such as **Node.js with Express**, **Django**, or **Ruby on Rails** will be used to implement the server-side logic, APIs, and database interactions.
- **Database Integration**: The project will incorporate a **relational database** (like **MySQL** or **PostgreSQL**) or a **NoSQL database** (like **MongoDB**) to store product details, user information, order history, and transaction data.
- **User Authentication and Authorization**: The backend will provide secure user authentication (e.g., JWT tokens or OAuth) for logging in, registering, and managing user accounts. This will allow users to securely log in and view personalized information, such as their shopping history.
- **Cart and Order Management**: The backend will manage the shopping cart and the checkout process. When users add products to the cart, the data will be stored and processed on the server, with transaction details being saved after the checkout process.
- **Payment Integration**: Future development will include integrating a secure payment gateway (e.g., **Stripe** or **PayPal**) to enable users to make payments directly through the platform.

2. **Advanced Product Search and Filters**The current application provides basic product browsing, but future enhancements will focus on providing more advanced search and filtering features, including:

Search by Categories: Users will be able to search for products within specific categories (e.g., Electronics, Clothing, etc.).

Faceted Search: Filters like price range, ratings, size, color, and brand will be implemented to help users find products more easily.

Sorting: Users will be able to sort products by price, popularity, new arrivals, and customer ratings.

3. Improved Performance and Optimization

To ensure that the web application scales effectively with increasing traffic, further optimization will be done in the backend:

- **Caching**: Implementing server-side caching strategies (e.g., Redis) to reduce database load and speed up content delivery.
- **Load Balancing**: Use load balancing techniques to distribute incoming traffic efficiently across multiple servers.
- **Asynchronous Processing**: Background tasks such as email notifications and payment processing will be handled asynchronously to avoid blocking the main request-response cycle.

4. Advanced Analytics and Reporting

- Description: Introduce advanced analytics tools for both users and administrators. This could involve tracking user behavior, preferences, and event booking trends to better understand the market and improve service offerings.
- Technologies: Google Analytics, Tableau, or custom analytics dashboards.
- Technologies: Integration with email marketing services like Mailchimp and social media APIs.

CHAPTER 9

REFERENCES

1) D. Duckett, HTML and CSS: Design and Build Websites. Indianapolis, IN, USA: Wiley, 2011.

2) M. Haverbeke, Eloquent JavaScript: A Modern Introduction to Programming, 3rd ed. San Francisco, CA, USA: No Starch Press, 2018.

3) D. Flanagan, JavaScript: The Definitive Guide, 7th ed. Sebastopol, CA, USA: O'Reilly Media, 2020.

4) R. K. Soni, "Responsive Web Design and Development for E-Commerce Websites," International Journal of Computer Applications, vol. 176, no. 28, pp. 20–24, May 2020.

5) J. Garrett, The Elements of User Experience: User-Centered Design for the Web and Beyond, 2nd ed. Indianapolis, IN, USA: New Riders, 2011.

6) S. Y. Liang, "Frontend Testing Strategies for Modern Web Applications," IEEE Software Engineering Journal, vol. 36, no. 4, pp. 85–90, Jul. 2020.

7) E. Freeman and E. Robson, Head First Design Patterns: Building Extensible and Maintainable Object-Oriented Software, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2020.

8) R. J. Munz and S. Munz, "A Guide to Testing in the React Ecosystem," Proceedings of the ACM Conference on Human-Computer Interaction, pp. 15–19, 2021.

9) K. Johnson, "Applying Accessibility Standards to Web Applications," Journal of Web Engineering, vol. 16, no. 2, pp. 45–62, Apr. 2019.

- FOREVER Ecommerce website- https://forever09.netlify.app/