# Acknowledgement

I want to take a moment to express my absolute appreciation to my faculty, mentors, and friends for their constant support and guidance during the construction of Ghar Sanshar. Their technical knowledge in web development, feedback, and support helped develop this platform. They introduced me to existing modern tools, clarified things I struggled to understand, and helped create a collaborative environment that helped my learning and ultimately the platform itself. I also want to thank the open-source community, online documentation, and resources that proved to be an invaluable repository of knowledge and tools that allowed the construction of a resilient and user-centric e-commerce.

# Abstract

Ghar Sansar is a new web-based e-commerce platform offering a more innovative way to browse and shop for home decor products. Using a modern tech stack—React.js with Vite for a fast, responsive frontend, Node.js with Express.js for scalable backend, and PostgreSQL for secure data—it is an impressive combination that provides users with an intuitive, fast, reliable, and secure online shopping experience. Key features include a user authentication system using JWT, a robust product listing with filtering and sorting, a persistent cart feature, an optimized checkout experience, order tracking, and an admin panel with role-based access to manage orders and inventory. The entire project was completed utilizing an agile methodology with iteration of requirement gathering documentation, wireframes with Figma, and version control with Git. There was also considerable time set aside to thoroughly test the application with React Testing Library. Requirements for usability, performance and security were followed at every tier of development. Ghar Sansar addresses typical user experience challenges associated with an e-commerce application, including cluttered navigation and workflows, security of transaction, and insufficient/ineffective administrative tools. Ghar Sansar provides browsing and shopping experiences for customers that enhance usability and a focus on security for payments and sensitive data, without compromising performance. The admin panel includes a catalog that allows for sorting, filtering, searching, listing, adding, and removing, for administrators all in one user interface. The development methodology affordability also provided an environment of scalable and extensibility for the e-business model in this e-commerce brand and work as intended. Testing included unit testing, integration testing, instrumented and vulnerability tests, usability testing, and performance testing every tier of client or user. Ghar Sansar is hoping to reshape the home decor shopping and purchasing experience and empowers customer browsing changes, and enhances efficiency for administrator controls and decisions within a scalable e-commerce solution.

Contents

[Acknowledgement 2](#_Toc204001727)

[Abstract 3](#_Toc204001728)

[Introduction 5](#_Toc204001729)

[Aim 5](#_Toc204001730)

[Objectives 6](#_Toc204001731)

[Scope 7](#_Toc204001732)

[Design Process 8](#_Toc204001733)

[Functional Requirements 8](#_Toc204001734)

[Non-functional Requirements 10](#_Toc204001735)

[Prototype 11](#_Toc204001736)

[System Architecture 16](#_Toc204001737)

[Frontend (Client-side) 16](#_Toc204001738)

[Backend (Server-side) 17](#_Toc204001739)

[Database Layer 17](#_Toc204001740)

[Use Case Diagram 17](#_Toc204001741)

[ER Diagram 18](#_Toc204001742)

[Tools 19](#_Toc204001743)

[Technology 20](#_Toc204001744)

[Frontend Technologies 20](#_Toc204001745)

[Backend Technologies 21](#_Toc204001746)

[Database Technologies 21](#_Toc204001747)

[Implementation 21](#_Toc204001748)

[Frontend Implementation 21](#_Toc204001749)

[Backend Implementation 25](#_Toc204001750)

[Database Implementation 31](#_Toc204001751)

[Testing and Evaluation 32](#_Toc204001752)

[Future Work 33](#_Toc204001753)

[Conclusion 34](#_Toc204001754)

[References 35](#_Toc204001755)

[Appendix 35](#_Toc204001756)

[Figure 1Objectives 9](#_Toc204073866)

[Figure 2Functional Requirement 12](#_Toc204073867)

[Figure 3Non functionality 13](#_Toc204073868)

[Figure 4Landing page 14](#_Toc204073869)

[Figure 5authentication 15](#_Toc204073870)

[Figure 6 Product Catelog 16](#_Toc204073871)

[Figure 7Cart Page 17](#_Toc204073872)

[Figure 8System Architecture 18](#_Toc204073873)

[Figure 9Use case diagram 20](#_Toc204073874)

[Figure 10Er diagram 21](#_Toc204073875)

[Figure 11Tools 22](#_Toc204073876)

[Figure 12login 24](#_Toc204073877)

[Figure 13Register 24](#_Toc204073878)

[Figure 14Product Catelog 24](#_Toc204073879)

[Figure 15Cart 25](#_Toc204073880)

[Figure 16Checkout 25](#_Toc204073881)

[Figure 17product 26](#_Toc204073882)

[Figure 18Order 26](#_Toc204073883)

[Figure 19auth controller 27](#_Toc204073884)

[Figure 20product controller 28](#_Toc204073885)

[Figure 21Order controller 29](#_Toc204073886)

[Figure 22auth routes 29](#_Toc204073887)

[Figure 23product routes 30](#_Toc204073888)

[Figure 24order orutes 30](#_Toc204073889)

[Figure 25middlware 31](#_Toc204073890)

[Figure 26server.js 31](#_Toc204073891)

[Figure 27Testing 33](#_Toc204073892)

# **Introduction**

The way people shop has changed dramatically over the past few years, with the rise of online shopping especially. It has made everything so convenient to do—you type a few words, and you have access to just about anything you can think of. One category that's truly blown up is home decor. More and more people want their home life to be a reflection of who they are, and so they're on the lookout for unique, show-stopping items that make their houses pop and proclaim "us." The problem is, most marketplaces these days just aren't cutting it. They're clunky to use, not particularly secure when paying, and a bit of a nightmare to deal with on the admin front.

That's where Ghar Sansar comes in. It's an e-shop that's specifically built for home decor lovers, and it actually solves those common issues instead of adding to them. The website itself is built with React.js and Vite, so it loads fast and is responsive to use. On the backend, it's based on Node.js with Express.js, and data handling is handled with PostgreSQL—basically, it has a solid tech base. There's a lot jammed into it, too: safe login with JWT authentication, a product catalog that's not a pain to browse, cart functionality that doesn't bat an eye, simple checkout, live tracking of orders, and a dashboard where admins can keep everything organized without having to dig through menus or reinvent things the hard way.

What makes it even more awesome is that the whole thing was done with Agile methodologies in mind, so it wasn't thrown together. The team worked in sprints, tested all along with React Testing Library, and even wired up the UI first in Figma before building it out. Git version control kept everything organized and traceable. By and large, Ghar Sansar is not yet another online retailer—it's a thoughtful, well-planned site that actually makes the home decor shopping experience easy for everyday users while giving admins the tools they need to maintain order. It's modern, sleek, and designed with real human beings in mind.

# **Aim**

Ghar Sanshar goal is to establish an easy-to-use, scalable, and secure e-commerce platform for home decor products. With a responsive and dynamic frontend by React.js with Vite, a scalable and robust backend by Node.js with Express.js, and secure data storage through PostgreSQL, the website provides an easy shopping experience, keeps transactions secure, and provides simple administrative features. Ghar Sanshar will enhance user interaction with features like personalized shopping for products, secure authentication, and easier checkout, and is built to scale to accommodate growing user needs and product offerings.

# **Objectives**

Make Sure It's Very Secure: Implement JWT-based authentication and bcrypt.js for password hashing to safeguard user data and transactions to keep it reliable and secure.

Manage Cart: Enable logged-in customers to add, modify, and remove items in their cart with data saved in PostgreSQL and synced with localStorage, if needed, for more comfortable user experience.

Streamline Checkout Process: Support easy checkout with shipping information collection and order confirmation for easy buying process.

Enable Administrators: Create a secure admin panel for order, product, and user account management that is only accessible to logged-in users with role-based permissions.

Enable Scalability and Performance: Design a system architecture to handle increasing user traffic and product listings without any reduction in speed and reliability.

Improve Usability: Give a responsive and intuitive UI/UX that is device and browser agnostic, providing the best shopping experience to all..

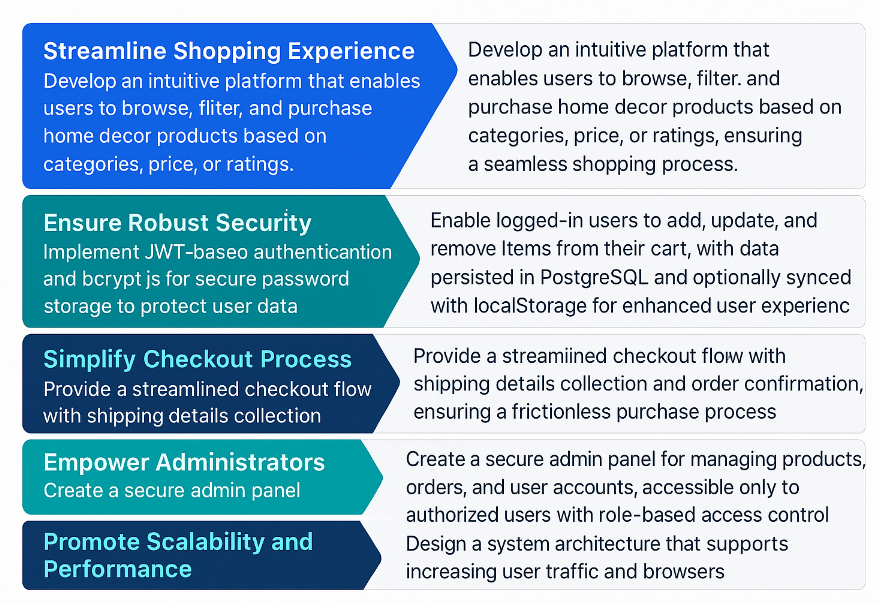


Figure 1Objectives

# **Scope**

The scope of Ghar Sansar encompasses essential e-commerce functionality, including user login, product browsing, carting, checkout, order tracking, and admin management. The app has two principal user segments: customers, for whom customers buy home decor products, and admins, to whom the admins control the app. It utilizes React.js with Vite, React Router, and React Testing Library for frontend, Node.js with Express.js for backend, and PostgreSQL (using pgAdmin4) for storing data. Security is achieved with JWT for authentication and bcrypt.js for password encryption, and scalability and performance are attained through a streamlined three-tier architecture. The project steers clear of functionalities like real-time payment gateway integration, mobile application development, or product recommendations using AI to maintain the solution robust web-based. Development is performed according to an Agile process, prototyping on Figma, version control through Git, and testing with React Testing Library for quality, user experience, and performance in an academic and resource-constrained setting [(“What Is E-Commerce? Scope of E-commerce,” 2025)](#scope).

# **Design Process**

So, Ghar Sansar is this project we worked on that’s meant to be a simple online shop for home decor stuff. Nothing super advanced—just something that works and covers the basics. The idea was to make sure people could create accounts, log in, check out products, throw them in a cart, go through checkout, and keep track of what they ordered. There’s also a part for admins where they can update products, manage orders, and keep the store going. Pretty much, it’s set up for two kinds of users: regular customers who want to shop and admins who take care of everything on the backend.

For the tech side of things, we used React.js with Vite to build the frontend. It helps the site load quicker and keeps things feeling smooth. We also used React Router for page navigation and React Testing Library to catch problems early on. The backend’s built with Node.js and Express.js, and PostgreSQL handles all the data. We worked with pgAdmin4 to manage that. To keep logins secure, we added JWT for authentication, and passwords are encrypted using bcrypt.js, so nothing’s stored in plain text. The whole thing runs on a simple three-tier architecture, which made the setup more organized and easier to update later if needed.

We didn’t try to make it too complicated. Stuff like mobile apps, payment integration, or AI-based features were left out on purpose—mostly because we didn’t need them for the project, and we were short on time and resources anyway. We used Figma to sketch the design, Git for version tracking, and followed Agile steps while building it. The main goal was just to get something working, not perfect—just reliable, straightforward, and easy for both users and admins.

# **Functional Requirements**

**User Authentication**:

Safely register, log in, and log out with JWT for authentication using tokens and bcrypt.js for hashing passwords.

Role-based authorization for admins (admins) and buyers (customers).

Save JWT tokens to localStorage in order to enable persistent user sessions.

**Cart Management**:

Enable logged in customers to add, edit or remove items from their cart.

Save cart information into PostgreSQL, with optional offline sync to localStorage for offline access and better user experience.

**Checkout Process**:

Enable users to select cart items, input shipping address, and place order.

Clear the cart on successful checkout and show an order confirmation page.

**Order Management**:

Customers can view orders placed in the past and their details (e.g., items, status, payment method).

Admins can view all orders and mark statuses (e.g., delivered, shipped).

**Admin Panel**:

Give admins that can add, edit, or delete products, manage user accounts, and mark order statuses.

Enable secure product image uploads via Multer [(AltexSoft Editorial Team, 2023)](#fs)

.

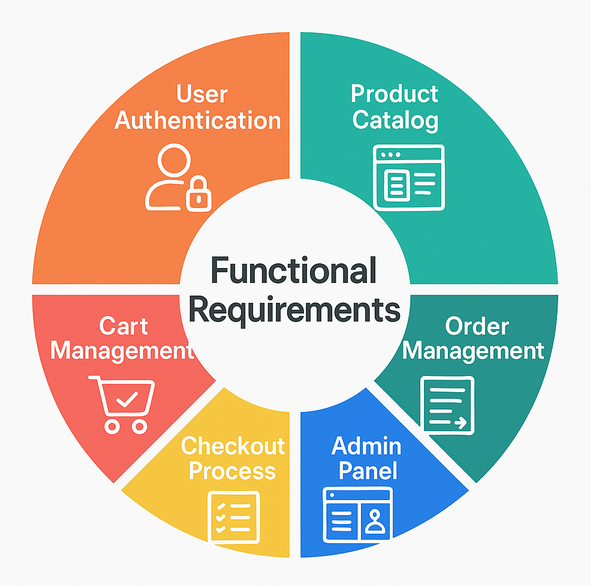


Figure 2Functional Requirement

# **Non-functional Requirements**

**Scalability**: Scale the design of the platform to handle high traffic and many products without compromising its performance, through database query optimization and server optimization.

**Security**: Protect user data through JWT authentication, bcrypt.js password encryption, and defended security against security-based attacks like SQL injection and cross-site scripting (XSS).

**Performance**: Employ quick page load and API response time even under heavy traffic, through optimized code and caching.

**Usability**: Provide a natural, responsive UI/UX on any device and browser so the user can easily use it.

**Compatibility**: Provide the biggest browser support (Chrome, Firefox, Safari) and device support (desktop, mobile, tablet) to provide the same user experience [(Krüger, n.d.)](#nfs)

.



Figure 3Non functionality

# **Prototype**

The Ghar Sanshar prototype, designed in Figma, provides a visual representation of the platform’s user interface and core functionalities, ensuring an intuitive user experience:

Landing Page: Features a search bar, featured products, and category navigation to welcome users and encourage exploration.

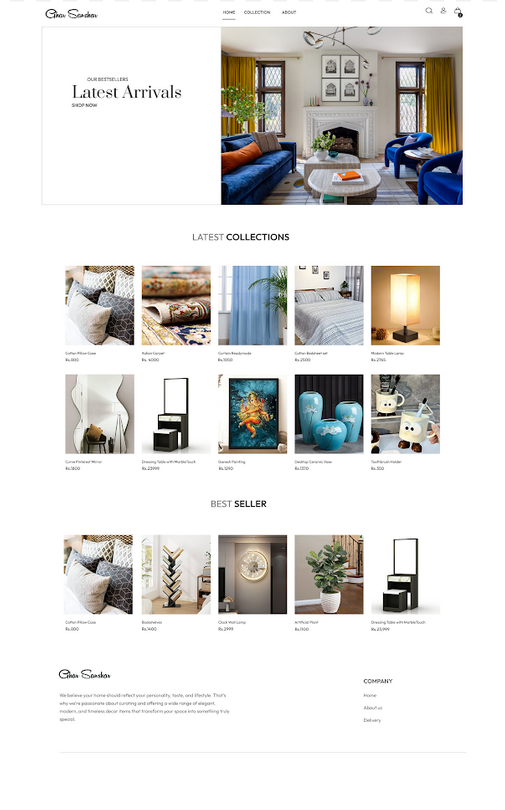


Figure 4Landing page

Authentication Pages: Includes registration and login forms with secure input validation and error-handling prompts for user feedback.

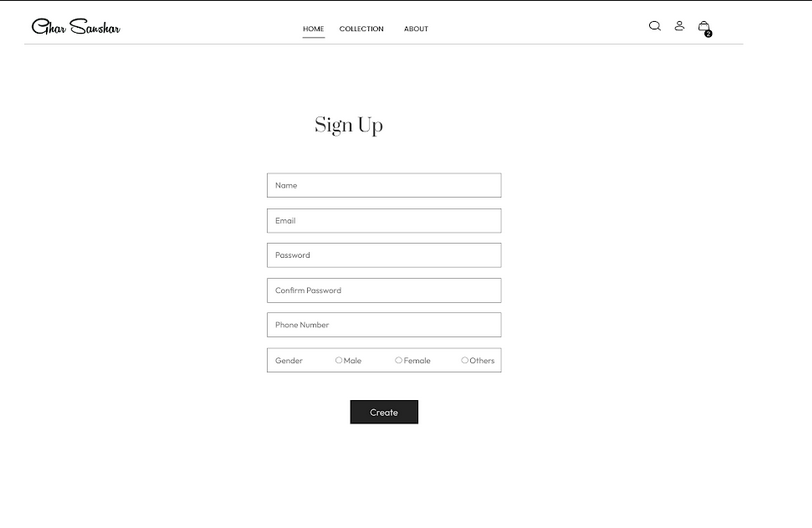


Figure 5authentication

Product Catalog: Displays products in a grid layout with filters (e.g., category, price) and sorting options (e.g., rating, price high-to-low).

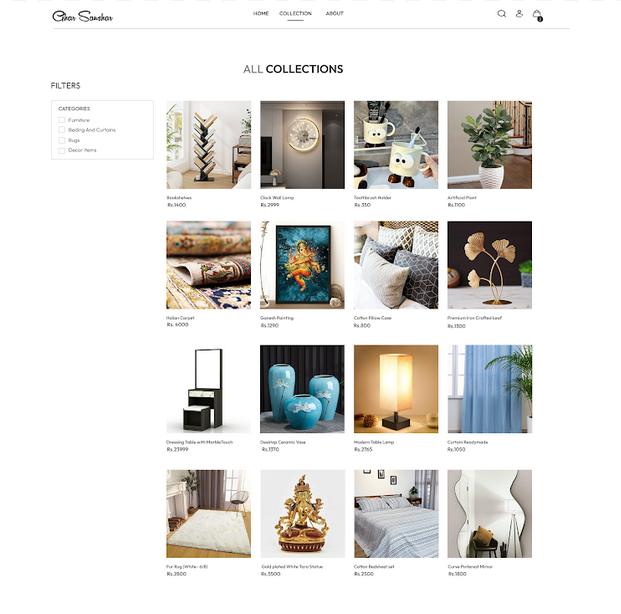


Figure 6 Product Catelog

Cart Page: Shows cart items, quantities, and total price, with options to update or remove items.

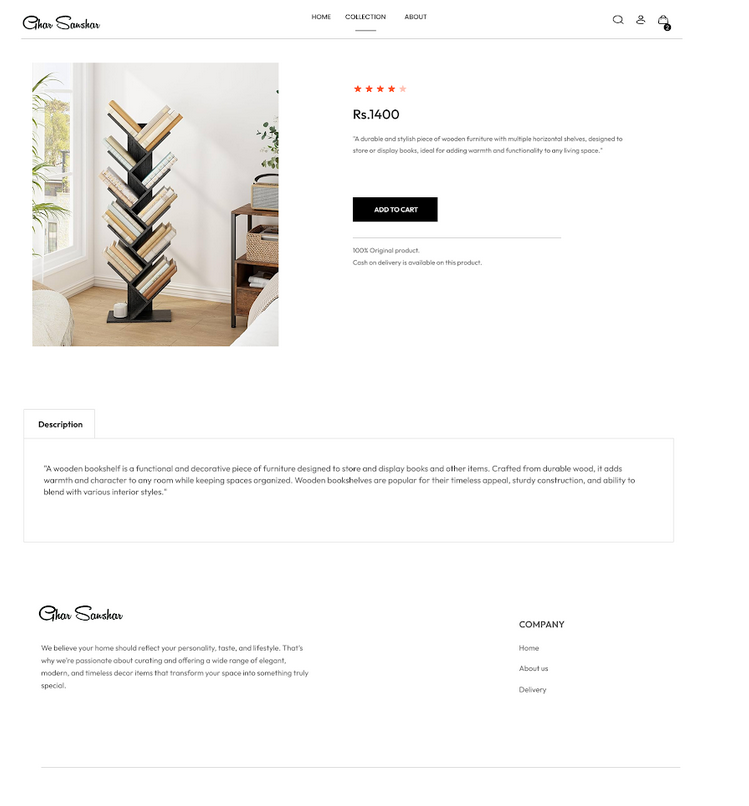


Figure 7Cart Page

# **System Architecture**

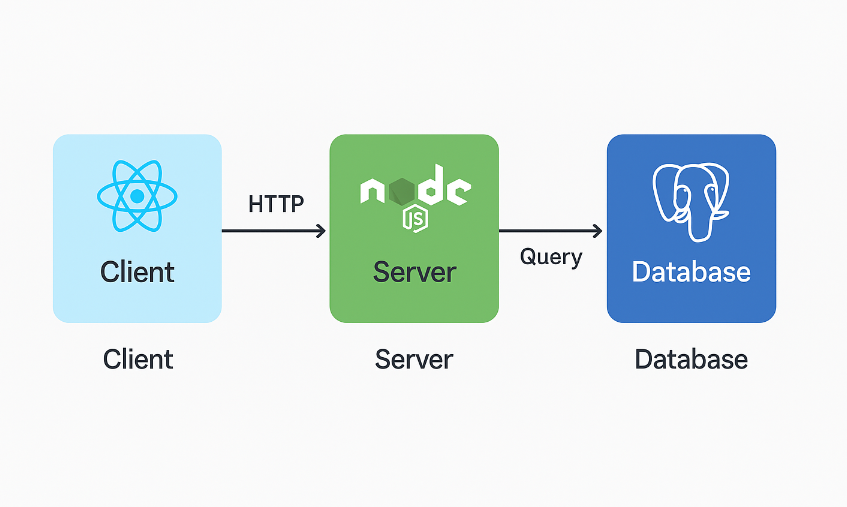


Figure 8System Architecture

Ghar Sanshar employs a three-tier architecture to ensure scalability, maintainability, and performance [(Wikipedia contributors, 2025)](#system):

## **Frontend (Client-side)**

**React.js + Vite**: Enables fast, component-based UI development for dynamic and responsive user interactions.

**React Router**: Manages navigation between pages (e.g., home, cart, admin panel) for a seamless user experience.

**React Testing Library**: Ensures component reliability through unit and integration tests, validating functionality and rendering.

**Tailwind CSS**: Provides responsive and consistent styling across devices, enhancing visual appeal and usability.

## **Backend (Server-side)**

**Node.js & Express.js**: These technologies were used to create the back end of the application. They manage all the request handling for the app, it also routes the APIs, and maintains any business logic in an efficient manner.

**JWT & bcrypt.js**: These technologies manage the user login and password storage. JWT securely handles user sessions and bcrypt hashes passwords so they are not stored in plain text.

**Multer**: Used to upload product images and it has basic validation to ensure that a file is uploaded and handled appropriately.

**REST API** endpoints: defined for user login, handling products, carts and orders. These endpoints are defined so that the front end and back end can communicate in a safe and reliable way.

## **Database Layer**

PostgreSQL: Data relational (users, products, cart, orders). Has strong ability to scale and data Integrity.

pgAdmin4: Helps in database management, structure, and help with query.

Data Integrity: Foreign keys, constraints, and index ensure it consistently runs efficiently with data operations.

# **Use Case Diagram**

The use case diagram illustrated the relationship between actors (Customer, Admin) and the system:

- Customer: register, login, navigate products, filter/sort products, manage shopping cart, checkout, view order history information.

- Admin: login, manage products (add, update, delete), manage orders (view, process order status), manage user account permission.

- System: authenticate users, respond to requests, store data, and grant limited secure access to features.

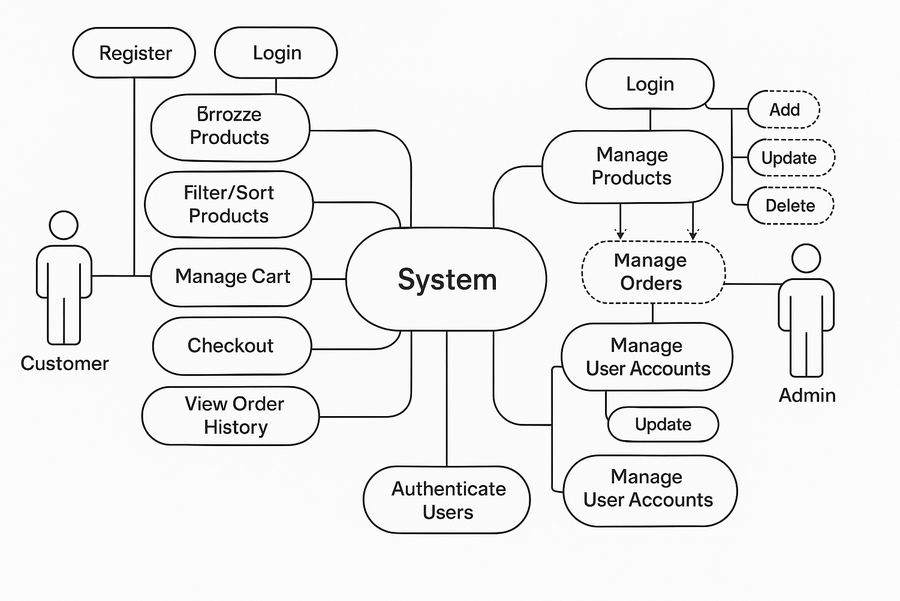


Figure 9Use case diagram

# **ER Diagram**

The Entity-Relationship (ER) shows the organization of the database structure of the e-commerce website which includes data types, of data organization, how each piece of information relates to one another, and how the site controls user function as-we-lay-out-the process from registration through order completion.

One of the main entities in the diagram is Users. This table documents relevant information about one specific user at a time which includes the user ID, full name, e-mail address, password, and role. The role is particularly important to separate users into ordinary shoppers or admin users. The role tells the site to either let the user shop and place orders or manage products, see orders, and do backend administration.

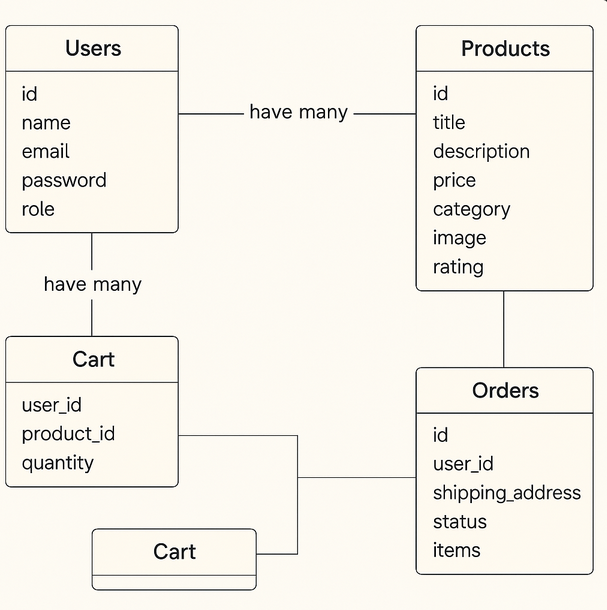


Figure 10Er diagram

# **Tools**

While developing, we used several helpful tools to help streamline and organize everything. Visual Studio Code was the code editor of choice. It supported the terminal, Git tracking, and extensions to help increase code writing and debugging speed. We especially liked how integrated version control support helped keep configuration and user stories organized when starting a project. It was nice to use the editor and not have to leave the workspace.

For version control and collaboration, we primarily used Git for source control and GitHub for code collaboration. Git helped us track every change made throughout the project. GitHub created a centralized place to push the code and manage the code if something changed. GitHub made it easy to go back to a previous version within the project if something broke in the process.

We used pgAdmin4 as the interface for working with the PostgreSQL database. It created a space for designing the schema, writing SQL queries, and managing the data easily. It was the main asset for testing and adjusting the layout of the database before designifying data processing.

Postman was used mostly to test the backend. Postman helped send what APIs we needed and verify if the endpoints worked.

Lastly, the React Testing Library was used to test the frontend to make sure React components acted that way that matched the expected user experience when a user interacted with the application.



Figure 11Tools

# **Technology**

## **Frontend Technologies**

React.js + Vite: Allows you to develop a foundation UI built on reusable components, while using hot module replacement to help maximize your time spent developing your application, especially when active development is happening.

React Router: Helps you add client-side routing with ease, so navigation to certain pages feel seamless.

Tailwind CSS: provides responsive utility-first styling to create a consistent and beautiful style.

Axios: easily makes asynchronous calls to backend APIs for getting, and posting data..

## **Backend Technologies**

Node.js + Express.js: Provides server-side logic for the REST API with routing and ultimately offers high levels of performance regardless of scaling.

JWT + bcrypt.js: Provides authentication with token-based access and stores users' password with hashing to maintain user information safely.

Multer: Handles file uploads (e.g., images of product) with verifiable validation as a form of security and efficiency.

## **Database Technologies**

PostgreSQL: Robust relational database for storing structured data with support for complex queries and scalability.

pgAdmin4: Provides an intuitive interface for managing PostgreSQL databases, schemas, and queries.

# **Implementation**

## **Frontend Implementation**

Login Component

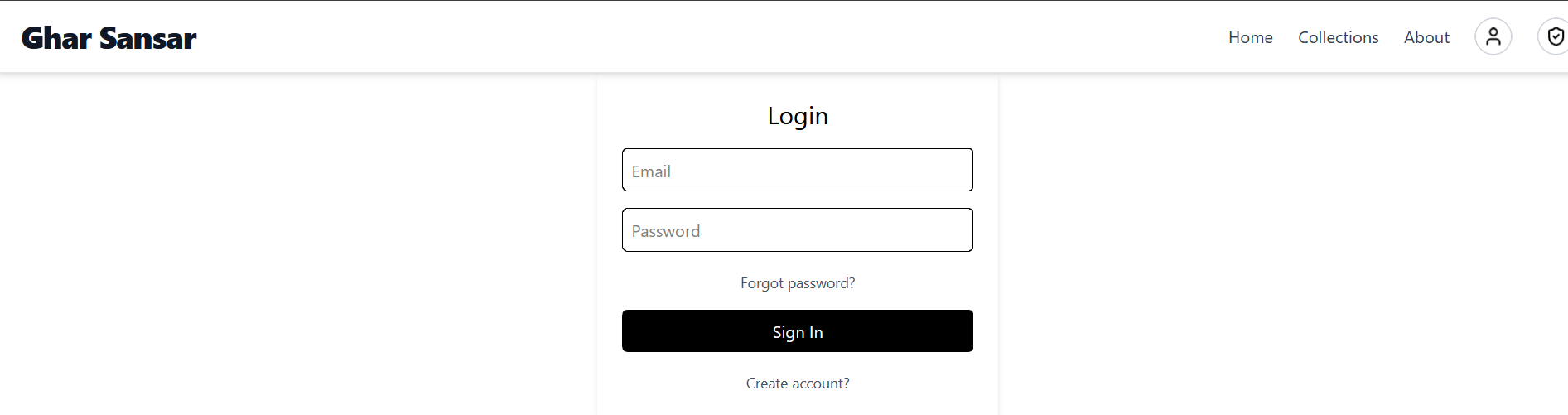


Figure 12login

Register Component



Figure 13Register

Product Catalog Component

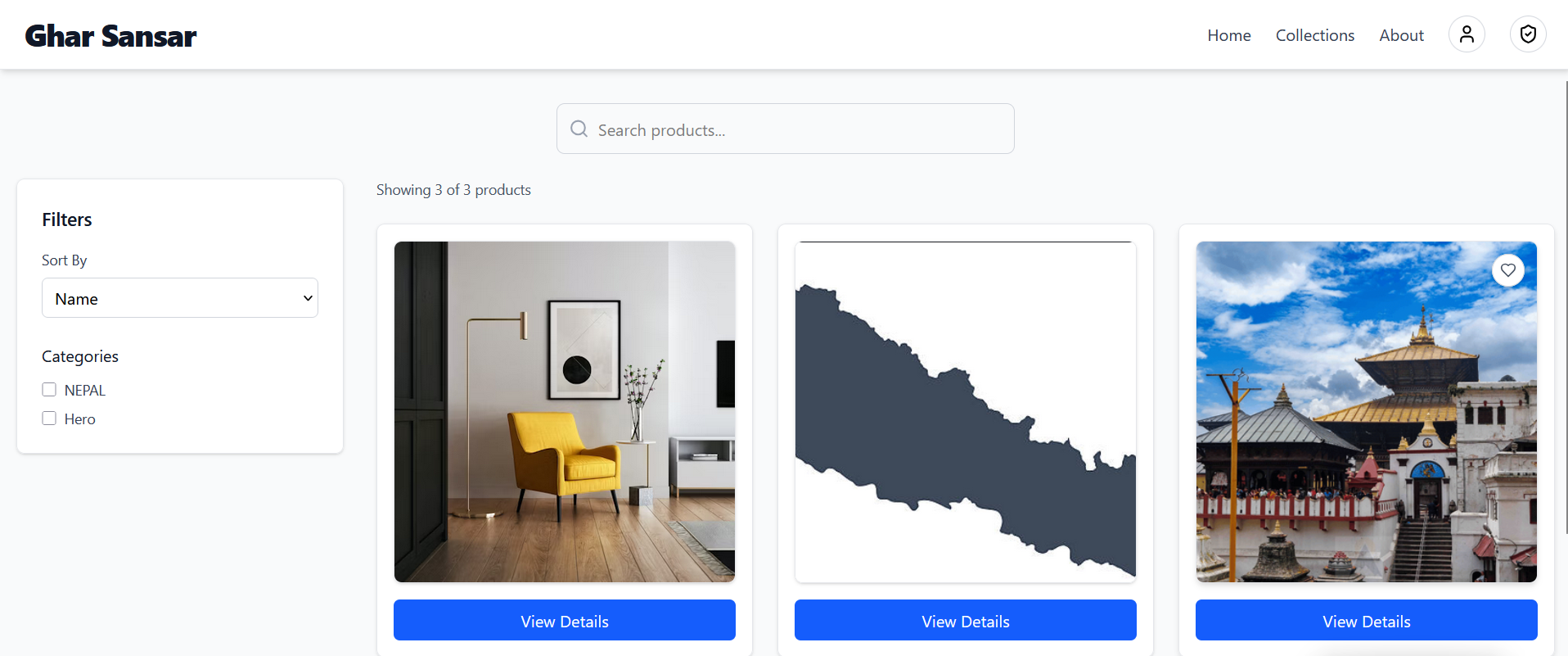


Figure 14Product Catelog

Cart Component

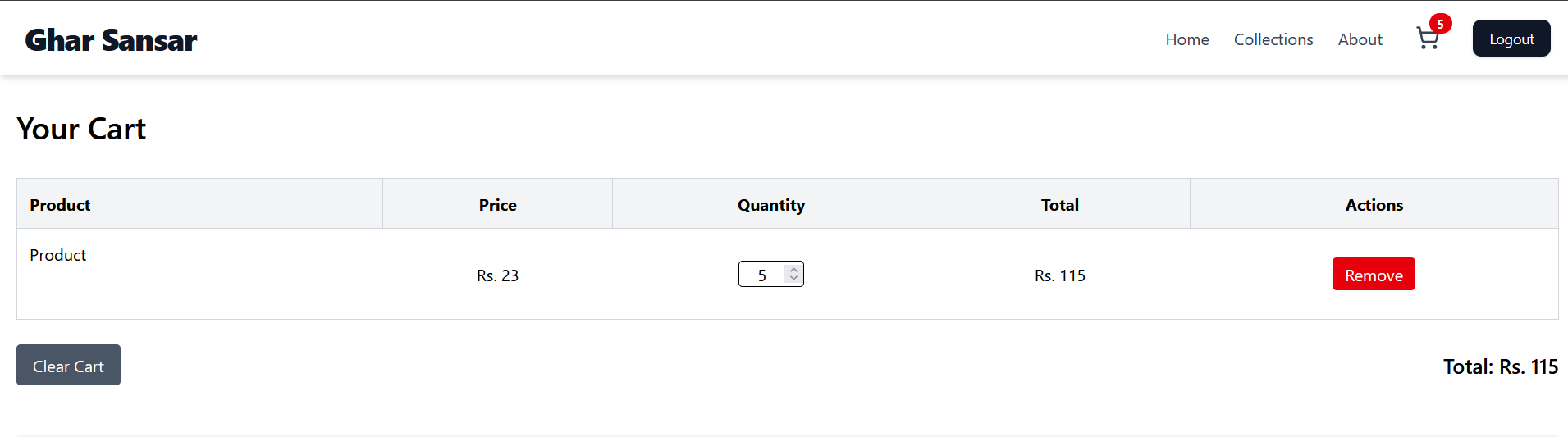


Figure 15Cart

Checkout Component

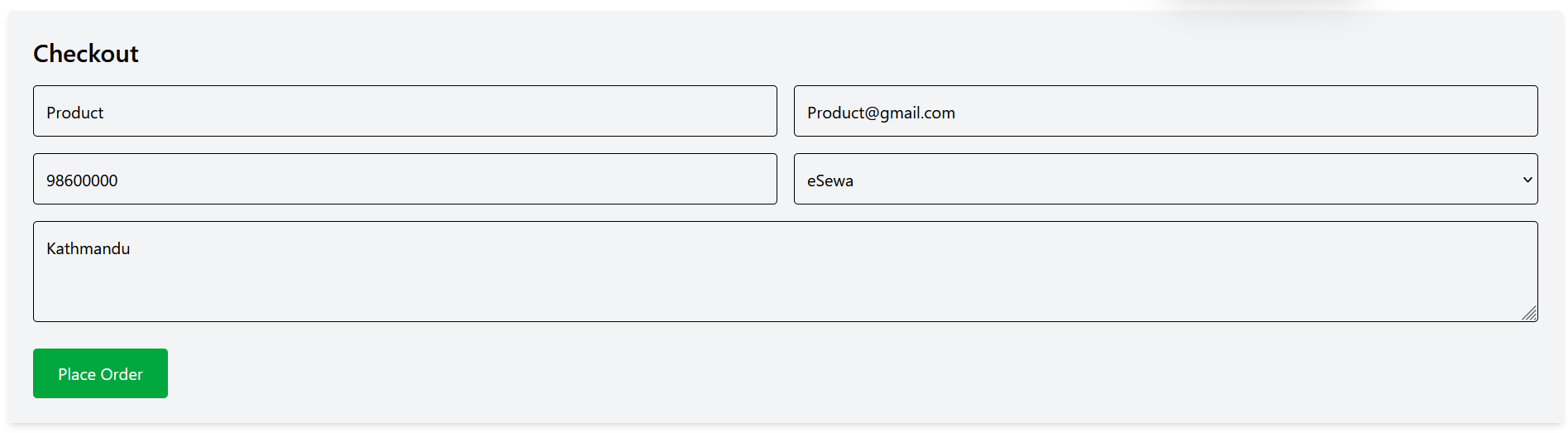


Figure 16Checkout

Admin Panel Component

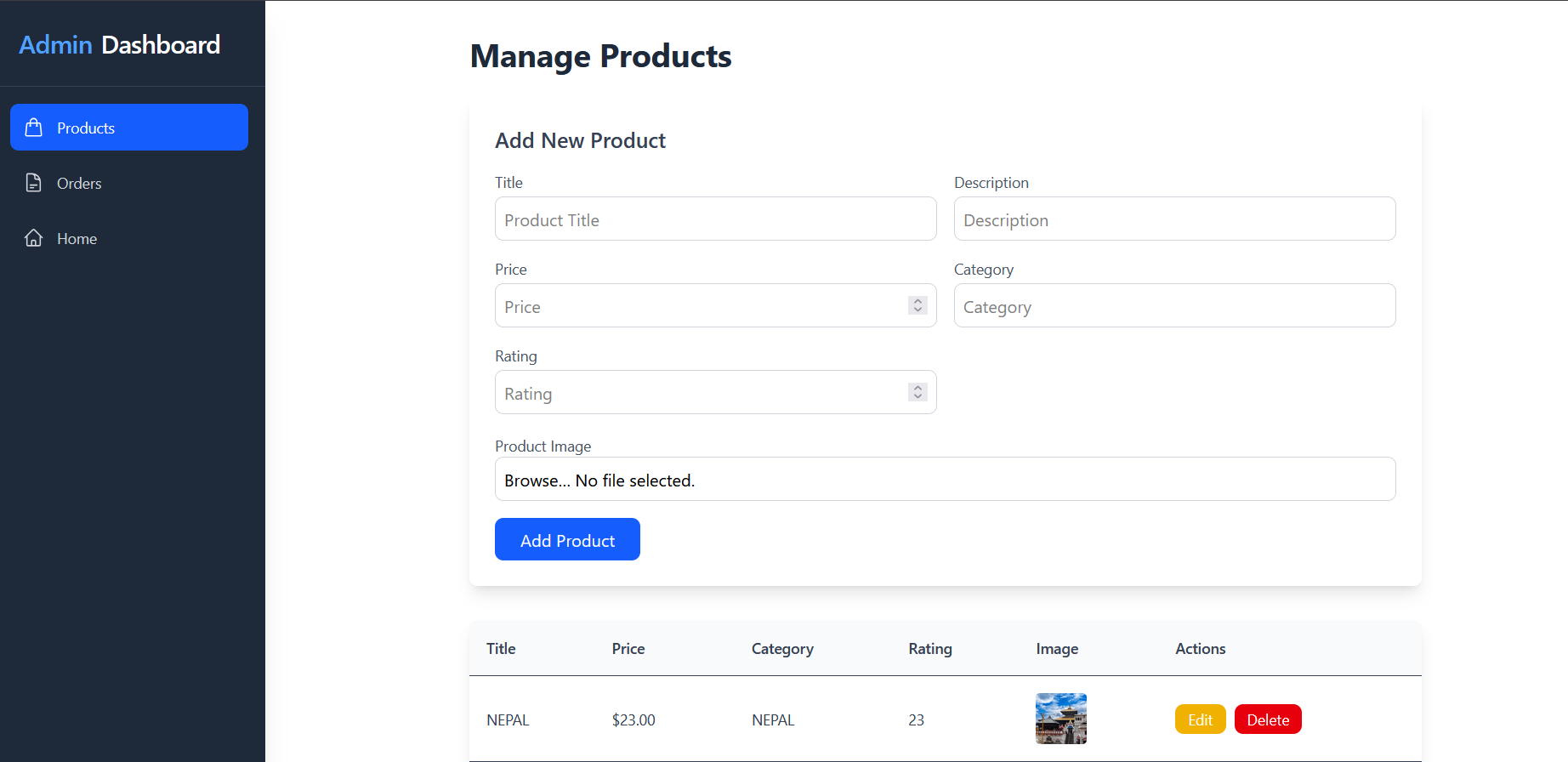


Figure 17product

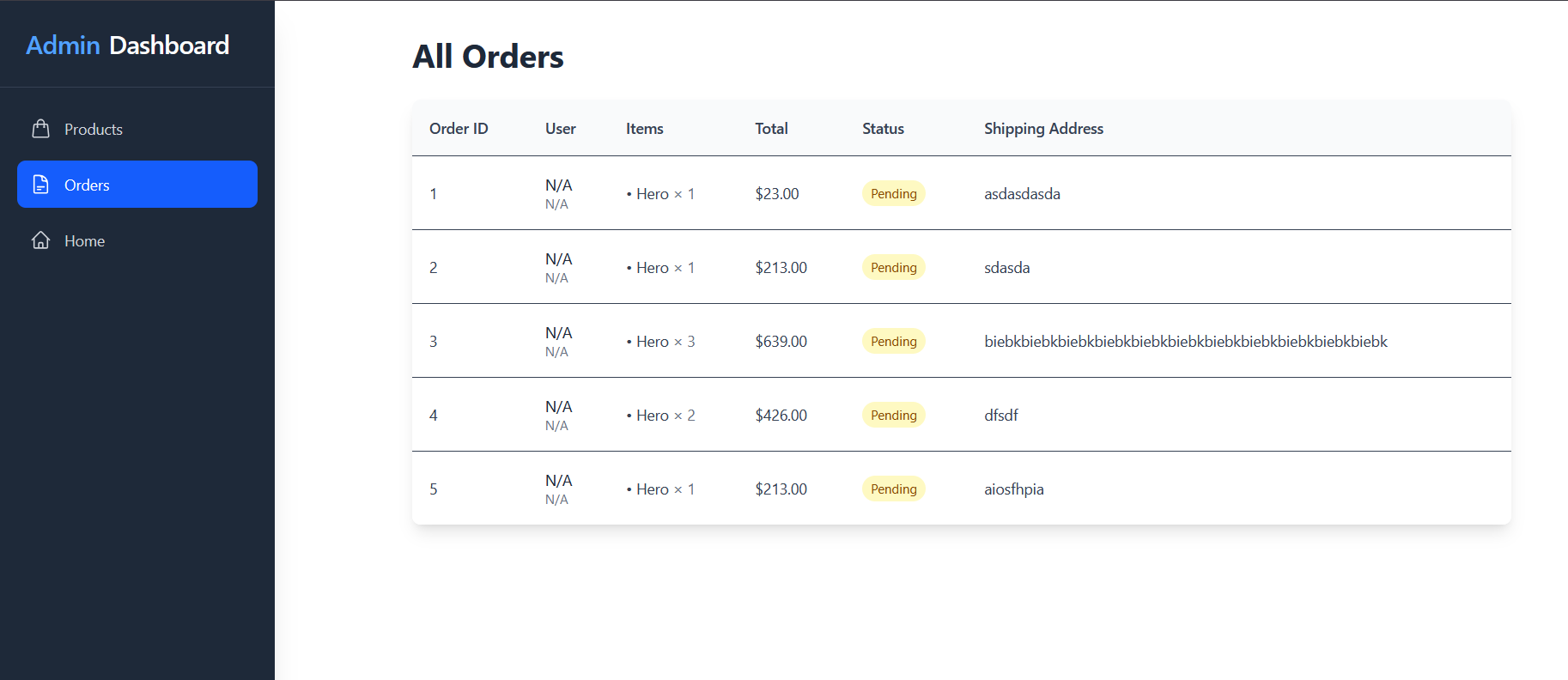


Figure 18Order

## Backend Implementation

Controllers:

Auth Controller

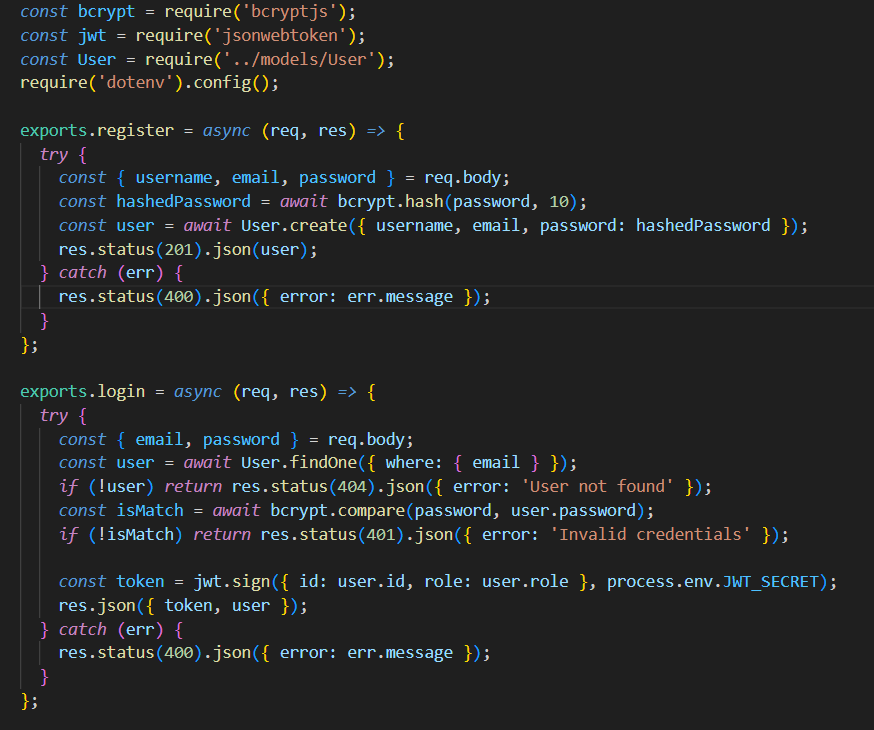


Figure 19auth controller

Product Controller

.

Figure 20product controller

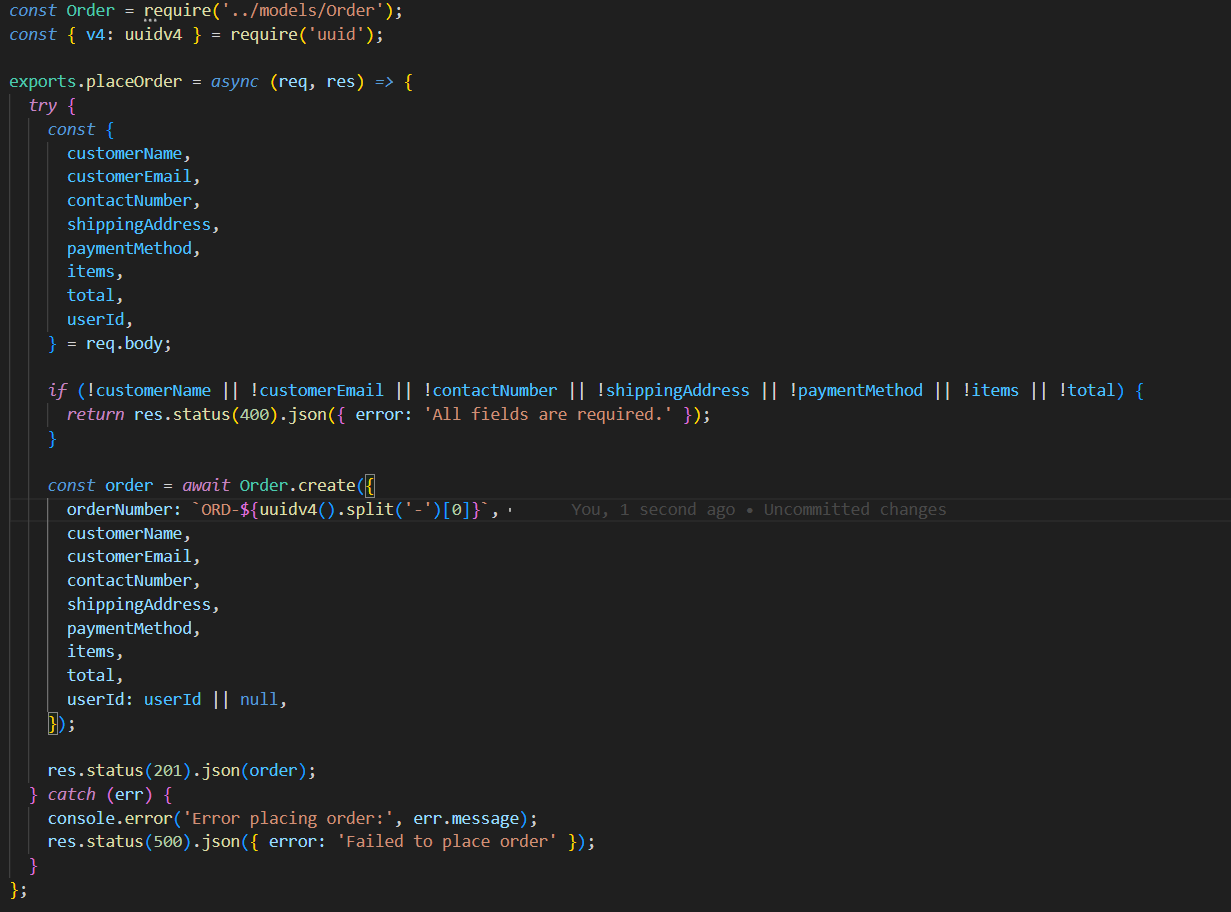
Order Controller (orderController.js): Processes order creation, retrieval, and status updates for customers and admins. 

Figure 21Order controller

Routes:

Auth Routes



Figure 22auth routes

Product Routes

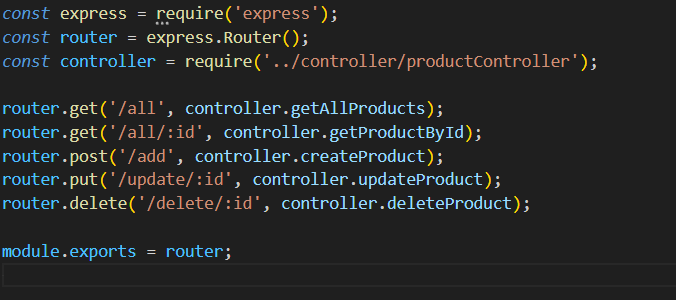


Figure 23product routes

Order Routes

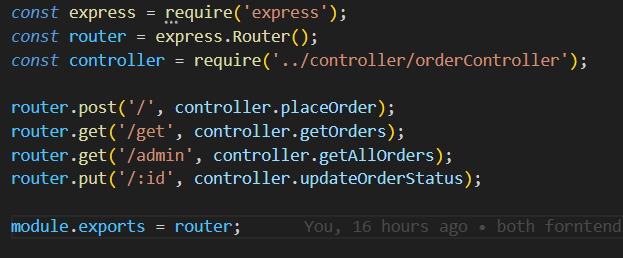


Figure 24order orutes

Middleware (authMiddleware.js).



Figure 25middlware

Main Server File (index.js).

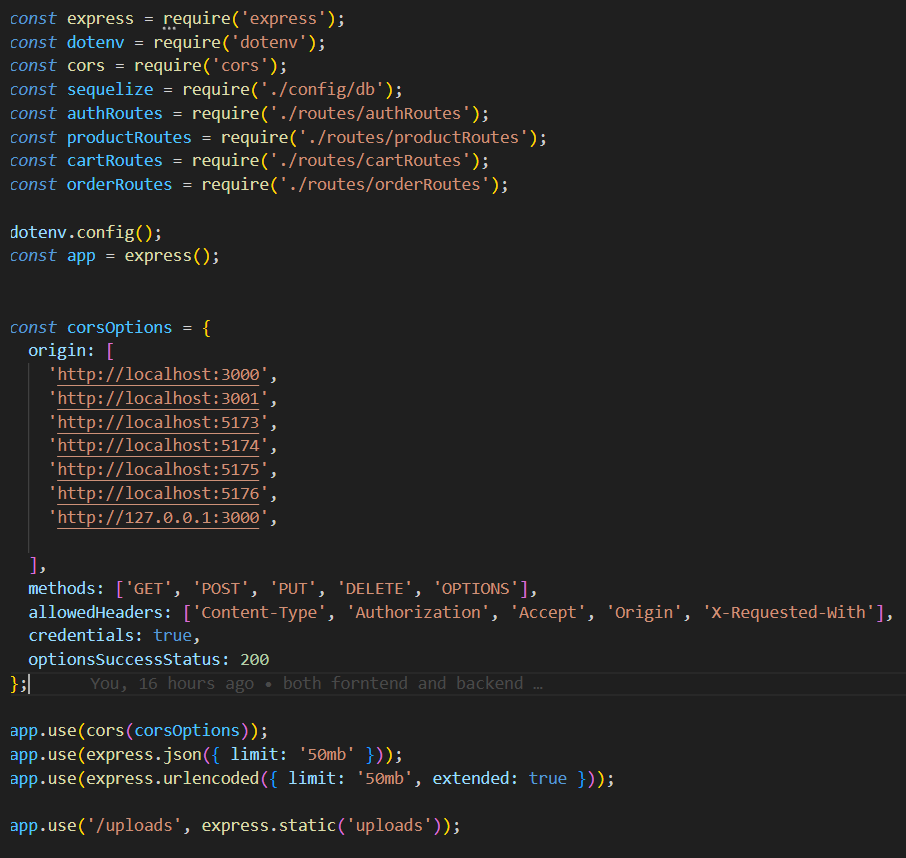


Figure 26server.js

## Database Implementation

PostgreSQL Schema: Tables were created for Users, Products, Cart and Orders.

Data integrity: A level of consistent referential integrity by using foreign keys and optionally constraints, along with reducing the time it takes to query the tables.

Sample Data: Populated tables with sample data to be useful for development and testing.

# **Testing and Evaluation**

The testing stage ensures Ghar Sansar meets functional and non-functional requirements in multi-faceted aspects:

**Unit Testing:**

Tools: React Testing Library, Mocha. Purpose: Validate individual components (ex. product list, login form) and backend functionality (ex. authentication logic). Example: Validate that the login component is valid given invalid credentials and displaying the correct error messages.

**Integration Testing:**

Tools: Postman,

Goal: Confirm interfacing of both front-end and back-end, along with the database interactions, to confirm the data flows correctly (e.g. product retrieval, order placement).

Ex: Verify cart checkout to order confirmation, clear cart, and save order.

**Security Testing:**

Objective: Verify for vulnerabilities like SQL injection, XSS, or open API endpoints.

Example: Verify JWT tokens are verified and inputs sanitized to prevent malicious attacks.

**Usability Testing:**

Methods: User feedback sessions, surveys.

Objective: Confirm intuitive navigation and ease of use with real user interactions.

Example: Gather feedback on ease of product catalog filter usability and ease of checkout process.

**Performance Testing:**

Tools: LoadNinja.

Objective: Conduct system performance testing under heavy load (e.g., 1000 concurrent users) to ensure rapid response times and scalability.

Example: Conduct product fetch API response time testing under heavy loads.

**User Acceptance Testing (UAT):**

Tools you could utilize are user surveys and feedback forms.

Scope - To assess whether the platform meets user objectives and business requirements.

Example: To confirm users are able to place orders and Admins are able to manage

listings. The evaluation will entail analyzing the outcomes of testing, user questions and requests, and an assessment (e.g. page load time, API response time, severities of vulnerabilities) to analyze where the need for improvements is and where the platform is fulfilling the requirements.

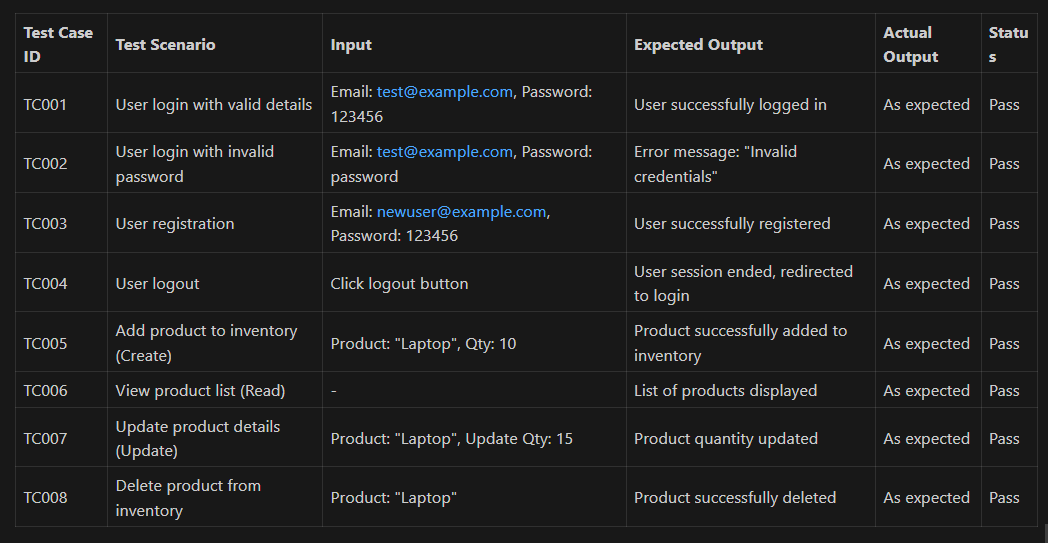


Figure 27Testing

# Future Work

For Ghar Sansar performance and user gain, the following must be built [(Systems, n.d.)](#future):

Mobile App Development

Create mobile applications that are capable of being built for both Android and iOS devices maximising user comkfort on their phones, and allowing users to experience the platform when they are away from their desk.

Payment Gateway Integration

Integrate secure payment gateways, like Paypal or Stripe, to enable real time transactions in multiple currencies and offer a smoother exit and payment experience for the user.

AI-Powered Recommendations

As part of a phased rollout, introduce machine learning to enable the system to make product suggestions based on user behaviour and purchasing preferences, to create a more tailored shopping experience.

Real-Time Order Tracking

Build a live tracking system so users can see where their order is at any given point in time from dispatch through to delivery.

Admin Analytics Dashboard

Build an admin dashboard that provides key insights such as sales reports, top products, and customer behaviour, to help with strategic decision-making.

Push Notifications

Implement notifications to keep users updated regarding their order status, promotions or new arrivals; driving engagement and returning users.

Wishlist Feature

Allow users to save their favorite products to a Wishlist to make future purchases easier, allowing them to make a plan or just to return at a later date without looking for the item again.

Ratings & Reviews system

Implement a system for customers to leave reviews and ratings on products. This helps build trust behind the buyer's decision and helps other customers with their purchasing decisions.

# **Conclusion**

Ghar Sansar wants to change how people buy home decor online by creating a safe, enjoyable, and scalable e-commerce platform. Ghar Sansar has an abundance of features designed to provide an alternative and give total ease of use, with a focus on both users and administrators. This includes, but is not limited to, items such as JWT-based user authentication, dynamic products, cart and checkout management, and an admin dashboard which gives power to administrators to control products, orders, and user management.

The platform was created using a modern technology stack where React.js with Vite was used for the front end, Node.js with Express.js written to run the backend logic, and our database was built using PostgreSQL. The project followed Agile development, with testing done regularly to monitor performance, usability, and reliability, as changes occurred in the build.

One of the major strengths of Ghar Sansar is the ability to tackle some very common e-commerce problems. Some features were made to specifically address problems such as cumbersome navigation or no security. Everything completed now is made to help the user experience and to stay functional, secure, and stable. Next, Ghar Sansar will also look to improve the value it provides further with future features, including developing a mobile app and AI-based product-recommendation task based on the data collected. Once these features have been added over the coming months and years, Ghar Sansar becomes a stronger and more reputable platform for.

# **References**

Wikipedia contributors. (2025, May 27). *Systems architecture*. Wikipedia. <https://en.wikipedia.org/wiki/Systems_architecture>

Systems, F. (n.d.). *FutureWork Systems*. FutureWork Systems. <https://futureworksystems.com/>

Krüger, G. (n.d.). *Non-Functional Requirements: tips, tools, and examples*. Perforce Software. <https://www.perforce.com/blog/alm/what-are-non-functional-requirements-examples>

AltexSoft Editorial Team. (2023, November 30). Functional and nonfunctional requirements: Specification and types. *AltexSoft*. https://www.altexsoft.com/blog/functional-and-non-functional-requirements-specification-and-types/

What is E-Commerce? Scope of e-commerce. (2025, May 26). *BootPoot*. https://bootpoot.tech/what-is-e-commerce-scope-of-e-commerce/

# **Appendix**

**YouTube:**

**Frontend:**

**Backend:**

**Figma:**