TABLE OF CONTENTS

1. PROJECT INTRODUCTION

Project Title: Shopfinity

Overview:
Shopfinity is designed to address the existing problem of an unreliable and fragmented online shopping experience. It aims to create a centralized e-commerce marketplace, providing a comprehensive solution that enables customers to find quality products at competitive prices, ensures secure transactions, and simplifies buying and selling processes. It includes product categorization, real-time order tracking, customer reviews, personalized recommendations, secure payment gateways, and a comprehensive seller and admin management dashboard.

2. FUNCTIONAL & NON-FUNCTIONAL REQUIREMENTS

FUNCTIONAL REQUIREMENTS:

User-related:

- Users can search products by name.
- Users can filter products (by category, price, and ratings).
- Users can securely reset passwords.
- Users can create and manage personal profile details.
- Users can add products to the cart and proceed to checkout.
- Users can select preferred payment methods.
- Users can securely enter payment details.
- Users can leave product reviews and ratings.

Seller-related:

- Sellers can add new products (images, descriptions, pricing).
- Sellers can mark customer orders as shipped.

Admin-related:

- Admin can manage user accounts (view, suspend, delete).
- Admin can manage and monitor product uploads to prevent unethical listings.

NON-FUNCTIONAL REQUIREMENTS

Security:

- Secure payment processing.
- Data encryption.
- Secure password reset mechanisms.

Performance:

- Fast and responsive user interface.
- Quick search and filter functionality.

Usability:

- Intuitive product search and filter options.
- Easy-to-use checkout and payment interface.
- Efficient profile and account management.

Reliability:

- Accurate order tracking and management.
- Consistent, reliable payment processes.
- Reliable user and product management processes.

3. IMPLEMENTED USER STORIES

Admin User Stories:

- As an admin, I want to manage users so that I can ensure the platform remains secure.
- As an admin, I want to manage products so that I can prevent unethical uploads.

Seller User Stories:

- As a seller, I want to add new products to my store so that customers can purchase them.
- As a seller, I want to mark orders as shipped so that customers know their orders are on the way.

Customer User Stories:

- As a user, I want to search for products by name so that I can quickly find what I need.
- As a user, I want to filter products by category, price, and ratings so that I can refine my search.
- As a user, I want to add products to my cart so that I can purchase multiple items at once.
- As a user, I want to proceed to checkout so that I can finalize my purchase.
- As a user, I want to select a preferred payment method so that I can complete my purchase conveniently.
- As a user, I want to securely enter my payment details so that my financial data remains protected.
- As a user, I want to leave a review and rate products so that I can share my experience with others.
- As a user, I want to securely reset my password if I forget it so that I can regain access to my account.
- As a user, I want to create and manage my profile so that I can personalize my shopping experience.

4. PRODUCT BACKLOG

#	USER STORY	ACC. CRITERIA	IMPOR.	ESTIMATE	TYPE
1	As a user, I want to search for products by name so that I can quickly find what I need.	The search bar allows users to enter product names and displays relevant results.	High	3	Search
2	As a user, I want to filter products by category, price, and	Filters display accurate results based	High	5	Search

	ratings so that I can refine my search results.	on selected criteria.			
3	As a user, I want to view product details, including images, price, and description so that I can make informed decisions.	Clicking a product shows its details page with all relevant information.	High	3	Report/View
4	As a user, I want to add products to my cart so that I can purchase multiple items at once.	Clicking "Add to Cart" places the item in the shopping cart.	High	3	Workflow
5	As a user, I want to update the quantity of products in my cart so that I can adjust my order before checkout.	Cart allows users to change product quantities before checkout.	High	3	Workflow
6	As a user, I want to remove items from my cart so that I can manage my purchases.	Users can delete items from the cart before checkout.	High	2	Workflow
7	As a user, I want to proceed to checkout so that I can finalize my purchase.	Clicking "Checkout" redirects users to payment and shipping details.	High	4	Workflow
8	As a user, I want to enter and save my shipping address so that I don't have to enter it every time.	Address can be saved in the user profile and auto-filled during checkout.	Medium	3	Manage data
9	As a user, I want to select a preferred payment method so that I can complete my purchase conveniently.	Users can choose from multiple payment options (credit/debit	High	4	Payment

		card, COD, etc.).			
10	As a user, I want to cancel my order before it is shipped so that I have control over my purchases.	Orders can be canceled before reaching the "Shipped" status.	Medium	4	Workflow
11	As a seller, I want to add new products to my store so that customers can purchase them.	Sellers can upload product images, descriptions, and prices.	High	5	Manage data
12	As a seller, I want to update product details so that I can modify prices and stock.	Sellers can edit product listings at any time.	High	3	Manage data
13	As a seller, I want to receive notifications when a new order is placed so that I can process it quickly.	Notifications appear in the seller dashboard for new orders.	High	3	Report/view
14	As a seller, I want to mark orders as shipped so that customers know their orders are on the way.	Sellers can update order status from "Processing" to "Shipped."	High	3	Workflow
15	As an admin, I want to manage users so that I can ensure the platform remains secure.	Admin can view, suspend, or delete user accounts if necessary.	High	5	Manage data
16	As an admin, I want to approve or reject seller applications so that only verified sellers can list products.	Sellers require admin approval before listing items.	High	4	Manage data
17	As a user, I want to create and manage my profile so that I can personalize my	Users can update name, email, and contact details	Medium	3	Manage data

	shopping experience.	in their profile.			
18	As an admin, I want to manage products so that I can prevent unethical uploads.	Admin dashboard displays all transactions and suspicious activity alerts.	High	5	Report/view

5. SPRINT BACKLOGS

5.1 Sprint 1 Backlog

User Registration & Login

Users and sellers can create accounts and log in securely.

Priority: High

Product Search & Filtering

Users can search for products by name and apply filters.

Priority: High

View Product Details

Users can view product descriptions, images, and pricing.

Priority: High

Shopping Cart Management

Users can add and remove items from their cart.

Priority: High

Checkout & Payment Processing

Users can enter payment details and place orders.

Priority: High

Seller Product Management

Sellers can add, update, and remove products.

Priority: High

Basic Order Management (User & Seller)

Users can track orders, and sellers get notifications of new orders.

Priority: High

5.2 Sprint 2 Backlog

Apply Discount Codes at Checkout

Users can enter discount codes to receive price reductions.

Priority: Medium

Save Payment Methods for Faster Checkout

Users can securely store payment details for future use.

Priority: Medium

View Order History

Users can view past orders and their statuses.

Priority: Medium

Seller Sales Reports

Sellers can view revenue, order trends, and analytics.

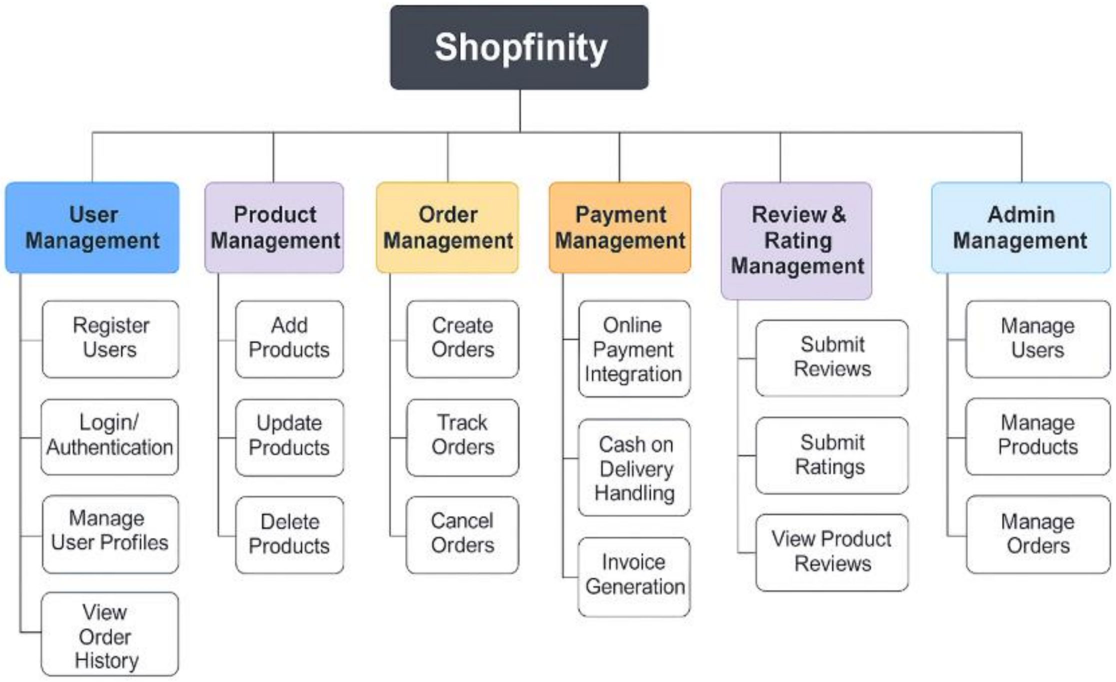
Priority: Medium

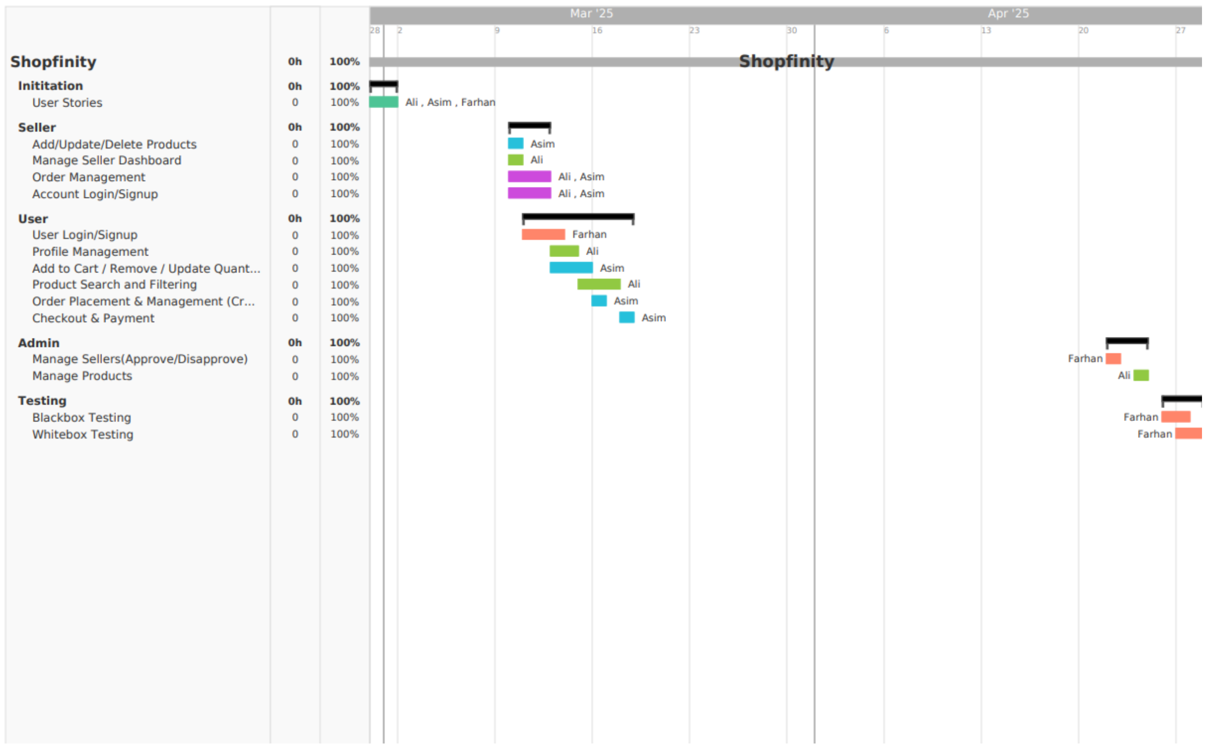
Admin User Management

Admins can block/unblock users as needed.

Priority: High

6. PROJECT PLAN





Shopfinity	**0h**	**100%**
Initiation	**0h**	**100%**
User Stories	0	100%
Seller	**0h**	**100%**
Add/Update/Delete Products	0	100%
Manage Seller Dashboard	0	100%
Order Management	0	100%
Account Login/Signup	0	100%
User	**0h**	**100%**
User Login/Signup	0	100%
Profile Management	0	100%
Add to Cart / Remove / Update Quant...	0	100%
Product Search and Filtering	0	100%
Order Placement & Management (Cr...	0	100%
Checkout & Payment	0	100%
Admin	**0h**	**100%**
Manage Sellers(Approve/Disapprove)	0	100%
Manage Products	0	100%
Testing	**0h**	**100%**
Blackbox Testing	0	100%
Whitebox Testing	0	100%

7. ARCHITECTURE DIAGRAM

Identifying Subsystems

Shopfinity
This is the main package, representing the entire Shopfinity e-commerce application.

Frontend:

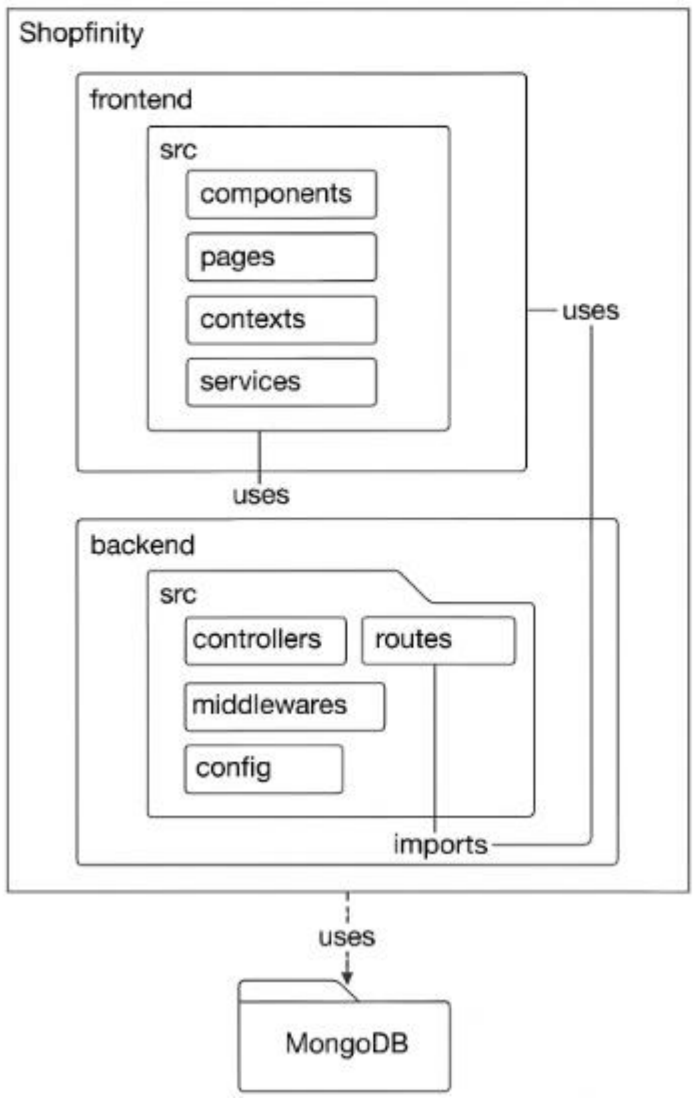
This package encapsulates all the **client-side code**, built with **React.js** and Vite.

- **src:**
  The source directory, a standard convention in React projects, containing the main application code.
- **components:**
  This sub-package holds the reusable UI components of the application (e.g., Header, Footer, ProductCard, CartItem).
- **pages:**
  This sub-package contains major screens/pages like HomePage, ProductDetailsPage, CartPage, AdminDashboardPage.
- **contexts:**
  Provides global state management for things like user authentication, cart management, and order tracking.
- **services:**
  Handles API interactions, encapsulating all Axios HTTP requests to the backend.
- **assets:**
  Stores images, icons, and other static files used throughout the frontend.
- **tests:**
  (If present) Includes unit and integration tests for frontend components and pages.
- **Uses (Dependency Arrow to backend):**
  The src package interacts with the backend package via RESTful APIs to fetch and manipulate product, user, and order data.
- **Imports (Dependency Arrow from components to pages and contexts):**
  Components are imported by pages and contexts to build user interfaces and provide functionalities.

Backend:

This package contains the **server-side code**, built with **Node.js**, **Express.js**, and **MongoDB**.

- **src:**
  The main source directory for backend logic.
- **controllers:**
  Contains the business logic for various features like user management, order handling, and product management.
- **routes:**
  Defines all RESTful API endpoints (e.g., /api/products, /api/orders, /api/users) and connects them to the appropriate controllers.
- **models:**
  Defines the Mongoose schemas and data models for the MongoDB collections (e.g., User, Product, Order, Review).
- **middlewares:**
  Contains middleware functions for authentication (JWT), error handling, and validation.
- **config:**
  Configuration files for connecting to MongoDB, handling environment variables, and setting up cloud services like Cloudinary.
- **tests:**
  (If present) Includes unit and integration tests for backend logic like controllers and models.
- **Uses (Dependency Arrow to models and routes):**
  The backend's routes depend on models for database interaction, and controllers for business logic processing.
- **Imports (Dependency Arrow from routes to models and controllers):**
  The routes package imports and uses models and controllers to handle the business workflows and serve responses.



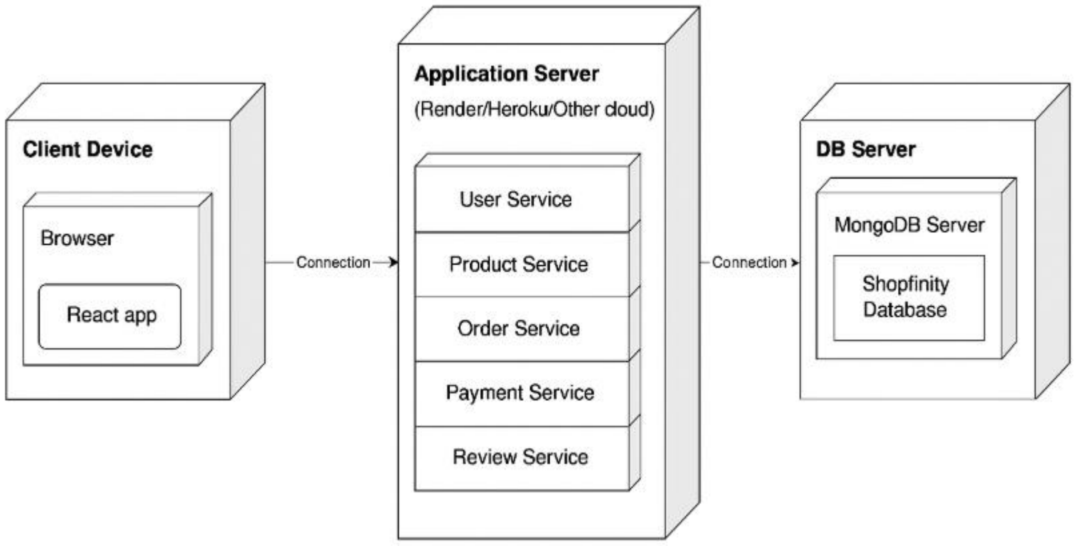
Architecture Styles

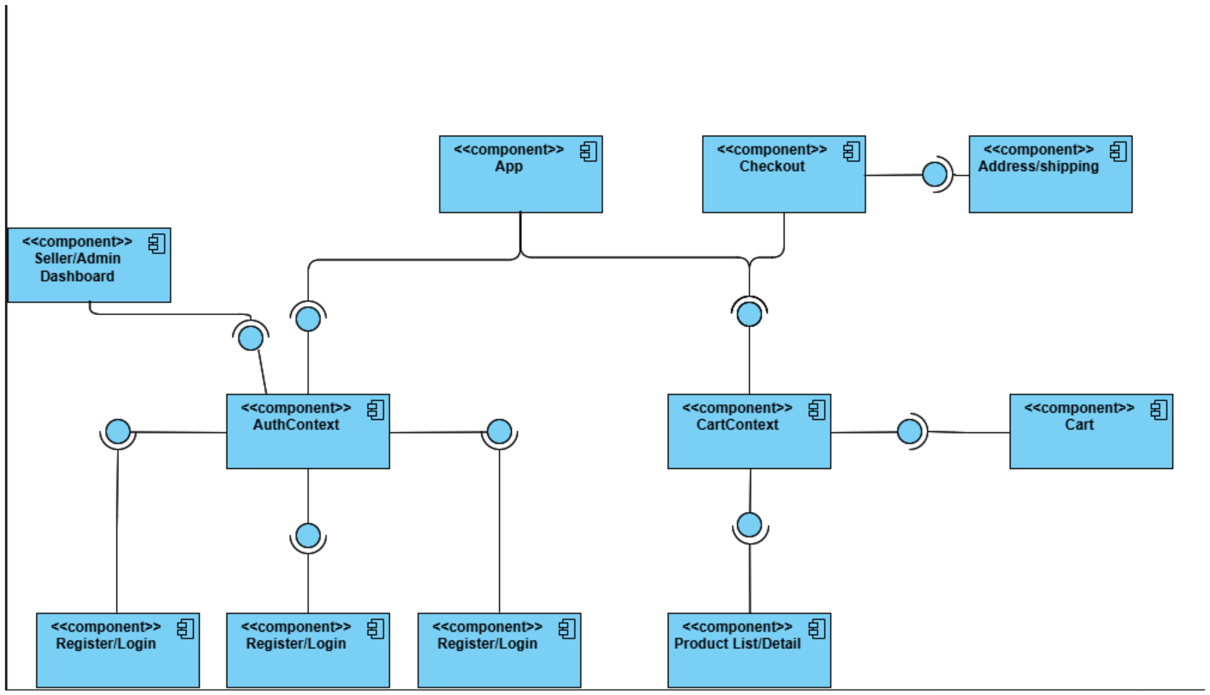
The **Shopfinity** web-based e-commerce application is built using the **MERN stack (MongoDB, Express.js, React.js, Node.js)** and follows the **Model-View-Controller (MVC)** architectural style.

- **Model:**
  The data layer is handled using **MongoDB**, where all e-commerce-related data (e.g., users, products, orders, reviews) are stored. The models define the structure of the database collections and are used to interact with the database through Mongoose schemas.

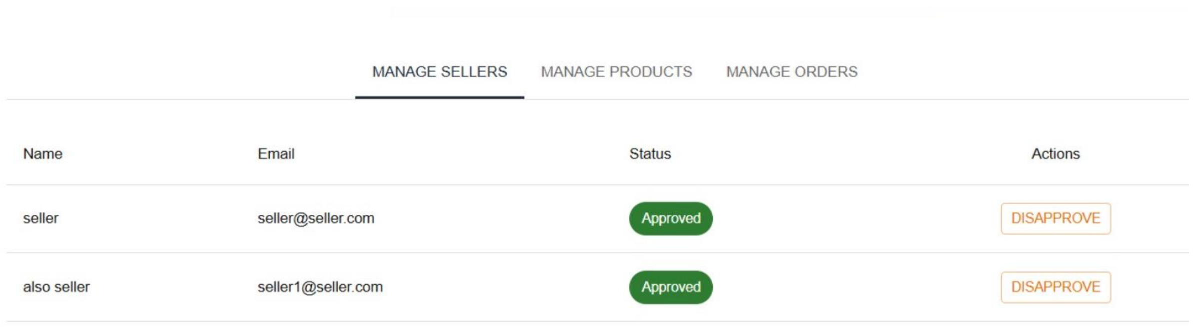
- **View:**
  The frontend is developed using **React.js**, which acts as the View in the MVC pattern. React is responsible for presenting dynamic and responsive user interfaces, such as product listings, shopping carts, and user dashboards. It communicates with the backend via RESTful APIs to fetch and display data.
- **Controller:**
  The controllers are implemented in **Express.js** within the **Node.js** backend. Controllers handle the business logic, processing incoming HTTP requests, interacting with the models (database layer), and sending appropriate responses back to the frontend.

By using MVC, **Shopfinity** clearly separates concerns, making the system more organized, maintainable, scalable, and testable. Each part (Model, View, Controller) is independently developed and maintained, improving collaboration among team members and simplifying future enhancements of the application.

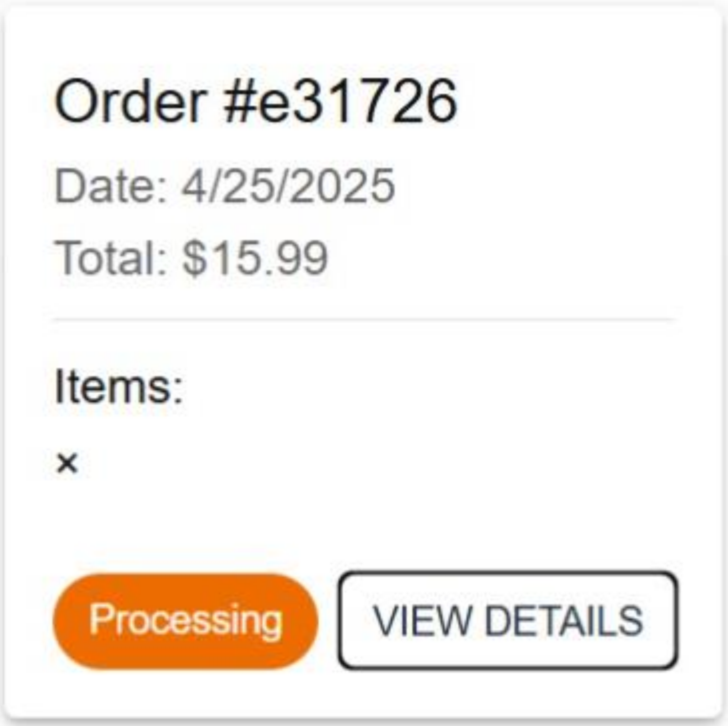



<<component>> App
<<component>> Checkout
<<component>> Address/shipping
<<component>> Seller/Admin Dashboard
<<component>> AuthContext
<<component>> CartContext
<<component>> Cart
<<component>> Register/Login
<<component>> Register/Login
<<component>> Register/Login
<<component>> Product List/Detail

8. ACTUAL IMPLEMENTATION SCREENSHOTS



Name	Email	Status	Actions
seller	seller@seller.com	Approved	DISAPPROVE
also seller	seller1@seller.com	Approved	DISAPPROVE



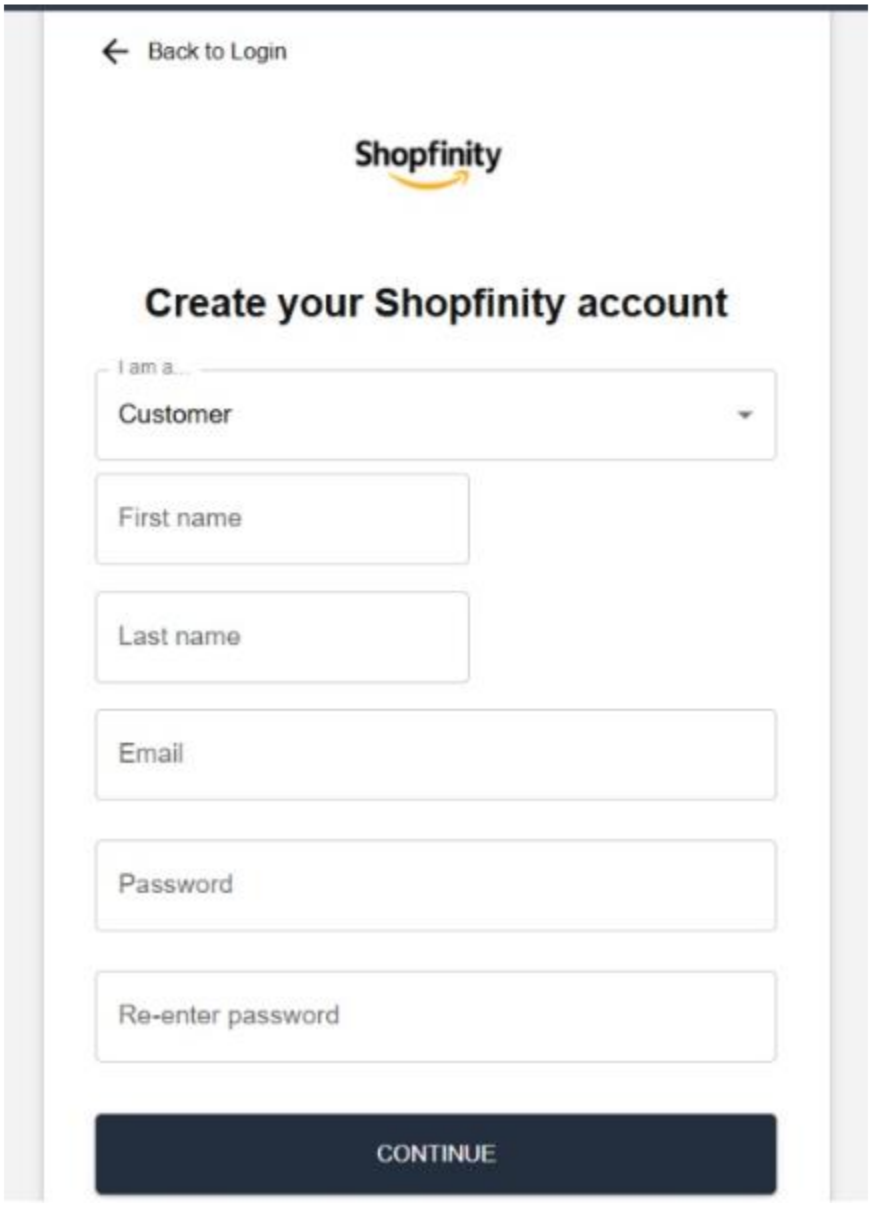
Order #e31726

Date: 4/25/2025

Total: $15.99

Items:

×



← Back to Login

Shopfinity

Create your Shopfinity account

I am a...

Customer

First name

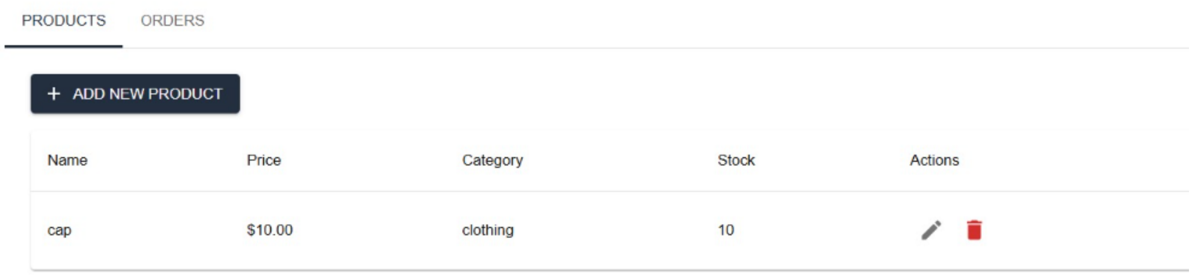
Last name

Email

Password

Re-enter password

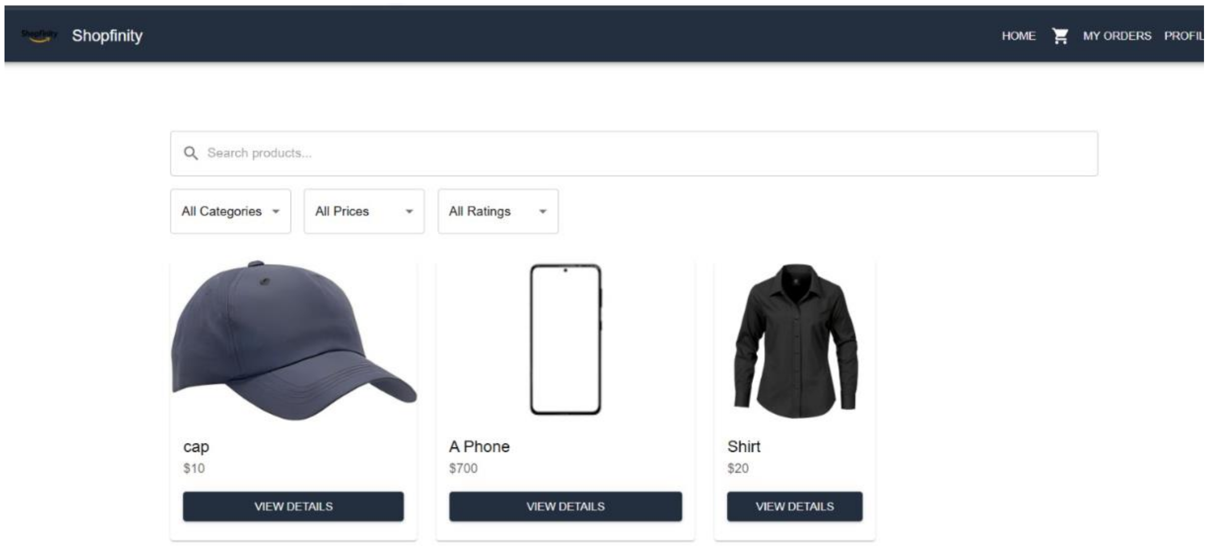
CONTINUE

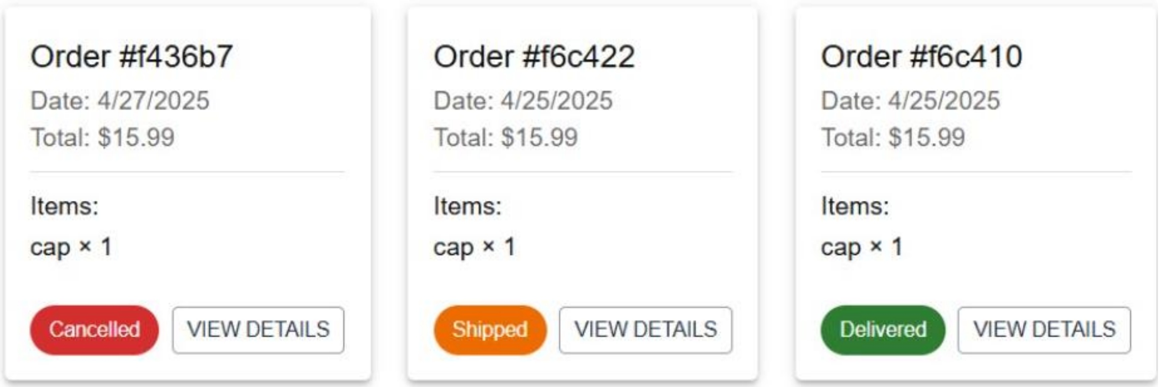


PRODUCTS ORDERS

+ ADD NEW PRODUCT

Name	Price	Category	Stock	Actions
cap	$10.00	clothing	10	✎ 🗑



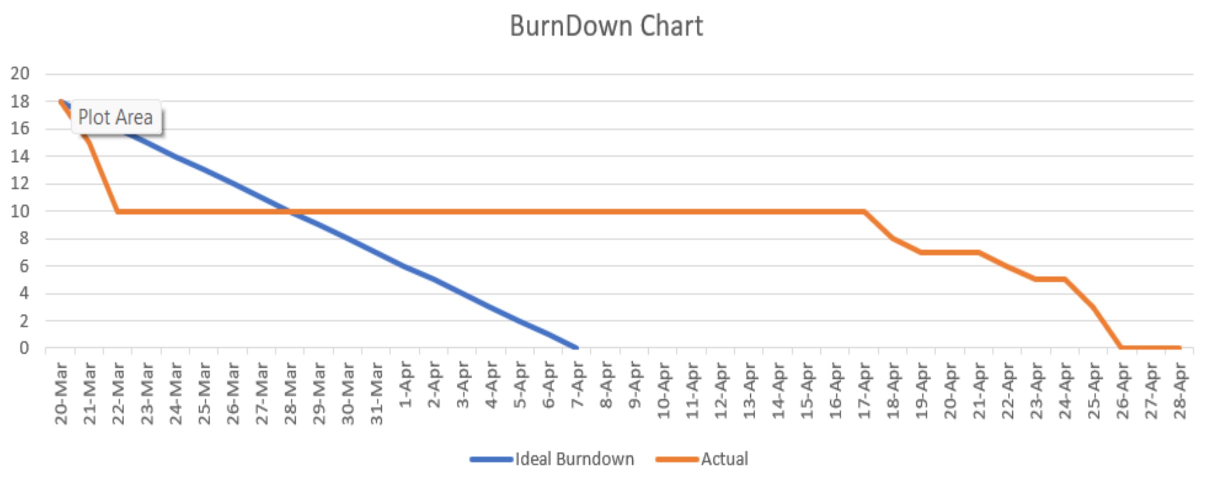


Order #f436b7
Date: 4/27/2025
Total: $15.99
Items:
cap × 1

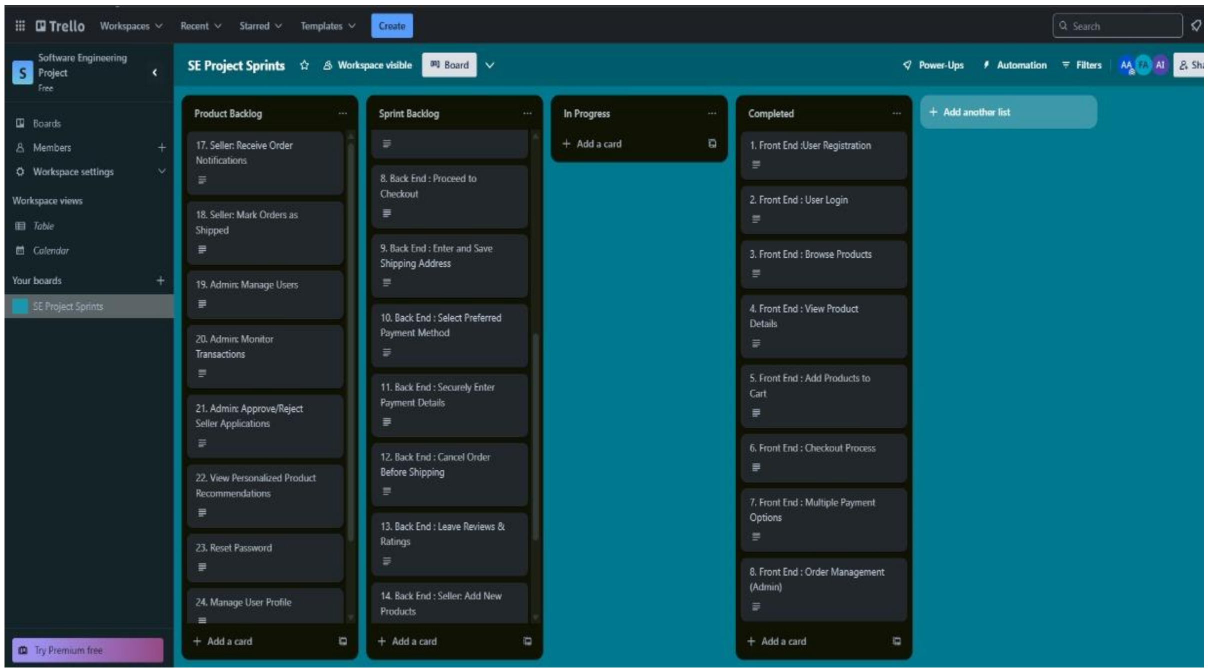
Order #f6c422
Date: 4/25/2025
Total: $15.99
Items:
cap × 1

Order #f6c410
Date: 4/25/2025
Total: $15.99
Items:
cap × 1

9. PRODUCT BURN DOWN CHART:



10. TRELLO BOARD SCREENSHOTS



Product Backlog

- 17. Seller: Receive Order Notifications
- 18. Seller: Mark Orders as Shipped
- 19. Admin: Manage Users
- 20. Admin: Monitor Transactions
- 21. Admin: Approve/Reject Seller Applications
- 22. View Personalized Product Recommendations
- 23. Reset Password
- 24. Manage User Profile

+ Add a card

Sprint Backlog

- 8. Back End : Proceed to Checkout
- 9. Back End : Enter and Save Shipping Address
- 10. Back End : Select Preferred Payment Method
- 11. Back End : Securely Enter Payment Details
- 12. Back End : Cancel Order Before Shipping
- 13. Back End : Leave Reviews & Ratings
- 14. Back End : Seller: Add New Products

+ Add a card

In Progress

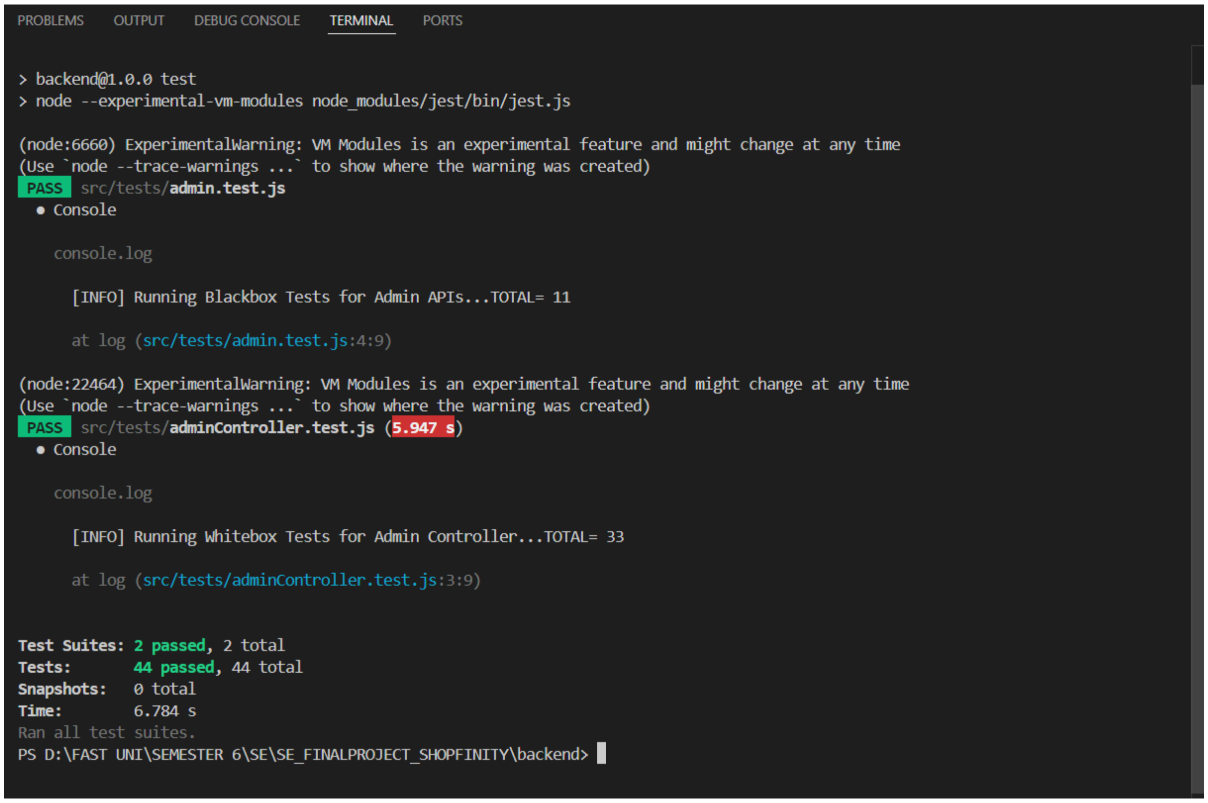
+ Add a card

Completed

- 1. Front End :User Registration
- 2. Front End : User Login
- 3. Front End : Browse Products
- 4. Front End : View Product Details
- 5. Front End : Add Products to Cart
- 6. Front End : Checkout Process
- 7. Front End : Multiple Payment Options
- 8. Front End : Order Management (Admin)

+ Add a card

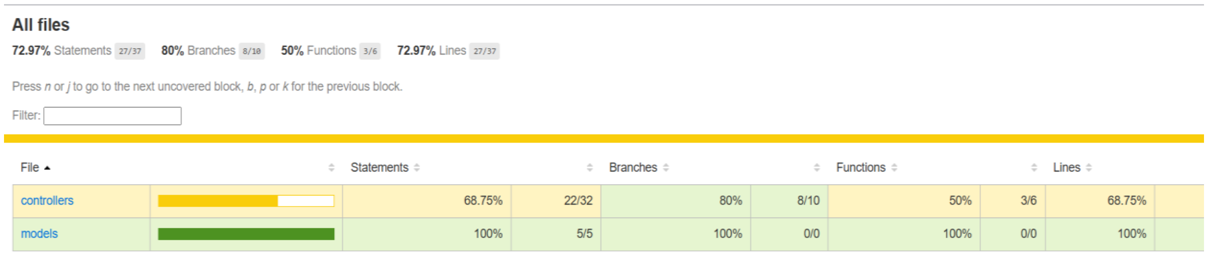
11. TEST CASES-BLACK BOX



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
> backend@1.0.0 test
> node --experimental-vm-modules node_modules/jest/bin/jest.js

(node:6660) ExperimentalWarning: VM Modules is an experimental feature and might change at any time
(Use `node --trace-warnings ...` to show where the warning was created)
 PASS  src/tests/admin.test.js
  ● Console

    console.log

      [INFO] Running Blackbox Tests for Admin APIs...TOTAL= 11

      at log (src/tests/admin.test.js:4:9)

(node:22464) ExperimentalWarning: VM Modules is an experimental feature and might change at any time
(Use `node --trace-warnings ...` to show where the warning was created)
 PASS  src/tests/adminController.test.js (5.947 s)
  ● Console

    console.log

      [INFO] Running Whitebox Tests for Admin Controller...TOTAL= 33

      at log (src/tests/adminController.test.js:3:9)


Test Suites: 2 passed, 2 total
Tests:       44 passed, 44 total
Snapshots:   0 total
Time:        6.784 s
Ran all test suites.
PS D:\FAST UNI\SEMESTER 6\SE\SE_FINALPROJECT_SHOPFINITY\backend> 
```

12. TEST CASES-WHITE BOX



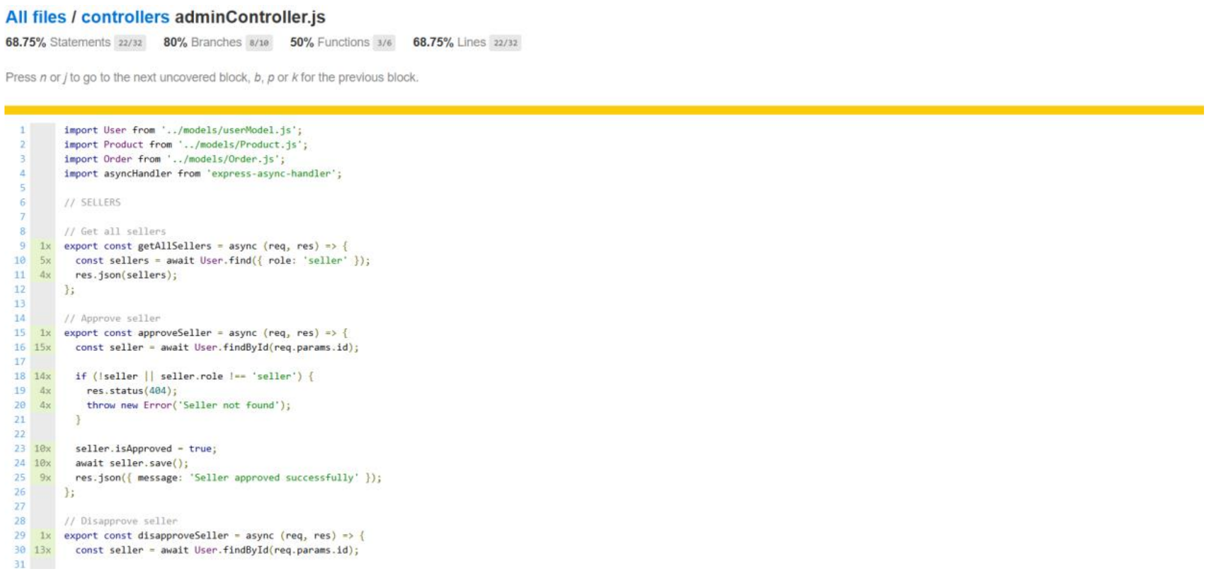
All files

72.97% Statements 27/37 **80%** Branches 8/10 **50%** Functions 3/6 **72.97%** Lines 27/37

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File ▲		Statements		Branches		Functions		Lines	
controllers		68.75%	22/32	80%	8/10	50%	3/6	68.75%	
models		100%	5/5	100%	0/0	100%	0/0	100%	

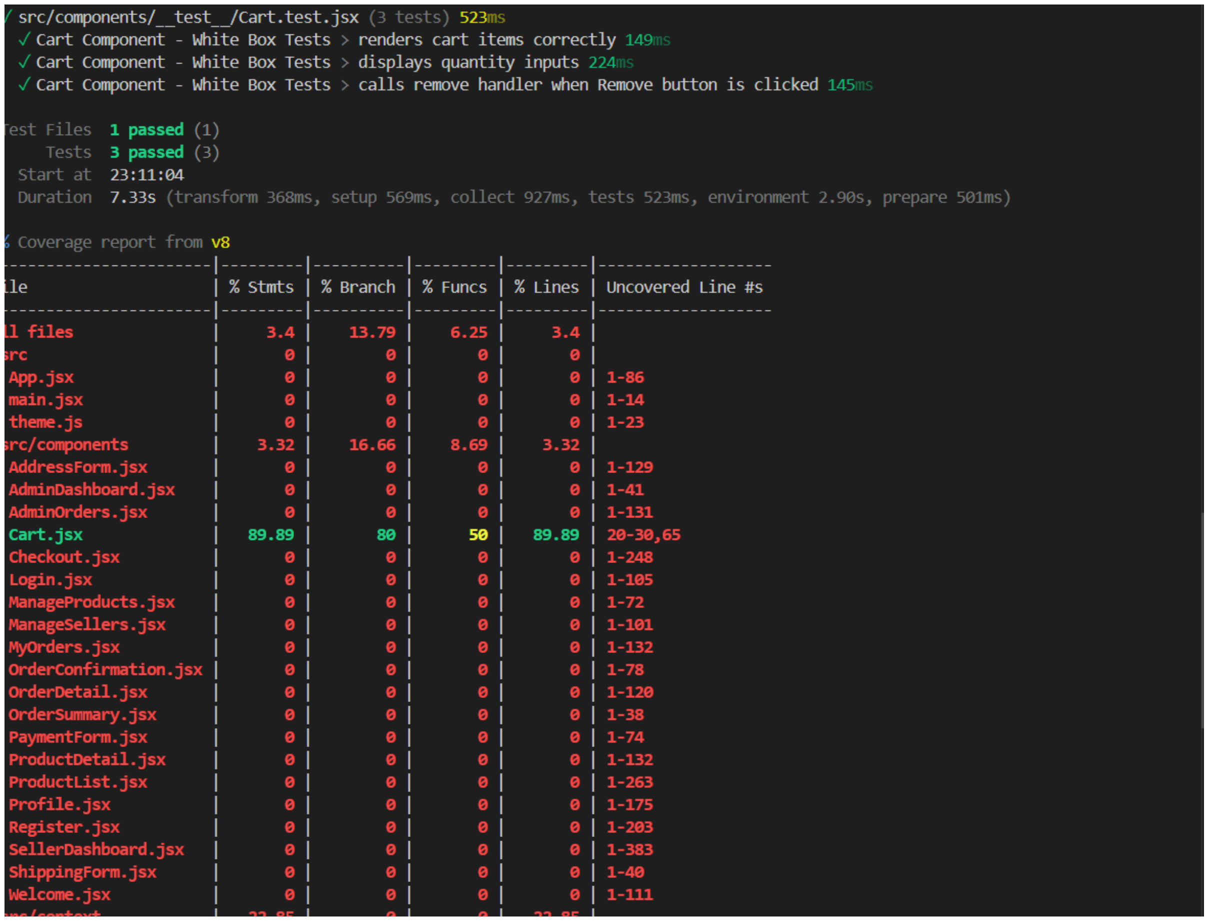


All files / controllers adminController.js

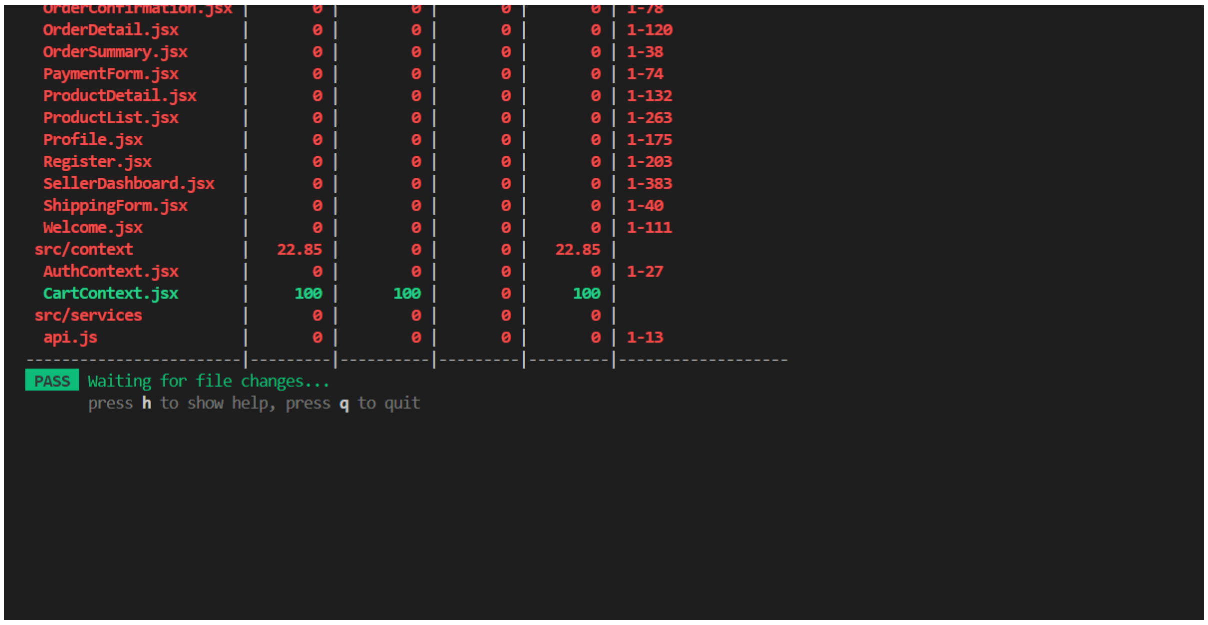
68.75% Statements 22/32 **80%** Branches 8/10 **50%** Functions 3/6 **68.75%** Lines 22/32

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

```
    import User from '../models/userModel.js';
    import Product from '../models/Product.js';
    import Order from '../models/Order.js';
    import asyncHandler from 'express-async-handler';

    // SELLERS

    // Get all sellers
1x  export const getAllSellers = async (req, res) => {
5x    const sellers = await User.find({ role: 'seller' });
4x    res.json(sellers);
    };

    // Approve seller
1x  export const approveSeller = async (req, res) => {
15x   const seller = await User.findById(req.params.id);

14x   if (!seller || seller.role !== 'seller') {
4x      res.status(404);
4x      throw new Error('Seller not found');
      }

10x   seller.isApproved = true;
10x   await seller.save();
9x    res.json({ message: 'Seller approved successfully' });
    };

    // Disapprove seller
1x  export const disapproveSeller = async (req, res) => {
13x   const seller = await User.findById(req.params.id);

```



```
/ src/components/__test__/Cart.test.jsx (3 tests) 523ms
  ✓ Cart Component - White Box Tests > renders cart items correctly 149ms
  ✓ Cart Component - White Box Tests > displays quantity inputs 224ms
  ✓ Cart Component - White Box Tests > calls remove handler when Remove button is clicked 145ms

est Files  1 passed (1)
     Tests  3 passed (3)
  Start at  23:11:04
  Duration  7.33s (transform 368ms, setup 569ms, collect 927ms, tests 523ms, environment 2.90s, prepare 501ms)

% Coverage report from v8
----------------------|---------|----------|---------|---------|-------------------
ile                   | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------------------|---------|----------|---------|---------|-------------------
ll files              |     3.4 |    13.79 |    6.25 |     3.4 |
src                   |       0 |        0 |       0 |       0 |
 App.jsx              |       0 |        0 |       0 |       0 | 1-86
 main.jsx             |       0 |        0 |       0 |       0 | 1-14
 theme.js             |       0 |        0 |       0 |       0 | 1-23
src/components        |    3.32 |    16.66 |    8.69 |    3.32 |
 AddressForm.jsx      |       0 |        0 |       0 |       0 | 1-129
 AdminDashboard.jsx   |       0 |        0 |       0 |       0 | 1-41
 AdminOrders.jsx      |       0 |        0 |       0 |       0 | 1-131
 Cart.jsx             |   89.89 |       80 |      50 |   89.89 | 20-30,65
 Checkout.jsx         |       0 |        0 |       0 |       0 | 1-248
 Login.jsx            |       0 |        0 |       0 |       0 | 1-105
 ManageProducts.jsx   |       0 |        0 |       0 |       0 | 1-72
 ManageSellers.jsx    |       0 |        0 |       0 |       0 | 1-101
 MyOrders.jsx         |       0 |        0 |       0 |       0 | 1-132
 OrderConfirmation.jsx |      0 |        0 |       0 |       0 | 1-78
 OrderDetail.jsx      |       0 |        0 |       0 |       0 | 1-120
 OrderSummary.jsx     |       0 |        0 |       0 |       0 | 1-38
 PaymentForm.jsx      |       0 |        0 |       0 |       0 | 1-74
 ProductDetail.jsx    |       0 |        0 |       0 |       0 | 1-132
 ProductList.jsx      |       0 |        0 |       0 |       0 | 1-263
 Profile.jsx          |       0 |        0 |       0 |       0 | 1-175
 Register.jsx         |       0 |        0 |       0 |       0 | 1-203
 SellerDashboard.jsx  |       0 |        0 |       0 |       0 | 1-383
 ShippingForm.jsx     |       0 |        0 |       0 |       0 | 1-40
 Welcome.jsx          |       0 |        0 |       0 |       0 | 1-111
src/context           |   22.85 |        0 |       0 |   22.85 |
```



```
  OrderConfirmation.jsx |       0 |        0 |       0 |       0 | 1-78
  OrderDetail.jsx       |       0 |        0 |       0 |       0 | 1-120
  OrderSummary.jsx      |       0 |        0 |       0 |       0 | 1-38
  PaymentForm.jsx       |       0 |        0 |       0 |       0 | 1-74
  ProductDetail.jsx     |       0 |        0 |       0 |       0 | 1-132
  ProductList.jsx       |       0 |        0 |       0 |       0 | 1-263
  Profile.jsx           |       0 |        0 |       0 |       0 | 1-175
  Register.jsx          |       0 |        0 |       0 |       0 | 1-203
  SellerDashboard.jsx   |       0 |        0 |       0 |       0 | 1-383
  ShippingForm.jsx      |       0 |        0 |       0 |       0 | 1-40
  Welcome.jsx           |       0 |        0 |       0 |       0 | 1-111
 src/context            |   22.85 |        0 |       0 |   22.85 |
  AuthContext.jsx       |       0 |        0 |       0 |       0 | 1-27
  CartContext.jsx       |     100 |      100 |       0 |     100 |
 src/services           |       0 |        0 |       0 |       0 |
  api.js                |       0 |        0 |       0 |       0 | 1-13
------------------------|---------|----------|---------|---------|-------------------
PASS  Waiting for file changes...
       press h to show help, press q to quit
```

13. WORK DIVISION BETWEEN GROUP MEMBERS

. **Farhan Ahmed:**

- Developed User Management module: Registration, Login, Profile Management, Password Reset
- Developed Authentication APIs (secure JWT, auth middleware)
- Implemented Admin Management of Users (approve/disapprove sellers)
- Performed all Blackbox and Whitebox Testing using Jest and Supertest
- Generated Test Coverage Report

. **Asim:**

- Developed Product Management Module: Add, Update, Delete Products
- Developed Cart Management Module: Add to cart, Remove from cart, Update cart quantities
- Developed Order Placement and Management: Create orders, Cancel orders
- Integrated Checkout and Payment Processing with multiple payment options

. **Ali:**

- Developed Seller Dashboard for managing products and viewing sales
- Developed Product Search and Filter functionalities
- Implemented Notifications for sellers (New order notifications)

. **Testing:**

- Farhan was responsible for all Testing (Black-box and White-box).
- Unit tests were written for all critical backend functions using Jest.

. **Documentation:**

- Documentation tasks (final report, user stories validation, burndown chart, Gantt chart, screenshots, notes) were divided equally among Farhan, Asim, and Ali.

14. LESSON LEARNT BY GROUP

- We learned the importance of **proper project planning and task distribution** to ensure timely completion of all modules.
- We understood how **modular backend architecture** (separating controllers, routes, models) improves scalability and maintainability.
- We gained practical experience in **writing unit tests** (white-box) and **API tests** (black-box) using **Jest** and **Supertest**.
- We realized that **test coverage reports** help in identifying which parts of code are well tested and which need more focus.
- We experienced real-world **bug fixing and optimization** cycles, especially during integration and testing phases.
- We learned how to use tools like **Gantt Charts** and **Burndown Charts** to track progress and manage deadlines effectively.
- We understood that **team communication** and **regular updates** are crucial when multiple members work on different modules.
- We became confident in generating **professional documentation** (user stories validation, coverage notes, testing reports).
- We realized that **boundary value analysis** and **equivalence class partitioning** make blackbox testing more systematic.

- Finally, we learned that **early and frequent testing** saves a lot of effort later in fixing bugs at the last minute.

GITHUB LINK:

https://github.com/Ali1Ahmad/Software_Engineering_Project