

Technical paper for full-stack E-commerce website development project using React, Node.js and MySQL

Aditya Ajay Gupta

6th Semester Bachelor of Computer Engineering Student

Pillai College of Engineering, Affiliated to Mumbai University

Internship at Fermion Infotech Private Limited

Navi Mumbai, Maharashtra, India.

Email address: adigup26@gmail.com

Mobile number 00 91 9619142085

Abstract. I am hereby presenting a technical paper on design and implementation of a full-stack e-commerce website development project using latest technologies like React [1] for frontend, supported by Node.js [2] & Express [3] for Backend and database from MySQL [4]. Website tested for various functions and found to be responsive and robust.

1 INTRODUCTION

The extensive use of E-commerce platforms in our day-to-day lives sparked my interest in studying and understanding the front-end, back-end, and database to develop this user-friendly full-stack e-commerce website which offers user registration, login, cart management, delivery & order instructions, session management and payment gateway. Key learnings and future enhancements elaborately explained in this paper.

2 RELATED WORKS

Efficiency of an E-Commerce Web Application with MERN Stack and Modern Tools (October 2022) [6]

C.M. K De Silva, A. S De Silva, K.A. I Maduwantha, D.A.I.U Dewpura, D.I. De Silva and R.R.P De Zoysa

<https://ijemr.vandanapublications.com/index.php/j/article/view/928/803>

We referred many articles and found article [6] as a Related work. Both, aforesaid article [6] and our project, has used React for Frontend and Node.js for Backend. User authentication, Cart management, Product display and Checkout are similar in Article [6] and our project.

Article [6] backend uses MongoDB whereas I have used MySQL which allowed me to use structured schema for users, sessions, orders and payment details. Further my project integrates Razorpay for real-time payment processing and verifies payments through cryptographic signature validation, whereas in article [6] this capability not addressed.

3 FRAMEWORKS AND PROGRAMMING LANGUAGES

- React [1] framework & CSS language for front-end
- Node.js [2] & Express [3] for Back end
- MySQL Workbench [4] for Database

4 SYSTEM ARCHITECTURE

Three key components namely the Front end, Back end and Database of this web application system are illustrated in Figure 1 having System architecture which is enclosed herewith.

System components described as follows:

4.1 Front End

React framework [1] frontend layer provides a user-friendly, dynamic & responsive interface. Product data are fetched from API.

Frontend communicates seamlessly with the Backend through HTTP requests to API endpoints.

Front end includes User registration, Login & Log-out process, Form validation & Error handling, Visual representation & other details for products, Cart operations, Ordering & Delivery instructions, and Payment Gateway.

Figure 2 to 11 having Snapshots of Frontend pages are enclosed herewith for ready reference.

4.2 Back End

Node.js [2] and Express [3] frameworks handle HTTP requests and facilitate communication between the front end, server, and database.

Back-end frameworks provide User authentication & authorization, Session management, Database operation, and Password encryption.

Figure 12 having Schema design of backend enclosed herewith for ready reference.

4.3 Database

User information, Session data, and Shopping cart data are stored using the My SQL Workbench [4] database system. The database includes the following tables:

- User table (User ID, User Name, Password, Full Name, Email, Phone)
- Session table (Session ID, Expires, Cookie data)
- Cart table (Cart ID, User ID, Product ID, Quantity)
- Order table (User ID, Order ID, Cart data, Total amount, Order instructions, Delivery instructions, Payment ID, Payment status, Delivery address)

5 IMPLEMENTATION & USAGE

5.1 User Registration

Easy to use Front end form captures details fed by Users like Full Name, Password, Email address, and Phone number.

Refer enclosed Fig. 3 having snapshot of this frontend page.

5.2 User Login Process

- User fills User name & Password in front end Login form
- The back end crosschecks credentials fed by the user from the database.
- Password verification is done by bcrypt [5] comparison.

Refer enclosed Fig. 4 having snapshot of this frontend page.

5.3 Session management

- Upon successful authentication session is created and a Session ID is assigned to each session using an HTTP cookie.
- Session with its ID stored in the Session table.
- User name with the Hello prefix is displayed upon session storage.
- When a user gives the Logout command session is destroyed.

Fig. 12 having Session management code snippets attached for ready reference.

5.4 Shopping cart

Enables continuous storage of product information selected by the user across various sessions.

- Add to cart
- Check user authentication, Facilitate the addition and updation of products in the cart.
- Assign order Quantity to items in the cart
- Allows to modify quantify which is 1 by default.
- Remove the item from the cart.
- Updation of Prices as per the quantity of items in the cart.

- No access to the Cart after the session is destroyed.

Refer enclosed Fig. 5 & 6 having snapshots of these frontend pages.

5.5 Order & Delivery instructions

Frontend form enables the user to Type in customized user order instructions like Gift wrap, Fragile, Include bill with consignment, and the User fills in the Delivery address along with customized delivery instructions like Leave at the security desk, Call upon arrival, etc.

- Upon clicking the Proceed to pay button, filled-in user inputs are captured in the frontend and sent to the backend, for generating Order ID and saving Order & Delivery instructions.

Refer enclosed Fig. 7 & 8 having snapshots of these frontend pages.

5.6 Payment Gateway

Enables secured and real-time Payment using Razorpay payment gateway.

- Upon clicking the Checkout Now button, a post request is sent to the backend to refer Order ID and create a Razorpay order for the total cart amount.
- Subsequently, Frontend displays the Razorpay checkout widget using the Order ID.
- Upon receipt of payment instructions from the user, Razorpay sends the Payment ID and Transaction signature to the frontend and for backend verification.
- Upon successful payment Razorpay via post requests informs the backend to save full order data including Payment ID.
- Simultaneously Razorpay post requests inform frontend to display the Payment successful message.

Refer enclosed Fig. 9 & 10 having snapshots of these frontend pages.

Fig. 13 having Razor Payment Gateway Integration code snippets attached for ready reference.

6 LEARNINGS

6.1 Frontend learnings

- Use of Function based components in React [1]
- Use of Hooks functions (Custom Hook & Inbuilt hook)
- Create a context that acts like a Cloud where all pages will share the same Cart data
- Incorporated Filter function which creates a new array containing only those items that did not match the condition.
- Implemented Webkit CSS function for Media queries.

- Created Media responsiveness (1553 pixels, 1386 pixels, 1112 pixels, 600 pixels).
- Added responsive navigation.
- Created Dynamic slider for Landing page
- Added Out of stock / Added to cart Pop-ups

6.2 Backend learnings

- Used Express session for website to store data across multiple HTTP requests
- Used CORs (Cross-origin requests)
- Established secure Session management using express-session with MySQL
- Implemented password hashing with bcrypt for secure user authentication
- Designed APIs for login, registration, cart operations, and order management
- Connected MySQL database with Express for dynamic CRUD operations
- Cross-Origin Resource Sharing (CORS) for communication between frontend and backend
- Built a Razorpay integration for secure and dynamic payment processing
- Verified payments using cryptographic signature validation on the backend
- Stored complete order metadata (cart items, payment info, instructions) upon successful payment
- Added robust error handling and validation for user sessions and database interactions

7 TESTING

- Manually tested user authentication flows (login, logout, registration) across different user roles.
- Stimulated various cart operations (add, remove, update quantity) to check state consistency
- Checked Session persistence using browser dev tools and cookie inspection
- Validated Razorpay payment flow using test keys and monitored success/failure redirections
- Ensured proper form validation.
- Browser resize and mobile emulation tools used for testing responsiveness and UI behaviors.

8 RESULTS & CONCLUSION

Satisfactorily achieved the development of full stack eCommerce website using the latest technologies like React [1], Node.js [2], Express [3], and MySQL workbench [4].

9 FUTURE ENHANCEMENTS

- Customer feedback
- Goods return & refund policy
- Complete Backend panel

10 ACKNOWLEDGEMENTS

The author thankfully acknowledges guidance and contribution of his mentor Ms. Monali Tayade at M/s. Fermion Infotech Pvt. Ltd., Vashi, Navi Mumbai, India. Email address: monalit@fermion.in.

11 REFERENCES

- [1] React: JavaScript library by Meta (Facebook) <https://reactjs.org>.
- [2] Node.js, Node.js Foundation: Cross-platform JavaScript runtime environment. <https://nodejs.org>.
- [3] Express: Minimal & flexible Node.js web application framework. <https://expressjs.com>.
- [4] MySQL: Open-source database by Oracle Corporation. <https://dev.mysql.com>.
- [5] bcrypt: Node package manager (npm) used in Node.js for securely hashing passwords. <https://www.npmjs.com/package/bcrypt>.
- [6] Efficiency of an E-Commerce Web Application with MERN Stack and Modern Tools (October 2022). C.M. K De Silva, A. S De Silva, K.A. I Maduwantha, D.A.I.U Dewapura, D.I. De Silva and R.R.P De Zoysa.

12 ANNEXURE

Following System architecture, Snapshots of the Frontend pages, Backend design schema and Code snippets enclosed:

Fig. 1 System architecture

Fig. 2 Website Landing page.

Fig. 3 User registration page

Fig. 4 User Login page

Fig. 5 Product display page

Fig. 6 Cart page

Fig. 7 Checkout / Ordering page

Fig. 8 Delivery Instruction page

Fig. 9 Payment gateway page

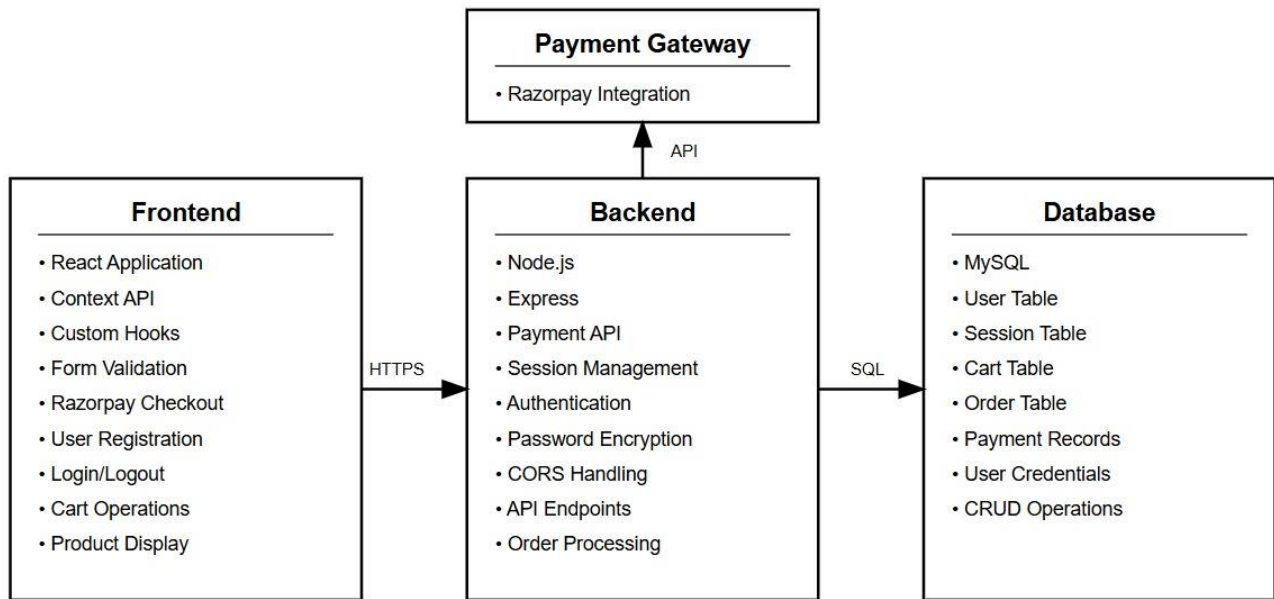
Fig. 10 Payment success page

Fig. 11 Schema design of backend

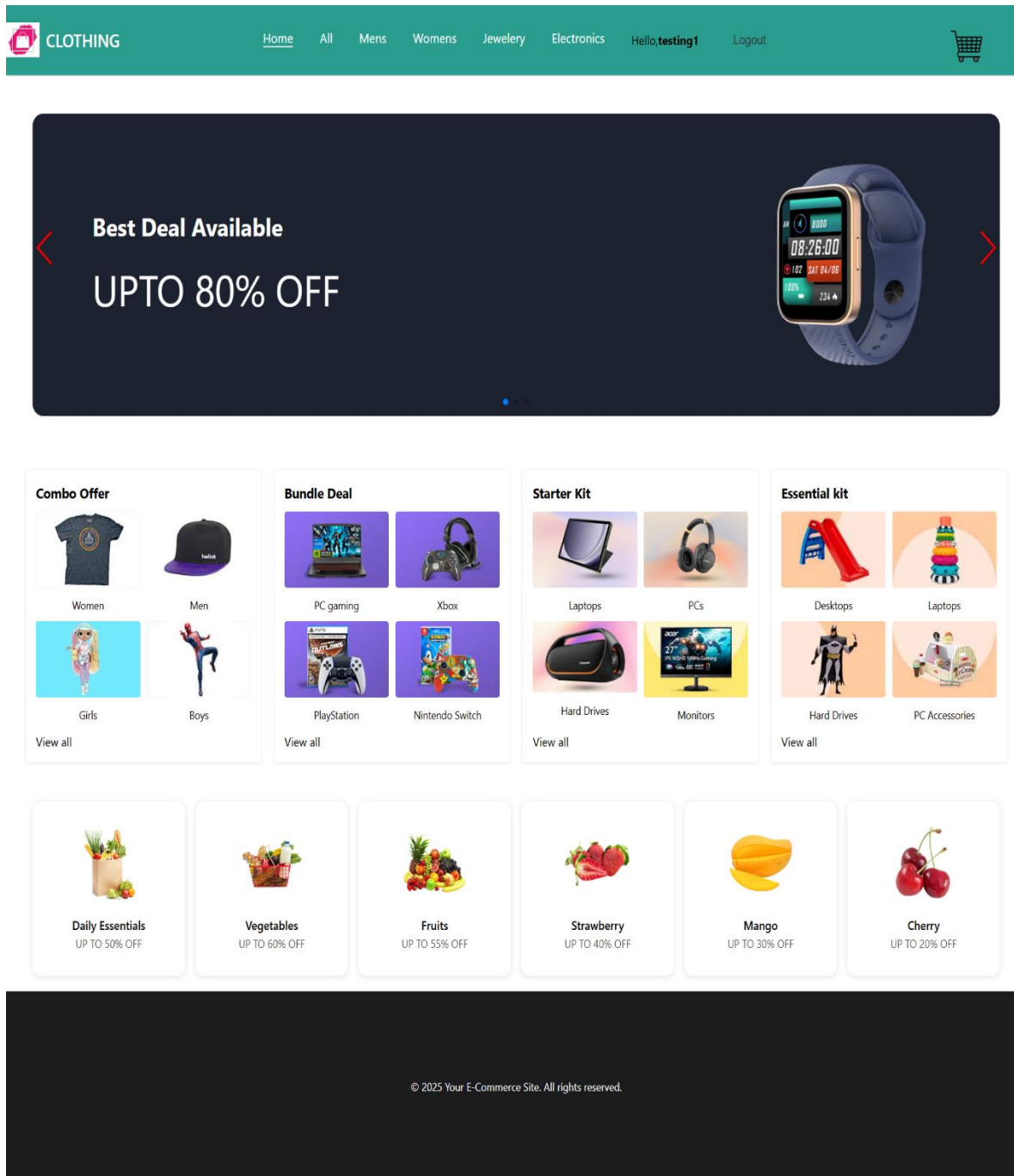
Fig. 12 Session management code snippet

Fig. 13 Razorpay Payment integration code snippet


Annexure : System architecture, Frontend pages, Backend design schema and Code snippets




"Fig. 1" System architecture




“Fig. 2” Website landing page

CLOTHING

[Home](#)[All](#)[Mens](#)[Womens](#)[Jewelry](#)[Electronics](#)[Profile](#)[Logout](#)







Create an Account

Join us and start shopping today


Full Name

 Your full name


Email

 Email address


Phone

 Phone number

Username

 Choose a username

Password

 Create a password

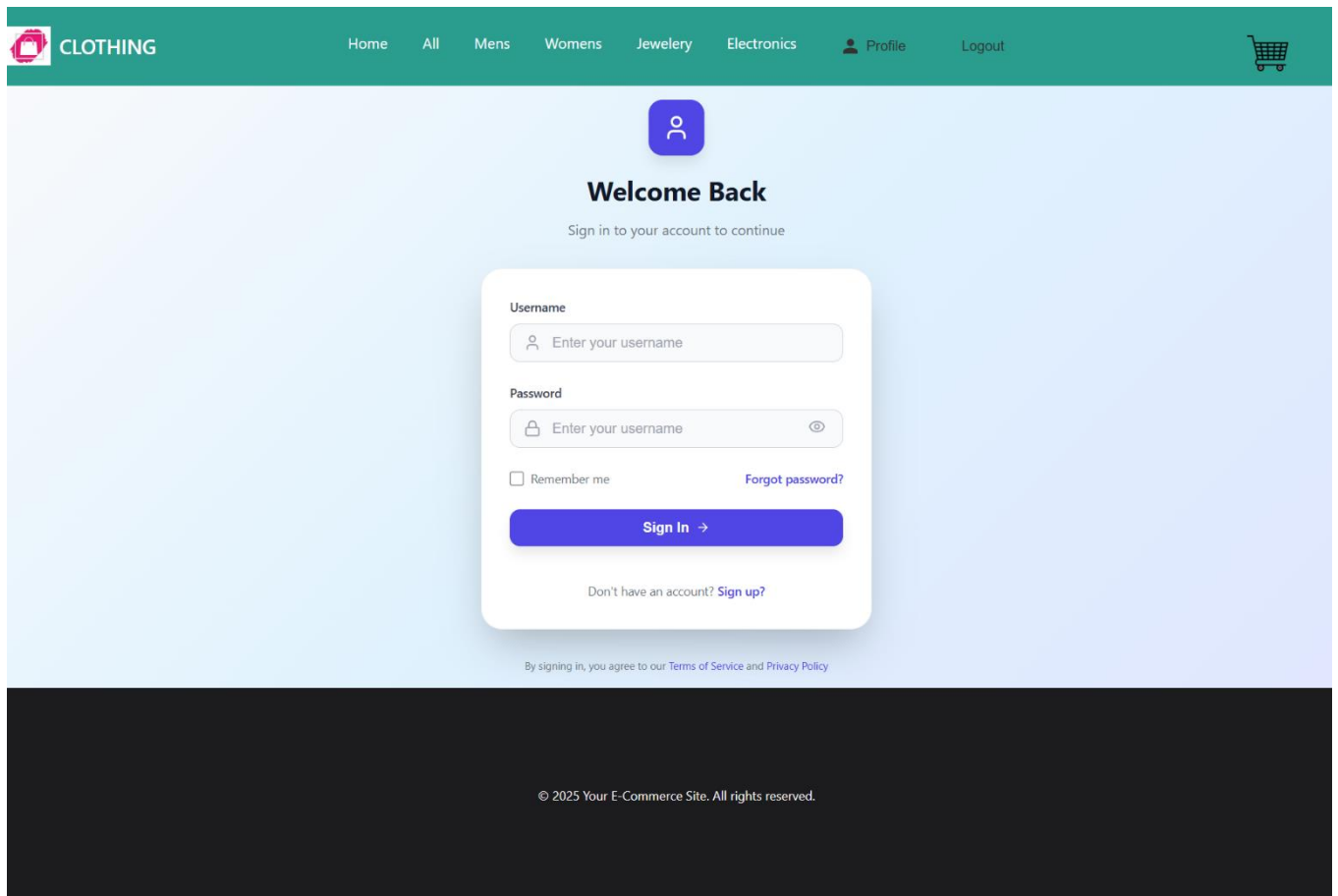
Sign Up →

Already have an account? [Login](#)

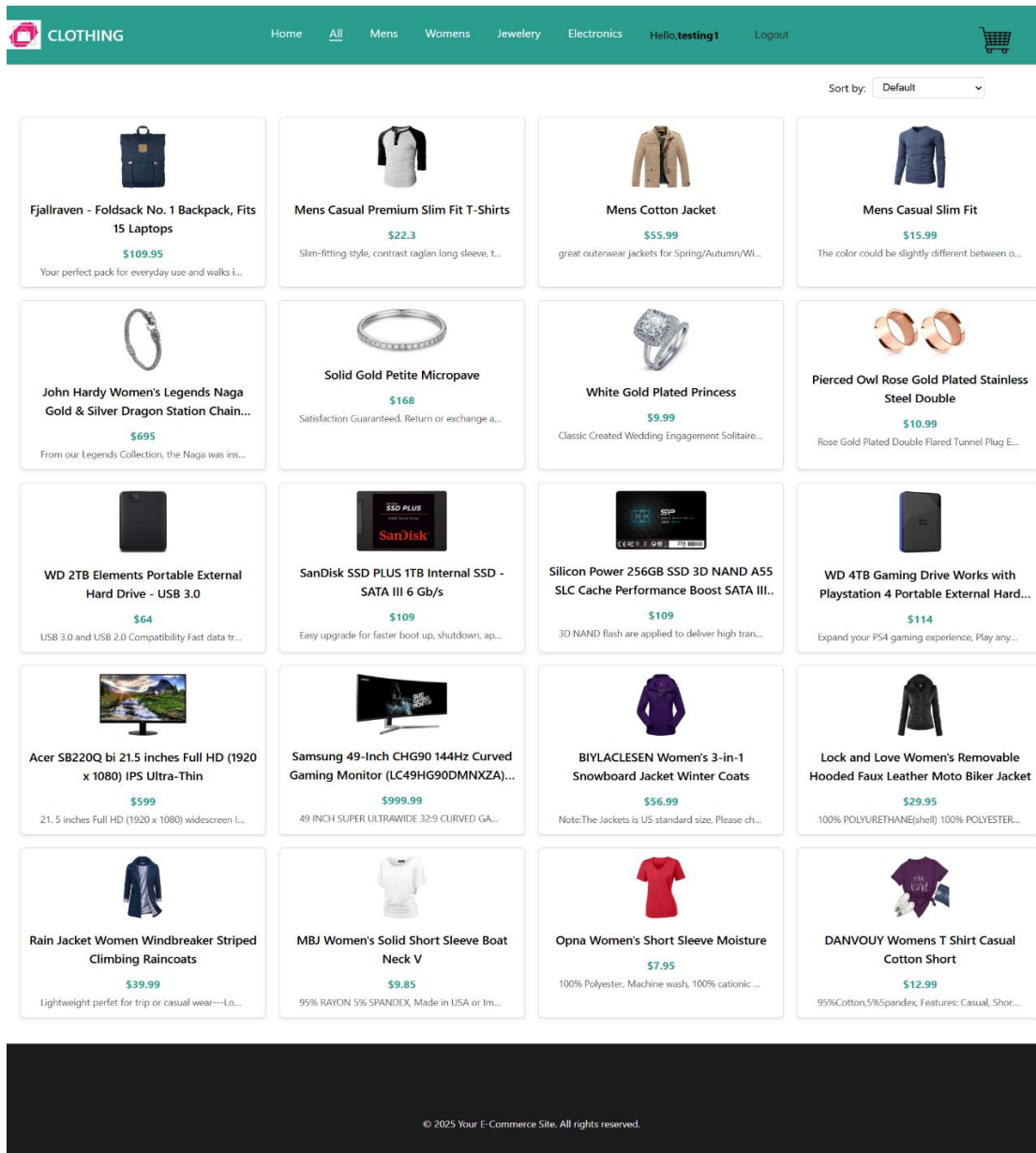
By signing up, you agree to our [Terms of Service](#) and [Privacy Policy](#).

© 2025 Your E-Commerce Site. All rights reserved.

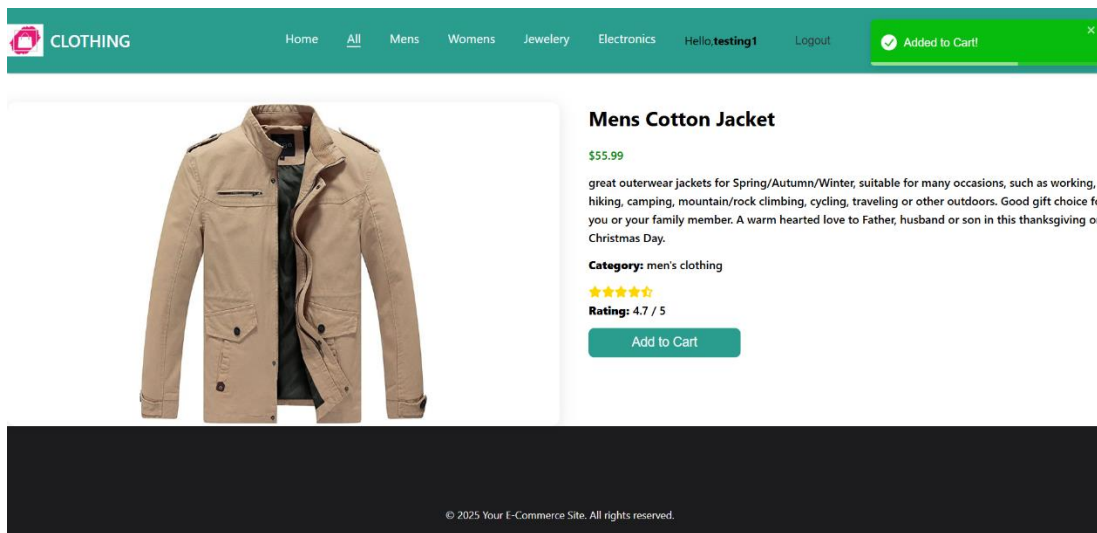
"Fig. 3" User registration page



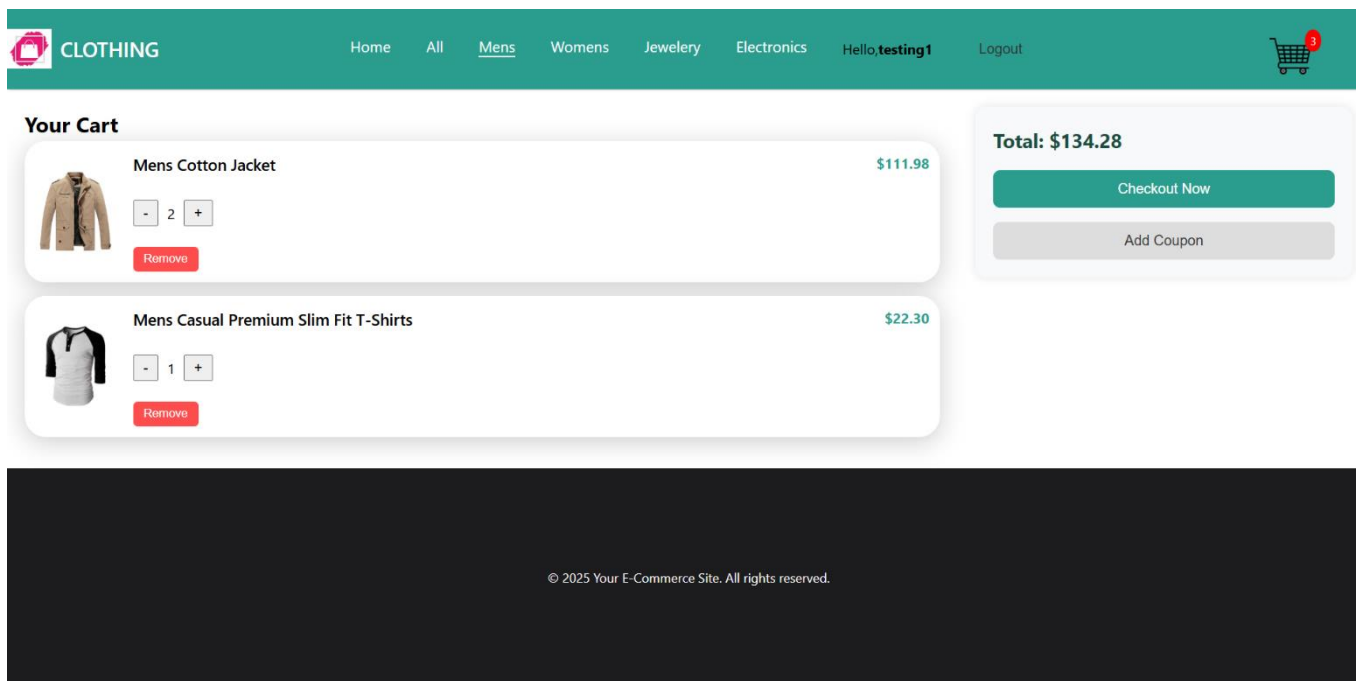
“Fig. 4” User login page




“Fig. 5” Product display page




“Fig. 6” Cart page



“Fig.7” Checkout Ordering page

CLOTHING

[Home](#) [All](#) [Mens](#) [Womens](#) [Jewelry](#) [Electronics](#) [Hello,testing1](#) [Logout](#)



Checkout

Delivery Address

Address Line 1 *

1501, B block , SEA towers

Address Line 2

City *

Navi Mumbai

State *

Maharashtra

Zip Code *

400706

Country *

India

Order Instructions

Pack as a gift

Delivery Instructions

Leave at Doorstep


Order Summary

Total: ₹134.28


Proceed to Pay ₹134.28

© 2025 Your E-Commerce Site. All rights reserved.

“Fig. 8” Delivery instructions page

CLOTHING

[Home](#) [All](#) [Mens](#) [Womens](#) [Jewelry](#) [Electronics](#) [Hello,testing1](#) [Logout](#)



Checkout

Delivery Address

Address Line 1 *

1501, B block , SEA towers

Address Line 2

City *

Navi Mumbai

State *

Maharashtra

Zip Code *

400706

Country *

India

Order Instructions

Pack as a gift

Delivery Instructions

Leave at Doorstep

Order Summary

Total: ₹134.28


Proceed to Pay ₹134.28

Y Your Store

Price Summary


₹134.28


Using as +91 99999 99999 >


Secured by 


Payment Options


Recommended

UPI 

Cards 


EMI 

Netbanking 

Wallet 

More options

UPI QR



Scan the QR using any UPI App

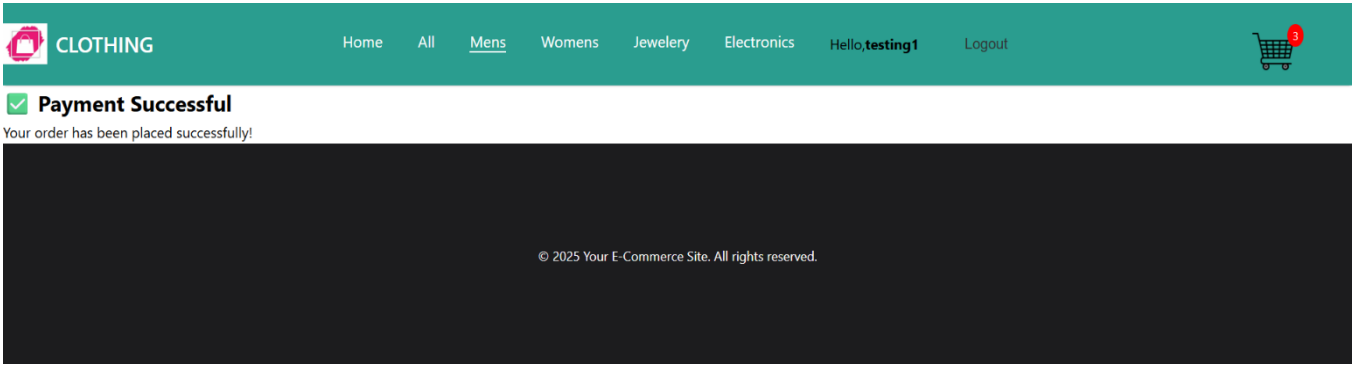
Pay with UPI ID / Number

testing@ybl

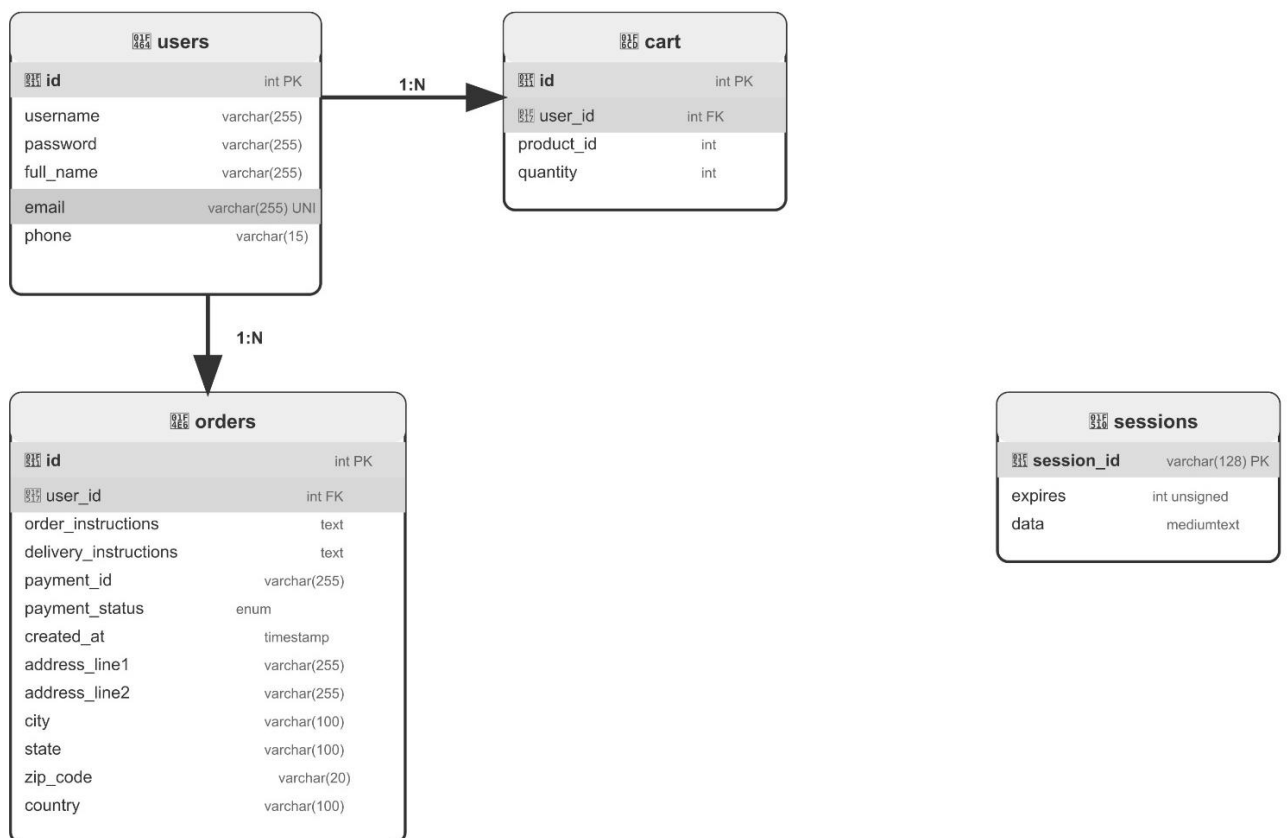
Verify and Pay

Test Mode

“Fig. 9” Payment Gateway page



“Fig. 10” Payment success page



“Fig. 11” Schema design of backend

```
338 });
339
340 // LOGOUT
341 app.post('/logout', (req, res) => {
342   req.session.destroy(err => {
343     if (err) return res.status(500).send('Logout error');
344     res.clearCookie('connect.sid');
345     res.json({ success: true });
346   });
347 });
348
```

```
321 //session created upon login
322 req.session.user = { id: user.id, username: user.username };
323 console.log(" Login successful, session created");
324 return res.json({ success: true, user: req.session.user });
325 });
326
```

```
ecom > backend > JS index.js > ...
294 app.post('/login', (req, res) => {
295   db.query('SELECT * FROM users WHERE username = ?', [username], async (err, results) => {
322     return res.status(401).json({ success: false, message: 'Invalid credentials (wrong password)' });
323   });
324
325   req.session.user = { id: user.id, username: user.username };
326   console.log(" Login successful, session created");
327   return res.json({ success: true, user: req.session.user });
328 });
329
330 // SESSION Check
331 app.get('/session', (req, res) => {
332   if (req.session.user) {
333     res.json({ loggedIn: true, user: req.session.user });
334   } else {
335     res.json({ loggedIn: false });
336   }
337 });
338
```

```
File Edit Selection View Go Run Terminal Help
ecommerce
EXPLORER
  ECOMMERCE
    ecom
      backend
        node_modules
        JS index.js
        package-lock.json
        package.json
        server.js
        node_modules
        public
          favicon.ico
          index.html
          logo192.png
          logo512.png
          manifest.json
          robots.txt
        src
          Components
          Context
          Pages
            Cart.css
            Cart.jsx
            Checkout.jsx
            Details.css
            Details.jsx
            Footer.css
            Footer.jsx
            home.css
            Home.jsx
            Login.css
            Login.jsx
          OUTLINE
  App.css
  ShopCategory.css
  Cart.css
  JS index.js ...backend
  server.js
  Checkout.jsx
  OrderSuccess.jsx
  OrderFailed.jsx

ecom > backend > JS index.js > ...
1  const express = require('express');
2  const session = require('express-session');
3  const mysql = require('mysql2');
4  const MySQLStore = require('express-mysql-session')(session);
5
6  const cors = require('cors');
7  const bcrypt = require('bcrypt');
8  const nodemailer = require('nodemailer');
9  const Razorpay = require('razorpay');
10 const app = express();
11 const PORT = 5000;
12 const crypto = require('crypto');
13
14 const db = mysql.createConnection({
15   host: 'localhost',
16   user: 'root',
17   password: 'Aditya',
18   database: 'ecommerce'
19 });
20
21 const sessionStore = new MySQLStore({}, db.promise());
22 app.use(express.json()); // built-in JSON parser
23
24 //cors
25 app.use(cors({
26   origin: 'http://localhost:3000',
27   credentials: true
28 }));
29
30 //session setup
31 app.use(session({
32   secret: 'secretKey123',
33   resave: false,
34   saveUninitialized: false,
35   store: sessionStore,
36   cookie: { secure: false }
37 }));
```

“Fig. 12” Session management Code snippets

```

app.post('/create-order', async (req, res) => {
  try {
    console.log("📄 Received /create-order body:", req.body);
    console.log("👤 Session user:", req.session.user);

    // Check if user is logged in
    if (!req.session.user) {
      console.log("❌ User not logged in");
      return res.status(403).json({ error: "User not logged in" });
    }

    if (!req.body || typeof req.body.amount === 'undefined') {
      console.log("❌ Bad Request: Missing amount in body");
      return res.status(400).json({ error: "Amount is required" });
    }

    const { amount } = req.body;

    if (amount <= 0) {
      console.log("❌ Invalid amount:", amount);
      return res.status(400).json({ error: "Amount must be greater than 0" });
    }

    const options = {
      amount: Math.round(amount * 100), // Convert to paise
      currency: 'INR',
      receipt: `rcpt_${Date.now()}_${req.session.user.id}`
    };

    console.log("🔗 Creating Razorpay order with options:", options);
    const order = await razorpay.orders.create(options);
    console.log("✅ Razorpay Order Created:", order);

    res.json(order);
  } catch (err) {
    console.error("❌ Razorpay order creation failed:", err);
    res.status(500).json({
      error: "Failed to create order",
      details: err.message
    });
  }
});

```

<ul style="list-style-type: none"> 📄 Details.jsx # Footer.css 📄 Footer.jsx # home.css 📄 Home.jsx # Login.css 📄 Login.jsx <p>> OUTLINE</p>	<pre> 79 }); 80 81 //razorpay 82 // Razorpay instance 83 const razorpay = new Razorpay({ 84 key_id: 'rzp_test_o08YVZwKccbv9M', 85 key_secret: 'Kk7Sp1T1mDRiFU4Px2hBNWsR' 86 }); 87 </pre>
---	---

"Fig. 13" Razorpay Payment Gateway Integration code snippets