

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

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 for frontend, supported by Node.js & Express for Backend and database from MySQL. Website tested for various functions and found to be responsive and robust.

I 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.

II 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.

III 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

IV 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:

A. 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.

B. 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.

C. 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)

V IMPLEMENTATION & USAGE

A. 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.

B. 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.

C. 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.

D. 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.

E. 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.

F. 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.

VI LEARNINGS

A. 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

B. 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

VII 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.

VIII 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].

IX FUTURE ENHANCEMENTS

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

X 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.

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 Dewpura, D.I. De Silva and R.R.P De Zoysa.

XII 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

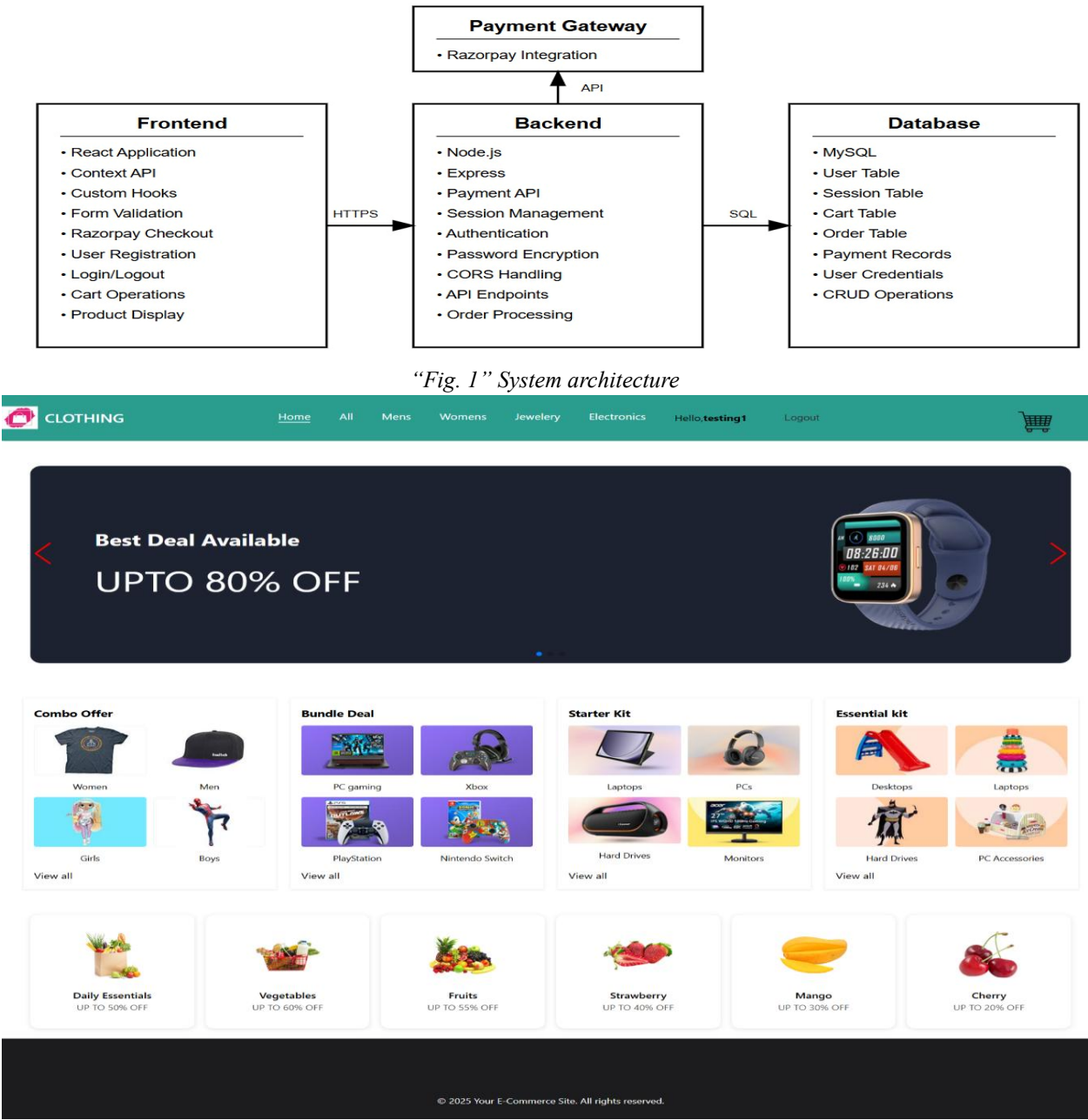
Fig. 10 Payment success page

Fig. 11 Schema design of backend

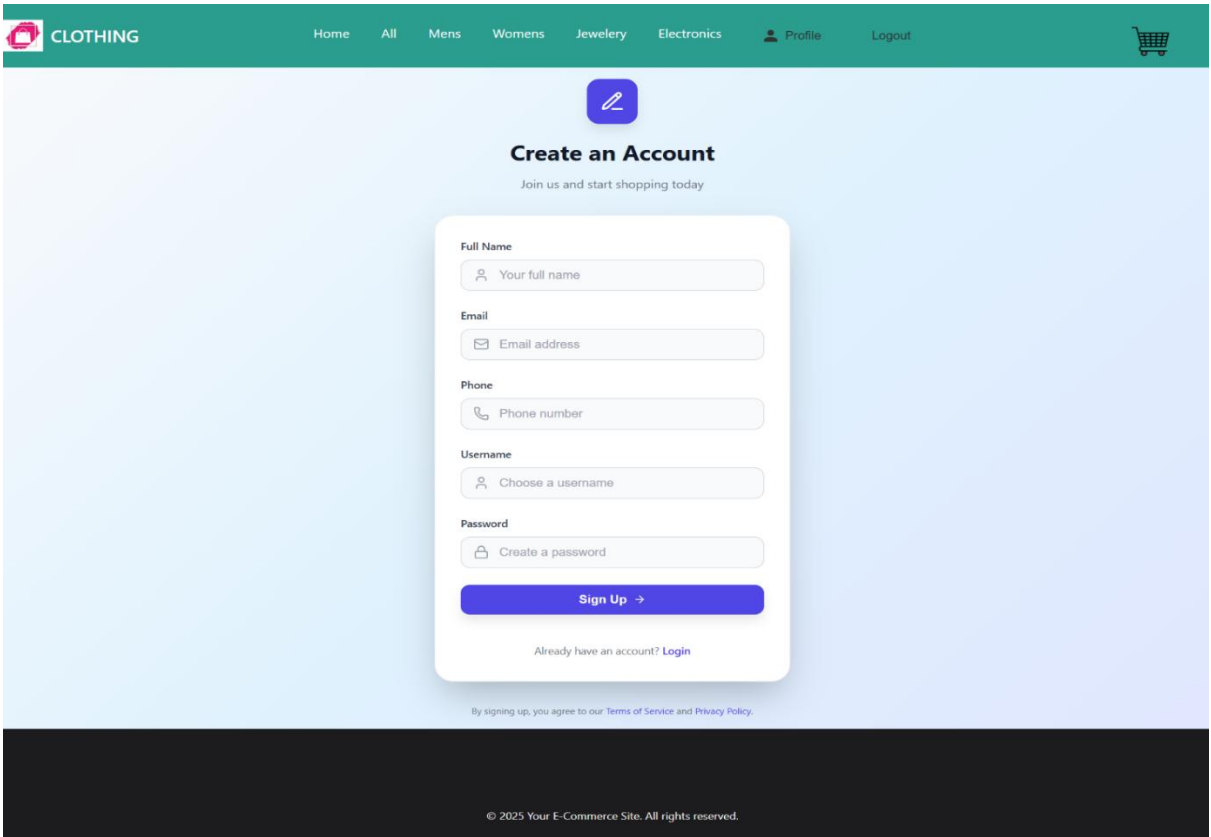
Fig. 12 Session management code snippet

Fig. 13 Razorpay Payment integration code snippet

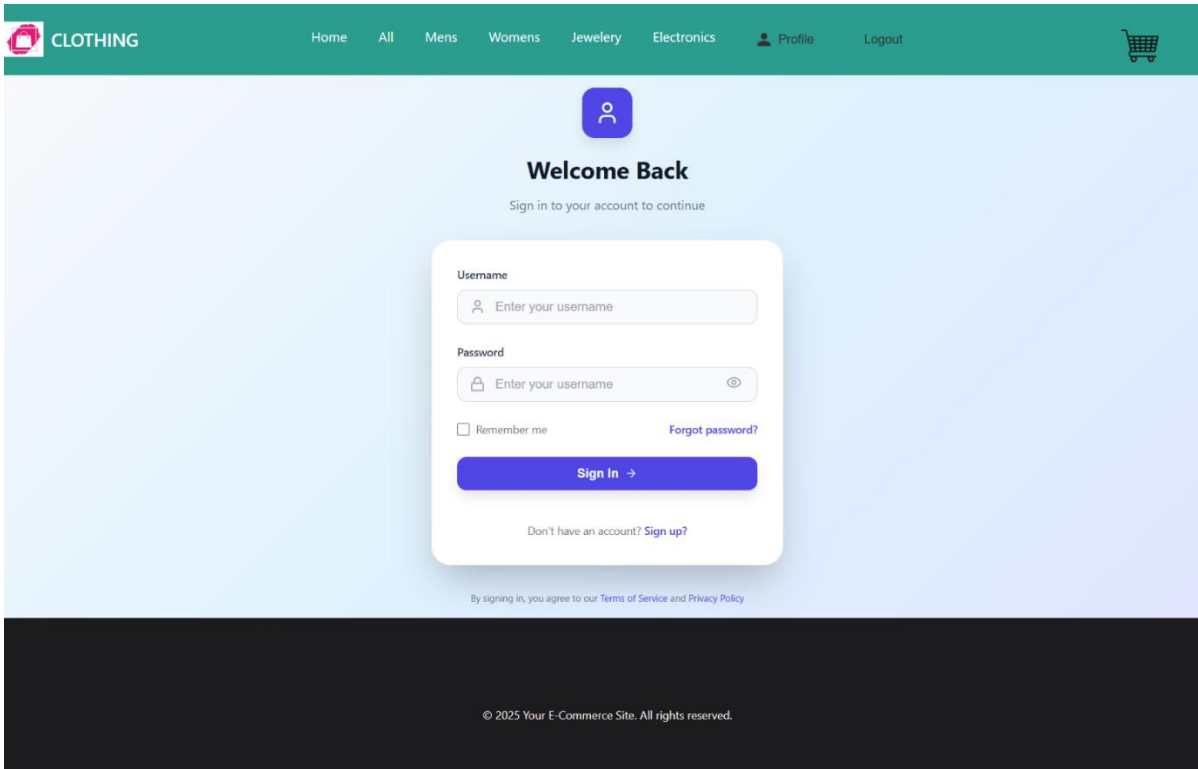
Annexure: System architecture, Frontend pages, Backend design schema and Code snippets



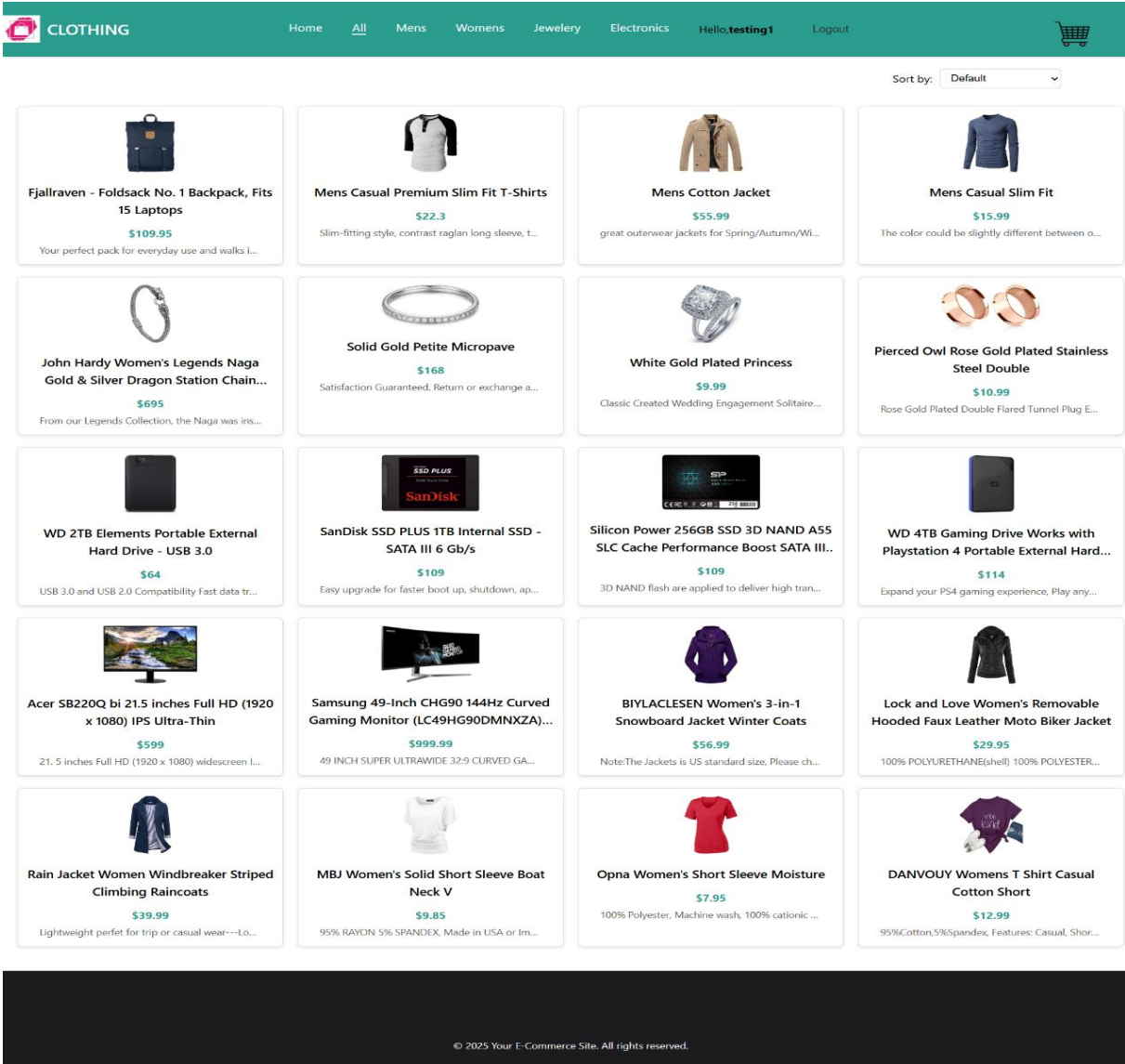
“Fig. 2” Website landing page



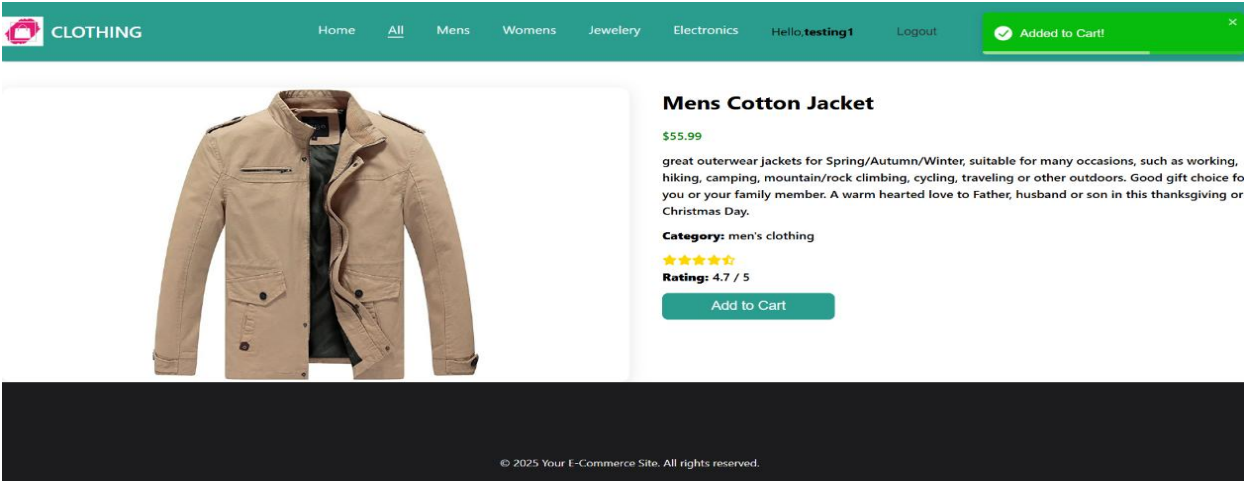
“Fig. 3” User registration page



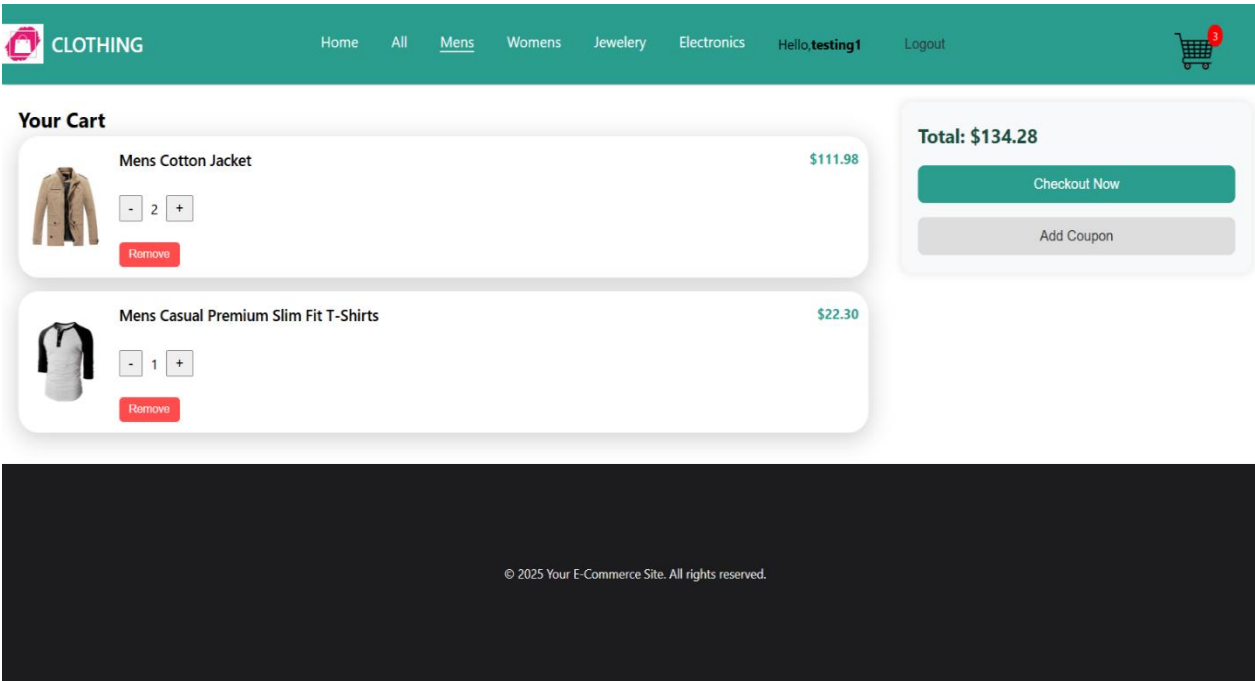
“Fig. 4” User login page



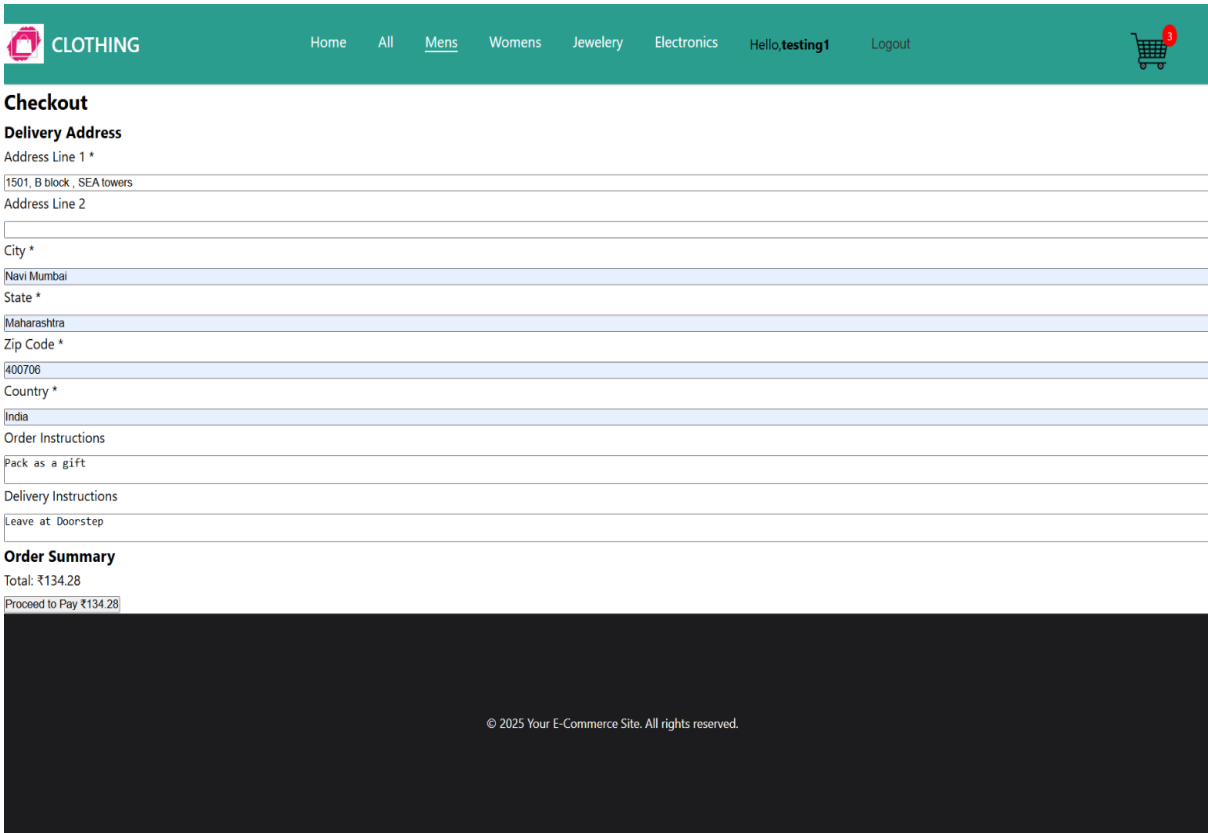
“Fig. 5” Product display page



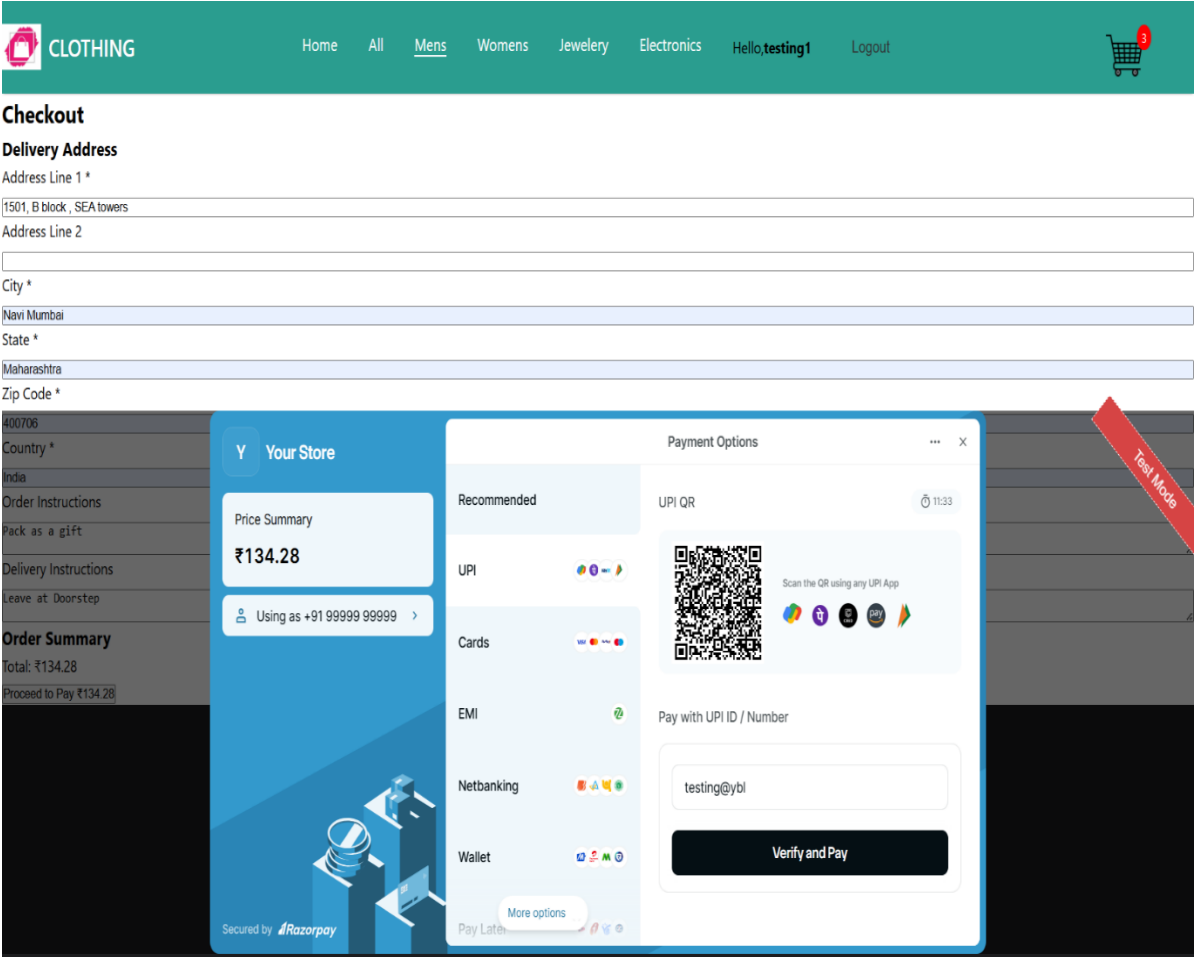
“Fig. 6” Cart page



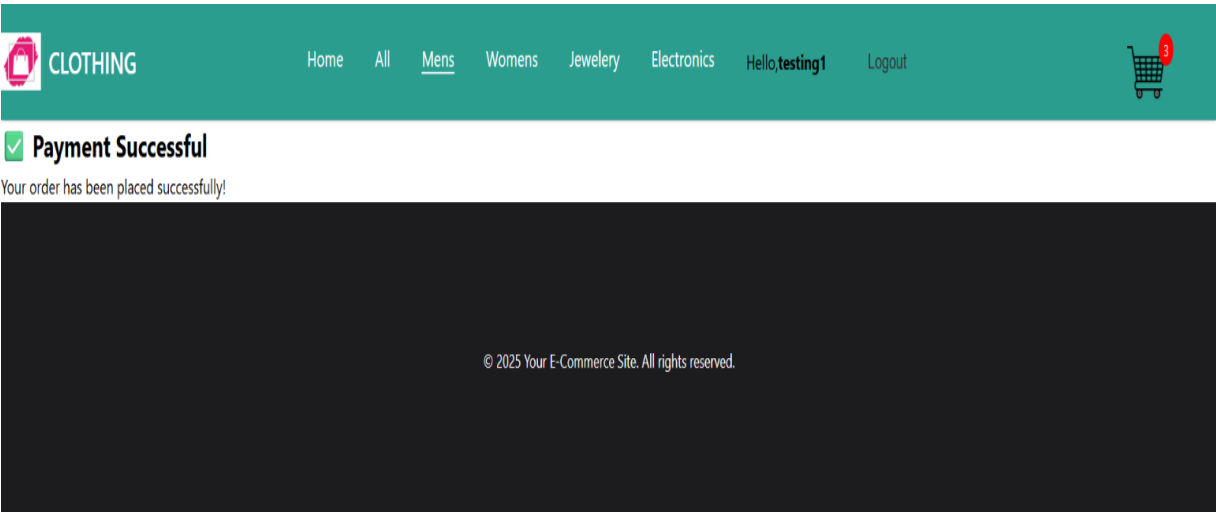
“Fig.7” Checkout Ordering page



“Fig. 8” Delivery instructions page



“Fig. 9” Payment Gateway page



“Fig. 10” Payment success page



“Fig. 11” Schema design of backend

```

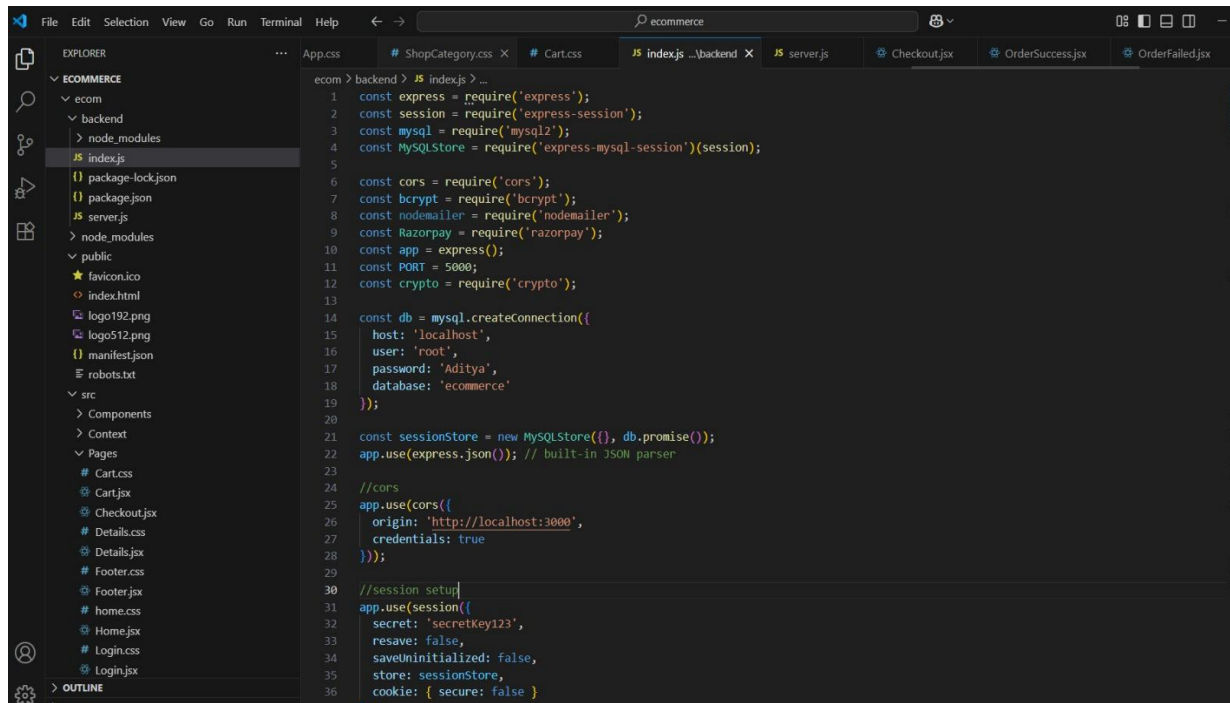
    > Components
    > Context
    > Pages
    # Cart.css
    # Cart.jsx
    # Checkout.jsx
    # Details.css
    # Details.jsx
    # Footer.css
    # Footer.jsx
    # Home.css
    # Home.jsx
    # Login.css
    # Login.jsx

    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

    322 //session created upon login
    323
    324 req.session.user = { id: user.id, username: user.username };
    325 console.log("Login successful, session created");
    326 return res.json({ success: true, user: req.session.user });
    327 });
    328
    329 });
    330

    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
    331 // SESSION Check
    332 app.get('/session', (req, res) => {
    333     if (req.session.user) {
    334         res.json({ loggedIn: true, user: req.session.user });
    335     } else {
    336         res.json({ loggedIn: false });
    337     }
    338 });
    339
    340
    341
    342
    343
    344
    345
    346
    347
    348
    349
    350
    351
    352
    353
    354
    355
    356
    357
    358
    359
    360
    361
    362
    363
    364
    365
    366
    367
    368
    369
    370
    371
    372
    373
    374
    375
    376
    377
    378
    379
    380
    381
    382
    383
    384
    385
    386
    387
    388
    389
    390
    391
    392
    393
    394
    395
    396
    397
    398
    399
    400
    401
    402
    403
    404
    405
    406
    407
    408
    409
    410
    411
    412
    413
    414
    415
    416
    417
    418
    419
    420
    421
    422
    423
    424
    425
    426
    427
    428
    429
    430
    431
    432
    433
    434
    435
    436
    437
    438
    439
    440
    441
    442
    443
    444
    445
    446
    447
    448
    449
    450
    451
    452
    453
    454
    455
    456
    457
    458
    459
    460
    461
    462
    463
    464
    465
    466
    467
    468
    469
    470
    471
    472
    473
    474
    475
    476
    477
    478
    479
    480
    481
    482
    483
    484
    485
    486
    487
    488
    489
    490
    491
    492
    493
    494
    495
    496
    497
    498
    499
    500
    501
    502
    503
    504
    505
    506
    507
    508
    509
    510
    511
    512
    513
    514
    515
    516
    517
    518
    519
    520
    521
    522
    523
    524
    525
    526
    527
    528
    529
    530
    531
    532
    533
    534
    535
    536
    537
    538
    539
    540
    541
    542
    543
    544
    545
    546
    547
    548
    549
    550
    551
    552
    553
    554
    555
    556
    557
    558
    559
    560
    561
    562
    563
    564
    565
    566
    567
    568
    569
    570
    571
    572
    573
    574
    575
    576
    577
    578
    579
    580
    581
    582
    583
    584
    585
    586
    587
    588
    589
    590
    591
    592
    593
    594
    595
    596
    597
    598
    599
    600
    601
    602
    603
    604
    605
    606
    607
    608
    609
    610
    611
    612
    613
    614
    615
    616
    617
    618
    619
    620
    621
    622
    623
    624
    625
    626
    627
    628
    629
    630
    631
    632
    633
    634
    635
    636
    637
    638
    639
    640
    641
    642
    643
    644
    645
    646
    647
    648
    649
    650
    651
    652
    653
    654
    655
    656
    657
    658
    659
    660
    661
    662
    663
    664
    665
    666
    667
    668
    669
    670
    671
    672
    673
    674
    675
    676
    677
    678
    679
    680
    681
    682
    683
    684
    685
    686
    687
    688
    689
    690
    691
    692
    693
    694
    695
    696
    697
    698
    699
    700
    701
    702
    703
    704
    705
    706
    707
    708
    709
    710
    711
    712
    713
    714
    715
    716
    717
    718
    719
    720
    721
    722
    723
    724
    725
    726
    727
    728
    729
    730
    731
    732
    733
    734
    735
    736
    737
    738
    739
    740
    741
    742
    743
    744
    745
    746
    747
    748
    749
    750
    751
    752
    753
    754
    755
    756
    757
    758
    759
    760
    761
    762
    763
    764
    765
    766
    767
    768
    769
    770
    771
    772
    773
    774
    775
    776
    777
    778
    779
    780
    781
    782
    783
    784
    785
    786
    787
    788
    789
    790
    791
    792
    793
    794
    795
    796
    797
    798
    799
    800
    801
    802
    803
    804
    805
    806
    807
    808
    809
    810
    811
    812
    813
    814
    815
    816
    817
    818
    819
    820
    821
    822
    823
    824
    825
    826
    827
    828
    829
    830
    831
    832
    833
    834
    835
    836
    837
    838
    839
    840
    841
    842
    843
    844
    845
    846
    847
    848
    849
    850
    851
    852
    853
    854
    855
    856
    857
    858
    859
    860
    861
    862
    863
    864
    865
    866
    867
    868
    869
    870
    871
    872
    873
    874
    875
    876
    877
    878
    879
    880
    881
    882
    883
    884
    885
    886
    887
    888
    889
    890
    891
    892
    893
    894
    895
    896
    897
    898
    899
    900
    901
    902
    903
    904
    905
    906
    907
    908
    909
    910
    911
    912
    913
    914
    915
    916
    917
    918
    919
    920
    921
    922
    923
    924
    925
    926
    927
    928
    929
    930
    931
    932
    933
    934
    935
    936
    937
    938
    939
    940
    941
    942
    943
    944
    945
    946
    947
    948
    949
    950
    951
    952
    953
    954
    955
    956
    957
    958
    959
    960
    961
    962
    963
    964
    965
    966
    967
    968
    969
    970
    971
    972
    973
    974
    975
    976
    977
    978
    979
    980
    981
    982
    983
    984
    985
    986
    987
    988
    989
    990
    991
    992
    993
    994
    995
    996
    997
    998
    999
    1000

```



```

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 }

```

“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;

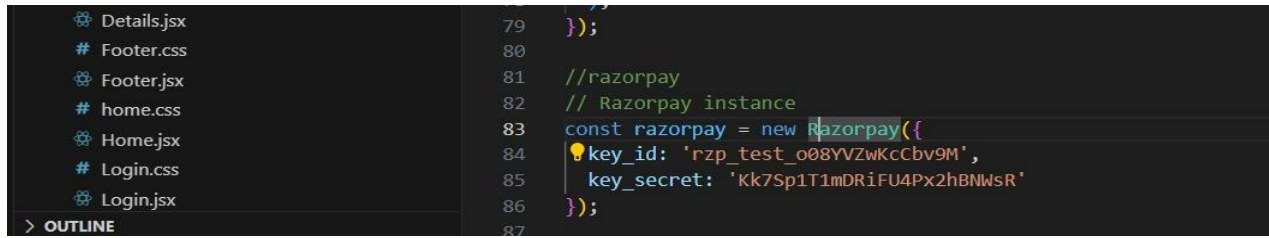
    if (amount <= 0) {
      console.log("❌ Invalid amount:", amount);
      return res.status(400).json({ error: "Amount must be greater than 0" });
    }

    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
    });
  }
});

```



The image shows a code editor with a dark theme. On the left, there is a file explorer sidebar with a tree view containing the following files: Details.jsx, Footer.css, Footer.jsx, home.css, Home.jsx, Login.css, and Login.jsx. Below the file list is a button labeled '> OUTLINE'. The main editor area displays JavaScript code for Razorpay integration. The code is as follows:

```
79 });  
80  
81 //razorpay  
82 // Razorpay instance  
83 const razorpay = new Razorpay({  
84   key_id: 'rzp_test_o08YVZwKcCbv9M',  
85   key_secret: 'Kk7Sp1T1mDRiFU4Px2hBNWsR'  
86 });  
87
```

“Fig. 13” Razorpay Payment Gateway Integration code snippets