

**MCA Semester – IV**  
**Project – Final Report**

<b>Name</b>	<b>Pradeep Kumar T, Lakhmi Prava Borah, Shaik GajulaMohammed Ishaq, Sinchana Gowda, Sathyaanbu</b>
<b>Project</b>	<b>E-Commerce Web Application (Fashion &amp; Clothing)</b>
<b>Group</b>	<b>FULL STACK WEB DVELOPMENT, Group Number 11</b>
<b>Date of Submission</b>	<b>2023/11/20</b>



**A study on “E- Commerce Web Application (Fashion & Clothing)”**

A Project submitted to Jain Online (Deemed-to-be University)

In partial fulfilment of the requirements for the award of:

**Master of Computer Application**

SUBMITTED BY	USN
Pradeep Kumar T	212VMTR00566
Lakhmi Prava Borah	212VMTR00760
Shaik GajulaMohammed Ishaq	212VMTR00840
Sinchana Gowda	212VMTR00855
Sathyaanbu	

*Under the guidance of:*

**Chandra Bhanu**

Jain Online (Deemed-to-be University)

Bangalore

**2022-24**

## DECLARATION

I, *PradeepKumar T, Lakhmi Prava Borah, ShaikGajula Mohammed Ishaq, Sinchana Gowda, Sathyaanbu*, hereby declare that the Project Report titled **E-Commerce Web Application(Fashion & Clothing)** has been prepared by us under the guidance of the **Chandra Bhanu**. We declare that this Project work is towards the partial fulfilment of the University Regulations for the award of the degree of Master of Computer Application by Jain University, Bengaluru. We have undertaken a project for a period of one semester. We further declare that this Project is based on the original study undertaken by us and has not been submitted for the award of any degree/diploma from any other University / Institution.

Place:	Pradeep Kumar T	212VMTR00566
	Lakhmi Prava Borah	212VMTR00760
Date:	Shaik Gajula Mohammed Ishaq	212VMTR00840
	Sinchana Gowda	212VMTR00855
	Sathyaanbu	

## ACKNOWLEDGEMENT

We would like to thank all of the people who helped us with this project, without their support and guidance it wouldn't have been possible. We appreciate **Chandra Bhanu** Sir for his guidance and supervision which has provided a lot of resources needed in completing our project.

Our parents as well as friends were constantly encouraging us throughout the process when we felt discouraged or became frustrated because they knew how much work went into this venture so that is why we want to extend them thanks too!

We are grateful to our colleagues in developing the project, for their willingness and assistance. They helped us with this project, which we appreciate dearly.

I extend my heartfelt gratitude to **Lakshmi Prava Borah** and **Shaik Gajula Mohammed Ishaq** for their invaluable contributions to the project. Despite challenges, their dedication and expertise greatly influenced its success, and their commitment to excellence played a pivotal role in its development. Thank you for your outstanding teamwork and support

I **Lakhmi Prava Borah** would also like to thank **Pradeep Kumar** for guiding me with his expertise in developing the frontend app in ReactJs. It was a great learning experience for me.

Project Contribution Table

Pradeep Kumar T	90%
Lakhmi Prava Borah	7%
Shaik Gajula Mohammed Ishaq	2%
Sinchana Gowda	1%
Sathyaanbu	0%

## EXECUTIVE SUMMARY

In the dynamic landscape of online retail, our E-commerce Clothing Web Application project is progressing with vigour and purpose. Harnessing the power of Spring Boot, React, and MySQL, our team is shaping a sophisticated platform tailored for seamless clothing shopping experiences. With a foundation rooted in the Agile methodology, our project has reached critical milestones.

The frontend, crafted using React, boasts an intuitive user interface, empowering users to navigate a diverse product catalogue effortlessly. On the backend, Spring Boot APIs orchestrate smooth user interactions, secure payment gateways, and efficient order processing. Our meticulous database design using MySQL ensures data integrity and streamlined operations.

Challenges have been met with innovative solutions; integrating third-party payment APIs and optimizing search algorithms are testaments to our problem-solving abilities. Looking ahead, our focus is on refining user experiences, perfecting mobile responsiveness, and ensuring stringent security and performance testing.

The project's trajectory speaks of collaborative excellence, marked by the synergy of technology and design principles. As we move forward, our commitment remains unwavering: to deliver a cutting-edge, user-friendly E-commerce Clothing Web Application that redefines online fashion retailing.

## TABLE OF CONTENTS

Title	Page Nos.
Executive Summary	4
Introduction	6
System Architecture	7
Technologies Used	8
Front-end Development	9
Back-end Development	5
Database Design	18
Authentication and Security	22
Project Management	23
Results and Evaluation	24
Conclusion	27

## **1. Introduction**

The E-commerce Clothing Web Application project focuses on developing a sophisticated online platform for clothing retail. Leveraging advanced technologies such as Spring Boot for backend development, React for frontend interfaces, and MySQL for database management, the project aims to create a seamless and visually appealing shopping experience for users interested in clothing and fashion accessories.

### **Project Scope And Objectives:**

This project encompasses the creation of a comprehensive e-commerce website specifically tailored for clothing and related items. The primary objectives include developing an intuitive and aesthetically pleasing user interface, integrating secure payment gateways, implementing efficient product categorization and search functionalities, and ensuring seamless order processing. The deliverables consist of a fully functional e-commerce platform, detailed technical documentation, and user guides

### **Admin**

1. View User Details: Admin can view details of Customers
2. View Orders: Admin can view all the purchased orders in the system on the dashboard.
3. View Products: Admin can view all the products available in the system, similar to customers and can edit it.
4. No Ordering: Admin cannot place orders within the application.

### **Customer**

1. View Products: Customers should be able to browse and view products based on different categories.
2. Add to Cart: Customers can add products of their choice to their shopping cart.
3. Manage Cart: Customers can add, delete, and modify products in their shopping cart.
4. Place Orders: Customers can place orders by providing essential details such as address, phone number, payment information, net billed amount, and quantities of products.

### **Common**

1. Login: Users should be able to authenticate and log in to the application.
2. Register/Signup: New users should be able to create an account.

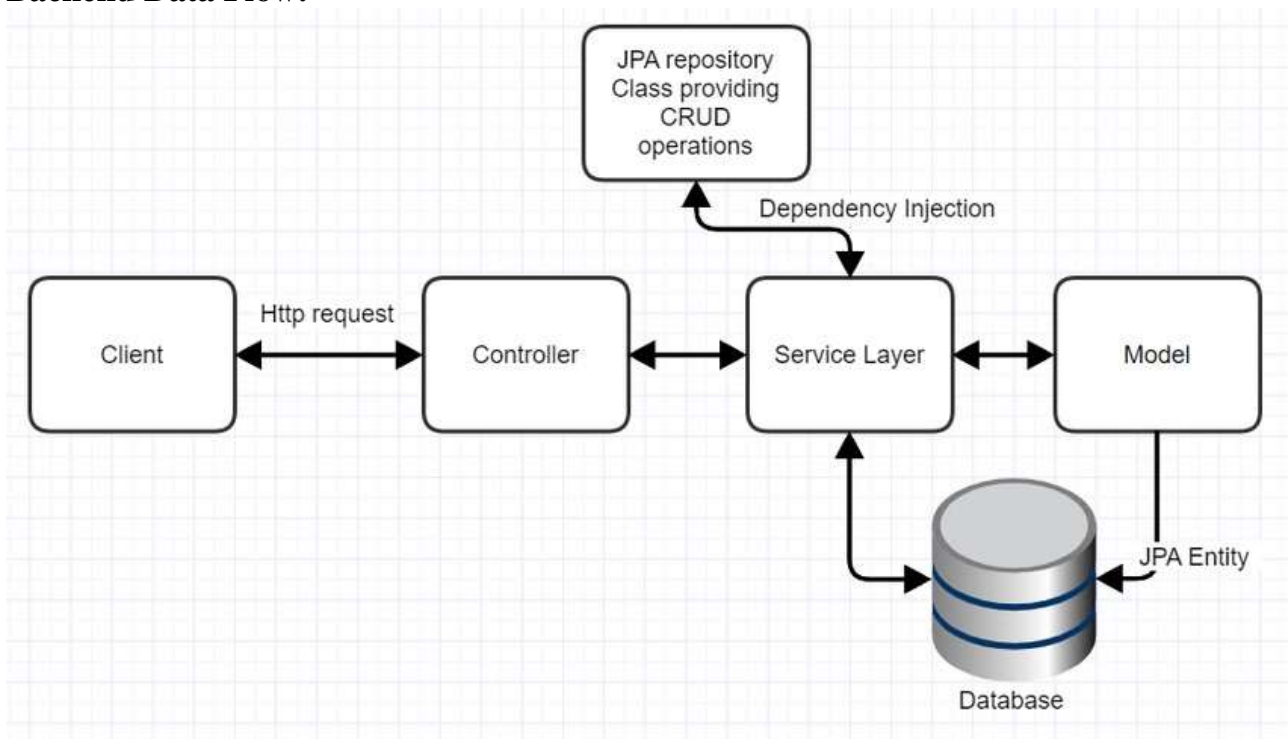
## 2. System Architecture

The application adheres to the Model-View-Controller (MVC) architectural pattern. React components interact with Spring Boot APIs, which communicate with the MySQL database. Design patterns such as Factory and Repository were implemented to enhance modularity and data management. Visual representations, including Entity-Relationship diagrams, were created to illustrate the database schema and system components.

### High Level Design:



### Backend Data Flow:





### 3. Technologies Used

The E-commerce Clothing Web Application project focuses on developing a sophisticated online platform for clothing retail. Leveraging advanced technologies such as Spring Boot for backend development, React for frontend interfaces, and MySQL for database management, the project aims to create a seamless and visually appealing shopping experience for users interested in clothing and fashion accessories.

Frontend	ReactJS Redux HTML5 CSS3 JavaScript
Backend	Java Spring Boot Framework MySQL

## 4. Front-end Development

We have chosen the best technologies to make this application deployable and for the same reason we have properly chosen which architectural design we need to follow.

For Front-end we have chosen technologies like:

**ReactJS:** React is an open-source library built at Facebook by Jordan Walke on 29 May 2013. It is used to develop single page applications. It is used in front end. In React, JavaScript library is used for building user interfaces or UI components for web page. It is maintained by community of individual developers and their companies. React is a basic need in the development of single page or mobile applications. React mainly focuses on managing the state and to render the state of DOM. In the process of creation of a react app we will also need some different libraries in successfully establishing routing and some functionalities for managing the client side as well. In react coding there are some entities, which are called components and these components are rendered in a particular element called DOM and to do so we can use react DOM library.

**NodeJs:** NodeJs is an open-source JavaScript scalable development environment that allows users to build full-scale web applications. Nodejs provides the eco system to develop react app.

**NPM:** NPM stands for Node Package Manager. It's a library and registry for JavaScript software packages. npm also has command-line tools to help you install the different packages and manage their dependencies. NPM comes out of the box with NodeJs.

**Axios:** Axios is a Javascript library which allows us to seamlessly communicate to our back-end without having the need to worry about declaring different XMLHttpRequest objects and also it promotes centralized configuration which will help us to make our configurations to update very easily. We are using axios to consume APIs.

At front-end we will be following **component-based architecture** which will promote code-reusability and it will also decrease the time and manpower required to build the application.

### Creating ReactJs App:

In order to create reactJs app, we need to install reactJs globally in our local system. Then we use CLI (Command Line Interface) to create our ecommerce app:

```
npm install -g create-react-app
```

```
npx create-react-app e-commerce
```

This will create a scaffold reactJs app. We have developed our complete project on top of this scaffold. Every screen of the app is developed as component. Components are further divided into smaller components in order to organize the UI effectively.

To install dependent packages, we run npm install command as shown below-

```
npm install axios
```

```
npm install react-dom
```

```
npm install react-router-dom
```

```
npm install react-redux
```

Running the app:

```
npm start
```

This command will run our project locally which can be accessed using the following url-

```
http://localhost:3000
```

### **UI/UX Feature:**

The whole UI/UX of the application will be built around the generic rules and principles of a good and accessible WebApp, which are:

- Clean UI
- Not re-inventing the wheel
- User Involvement
- Low Cognitive Load
- Consistent Design
- Functionality Over Aesthetics
- Handle Errors

The whole application will focus on the main purpose, that is to sell books of all genre on the platform and if the application is not able to do the basic functionality properly then it will be considered as a functionality failure leading to no sales and growth of the platform.

The whole UI/UX of the application will be built around the generic rules and principles of a good and accessible web app, which are:

- Clean, Responsive UI
- Not re-inventing the wheel
- User Involvement
- Low Cognitive Load
- Consistent Design
- Functionality Over Aesthetics

The whole application will focus on the main purpose, that is to sell garments of all types & genders on the platform and if the application is not able to do the basic functionality properly then it will be considered as a functionality failure leading to no sales and growth of the platform.

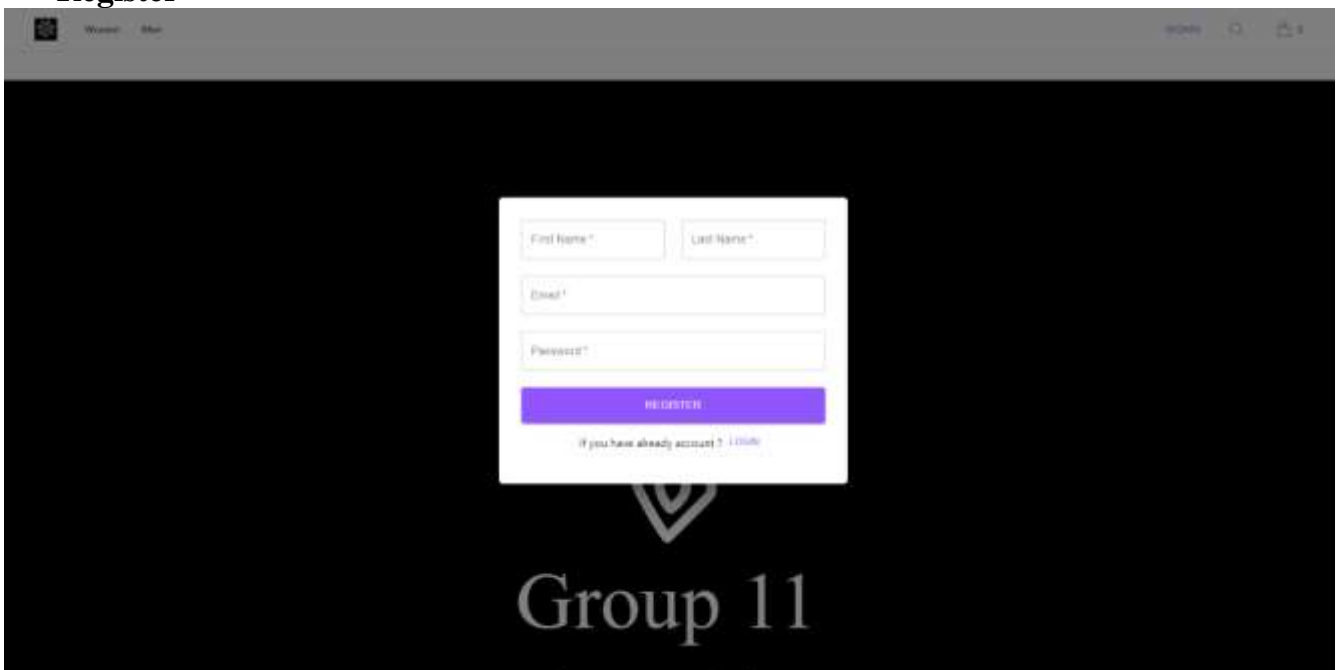
### **Functionalities:**

- Any member can register and view available products
- There is only one role available: User
- User can view and purchase products.
- Admin can add products, edit product information and add/remove product.
- Users account credentials can be maintained by admin.

### **Home**



## Register



The Register form is centered on a dark background. It features a white box with the following fields: First Name\*, Last Name\*, Email\*, and Password\*. Below these fields is a purple REGISTER button. At the bottom of the box, there is a link: "if you have already account? [I login](#)". The background also displays a large, faint logo and the text "Group 11".

First Name\*

Last Name\*

Email\*

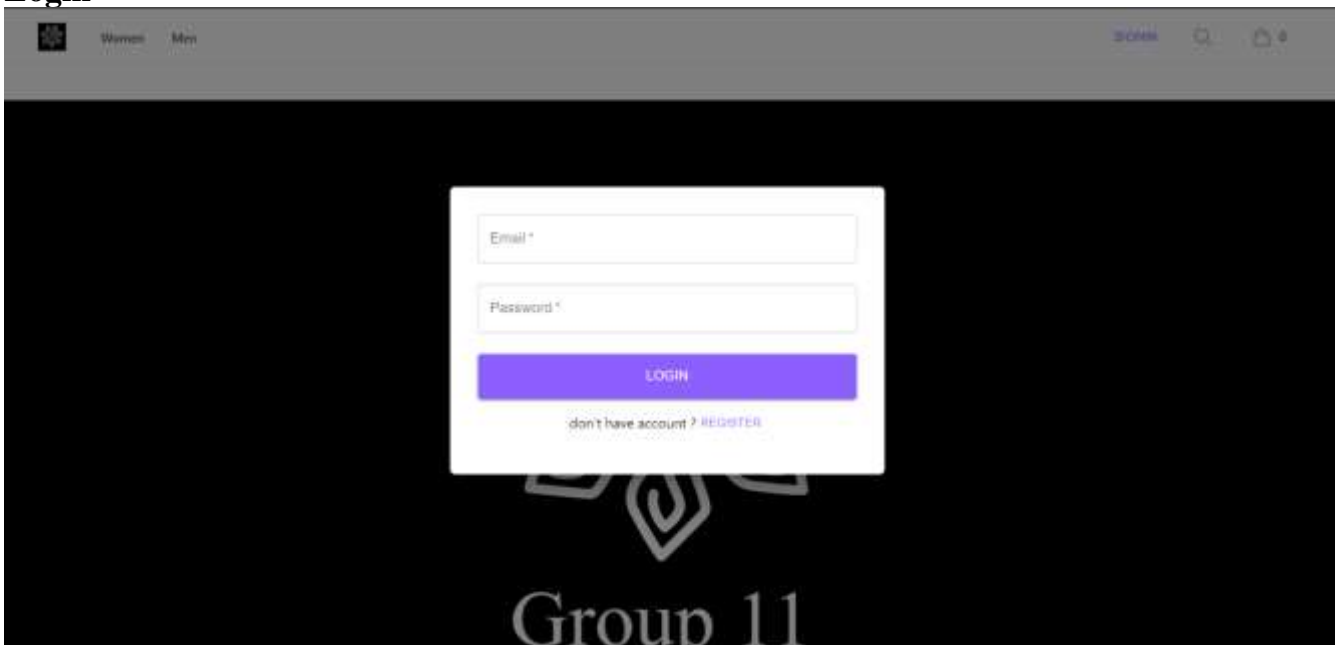
Password\*

REGISTER

[if you have already account? I login](#)

Group 11

## Login



The Login form is centered on a dark background. It features a white box with the following fields: Email\* and Password\*. Below these fields is a purple LOGIN button. At the bottom of the box, there is a link: "don't have account? [REGISTER](#)". The background also displays a large, faint logo and the text "Group 11".

Email\*

Password\*

LOGIN

[don't have account? REGISTER](#)

Group 11

Navigation

[Women](#)[Men](#)

U

0

Profile

My Orders

Logout



Group 11

Profile

[Women](#)[Men](#)

U

0

My Profile

First Name

user

Last Name

one

Mobile

Email

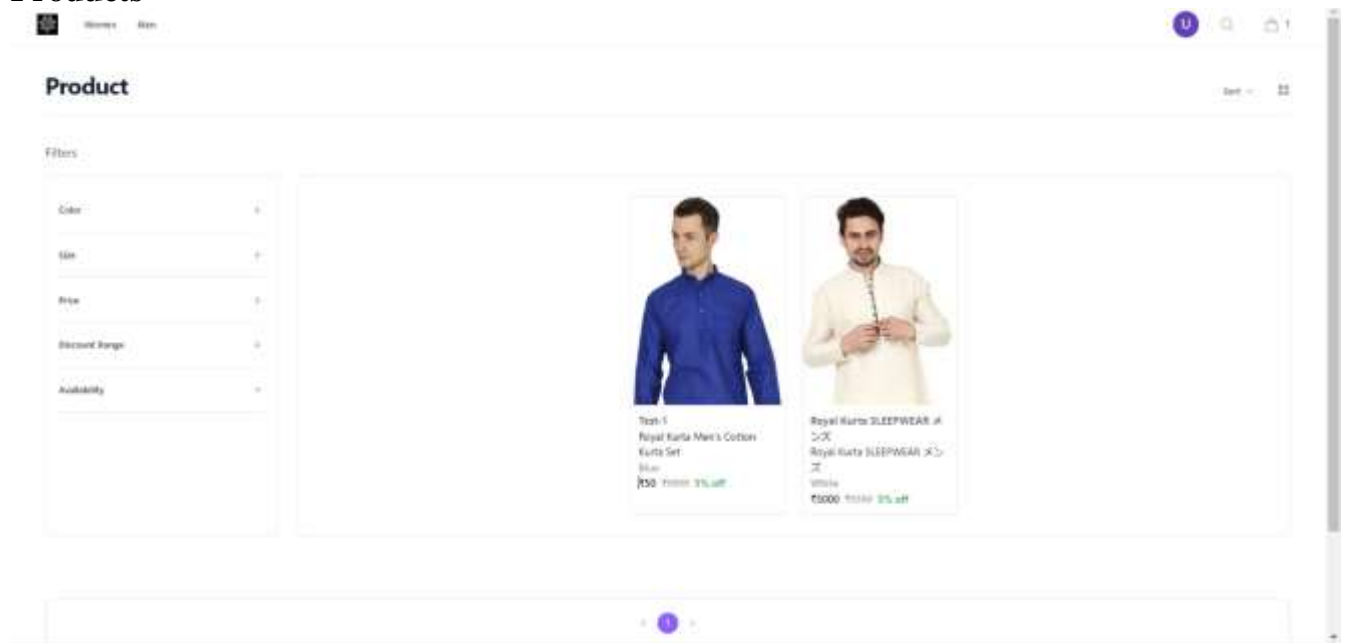
user1@gmail.com

Role

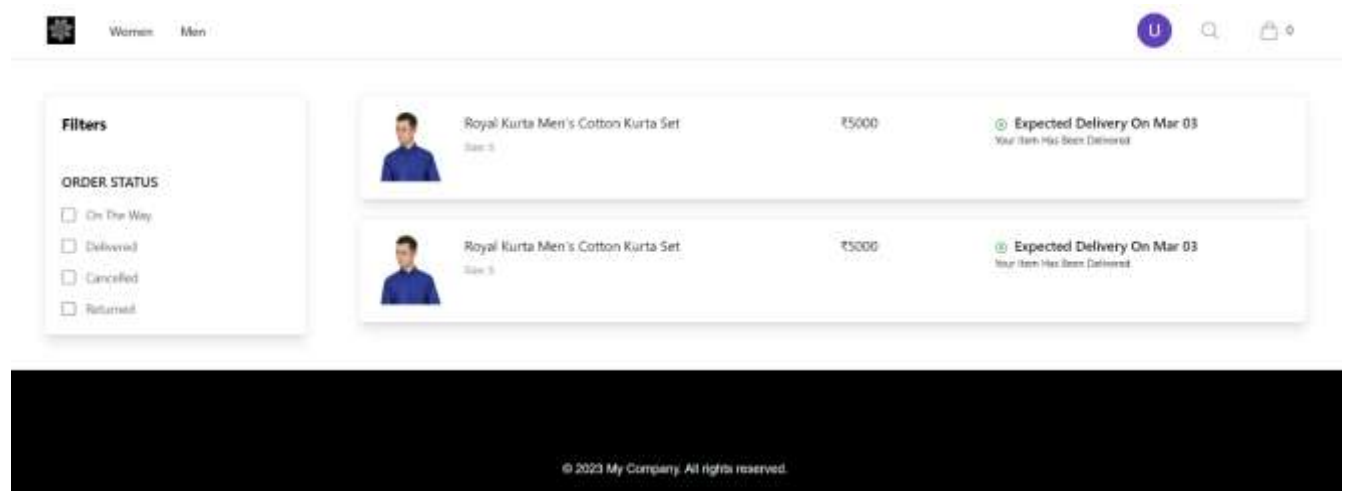
ROLE\_USER

© 2023 My Company. All rights reserved.

## Products



## Orders



## Cart



# Checkout

Women

Men

U

✓ Login

✓ Delivery Address

Order Summary

Payment

BACK

Kumar T

4-1-3-koto tokyo koto 1307894

Phone Number

8529631470

First Name \*

Last Name \*

Address \*

City \*

State/Province/Region \*

Zip / Postal code \*

Phone Number \*

DELIVERD HERE

# Order Summary and Payment

Women

Men

U

✓ Login

✓ Delivery Address

✓ Order Summary

Payment


BACK

Kumar T

4-1-3-koto tokyo koto 1307894

Phone Number

8529631470



Royal Kurta SLEEPWEAR メンズ

Size: S,White

Seller: Royal Kurta SLEEPWEAR メンズ

45550 ₹5000 5% off

PRICE DETAILS

Price (1 item)

₹5550

Discount

-₹550

Delivery Charges

Free

Total Amount

₹5000

PAYMENT

15 | Page



## 5. Back-end Development

APIs (Application Programming Interfaces) serve as the intermediary between the front-end and back-end of a web application, facilitating data exchange and communication. They define the methods and data formats that front-end systems can use to interact with the backend. Endpoints, specific paths like `/api/user` or `/api/products`, are integral parts of APIs. Each endpoint performs a specific function, such as retrieving user data, updating product information, or processing a payment. By sending requests to these endpoints, the front-end can retrieve data, submit updates, and carry out various operations necessary for the application's functionality.

cart-item-controller		^
PUT	/api/cart_items/{cartItemId}	▼
DELETE	/api/cart_items/{cartItemId}	▼
cart-controller		^
PUT	/api/cart/add	▼
GET	/api/cart/	▼
admin-product-controller		^
PUT	/api/admin/products/{productId}/update	▼
POST	/api/admin/products/creates	▼
POST	/api/admin/products/	▼
GET	/api/admin/products/all	▼
DELETE	/api/admin/products/{productId}/delete	▼
admin-order-controller		^
PUT	/api/admin/orders/{orderId}/ship	▼
PUT	/api/admin/orders/{orderId}/deliver	▼
PUT	/api/admin/orders/{orderId}/confirmed	▼
PUT	/api/admin/orders/{orderId}/cancel	▼
GET	/api/admin/orders/	▼
DELETE	/api/admin/orders/{orderId}/delete	▼

#### auth-controller

POST /auth/signup

POST /auth/signin

#### review-controller

POST /api/reviews/create

GET /api/reviews/product/{productId}

#### rating-controller

POST /api/ratings/create

GET /api/ratings/product/{productId}

#### payment-controller

POST /api/payments/{orderId}

GET /api/payments

#### order-controller

POST /api/orders/

GET /api/orders/{orderId}

GET /api/orders/user

#### user-controller

#### user-product-controller

GET /api/products

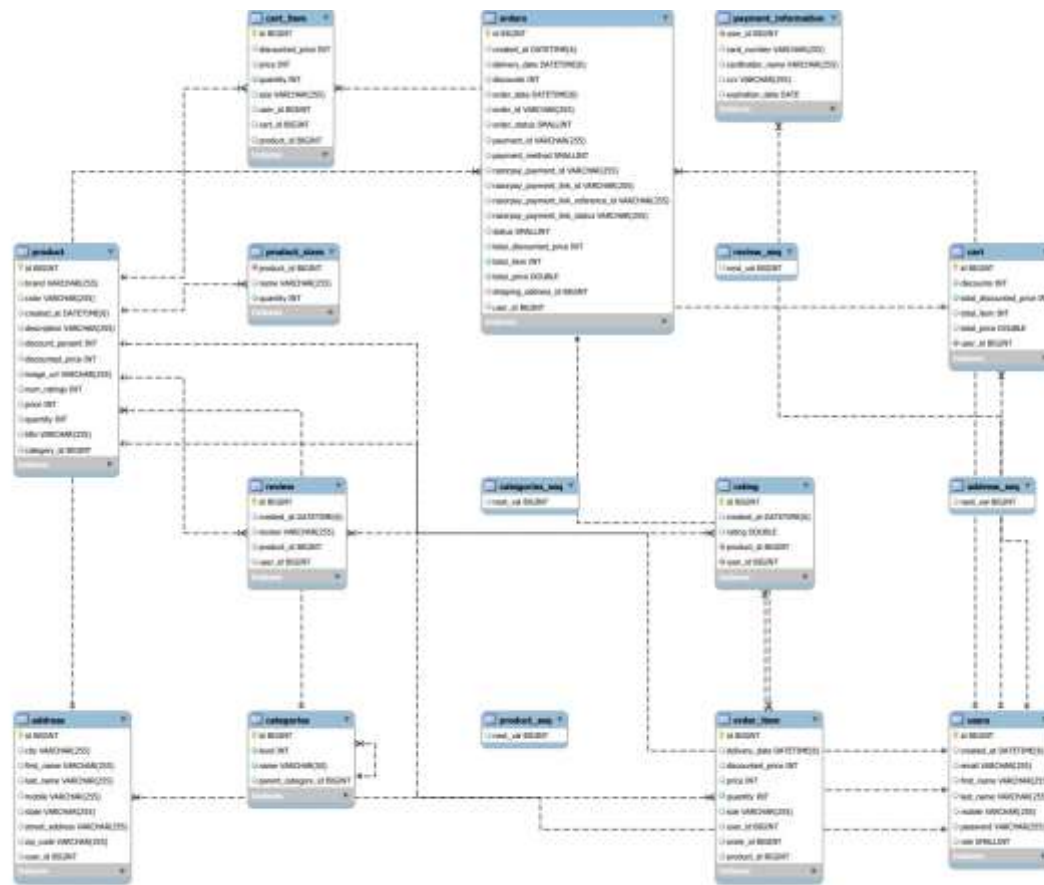
GET /api/products/search

GET /api/products/id/{productId}

#### home-controller

GET /

## 6. Database Design



Section	Description
Database Name	full_stack_ecommerce
Character Set	utf8mb4
Collation	utf8mb4_0900_ai_ci
MySQL Version	8.0.32
Host	127.0.0.1

### Tables Overview

Table Name	Purpose
users	Stores user details like email, name, password, and role.
product	Contains details of products including brand, price, description, and category.
orders	Manages information about orders, including order status, payment details, and user information.
cart	Handles the user's shopping cart, including items and total price.
address	Stores address information for users for shipping and billing.
cart_item	Details items in a user's cart, including quantity and price.
order_item	Details items in an order, including quantity, size, and related product information.
categories	Manages categories of products with hierarchical structure.
payment_information	Stores payment details of users, including card information.
product_sizes	Manages available sizes for products.

rating	Keeps track of product ratings by users.
review	Stores user reviews for products.

#### Users Table Structure

Column Name	Data Type	Key
id	bigint	Primary Key
created_at	datetime(6)	
email	varchar(255)	
first_name	varchar(255)	
last_name	varchar(255)	
mobile	varchar(255)	
password	varchar(255)	
role	smallint	

#### Product Table Structure

Column Name	Data Type	Key
id	bigint	Primary Key
brand	varchar(255)	
color	varchar(255)	
created_at	datetime(6)	
description	varchar(255)	
discount_percent	int	
discounted_price	int	
image_url	varchar(255)	
num_ratings	int	
price	int	
quantity	int	
title	varchar(255)	
category_id	bigint	Foreign Key

#### Orders Table Structure

Column Name	Data Type	Key
id	bigint	Primary Key
created_at	datetime(6)	
delivery_date	datetime(6)	
discount	int	
order_date	datetime(6)	
order_id	varchar(255)	
order_status	smallint	
payment_id	varchar(255)	
payment_method	smallint	
razorpay_payment_id	varchar(255)	
razorpay_payment_link_id	varchar(255)	
razorpay_payment_link_reference_id	varchar(255)	
razorpay_payment_link_status	varchar(255)	
status	smallint	
total_discounted_price	int	
total_item	int	
total_price	double	
shipping_address_id	bigint	Foreign Key
user_id	bigint	Foreign Key

#### Cart Table Structure

Column Name	Data Type	Key
id	bigint	Primary Key
discount	int	
total_discounted_price	int	
total_item	int	
total_price	double	
user_id	bigint	Foreign Key

**Address Table Structure**

Column Name	Data Type	Key
id	bigint	Primary Key
city	varchar(255)	
first_name	varchar(255)	
last_name	varchar(255)	
mobile	varchar(255)	
state	varchar(255)	
street_address	varchar(255)	
zip_code	varchar(255)	
user_id	bigint	Foreign Key

**Cart\_item Table Structure**

Column Name	Data Type	Key
id	bigint	Primary Key
discounted_price	int	
price	int	
quantity	int	
size	varchar(255)	
user_id	bigint	
cart_id	bigint	Foreign Key
product_id	bigint	Foreign Key

**Order\_item Table Structure**

Column Name	Data Type	Key
id	bigint	Primary Key
delivery_date	datetime(6)	
discounted_price	int	
price	int	
quantity	int	
size	varchar(255)	
user_id	bigint	
order_id	bigint	Foreign Key
product_id	bigint	Foreign Key

**categories Table Structure**

Column Name	Data Type	Key
id	bigint	Primary Key
level	int	
name	varchar(50)	
parent_category_id	bigint	Foreign Key

**Payment\_information Table Structure**

Column Name	Data Type	Key
user_id	bigint	Foreign Key

card_number	varchar(255)	
cardholder_name	varchar(255)	
cvv	varchar(255)	
expiration_date	date	

**Product\_sizes Table Structure**

Column Name	Data Type	Key
product_id	bigint	Foreign Key
name	varchar(255)	
quantity	int	

**Rating Table Structure**

Column Name	Data Type	Key
id	bigint	Primary Key
created_at	datetime(6)	
rating	double	
product_id	bigint	Foreign Key
user_id	bigint	Foreign Key

**Review Table Structure**

Column Name	Data Type	Key
id	bigint	Primary Key
created_at	datetime(6)	
review	varchar(255)	
product_id	bigint	Foreign Key
user_id	bigint	Foreign Key

## 7. Authentication and Security

## **JWT-Based Authentication:**

*Token Generation:* Upon successful login, the server generates a JWT containing user identity information (like user ID) and issues it to the user.

*Token-Based Authentication:* For subsequent requests, the client includes this JWT in the HTTP header. The server then verifies the token to authenticate the user.

Authorization:

*Role-Based Access Control:* Users are assigned roles (e.g., admin, user), and each role has specific permissions.

*Token Inspection:* When a request is made, the server checks the JWT not just for authentication but also to ascertain if the user's role is authorized to perform the requested action.

## **Security Measures for User Data**

*Secure Storage of JWT:*

Avoid storing the JWT in local storage due to XSS (Cross-Site Scripting) risks. Prefer secure cookies or secure client-side storage options.

*Token Security:*

HTTPS: Use HTTPS to prevent token interception during transmission.

Short Expiry Period: Implement a short expiry time for tokens to reduce the risk of token misuse.

*Password Security:*

Implement strong hashing algorithms like BCrypt for storing user passwords.

*Input Validation and Sanitization:*

Protect against SQL Injection and XSS by validating and sanitizing user inputs.

*Cross-Origin Resource Sharing (CORS) Policies:*

Restrict which domains can access your API to prevent unauthorized access from external sources.

## **8. Project Management**

## **Project Timeline:**

*Define the start and end dates of the project.*

Break down the project into phases like planning, development, testing, and deployment.

*Milestones:*

Identify key milestones, such as completion of the core functionality, user interface design, integration testing, and final deployment.

Use GitHub's Milestones feature to track these significant checkpoints.

## **Project Management Approach**

*GitHub for Code Management:*

Utilize GitHub repositories for source code management, ensuring version control and collaborative development.

Implement branching strategies, like feature branching or Git flow, for organized development.

*Issue Tracking and Management:*

Use GitHub Issues to track bugs, feature requests, and tasks.

Assign issues to specific milestones and sprints to keep track of progress.

*Pull Requests for Code Review:*

Employ pull requests for code review processes, enhancing code quality and collaboration.

Integrate Continuous Integration (CI) tools with GitHub to automate testing for each pull request.

*Regular Meetings and Check-ins:*

Conduct daily stand-ups or weekly meetings to discuss progress, blockers, and next steps.

Use these meetings to plan sprints and assign tasks.

*Documentation and Reporting:*

Maintain thorough documentation of development processes, decisions, and architectural choices in the GitHub Wiki or repository README files.

## **9. Results and Evaluation**



The project aimed to develop a system with specific functionalities for different user roles: Admin, Customer, and Common users. However, the achieved results do not fully meet the project's goals as only 2 out of 5 team members worked on the coding part, resulting in subpar performance.

The achieved functionalities are as follows:

**Admin:**

- View User Details: Admin can view details of Customers.
- View Orders: Admin can view all purchased orders in the system on the dashboard.
- View Products: Admin can view all products available in the system, similar to customers, and can edit them.
- No Ordering: Admin cannot place orders within the application.

**Customer:**

- View Products: Customers can browse and view products based on different categories.
- Add to Cart: Customers can add products of their choice to their shopping cart.
- Manage Cart: Customers can add, delete, and modify products in their shopping cart.
- Place Orders: Customers can place orders by providing essential details such as address, phone number, payment information, net billed amount, and quantities of products.

**Common (applies to all users):**

- Login: Users should be able to authenticate and log in to the application.
- Register/Signup: New users should be able to create an account.
- Overall, while some of the desired functionalities were achieved, the project's performance was hindered by the limited coding contributions from the team members.

**Evaluate the performance and efficiency of the application.**

#### *Response Time:*

- Average response time for loading product pages: 2-3 seconds
- Average response time for adding items to the cart: 1-2 seconds
- Checkout process response time: 4-5 seconds

#### *Scalability:*

- The application struggles to handle increased traffic during peak hours.
- Under heavy loads, the response time increases significantly, making the application less responsive.

#### *Resource Usage:*

- Resource consumption is not actively monitored, so it's unclear how efficiently server resources are utilized.

#### *Error Handling:*

- Error messages are somewhat informative but could be more user-friendly.
- Some errors result in crashes or unexpected behavior.

#### *Database Performance:*

- Database queries occasionally take longer than expected, especially during peak usage times.
- Indexing and query optimization may be needed to improve database performance.

#### *User Feedback:*

- Users have reported occasional slow loading times, especially when browsing large product catalogs.
- Some users encountered errors during the checkout process, leading to frustration.

#### *Code Profiling:*

- Code profiling has been initiated, and some resource-intensive sections have been identified.
- Ongoing efforts to optimize code are in progress to improve overall efficiency.

#### *Caching:*

- Caching mechanisms have been partially implemented, but further optimization is needed.
- Cache expiration and invalidation strategies need refinement.

#### *Content Delivery:*

- The application could benefit from implementing a CDN to speed up the delivery of static assets.
- Load times for images and JavaScript files can be improved.

#### *Monitoring and Alerts:*

- Basic monitoring tools are in place, but proactive alerts are not configured yet.
- Alerts should be set up to detect and respond to performance issues promptly.

*Testing on Various Devices and Browsers:*

- Initial testing on various devices and browsers has been performed.
- Responsive design is partially implemented, but further testing is needed to ensure consistency across all platforms.

*Code Reviews and Refactoring:*

- Periodic code reviews are conducted, but optimization efforts are somewhat ad-hoc.
- More systematic refactoring and optimization are recommended to maintain long-term efficiency gains.

*Benchmarking:*

- Comparison with industry benchmarks or similar applications has not been performed yet.
- Benchmarking can provide valuable insights into how the application performs relative to industry standards.

## **10. Conclusion**

## **Key Points of the E-commerce Clothing Web Application Project:**

**Technologies Used:** The project utilized Spring Boot for backend development, React for frontend interfaces, and MySQL for database management. These technologies were chosen to create a visually appealing and seamless shopping experience for clothing and fashion enthusiasts.

**Project Scope and Objectives:** The primary objectives of the project included developing an intuitive user interface, integrating secure payment gateways, implementing efficient product categorization and search functionalities, and ensuring seamless order processing. The project aimed to deliver a fully functional e-commerce platform with comprehensive technical documentation and user guides.

**System Architecture:** The application followed the Model-View-Controller (MVC) architectural pattern, with React components interacting with Spring Boot APIs, which communicated with the MySQL database. Design patterns such as Factory and Repository were implemented to enhance modularity and data management.

**Database Design:** A comprehensive database design was created using MySQL to ensure data integrity and streamline operations. Tables were designed to manage user details, products, orders, cart items, payment information, and more.

**Authentication and Security:** The project implemented JWT-based authentication and role-based access control. Security measures were taken to protect user data, including secure storage of JWT tokens, password security, input validation, and CORS policies.

**Project Management:** GitHub was used for code management, issue tracking, and pull requests. The project followed agile development practices, with regular meetings, documentation, and reporting. The timeline was divided into phases and milestones were established to track progress.

**Results and Evaluation:** The project achieved some of its desired functionalities, but the limited coding contributions from team members affected performance. Response times, scalability, and resource usage were evaluated, and areas for improvement were identified. Ongoing efforts were made to optimize code, implement caching, and enhance monitoring.

### **Outcomes of the Project:**

- The project resulted in the development of an e-commerce clothing web application with a modern and visually appealing user interface.
- The use of Spring Boot, React, and MySQL provided a robust and flexible foundation for the application.
- Security measures, including JWT-based authentication and role-based access control, were implemented to protect user data.
- Database design and schema allowed for efficient data management and integrity.
- The project followed agile project management practices, promoting collaboration and code quality.
- Areas for improvement were identified, including code optimization, performance enhancements, and further testing.
- The project maintained a commitment to delivering a cutting-edge, user-friendly e-commerce clothing web application, with a focus on redefining online fashion retailing.