

Naan Mudhalvan Project

E-commerce Application

INTRODUCTION

We proudly introduce our web application in e-commerce digital platform to order your daily needs effeciently and with ease.

We named our application : **GKMcart**

This application is no different from various applications out there in user interface level, but has the ability to work effeciently with the help of Mongo-DB.

Which is an Database management system that works with your data you enter and provide the service that needed according to the system.

GKMcart is your one-stop destination for effortless online shopping. With a user-friendly interface and a comprehensive product catalog, finding the perfect items has never been easier. Seamlessly navigate through detailed product descriptions, customer reviews, and available discounts to make informed decisions.

Enjoy a secure checkout process and receive instant order confirmation. For sellers, our robust dashboard provides efficient order management and insightful analytics to drive business growth. Experience the future of online shopping with GKMcart today.

TECHNICAL ARCHITECTURE:

In this architecture:

The frontend is represented by the "Frontend" section, including user interface components such as User Authentication, Cart, Products, Profile, Admin ,dashboard, etc.,

The backend is represented by the "Backend" section, consisting of API endpoints for Users, Orders, Products, etc.,It also includes Admin Authentication and an Admin Dashboard.

The Database section represents the database that stores collections for Users, cart, Orders and Product.

Backend Development

1. Setup express server:

- Create index.js file.
- Create an express server on your desired port number.
- Define API's

Now your express is successfully created.

Set Up Project Structure:

- Create a new directory for your project and set up a package.json file using the npm init command.
- Install necessary dependencies such as Express.js, Mongoose, and other required packages.

2. Database Configuration:

- Set up a MongoDB database either locally or using a cloud-based MongoDB service like MongoDB Atlas or use locally with MongoDB compass.
- Create a database and define the necessary collections for admin, users, products, orders and other relevant data.

3. Create Express.js Server:

- Set up an Express.js server to handle HTTP requests and serve API endpoints.
- Configure middleware such as body-parser for parsing request bodies and cors for handling cross-origin requests.

4. Define API Routes:

- Create separate route files for different API functionalities such as users, orders, and authentication.
- Define the necessary routes for listing products, handling user registration and login, managing orders, etc.
- Implement route handlers using Express.js to handle requests and interact with the database.

5. Implement Data Models:

- Define Mongoose schemas for the different data entities like products, users, and orders.
- Create corresponding Mongoose models to interact with the MongoDB database.

- Implement CRUD operations (Create, Read, Update, Delete) for each model to perform database operations.

6. User Authentication:

- Create routes and middleware for user registration, login, and logout.
- Set up authentication middleware to protect routes that require user authentication.

7. Admin Functionality:

- Implement routes and controllers specific to admin functionalities such as adding products, managing user orders, etc.
- Add necessary authentication and authorization checks to ensure only authorized admins can access these routes.

Database Development

Create database in cloud

- Install Mongoose.
- Create database connection.

1. User Schema:

- Schema: userSchema
- Model: 'User'
- The User schema represents the user data and includes fields such as username, email, and password.

- It is used to store user information for registration and authentication purposes.
- The email field is marked as unique to ensure that each user has a unique email address

2. Product Schema:

- Schema: productSchema
- Model: 'Product'
- The Product schema represents the data of all the products in the platform.
- It is used to store information about the product details, which will later be useful for ordering .

3. Orders Schema:

- Schema: ordersSchema
- Model: 'Orders'
- The Orders schema represents the orders data and includes fields such as userId, product Id, product name, quantity, size, order date, etc.,
- It is used to store information about the orders made by users.
- The user Id field is a reference to the user who made the order.

4. Cart Schema:

- Schema: cartSchema
- Model: 'Cart'

- The Cart schema represents the cart data and includes fields such as userId, product Id, product name, quantity, size, order date, etc.,
- It is used to store information about the products added to the cart by users.
- The user Id field is a reference to the user who has the product in cart.

5. Admin Schema:

- Schema: adminSchema
- Model: 'Admin'
- The admin schema has essential data such as categories, banner.

Frontend development

1. Setup React Application:

- Create a React app in the client folder.
- Install required libraries
- Create required pages and components and add routes.

2.Design UI components:

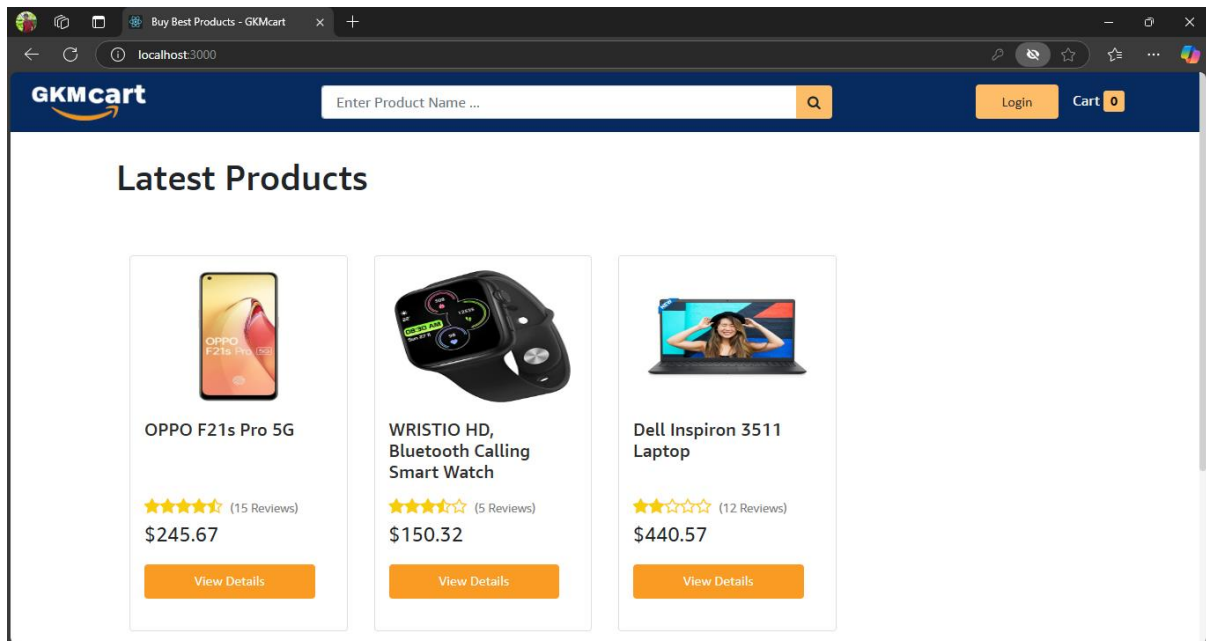
- Create Components.
- Implement layout and styling.
- Add navigation.

3.Implement frontend logic:

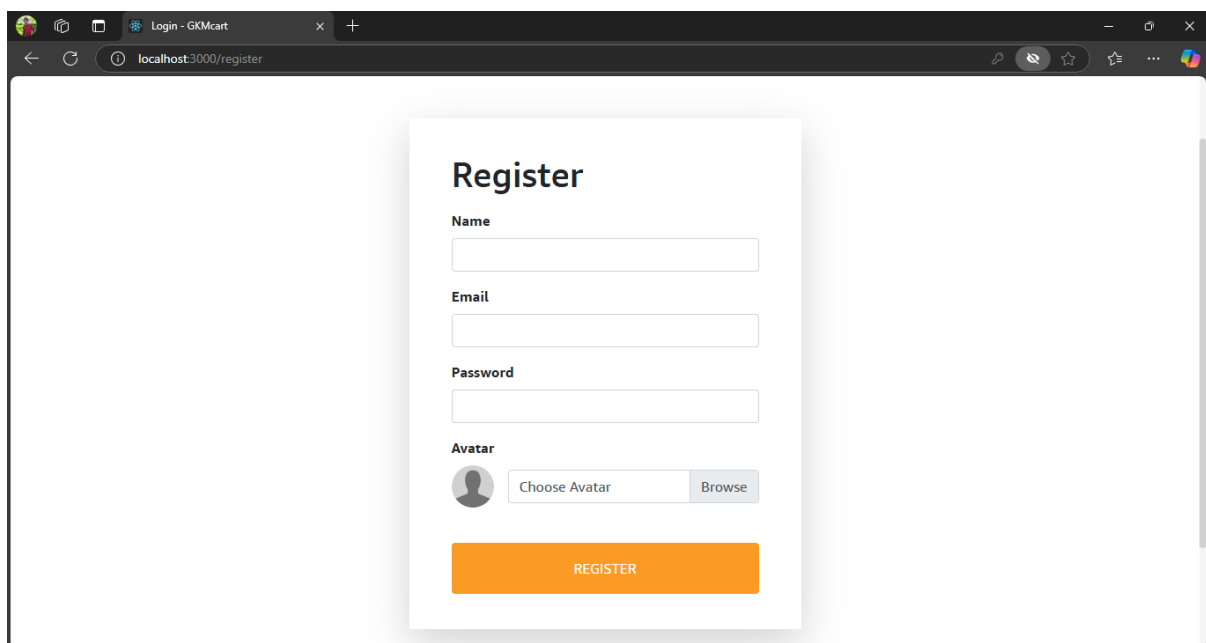
- Integration with API endpoints.
- Implement data binding.

Demo of the application:

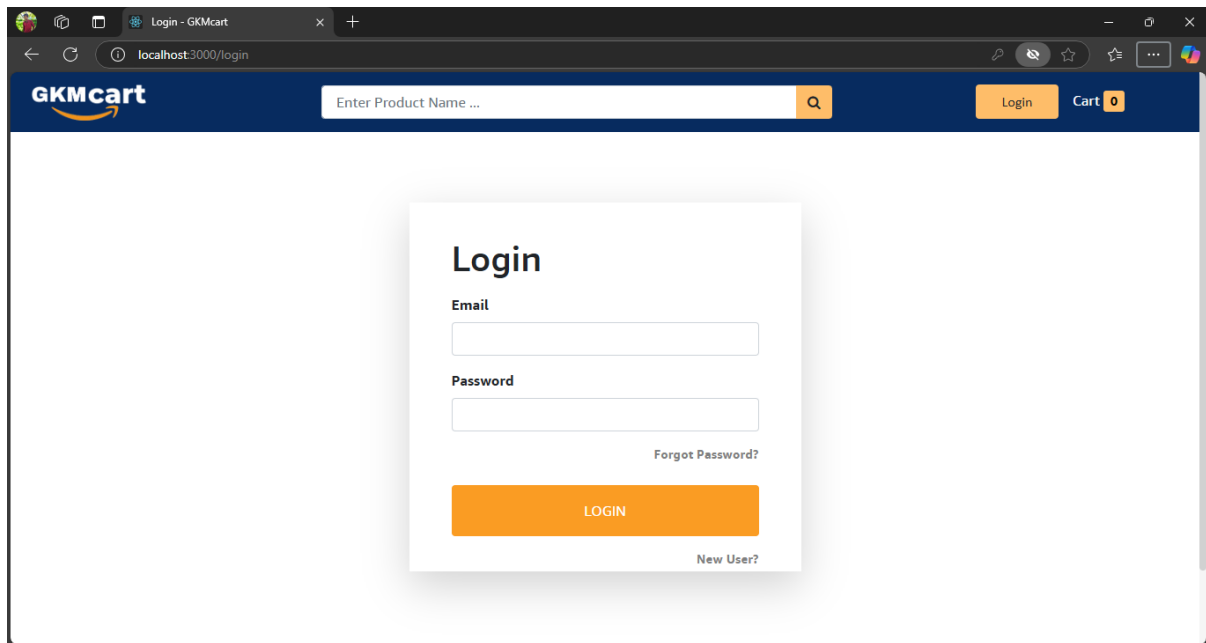
Home page:



Register page:



Login Page:



The screenshot shows a web browser window with the URL `localhost:3000/login`. The page features a dark blue header with the GKMcart logo, a search bar, and a 'Login' button. A central white login form is displayed, containing fields for 'Email' and 'Password', a 'Forgot Password?' link, a 'LOGIN' button, and a 'New User?' link.

GKMcart Enter Product Name ... Login Cart 0

Login

Email

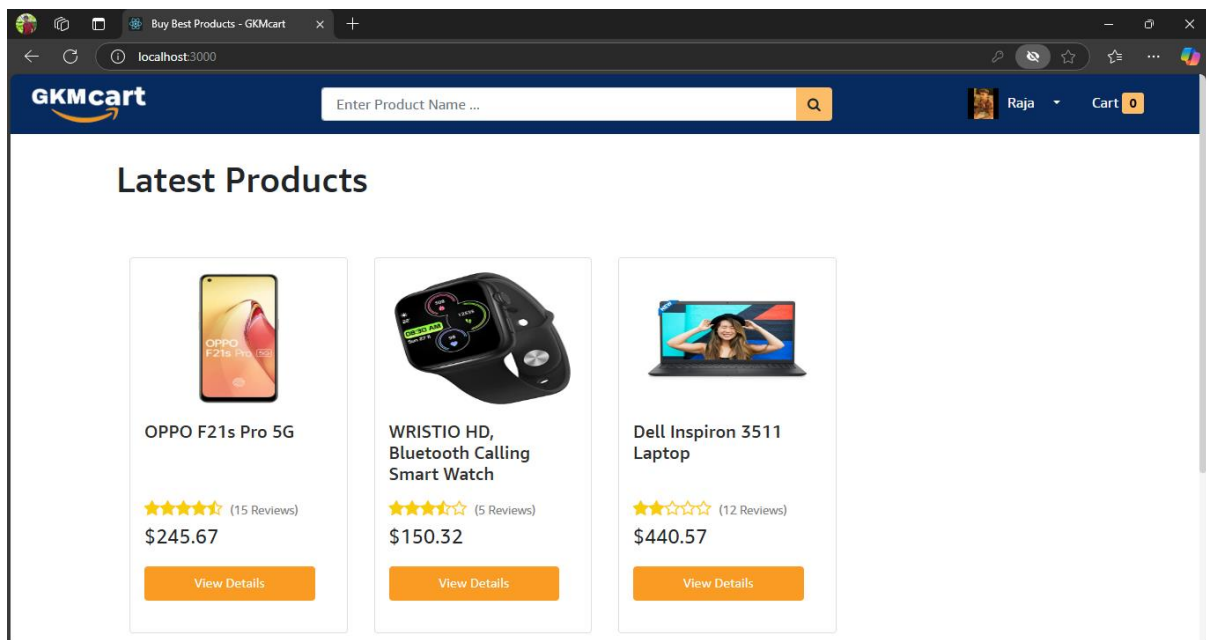
Password

[Forgot Password?](#)

[LOGIN](#)

[New User?](#)


After login home Page:



The screenshot shows the GKMcart home page after a user has logged in. The header now includes the user's name 'Raja' and a 'Cart 0' button. The main content area is titled 'Latest Products' and displays three product cards: 'OPPO F21s Pro 5G', 'WRISTIO HD, Bluetooth Calling Smart Watch', and 'Dell Inspiron 3511 Laptop'. Each card shows the product image, name, star rating, number of reviews, price, and a 'View Details' button.

GKMcart Enter Product Name ... Raja Cart 0

Latest Products




OPPO F21s Pro 5G

★★★★★ (15 Reviews)

\$245.67

[View Details](#)




WRISTIO HD, Bluetooth Calling Smart Watch

★★★★★ (5 Reviews)

\$150.32

[View Details](#)



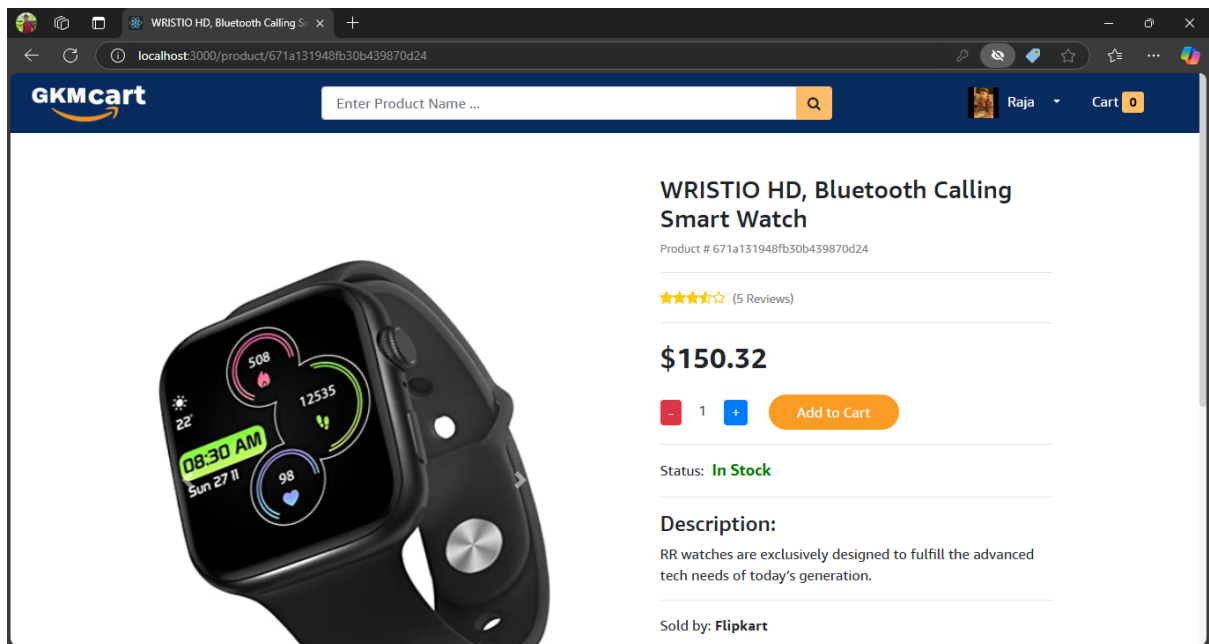
Dell Inspiron 3511 Laptop

★★★★★ (12 Reviews)

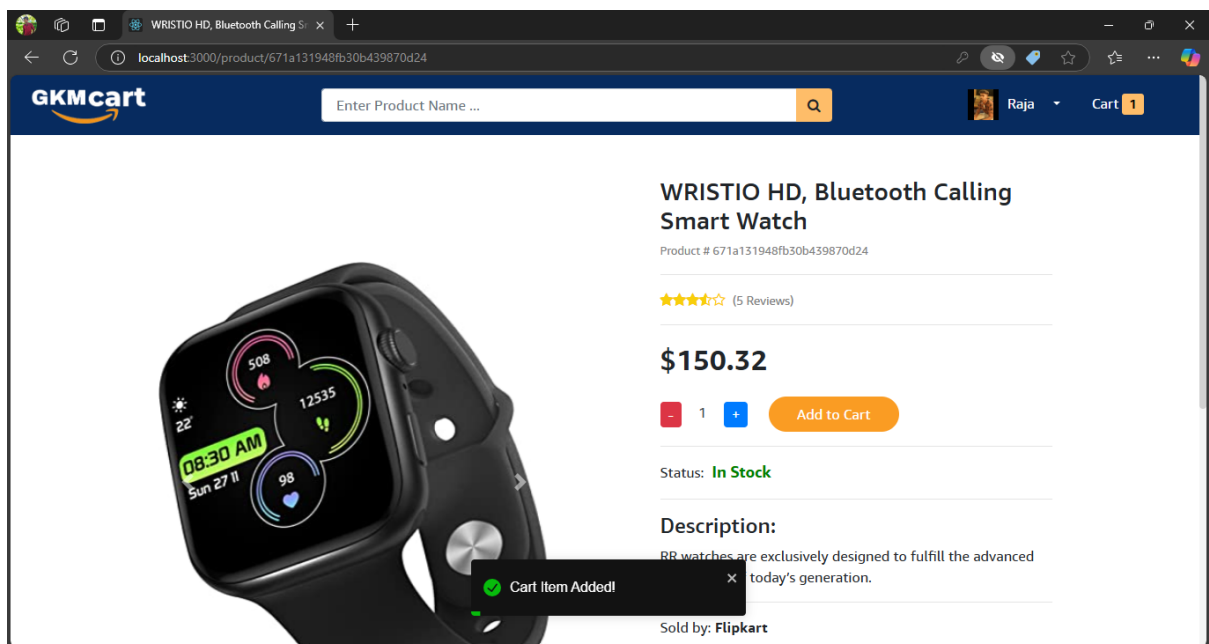
\$440.57

[View Details](#)

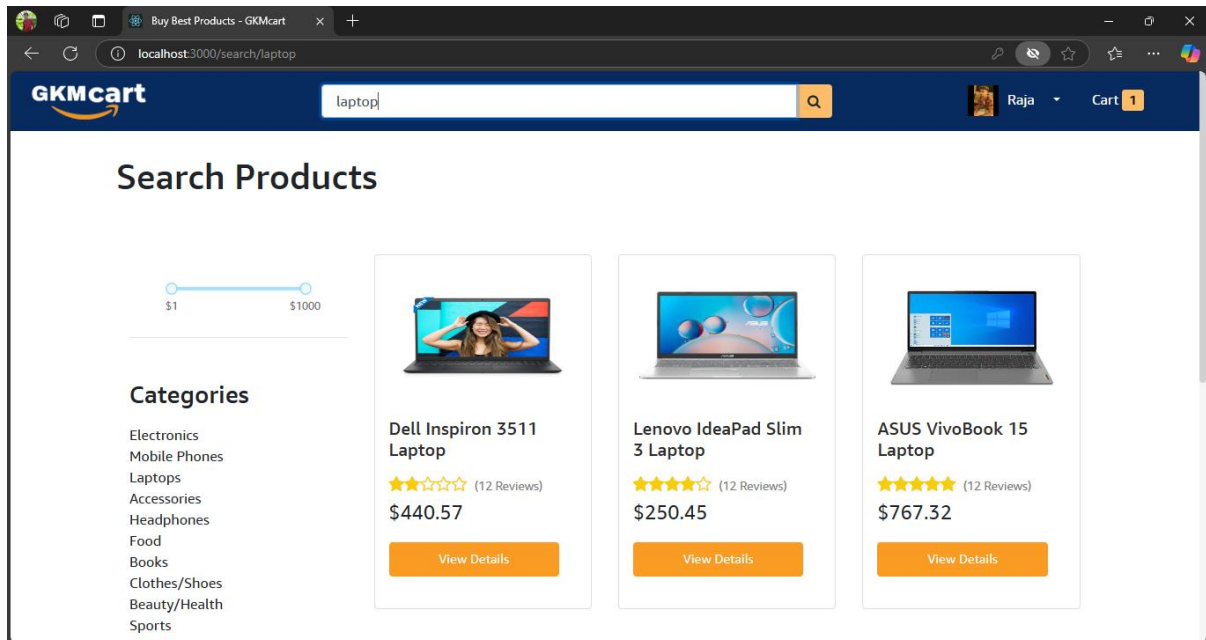
Visiting a product:



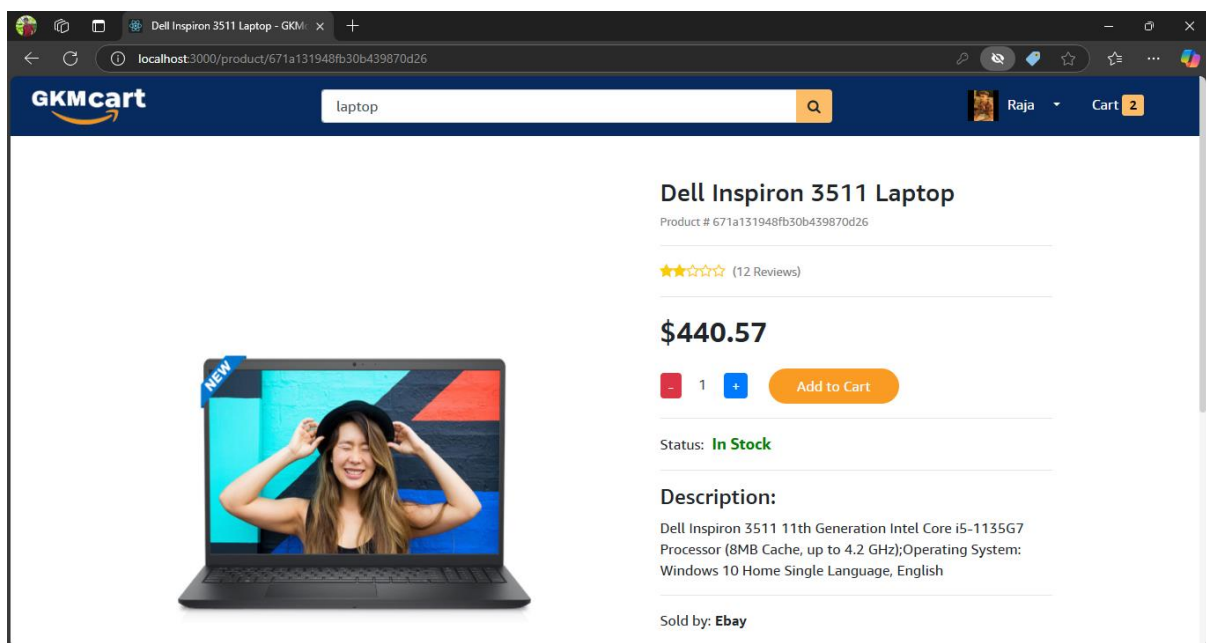
Adding to cart:



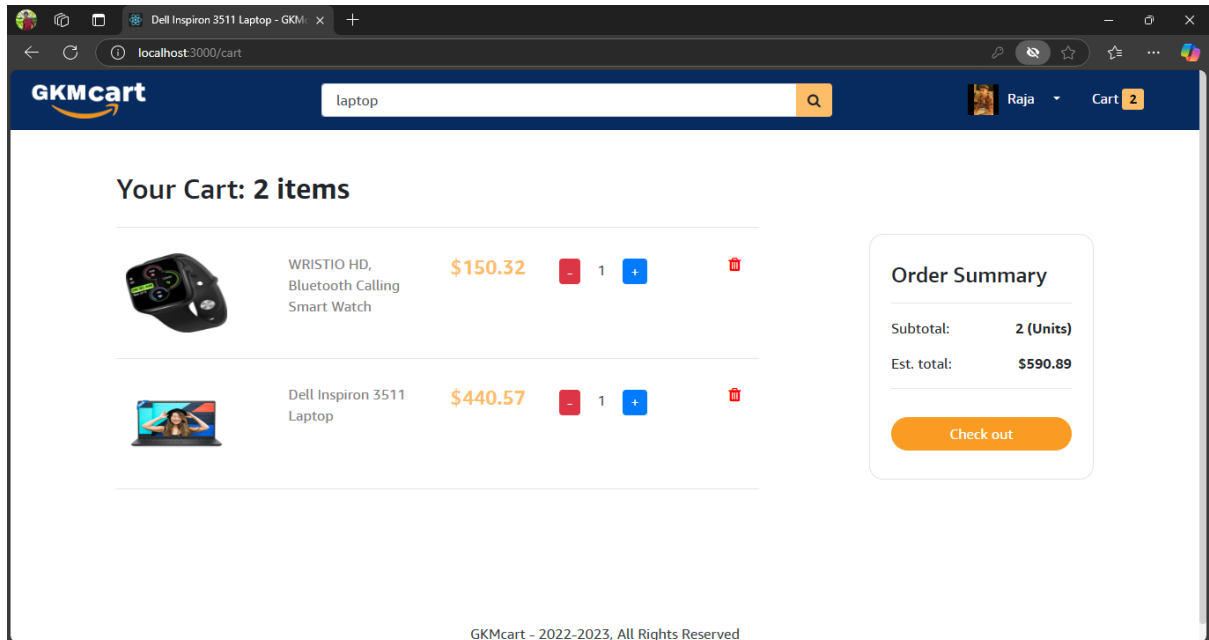
Searching for a product:



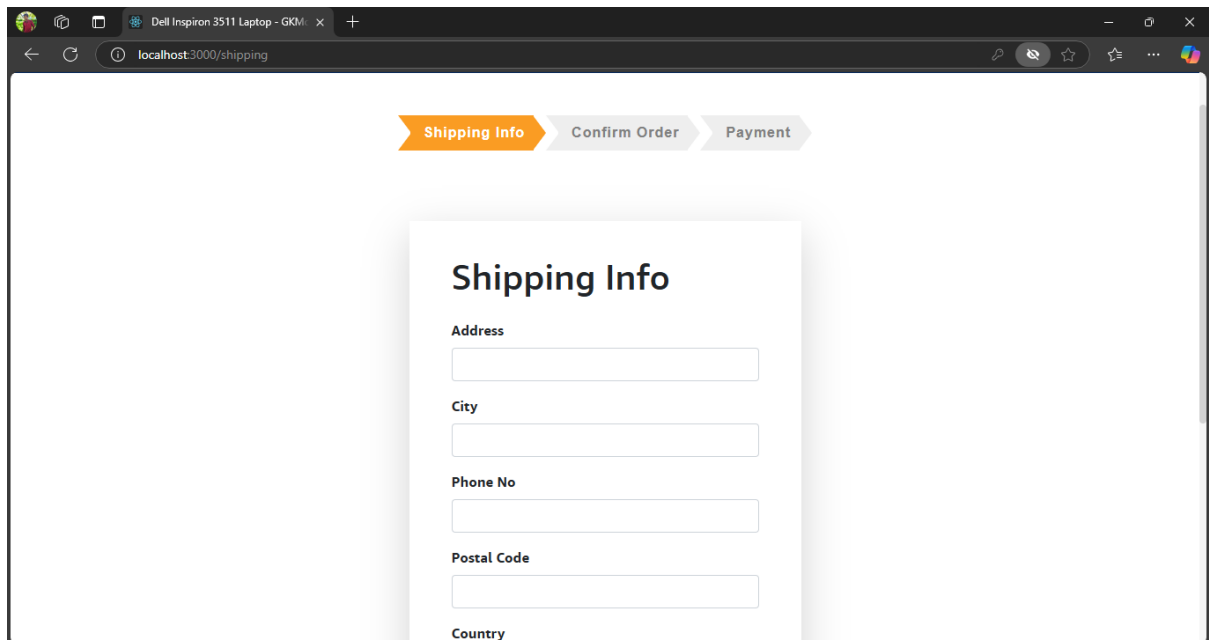
Selecting another product:



Visiting our cart:



Placing order:





Confirming order:

The screenshot shows a web browser window with the URL `localhost:3000/order/confirm`. The page has a navigation bar with three tabs: **Shipping Info**, **Confirm Order** (active), and **Payment**.

Shipping Info

Name: Raja
Phone: 07200484930
Address: no 7 appar samy koil street, chennai, 600004, Taml, India

Your Cart Items:

	WRISTIO HD, Bluetooth Calling Smart Watch	1 x \$150.32 = \$150.32
	Dell Inspiron 3511 Laptop	1 x \$440.57 = \$440.57

Order Summary

Subtotal:	\$590.89
Shipping:	\$0
Tax:	\$29.54
Total:	\$620.43

[Proceed to Payment](#)

Payment process:

The screenshot shows a web browser window with the URL `localhost:3000/payment`. The page has a dark blue header with the **GKMcart** logo, a search bar, and a user profile section showing **Raja** and **Cart 2**.

Card Info

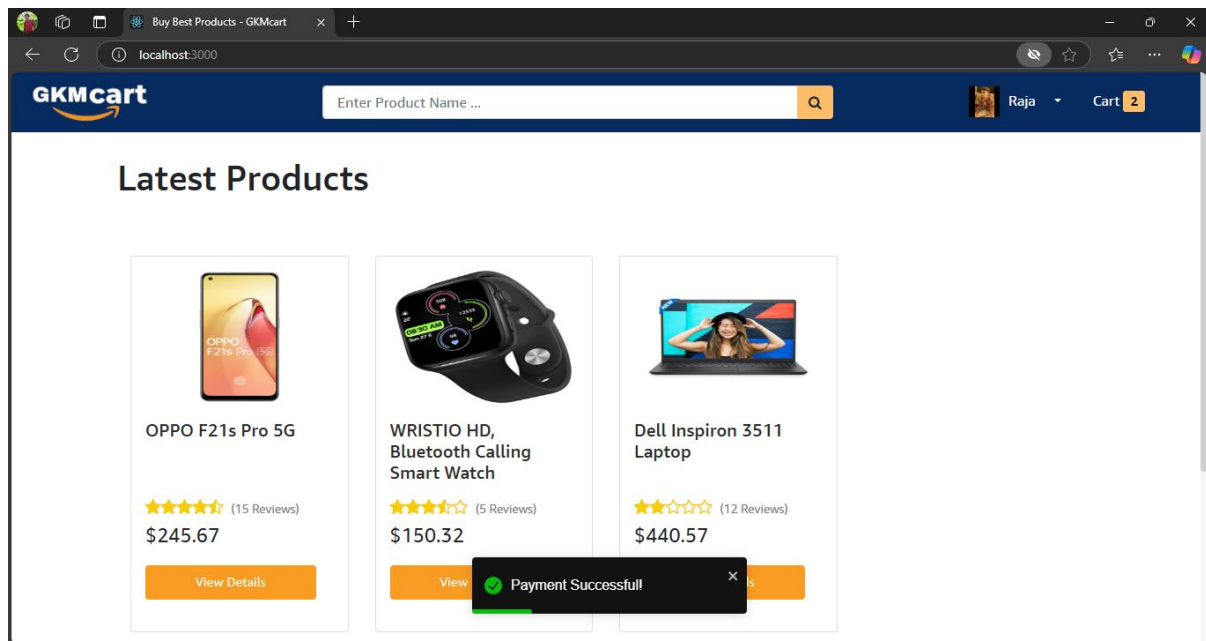
Card Number

Card Expiry

Card CVC

[Pay - \\$620.43](#)

Ordered Successfully :



Outro:

Therefore now the application is functionable.

The code presented in the github can be run and the same will done.

Thank you for spending your time in our project.

-Team.