ShopEase – A MERN-Based E-Commerce Platform

A PROJECT REPORT

Submitted by: ***Gursharan Singh***

23BAI70554

in partial fulfilment for the award of the degree of

Bachelor Of Engineering (BE)

IN

in Computer Science & Engineering (CSE)

Specialization in Artificial Intelligence & Machine Learning (in collaboration with IBM)

Chandigarh University

November & 2025

BONAFIDE CERTIFICATE

Certified that this project report **"E-COMMERCE WEBSITE"** is the bonafide work of **"GUNPREET SINGH AND SPARSH SAHNI"** who carried out the project work under my/our supervision.

SIGNATURE:

SIGNATURE:
(Signature of the AGM-Technical)

EXTERNAL EXAMINER:

INTERNAL EXAMINER:

Submitted for the project viva-voce examination held on :

E-Commerce Website

Project Description

The **E-Commerce Website** project aims to create a dynamic and user-friendly online shopping platform where customers can browse, select, and purchase products effortlessly. This full-stack web application provides a seamless experience for both users and administrators, integrating essential features of a real-world e-commerce system.

The application is built using **MongoDB, Express.js, React.js, and Node.js (MERN stack)**, ensuring a scalable, secure, and efficient architecture. It demonstrates comprehensive full-stack development skills, covering frontend design, backend logic, and database management.

The system enables users to:

- Create and manage user accounts with secure authentication and authorization.
- Browse products by category, brand, or keyword using advanced search and filtering.
- Add items to the cart and proceed with a smooth checkout process.
- View order history and track order status in real-time.
- Manage personal profiles, addresses, and payment preferences.

For administrators, the platform includes a powerful dashboard to:

- Add, update, and delete products.
- Manage inventory and monitor sales.
- View customer activity and generate analytical reports.

This project follows best practices in modern software engineering, including **MVC architecture**, **RESTful APIs**, **form validation**, **error handling**, and **secure database operations**. The final system provides a **responsive, efficient, and interactive** shopping experience while showcasing strong technical proficiency in full-stack web development.

ER DIAGRAM

The system consists of four core entities: **User**, **Product**, **Order**, and **Cart**. Their relationships establish the foundation of the e-commerce platform's functionality.

Entities & Relationships:

User → Order : One-to-Many (A user can place multiple orders)

User → Cart : One-to-One (Each user has a unique cart)

Product → Order : Many-to-Many (A product can appear in many orders, and an order can contain multiple products)

Product → Cart : Many-to-Many (A product can be added to many carts, and a cart can contain multiple products)

User → Review : One-to-Many (A user can submit multiple product reviews)

Product → Review : One-to-Many (Each product can have multiple reviews)

DATABASE SCHEMA

Below is the high-level database schema designed using **MongoDB** and **Mongoose**.

User Schema

name: String

email: String (unique)

password: String (hashed)

phone: String

address: Array of Objects (for multiple addresses)

isAdmin: Boolean

cart: Cart ID

orders: Array of Order IDs

createdAt: Date

Product Schema

name: String

description: String

price: Number

category: String

stock: Number

image: String (Cloudinary URL)

ratings: Array of Review IDs

createdAt: Date

Cart Schema

user: User ID

products: Array of Objects → { productId: Product ID, quantity: Number }

totalPrice: Number

updatedAt: Date

Order Schema

user: User ID

items: Array of Objects → { productId: Product ID, quantity: Number, price: Number }

totalAmount: Number

status: String (e.g., “Pending”, “Shipped”, “Delivered”)

paymentMethod: String

createdAt: Date

Review Schema

rating: Number (1–5)

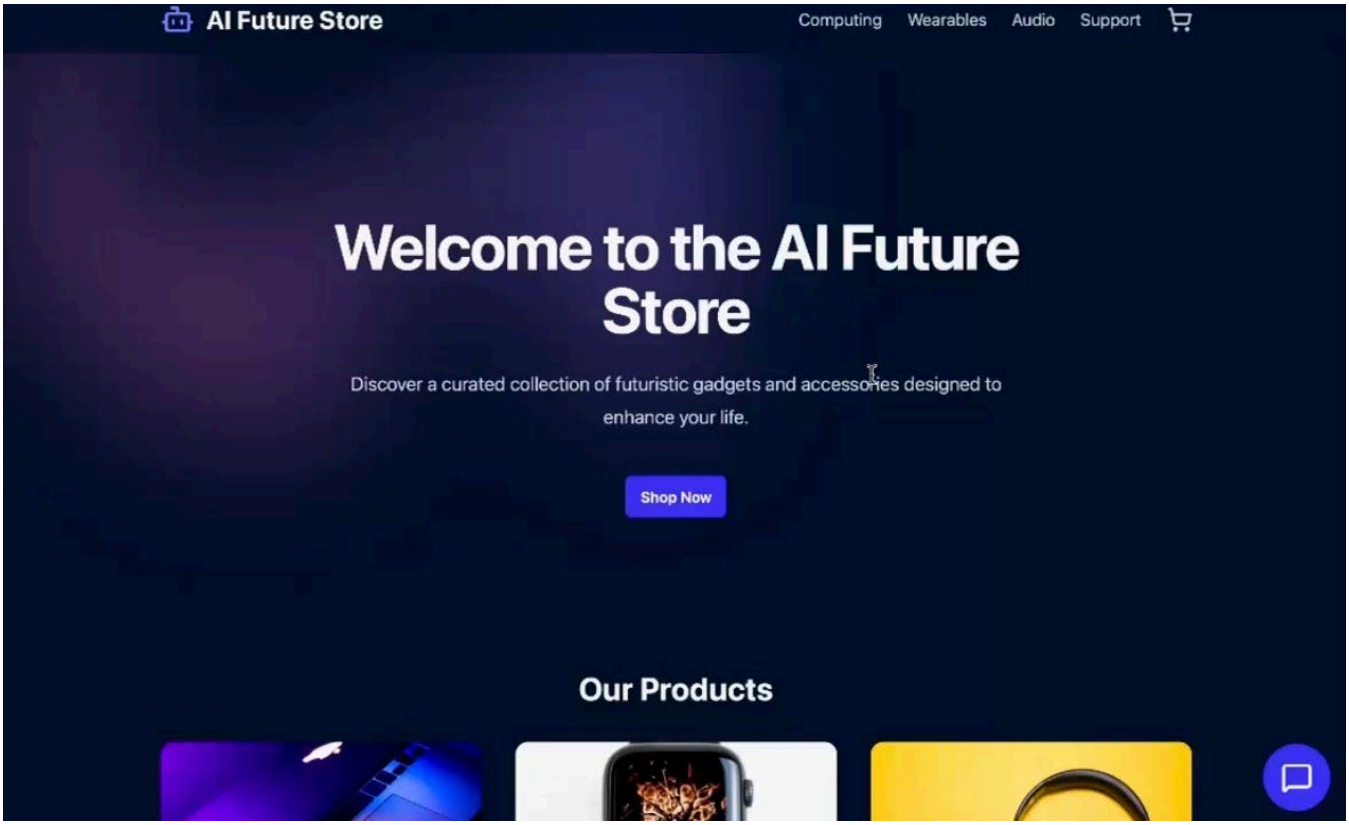
comment: String

user: User ID

product: Product ID

createdAt: Date

FRONT-E



ND SCREENS

AI Future Store

Computing Wearables Audio Support

Computing

Quantum-Core Laptop

A sleek, powerful laptop with a quantum processor for lightning-fast computing.

$2499.99 Add

AI Generative Art Canvas

A digital canvas that reads your creative intent and generates stunning, unique artwork in seconds.

$2200.00 Add

Quantum External Drive

Terabytes of data transferred in seconds. The speed of quantum computing in the palm of your hand.

$950.00 Add

AI Future Store

Computing Wearables Audio Support

Shop Now

Our Products

Quantum-Core Laptop

A sleek, powerful laptop with a quantum processor for lightning-fast computing.

$2499.99 Add

Chrono-Shift Smartwatch

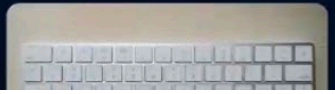
A stylish smartwatch that not only tells time but also monitors your health with futuristic sensors.

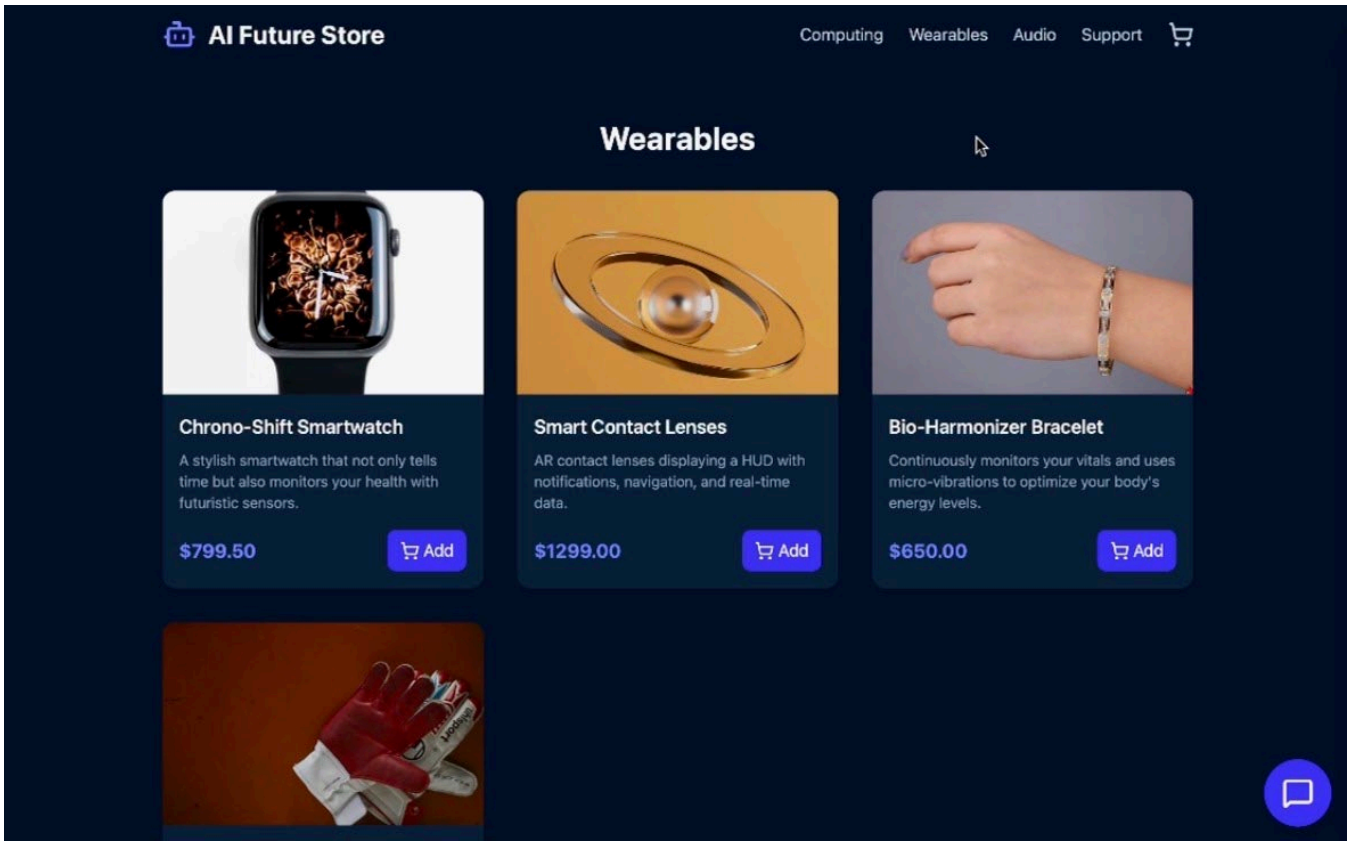
$799.50 Add

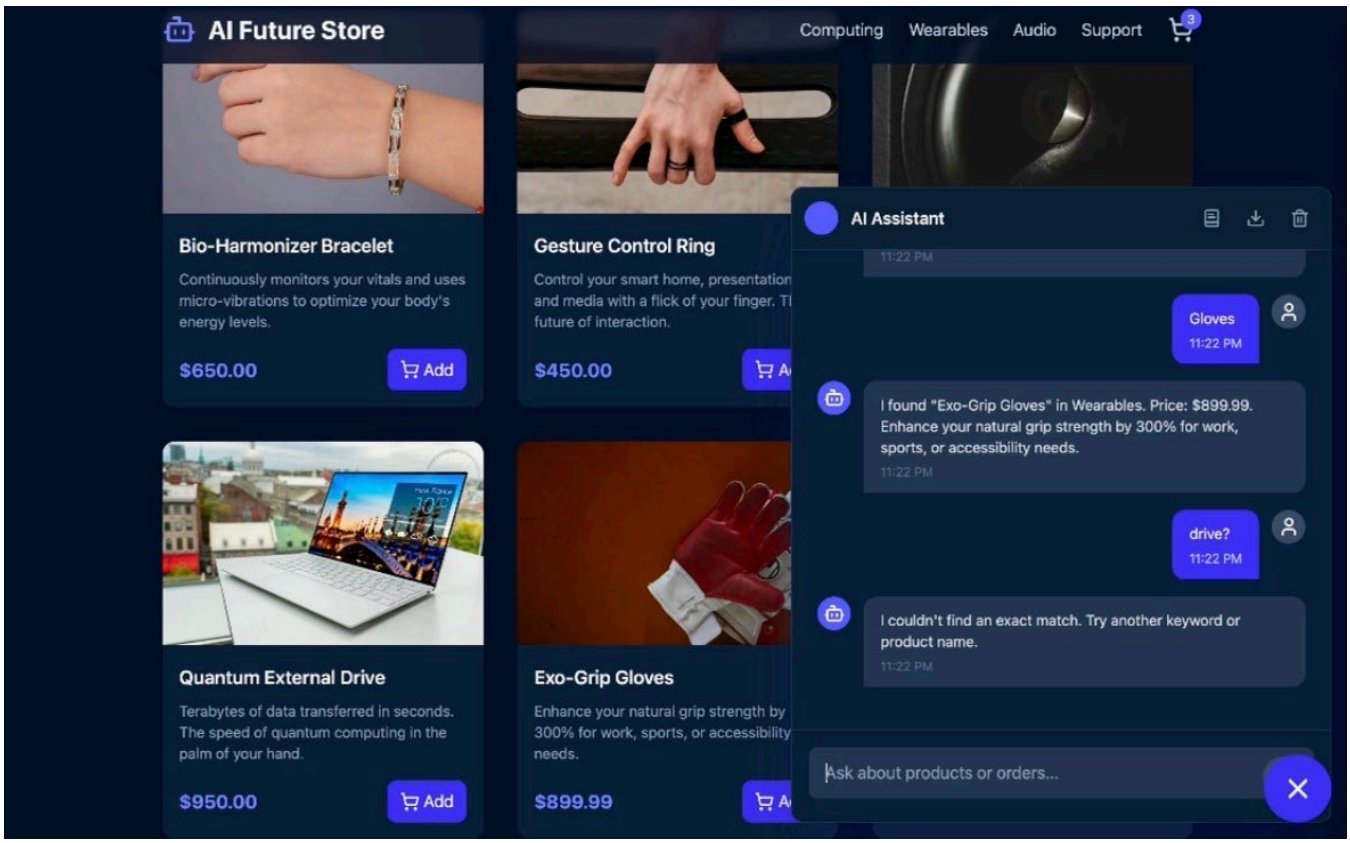
Neuro-Link Headphones

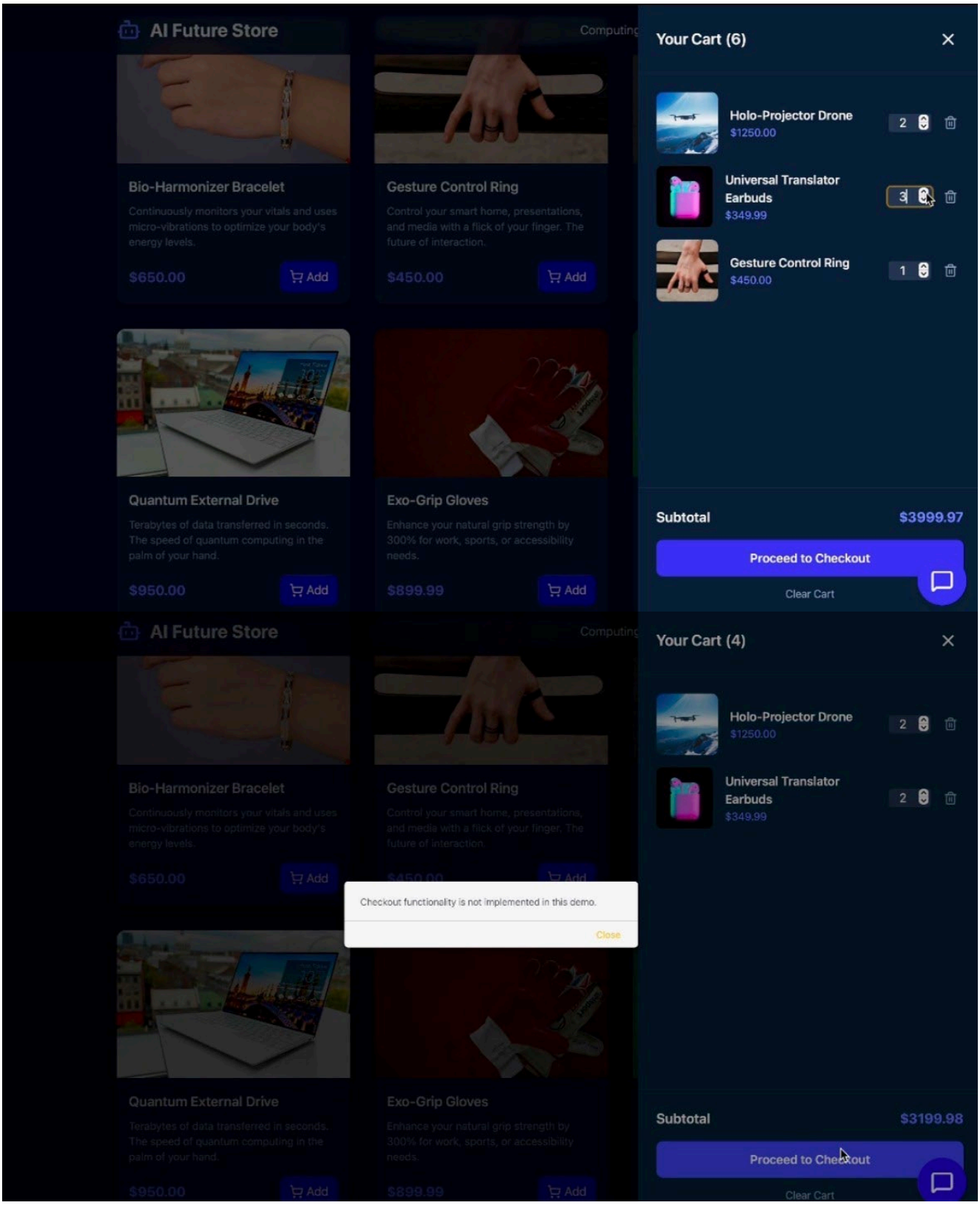
Immersive audio experience with brainwave synchronization for enhanced focus and relaxation.

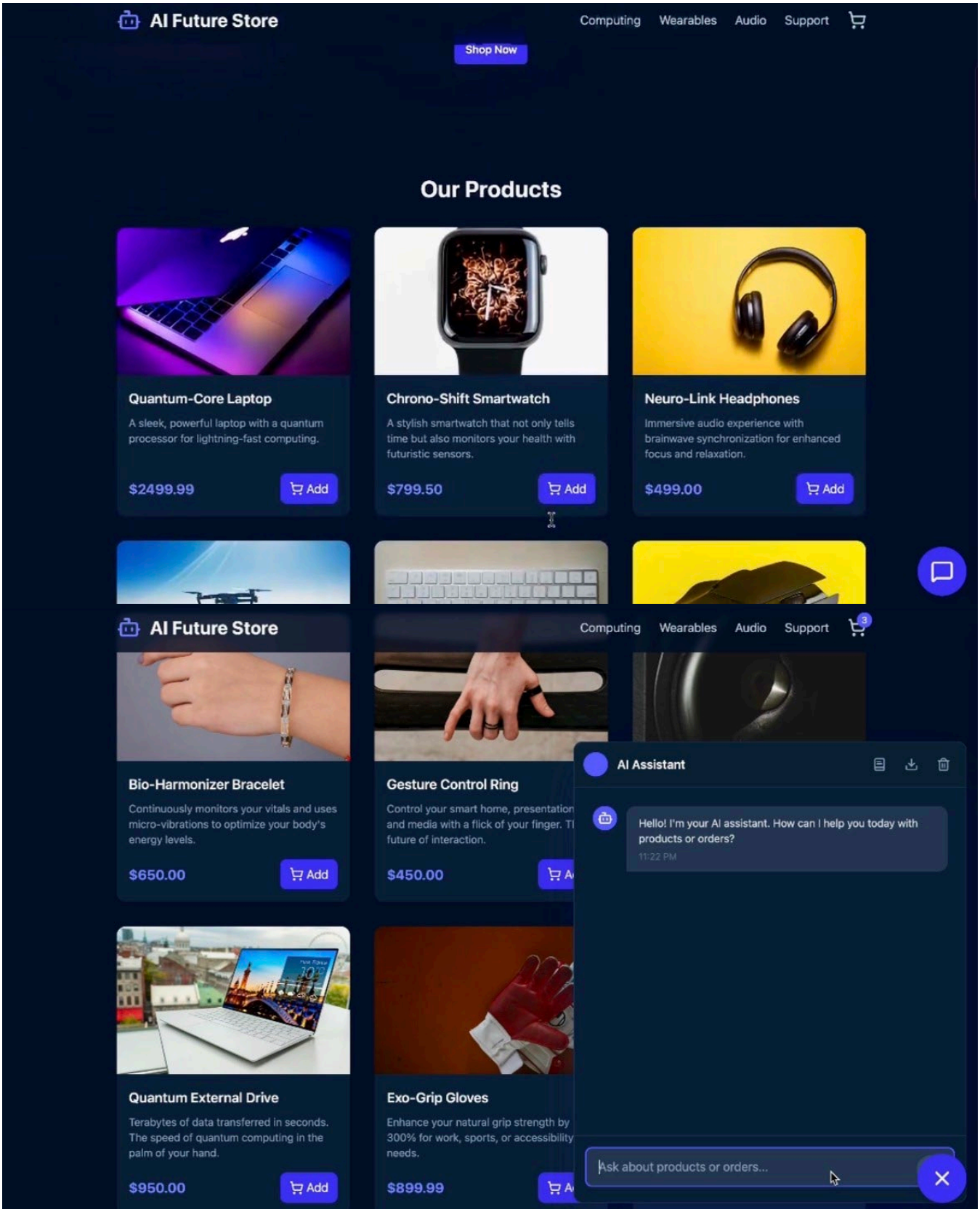
$499.00 Add

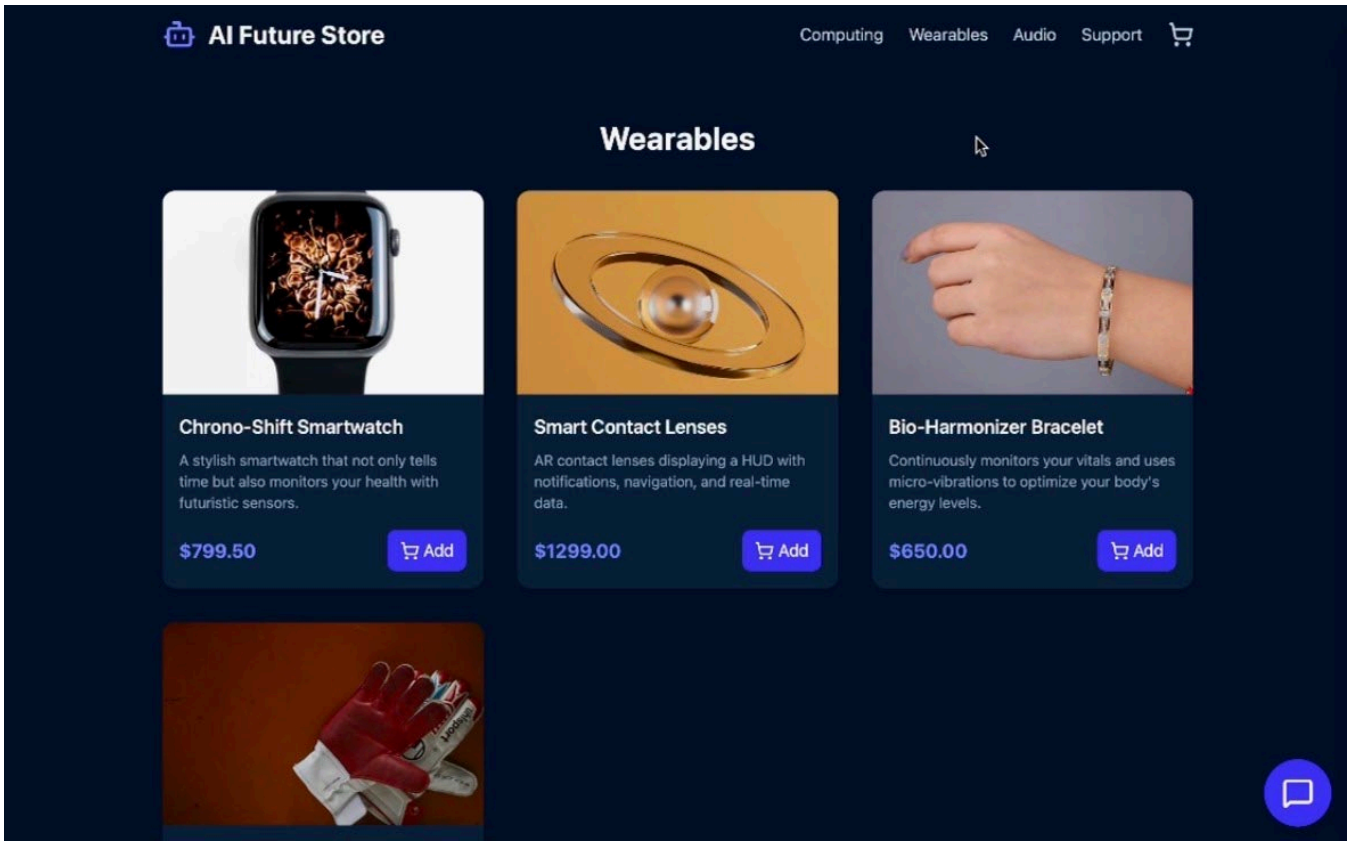



AI Future Store
Computing
Wearables
Audio
Support
Wearables
Chrono-Shift Smartwatch
A stylish smartwatch that not only tells time but also monitors your health with futuristic sensors.
$799.50
Add
Smart Contact Lenses
AR contact lenses displaying a HUD with notifications, navigation, and real-time data.
$1299.00
Add
Bio-Harmonizer Bracelet
Continuously monitors your vitals and uses micro-vibrations to optimize your body's energy levels.
$650.00
Add


AI Future Store
Computing
Wearables
Audio
Support
Bio-Harmonizer Bracelet
Continuously monitors your vitals and uses micro-vibrations to optimize your body's energy levels.
$650.00
Add
Gesture Control Ring
Control your smart home, presentation
and media with a flick of your finger. T
future of interaction.
$450.00
Quantum External Drive
Terabytes of data transferred in seconds. The speed of quantum computing in the palm of your hand.
$950.00
Add
Exo-Grip Gloves
Enhance your natural grip strength by 300% for work, sports, or accessibility needs.
$899.99
AI Assistant
11:22 PM
Gloves
11:22 PM
I found "Exo-Grip Gloves" in Wearables. Price: $899.99. Enhance your natural grip strength by 300% for work, sports, or accessibility needs.
11:22 PM
drive?
11:22 PM
I couldn't find an exact match. Try another keyword or product name.
11:22 PM
Ask about products or orders...


AI Future Store
Computing
Bio-Harmonizer Bracelet
Continuously monitors your vitals and uses micro-vibrations to optimize your body's energy levels.
$650.00
Add
Gesture Control Ring
Control your smart home, presentations, and media with a flick of your finger. The future of interaction.
$450.00
Add
Quantum External Drive
Terabytes of data transferred in seconds. The speed of quantum computing in the palm of your hand.
$950.00
Add
Exo-Grip Gloves
Enhance your natural grip strength by 300% for work, sports, or accessibility needs.
$899.99
Add
Your Cart (6)
Holo-Projector Drone
$1250.00
2
Universal Translator Earbuds
$349.99
3
Gesture Control Ring
$450.00
1
Subtotal
$3999.97
Proceed to Checkout
Clear Cart
AI Future Store
Computing
Bio-Harmonizer Bracelet
Continuously monitors your vitals and uses micro-vibrations to optimize your body's energy levels.
$650.00
Add
Gesture Control Ring
Control your smart home, presentations, and media with a flick of your finger. The future of interaction.
Add
Checkout functionality is not implemented in this demo.
Close
Quantum External Drive
Terabytes of data transferred in seconds. The speed of quantum computing in the palm of your hand.
$950.00
Add
Exo-Grip Gloves
Enhance your natural grip strength by 300% for work, sports, or accessibility needs.
$899.99
Add
Your Cart (4)
Holo-Projector Drone
$1250.00
2
Universal Translator Earbuds
$349.99
2
Subtotal
$3199.98
Proceed to Checkout
Clear Cart


AI Future Store
Computing
Wearables
Audio
Support
Shop Now
Our Products
Quantum-Core Laptop
A sleek, powerful laptop with a quantum processor for lightning-fast computing.
$2499.99
Add
Chrono-Shift Smartwatch
A stylish smartwatch that not only tells time but also monitors your health with futuristic sensors.
$799.50
Add
Neuro-Link Headphones
Immersive audio experience with brainwave synchronization for enhanced focus and relaxation.
$499.00
Add
AI Future Store
Computing
Wearables
Audio
Support
Bio-Harmonizer Bracelet
Continuously monitors your vitals and uses micro-vibrations to optimize your body's energy levels.
$650.00
Add
Gesture Control Ring
Control your smart home, presentation and media with a flick of your finger. T future of interaction.
$450.00
Quantum External Drive
Terabytes of data transferred in seconds. The speed of quantum computing in the palm of your hand.
$950.00
Add
Exo-Grip Gloves
Enhance your natural grip strength by 300% for work, sports, or accessibility needs.
$899.99
AI Assistant
Hello! I'm your AI assistant. How can I help you today with products or orders?
11:22 PM
Ask about products or orders...


AI Future Store
Computing
Wearables
Audio
Support
Wearables
Chrono-Shift Smartwatch
A stylish smartwatch that not only tells time but also monitors your health with futuristic sensors.
$799.50
Add
Smart Contact Lenses
AR contact lenses displaying a HUD with notifications, navigation, and real-time data.
$1299.00
Add
Bio-Harmonizer Bracelet
Continuously monitors your vitals and uses micro-vibrations to optimize your body's energy levels.
$650.00
Add

MAIN MODULES / PAGES

1. Landing / Home Page

Displays featured and trending products.

Includes search functionality with category and price filters.

Highlights offers, discounts, and new arrivals.

---

2. Product Details Page

Displays full product information (name, description, price, stock, ratings).

Shows product images with zoom/preview options.

Includes “Add to Cart” and “Buy Now” buttons.

Displays customer reviews and related products.

---

3. Cart and Checkout Page

Shows list of products added by the user.

Allows users to update quantity or remove items.

Displays total price and applicable discounts.

Provides checkout process with payment and address confirmation.

---

4. User Dashboard

Displays user profile and order history.

Allows users to manage addresses and payment methods.

Provides order tracking and invoice download options.

---

5. Admin Dashboard

Allows admin to add, edit, and delete products.

Manages inventory and categories.

Monitors sales and user activity through analytics.

---

6. Authentication Pages

Includes user registration and login with form validation.

Supports password encryption and OTP/email verification.

OUTPUT SCREENS

User registration and OTP verification screen.

Successful login confirmation.

Product added to cart confirmation.

Order placed successfully message.

Admin product management dashboard.

User order history and profile management screen.

LIMITATIONS & FUTURE SCOPE

Current Limitations

No real-time chat support between users and admin.

Limited analytics for admin (sales insights, user tracking).

Basic UI without dynamic personalization.

Future Enhancements

Migration to **React/Next.js** for improved UI and faster rendering.

Integration of **AI-powered product recommendations** and smart search.

Implementation of **real-time order tracking and notifications**.

Support for **multiple payment gateways and wallet systems**.

Development of a **mobile application version** for Android and iOS.

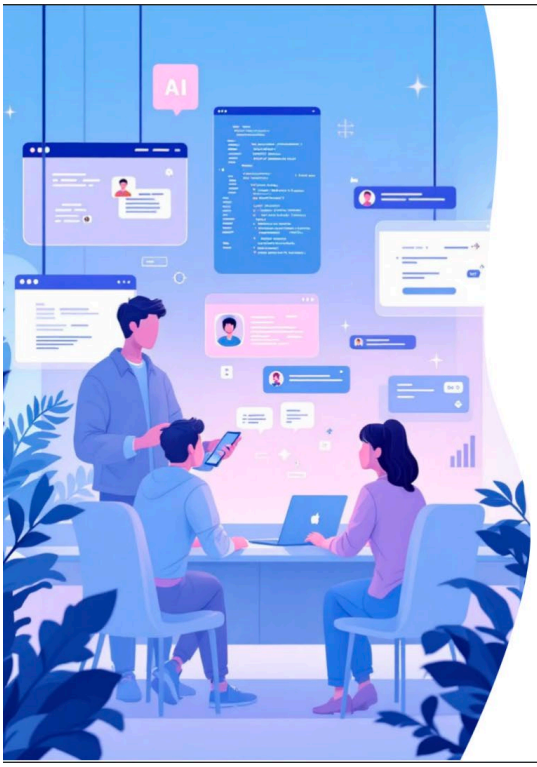
GITHUB URL

Click here to visit the Git Repo

DEPLOYED LINK

Click here to see the deployed version

Pre

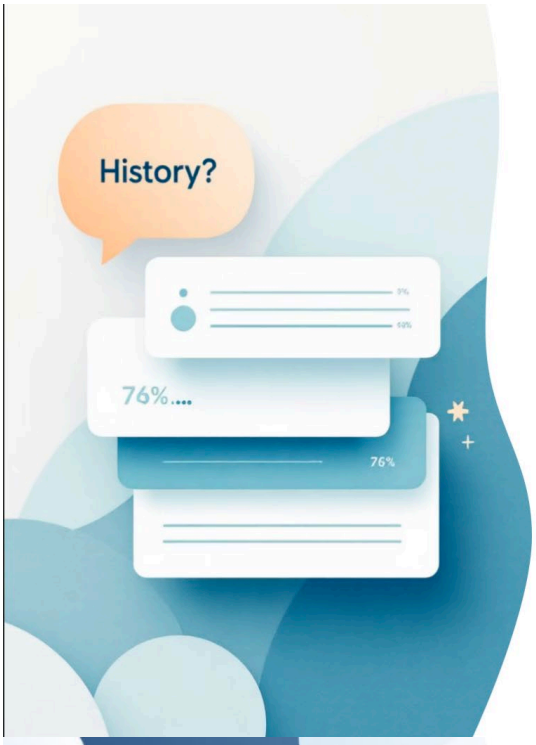


Building Production-Grade AI Chat

MERN Stack Implementation with Provider Abstraction

This presentation guides full-stack developers and engineering leads through architecting, building, and deploying a production-grade AI chat feature in MERN applications. We'll cover provider-agnostic backends, streaming architectures, security practices, and observable systems.

sentation Slides



Context Management & Memory Strategy

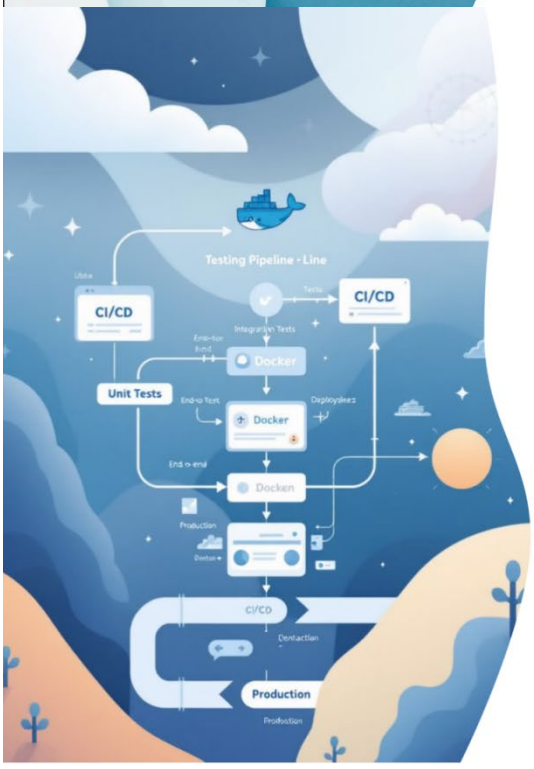
Maintain per-session context while controlling token growth. Store messages in MongoDB with timestamps and token counts. When context exceeds thresholds, trigger automatic summarization that condenses older messages into brief summaries, preserving key details while freeing tokens for new interactions.

Session Structure

- userId, sessionId, createdAt
- systemPrompt (configurable)
- messages array with role, content, tokens, timestamp
- metadata: provider, model, temperature

Guardrails

- Max context window enforcement
- TTL index for stale session cleanup
- User controls: clear history, reset system prompt
- Export/download conversations as JSON or PDF



Testing, Deployment & Operations

Test the system comprehensively: use mock providers for deterministic unit tests, write contract tests on payload shapes to catch breaking changes, run E2E tests validating streaming and context retention. Deploy via Docker Compose locally, containerize services for cloud, and use environment-driven configs and feature flags for safe rollouts.

1. **Unit & Mocks**

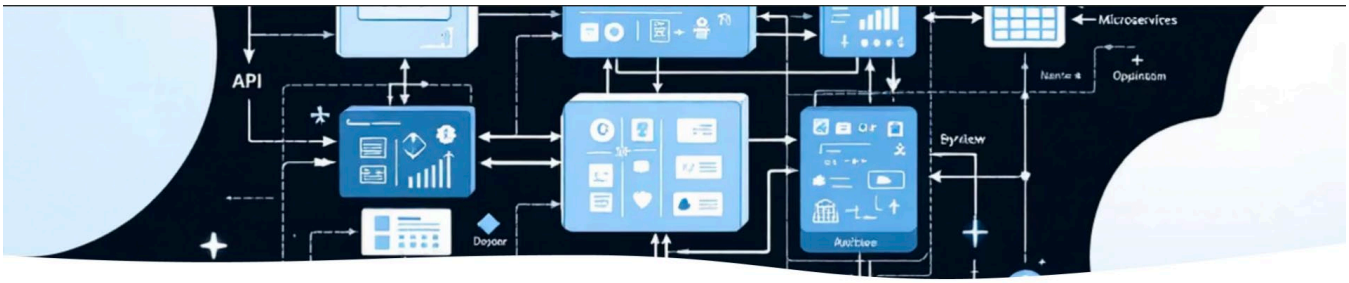
   Mock OpenAI and Dialogflow adapters, test guardrail logic, validate token counting independently.

2. **Integration Tests**

   Spin up MongoDB, test adapter implementations, verify persistence and streaming with real providers in staging.

3. **E2E & Ops**

   Puppeteer scripts for full chat flow, Docker Compose for local dev, feature flags for model rollout, seed scripts for demo sessions.



The Challenge: Integrating AI Chat at Scale

Building a chat feature means orchestrating multiple concerns simultaneously: provider flexibility, real-time streaming, persistent context, safety controls, and cost management. A naive monolithic approach creates technical debt. We need abstraction layers, pluggable services, and observability from day one.

🎯 Provider Agnostic

Swap OpenAI or Dialogflow without refactoring code

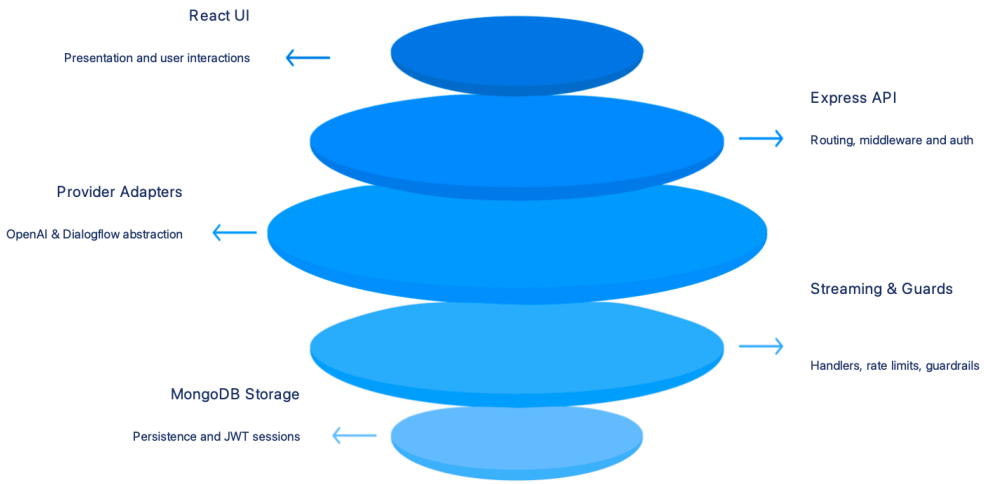
⚡ Streaming First

SSE or WebSocket for real-time token delivery

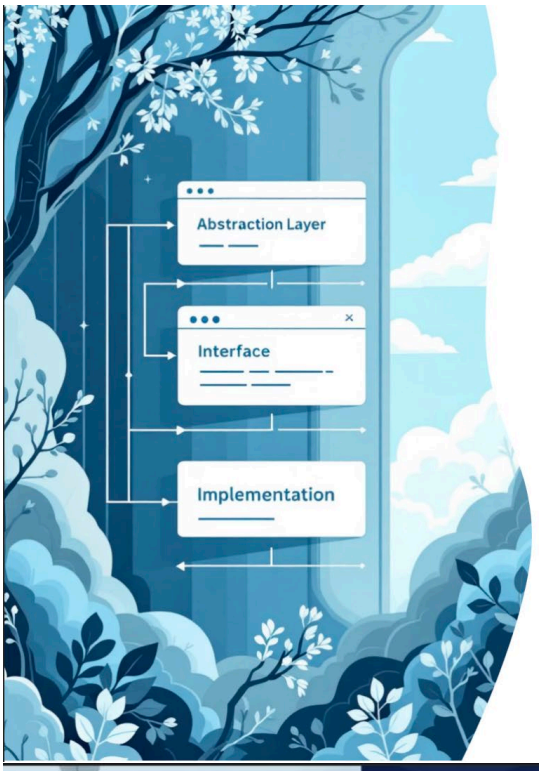
🔒 Secure & Auditable

JWT auth, rate limits, guardrails, and structured logs

Architecture Overview



The architecture separates concerns into distinct layers: React UI handles presentation and user interactions, Express middleware manages auth and rate limiting, pluggable provider adapters handle API specifics, and MongoDB persists all conversation state. This modular design allows teams to evolve each independently.



Provider Abstraction Layer

Create a unified provider interface that both OpenAI and Dialogflow implement. Each adapter normalizes requests and responses into a canonical format, handling timeout logic, retry strategies, and error transformation. Configuration drives which provider activates per environment.

1

Define Interface

sendMessage(context, userInput, config) returns standardized response shape with tokens, content, and metadata.

2

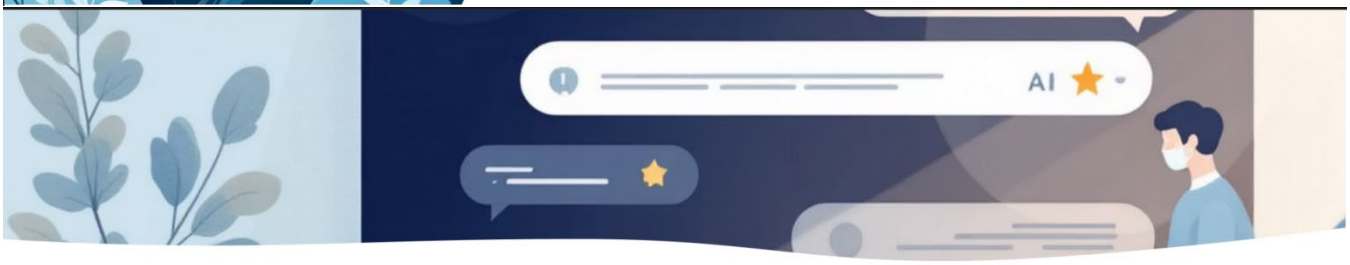
Implement Adapters

OpenAI adapter uses fetch to call /chat/completions; Dialogflow adapter integrates session management and intent detection.

3

Factory Registration

Instantiate the correct adapter based on NODE_ENV or runtime feature flags, with dependency injection for testability.



React Chat UI & User Experience

Build a responsive React component that displays streamed messages incrementally with typing indicators, enables message grouping by timestamp, and provides controls for provider/model selection. Implement error boundaries with retry logic, keyboard navigation, and accessibility features. Show token usage and remaining budget context to users.

Message Feed

Render user and AI messages with avatar, timestamp, and optional copy/delete actions. Scroll-to-latest on new messages.

Input Area

Text field with auto-focus, multi-line support, send/cancel buttons, and visual feedback during streaming.

Controls Panel

Clear history, summarize session, switch provider/model, adjust temperature, view budget status and token count.

Streaming Responses & Real-Time Delivery

Implement Server-Sent Events (SSE) or WebSocket endpoints for /chat that stream partial tokens as they arrive. This improves perceived performance and allows cancellation mid-response. Handle backpressure by pausing reads when buffers fill, and emit heartbeats to detect stale connections.

Client connects

POST /api/chat/stream with message and session ID.

Backend fetches context

Query MongoDB for session history and system prompts.

Stream tokens

Call provider adapter; emit each token via SSE or WebSocket frame.

Persist on complete

Insert full message and metadata into messages collection; update token counts.

Security, Safety & Observability

Secure the system with JWT authentication, rate limiting per user and workspace, input validation to block prompt injection, and basic content policies. Maintain structured logs with correlation IDs, metrics for latency and token usage, and simple per-user budget caps to manage costs.

Auth & Secrets

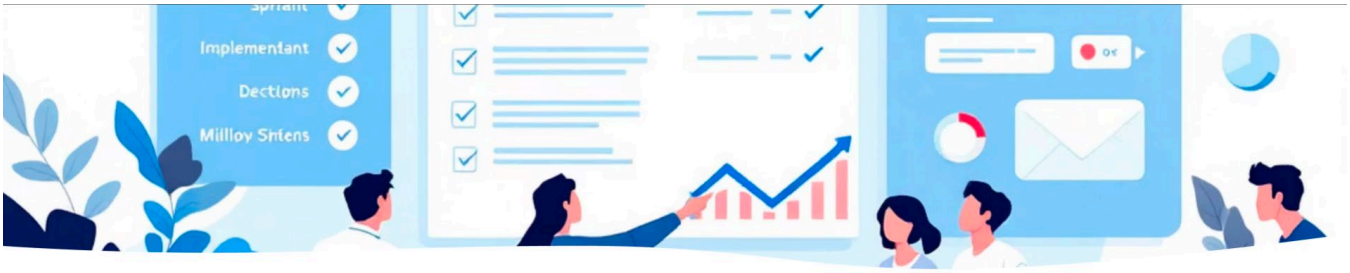
httpOnly cookies + JWT, env-based API keys, no secrets in logs or responses.

Guardrails

Input validation, regex filters for injection patterns, content policy checks before sending to provider.

Observability

Correlation IDs on every request, latency histograms, error rates by provider, token usage trends, budget alerts.



Key Takeaways & Next Steps

🏗️ Build for Scale

Abstract providers early, design streaming from day one, persist everything, and instrument metrics before shipping.

🔐 Security First

Protect secrets, validate inputs, rate limit aggressively, and maintain audit logs for compliance and debugging.

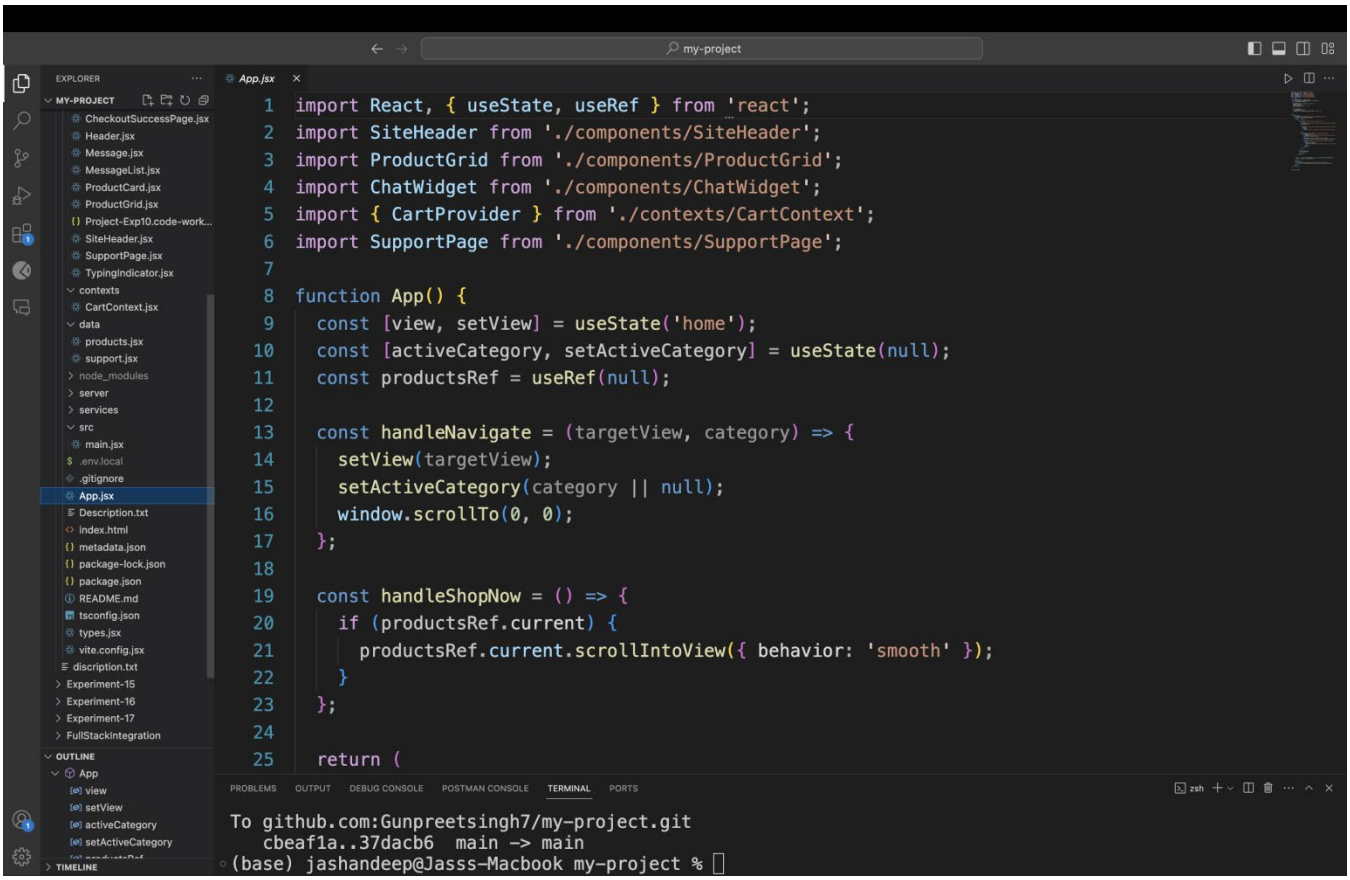
📈 Iterate Safely

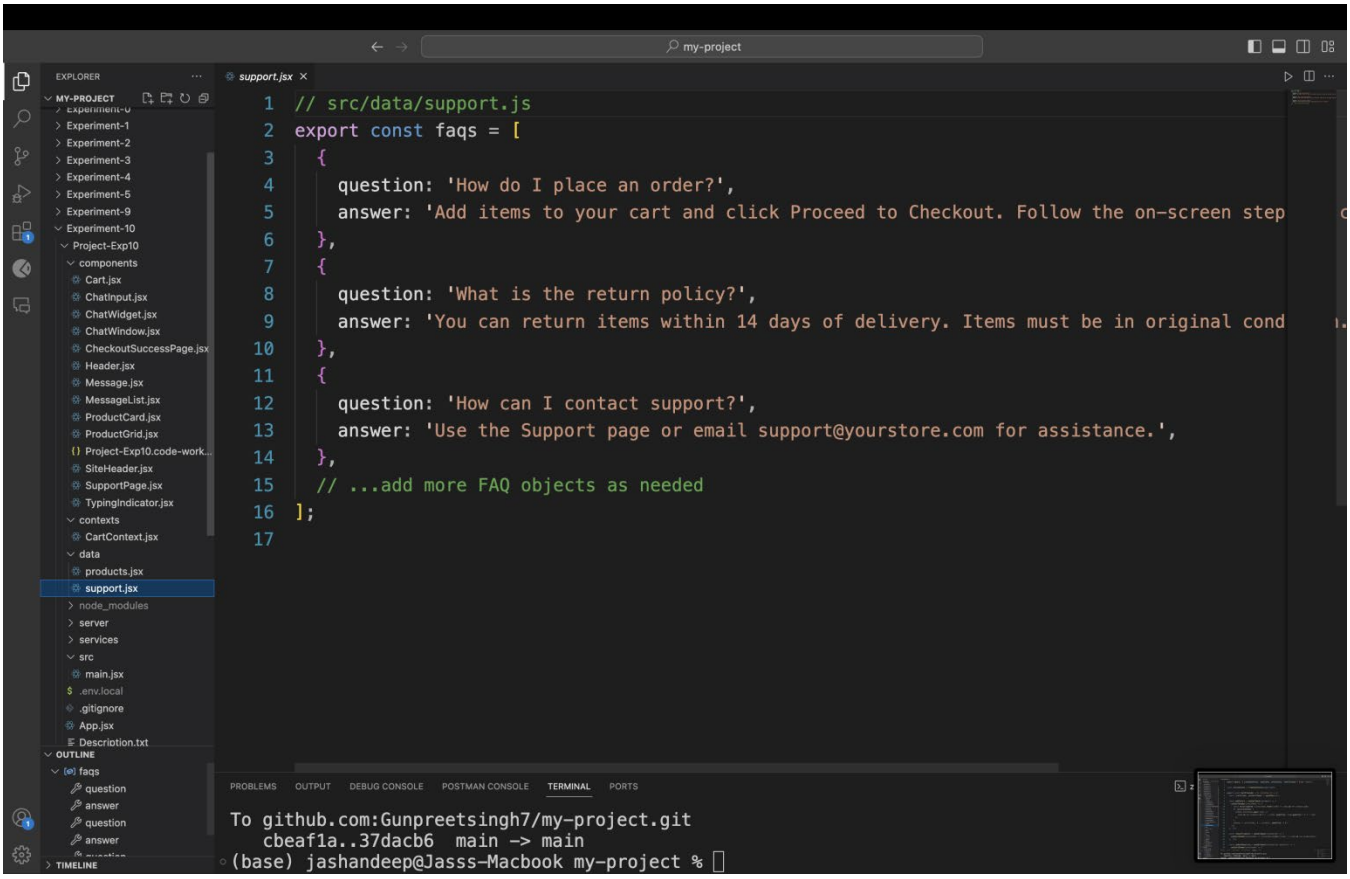
Test rigorously, use feature flags for rollouts, monitor observability signals, and prepare for cost and error spikes.

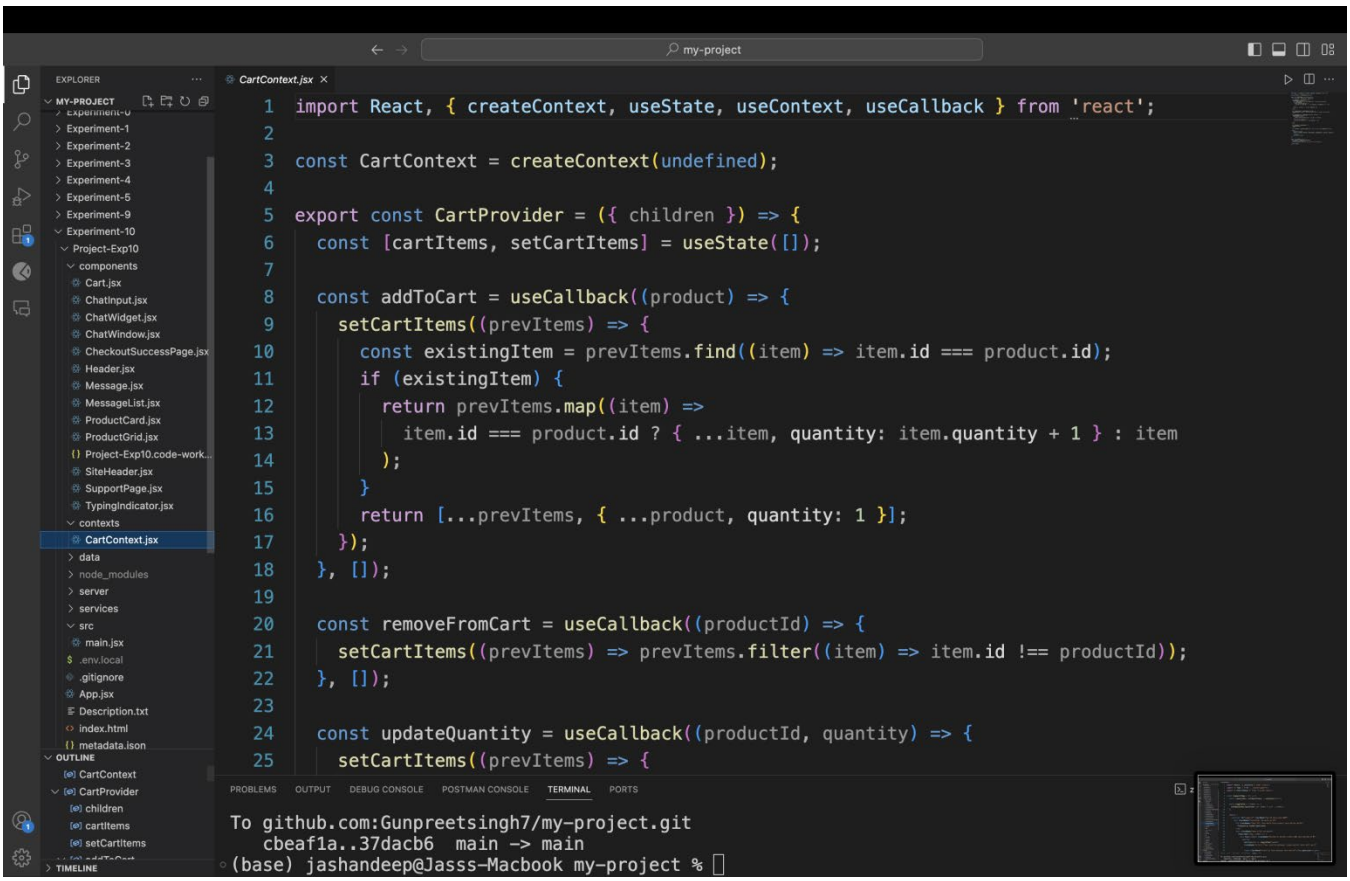
Start here: Set up provider adapters and core streaming infrastructure; build React UI component in parallel; integrate guardrails and persistence; add observability; deploy and measure before optimizing.

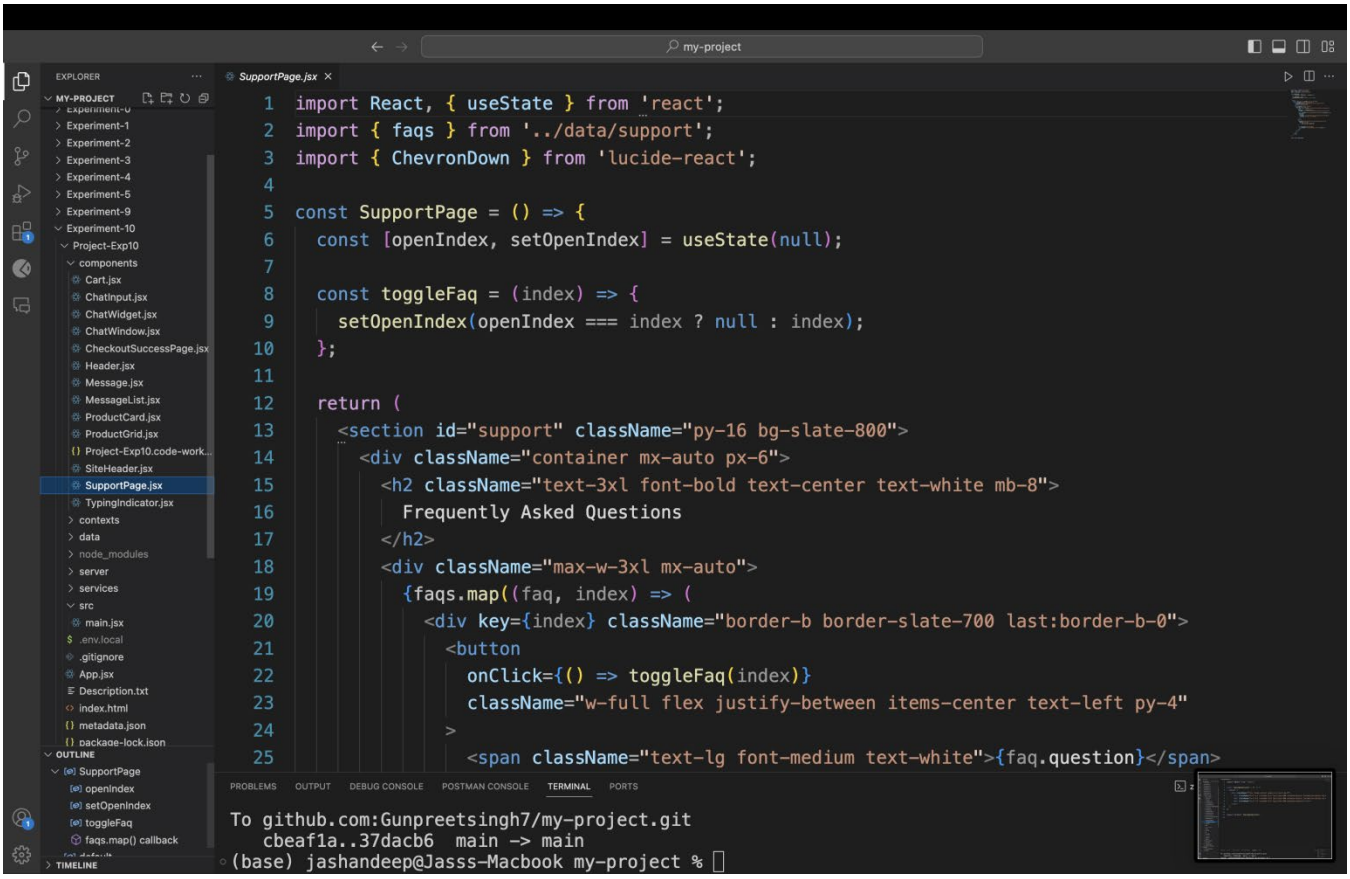
PROJECT CODE

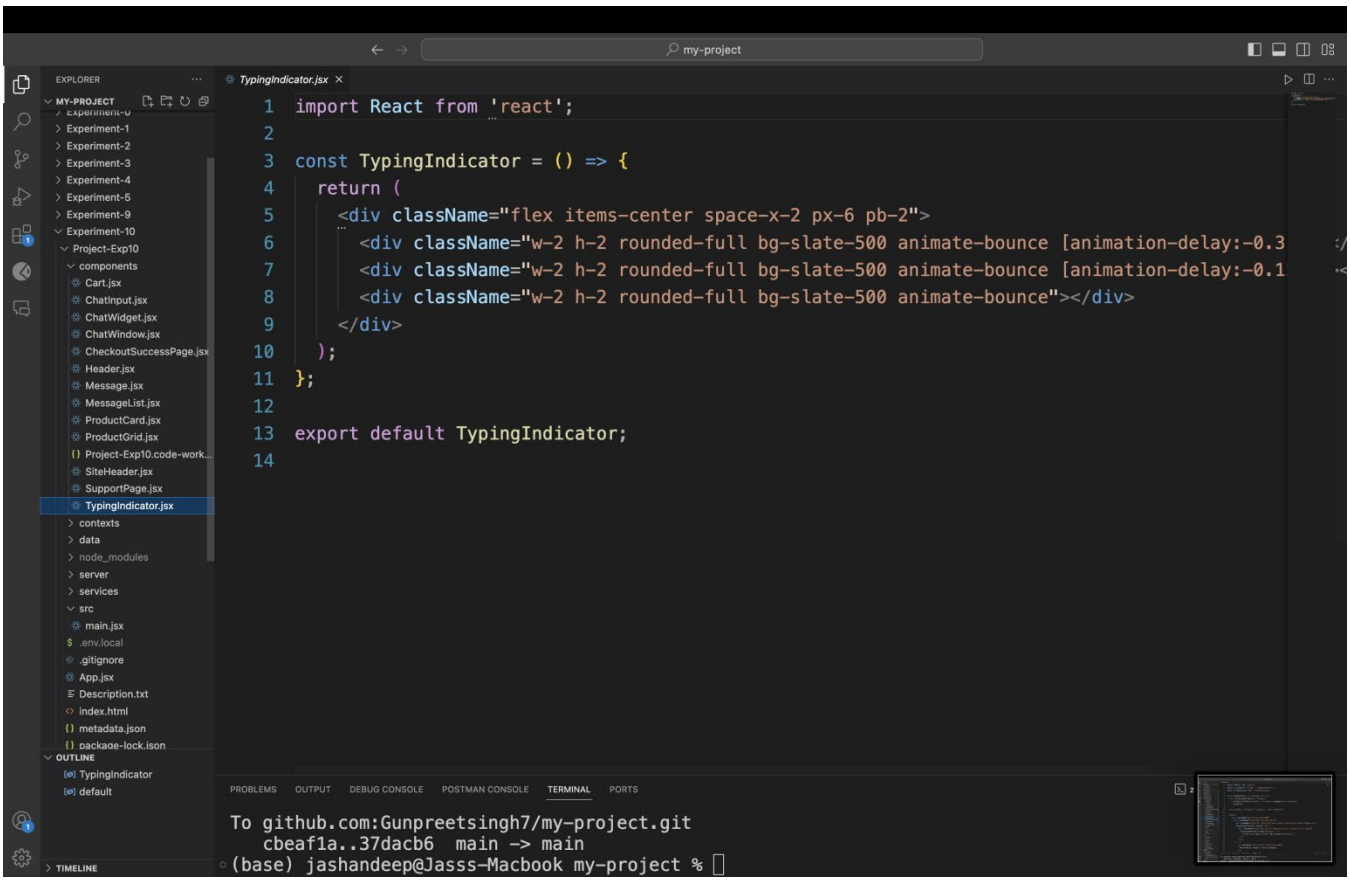
Provided structured code layout:

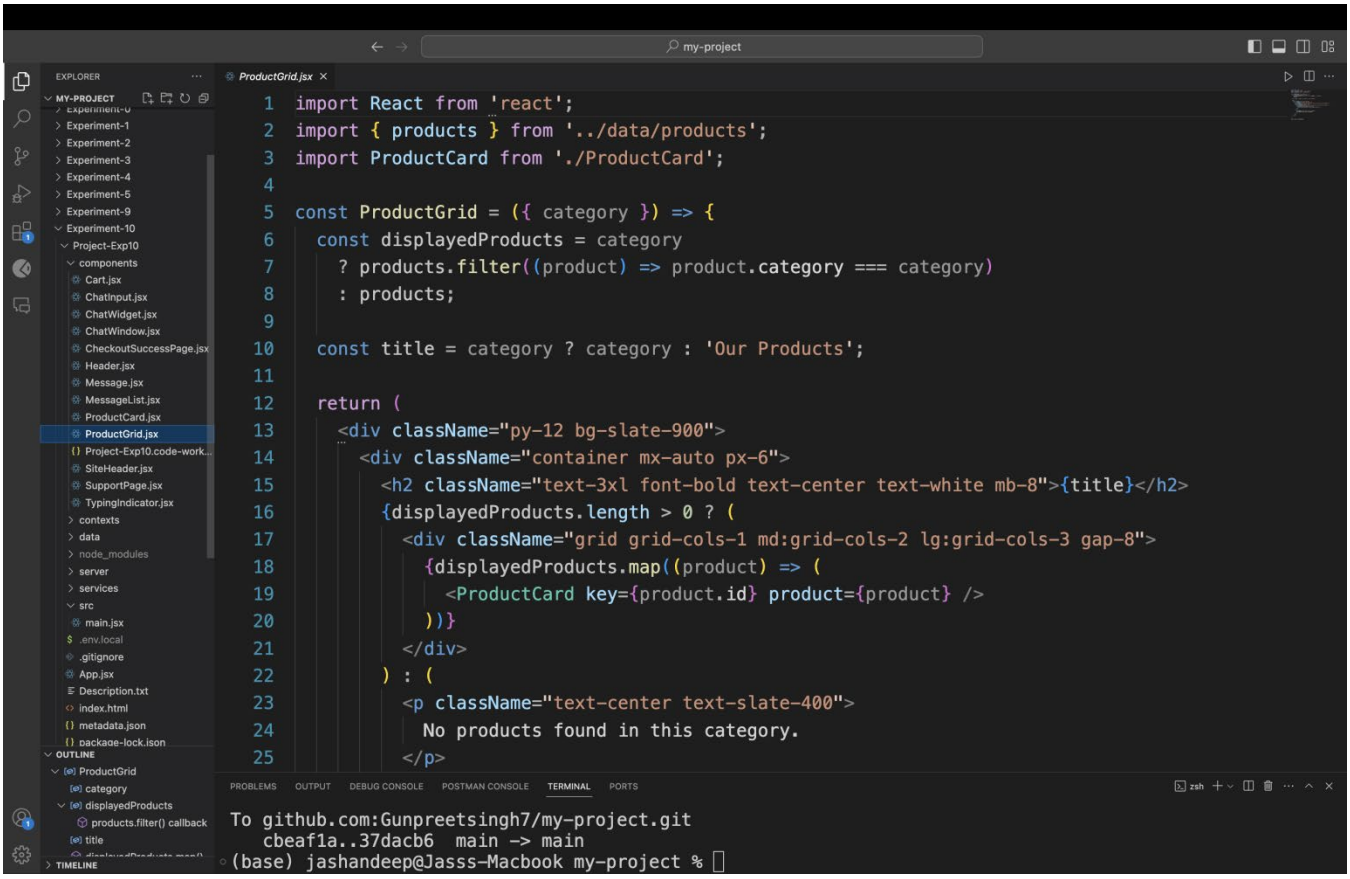


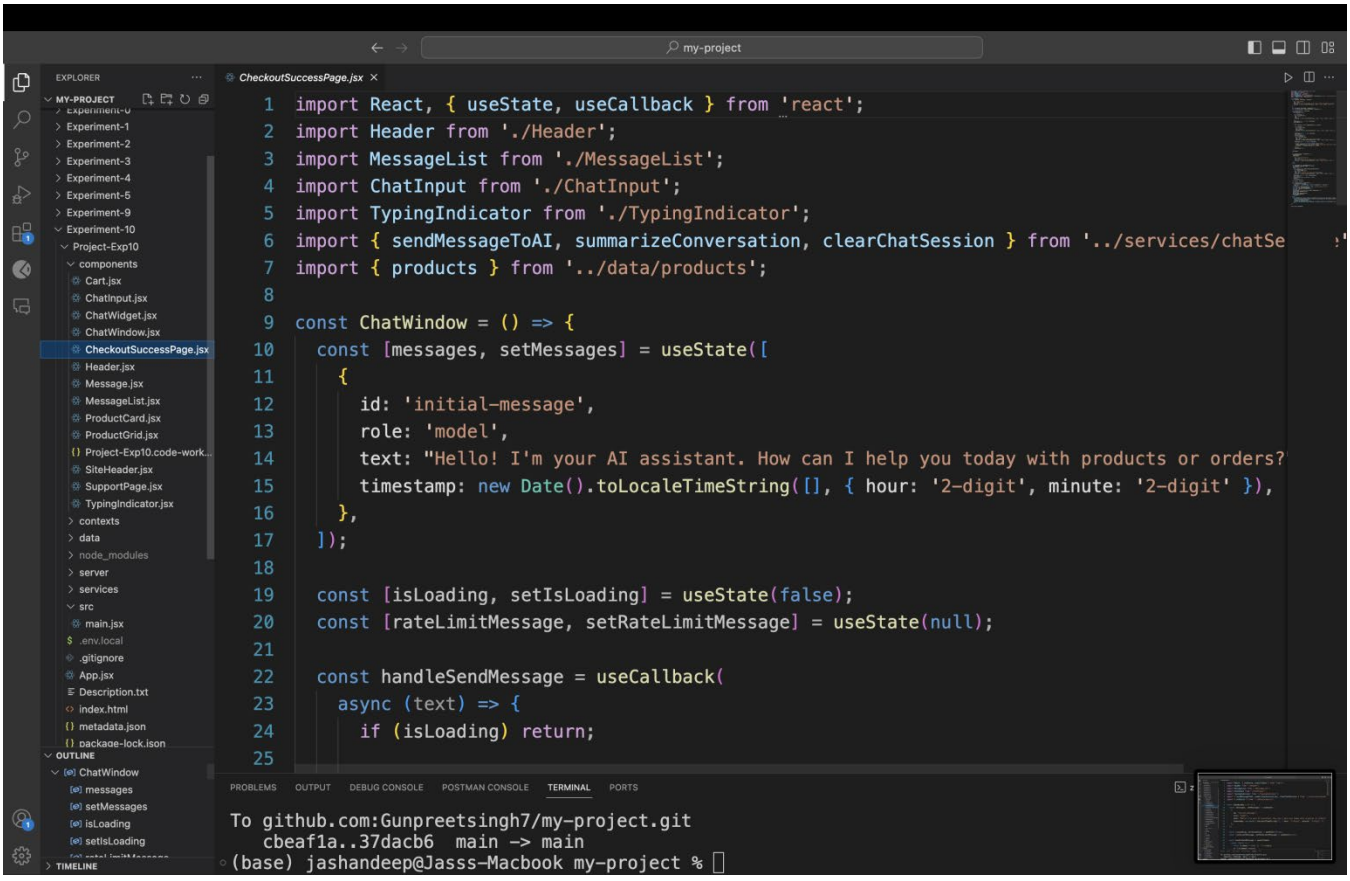

my-project
support.jsx
// src/data/support.js
export const faqs = [
{
question: 'How do I place an order?',
answer: 'Add items to your cart and click Proceed to Checkout. Follow the on-screen step
},
{
question: 'What is the return policy?',
answer: 'You can return items within 14 days of delivery. Items must be in original cond
},
{
question: 'How can I contact support?',
answer: 'Use the Support page or email support@yourstore.com for assistance.',
},
// ...add more FAQ objects as needed
];
PROBLEMS OUTPUT DEBUG CONSOLE POSTMAN CONSOLE TERMINAL PORTS
To github.com:Gunpreetsingh7/my-project.git
cbeaf1a..37dacb6 main -> main
(base) jashandeep@Jasss-Macbook my-project %

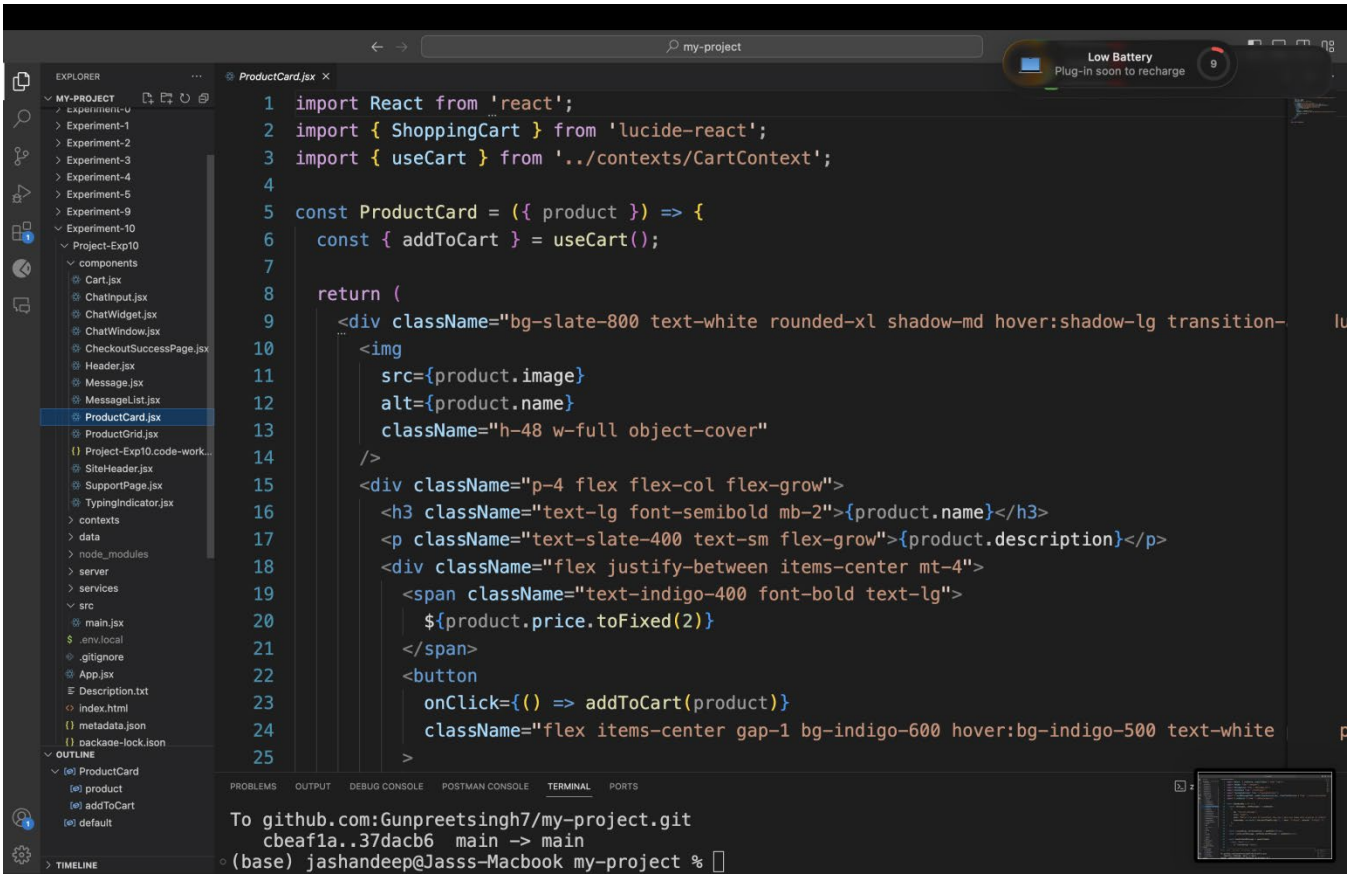

my-project
CartContext.jsx
import React, { createContext, useState, useContext, useCallback } from 'react';
const CartContext = createContext(undefined);
export const CartProvider = ({ children }) => {
const [cartItems, setCartItems] = useState([]);
const addToCart = useCallback((product) => {
setCartItems((prevItems) => {
const existingItem = prevItems.find((item) => item.id === product.id);
if (existingItem) {
return prevItems.map((item) =>
item.id === product.id ? { ...item, quantity: item.quantity + 1 } : item
);
}
return [...prevItems, { ...product, quantity: 1 }];
});
}, []);
const removeFromCart = useCallback((productId) => {
setCartItems((prevItems) => prevItems.filter((item) => item.id !== productId));
}, []);
const updateQuantity = useCallback((productId, quantity) => {
setCartItems((prevItems) => {
PROBLEMS OUTPUT DEBUG CONSOLE POSTMAN CONSOLE TERMINAL PORTS
To github.com:Gunpreetsingh7/my-project.git
cbeaf1a..37dacb6 main -> main
(base) jashandeep@Jasss-Macbook my-project %


my-project
EXPLORER
MY-PROJECT
Experiment-1
Experiment-2
Experiment-3
Experiment-4
Experiment-5
Experiment-9
Experiment-10
Project-Exp10
components
Cart.jsx
ChatInput.jsx
ChatWidget.jsx
ChatWindow.jsx
CheckoutSuccessPage.jsx
Header.jsx
Message.jsx
MessageList.jsx
ProductCard.jsx
ProductGrid.jsx
Project-Exp10.code-work...
SiteHeader.jsx
SupportPage.jsx
TypingIndicator.jsx
contexts
data
node_modules
server
services
src
main.jsx
.env.local
.gitignore
App.jsx
Description.txt
index.html
metadata.json
package-lock.json
OUTLINE
SupportPage
openIndex
setOpenIndex
toggleFaq
faqs.map() callback
TIMELINE
SupportPage.jsx
1 import React, { useState } from 'react';
2 import { faqs } from '../data/support';
3 import { ChevronDown } from 'lucide-react';
4
5 const SupportPage = () => {
6 const [openIndex, setOpenIndex] = useState(null);
7
8 const toggleFaq = (index) => {
9 setOpenIndex(openIndex === index ? null : index);
10 };
11
12 return (
13 <section id="support" className="py-16 bg-slate-800">
14
15 <h2 className="text-3xl font-bold text-center text-white mb-8">
16 Frequently Asked Questions
17 </h2>
18
19 {faqs.map((faq, index) => (
20
21 <button
22 onClick={() => toggleFaq(index)}
23 className="w-full flex justify-between items-center text-left py-4"
24 >
25 {faq.question}
PROBLEMS
OUTPUT
DEBUG CONSOLE
POSTMAN CONSOLE
TERMINAL
PORTS
To github.com:Gunpreetsingh7/my-project.git
cbeaf1a..37dacb6 main -> main
(base) jashandeep@Jasss-Macbook my-project %


my-project
EXPLORER
MY-PROJECT
Experiment-1
Experiment-2
Experiment-3
Experiment-4
Experiment-5
Experiment-9
Experiment-10
Project-Exp10
components
Cart.jsx
ChatInput.jsx
ChatWidget.jsx
ChatWindow.jsx
CheckoutSuccessPage.jsx
Header.jsx
Message.jsx
MessageList.jsx
ProductCard.jsx
ProductGrid.jsx
Project-Exp10.code-work...
SiteHeader.jsx
SupportPage.jsx
TypingIndicator.jsx
contexts
data
node_modules
server
services
src
main.jsx
.env.local
.gitignore
App.jsx
Description.txt
index.html
metadata.json
package-lock.json
OUTLINE
TypingIndicator
default
TIMELINE
TypingIndicator.jsx
1 import React from 'react';
2
3 const TypingIndicator = () => {
4 return (
5
6 <div className="w-2 h-2 rounded-full bg-slate-500 animate-bounce [animation-delay:-0.3
7 <div className="w-2 h-2 rounded-full bg-slate-500 animate-bounce [animation-delay:-0.1
8
9
10);
11 };
12
13 export default TypingIndicator;
14
PROBLEMS
OUTPUT
DEBUG CONSOLE
POSTMAN CONSOLE
TERMINAL
PORTS
To github.com:Gunpreetsingh7/my-project.git
cbeaf1a..37dacb6 main -> main
(base) jashandeep@Jasss-Macbook my-project %


my-project
EXPLORER
MY-PROJECT
Experiment-1
Experiment-2
Experiment-3
Experiment-4
Experiment-5
Experiment-9
Experiment-10
Project-Exp10
components
Cart.jsx
ChatInput.jsx
ChatWidget.jsx
ChatWindow.jsx
CheckoutSuccessPage.jsx
Header.jsx
Message.jsx
MessageList.jsx
ProductCard.jsx
ProductGrid.jsx
Project-Exp10.code-work...
SiteHeader.jsx
SupportPage.jsx
TypingIndicator.jsx
contexts
data
node_modules
server
services
src
main.jsx
.env.local
.gitignore
App.jsx
Description.txt
index.html
metadata.json
package-lock.json
OUTLINE
ProductGrid
category
displayedProducts
products.filter() callback
title
TIMELINE
ProductGrid.jsx
1 import React from 'react';
2 import { products } from '../data/products';
3 import ProductCard from './ProductCard';
4
5 const ProductGrid = ({ category }) => {
6 const displayedProducts = category
7 ? products.filter((product) => product.category === category)
8 : products;
9
10 const title = category ? category : 'Our Products';
11
12 return (
13
14
15 <h2 className="text-3xl font-bold text-center text-white mb-8">{title}</h2>
16 {displayedProducts.length > 0 ? (
17
18 {displayedProducts.map((product) => (
19 <ProductCard key={product.id} product={product} />
20))}
21
22) : (
23
24 No products found in this category.
25
PROBLEMS
OUTPUT
DEBUG CONSOLE
POSTMAN CONSOLE
TERMINAL
PORTS
zsh
To github.com:Gunpreetsingh7/my-project.git
cbeaf1a..37dacb6 main -> main
(base) jashandeep@Jasss-Macbook my-project %

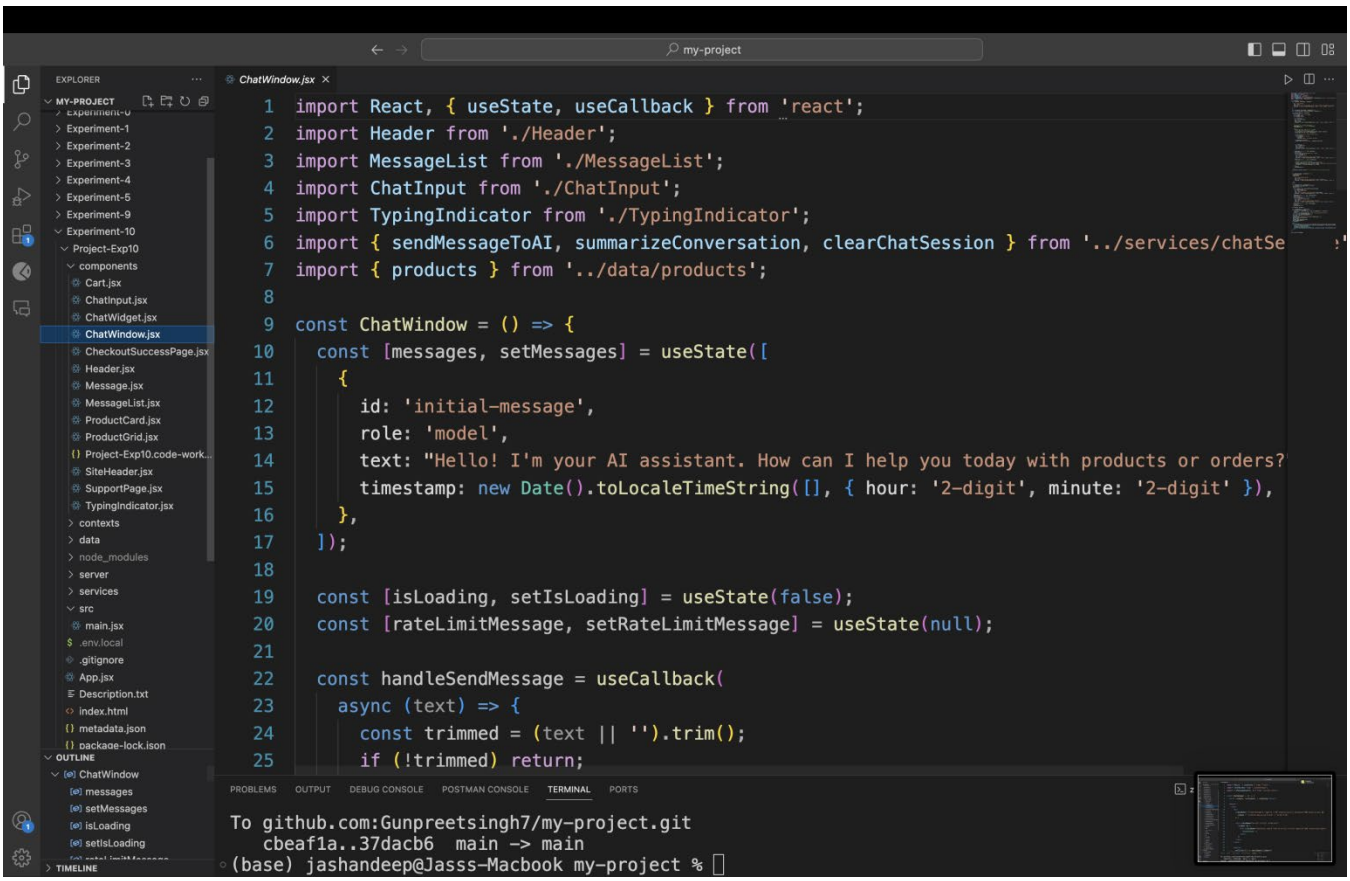

my-project
EXPLORER
MY-PROJECT
Experiment-1
Experiment-2
Experiment-3
Experiment-4
Experiment-5
Experiment-9
Experiment-10
Project-Exp10
components
Cart.jsx
ChatInput.jsx
ChatWidget.jsx
ChatWindow.jsx
CheckoutSuccessPage.jsx
Header.jsx
Message.jsx
MessageList.jsx
ProductCard.jsx
ProductGrid.jsx
Project-Exp10.code-work...
SiteHeader.jsx
SupportPage.jsx
TypingIndicator.jsx
contexts
data
node_modules
server
services
src
main.jsx
.env.local
.gitignore
App.jsx
Description.txt
index.html
metadata.json
package-lock.json
OUTLINE
ChatWindow
messages
setMessages
isLoading
setIsLoading
TIMELINE
CheckoutSuccessPage.jsx
1 import React, { useState, useCallback } from 'react';
2 import Header from './Header';
3 import MessageList from './MessageList';
4 import ChatInput from './ChatInput';
5 import TypingIndicator from './TypingIndicator';
6 import { sendMessageToAI, summarizeConversation, clearChatSession } from '../services/chatSe
7 import { products } from '../data/products';
8
9 const ChatWindow = () => {
10 const [messages, setMessages] = useState([
11 {
12 id: 'initial-message',
13 role: 'model',
14 text: "Hello! I'm your AI assistant. How can I help you today with products or orders?
15 timestamp: new Date().toLocaleTimeString([], { hour: '2-digit', minute: '2-digit' }),
16 },
17]);
18
19 const [isLoading, setIsLoading] = useState(false);
20 const [rateLimitMessage, setRateLimitMessage] = useState(null);
21
22 const handleSendMessage = useCallback(
23 async (text) => {
24 if (isLoading) return;
25
PROBLEMS
OUTPUT
DEBUG CONSOLE
POSTMAN CONSOLE
TERMINAL
PORTS
To github.com:Gunpreetsingh7/my-project.git
cbeaf1a..37dacb6 main -> main
(base) jashandeep@Jasss-Macbook my-project %

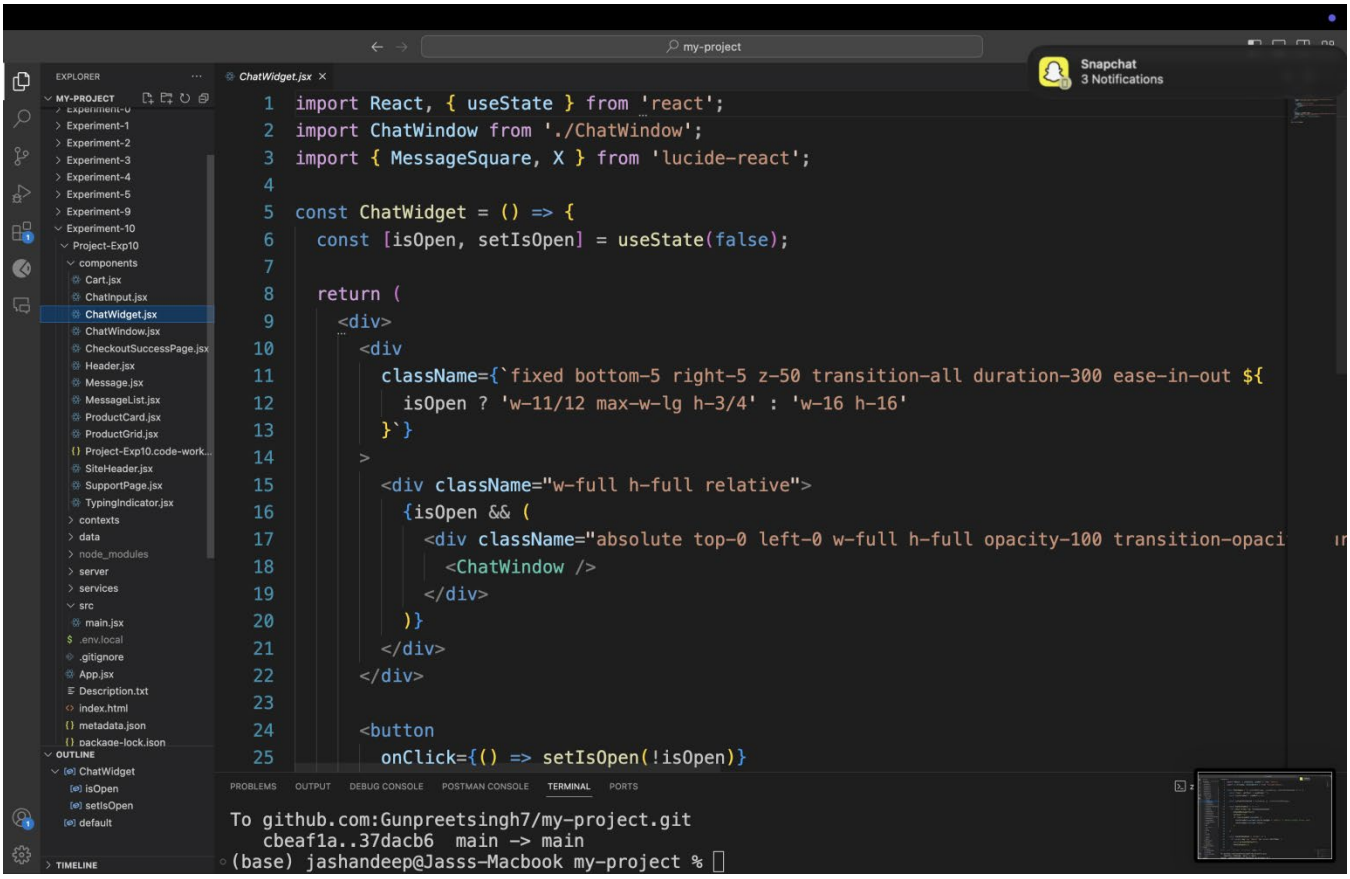

my-project
Low Battery
Plug-in soon to recharge
EXPLORER
MY-PROJECT
ProductCard.jsx
import React from 'react';
import { ShoppingCart } from 'lucide-react';
import { useCart } from '../contexts/CartContext';
const ProductCard = ({ product }) => {
const { addToCart } = useCart();
return (
<div className="bg-slate-800 text-white rounded-xl shadow-md hover:shadow-lg transition-
<img
src={product.image}
alt={product.name}
className="h-48 w-full object-cover"
/>

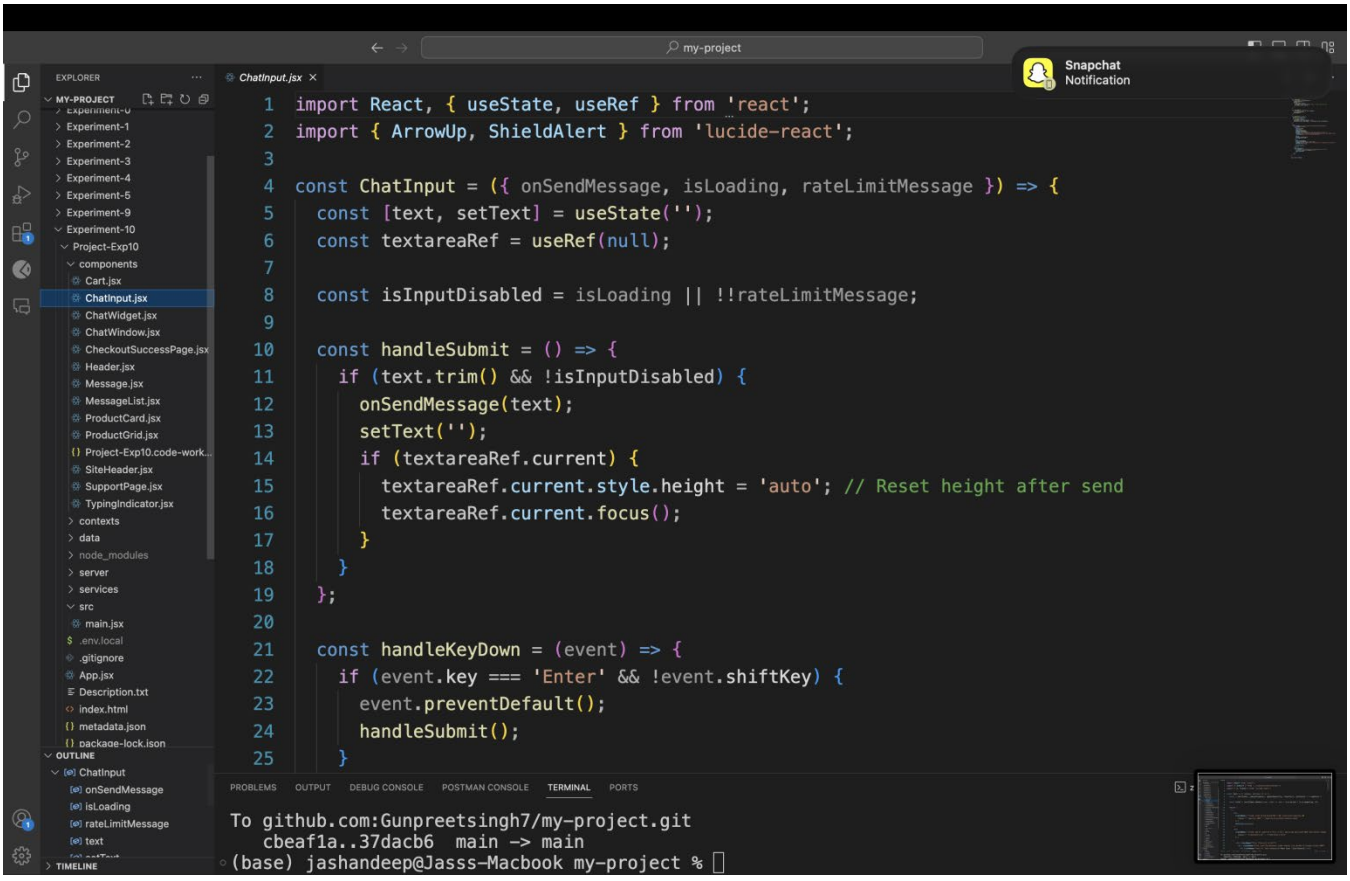
<h3 className="text-lg font-semibold mb-2">{product.name}</h3>
{product.description}

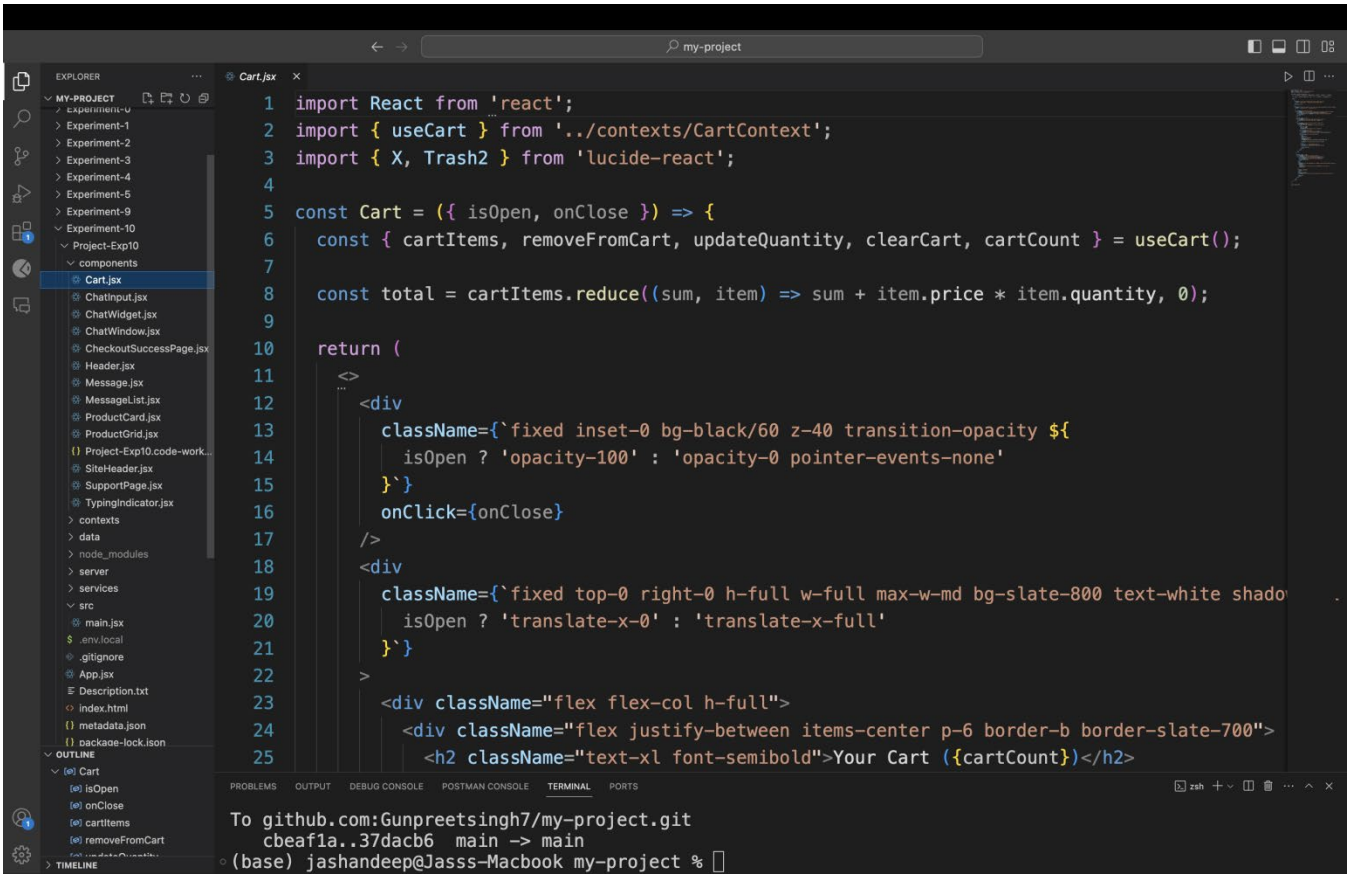
${product.price.toFixed(2)}

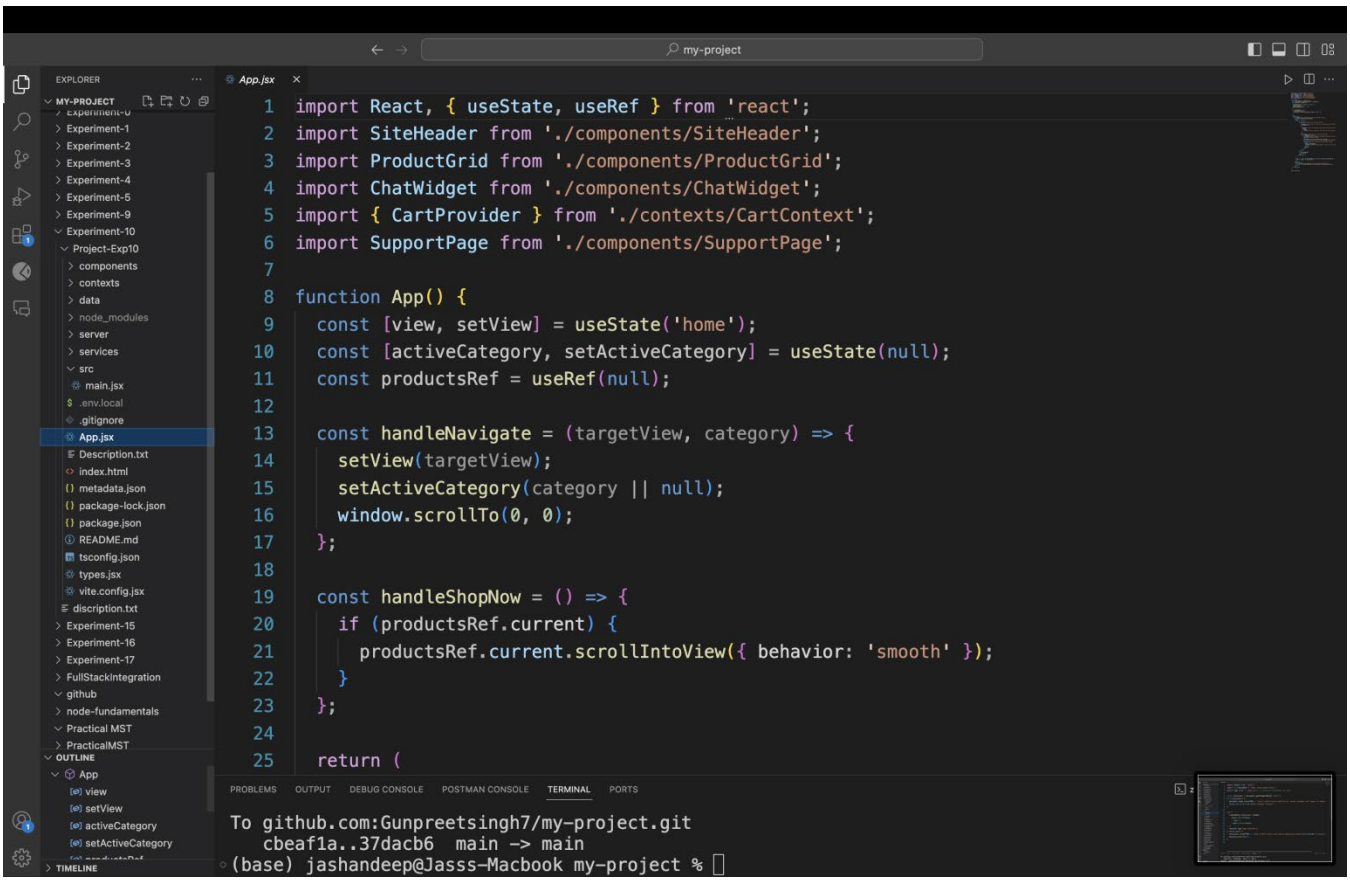
<button
onClick={() => addToCart(product)}
className="flex items-center gap-1 bg-indigo-600 hover:bg-indigo-500 text-white
>
PROBLEMS
OUTPUT
DEBUG CONSOLE
POSTMAN CONSOLE
TERMINAL
PORTS
To github.com:Gunpreetsingh7/my-project.git
cbeaf1a..37dacb6 main -> main
(base) jashandeep@Jasss-Macbook my-project %

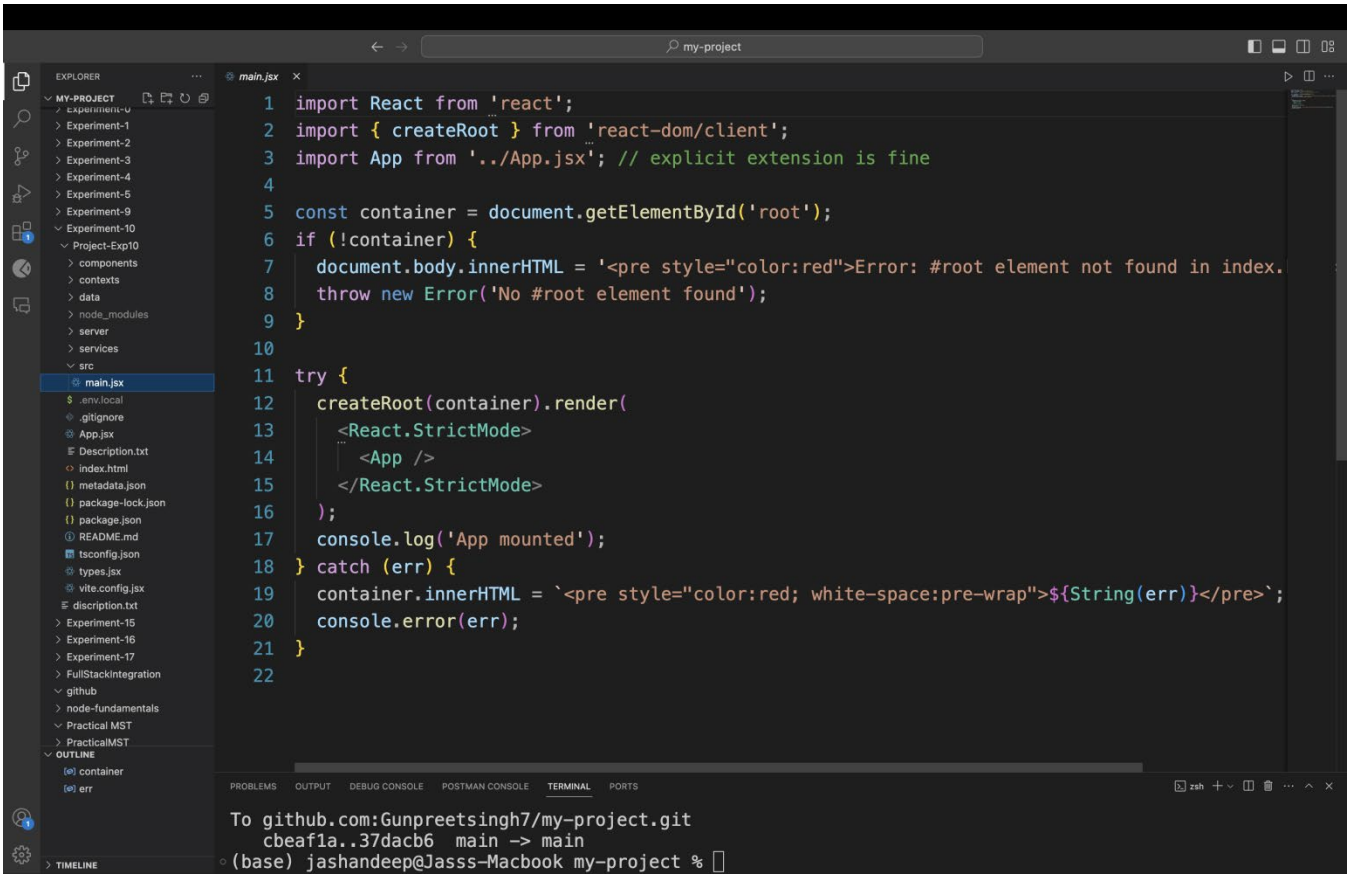

my-project
EXPLORER
MY-PROJECT
ChatWindow.jsx
import React, { useState, useCallback } from 'react';
import Header from './Header';
import MessageList from './MessageList';
import ChatInput from './ChatInput';
import TypingIndicator from './TypingIndicator';
import { sendMessageToAI, summarizeConversation, clearChatSession } from '../services/chatSe
import { products } from '../data/products';
const ChatWindow = () => {
const [messages, setMessages] = useState([
{
id: 'initial-message',
role: 'model',
text: "Hello! I'm your AI assistant. How can I help you today with products or orders?
timestamp: new Date().toLocaleTimeString([], { hour: '2-digit', minute: '2-digit' }),
},
]);
const [isLoading, setIsLoading] = useState(false);
const [rateLimitMessage, setRateLimitMessage] = useState(null);
const handleSendMessage = useCallback(
async (text) => {
const trimmed = (text || '').trim();
if (!trimmed) return;
PROBLEMS
OUTPUT
DEBUG CONSOLE
POSTMAN CONSOLE
TERMINAL
PORTS
To github.com:Gunpreetsingh7/my-project.git
cbeaf1a..37dacb6 main -> main
(base) jashandeep@Jasss-Macbook my-project %


my-project
Snapchat
3 Notifications
EXPLORER
MY-PROJECT
ChatWidget.jsx
1 import React, { useState } from 'react';
2 import ChatWindow from './ChatWindow';
3 import { MessageSquare, X } from 'lucide-react';
4
5 const ChatWidget = () => {
6 const [isOpen, setIsOpen] = useState(false);
7
8 return (
9
10 <div
11 className={`fixed bottom-5 right-5 z-50 transition-all duration-300 ease-in-out ${
12 isOpen ? 'w-11/12 max-w-lg h-3/4' : 'w-16 h-16'
13 }`}
14 >
15
16 {isOpen && (
17 <div className="absolute top-0 left-0 w-full h-full opacity-100 transition-opaci
18 <ChatWindow />
19
20)}
21
22
23
24 <button
25 onClick={() => setIsOpen(!isOpen)}
PROBLEMS OUTPUT DEBUG CONSOLE POSTMAN CONSOLE TERMINAL PORTS
To github.com:Gunpreetsingh7/my-project.git
cbeaf1a..37dacb6 main -> main
(base) jashandeep@Jasss-Macbook my-project %


my-project
Snapchat
Notification
EXPLORER
MY-PROJECT
ChatInput.jsx
1 import React, { useState, useRef } from 'react';
2 import { ArrowUp, ShieldAlert } from 'lucide-react';
3
4 const ChatInput = ({ onSendMessage, isLoading, rateLimitMessage }) => {
5 const [text, setText] = useState('');
6 const textareaRef = useRef(null);
7
8 const isInputDisabled = isLoading || !!rateLimitMessage;
9
10 const handleSubmit = () => {
11 if (text.trim() && !isInputDisabled) {
12 onSendMessage(text);
13 setText('');
14 if (textareaRef.current) {
15 textareaRef.current.style.height = 'auto'; // Reset height after send
16 textareaRef.current.focus();
17 }
18 }
19 };
20
21 const handleKeyDown = (event) => {
22 if (event.key === 'Enter' && !event.shiftKey) {
23 event.preventDefault();
24 handleSubmit();
25 }
PROBLEMS OUTPUT DEBUG CONSOLE POSTMAN CONSOLE TERMINAL PORTS
To github.com:Gunpreetsingh7/my-project.git
cbeaf1a..37dacb6 main -> main
(base) jashandeep@Jasss-Macbook my-project %


my-project
EXPLORER
MY-PROJECT
Experiment-1
Experiment-2
Experiment-3
Experiment-4
Experiment-5
Experiment-9
Experiment-10
Project-Exp10
components
Cart.jsx
ChatInput.jsx
ChatWidget.jsx
ChatWindow.jsx
CheckoutSuccessPage.jsx
Header.jsx
Message.jsx
MessageList.jsx
ProductCard.jsx
ProductGrid.jsx
Project-Exp10.code-work...
SiteHeader.jsx
SupportPage.jsx
TypingIndicator.jsx
contexts
data
node_modules
server
services
src
main.jsx
.env.local
.gitignore
App.jsx
Description.txt
index.html
metadata.json
package-lock.json
OUTLINE
Cart
isOpen
onClose
cartItems
removeFromCart
TIMELINE
Cart.jsx
1 import React from 'react';
2 import { useCart } from '../contexts/CartContext';
3 import { X, Trash2 } from 'lucide-react';
4
5 const Cart = ({ isOpen, onClose }) => {
6 const { cartItems, removeFromCart, updateQuantity, clearCart, cartCount } = useCart();
7
8 const total = cartItems.reduce((sum, item) => sum + item.price * item.quantity, 0);
9
10 return (
11 <>
12 <div
13 className={`fixed inset-0 bg-black/60 z-40 transition-opacity ${
14 isOpen ? 'opacity-100' : 'opacity-0 pointer-events-none'
15 }`}
16 onClick={onClose}
17 />
18 <div
19 className={`fixed top-0 right-0 h-full w-full max-w-md bg-slate-800 text-white shado
20 isOpen ? 'translate-x-0' : 'translate-x-full'
21 }`}
22 >
23
24
25 <h2 className="text-xl font-semibold">Your Cart ({cartCount})</h2>
PROBLEMS
OUTPUT
DEBUG CONSOLE
POSTMAN CONSOLE
TERMINAL
PORTS
zsh
To github.com:Gunpreetsingh7/my-project.git
cbeaf1a..37dacb6 main -> main
(base) jashandeep@Jasss-Macbook my-project %


my-project
EXPLORER
MY-PROJECT
Experiment-1
Experiment-2
Experiment-3
Experiment-4
Experiment-5
Experiment-9
Experiment-10
Project-Exp10
components
contexts
data
node_modules
server
services
src
main.jsx
.env.local
.gitignore
App.jsx
Description.txt
index.html
metadata.json
package-lock.json
package.json
README.md
tsconfig.json
types.jsx
vite.config.jsx
discription.txt
Experiment-15
Experiment-16
Experiment-17
FullStackIntegration
github
node-fundamentals
Practical MST
PracticalMST
OUTLINE
App
view
setView
activeCategory
setActiveCategory
TIMELINE
App.jsx
1 import React, { useState, useRef } from 'react';
2 import SiteHeader from './components/SiteHeader';
3 import ProductGrid from './components/ProductGrid';
4 import ChatWidget from './components/ChatWidget';
5 import { CartProvider } from './contexts/CartContext';
6 import SupportPage from './components/SupportPage';
7
8 function App() {
9 const [view, setView] = useState('home');
10 const [activeCategory, setActiveCategory] = useState(null);
11 const productsRef = useRef(null);
12
13 const handleNavigate = (targetView, category) => {
14 setView(targetView);
15 setActiveCategory(category || null);
16 window.scrollTo(0, 0);
17 };
18
19 const handleShopNow = () => {
20 if (productsRef.current) {
21 productsRef.current.scrollIntoView({ behavior: 'smooth' });
22 }
23 };
24
25 return (
PROBLEMS
OUTPUT
DEBUG CONSOLE
POSTMAN CONSOLE
TERMINAL
PORTS
To github.com:Gunpreetsingh7/my-project.git
cbeaf1a..37dacb6 main -> main
(base) jashandeep@Jasss-Macbook my-project %


my-project
EXPLORER
MY-PROJECT
Experiment-1
Experiment-2
Experiment-3
Experiment-4
Experiment-5
Experiment-9
Experiment-10
Project-Exp10
components
contexts
data
node_modules
server
services
src
main.jsx
.env.local
.gitignore
App.jsx
Description.txt
index.html
metadata.json
package-lock.json
package.json
README.md
tsconfig.json
types.jsx
vite.config.jsx
discription.txt
Experiment-15
Experiment-16
Experiment-17
FullStackIntegration
github
node-fundamentals
Practical MST
PracticalMST
OUTLINE
container
err
TIMELINE
main.jsx
1 import React from 'react';
2 import { createRoot } from 'react-dom/client';
3 import App from '../App.jsx'; // explicit extension is fine
4
5 const container = document.getElementById('root');
6 if (!container) {
7 document.body.innerHTML = '<pre style="color:red">Error: #root element not found in index.
8 throw new Error('No #root element found');
9 }
10
11 try {
12 createRoot(container).render(
13 <React.StrictMode>
14 <App />
15 </React.StrictMode>
16);
17 console.log('App mounted');
18 } catch (err) {
19 container.innerHTML = `<pre style="color:red; white-space:pre-wrap">${String(err)}</pre>`;
20 console.error(err);
21 }
22
PROBLEMS
OUTPUT
DEBUG CONSOLE
POSTMAN CONSOLE
TERMINAL
PORTS
zsh
To github.com:Gunpreetsingh7/my-project.git
cbeaf1a..37dacb6 main -> main
(base) jashandeep@Jasss-Macbook my-project %

BIBLIOGRAPHY

Online tutorials on **MERN Stack Development** (MongoDB, Express.js, React.js, Node.js)

Stack Overflow developer discussions and troubleshooting threads

Official Express.js Documentation - https://expressjs.com/

Official MongoDB Documentation - https://www.mongodb.com/docs/

Mongoose Documentation (Schema design and data modeling) https://mongoosejs.com/docs/

Node.js Official Documentation (Backend framework and modules) https://nodejs.org/en/docs

React.js Official Documentation (Frontend development and components) https://react.dev/

Cloudinary Documentation (Product image uploads and media handling)

https://cloudinary.com/documentation

Nodemailer Documentation (Email verification and OTP functionality)

https://nodemailer.com/about/

Joi Validation Documentation (Form validation and input handling) - https://joi.dev/api/

Stripe Documentation (For payment gateway integration reference) https://stripe.com/docs

GitHub Repositories & Open-Source Examples

(General reference to full-stack e-commerce projects)

MDN Web Docs (JavaScript, HTTP, Web Security, DOM concepts) -

https://developer.mozilla.org/

GeeksforGeeks / W3Schools (For conceptual understanding and implementation guidance) - https://www.geeksforgeeks.org/
