📘MERN Course Commerce App – Full Stages, Prompts & Testing Guide

---

🩲Live App Links

- **Frontend**: https://bejewelled-pothos-9dae62.netlify.app/
- **Backend**: https://course-commerce-bakend.onrender.com
- **GitHub**: https://github.com/m-ai-stud-io/course-commerce

---

🧪Testing the Application

🎓As a Normal User:

1. Visit the site and click "Register"
2. Create a new user with email and password
3. Login and verify:
4. Browse and view courses
5. Add to cart and checkout
6. Access Dashboard to view purchased courses
7. Admin features are not accessible

💼As an Admin:

- **Email**: `react@react.com`
- **Password**: `react`

- Login with the above credentials

- Access Admin Dashboard:
- Add/Edit/Delete courses
- View all listed courses

🛠️Stage 1: Project Setup

🕰️Tasks:

- Create folders `/client` and `/server`
- Basic React and Express apps
- Connect both with `gemini dev`
- Add test routes

📝**Gemini CLI Prompt:**

```
Create a full stack MERN application for an online course commerce site
called "Course Commerce". The frontend should be built with React and the
backend with Node.js and Express. The project should have separate folders
for /client and /server. The backend should include a basic Express setup
that listens on port 5000 and has one route at "/" returning "Hello from
Backend". The frontend should display "Hello React" on the home page. Both
frontend and backend should be ready to run together using `gemini dev`.
```

🔐Stage 2: Authentication

🕰️Tasks:

- User model with JWT auth
- Register/Login API
- Frontend login/register pages
- Store token & protect routes

📝Gemini CLI Prompt:

```
Add authentication to the backend in the /server folder. Create a User model
with fields: name, email, password, and role (default role: 'user'). Set up
bcryptjs for password hashing and jsonwebtoken for token generation. Create
two routes: POST /api/auth/register for user registration and POST /api/auth/
login for login. In registration, hash the password before saving. In login,
validate the email and password, and return a signed JWT token if successful.
Connect the routes in index.js under /api/auth.
```

🔒Protect Routes:

```
In the React frontend, create a ProtectedRoute component that checks if a
user is logged in by verifying if a JWT token exists in localStorage. If the
token is present, allow access to the child component; if not, redirect the
user to the /login page. Use React Router for routing. Apply this
ProtectedRoute component to secure pages like the user dashboard and any
future admin pages.
```

📊Stage 3: Admin Course Management

🕰️Tasks:

- Course model & CRUD routes
- Admin-only protection
- Admin dashboard in frontend

- Add/Edit/Delete functionality

📝**Gemini CLI Prompt:**

```
In the backend, create a Course model with title, description, price, image,
and videoUrl. Add CRUD routes: POST, GET, PUT, DELETE for /api/courses.
Protect creation, update, and delete with JWT role middleware (admin only).
In the frontend, create an AdminDashboard.jsx to list, add, edit, and delete
courses. Create a CourseForm.jsx for input. Protect admin dashboard using the
ProtectedRoute.
```

📚Stage 4: Course Display for Users

🕰️**Tasks:**

- Courses.jsx to show all
- CourseDetail.jsx for individual view
- Display title, image, price, video link
- Add to Cart button

📝**Gemini CLI Prompt:**

```
Create Courses.jsx and CourseDetail.jsx in the /src/pages folder. Courses.jsx
fetches from /api/courses and displays in card layout. Each card has title,
image, price, and view details button. CourseDetail.jsx shows the full info
of a course and includes an "Add to Cart" button.
```

🛒Stage 5: Cart & Checkout

🕰️**Tasks:**

- Create CartContext
- Add to Cart from CourseDetail
- Cart.jsx page to review items
- Checkout + payment integration
- Save orders to MongoDB

📝**Gemini CLI Prompt:**

```
Create a CartContext in the React frontend. Display selected courses in
Cart.jsx with total price and remove option. Create Checkout.jsx to simulate
payment. In backend, add /api/checkout to process Stripe test payments and
save Order model with user ID, courses, total amount.
```

Stage 6: User Dashboard

🕰️**Tasks:**

- Dashboard.jsx to show purchased courses
- Show profile info (name, email)
- Only accessible by logged-in users

📝**Gemini CLI Prompt:**

```
Create Dashboard.jsx in the frontend. Show user's orders fetched from /api/
orders. Display course image, title, videoUrl. Include user info like name
and email. Protect dashboard with JWT token.
```

🕔Stage 7: UI Polish & Deployment

🕰️**Tasks:**

- Style with modern UI & glassmorphism
- Improve layout, spacing, colors
- Add toast messages and loading states
- Deploy frontend to Netlify, backend to Render

📝**Gemini CLI Prompt:**

```
Enhance frontend with glassmorphism: translucent backgrounds, shadows,
rounded corners, smooth transitions. Improve button styles, padding, layout
spacing. Add react-toastify for error/success messages. Deploy React app to
Netlify and Express backend to Render. Use .env for secrets and update
frontend API URLs to point to Render.
```

✅Your full MERN course commerce app is complete with:

- All development stages
- Gemini CLI prompts
- Testing credentials
- Deployment and links

You're ready to launch, iterate, or scale! 🚯