EtRaDe - E-Commerce Project Progress

1 Project Overview

Full-stack e-commerce application with React frontend and Node.js backend, supporting both manual and Google authentication.

---

2 What We Have Completed

2.1 Backend Infrastructure

- Express.js server setup with CORS and middleware
- MongoDB connection and database setup
- Cookie parser and body parser configuration
- Environment variables support (.env)

2.2 Database Models

2.2.1 Product Model (`models/products.js`)

- Complete product schema with rating system
- ES modules export format

2.2.2 User Model (`models/user.js`)

- Hybrid authentication support (manual + Google)
- Enhanced fields: `emailVerified`, `isActive`, `lastLogin`, `updatedAt`
- Automatic timestamps

2.3 Authentication System

2.3.1 JWT Utility (`utils/jwt.js`)

- Centralized token generation and verification
- Secure cookie configuration
- Environment-based settings

2.3.2 Auth Controllers (`controllers/auth.js`)

- **Registration**: Single endpoint for both manual and Google users
- **Login**: Smart login handling both authentication types
- **Token Verification**: Route for automatic login validation

2.3.3 Auth Middleware (`middlewares/authMiddleware.js`)

- `verifyToken`: Protects routes requiring authentication
- `optionalAuth`: For routes that work with/without auth
- Support for both cookies and Authorization headers

2.4 API Routes Structure

2.4.1 Auth Routes (`/api/auth/`)

- `POST /register` - User registration
- `POST /login` - User login
- `GET /verify-token` - Token validation for auto-login

2.4.2 Product Routes (`/api/`)

- `GET /products` - Public product viewing (with optional user context)

2.4.3 User Profile Routes (`/api/user/profile/`)

- `GET /` - Get user profile
- `PUT /` - Update profile
- `PUT /change-password` - Change password (manual users only)

2.5 Security Features

- HTTP-only cookies for token storage
- CORS configuration with credentials
- Password hashing with bcrypt
- JWT token expiration (7 days)
- Account status validation
- Cross-authentication protection

3 Next Steps (In Order)

3.1 Phase 1: Complete Backend API

1. **Product Management**

   Create `getProductById` controller

   Add product search/filter functionality

   Add product categories endpoint

   Implement product CRUD operations (admin)

2. **Shopping Cart System**

   Create Cart model (`models/cart.js`)

   Create Cart controller (`controllers/cart.js`)

   Cart routes: Add, Remove, Update quantity, Clear

   Link cart to user sessions

3. **Order Management**

   Create Order model (`models/order.js`)

   Create Order controller (`controllers/order.js`)

   Order placement and history endpoints

   Order status tracking

4. **Additional Features**

   Wishlist functionality

   Product reviews and ratings

   User address management

3.2 Phase 2: Frontend Development

5. **React App Setup**

   Configure routing (React Router)

   Set up authentication context/state management

   Create reusable components (Header, Footer, etc.)

6. **Authentication UI**

   Login page with manual and Google auth

   Registration page

   Implement auto-login with `/verify-token`

   User profile and settings pages

7. **Product Pages**

Product listing page
Product detail page
Search and filter functionality
Product categories

8. **Shopping Features**

Shopping cart UI
Checkout process
Order confirmation and history
Wishlist interface

3.3 Phase 3: Integration & Testing

9. **Google Authentication**

Set up Firebase project
Configure Google OAuth
Integrate with backend authentication

10. **Payment Integration**

Choose payment gateway (Stripe/PayPal)
Implement payment processing
Order confirmation system

11. **Testing & Deployment**

API testing (Postman/Thunder Client)
Frontend testing
Error handling and validation
Production deployment setup

4 Current File Structure

```
Backend/
 controllers/
    auth.js
    products.js
 db/
    database.js
 middlewares/
    authMiddleware.js
 models/
    products.js
    user.js
 routes/
    authRoute.js
    productRoute.js
    userProfile.js
 utils/
    jwt.js
 index.js
 package.json

Frontend/
 src/
    App.jsx
    main.jsx
    index.css
 index.html
 package.json
```

5 Environment Setup Required

Create `.env` file in Backend with:

```
MONGODB_URI=your_mongodb_connection_string
JWT_SECRET=your_super_secret_jwt_key
NODE_ENV=development
PORT=5000
```

6 Key Design Decisions Made

1. **Authentication Strategy**: Hybrid system supporting both manual and Google login
2. **Product Access**: Public viewing with login required for actions
3. **Token Management**: HTTP-only cookies for security + localStorage option
4. **API Structure**: RESTful design with clear separation of concerns
5. **Middleware Usage**: Smart authentication with `verifyToken` and `optionalAuth`

7 Immediate Next Action

Start with Phase 1, Step 1: Complete the product management system by adding:

- Product detail retrieval
- Search and filter functionality
- Product categories

This will give you a solid foundation before moving to cart and order systems.