Full Stack Onboarding

Project Overview:

Build a simple e-commerce platform where users can browse products, add them to a cart, and place orders. The platform should include authentication, authorization, and user management. The frontend will be built using Next.js with Tailwind CSS, and React Query (TanStack) for data fetching. The backend will use NestJS, TypeORM, and PostgreSQL.

Step 1: Project Setup & Authentication

Tasks:

1. **Backend Setup:**
   - Initialize a NestJS project.
   - Install and configure TypeORM with PostgreSQL.
   - Create database entities for `User` and `Role`.
   - Implement user registration and login using JWT for authentication.
   - Use guards to protect routes that require authentication.
   - Write unit tests for authentication-related services and controllers.
2. **Frontend Setup:**
   - Initialize a Next.js project.
   - Set up Tailwind CSS.
   - Create pages for user registration and login.
   - Implement forms for registration and login, integrating them with the backend using React Query.
   - Use Yup for form validation.

Deliverables:

- A NestJS project with user authentication and protected routes.
- A Next.js project with user registration and login functionality, with Yup validation.

Step 2: User Management & Authorization

Tasks:

1. **Backend:**
   - Create an admin role and implement role-based access control.
   - Implement guards to restrict access to certain routes based on roles (e.g., only admins can access certain resources).
   - Create a route for managing users (CRUD operations).
   - Write unit tests for user management services and controllers.
2. **Frontend:**
   - Implement pages and components for user management (admin only).
   - Use React Query to fetch and manage user data.
   - Use Yup for form validation in user management forms.

Deliverables:

- Backend routes and guards for role-based access control with unit tests.
- Frontend pages for managing users with Yup validation.

Step 3: Product Management

Tasks:

1. **Backend:**
   - Create database entities for `Product` and `Category`.
   - Implement CRUD operations for products and categories.
   - Secure product management routes so only admins can add, edit, or delete products.
   - Write unit tests for product management services and controllers.
2. **Frontend:**
   - Implement pages for viewing products, adding new products, editing existing products, and deleting products (admin only).
   - Use React Query to fetch product data and manage state.
   - Use Yup for form validation in product management forms.

Deliverables:

- A fully functional backend for managing products with unit tests.
- Frontend pages for product management with Yup validation.

Step 4: Shopping Cart & Orders

Tasks:

1. **Backend:**
   - Create database entities for `Cart`, `CartItem`, and `Order`.
   - Implement endpoints for adding items to the cart, removing items, and placing orders.
   - Implement JWT guards to ensure that only authenticated users can add to cart and place orders.
   - Write unit tests for cart and order services and controllers.
2. **Frontend:**
   - Create a shopping cart page where users can add or remove items.
   - Implement order placement functionality.
   - Use React Query to manage cart and order state.

Deliverables:

- Backend functionality for managing carts and orders with unit tests.
- Frontend pages for shopping cart and order placement.

Step 5: Final Touches & Deployment

Tasks:

1. **Frontend:**
   - Add Tailwind CSS styling to improve the UI/UX.
   - Implement a responsive design for all pages.
   - Add client-side validation using Yup for all forms.
2. **Backend:**
   - Write additional unit tests for key services and controllers in NestJS.
   - Optimize database queries and add pagination to product lists.
3. **Deployment:**
   - Deploy the application using Docker.
   - Provide instructions for deploying the project locally.

Deliverables:

- Fully styled and responsive frontend with Yup validation.
- Comprehensive unit tests and optimized backend.
- Deployed project with Docker instructions.

Bonus (Optional):

- **Cypress Automation:** Implement end-to-end testing using Cypress to automate the testing of critical user flows (e.g., registration, login, product management, cart operations).
- **Implement a Wishlist feature:** Allow users to add products to a wishlist.

This step-by-step project will provide the intern with a comprehensive understanding of the technologies mentioned and allow them to build a full-stack application from scratch.