Frontend Developer Intern – MERN Stack (Onsite)

SKILL ASSESSMENT TASK

Project Overview:
This project is a robust E-commerce platform built using industry-standard technologies and best practices. It provides secure user authentication with JWT, comprehensive product management with CRUD operations powered by Axios, and dynamic category management for efficient organization. The backend is built with Express.js and Mongoose, ensuring reliable data handling and robust error management.

Technology Stack

- Programming Language: JavaScript / TypeScript
- Web Framework: Next.js / React
- Backend: Node.js + Express.js + MongoDB
- Authentication: JWT (JSON Web Tokens)
- State Management: Redux / React Context API
- Email Service: Nodemailer
- Styling: Tailwind CSS + Framer Motion
- Notifications: React Toastify
- CSS Framework : Tailwind CSS

Integration of Backend Functionality:
To integrate the backend functionality for the E-commerce platform, implement authentication, CRUD operations for various entities, and ensure a secure and well-structured approach. Here's a breakdown:

1. Authentication

User Registration
- Inputs: Name, Email, Password, and Photo.
- Storage: Store user photos either in a public directory or via a public URL.
- Validation: Implement rigorous email and password validation.

Login
- Mechanism: Users log in with email and password.
- JWT Tokens:
  - Access Token: Valid for 1 hour.

2. User Roles
- Roles: Admin, User.
- Functionality differs based on user roles.

3. CRUD Operations
*All CRUD operations are implemented using **Axios** / **Redux** and **RTK Query***

Products
- Manage Name, Slug, Multiple Photos, Description, Price, Discount, Stock Status (true/false), Status (active/inactive), and Categories.

Orders
- Manage order details including status and filtering by date.

4. Backend Implementation

Technology Stack
- Framework: Express.js (for RESTful API), Mongoose(Optional)
- Authentication: JWT-based authentication using jsonwebtoken and bcrypt for password hashing
- Database: MongoDB

Code Structure
- Controllers: Handle logic for user authentication, CRUD operations, etc.
- Models: Define schemas for users, products and orders
- Routes: Define endpoints for accessing different functionalities
- Middleware: Implement authentication middleware to protect routes

5. Frontend Integration

Registration and Login Forms
- Handle form submissions and validations

Admin Dashboard
- Create pages for managing users, products, orders

Product Pages
- Implement product detail pages, add-to-cart functionality, and show related products

6. Add to Cart Functionality

Button State Change
- Initially display "Add to Cart" on the product page
- Add to Cart: On click, add the product to the shopping cart and change the button to "View Cart"
- View Cart: Clicking "View Cart" opens a modal displaying the shopping cart.

7. Shopping Cart Functionality
- Manage Quantities: Allow users to increment or decrement product quantities using input fields and buttons
- Stock Handling: Display "Stock Out" if a product's quantity reaches zero and prevent addition or increment

8. Cart Summary
- Summary Calculation: Include total price and shipping charges

9. Modal Interaction
- Cart Modal: Clicking on a shopping cart item opens the shopping cart modal

10. State Management (Optional)
- Use Redux for state management and redux-persist to persist shopping cart data across sessions

11. Checkout Process
- Clicking "Proceed to Checkout" redirects logged-in users to the shipping page; prompt login for others

12. Product Details Page

- Display all product information with options to add to the cart
- Allow users to manage quantity (increase/decrease)
- Change "Add to Cart" button to "View Cart" once the product is added

13. Deployment and Testing

- Deployment: Ensure the application is deployed on a platform like Vercel or Heroku
- Testing: Write tests for API endpoints and frontend components to ensure functionality and security

14. Documentation

- README: Provide clear instructions for running the application locally, including all necessary environment variables

Notes / Best Practices

- Use pagination on all admin dashboard tables and product pages or Minimum one table
- Ensure 100% error handling on all functions and pages
- All components must be reusable
- Follow industry-standard coding practices
- Maintain an organized file structure
- Demonstrate best practices in code organization, security, and user experience
- Include the .env file (cannot be ignored)
- Add meaningful comments in the code
- All variables must use types
- Use Redux for all state management

Submission Guidelines

Send completed tasks to: career@thezoomit.com

- GitHub Repository: Include all code and a README file
- Provide the .env file
- Provide separate frontend and backend GitHub repositories
- Provide credentials for Super Admin and Admin accounts
- Live Deployment: Include the link to the deployed application
- Database Design: Include a PDF outlining the database schema

Deadline
September 05, 2025, 11:59 PM

Submission Rule:

- Task-Related Assistance: Seeking help from any ZOOM IT team members regarding this task is strictly prohibited. Violation of this rule will result in immediate disqualification of your assessment.
- Ensure you adhere to the submission deadline and all rules outlined above.
- For any questions or clarifications, please reach out. We look forward to reviewing your submissions and evaluating your skills for the Frontend Developer Intern – MERN Stack (Onsite) position at ZOOM IT.

Support Contact:
For assistance, reach out via email at ***support@thezoomit.com*** *or* ***info@thezoomit.com***