### Technology Used: MERN Stack (MongoDB, ExpressJS,ReactJS,NodeJS)

#### API Endpoints

**POST /api/signin**: Authenticates a user with email and password, returning JWT tokens (access and refresh token) upon successful login.

**POST /api/signup**: Registers a new user with email, password, and other required details.

**POST /api/update-profile**: Updates the authenticated user’s profile information, including profile image upload.

**GET /api/self**: Retrieves the profile details of the currently authenticated user.

**GET /api/refresh-tokens**: Refreshes JWT tokens using a valid refresh token, providing a new access token.

**POST /api/logout**: Logs out the authenticated user by invalidating the current session/token.

**POST /api/items**: Adds a new item, with image upload, for the authenticated user.

**PUT /api/items/:id**: Updates an existing item by ID, including images, for the authenticated user.

**DELETE /api/items/:id**: Deletes an item by ID for the authenticated user.

**GET /api/items**: Retrieves a list of all items.

**GET /api/items/my-items**: Retrieves a list of items owned by the authenticated user.

**GET /api/items/recents**: Retrieves a list of recently added items.

**GET /api/items/details/:id**: Retrieves details of a specific item by ID.

**GET /api/items/seller**: Retrieves items listed by a specific seller.

**POST /api/items/wishlists**: Adds or removes an item from the authenticated user’s wishlist.

**GET /api/items/wishlists**: Retrieves the authenticated user’s wishlist.

**POST /api/orders/buy**: Creates a new purchase order for the authenticated user.

**POST /api/orders/rent**: Creates a new rental order for the authenticated user.

**POST /api/orders/swap**: Creates a new swap order for the authenticated user.

**GET /api/orders/my-orders**: Retrieves a list of orders made by the authenticated user.

**GET /api/orders/my-sales**: Retrieves a list of sales made by the authenticated user.

**PUT /api/orders/update**: Updates the status of an order for the authenticated user.

**GET /api/orders/notifications**: Retrieves notifications related to orders for the authenticated user.

**GET /api/orders/notifications/markAsRead**: Marks all order-related notifications as read for the authenticated user.

#### Libraries for Backend Used

**bcrypt**: Used for hashing passwords for secure storage.

**cloudinary**: Handles image and file uploads to Cloudinary cloud service.

**cookie-parser**: Parses cookies attached to the client request object.

**cors**: Enables Cross-Origin Resource Sharing for handling requests from different domains.

**dotenv**: Loads environment variables from a .env file into process.env.

**express**: Web framework for building RESTful APIs and handling HTTP requests.

**joi**: Validates user input data to ensure it meets specified criteria.

**jsonwebtoken**: Generates and verifies JSON Web Tokens (JWT) for authentication.

**mongoose**: Provides MongoDB object modeling for creating schemas and interacting with MongoDB.

**multer**: Middleware for handling multipart/form-data, primarily used for file uploads.

**nodemailer**: Sends emails from Node.js applications.

**uuid**: Generates unique identifiers (UUIDs) for resources.

**nodemon** (Dev dependency): Automatically restarts the server when file changes in the directory are detected during development.

#### Frontend

**React JS**: JavaScript library for building user interfaces.

#### Libraries for Frontend

**tanstack/react-query**: Handles server state management and data fetching in React applications.

**axios**: HTTP client used for making API requests.

**date-fns**: Utility library for date manipulation and formatting.

**formik**: Form library for managing form state and validation in React applications.

**moment**: Library for parsing, validating, manipulating, and formatting dates.

**react**: JavaScript library for building user interfaces.

**react-dom**: Provides DOM-specific methods that can be used at the top level of a React application.

**react-hot-toast**: Library for displaying customizable toast notifications in React.

**react-modal**: Component for creating accessible modals using React.

**react-router-dom**: Handles routing and navigation in React applications.

**react-select**: Provides a customizable select box component for React.

**swiper**: Library for creating modern touch sliders and carousels.

**yup**: Schema builder for value parsing and validation.

**zustand**: Small, fast, and scalable state management library for React.

### Setup Instructions

Ensure you have all the necessary software installed, such as Node.js and npm.

#### Backend Setup

1. **Extract the folders**:

Open it in VSCode or terminal and use the below command:

cd backend

1. **Install Dependencies**:

* npm install

1. **Set Up Environment Variables**: Update a .env file in the root directory and add your environment variables (e.g., MongoDB database URL, JWT secret).
2. **Run the Server**: For development, use this command:

* npm run dev

If your terminal shows that your Database is connected then it means we are good to proceed.

#### Frontend Setup

1. **Open folder in terminal**:

* Open another terminal in parallel and keep the backend server on. Use below command
* cd frontend
* No need to update .env in frontend folder if you are running locally.

1. **Install Dependencies**:

* npm install

1. **Run the Development Server**:

* npm run dev

this will start your frontend and you can use as you want.

OPTIONAL:

1. **Build the Project (for this you need to change frontend and backend URLs in both frontend and backend folder’s .env files)**:

* npm run build