**User Features**

1. **User Registration (POST /users/register):**
   * **Users can register by providing their name, email, and password.**
   * **Passwords are securely hashed before being stored in the database.**
   * **On successful registration, a JWT token is generated and returned for future authentication.**
2. **User Login (POST /users/login):**
   * **Registered users can log in using their email and password.**
   * **On successful login, a JWT token is returned.**
   * **This token is used to authenticate and authorize users for protected routes.**
3. **Role-Based Access (Admin and Normal Users):**
   * **Normal Users can:**
     + **Browse products.**
     + **Add reviews for products.**
     + **Update or delete only their own reviews (but not reviews written by others).**
   * **Admin Users:**
     + **Can create, update, and delete products.**
     + **Can delete any review.**
4. **Protected Routes (Authentication):**
   * **JWT-based protection ensures that users must be logged in to perform certain actions (e.g., create a product or add a review).**
   * **The middleware protect checks the token validity and assigns the user to the request object.**
   * **Role-based access is enforced using the admin middleware, ensuring only admin users can perform certain actions.**
5. **User Permissions and Access:**
   * **Normal users can access:**
     + **Public routes such as browsing products and viewing product details.**
     + **Protected routes such as adding reviews, updating their own reviews.**
   * **Admin users can access:**
     + **All product management routes (create, update, delete).**
     + **All review management routes (including the ability to delete any review).**
6. **Forgot password implemented**

**Product Features**

1. **Product Creation (POST /products):**
   * **Admin only can create new products by providing details such as name, description, price, size, woodType, and dateOfMade.**
2. **Product Update (PUT /products/):**
   * **Admin only can update existing products by providing the product ID and the updated details.**
3. **Product Deletion (DELETE /products/):**
   * **Admin only can delete a product by its ID.**
4. **Browse All Products (GET /products):**
   * **Anyone can browse all the available products in the store.**
5. **Product Search and Filters (GET /products/search):**
   * **Users can search products based on various filters like name, category, price range, woodType, and size.**
   * **Full-text search is enabled for name and description.**
6. **Product Sorting (GET /products/sort):**
   * **Users can sort products by price, createdAt, or other attributes in ascending or descending order.**
7. **Product Pagination (GET /products/paginate):**
   * **Products can be paginated to limit the number of results displayed per page.**
8. **Filter Products by Category (GET /products/category/**

**):**

* + **Users can filter products based on their categories (e.g., Chair, Table).**

1. **Recently Added Products (GET /products/recent):**
   * **Users can view the most recently added products, limited to the last 10 entries.**
2. **Upload Product Images (POST /products/**

**/upload):**

* **Admin only can upload images to products using multer for local image storage.**

**Review Features**

1. **Add a Review (POST /reviews/**

**/review):**

* + **Authenticated users can add a review for a product by providing a rating (1-5) and a comment.**

1. **Update/Delete a Review (DELETE /reviews/**

**):**

* + **Authenticated users can delete their own reviews.**
  + **Admin users can delete any review.**
  + **Each review is tied to both the product and the user who posted it.**

**Middleware Implemented**

1. **Authentication Middleware (protect):**
   * **Checks the presence of a valid JWT token in the request header to authenticate users.**
   * **Attaches the authenticated user to the request object (req.user).**
2. **Authorization Middleware (admin):**
   * **Ensures that only users with isAdmin: true can access admin routes for product management.**

**Security**

* **JWT Tokens: Used for secure user authentication and session management.**
* **Password Hashing: User passwords are securely hashed using bcrypt before being stored in the database.**
* **Protected Routes: Only authenticated users can access certain routes, and only admin users can manage sensitive data such as products and reviews.**
* **Email verification is used where verified user can review and purchase product.**

**Cart**

* **For logged-in users: The cart is stored in the database (MongoDB), and they must be authenticated.**
* **For guest users: The cart is stored in the session, so no authentication or token is required.**
* **Expired product check**
* **Stock check**
* **Guest cart to user cart migration and**
* **cart summary.**

**Order management:**

* **Order Creation & Checkout Process:**
* **Create New Order: Users can place orders with shipping and billing details.**
* **Re-validate Stock & Price: Before confirming the order, the system checks for stock availability and price consistency.**
* **Inventory Locking: During checkout, the system reduces stock from the inventory to "lock" the items for the user.**
* **Tax, Shipping, and Discounts: The system calculates additional costs like taxes and shipping, and applies discounts if provided.**
* **Payment Methods:**
  + **Cash on Delivery (COD): Orders are placed and marked as unpaid, awaiting delivery.**
  + **Online Payment (Dummy): Simulates a successful or failed payment to mimic the online payment process.**
* **Restore Inventory on Payment Failure: If an online payment fails, the inventory is restored, unlocking stock.**
* **Admin Notifications & Alerts:**
* **Real-time Order Notifications: Admins are notified via Socket.io when a new order is placed.**
* **Low Stock Alerts: When product stock falls below a threshold (e.g., 5 units), a real-time alert is sent to admins.**
* **Socket.io Integration: Used for real-time communication, ensuring admins receive updates about orders and inventory issues.**
* **Stock Management:**
* **Automatic Stock Reduction: Stock is automatically reduced when an order is confirmed.**
* **Low Stock Alerts: If the stock of a product drops below the defined threshold, admins are notified in real-time.**
* **Stock Restoration: If an order is cancelled or fails, the stock is restored to the inventory.**
* **Order Tracking & Status Management:**
* **Order Status Updates: Admins can change the status of an order (e.g., "Pending", "Shipped", "Delivered").**
* **Cancel or Refund Orders: Admins can cancel an order or issue a refund if necessary. Shipped orders cannot be cancelled.**
* **Inventory Management:**
* **Restock Products: Admins can restock products manually to manage inventory.**
* **Manage Inventory: Through the admin panel, admins can monitor and update inventory levels.**
* **User-Specific Order Management:**
* **View My Orders: Users can view their own order history through the /myorders route.**
* **Order Details: Detailed information about each order is provided, including shipping, billing, and product information.**

**Admin Panel:  
 Inventory Summary:**

* **Shows total products and low stock products (those below a certain threshold like 5 units).**
* **Automatically triggers low-stock alerts based on the quantity.**

** Order Summary:**

* **Displays total orders, pending orders, delivered orders, and total revenue.**
* **Helps admins track the overall sales and revenue flow.**

** User Activity:**

* **Fetches recent sign-ups (within the last 30 days) and displays total user count.**
* **Admins can monitor user registration trends and engagement.**

**Summary:**

**1. User Authentication & Authorization:**

* **User Registration (with email verification).**
* **Login (with cart persistence and guest-to-user cart migration).**
* **Password Reset (via email).**
* **Role-based Access (with user/admin roles).**

**2. Cart Management:**

* **Add/Remove items to/from cart.**
* **Guest Cart (session-based) and User Cart (DB-based).**
* **Cart Persistence across sessions for logged-in users.**
* **Cart migration after login from guest to user cart.**

**3. Checkout & Order Management:**

* **Checkout process with stock validation and price verification.**
* **Cash on Delivery and Online Payment (dummy).**
* **Order creation, with real-time inventory locking and low-stock alerts.**
* **Order cancellation, inventory restoration, and order status updates (for users and admin).**

**4. Admin Features:**

* **Admin Order Management (viewing orders, updating status, canceling orders).**
* **Inventory management (stock reduction, low-stock alerts).**
* **Order and Product Notifications using Socket.io for real-time admin updates.**
* **Admin Dashboard backend prepared for further frontend development.**

**What’s Left for Basic Functionality?**

1. **Testing: Perform end-to-end testing of all implemented features to ensure functionality.**
2. **Frontend UI:**
   * **Implement the admin dashboard UI (to receive notifications, manage inventory, view user orders).**
   * **User-facing features like cart, checkout, and order history need a frontend if not done already.**
3. **Real Payment Gateway Integration: If you want to go beyond the dummy payment setup.**
4. **Error Handling and Logging: You can add more comprehensive error logging and notifications in case of failures.**

**Next Possible Features:**

* **Email/SMS Notifications for users (e.g., order confirmation, shipping updates).**
* **Advanced Search/Filters for products (by category, price range, etc.).**
* **Discount Management: Implement discount codes, loyalty points, etc.**
* **Review and Rating System for products.**

**But from a basic backend perspective, you are in a strong position! You can now build on this solid foundation to add advanced features and refine the system as needed.**

**Note: For the new registration, email verification is implemented using *Ethereal.* Later we can remove this feature and use google signup directly or keep email verification as well as option of google signup.**

**Payment module not implemented.**