**Detailed Problem Statement: REST API for Products Display Page in ExpressJS eCommerce Application**

**Context:**

You are building a RESTful API using ExpressJS for an eCommerce platform. The API will serve product data to the front-end application for the Products Display Page, enabling users to browse, filter, sort, and paginate through the product catalog.

**Objective:**

Design and implement a RESTful API endpoint /api/products in ExpressJS that allows clients (web, mobile apps) to retrieve product information efficiently, with support for filtering, sorting, and pagination.

**Functional Requirements:**

1. **API Endpoint**
   * Create a GET /api/products endpoint that returns a paginated list of products.
   * The API response must include:
     + id (unique product identifier)
     + name
     + price
     + description (brief)
     + imageUrl (thumbnail or main image)
     + category
     + stockStatus (in stock/out of stock)
     + Optional: rating, popularityScore
2. **Filtering**
   * Support query parameters to filter results, for example:
     + category (e.g., /api/products?category=electronics)
     + priceMin and priceMax (e.g., /api/products?priceMin=10&priceMax=100)
     + inStock boolean flag to only return products currently available.
3. **Sorting**
   * Support sorting via a query parameter, e.g.:
     + sort=price\_asc or sort=price\_desc
     + sort=rating\_desc
     + Default sorting by product name ascending.
4. **Pagination**
   * Support pagination using page and limit query parameters.
   * For example, /api/products?page=2&limit=20 returns the second page with 20 products per page.
   * Include metadata in the response like total product count, current page, total pages.
5. **Error Handling**
   * Validate query parameters and return appropriate HTTP status codes for invalid input (e.g., 400 Bad Request).
   * Return meaningful error messages for unexpected failures (e.g., 500 Internal Server Error).
   * Return a 404 Not Found if no products match the filter criteria.

**Non-Functional Requirements:**

* Use ExpressJS middleware for query validation and error handling.
* Use async/await for asynchronous database operations.
* Optimize database queries for performance, using indexes if necessary.
* Implement logging for errors and request details.
* Ensure API security by sanitizing inputs to prevent injection attacks.
* Follow REST API best practices for status codes, headers, and response formats (preferably JSON).

**Example API Usage:**

**Request:**

bash

CopyEdit

GET /api/products?category=clothing&priceMin=20&priceMax=100&sort=price\_asc&page=1&limit=10

**Response:**

json

CopyEdit

{

"page": 1,

"limit": 10,

"totalProducts": 125,

"totalPages": 13,

"products": [

{

"id": "123",

"name": "Casual T-Shirt",

"price": 25.99,

"description": "Comfortable cotton t-shirt",

"imageUrl": "https://example.com/images/product123.jpg",

"category": "clothing",

"stockStatus": "in stock",

"rating": 4.5

},

...

]

}

**Deliverables:**

* ExpressJS route handlers for /api/products implementing filtering, sorting, and pagination.
* Middleware for input validation and error handling.
* Documentation specifying API usage, supported query parameters, and response schema.