1. **Writing a Server in Express:** Create a simple Express server that responds with "Hello, World!" on the root route.
2. **Creating a RESTful Route:** Implement a GET API that returns a list of users in JSON format.
3. **Handling POST Requests:** Build an API that accepts user data via a POST request and stores it in an array.
4. **Using Middleware in Express:** Implement a custom middleware that logs request details (method, URL, timestamp).
5. **Setting Up Route Parameters:** Create an API that accepts a dynamic user ID in the URL and returns user details.
6. **Handling Query Parameters:** Develop a search endpoint that filters users based on query parameters (e.g., ?name=John).
7. **Sending HTTP Status Codes Properly:** Modify an API to return appropriate HTTP status codes for success and errors.
8. **Using Environment Variables:** Configure an Express server to read port numbers and database URLs from a .env file.
9. **Serving Static Files:** Create an Express server that serves static files like images or HTML pages.
10. **Error Handling Middleware:** Implement a global error-handling middleware in Express for structured API error responses.
11. **CRUD Operations with Express and a JSON File:** Build a simple API that allows users to create, read, update, and delete (CRUD) entries from a JSON file.
12. **Implementing Authentication Using JWT:** Develop an API that registers users, generates JWT tokens, and validates protected routes.
13. **Connecting an Express API to MongoDB:** Create an API that saves user data to MongoDB using Mongoose.
14. **Paginating API Responses:** Implement pagination for a large dataset using query parameters (?page=1&limit=10).
15. **Implementing API Rate Limiting:** Use express-rate-limit to prevent excessive API requests from a single client.
16. **Validating Request Data Using Joi:** Ensure data integrity by validating API requests using the Joi library.
17. **Handling File Uploads in Express:** Build an API that allows users to upload profile pictures using multer.
18. **Using CORS in Express:** Enable CORS for your API and configure it to allow requests from specific domains.
19. **Adding API Versioning:** Implement versioning (/v1/ and /v2/) for an API to manage updates without breaking clients.
20. **Writing Unit Tests for an Express API:** Use Jest and Supertest to write test cases for API endpoints.
21. **Implementing Role-Based Access Control (RBAC):** Develop an API where users have different roles (admin, user) with restricted access to certain endpoints.
22. **Implementing WebSockets Alongside REST API:** Extend an Express API to support real-time notifications using socket.io.
23. **Building a Nested Resource API:** Develop an API where a user can have multiple orders, requiring proper relationships between endpoints (/users/:id/orders).
24. **Building a Complete RESTful Authentication System:** Implement user registration, login, password reset, email verification, and protected routes.
25. **Building a Scalable REST API with Microservices:** Design an Express-based REST API where different functionalities (auth, orders, users) are separated into microservices and communicate via an API gateway.