1. Role-Based Access Control (RBAC)

Implement **role-based authentication** in an API where:

- **Admin** users can create, update, delete any resource.
- **Regular users** can only read data.

2. REST API with MySQL and TypeORM

Build a **CRUD API** using **Express.js, MySQL, and TypeORM**. Create models for `users` and `orders` and expose endpoints to manage them.

3. Weather API Fetcher

Create an API that **fetches real-time weather data** from an external API (like OpenWeatherMap) and returns the result in a structured format.

4. File Upload API

Create an API that allows users to **upload images** using `multer`. Save the images in a local `uploads/` directory.

5. User Authentication System

Implement user authentication using **JWT (JSON Web Tokens)**. Create routes for:

- `POST /register` → Register a user
- `POST /login` → Login and get a token
- `GET /profile` → Get user details (protected route)

6. Employee Management API

Create an **Express.js API** that performs CRUD operations for employees, storing data in `employees.json` using the `fs` module.

7. Pipe Streams

Create a script that reads data from `input.txt` and **pipes it** to `output.txt`.

8. Simple REST API with Express

Build a REST API using Express that handles:

- `GET /users`
- `POST /users`
- `PUT /users/:id`
- `DELETE /users/:id`

9. Implement Middleware in Express

Write a middleware that logs each request's **method, URL, and timestamp**.

10. Promise.all vs Promise.any vs Promise.race

Write an example demonstrating:

- `Promise.all()`
- `Promise.any()`
- `Promise.race()` Compare their behavior.

11. Async/Await with Multiple API Calls

Fetch data from two APIs sequentially using `async/await`, then merge and return the results.

12. Retry Failed Promise

Write a function that **retries a failed promise** up to **3 times** before giving up.

13. Manipulate Buffers

- Create a buffer of **20 bytes**.
- Write `"Node.js Buffers"` into it.
- Convert it to a string and print it.

14. Query Parameters Validation

Create an API that only accepts requests with a **valid query parameter** (`/search?query=someValue`), otherwise returns a `400` error.

15. SQL Database Connection with Node.js

Use `mysql2` or `pg` to connect to **MySQL/PostgreSQL**, fetch data, and display it.

16. Throttling API Calls

Limit an API function to be called **only once per second**, even if called multiple times.

or

Order Discount API

Apply a 20% discount on total order value if the customer has more than 5 orders

17. Implement Role-Based Access Control (RBAC)

Secure an API with **Admin, User, and Guest roles** using **JWT and middleware**.

18. Implement a Custom `delay(ms)` Function Using Promises

19. Implement SQL Transactions Using Node.js

Write a **transaction-based query** in MySQL

1. Withdraws money from `account_A`
2. Deposits it into `account_B`
3. **Rolls back** if an error occurs