Name: Keo Hengneitong

Group: SE 04

IDTB100193

REFLECTIVE QUESTIONS – PERSONAL RESPONSES

Middleware & Architecture

1. **Advantages of Middleware in Express:**
   Middleware in Express has been a real game-changer for me. It lets me break down complex request-processing logic into smaller, manageable units. For example, I can use separate middleware for logging, authentication, and error handling. This modularity makes the application much easier to understand and debug.

2. **Why Use Dedicated Middleware Files:**
   Initially, I kept all my middleware in one file, and it quickly got messy. Moving each piece into its own file helped me stay organized and made the project more scalable. When I return to my code weeks later, it's much easier to pick up where I left off.

3. **Scaling for User Roles (Admin vs Student):**
   If I were to support multiple user roles, I'd probably create middleware that checks the user's role after authentication. I'd group routes based on roles and load the correct middleware for each. This would help ensure that, for instance, only admins can access admin-only routes without duplicating code.

Query Handling & Filtering

4. **Handling Conflicting Query Parameters:**
   In situations where `minCredits=4` and `maxCredits=3`, I would treat it as a logic error and return a clear message to the user. That kind of conflict often comes from typos or misunderstandings, so catching it early improves UX and prevents confusing results.

5. **Making Filtering User-Friendly:**
   I'd want to handle typos in a helpful way—maybe using fuzzy search or even a simple spell-checking library. If someone types "falll" instead of "fall," the system should still understand what they mean. It's all about forgiving user mistakes and improving accessibility.

Security & Validation

6. **Limitations of Query Parameters for Auth:**
   Using query parameters for authentication can be dangerous because URLs get logged everywhere—browser history, server logs, etc. A better approach would be bearer tokens in headers, or even better, cookies with HTTP-only flags for web-based apps.

7. **Why Validation & Sanitization Matter:**
   Not validating query parameters is like leaving your door wide open. Attackers can exploit it with injections or malformed input. It also helps maintain data integrity and makes the backend more resilient to unexpected input.

Abstraction & Reusability

8. **Reusable Middleware:**
   Yes, some middleware can be reused—like auth checks or error handlers. I'd extract these into their own modules or even create an NPM package with documentation, examples, and options for customization.

9. **Designing for Future Filters:**
   I'd use dynamic query parsing and a centralized filter handler that takes an object of filter types and values. Middleware could then be chained depending on what filters are present. That way, I can easily add new ones like course format or time slot later.

Bonus – Real-World Thinking

10. **API Under High Traffic:**
    Under high load, this API could slow down or crash without some improvements. I'd add rate limiting to avoid abuse, caching for frequent queries, and maybe even a reverse proxy like NGINX. Monitoring and logging would also help catch problems early.