Reflective Questions

Backend Development

Course: T3Y2: Backend Development
Lecturer: Kheang Kim Ang
Student's Name: Chea Ilong
ID: 100022
Group: 1 SE
Generation: 10

Middleware & Architecture

1. What are the advantages of using middleware in an Express application?

Answer: Middleware allows modular handling of requests, simplifies tasks like logging, authentication, and error handling, and improves code clarity and reusability.

2. How does separating middleware into dedicated files improve the maintainability of your code?

Answer: It keeps the main application file clean, promotes separation of concerns, and makes it easier to debug, test, and update individual components.

3. If you had to scale this API to support user roles (e.g., admin vs student), how would you modify the middleware structure?

Answer: I'd create role-based middleware that checks user roles and restricts access to specific routes.

Query Handling & Filtering

4. How would you handle cases where multiple query parameters conflict or are ambiguous (e.g., minCredits=4 and maxCredits=3)?

Answer: I'd add validation logic to detect conflicting parameters and return a clear error message, guiding the user to correct their input.

5. What would be a good strategy to make the course filtering more user-friendly (e.g., handling typos in query parameters like "falll" or "dr. smtih")?

Answer: Use a fuzzy matching library like Fuse.js to compare user input with valid values (e.g., semesters or instructor names). It can detect close matches even with typos, so if a user types "falll" or "dr. smtih", the system can still match it to "fall" or "Dr. Smith". This makes filtering more user-friendly by tolerating minor errors in query parameters.

Security & Validation

6. What are the limitations of using a query parameter for authentication (e.g., ?token=xyz123)? What alternatives would be more secure?

Answer: Query parameters are exposed in URLs and logs, making them insecure. Alternatives like HTTP headers (e.g., Authorization: Bearer ¡token¿) or cookie-based authentication are safer.

7. Why is it important to validate and sanitize query inputs before using them in your backend logic?

Answer: To prevent security issues like injection attacks and ensure the application behaves predictably with clean, correct input.

Abstraction & Reusability

8. Can any of the middleware you wrote be reused in other projects? If so, how would you package and document it?

Answer: Yes, common middleware like logging, error handling, or authentication can be reused. I would package them as NPM modules with clear documentation and usage examples.

9. How could you design your route and middleware system to support future filters (e.g., course format, time slot)?

Answer: I'd design modular middleware that reads supported filters from a config file or schema, allowing easy addition of new filters without changing core logic.

Bonus – Real-World Thinking

10. How would this API behave under high traffic? What improvements would you need to make for production readiness (e.g., rate limiting, caching)?

Answer: Under high traffic, the API could slow down or stop working if too many people use it at once. To fix that, I would add rate limiting to block users from sending too many requests in a short time. I would also use caching to store common results, turn on compression to make data smaller and faster, and add more servers with a load balancer to share the traffic.