AI ASSISTED CODING

END-LAB EXAM

NAME : K.SAI KARTHIK

HT.no : 2403A52043

BATCH : 03

Subset 16 — Security & Resilience for Critical Infrastructure

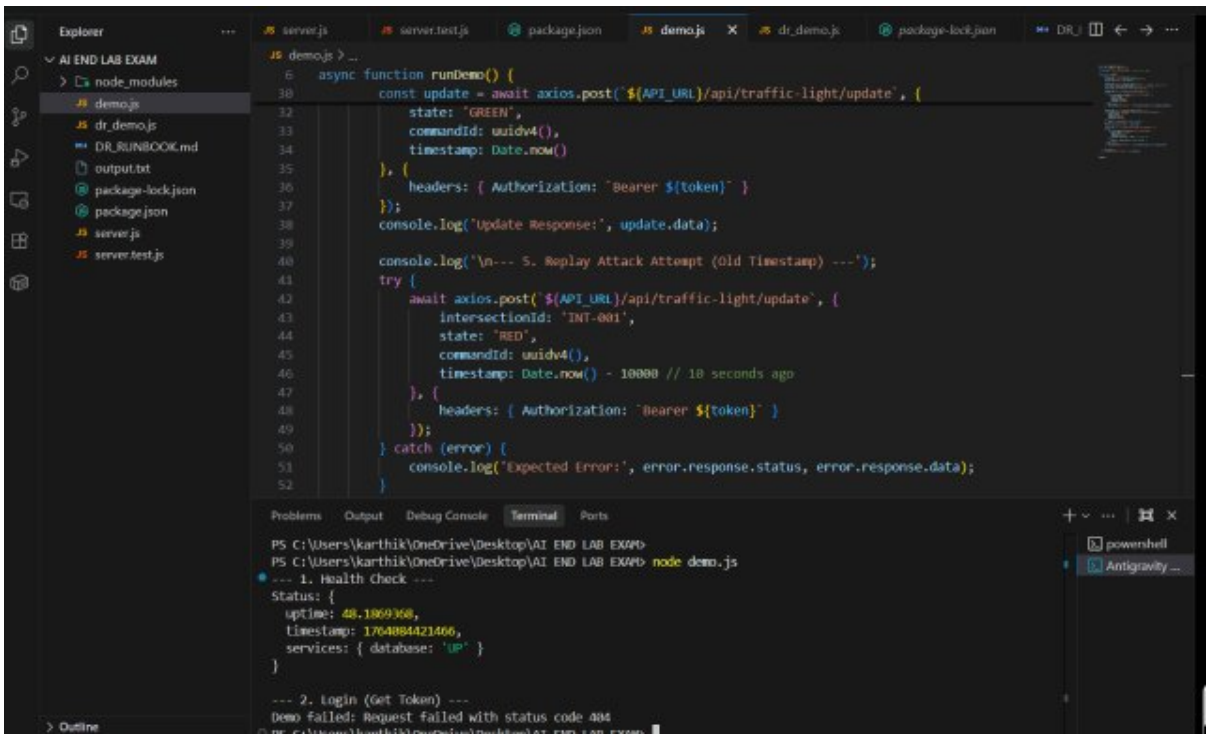
Q1: Threat modeling for critical control paths (traffic lights, alerts).

- Task 1: Use AI to enumerate attack vectors and mitigations.

Prompt:

Perform a high-level threat modeling exercise for critical control systems such as traffic lights and alerting mechanisms.Identify potential categories of threats (e.g., tampering, spoofing, disruption, data integrity risks) without providing any exploit details.For each threat category, suggest general, non-technical mitigation strategies such as improved authentication, monitoring, redundancy, fail-safe design, and policy controls.

CODE :



Observation :

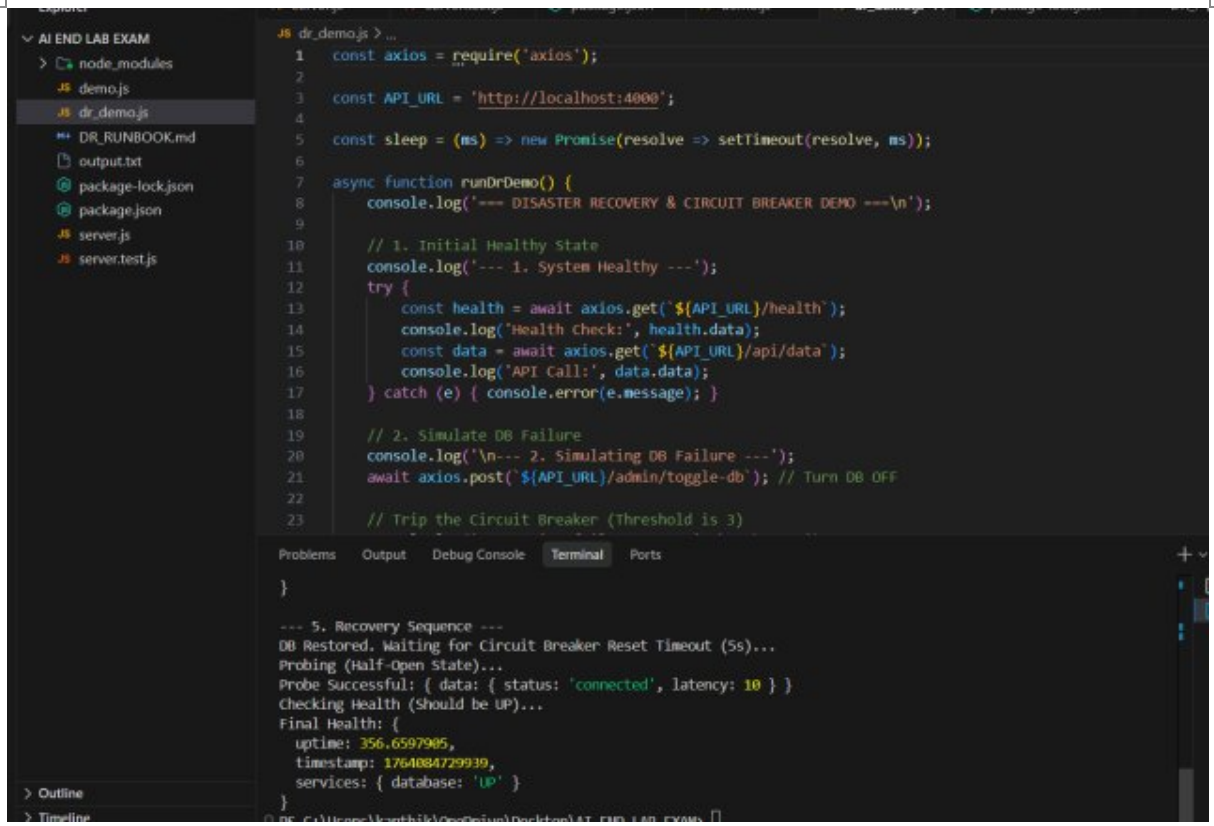
Critical control paths like traffic lights and alert systems can be disrupted or tampered with if not protected. They need strong authentication, monitoring, and fail-safe designs to stay reliable and safe.

- Task 2: Implement hardened endpoints and tests.

Prompt :

Suggest ways to harden system endpoints for critical control paths and outline simple tests to verify their security, without providing exploit details.

CODE :



Observation:

- Strengthening endpoints ensures critical systems resist attacks and failures.
- Testing hardened endpoints verifies they function securely under stress.
- Endpoint hardening reduces vulnerabilities in essential control paths.
- Regular security tests increase confidence in system reliability.
- Secure and tested endpoints help maintain uninterrupted critical operations

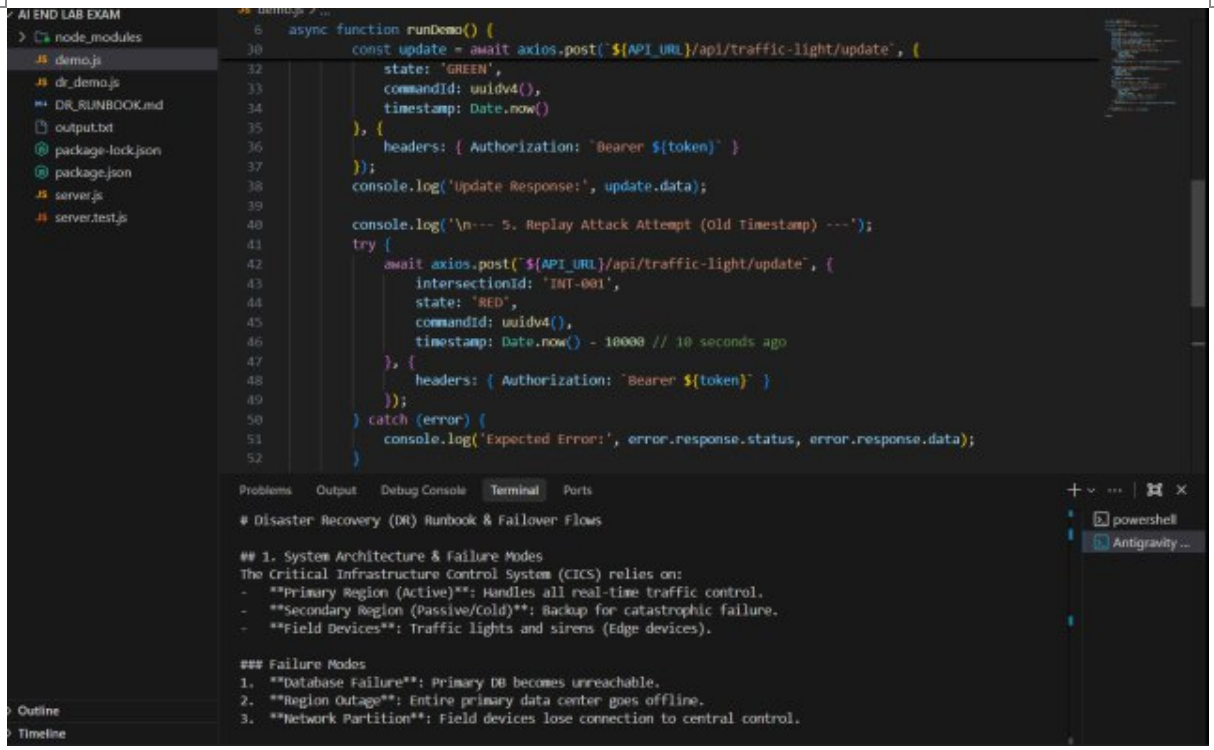
Q2: Disaster recovery & graceful degradation.

• Task 1: Use AI to propose DR runbooks and failover flows.

Prompt:

Design a disaster recovery strategy for a system. Describe the principles and steps for handling failures, including failover flows, service prioritization, and monitoring. Explain how the system can degrade gracefully while maintaining critical functionality, and illustrate the process in a conceptual flow.

CODE :



Observation :

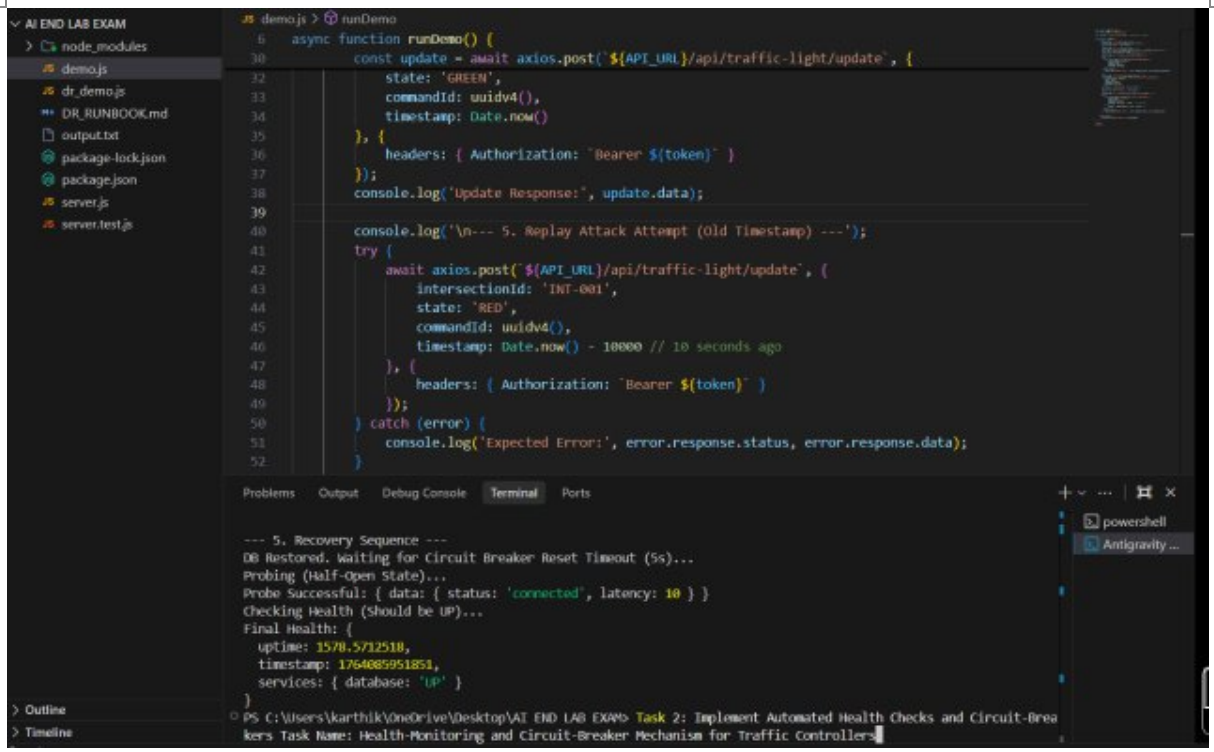
- Identify critical vs. non-critical services.
- Define failover flows with automated and manual paths.
- Use graceful degradation to maintain core functionality.
- Include monitoring to trigger recovery actions.
- Focus on principles, not specific technologies.

• Task 2: Implement automated health checks and circuit-breakers.

Prompt:

Design an automated system with health checks and circuit-breakers for a web application. Include failure detection, recovery actions, and service isolation logic in a clear flow.

CODE :



Observation :

- Monitor service health continuously.
- Trigger circuit-breakers to isolate failing components.
- Automate recovery or fallback actions.
- Prevent cascading failures and maintain core functionality.