Why did you choose your folder structure?

I used an MVC based folder structure to separate concerns clearly. Models handle database schemas, controllers contain business logic, routes define APIs, and middleware manages authentication. This structure improve readability, maintainability, and makes the codebase easy to scale.

How did you implement wallet atomicity to avoid race conditions?

Wallet operation follow a strict sequence: validate input, fetch wallet, check balance, update balance, and then create a transaction record. This ensures logical consistency. In production, MongoDB transactions or atomic update operators would be used to fully prevent race condition.

Explain the trade-offs in your indexing strategy.

I indexed only frequently queried and unique fields such as user email, wallet user_id, and voucherId. This improves read performance while avoiding excessive indexing, Which could slow write operation and increase storage cost.

How would you scale this system if user count reaches 10 million?

The system can be scaled using database sharding, caching with Redis, Microservices for wallet and auth, and horizontal scaling with load balancers.

What do you consider the riskiest part of your current design?

The design focuses on clean structure, controlled wallet logic, minimal indexing, and scalability readiness.