Interview Process (Stage II)

Backend Tech Interview Prep Guide (Node.js / Express.js)

1. Node.js Core and Runtime Fundamentals

- Event loop internals
- Concurrency model
- Worker Threads vs Child Processes
- Async patterns
- Debugging runtime issues

2. Express.js and HTTP Middleware

- How middleware stacks work
- Writing reusable middleware for:
- Handling async errors properly (next(err) or centralized error handler).
- File upload handling (multer, stream-based uploads).
- Performance optimizations: compression, caching headers, pagination.
- Middleware debugging.

3. APIs and Real-Time Systems

- REST API design: resource modeling, status codes, idempotency, pagination.
- WebSocket and SSE fundamentals:
- Long polling fallback strategies.
- Rate limiting and DoS prevention strategies.

- Handling live updates (e.g., stock, patient, or sensor data).
- Debugging focus: Watching for dropped connections and reconnect loops.
- Debugging focus: API testing

4. Data Consistency and Concurrency

- Database transactions and isolation levels (especially Postgres).
- Locking mechanisms
- Handling concurrent writes and inventory consistency.
- Implementing **transactional outbox** pattern for reliable event emission.
- Understanding idempotency and retry-safe design.
- Using ORMs (Prisma, TypeORM, etc) effectively in transactions.
- **Data Consistency and Concurrency** Debugging Techniques

5. Queues, Jobs, and Background Work

- Difference between:
    - Worker threads
    - Job queues
    - Microservices
- Implementing job queues (BullMQ, RabbitMQ, Cloud Tasks, SQS).
- Retry strategies, backoff, and failure alerts.
- Event-driven design (Pub/Sub, Kafka basics).
- Designing resilient background task processing.
- **Queues, Jobs, and Background** Debugging Techniques

6. Authentication, Authorization, and Security

- Auth flow fundamentals:

    - Session-based vs token-based.
    - Refresh-token storage
    - Secure practices
- Middleware for security
- Password storage best practices.
- Managing secrets (dotenv, environment separation, vaults).
- Content Security Policy (CSP), secure cookies, HTTPS, TLS basics.
- **Authentication, Authorization, and Security** Debugging Techniques

7. Performance, Scalability, and Observability

- Profiling and optimizing Node.js:
    - Avoiding blocking the event loop.
    - Efficient database queries.
- Horizontal vs vertical scaling.
- Load balancing and sticky sessions.
- Caching strategies
- Observability Techniques
- **Performance, Scalability, and Observability** Debugging Techniques

8. Testing and Debugging Practices

- Unit vs integration vs end-to-end testing.
- Tools: Jest, Supertest, Vitest.
- Mocking databases, APIs, and queues.
- Ensuring deterministic tests.
- Logging for debugging test failures.
- Using Postman collections and Newman for regression testing.

Debugging focus:

- Typical debugging checklist to follow when something fails?
- Intermittent test failures (timing, async issues).
- Flaky integration tests with async jobs or sockets.
- Detecting side effects due to shared state.

9. Database and ORM Usage

- Schema design: normalization, foreign keys, indexing.
- Query optimization and EXPLAIN plans.
- ORMs (Prisma/TypeORM):
    - Transactions and cascading deletes.
    - Lazy vs eager loading.
    - Handling migrations safely.
- Migrations and seeding strategies.

Debugging focus:

- Detecting slow queries.
- Handling database connection leaks.
- Understanding query logs and transaction boundaries.