Backend Developer Assessment

ASSESSMENT INSTRUCTIONS

Position: Full Stack Developer

- **Start Time: 26th December 2025, 10:30 AM (IST)**
- **Deadline: 28th December 2025, 10:30 AM (IST)**

STRICT 48-HOUR DEADLINE - NO EXTENSIONS GRANTED

Submission Requirements:

• **Format:** Word document (.docx) , Submission must be done via GitHub commit timestamp + shared deployment links before the deadline.

• **File Name:** Backend Developer_Assessment_[Your_Name] _December 2025

• **Email to:** [career@purplemerit.com] with subject "PM Assessment Submission - [Your Name]"

❖ LATE SUBMISSIONS ARE DISQUALIFIED.

ASSESSMENT OVERVIEW

This assignment evaluates your ability to **design and build scalable backend systems** with a strong focus on **API design, security, performance, and real-time communication**.

You are expected to approach this as a **production-grade backend service**, not a demo.

Problem Statement

Build a Real-Time Collaborative Workspace Backend

Design and implement a backend service that powers a **real-time collaborative workspace** for developers (similar to a simplified collaborative coding platform).

The system should support:

- Secure authentication
- Project & workspace management
- Real-time collaboration events
- Asynchronous job processing
- Cloud-ready, scalable architecture

Functional Requirements:

1. Authentication & Authorization

- JWT or OAuth2-based authentication
- Role-based access control:
    - Owner
    - Collaborator
    - Viewer
- Token refresh mechanism
- API rate limiting

2. Project & Workspace APIs

Implement RESTful APIs to:

- Create, update, delete projects
- Manage workspaces
- Invite collaborators
- Assign and update user roles

Expectations:

- API-first design
- Proper HTTP status codes
- OpenAPI (Swagger) documentation

3. Real-Time Collaboration

- WebSocket-based communication
- Broadcast events such as:
    - User join/leave

  - File change events (mocked payloads)
  - Activity or cursor updates
- Event distribution using:
  - Redis Pub/Sub **or**
  - Kafka / RabbitMQ / AWS SQS

4. Asynchronous Job Processing

Simulate a **code execution or background task system**:

- Accept job requests via API
- Push jobs to a message queue
- Background worker processes jobs
- Persist job results and status

Must include:

- Retry logic
- Failure handling
- Idempotent job processing

5. Data Storage

Use **multiple data stores**:

- Relational DB: PostgreSQL or MySQL
- Non-relational DB: MongoDB and/or Redis

Key considerations:

- Proper schema design
- Indexing
- Data integrity

Non-Functional Requirements

Performance & Scalability

- Redis caching
- Horizontally scalable design
- Async/non-blocking I/O where applicable

Testing

- Unit tests for core business logic
- Integration tests for APIs and auth flows
- Meaningful test coverage (≈70%)

Deployment & DevOps

- Dockerized services
- One of:
    - Docker Compose **or**
    - Kubernetes manifests
- CI/CD pipeline:
    - Lint
    - Test
    - Build

Security

- Input validation
- Protection against SQL/NoSQL injection
- Secure secrets via environment variables
- Proper CORS configuration
- Serverless function usage

- Observability (logging + metrics)
- Feature flags
- Clean Architecture / DDD patterns
- API versioning strategy

Submission Requirements

Deliverables

1. Git repository (public or private link)
2. README.md including:
    - Architecture overview (diagram optional)
    - Setup & run instructions
    - Design decisions and trade-offs
    - Scalability considerations
3. API documentation
4. Test instructions
5. Deployment instructions

Time Constraint & Submission

- **Time Limit:** 48 hours from the time you receive
- Submit:
    1. GitHub Repo Link
    2. Live Deployment Links
    3. Walkthrough Video Link
- Late submissions will be disqualified.

We look forward to reviewing your comprehensive approach. Best of luck!