Technical Assignment

Project Title: Mini SaaS / POS Backend System (API-First)

Submission Deadline: 17 January 2026, 10:00 AM

1. Problem Statement

You are required to design and develop a **Multi-Tenant POS / Inventory Management Backend System** using **Laravel**.

The system must be:

- API-first
- Secure
- Scalable
- Optimized for real-world, production-grade usage

Each business must operate as an **independent tenant**, and **strict data isolation between tenants is mandatory** at all levels of the system.

2. Authentication & Authorization

- Implement authentication using **Laravel Sanctum**
- Support the following user roles:
  - ❖ **Owner**
  - ❖ **Staff**
- Apply **role-based access control (RBAC)** using:
  - ❖ Laravel **Policies** or **Gates**
- Authorization logic **must not** be hard-coded inside controllers

3. Multi-Tenancy (Critical Requirement)

- Each tenant (business) must have **fully isolated data** for:
  - ❖ Products
  - ❖ Customers
  - ❖ Orders

- Tenant context must be resolved using the HTTP request header:
  - **X-Tenant-ID**
- Data isolation must be enforced across:
  - Database queries
  - Authorization checks
  - Business logic
- **Under no circumstances** should one tenant be able to access or infer another tenant's data

4. Inventory & Order Management

Product

Each product must contain the following attributes:

- Name
- SKU (**must be unique per tenant**)
- Price
- Stock quantity
- Low stock threshold

Order

- Orders may include **multiple products**
- Order creation must:
  - Accurately deduct stock
  - Prevent negative inventory
  - Use **database transactions**
- Supported order statuses:
  - Pending
  - Paid
  - Cancelled
- Cancelling an order must **correctly restore inventory stock**

5. Reporting Module

Implement the following reports:

- **Daily sales summary**
- **Top 5 selling products** (based on a selected date range)
- **Low stock report**

Reporting Requirements

- Queries must be optimized
- Avoid N+1 query issues
- Use eager loading
- Apply appropriate database indexing where required

6. Validation & Security

- Use **Form Request Validation** for all inputs
- Enforce authorization strictly via **Policies**
- Protect against:
  - Mass assignment vulnerabilities
  - Unauthorized access
- Implement **API rate limiting**
- Ensure secure error handling without exposing sensitive system details

7. Performance Considerations

- Use eager loading wherever applicable
- Optimize database queries
- Apply appropriate database indexes
- Clearly document all performance-related decisions in the **README**

8. API Design Standards

- Follow **RESTful API conventions**
- Maintain a **consistent JSON response structure**
- Use **Laravel API Resources**
- Implement **pagination** for list endpoints

9. Bonus (Optional – Not Mandatory)

Additional credit will be given for implementing any of the following:

- PHPUnit feature tests
- Docker-based development environment
- Swagger / OpenAPI documentation
- Background jobs for reporting or heavy operations

10. Submission Guidelines

Please submit the following:

- **GitHub repository link**
- **README.md**, including:
  - ❖ Project setup instructions
  - ❖ Architecture overview
  - ❖ Multi-tenancy strategy
  - ❖ Key design decisions and trade-offs
- Sample **Postman collection** or API usage examples
- **Short video demonstration** covering:
  - ❖ Overall system architecture
  - ❖ Tenant isolation strategy
  - ❖ Authentication and role-based access control
  - ❖ Inventory and order workflow (including stock handling)
  - ❖ Reporting features

Video Guidelines

- Duration: **5–10 minutes**
- Screen recording with voice explanation is preferred
- Video may be shared via:
    - Google Drive
    - YouTube (unlisted)
    - Similar platforms

11. Disqualification Criteria

Submissions may be rejected if:

- Tenant isolation is missing or incorrectly implemented
- Database transactions are not used for order-related operations
- Authorization logic is placed directly inside controllers
- Input validation is missing
- The solution is clearly copied from tutorials without meaningful customization

12. Evaluation Criteria

Your submission will be evaluated based on:

- Overall system architecture and code quality
- Multi-tenant data isolation
- Business logic correctness and transaction handling
- Security and performance considerations
- Code readability and documentation quality

Avanteca
Experience IT | Managed Services

Submission Instructions

Please submit your completed assignment by **replying to this email** with your **GitHub repository link** on or before:

17 January 2026, 10:00 AM

We appreciate the time and effort you invest in this assignment and look forward to reviewing your submission.

Best regards,
Avanteca Limited
Hiring Team
career@avantecatech.com