

# **Fidelity India Labs**

## **Python FastAPI Technical Assessment**



Date: 7th Oct 2025 | Duration: 60 Minutes

# Objective

---

## Purpose:

To evaluate your ability to design and implement a small backend system using:

- ▶ FastAPI + SQLAlchemy + Pydantic
- ▶ Authentication & Authorization (JWT)
- ▶ Transaction handling & AOP
- ▶ Async processing and testing

# The Scenario

---

You are building a **Mini Order Management Service** with:

- ▶ User login & role-based access
- ▶ Order creation with transaction safety
- ▶ Async notification after order success

# Key Functional Requirements

---

## ✓ Auth & Authorization

- JWT-based login & token validation
- Role-based access (Admin/User)

## ✓ Transaction Flow

- SQLAlchemy session management
- Simulate external "payment API"
- Rollback if payment fails
- Decorator-based logging

# Key Functional Requirements (Cont.)

---

## **Async Processing**

- Trigger async notification after order creation
- Write one test to validate async behaviour

# Tech & Tools

---

- **FastAPI** for API framework
- **SQLAlchemy ORM** for DB
- **Pydantic** for models
- **PyJWT / fastapi-jwt-auth** for tokens
- **pytest + HTTPX** for testing
- **(Optional) Testcontainers** for integration tests

# Evaluation Breakdown

---

Area	Weight
JWT Auth Flow	25
Transaction Mgmt	25
Exception & Decorators	15
Async Processing	20
Code Quality	15
Integration Test	+5 Bonus

# Submission Guidelines

---

## **Submit ZIP with:**

- Working code
- README.md explaining setup
- One test case file

## **Ensure app starts via:**

```
uvicorn main:app --reload
```



# Tips




---

- 💡 Use `Depends()` for JWT verification
- 💡 Use `@app.on_event("startup")` for DB setup
- 💡 Use FastAPI `BackgroundTasks` for async trigger
- 💡 Keep code modular & clean
- 💡 Include docstrings for clarity

## Wrap-Up

---

*"Think like a product engineer."*

- ▶  Correctness matters
- ▶  Code quality matters more
- ▶  Explainability matters most