# Campus Pay: Smart Card Café System

## Group Members:

- M. Hasham BSSE23053
- Muhammad Abdullah BSSE23087
- Mustansar Tanwir BSSE23027

## Prompts:

Design the backend workflow for a university café pos system where:

cafés register via a web portal, students place orders via a mobile app, rfid scan triggers order confirmation, Payment is deducted from a wallet balance.
Explain the backend services, APIs, database interactions, and real-time communication.

---

How should the backend handle an rfid scan event from a café pos device so that:

The student is identified, the pending order is fetched, a confirmation request is sent to the student app.
focus on API endpoints and event flow.

---

We are using Firebase for notifications.
explain how the backend should trigger a push notification to the student app after an rfid scan, and how the student's confirmation response should be handled securely.

---

Design backend logic to:

Check student wallet balance, deduct amount atomically, prevent double spending, handle failed or insufficient balance cases, assume PostgreSQL is used as the main database.

---

Explain how Redis can be used in this POS backend for:

Temporary order storage, session management, preventing duplicate rfid scans, include examples of data stored in Redis.

---

Design a PostgreSQL schema for this system including:

Users (students, café owners), cafés, orders, transactions, wallet balances
Make sure it supports scalability and data integrity.

Explain how Amazon S3 should be used in this system for:

static website files, menu images, receipts or logs, include recommended bucket policies.

---

Explain how to deploy a Python/Django backend on EC2 using:

Gunicorn as WSGI server, Nginx as reverse proxy, include port flow and request lifecycle.

---

What environment variables and secrets should be stored securely for this backend?
Include database credentials, Firebase keys, and Redis configuration, explain how IAM roles help here.

---

Explain how an Application Load Balancer (ALB) should be configured to route:

Student app API requests, café website traffic, to EC2 instances running the backend.

---

Design security group rules for:

ALB, EC2, RDS PostgreSQL, Redis
Ensure minimum exposure and proper access control.

---

Our architecture shows regions eu-north-1 and eu-south-1.
Explain why multiple regions might be used and how backend services should be deployed across them.

---

Explain how this backend can scale under heavy student usage using:

ALB, EC2 Auto Scaling, Redis caching
Focus on request flow.

---

Describe the complete backend flow from:
RFID scan → backend → Firebase notification → student confirmation → transaction completion
Include all components shown in the architecture diagram.

---

Explain how AWS Glue can be used to analyze:

Order history, peak usage times, café sales performance
describe how data moves from PostgreSQL/S3 to Glue.

---

Give a **step-by-step guide** to deploy this POS backend on AWS using:

EC2, ALB, RDS PostgreSQL, Redis, S3, IAM roles
Follow the provided architecture diagram exactly.

---

List common AWS deployment issues for this architecture (ALB not routing, Gunicorn errors, DB connection failures) and how to debug them.

---

What changes are required to make this backend production-ready?
Include:

HTTPS, Logging, Monitoring, Backup strategy for RDS.

---

How should the backend handle failures such as:

Student confirms order but payment fails, network disconnect during RFID scan, duplicate confirmation requests

---

Review our AWS architecture diagram and explain whether the backend flow, services, and deployment choices are correct for a university-scale POS system.