Campus Pay: Smart Card Cafe System

A Cloud-Based RFID-Enabled Point of Sale Platform on AWS

Authors:
M. Abdullah BSSE23087
M. Hasham BSSE23053
Mustansar Tanwir BSSE23027

Project Name:
Smart University Card Café POS System

Architecture:
AWS (EC2, Application Load Balancer, RDS PostgreSQL, Redis, S3, IAM, AWS Glue, Firebase)

Environment:
AWS Learner Lab

Date:
3rd January 2026

Contents

Abstract

This report presents the design, implementation, and deployment of a cloud-based Point of Sale (POS) system developed for university café environments. The system enables students to place orders via a mobile application, authenticate transactions using RFID-enabled student cards, and complete payments through a wallet-based system. Café vendors manage menus and orders through a centralized web portal.

The backend infrastructure is deployed on Amazon Web Services (AWS) using a scalable and secure architecture composed of EC2 instances, Application Load Balancer (ALB), Amazon RDS (PostgreSQL), Redis caching, and Amazon S3 for static assets. Real-time notifications are delivered using Firebase Cloud Messaging, while transactional integrity is ensured through atomic database operations. The proposed solution improves operational efficiency, reduces waiting times, and demonstrates the application of production-grade cloud architecture within an academic setting.

1. Introduction

1.1 Background & Motivation

University cafés often face challenges such as long queues, inefficient cash handling, and lack of centralized management. With increasing student populations and limited service windows, manual POS systems struggle to meet demand efficiently.

Advancements in cloud computing and mobile applications enable the development of scalable, real-time systems that streamline ordering and payment processes. This project leverages AWS cloud services to design a modern POS solution integrating RFID authentication, mobile ordering, and real-time notifications.

1.2 Problem Statement

The primary challenges addressed by this project include:

- Long waiting times due to manual ordering and payment
- Lack of centralized café management across campus
- Inefficient and insecure cash-based transactions
- No real-time order confirmation or status tracking
- Scalability issues during peak hours

1.3 Project Scope & Objectives

The objectives of this project are:

- Develop a centralized POS platform for multiple university cafés
- Enable students to order via a mobile application
- Integrate RFID-based authentication for order confirmation
- Implement a wallet-based digital payment system

- Deploy a scalable and secure backend on AWS
- Ensure real-time notifications and transactional integrity

2. System Architecture & Design

2.1 High-Level Architecture

The system follows a client–server architecture with clear separation between frontend interfaces, backend services, and data storage.

2.1.1 Frontend Layer

- Café Web Portal (menu management, order handling)
- Student Mobile Application (ordering, confirmation, wallet)

2.1.2 Backend Layer

- REST APIs hosted on EC2 instances
- Nginx as reverse proxy
- Gunicorn as WSGI application server

2.1.3 Infrastructure Layer (AWS)

- Application Load Balancer for traffic distribution
- Amazon RDS (PostgreSQL) for persistent storage
- Redis for caching and session management
- Amazon S3 for static assets
- Firebase for push notifications
- AWS Glue for analytics and reporting

2.2 Network Design (VPC & Load Balancing)

The backend is deployed within a secure AWS environment using an Application Load Balancer, EC2 instances, security groups, and IAM roles to ensure scalability, high availability, and fault tolerance.

2.3 Database Schema Design

The primary database is Amazon RDS (PostgreSQL).

2.3.1 Key Tables

- Users
- Cafes
- MenuItems
- Orders
- OrderItems
- Transactions
- Wallets

2.3.2 Design Considerations

- ACID compliance for financial transactions
- Foreign key relationships for data integrity
- Indexed columns for fast query performance

3. Implementation Details

3.1 Backend Logic (EC2, Nginx & Gunicorn)

Client requests are routed through the ALB to EC2 instances, where Nginx forwards requests to Gunicorn for application execution. Responses are returned to the client through the same path.

3.2 RFID & Order Confirmation Workflow

1. Student places order via mobile app
2. Order stored as pending in database
3. Student scans RFID card at café POS
4. Backend identifies student and pending order
5. Firebase sends confirmation notification
6. Student confirms order via app
7. Payment is processed if balance is sufficient

3.3 Payment Processing & Security

Atomic database transactions ensure wallet consistency, prevent double spending, and enforce authentication and authorization for all sensitive operations.

4. DevOps & Deployment

4.1 Deployment Strategy

The deployment includes IAM configuration, EC2 provisioning, ALB setup, RDS and Redis configuration, and application deployment with environment variables.

4.2 Static Assets & Storage

Amazon S3 is used to store static assets such as menu images, improving performance and reducing backend server load.

5. Testing & Observability

5.1 Testing Methodology

- API testing for order and payment flows
- RFID workflow validation
- End-to-end testing
- Failure case handling

5.2 Monitoring & Analytics

System metrics, logs, and analytics are monitored using AWS services and processed through AWS Glue for reporting.

6. Results & Discussion

The deployed system reduced order processing time, eliminated cash handling, improved user experience, and demonstrated scalable performance under concurrent usage.

7. Conclusion & Future Work

7.1 Conclusion

This project demonstrates a scalable and secure cloud-based POS system designed for university environments using AWS infrastructure.

7.2 Future Enhancements

- Auto Scaling Groups
- Advanced analytics dashboards
- ERP integration
- Offline RFID support
- Containerized deployment using Docker and ECS

Bibliography

[1] Amazon Web Services, *Amazon EC2 Documentation*, 2024.

[2] Amazon Web Services, *Application Load Balancer Guide*, 2024.

[3] Firebase, *Cloud Messaging Documentation*, 2024.

[4] PostgreSQL Global Development Group, *PostgreSQL Documentation*, 2024.

[5] Nginx, *Reverse Proxy Configuration Guide*, 2024.