Karachi Institute Of Economics & Technology (KIET)

Software Requirement Engineering (Lab)

Azain Rabbani(65842) SRE Lab (119005)

Lab Assignment 2

By Miss Jaweria

Question 01 — Sequence Diagram & Requirements (3 marks)

(a) Requirements for Process Payment

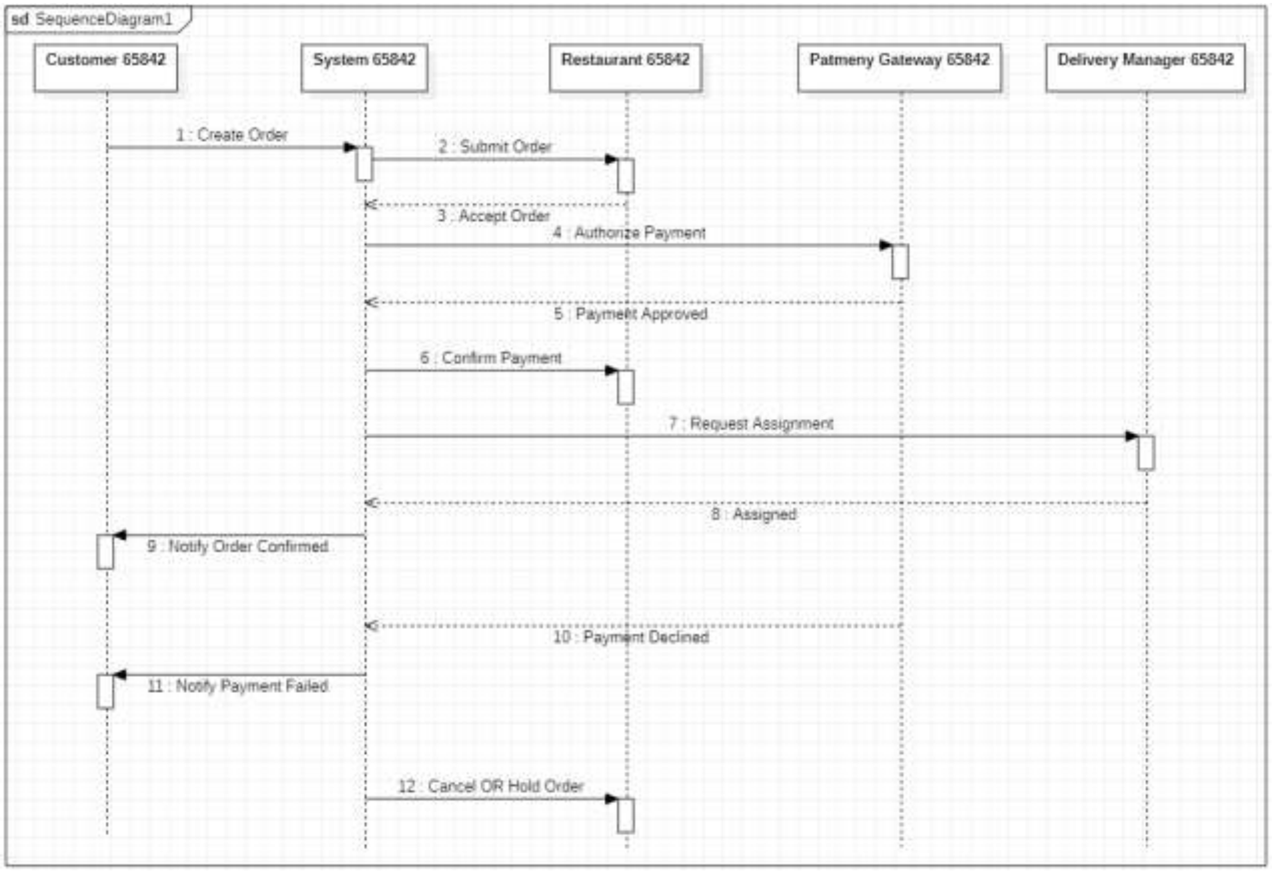
Functional Requirements (FRs)

1. **Initiate Payment:** System shall accept payment requests from customers for a given OrderID.
2. **Support Multiple Methods:** System shall support card (debit/credit), mobile wallet, and cash-on-delivery (COD) selection.
3. **Payment Gateway Integration:** System shall forward payment details to an external Payment Gateway for authorization.
4. **Confirm/Reject Payment:** System shall record and notify customer and restaurant of success or failure of the payment transaction.
5. **Refund Handling:** System shall support initiating refunds for cancelled orders or failed deliveries.

Non-Functional Requirements (NFRs)

1. **Security:** Payment data must be transmitted over TLS; card details must never be stored in plaintext and system must be PCI-DSS compliant.
2. **Availability:** Payment service should be highly available during peak hours (target 99.9% uptime).
3. **Performance:** Authorize/deny response from gateway should be returned within 3 seconds under normal conditions.
4. **Auditability:** Transactions should be auditable and retained according to retention policy (e.g., 7 years).
5. **Fault Tolerance:** System must handle gateway timeouts gracefully and retry or fall back to alternate payment paths.

(b) Sequence Diagram – Place Ordeer & Process Payment



Question 02 — Mini SRS (IEEE-like) (2 marks)

Software Requirements Specification (Mini)

Project: Online Food Delivery & Restaurant Management System
Author: Azain Rabbani (or your name)
Date: *(put submission date)*

1. Introduction

1.1 Purpose
This document provides a concise Software Requirements Specification for the Online Food Delivery & Restaurant Management System. It defines the system's main functions, constraints, and performance & security requirements.

1.2 Scope
The system enables customers to browse restaurants, place orders, pay online, track deliveries in real-time, and rate restaurants. Restaurants manage menus and process orders; delivery personnel update delivery statuses. The system will support concurrency, order history, and secure payments.

1.3 Definitions & Acronyms

- **Customer**: App user placing orders.
- **Restaurant**: Vendor managing menu & orders.
- **Delivery Personnel (Driver)**: Individual delivering orders.
- **Payment Gateway**: External payment processor.
- **Order**: Customer's placed purchase record.

2. Overall Description

2.1 Product Perspective
A multi-tenant web/mobile platform. Backend exposes REST APIs; frontend apps for Customer, Restaurant, and Driver interact with backend. Payment processing done via third-party gateway.

2.2 User Classes and Characteristics

- **End Customer**: registers, places orders.
- **Restaurant Admin**: manages menu, accepts orders.
- **Delivery Personnel**: receives assignments and updates status.
- **System Admin**: manages users, reports.

2.3 Operating Environment

- Cloud-hosted backend (Linux), HTTPS endpoints, mobile/web frontends.
- Database for users, orders, menus.

2.4 Constraints

- Must integrate with third-party payment gateway.
- Must follow local regulations for data retention & privacy.

3. Functional Requirements (summary)

(Select/numbering for SRS)

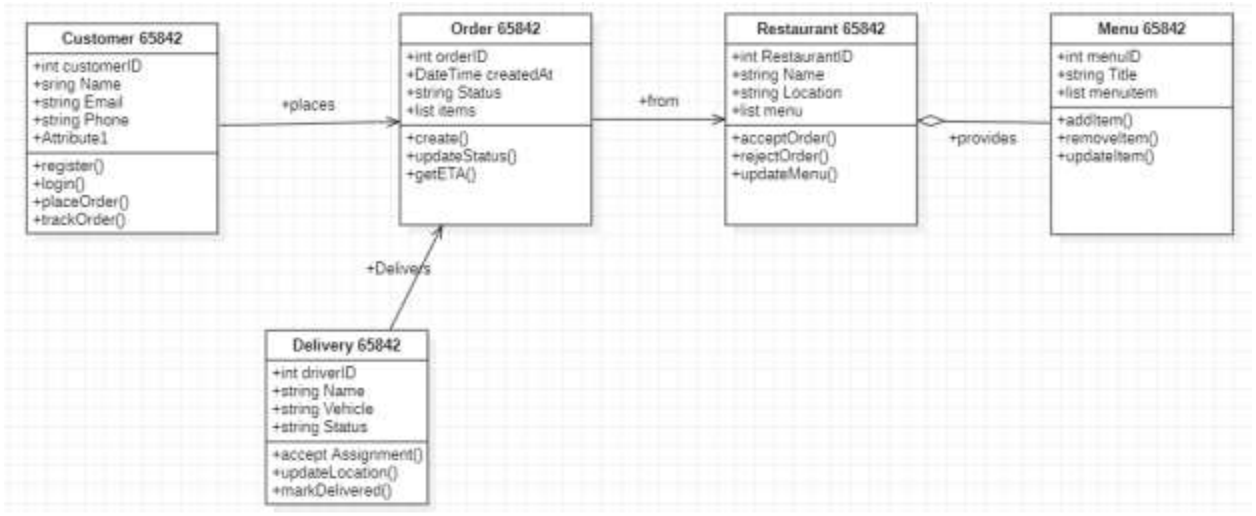
- **FR1 — User Authentication**: register/login, password reset.
- **FR2 — Restaurant & Menu Management**: create/edit menu items.
- **FR3 — Order Management**: create orders, accept/reject, update status.
- **FR4 — Payment Processing**: integrate with Payment Gateway, process authorizations/refunds.

- **FR5 — Delivery Management**: assign drivers, track location, update status.
- **FR6 — Notifications**: order and status notifications (push/SMS/email).
- **FR7 — Rating & Review**: customers rate orders and restaurants.
- **FR8 — Order History & Receipts**: view/download order receipts.

4. Non-Functional Requirements

- **NFR1 — Security**: TLS, hashed passwords, PCI compliance for payments.
- **NFR2 — Performance**: pages load < 2s, payment response < 3s.
- **NFR3 — Scalability**: support concurrent users and spikes.
- **NFR4 — Availability**: 99.9% uptime for ordering service.
- **NFR5 — Maintainability**: modular microservices & documented APIs.
- **NFR6 — Usability**: intuitive UI/UX for all user classes.

Question 03 — Class Diagram



Question 04 — Coding & Git Update

(a) Simple Console-based Python Program
(b) (Repo Link)

https://github.com/azainrabbani/SRE-LAB-Assignment-1-2

Question 05 — Security Analysis (1 mark)

(a) Three potential vulnerabilities

1. **Weak Authentication & Account Hijacking**
   - Attack: Credential stuffing, weak passwords, or missing MFA can allow unauthorized access to accounts.
2. **Payment Fraud / Man-in-the-Middle during Payment**
   - Attack: Intercepted or tampered communications with payment gateway, replay attacks, or fake confirmation pages.
3. **Sensitive Data Leakage**
   - Attack: Storing card numbers, user PII, or driver details in plaintext or exposing logs/API responses that leak data.

(b) Mitigations

1. **Mitigation for Authentication**
   - Enforce strong password policies, rate-limit login attempts, support MFA (OTP), and monitor suspicious logins.
2. **Mitigation for Payment Fraud**
   - Always use TLS for payments, integrate with a reputable PCI-compliant gateway, verify webhook signatures, do not store full card details; use tokenization.
3. **Mitigation for Data Leakage**
   - Encrypt PII at rest & in transit, mask sensitive data in logs, apply least privilege access controls, and perform regular audits & vulnerability scans.