CS5002NI Software Engineering

20% Group Coursework

Milestone number: Final Milestone

AY 2025-2026

Credit: 30

Group Name:			
SN	**Student Name**	**College ID**	**University ID**
1	**Abiram Bhatta (AI5)**	**NP01AI4A240144**	**24046558**
2	**Nayan Chhantyal (AI5)**	**NP01AI4A240111**	**24046601**
3	**Shlok Niroula (AI5)**	**NP01AI4A240932**	**24046644**
4	**Sudip Giri (AI5)**	**NP01AI4A240146**	**24046651**
5	**Yakendra Bam Rajawar (AI5)**	**NP01AI4A240127**	**24046664**

Milestone Due Date: 2026/01/19 (6:00pm)

Milestone Submission Date: 2025/01/19

DRIVE LINK FOR DRAW.IO FILES

https://drive.google.com/drive/folders/1K3VFSEodfeq0neEijmP7WBhf8JmxVtTR?usp=drive_link

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.

Table of Contents

Table of Figures

Table of Tables

1 Introduction

This report highlights the analysis and design of an Online Restaurant Management System for Gokyo Bistro, which is one of the most renowned restaurant outlets located in Jhamsikhel, Lalitpur. This task has been accomplished through group activity, which represents reality in terms of software development, in which collaboration and time management are critical.

Gokyo Bistro has been facing an increased demand for customers in recent times, which creates various problems in terms of running the business effectively, including poor table management, waiting times for customers to get seated, as well as accepting customer reservations as well as home delivery. In this respect, Gokyo Bistro needs an online system to manage customer reservations, accounts, orders, payments, reporting, as well as customer feedback.

The major aim of this project work is to utilize the Structured Analysis and Design technique proposed by Yourdon to analyze the existing flaws in the system and present a well-structured system solution. The report includes major software engineering deliverables such as the Business Case, Software Requirements Specifications (SRS), Data Flow Diagrams (DFD), Entity Relationship Diagram (ERD), Data Dictionary, Process Specifications, and Design Specifications. All the major components mentioned above showcase how the requirements of the system could be analyzed and designed in a well-structured way.

Along with group activities, each member of the group performs an individual analysis and design activity on a particular function of the system. This allows for contributions from group activity performance as well as individual performance. Overall, this report aims to design a practical and efficient system that supports Gokyo Bistro’s daily operations while applying basic software engineering principles.

Group Task

2 Business Case: Gokyo Bistro Online Management System

2.1 Executive Summary

This project initiative involves the development of a comprehensive web-based management system for Gokyo Bistro, located in Jhamsikhel, Lalitpur. As a popular dining destination offering indoor and outdoor options, the bistro is currently struggling to manage its manual operations amidst a significant surge in customer volume. This Business Case outlines the clear justification for transitioning to a Structured Software Engineering solution to optimize occupancy, manage home deliveries, and provide data-driven administrative oversight.

2.2 Problem Statement & Organizational Impact

Gokyo Bistro's current manual handling of table arrangements and order management has become a bottleneck for growth.

- **Revenue Leakage through Occupancy Limits:** Because customers cannot check availability or book in advance, many leave upon arrival due to the restaurant reaching maximum capacity.
- **Operational Strain from Delivery Demand:** There is a notable increase in home delivery requests that the current manual infrastructure cannot handle efficiently, leading to potential loss of market share to competitors.
- **Inconsistent Financial Oversight:** Without an automated system, generating financial reports and predicting future revenue trends is labor-intensive and prone to human error.

2.3 Proposed Solution & Value Delivery

The proposed system implements a Yourdon Structured Analysis approach to provide a cohesive digital ecosystem:

- **Guaranteed Table Reservations:** Visitors and Members can view live availability and secure bookings with a mandatory 1,200 NRP advance payment, ensuring committed revenue for the bistro.
- **Home Delivery Infrastructure:** A dedicated module for Members allows for streamlined order placement and secure online payment processing.

- **Predictive Administration:** The Admin portal will feature advanced financial reporting tools capable of predicting revenue for future years and recommending menu optimizations.

2.4 Financial Justification & Risk Mitigation

The system is designed to protect the bistro's profit margins through specific financial rules:

- **Cancellation Recovery:** A 500 NRP deduction fee for same-day cancellations ensures that the restaurant recovers costs for "no-shows" or last-minute changes.
- **Market Expansion:** By offering home delivery to Members, the bistro can generate revenue even when physical seating is at capacity.
- **Data-Driven Growth:** Management can move from reactive to proactive planning using the system's automated forecasting and feedback analysis tools.

2.5 Strategic Alignment & Expected Outcomes

The implementation of the Gokyo Bistro Online Management System serves as the foundation for the restaurant's digital transformation. By resolving current occupancy challenges and tapping into the specialized home delivery market, the system ensures that the restaurant remains a market leader in Lalitpur.

The anticipated outcomes include a measurable reduction in "no-show" losses, increased order accuracy for deliveries, and a higher level of customer retention through personalized member services. This project initiative provides the necessary framework to translate the Bistro’s organizational goals into a functional, data-driven reality.

3 Software Requirement Specification (SRS)

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document describes the functional and non-functional requirements for the Gokyo Bistro Online Management System. The system is designed to address operational challenges faced by Gokyo Bistro, including table management, order processing, and increased demand for home deliveries. It aims to provide an efficient online platform for table reservations, home delivery orders, menu management, and administrative reporting. The primary audience for this

document includes the development team, project stakeholders, testers, and maintainers.

1.2 Scope

The Gokyo Bistro Online Management System is an online platform that will manage customer interactions, internal operations, and administrative tasks related to table reservations, food ordering, menu management, financial reporting, and customer feedback.

The system will include the following major features:

- User authentication and management for Admin, Members, and Visitors.
- Online table reservation and cancellation with advance payment handling.
- Home delivery order placement and online payment processing (for Members).
- Menu management and promotion posting by the Admin.
- Financial report generation and revenue prediction for the Admin.
- Customer rating and feedback submission.

1.3 Definitions, Acronyms, and Abbreviations

SRS: Software Requirements Specification

Admin: Restaurant staff with full system management privileges.

Member: Registered, frequent customer of Gokyo Bistro.

Visitor: Unregistered user of the system.

Advance: The mandatory 1200 NPR payment required to secure a table reservation.

DFD: Data Flow Diagram

ERD: Entity Relationship Diagram

NPR: Nepalese Rupee

UI: User Interface

DBMS: Database Management System

1.4 Reference

- IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications

1.5 Overview

The rest of this document contains two main sections. Section 2 provides an overview of the system including product perspective, functions, user characteristics, constraints, and assumptions. Section 3 through 7 details the specific requirements including functional requirements, non-functional requirements, design constraints, and interface specifications.

2. Overall Description

2.1 Product Perspective

The Gokyo Bistro Restaurant Management System is a new, self-contained web-based application designed specifically to streamline Gokyo Bistro's restaurant operations. It replaces the current manual processes with an automated online solution. The system operates within the following environment:

- **Payment Gateway:** Interfaces with an external payment provider to handle secure online transactions for advance bookings and home deliveries.
- **Database Management System:** Connects to a backend database for the persistent storage of user profiles, menu items, orders, and reservation logs.
- **Web Browsers:** The system is accessed via standard web browsers, serving as the client-side environment for all users.
- **User Interfaces:** The system *provides* the primary interface for **Customers** and serves as the central administrative tool for **Admins**.

2.2 Product Functions

Major functions include:

- User authentication and authorization (Admin, Member, Visitor)
- Table reservation management with payment processing
- Reservation cancellation with refund calculation
- Beverage preference management

- Home delivery order placement and tracking
- Financial report generation with revenue prediction
- Bill and receipt generation
- Menu item management
- Customer rating and feedback collection
- Promotional offers and discount scheme management

2.3 User Classes and Characteristics

The system supports three user types:

Visitors: casual users; can browse menu, check table availability, make reservations without registering, and rate experience.

Members: registered users; additional privileges: store profile, place home-delivery orders, save beverage preferences, view order history.

Admin: restaurant staff; full control over menu, offers, billing, report generation, and user management.

2.4 General Constraints

Financial Constraint: A mandatory advance payment of 1200 NPR is required for all table bookings.

Cancellation Constraint: A cancellation fee of 500 NPR must be deducted from the advance payment if a reservation is canceled on the very day of the booking.

Access Constraint: Home delivery ordering is restricted to **Members** only.

Technology Constraint: The system must be a web-based application accessible via standard web browsers.

2.5 Assumptions and Dependencies

Assumptions:

- Users have access to internet-enabled devices

- Payment gateway services are available and reliable

- Restaurant has stable internet connectivity

Dependency:

- The accuracy of financial reports and revenue predictions depends on the quality and completeness of the data recorded in the system.

- Third-party payment gateway availability

- Database server uptime and reliability

- Web hosting infrastructure

3. Specific Requirements

3.1 External Interfaces

3.1.1 User Interfaces

Public web pages: Home, Menu, Offers, Reservation, Login/Register, Contact, Feedback.

Reservation page: calendar/time slot selector, table availability grid, beverage choices, payment widget for NPR 1200.

Admin portal: dashboard, menu management, offer management, reservations list, order management, report generator.

Member area: profile, reservation history, order history, saved beverage preferences.

3.1.2 Hardware Interfaces

The system shall not require any specialized hardware beyond standard server infrastructure and client devices capable of running a modern web browser as well as a thermal printer to print the receipts.

3.1.3 Software Interfaces

Payment Gateway: The system shall integrate with a secure third-party payment gateway to process advance payments for bookings and online payments for home delivery orders.

Database Management System: The system shall interface with a robust DBMS to store all operational and user data.

Predictive Analytics Tool: The system utilize a data processing module/library to analyze historical financial records, enabling the admin to predict revenue for future years.

3.1.4 Communications Interfaces

The system should use standard internet protocols (TCP/IP, HTTPS) for all communication. All sensitive data, including login credentials and payment information, must be transmitted over a secure, encrypted connection.

3.2 Functional Requirements

3.2.1 User Authentication and Management

- FR 1.1 The system shall allow Admin and Member users to log in using their credentials.
- FR 1.2 The system shall allow Visitors to register as Members by providing name, email, phone number, and password.
- FR 1.3 The system shall enforce role-based access control, restricting Admin-only functions to logged-in Admin users.

3.2.2 Table Reservation Management

- FR 2.1 The system shall display the availability of tables for a specified date and time.
- FR 2.2 The system shall allow users to reserve a table for a selected date and time.
- FR 2.3 The system requires a mandatory advance payment of 1200 NPR to confirm a reservation.
- FR 2.4 The system shall allow Visitor and Member users to cancel a reservation.
- FR 2.5 If a reservation is canceled before the final day, the system shall process a full refund of the advance payment.
- FR 2.6 If a reservation is canceled on the very day of the booking, the system shall deduct a 500 NPR fee from the advance payment before processing the refund.

3.2.3 Order and Menu Management

- FR 3.1 Admins shall be able to add new menu items with name, description, price, category, and availability.

3.2.4 Offers and Scheme Management

- FR 4.1 Admin shall be able to create, update, and remove offers and discount schemes.
- FR 4.2 Offers may be time-limited, conditional, or targeted.

3.2.5 Home Delivery Order Management

- FR 5.1 The system shall allow Members to place an order for home delivery.
- FR 5.2 The system shall provide an option for online payment for home delivery orders.

- FR 6.1 Users shall be able to select expected beverage choices from available list during or after booking.
- FR 6.2 Members shall have additional option to reserve specific wine/whiskey choices.

3.2.7 Billing and Receipt Generation

- FR 7.1 The system shall generate bills for orders and reservations.
- FR 7.2 Admin shall be able to generate/print bills and receipts.
- FR 7.3 The system shall store transaction records with unique invoice IDs.

3.2.8 Rating and Feedback Management

- FR 8.1 Visitors and Members shall be able to rate food, staff behavior, ambience on a scale and provide textual feedback.
- FR 8.2 The system shall allow Admin to review ratings and feedback.

3.2.9 Reporting and Revenue Prediction

- FR 9.1 Admin shall be able to generate financial reports.
- FR 9.2 The system shall provide simple revenue forecasting for the next 12 months based on historical sales.
- FR 9.3 The system shall recommend top-performing and under-performing menu items based on sales and ratings.

3.2.10 Payment Processing

- FR 10.1 The system shall process online payments for reservations and home delivery orders.
- FR 10.2 The system shall integrate with payment gateway for secure transactions.
- FR 10.3 The system shall handle payment confirmations and failures.
- FR 10.4 The system shall process refunds according to cancellation policies.

3.3 Non-Functional Requirements

3.3.1 Performance Requirements

- NFR 1.1 The system shall load all pages and process all transactions) within 3 seconds under normal load.
- NFR 1.2 The system shall be able to support a minimum of 50 concurrent users without degradation in performance.
- NFR 1.3 The system shall be capable of processing a minimum of 50 reservations per hour during peak times.
- NFR 1.4 Reservation transactions including payment processing shall complete end-to-end within 10 seconds subject to payment gateway latency.

3.3.2 Security

- NFR 2.1 All data in transit must use standard internet protocols.
- NFR 2.2 Sensitive payment data must not be stored; use payment tokenization and use encryptions.
- NFR 2.3 Passwords must be stored hashed with a modern algorithm.
- NFR 2.4 Implement role-based access control for Admin functions.

- NFR 2.5 The system shall protect against common web vulnerabilities.

3.3.3 Reliability

- NFR 3.1 Availability: 99% uptime.
- NFR 3.2 Data backup mechanisms.

3.3.4 Usability

- NFR 4.1 The system shall be usable by non-technical users; average user should complete a reservation in under 3 minutes.
- NFR5 4.2 The system design shall ensure that an average user can complete a table reservation in under 3 minutes.

3.3.5 Maintainability

- NFR 5.1 The system code shall be modular and well-documented to facilitate future updates and maintenance.
- NFR 5.2 The system shall provide configuration options for business rules without requiring code changes.

3.3.6 Portability

- NFR 6.1 The backend should be deployable on common cloud providers.
- NFR 6.2 The frontend must support major browsers and mobile screen sizes.

3.3.7 Legal & Regulatory

- NFR 7.1 The system shall comply with local consumer protection and e-commerce laws.
- NFR 7.2 Privacy policy must be available, and users must consent to collection of personal.

3.4 Design Constraints

- **Development Language:** The system must be developed using modern, maintainable, and scalable technologies.
- **Database:** The database design must support high concurrency and data integrity for reservation and order transactions.

4 Context Level Diagram (DFD Level 0)

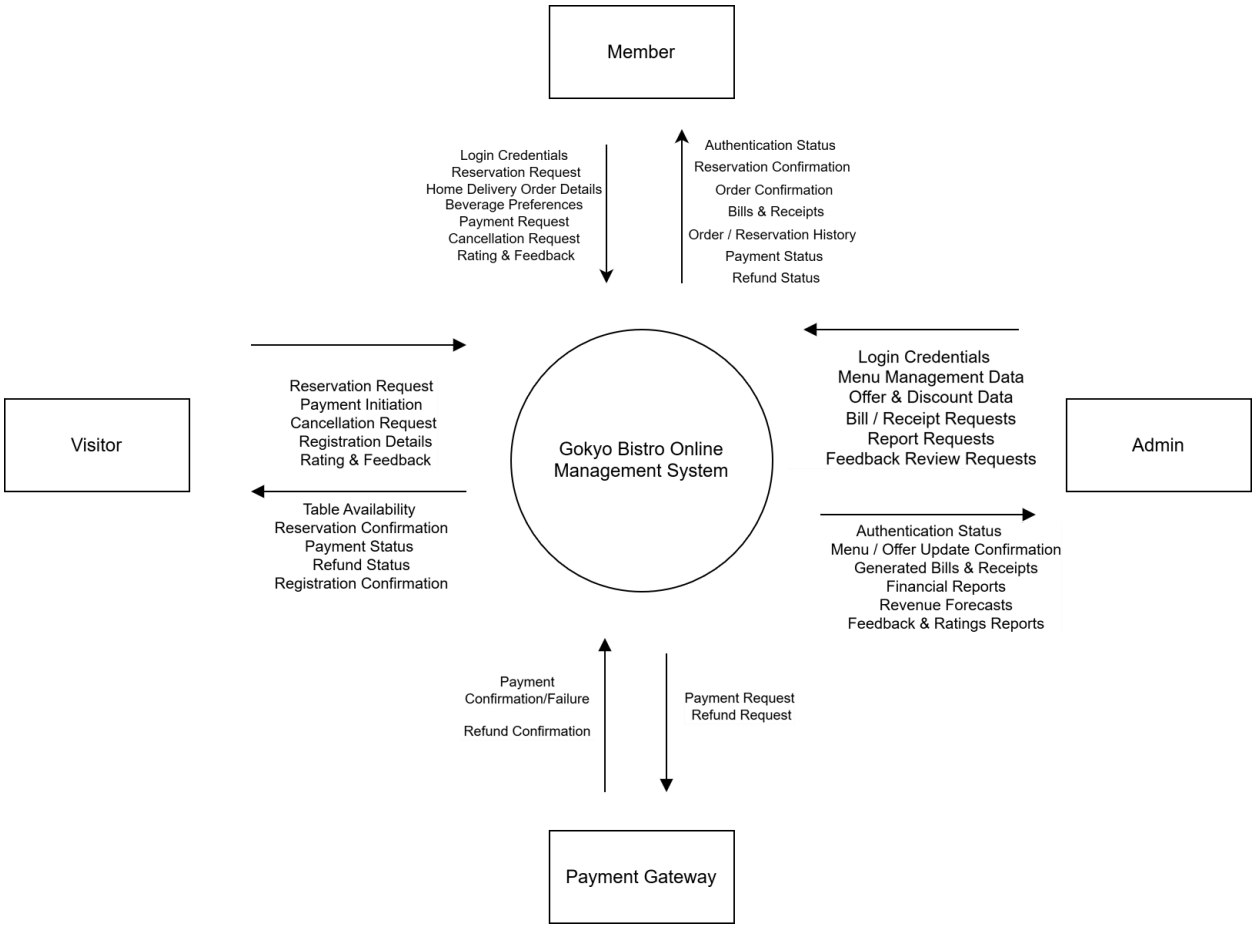

Figure 1: Context Level Diagram

5 Data Flow diagram (DFD)

5.1 DFD Level 1

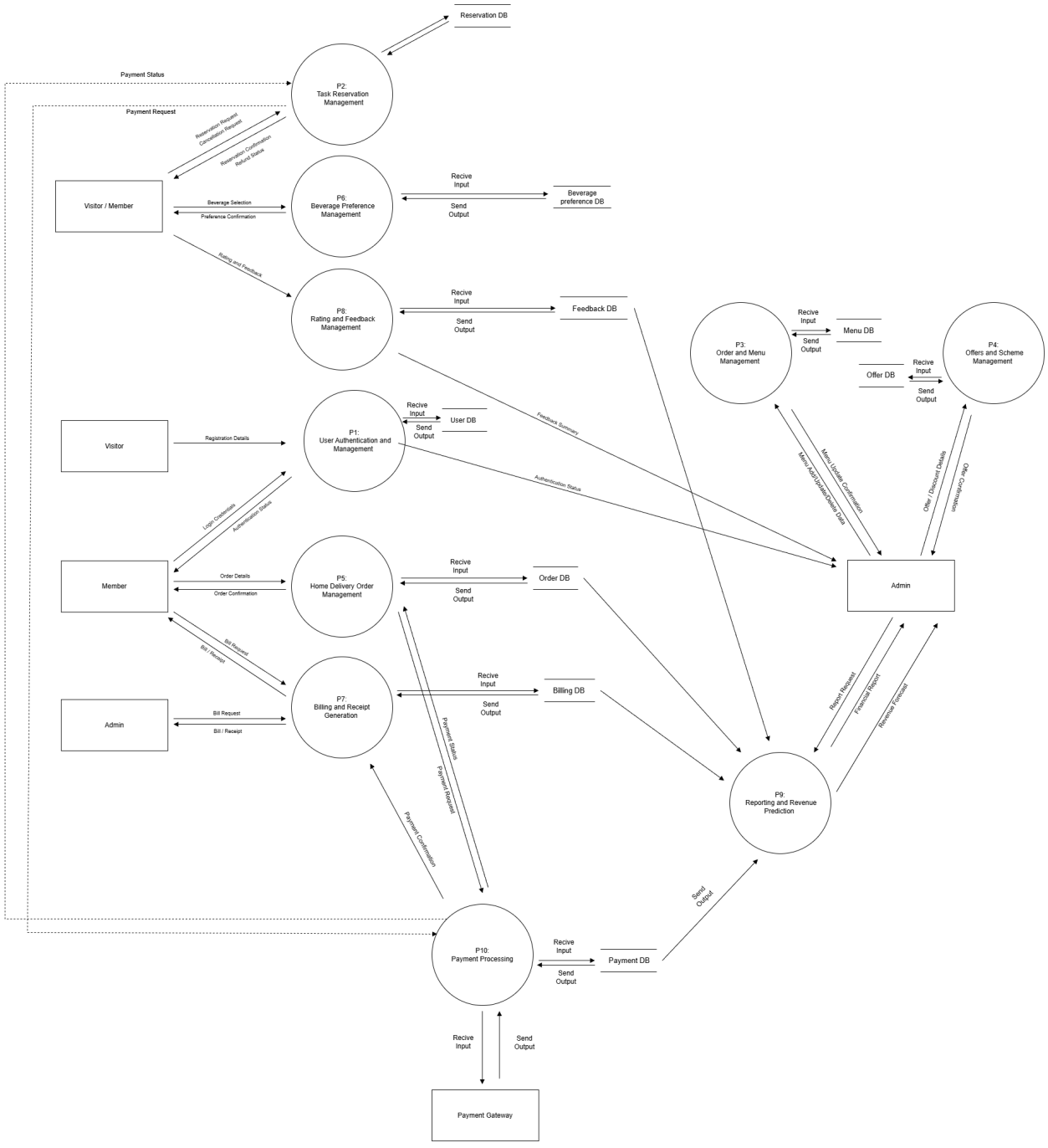

Figure 2: DFD Level 2

5.2 DFD Level 2

5.2.1 User Authentication and Authorization

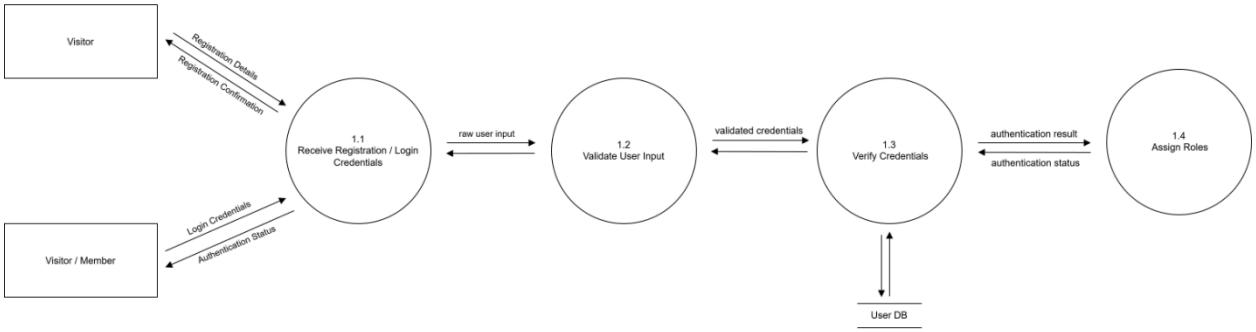

Figure 3: DFD level 2 for User Authentication and Authorization

5.2.2 Reporting and Revenue Prediction

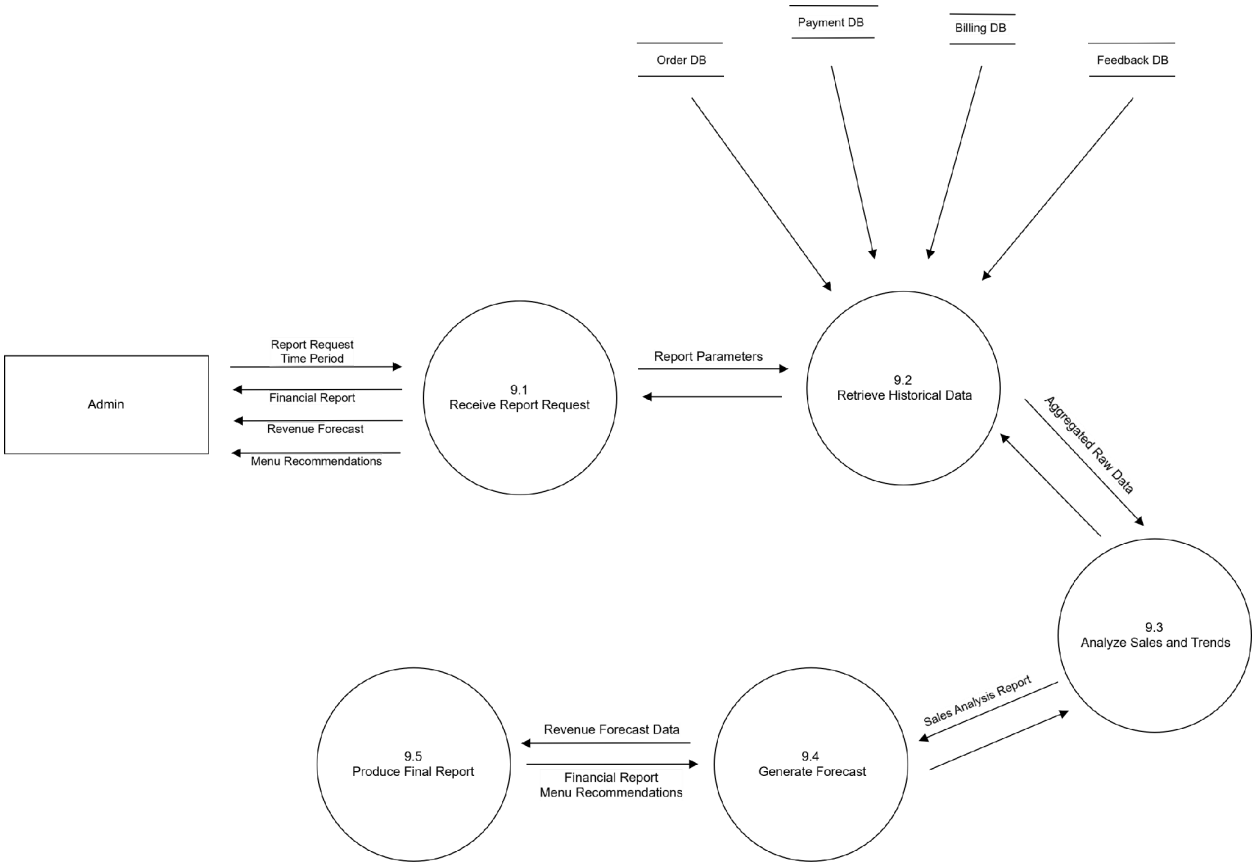

Figure 4: DFD Level 2 for Reporting and Revenue Prediction

5.2.3 Payment Processing

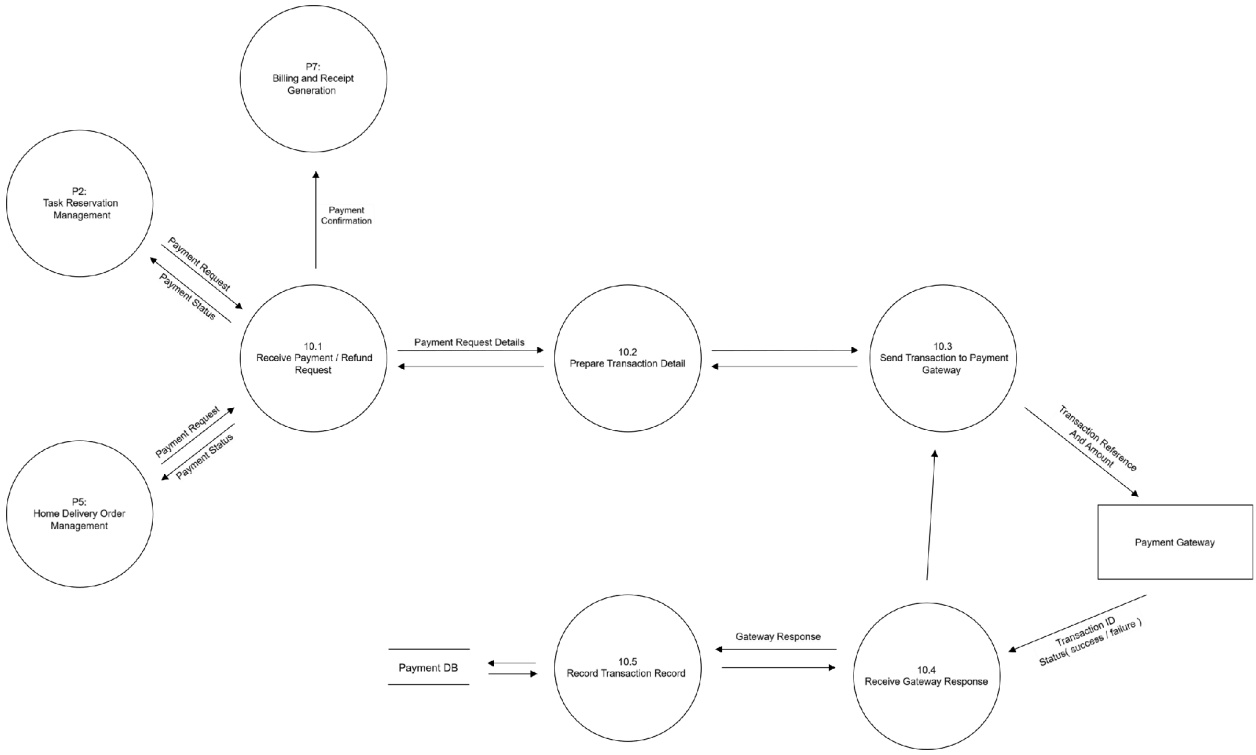

Figure 5: DFD Level 2 for Payment Processing

6 Entity Relationship Diagram

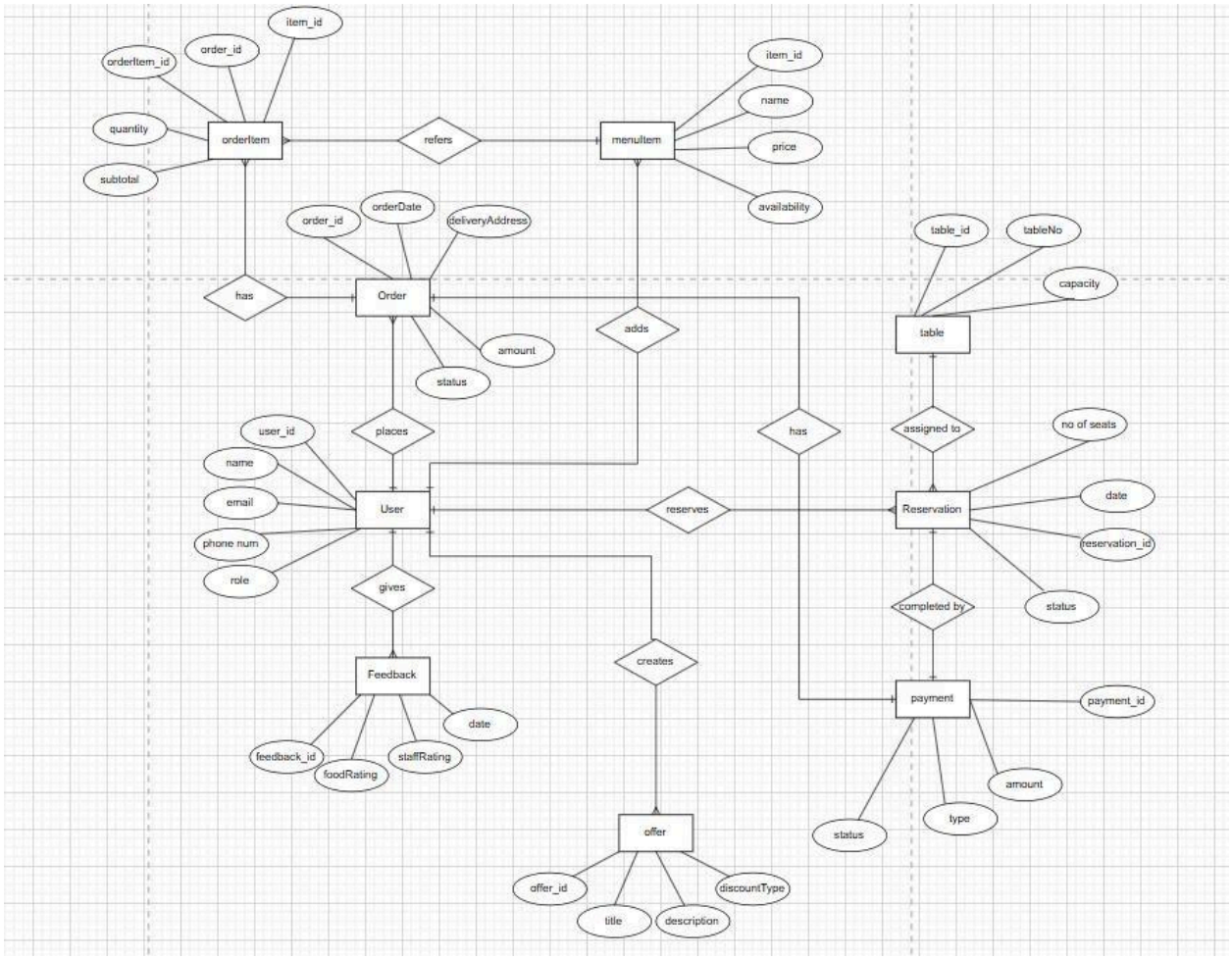

Figure 6: Entity Relationship Diagram

7 Data Dictionary

7.1 Data Flows

- **Login Credentials**:
  - Description: User-provided email/username and password for authentication
  - email + password
  - Source → Destination: User → Login Process

- **Registration Details**:
  - Description: Information provided by Visitor to become a Member
  - name + email + phone_number + password + address
  - Source → Destination: Visitor → Registration Process

- **Table Availability Request**:
  - Description: Request to check available tables for a specific date/time
  - date + time + number_of_seats
  - Source → Destination: User → Check Availability Process

- **Table Availability Response**:
  - Description: List of available tables returned to user
  - list of (table_id + table_type + capacity + status)
  - Source → Destination: Check Availability Process → User

- **Reservation Request**:
  - Description: Details for creating a new table reservation
  - user_id (or visitor details) + date + time + number_of_seats + beverage_preferences (optional) + advance_payment_details
  - Source → Destination: User → Book Table Process

- **Payment Details**:
  - Description: Information sent to/received from payment gateway
  - amount + transaction_id + payment_method + status
  - Source → Destination: Book Table / Place Order → Payment Gateway

- **Payment Confirmation**:
  - Description: Confirmation of successful payment
  - transaction_id + status + amount + timestamp

    - Source → Destination: Payment Gateway → System
- **Cancellation Request**:
    - Description: Request to cancel an existing reservation
    - reservation_id + cancellation_reason (optional)
    - Source → Destination: User → Cancel Reservation Process

- **Refund Details**:
    - Description: Calculated refund amount and transaction details
    - reservation_id + original_amount + deduction (0 or 500) + refund_amount + refund_transaction_id
    - Source → Destination: Cancel Reservation → Payment Gateway / User

- **Beverage Preferences**:
    - Description: Selected beverages
    - list of (beverage_id + quantity)
    - Source → Destination: User → Set Beverage Choices Process

- **Order Items Details**:
    - Description: List of items selected by the Member for a home delivery order
    - list of (item_id + quantity + unit_price + subtotal)
    - Source → Destination: Member → Place Order Process

- **Delivery Order Request**:
    - Description: Order placed by Member for home delivery
    - user_id + delivery_address + order_items + total_amount + payment_details
    - Source → Destination: Member → Place Order Process

- **Confirmed Order Items**:
    - Description: Stored details of items in a confirmed order
    - list of (order_item_id + order_id + item_id + quantity + price_at_time + subtotal)
    - Source → Destination: Place Order Process → Orders Store / Member

- **Order Confirmation**:
    - Description: Confirmation of placed order

    - order_id + order_items + total_amount + estimated_delivery_time + status
    - Source → Destination: Place Order Process → Member

- **Menu Item Details**:
    - Description: Information about a menu item
    - item_id + name + description + price + category + image_url + availability
    - Source → Destination: Admin → Update Menu Process / System → User

- **Offer Details**:
    - Description: Special offers or discount schemes
    - offer_id + title + description + discount_type + value + start_date + end_date + conditions
    - Source → Destination: Admin → Post Offers Process / System → User

- **Rating & Feedback**:
    - Description: Customer rating and comments
    - reservation_id/order_id + food_rating + staff_rating + ambience_rating + feedback_text
    - Source → Destination: User → Rate Experience Process

- **Financial Report Request**:
    - Description: Admin request for financial data or prediction
    - report_type + date_range
    - Source → Destination: Admin → Generate Report Process

- **Financial Report**:
    - Description: Generated report data
    - revenue_summary + expense_summary + predicted_revenue + menu_recommendations
    - Source → Destination: Generate Report Process → Admin

- **Bill/Receipt Data**:
    - Description: Details for generating bill or receipt
    - transaction_id + items + subtotal + taxes + total + payment_status
    - Source → Destination: System → Bill Generation Process

- **Payment Request**:

- Description: Request to process payment for reservation or order
- user_id + amount + payment_type (reservation/order) +
- reference_id (reservation_id/order_id)
- Source → Destination: Table Reservation/Home Delivery Order → Payment Processing

- **Payment Gateway Request**:
  - Description: Transaction details sent to external payment gateway
  - transaction_id + amount + payment_method +
  - merchant_details + callback_url
  - Source → Destination: Payment Processing → Payment Gateway

- **Payment Gateway Response**:
  - Description: Response from payment gateway
  - transaction_id + status (success/failure) +
  - gateway_reference + timestamp + error_message (if failed)
  - Source → Destination: Payment Gateway → Payment Processing

- **Refund Request**:
  - Description: Request to process refund for canceled reservation
  - reservation_id + original_payment_id + refund_amount + reason
  - Source → Destination: Cancel Reservation → Payment Processing

7.2 Data Stores

- **Users DB**:
  - Description: Stores information about all system users (Admin, Members, Visitors details if captured)
  - Contents (Key Fields): user_id (PK) + name + email + phone + password_hash + role + address + registration_date
  - Associated Processes: Registration, Login, Profile Management

- **Tables DB**:

- Description: Information about restaurant tables
- Contents (Key Fields): table_id (PK) + table_number + capacity + type (indoor/outdoor) + status
- Associated Processes: Check Availability, Book Table

- **Reservations DB**:
  - Description: All table reservation records
  - Contents (Key Fields): reservation_id (PK) + user_id (FK) + table_id (FK) + date + time + number_of_seats + status + advance_payment_id + beverage_preferences
  - Associated Processes: Book Table, Cancel Reservation, Admin View

- **Menu Items DB**:
  - Description: Catalog of all food and beverage items
  - Contents (Key Fields): item_id (PK) + name + description + price + category + availability + image_url
  - Associated Processes: Update Menu, View Menu, Ordering

- **Orders DB**:
  - Description: Home delivery orders placed by Members
  - Contents (Key Fields): order_id (PK) + user_id (FK) + order_date + delivery_address + items + total_amount + payment_id + status + estimated_delivery
  - Associated Processes: Place Order, Admin Order Management

- **Payments DB**:
  - Description: Transaction records for all payments and refunds
  - Contents (Key Fields): payment_id (PK) + transaction_id + amount + type (reservation/order/refund) + status + timestamp + user_id + gateway_reference

- Associated Processes: Payment Processing, Table Reservation, Home Delivery Order, Billing

- **Offers DB**:
  - Description: Active and past promotional offers
  - Contents (Key Fields): offer_id (PK) + title + description + discount_type + value + start_date + end_date
  - Associated Processes: Post Offers, View Offers

- **Ratings & Feedback DB**:
  - Description: Customer ratings and feedback
  - Contents (Key Fields): feedback_id (PK) + user_id (FK) + reservation_id/order_id (FK) + food_rating + staff_rating + ambience_rating + feedback_text + date
  - Associated Processes: Rate Experience, Admin Report

- **Financial Records DB**:
  - Description: Transaction logs used for reporting and prediction
  - Contents (Key Fields): record_id (PK) + date + revenue + expenses + order_id/reservation_id references
  - Associated Processes: Generate Report, Revenue Prediction

- **OrderItems DB**:
  - Description: Line items belonging to a home delivery order
  - Contents (Key Fields): order_item_id (PK) + order_id (FK) + item_id (FK) + quantity + price_at_time + subtotal
  - Associated Processes: Place Order, Admin Order Management, Reporting

- **Beverage Preferences DB**:
    - Description: Stores beverage selections for reservations
    - Contents (Key Fields): preference_id (PK) + reservation_id (FK) + beverage_id (FK) + quantity + special_requests
    - Associated Processes: Beverage Preference Management, Table Reservation

7.3 System Constants and Business Rules

- Mandatory Advance Payment:
    - Description: The fixed amount required to secure any table reservation.
    - Value: 1200 NPR
    - Related Data Element(s): advance_payment_details, advance_amount
    - Source Requirement: FR 2.3

- Same-Day Cancellation Fee:
    - Description: The fee deducted if a reservation is canceled on the day of the booking.
    - Value: 500 NPR
    - Related Data Element(s): deduction, refund_amount
    - Source Requirement: FR 2.6

- Home Delivery Access:
    - Description: Restriction on who can place a home delivery order.
    - Value: Restricted to Members only
    - Related Data Element(s): user_id (role must be Member), order_id
    - Source Requirement: Access Constraint

- Reservation Refund:
    - Description: Rule for processing refunds upon cancellation.
    - Value: Full refund if canceled before the day of booking; Refund = Advance - 500 NPR if canceled on the day
    - Related Data Element(s): refund_amount, original_amount, deduction
    - Source Requirement: FR 2.5, FR 2.6

- User Registration:
    - Description: Required fields for a Visitor to register as a Member.
    - Value: Name, Email, Phone Number, and Password are mandatory
    - Related Data Element(s): name, email, phone_number, password
    - Source Requirement: FR 1.2

8 Process specifications

8.1 Process: User Authentication and Management

Process Number: 1.1

Process Name: Receive Registration / Login Credentials

Input Data Flows: Registration Details (from Visitor), Login Credentials (from Visitor/Member)

Output Data Flows: raw user input (to 1.2), Registration Confirmation (to Visitor), Authentication Status (to user)

Process Logic:

1. Act as the primary interface for user entry.

2. Accept incoming data streams for both new registrations and returning logins.

3. Pass the raw data immediately to Process 1.2 for validation.

4. Receive final status results from subsequent processes and display confirmation or error messages back to the user interface.

Process Number: 1.2

Process Name: Validate User Input

Input Data Flows: raw user input (from 1.1)

Output Data Flows: validated credentials (to 1.3), invalid input error (to 1.1)

Process Logic:

1. Check that mandatory fields (email, password) are present.

2. Validate data formats (e.g., valid email structure, phone number length).

3. Sanitize inputs to prevent injection attacks.

4. If valid: Forward data to Process 1.3.

5. If invalid: Return error signal to Process 1.1.

Process Number: 1.3

Process Name: Verify Credentials

Input Data Flows: validated credentials (from 1.2)

Output Data Flows: authentication result (to 1.4), valid/invalid status (to 1.2/1.1)

Process Logic:

1. Query User DB (D1) for the provided email.

2. If Login: Compare stored hash with input password hash.

3. If Registration: Check if email/phone already exists in User DB.

   - If unique: Insert new record into User DB.

4. Pass the result (Success/Fail) and user ID to Process 1.4.

Process Number: 1.4

Process Name: Assign Roles

Input Data Flows: authentication result (from 1.3)

Output Data Flows: Final Session Object (to User)

Process Logic:

1. Read the user's role (Admin, Member, Visitor) from the verified user record.

2. Generate a session token with permissions specific to that role.

3. Finalize the login/registration process and grant access to the system.

8.2 Process: Reporting and Revenue Management

Process Number: 9.1

Process Name: Receive Report Request

Input Data Flows: Report Request (Time Period/Type from Admin)

Output Data Flows: Report Parameters (to 9.2), Final Financial Report (to Admin), Revenue Forecast (to Admin), Menu Recommendations (to Admin)

Process Logic:

1. Accept the specific date range and report type selected by the Admin.

2. Parse the parameters to determine which data sets are needed.

3. Trigger Process 9.2 to start data retrieval.

4. Collect all outputs from 9.3, 9.4, and 9.5 to assemble the final response for the Admin.

Process Number: 9.2

Process Name: Retrieve Historical Data

Input Data Flows: Report Parameters (from 9.1)

Output Data Flows: Aggregated Raw Data (to 9.3)

Process Logic:

1. Connect to Order DB, Payment DB, Billing DB, and Feedback DB.

2. Run queries to fetch all records matching the "Time Period" parameter.

3. Consolidate specific data points:

   - Total sales from Orders.

   - Successful transactions from Payments.

   - Customer ratings from Feedback.

4. Bundle this "Aggregated Raw Data" and send it to Process 9.3.

Process Number: 9.3

Process Name: Analyze Sales and Trends

Input Data Flows: Aggregated Raw Data (from 9.2)

Output Data Flows: Sales Analysis Report (to 9.4)

Process Logic:

1. Calculate Key Performance Indicators (KPIs): Total Revenue, Net Profit, Average Order Value.

2. Identify trends: Peak hours, best-selling items, underperforming items.

3. Structure these insights into a specific internal dataset ("Sales Analysis Report") for the forecasting engine.

Process Number: 9.4

Process Name: Generate Forecast

Input Data Flows: Sales Analysis Report (from 9.3)

Output Data Flows: Revenue Forecast Data (to 9.5), Financial Report & Menu Recommendations (to 9.5)

Process Logic:

1. Apply prediction algorithm (e.g., linear regression or moving average) to the trend data to estimate future revenue.

2. Flag menu items with high sales/high ratings (Recommended) and low sales/low ratings (To Remove).

3. Forward the specific forecast numbers and recommendation lists to Process 9.5.

Process Number: 9.5

Process Name: Produce Final Report

Input Data Flows: Revenue Forecast Data (from 9.4), Financial Report & Menu Recommendations (from 9.4)

Output Data Flows: Final Formatted Outputs (to 9.1)

Process Logic:

1. Take the raw data and analysis results.

2. Format them into a human-readable layout (PDF/HTML structure).

3. Include charts or summary tables.

4. Send the completed document back to Process 9.1 for delivery to the Admin.

8.3 Process 10.0: Payment Processing

Process Number: 10.1

Process Name: Receive Payment/Refund Request

Input Data Flows: Payment Request (from P2, P5), Payment Confirmation (to P7)

Output Data Flows: Payment Request Details (to 10.2), Payment Status (to caller)

Process Logic:

1. Act as the entry point for all financial transactions in the system.

2. Identify if the request is an "Incoming Payment" (Booking/Order) or "Refund" (Cancellation).

3. Validate that the amount and user details are correct.

4. Route the details to Process 10.2.

5. Upon completion, notify P7 (Billing) to generate the receipt.

Process Number: 10.2

Process Name: Prepare Transaction Detail

Input Data Flows: Payment Request Details (from 10.1)

Output Data Flows: Formatted Transaction Detail (to 10.3)

Process Logic:

1. Format the data into the specific JSON/XML structure required by the external Payment Gateway API.

2. Append merchant credentials (API Keys).

3. Pass the secure packet to Process 10.3.

Process Number: 10.3

Process Name: Send Transaction to Payment Gateway

Input Data Flows: Formatted Transaction Detail (from 10.2)

Output Data Flows: Transaction Reference & Amount (to External Gateway)

Process Logic:

1. Establish a secure HTTPS connection to the Payment Gateway.
2. Transmit the transaction payload.
3. Wait for the initial handshake or token acknowledgment.

Process Number: 10.4

Process Name: Receive Gateway Response

Input Data Flows: Transaction ID & Status (from External Gateway)

Output Data Flows: Gateway Response (to 10.5)

Process Logic:

1. Listening endpoint for the callback/webhook from the Payment Gateway.
2. Capture the result: Success, Failure, Insufficient Funds, etc.
3. Verify the digital signature of the response to ensure authenticity.
4. Forward the raw result to Process 10.5.

Process Number: 10.5

Process Name: Transaction Record

Input Data Flows: Gateway Response (from 10.4)

Output Data Flows: Final Status (back to 10.1 loop), Updates to Payment DB

Process Logic:

1. Write the transaction result into the Payment DB (D6).

   - Store Transaction ID, Timestamp, Amount, Status.

2. If successful, mark the associated Order/Reservation ID as "Paid".

3. Signal Process 10.1 that the cycle is complete so it can notify the user.

9 Structure Chart

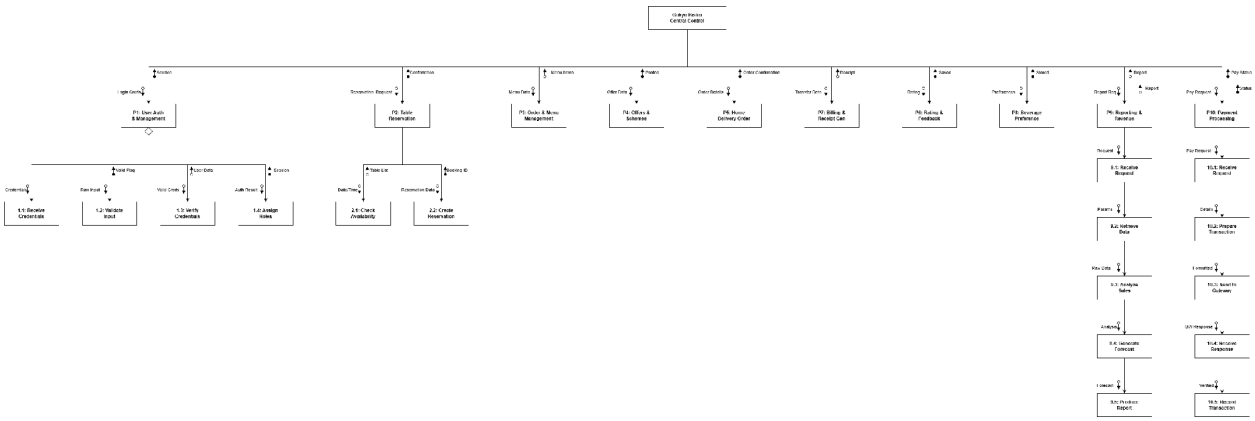

Figure 7: Group Structure Chart

10 Progress Log

10.1 Milestone 1: Analysis & Requirements Breakdown

Focus: Establishing the business need, system requirements, and initial data flow models.

Table 1: Milestone 1 Logbook

Student Name	Primary Contribution (Group Task)	Work Description
Abiram Bhatta (Leader)	**SRS & Data Dictionary**	Authored the **System Requirements Specification (SRS)**, defining functional/non-functional requirements and user classes. Compiled the initial **Data Dictionary** to standardize data element definitions across the project.
Nayan Chhantyal	**Business Case**	Developed the comprehensive **Business Case**, outlining the problem statement, organizational impact, financial justification, and strategic alignment for the Gokyo Bistro system.
Shlok Niroula	**Context & Level 1 DFDs**	Designed the **Context Diagram (Level 0)** to define system boundaries. Created the **Level 1 Data Flow Diagram**, decomposing the system into 10 core processes.
Sudip Giri	**Entity Relationship Diagram (ERD)**	Modeled the database structure, identifying key entities (User, Order, Payment, Table) and their relationships to create the normalized **ERD**.
Yakendra Bam Rajawar	**Project Introduction**	Wrote the **Project Introduction** and Overview, setting the scope, purpose, and background context for the entire documentation suite.

10.2 Milestone 2: Detailed Design & Specification

Focus: Refining analysis models, defining detailed logic, and structural design.

Table 2: Milestone 2 Logbook

Student Name	Primary Contribution (Group Task)	Detailed Work Description (Elaborated)
Abiram Bhatta (Leader)	**Project Tracking & Final Compilation**	**Project Coordination:** Tracked weekly progress and integrated artifacts from all members into the final Master Report. **Log Management:** Maintained the project progress log to ensure all deliverable deadlines were met. **Review:** Verified consistency between the SRS and final design artifacts.
Nayan Chhantyal	**Rules Refinement & SRS Updates**	**Business Logic Refinement:** Analyzed peer feedback and updated the Business Rules (e.g., cancellation fees, booking constraints).
Shlok Niroula	**DFD Refinement & Balancing**	**Advanced DFD Balancing:** Performed rigorous balancing checks between Parent and Child diagrams (Level 1 vs Level 2). **Error Correction:** Resolved data flow inconsistencies identified during the review phase, ensuring all inputs/outputs were accounted for.
Sudip Giri	**Group Structure Chart**	**Architectural Design:** Derived the hierarchical **Structure Chart** from the DFDs using Transform Analysis. **Module Mapping:** Mapped the Factoring of the system, defining the Control Module (Main System) and its subordinate worker modules.
Yakendra Bam Rajawar	**Process Specifications (PSpecs)**	**Logic Definition:** Authored detailed **Process Specifications (PSpecs)** for critical elementary processes (Auth, Payment, Reporting).

10.3 Individual Task Division

Table 3: Individual Task Division

Student Name	Assigned Component (Individual)
Abiram Bhatta	Booking Table
Nayan Chhantyal	Rate the experience
Shlok Niroula	Update Menu
Sudip Giri	Set Beverage Choice
Yakendra Bam Rajawar	Place order for home delivery

Individual Task

Member 1

Student Name: Abiram Bhatta

LMU ID: 24046558

Function chosen: Book Table

Context Level

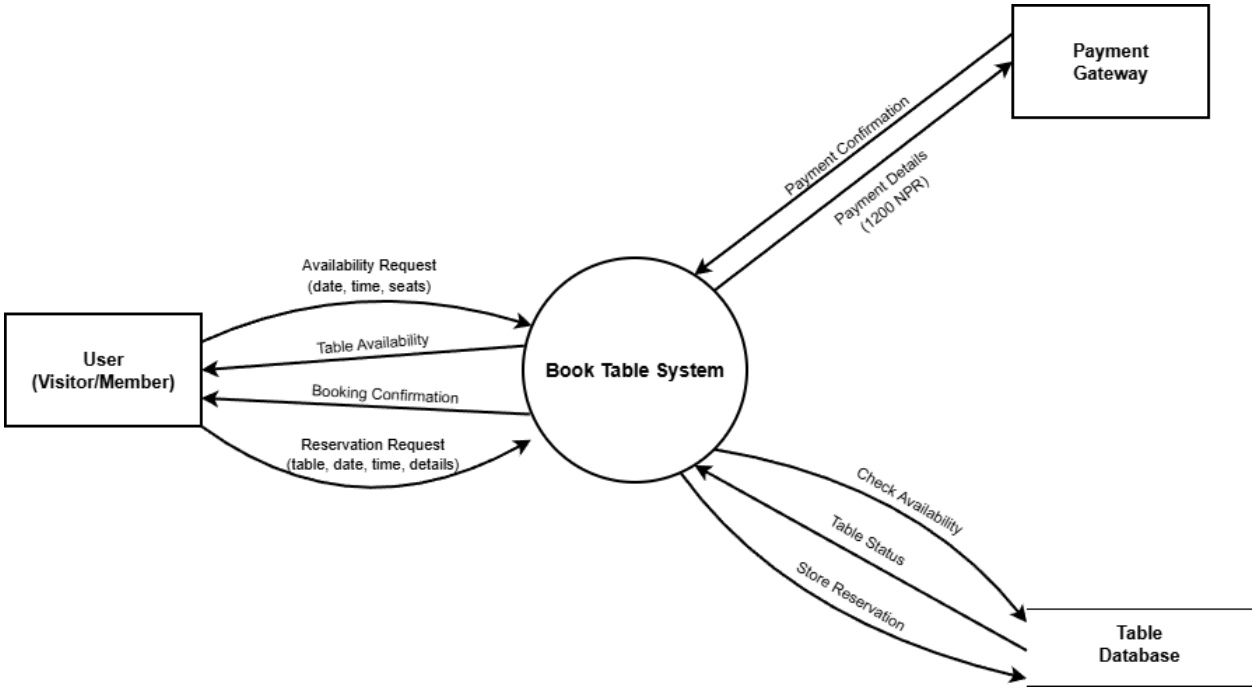

Figure 8: Member 1 DFD 0

Description

The Level 0 DFD provides a high-level logical view of the entire Book Table System, representing it as a single process "Book Table System" labeled. This system interacts primarily with two external entities: the User (Visitor/Member) and the Payment Gateway. The User initiates the process by sending a Book Table Request containing date, time, and preferences, and in return receives Table Availability and a final Booking Confirmation. The Payment Gateway is an external system that receives Payment Details from the process and returns a Transaction Status (Success/Failure) to confirm the financial transaction. This diagram establishes the system's boundaries and identifying its primary inputs and outputs without delving into internal processing details.

Level 1 DFD

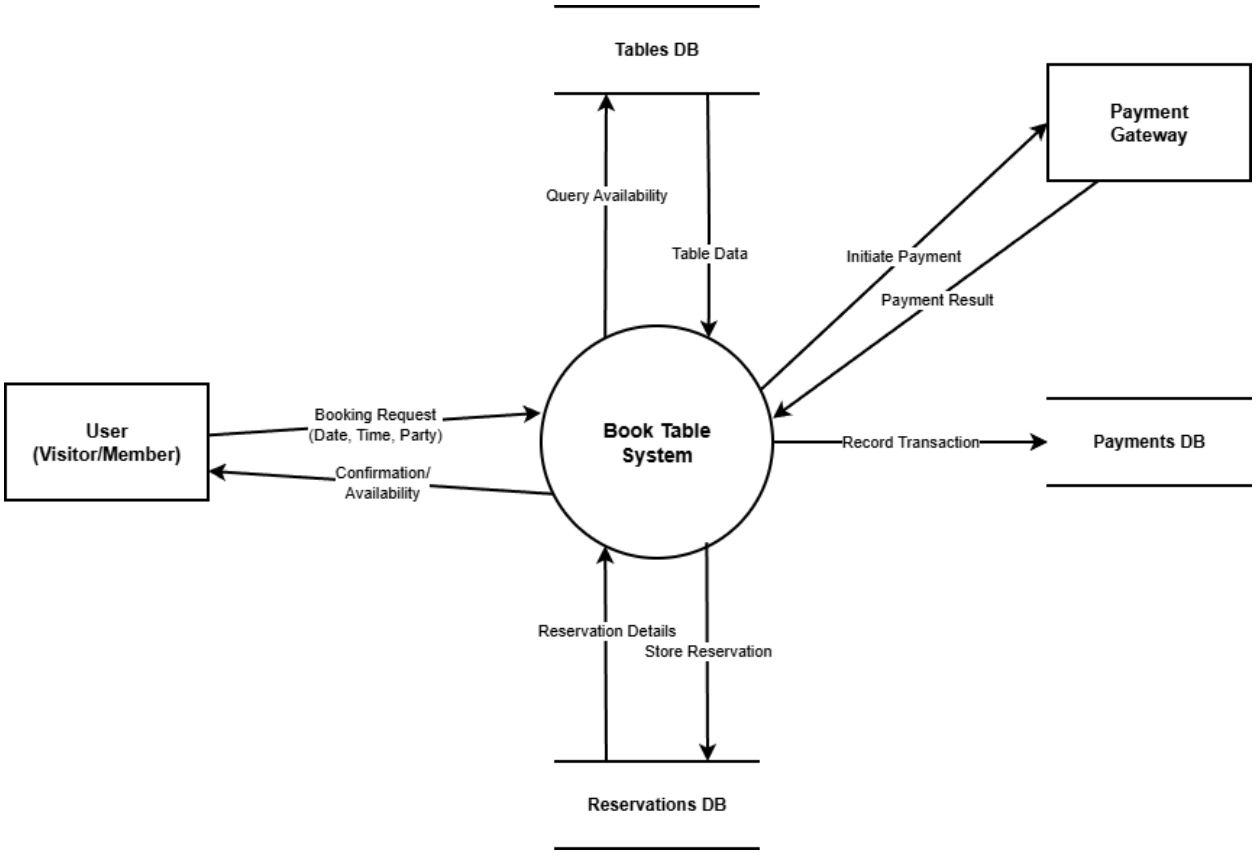

Figure 9: Member 1 DFD Level 1

Description

The Level 1 DFD decomposes the single Context Process into the system's four main functional processes: **1.0 Check Table Availability**, **2.0 Process Payment**, **3.0 Create Reservation**, and **4.0 Generate Confirmation**. This level introduces the system's internal data storage, specifically the **Tables DB**, **Payments DB**, and **Reservations DB**. The flow begins with the User's request entering Process 1.0, which queries the Tables DB to verify availability. If confirmed, the flow moves to Process 2.0, which interacts with the Payment Gateway and records the transaction in the Payments DB. Upon successful payment, Process 3.0 stores the booking details in the Reservations DB. Finally, Process 4.0 retrieves these details to generate and send a confirmation message back to the User, showcasing the sequential data flow through the primary system functions.

Level 2 DFD

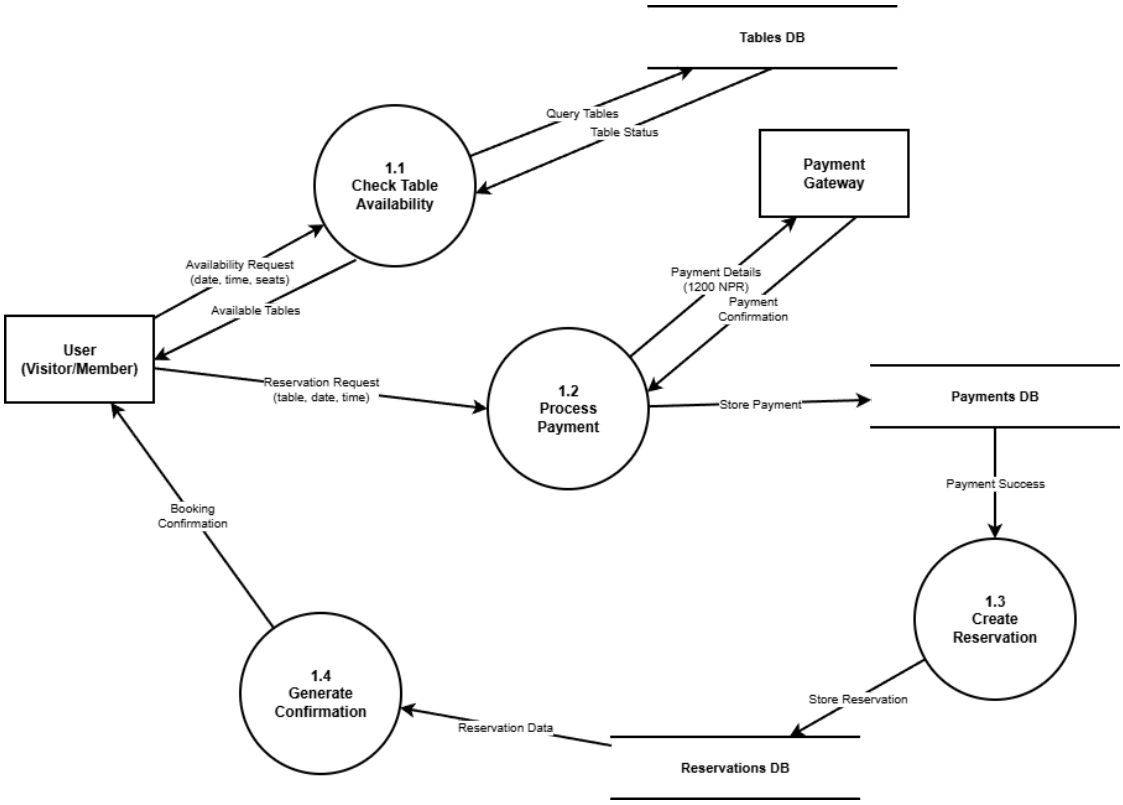

Figure 10: Member 1 DFD Level 2

Description

The Level 2 DFD offers a granular view of the specific sub-processes within each Level 1 function. For example, the **Check Table Availability** process is broken down into sub-processes like *Capture Request*, *Query Database*, and *Format Response*. This diagram clarifies exactly how data is transformed at each step. It illustrates how the *Capture Request* sub-process validates user input before passing structured criteria to *Query Database*, which performs the actual SQL lookup. The results are then passed to *Format Response* to create a user-friendly list. Similarly, the **Process Payment** function decomposes into *Initiate Payment* (handling the gateway handshake) and *Record Transaction* (logging the result), providing a detailed blueprint of the system's internal logic and error-handling pathways.

Structure Chart

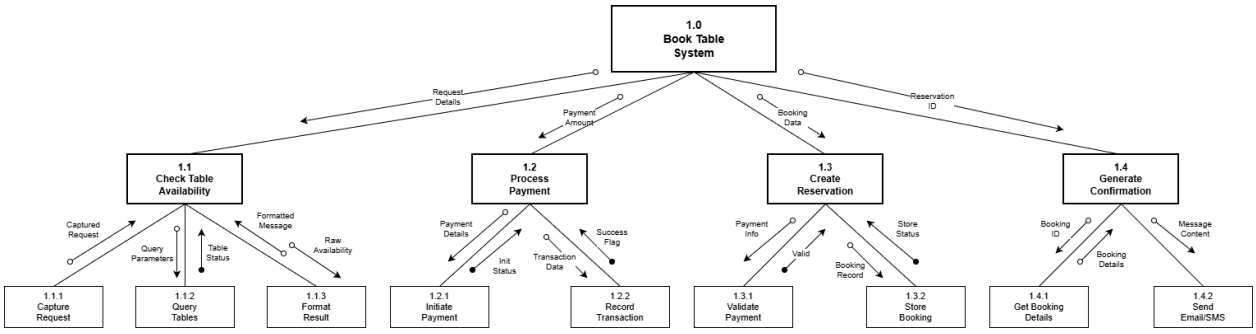

Figure 11: Structure Chart for Book Table

Module Specifications (Mspecs)

1.0 BOOK TABLE SYSTEM (Main Controller)

PURPOSE

Main controller module that coordinates the complete table booking process. It manages the linear workflow by sequentially calling the four main sub-systems: Availability Check, Payment, Reservation Creation, and Confirmation.

PSEUDOCODE

```
BEGIN BookTableSystem
    CALL 1.1_CheckTableAvailability(userRequest)
    IF availability_status = TRUE THEN
        CALL 1.2_ProcessPayment(paymentAmount)
        IF payment_success = TRUE THEN
            CALL 1.3_CreateReservation(bookingData)
            reservationID = GET_RETURN_VALUE()
            CALL 1.4_GenerateConfirmation(reservationID)
            RETURN success_message
        ELSE
            RETURN payment_error_message
        END IF
```

```
    ELSE
        RETURN no_tables_message
    END IF
END BookTableSystem
```

INPUT PARAMETERS: userRequest (Date, Time, Guests)

OUTPUT PARAMETERS: bookingResult (Status Message)

GLOBAL VARIABLES:

- systemConfig: SystemConfiguration (Timeouts, API keys)
- userSession: SessionData (Current User ID)

LOCAL VARIABLES:

- availability_status: Boolean
- payment_success: Boolean
- reservationID: String

CALLS: 1.1 Check Table Availability, 1.2 Process Payment, 1.3 Create Reservation, 1.4 Generate Confirmation

CALLED BY: User Interface (Visitor / Member)

1.1 CHECK TABLE AVAILABILITY

PURPOSE
Orchestrates the availability check by capturing the user's request, querying the database for matching tables, and formatting the results for the controller.

PSEUDOCODE

```
BEGIN CheckTableAvailability
```

```
    CALL 1.1.1_CaptureRequest(rawInput)
    searchCriteria = GET_RETURN_VALUE()

    CALL 1.1.2_QueryTables(searchCriteria)
    tableStatus = GET_RETURN_VALUE()
    CALL 1.1.3_FormatResult(tableStatus)
    RETURN availability_status
END CheckTableAvailability
```

INPUT PARAMETERS: userRequest

OUTPUT PARAMETERS: availabilityStatus (Boolean)

GLOBAL VARIABLES:

- dbConnection: DatabaseConnection (Read-Only access to Tables)

LOCAL VARIABLES:

- searchCriteria: SearchCriteriaObject
- tableStatus: ResultSet
- rawInput: String

CALLS: 1.1.1 Capture Request, 1.1.2 Query Tables, 1.1.3 Format Result

CALLED BY: 1.0 Book Table System

1.1.1 CAPTURE REQUEST

PURPOSE: Validates and parses raw user input into a structured query object.
CALLED BY: 1.1 Check Table Availability

1.1.2 QUERY TABLES

PURPOSE: Executes SQL SELECT queries against the Tables Database using the search criteria.

CALLED BY: 1.1 Check Table Availability

1.1.3 FORMAT RESULT

PURPOSE: Converts raw database rows into a standardized status flag or list for the controller.
CALLED BY: 1.1 Check Table Availability

1.2 PROCESS PAYMENT

PURPOSE

Manages the financial transaction. It initiates the request with the external gateway and, upon success, records the transaction details in the system database.

PSEUDOCODE

```
BEGIN ProcessPayment
    CALL 1.2.1_InitiatePayment(amount, cardDetails)
    gatewayResult = GET_RETURN_VALUE()

    IF gatewayResult = SUCCESS THEN
        CALL 1.2.2_RecordTransaction(gatewayResult)
        RETURN TRUE
    ELSE
        RETURN FALSE
    END IF
END ProcessPayment
```

INPUT PARAMETERS: paymentAmount

OUTPUT PARAMETERS: successFlag (Boolean)

GLOBAL VARIABLES:

- paymentGateway: GatewayAPIConnection
- paymentsDB: DatabaseConnection (Write access)

LOCAL VARIABLES:

- gatewayResult: PaymentResponseObject
- transactionID: String

CALLS: 1.2.1 Initiate Payment, 1.2.2 Record Transaction

CALLED BY: 1.0 Book Table System

1.2.1 INITIATE PAYMENT

PURPOSE: Sends encrypted transaction details to the Payment Gateway API and awaits response.

CALLED BY: 1.2 Process Payment

1.2.2 RECORD TRANSACTION

PURPOSE: Logs the successful transaction ID and timestamp into the Payments Database.

CALLED BY: 1.2 Process Payment

1.3 CREATE RESERVATION

PURPOSE
Handles the creation of the permanent reservation record. It validates the payment proof and stores the new booking in the database.

PSEUDOCODE

```
BEGIN CreateReservation
    CALL 1.3.1_ValidatePayment(paymentInfo)
    isValid = GET_RETURN_VALUE()

    IF isValid = TRUE THEN
        CALL 1.3.2_StoreBooking(bookingDetails)
        newID = GET_RETURN_VALUE()
        RETURN newID
    ELSE
        THROW VerificationException
    END IF
END CreateReservation
```

INPUT PARAMETERS: bookingData

OUTPUT PARAMETERS: reservationID

GLOBAL VARIABLES:

- reservationsDB: DatabaseConnection (Write access)

LOCAL VARIABLES:

- isValid: Boolean
- newID: String
- bookingDetails: ReservationRecord

CALLS: 1.3.1 Validate Payment, 1.3.2 Store Booking

CALLED BY: 1.0 Book Table System

1.3.1 VALIDATE PAYMENT

PURPOSE: Internal double-check to ensure payment was recorded before creating a booking.

CALLED BY: 1.3 Create Reservation

1.3.2 STORE BOOKING

PURPOSE: Generates a unique Reservation ID and INSERTS the record into the Reservations Database.

CALLED BY: 1.3 Create Reservation

1.4 GENERATE CONFIRMATION

PURPOSE
Finalizes the process by retrieving the complete booking details and dispatching the confirmation message to the user.

PSEUDOCODE

```
BEGIN GenerateConfirmation
    CALL 1.4.1_GetBookingDetails(reservationID)
    fullDetails = GET_RETURN_VALUE()

    messageContent = FormatTemplate(fullDetails)

    CALL 1.4.2_SendEmailSMS(messageContent)
    RETURN confirmation_sent
END GenerateConfirmation
```

INPUT PARAMETERS: reservationID

OUTPUT PARAMETERS: confirmationStatus

GLOBAL VARIABLES:

- notificationService: EmailSMSServiceConnection
- reservationsDB: DatabaseConnection (Read access)

LOCAL VARIABLES:

- fullDetails: BookingObject
- messageContent: String

CALLS: 1.4.1 Get Booking Details, 1.4.2 Send Email/SMS

CALLED BY: 1.0 Book Table System

1.4.1 GET BOOKING DETAILS

PURPOSE: Queries the Reservations DB to get all details (Date, Table No, User Name) for the message.

CALLED BY: 1.4 Generate Confirmation

1.4.2 SEND EMAIL/SMS

PURPOSE: Connects to the notification service to send the formatted Booking Confirmation to the user.
CALLED BY: 1.4 Generate Confirmation

Member 2

Student Name: Nayan Chhantyal

LMU ID: 24046601

Function chosen: Rate the Experience

Context Level Diagram

Figure 12: Member 2 DFD Level 0

Description

The Context Diagram provides the most abstract view of the "Rate the Experience" system, defining the boundary between the software and its external environment. It represents the entire system as a single process, showing how the Visitor/Member provides Rating Details and receives a Submission Confirmation. Simultaneously, it establishes the administrative interface, where the Admin sends Report Requests and receives the final Performance Recommendations. This diagram ensures all high-level stakeholders and their data requirements are accounted for before the system is broken down into technical parts.

DFD Level 1

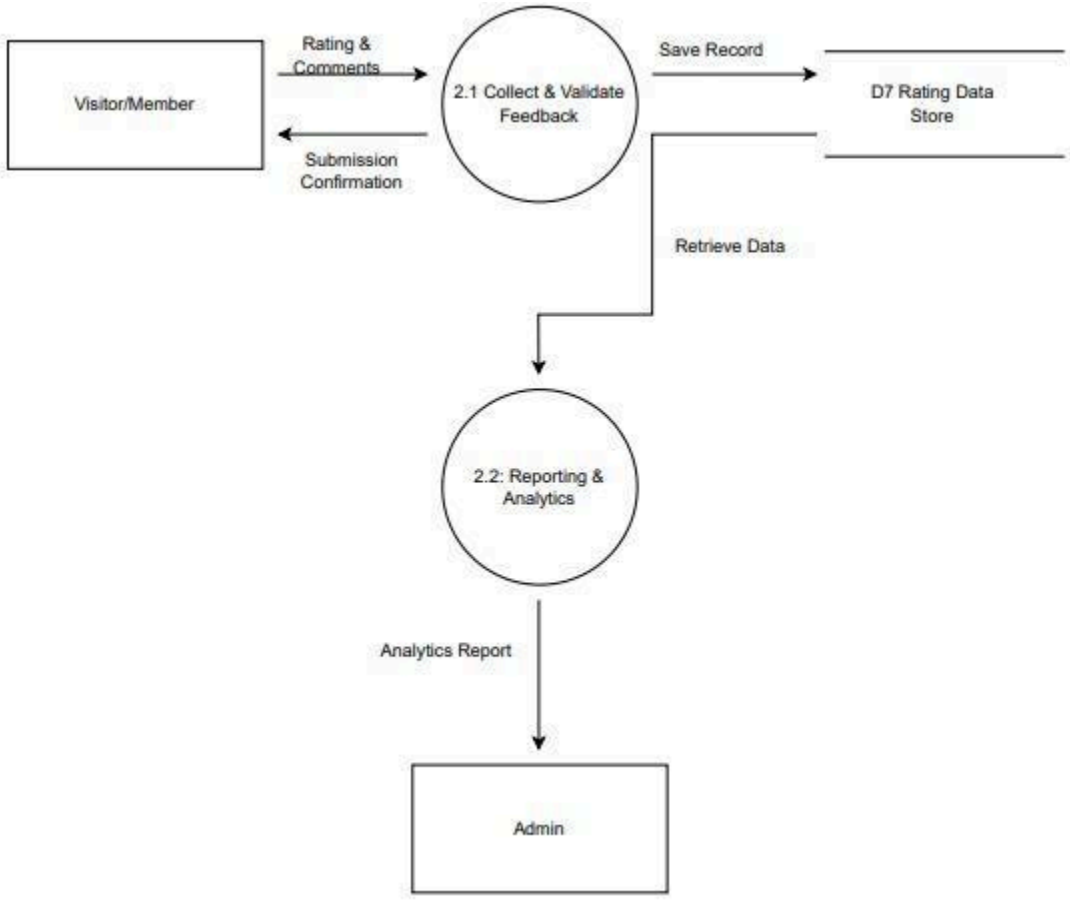

Figure 13: Member 2 DFD Level 1

Description

The Level 1 DFD decomposes the system into two primary functional areas: Feedback Collection (2.1) and Reporting & Analytics (2.2). This level introduces the D7 Rating Data Store, which acts as the central hub for the system's data persistence. The diagram illustrates the "Store and Forward" logic: Process 2.1 validates and "Saves" the record to the database, while Process 2.2 "Retrieves" that historical data to generate insights for the Admin. This view is critical for understanding the high-level data flow and the independence of the visitor and administrator functions.

DFD Level 2

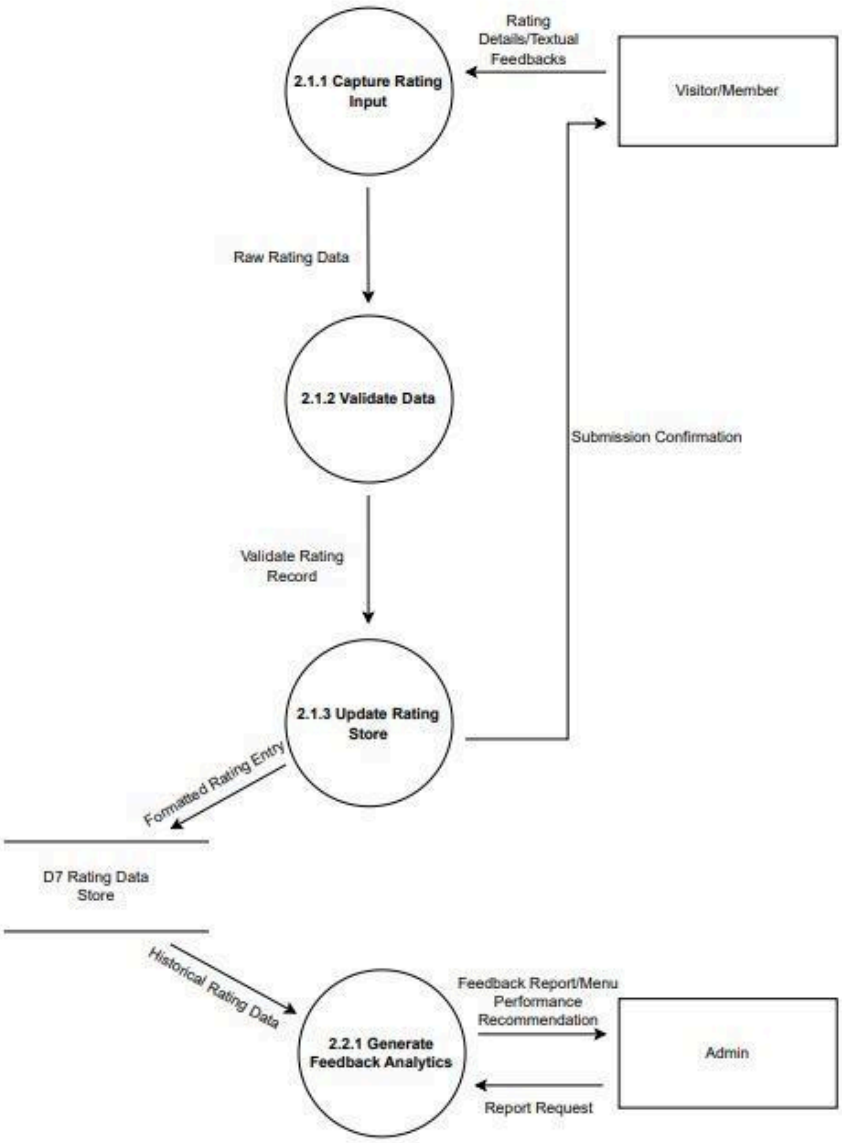

Figure 14: Member 2 DFD Level 2

Description

The Level 2 DFD provides a "deep dive" into the internal logic of the Feedback Processor. It breaks the main process down into specific, code-able sub-processes: Capture (2.1.1), Validate (2.1.2), and Update (2.1.3). This level of detail is necessary to show exactly how raw data from the visitor is transformed into a formatted, validated record before it ever touches the database. It also clarifies the error-handling path, ensuring that only clean data is stored while providing immediate feedback to the user via the Submission Confirmation path.

Structure Chart

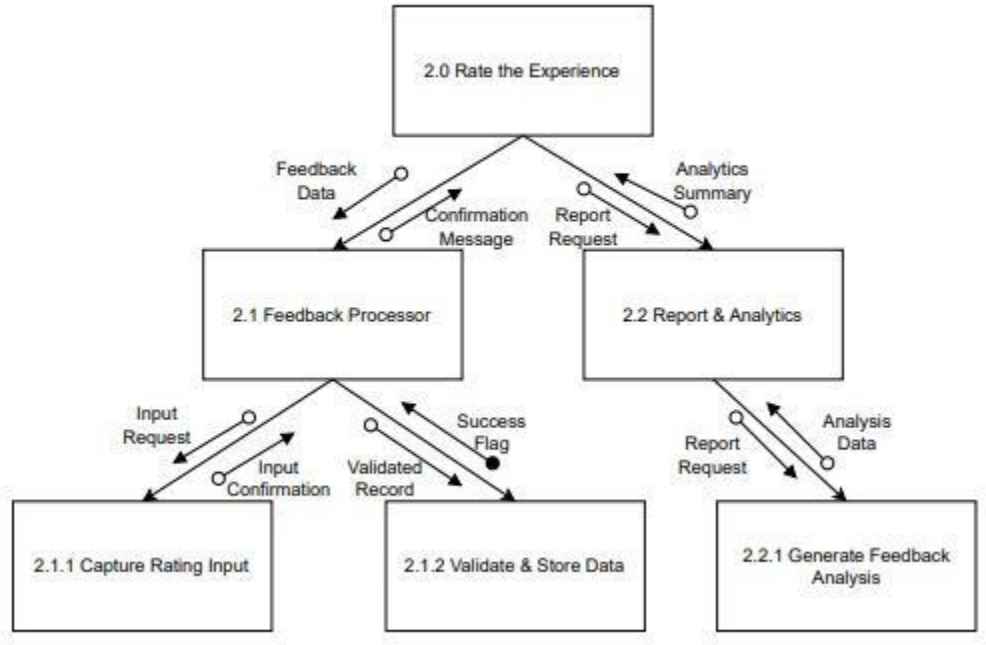

Figure 15: Member 2 Structure Chart

Description

The Structure Chart shifts the focus from "what data moves" to "how the software is controlled". At the top, the 2.0 Controller acts as the central executive, managing the execution of the lower-level modules. The chart uses Transform Analysis to separate the system into an input branch (2.1 Feedback Processor) and an output branch (2.2 Report & Analytics). By using Data Couples and Control Couples (like the Success Flag), the chart demonstrates a highly modular design where each box has a single, specific purpose. This hierarchical arrangement is a blueprint for the actual coding phase, showing exactly how different functions will call each other in the final program.

Module Specifications (Msepcs)

MODULE NAME: 2.0 RATE THE EXPERIENCE CONTROLLER

PURPOSE: Main controller module that coordinates the end-to-end feedback process from user input capture to analytics generation.

PSEUDOCODE

```
BEGIN RateExperienceController
    CALL CaptureRatingInput()
    RECEIVE rawRatingData
    CALL ValidateData(rawRatingData)
    RECEIVE validationFlag
    IF validationFlag IS Valid THEN
        CALL UpdateRatingStore(rawRatingData)
        IF writeStatus IS Success THEN
            DISPLAY "Submission Successful"
        END IF
    ELSE
        DISPLAY "Error: Invalid Feedback Data"
    END IF
    IF adminReportRequested THEN
        CALL GenerateFeedbackAnalytics()
    END IF
END RateExperienceController
```

INPUT PARAMETERS

userInput: RawUserFeedback (score, comments, category)

OUTPUT PARAMETERS

submissionResult: ProcessStatus (success, invalid_data, db_error)

CALLS

- CaptureRatingInput()
- ValidateData()
- UpdateRatingStore()
- GenerateFeedbackAnalytics()

MODULE NAME: 2.2 VALIDATE DATA

PURPOSE Validates the raw rating input to ensure it meets system constraints before storage.

PSEUDOCODE

```
BEGIN ValidateData
    INPUT rawRatingData
    IF (rawRatingData.score >= 1 AND rawRatingData.score <= 5) THEN
        IF (length(rawRatingData.comment) <= 500) THEN
            validationFlag = TRUE
        ELSE
            validationFlag = FALSE
        END IF
    ELSE
        validationFlag = FALSE
    END IF
    RETURN validationFlag
END ValidateData
```

INPUT PARAMETERS

rawRatingData: UnvalidatedFeedback

OUTPUT PARAMETERS

validationFlag: Boolean

MODULE NAME: 2.3 UPDATE RATING STORE

PURPOSE Manages the preparation and storage of a validated record, including formatting and unique ID assignment.

PSEUDOCODE

```
BEGIN UpdateRatingStore
    INPUT validatedRecord
    CALL FormatRatingTable(validatedRecord)
    formattedData = GetFormattedResult()
    CALL AssignIDAndTimestamp(formattedData)
    completeEntry = GetFinalRecord()
    CALL PhysicalWriteToDB(completeEntry)
    status = GetDBStatus()
    RETURN status
END UpdateRatingStore
```

CALLS

- FormatRatingTable()
- AssignIDAndTimestamp()
- PhysicalWriteToDB()

MODULE NAME: 2.3.2 ASSIGN ID & TIMESTAMP

PURPOSE Generates a unique identifier and records the current system time for a new rating entry.

PSEUDOCODE

```
BEGIN AssignIDAndTimestamp
    INPUT formattedRecord
    uniqueID = "RAT-" + GetCurrentDate() + GenerateRandom(100, 999)
    currentTime = GetSystemTime()
    completeEntry = Merge(formattedRecord, uniqueID, currentTime)
    RETURN completeEntry
END AssignIDAndTimestamp
```

LOCAL VARIABLES

- **uniqueID**: String
- **currentTime**: DateTime

MODULE NAME: 2.4 GENERATE FEEDBACK ANALYTICS

PURPOSE Retrieves historical data to generate performance reports for Admin review.

PSEUDOCODE

```
BEGIN GenerateFeedbackAnalytics
    INPUT reportRequest
    OPEN D7_RatingDataStore
    historicalData = FETCH ALL WHERE category = reportRequest.category
    reportContent = CALCULATE_AVERAGE(historicalData.scores)
    finalReport = FORMAT_REPORT(reportContent)
    SEND finalReport TO Admin
```

```
    RETURN success
END GenerateFeedbackAnalytics
```

GLOBAL VARIABLES

- **D7_RatingDataStore**: DatabaseConnection

Member 3

Student Name: Shlok Niroula

LMU ID: 24046644

Function chosen: Update Menu

Context Level Diagram

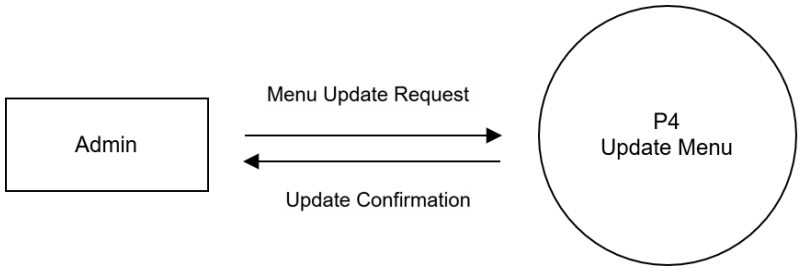

Figure 16: Member 3 DFD Level 0

Description

This DFD provides a high-level overview of the Update Menu system by showing its interaction with external entities. It represents the system as a single process that receives menu update requests from the administrator and returns confirmation or error messages after processing the request. Internal processes and data storage are intentionally hidden to clearly define the system boundary.

DFD level 1

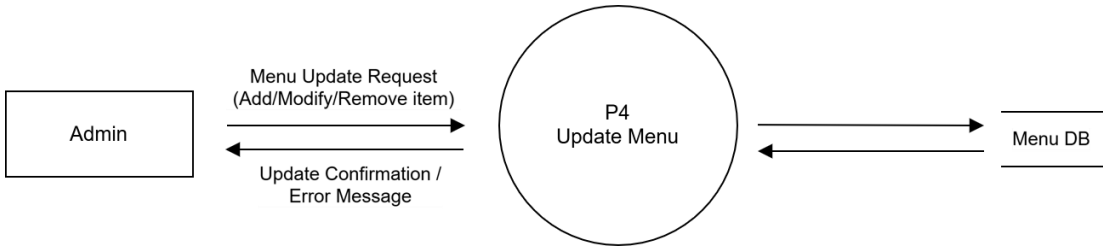

Figure 17: Member 3 DFD Level 1

Description

This DFD expands the context diagram by introducing the Menu Database to show how menu data is stored and retrieved. The Update Menu system remains a single process, but its interaction with persistent data storage is now visible. The administrator submits menu update requests, the system updates or retrieves menu records from the database, and a confirmation or error message is returned to the administrator.

DFD Level 2

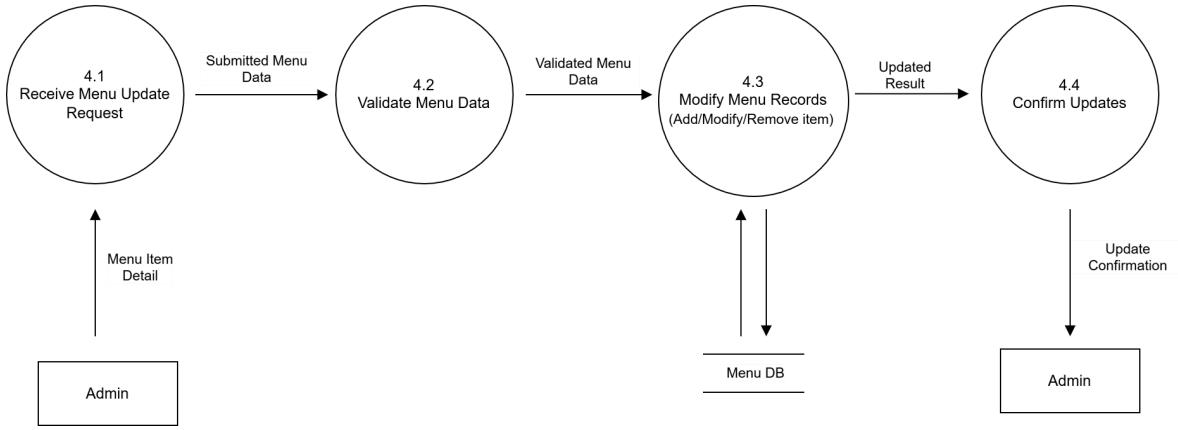

Figure 18: Member 3 DFD Level 2

Description

This DFD further decomposes the Update Menu system into its core internal processes: receiving the update request, validating menu data, modifying menu records, and confirming the update. It illustrates how data flows between these processes and the Menu Database to ensure that only validated menu changes are applied. This level provides a clear view of the internal workflow while maintaining logical abstraction.

Structure Chart

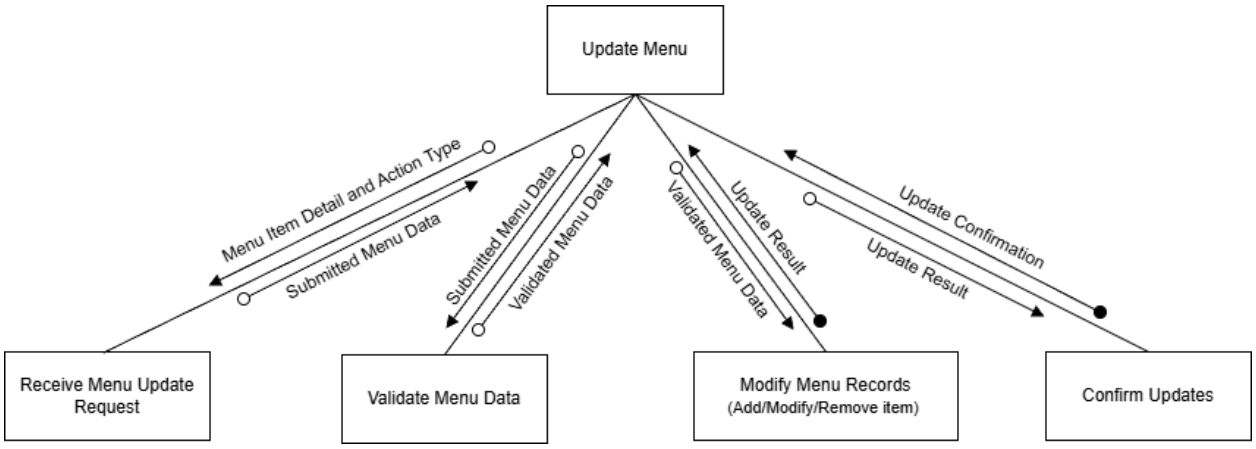

Figure 19: Member 3 Structure Chart

Module Specification (Mspecs)

MODULE NAME

P4 – Update Menu

PURPOSE

This module allows an administrator to manage menu items by receiving menu update requests, validating the submitted data, modifying menu records, and providing confirmation of the update operation.

PSEUDOCODE

```
BEGIN
   RECEIVE menuUpdateRequest FROM Admin

   SEND menuUpdateRequest TO ReceiveMenuUpdateRequest

   SEND receivedData TO ValidateMenuData

   IF validationSuccessful THEN
      SEND validatedData TO ModifyMenuRecords
      RECEIVE updateResult
      SEND updateResult TO ConfirmUpdate
   ELSE
      SEND validationError TO Admin
   ENDIF
END
```

INPUT PARAMETERS

- menuUpdateRequest
  (Menu item details and requested action)

OUTPUT PARAMETERS

- updateConfirmation
- errorMessage

GLOBAL VARIABLES

- None

LOCAL VARIABLES

- validatedMenuData
- updateResult

CALLS

- P4.1 – Receive Menu Update Request
- P4.2 – Validate Menu Data
- P4.3 – Modify Menu Records
- P4.4 – Confirm Update

CALLED BY

- Admin

MODULE NAME

P4.1 – Receive Menu Update Request

PURPOSE

Captures menu update requests submitted by the administrator.

PSEUDOCODE

```
BEGIN
   RECEIVE menuUpdateRequest FROM Admin
   FORWARD requestData TO ValidateMenuData
END
```

INPUT PARAMETERS

- Menu update request

OUTPUT PARAMETERS

- Submitted menu data

CALLS

- P4.2 – Validate Menu Data

CALLED BY

- P4 – Update Menu

MODULE NAME

P4.2 – Validate Menu Data

PURPOSE

Validates the menu update data for completeness, correctness, and authorization.

PSEUDOCODE

```
BEGIN
   RECEIVE submittedMenuData
   IF dataIsValid THEN
```

```
    FORWARD validatedData TO ModifyMenuRecords
  ELSE
    RETURN validationError
  ENDIF
END
```

INPUT PARAMETERS

- Submitted menu data

OUTPUT PARAMETERS

- Validated menu data
- Validation error

CALLS

- None

CALLED BY

- P4.1 – Receive Menu Update Request

MODULE NAME

P4.3 – Modify Menu Records

PURPOSE

Coordinates the modification of menu records based on validated menu data.

PSEUDOCODE

```
BEGIN
  RECEIVE validatedMenuData
```

PERFORM menuModification

RETURN modificationResult

END

INPUT PARAMETERS

- validatedMenuData
  (Validated menu item information and requested action)

OUTPUT PARAMETERS

- modificationResult
  (Result of menu modification operation)

GLOBAL VARIABLES

- None

LOCAL VARIABLES

- operationType
- processingStatus

CALLS

- Menu Database
- **P4.4 – Confirm Update**

CALLED BY

- **P4.2 – Validate Menu Data**

MODULE NAME

P4.4 – Confirm Update

PURPOSE

Generates a confirmation or error message based on the result of the menu update operation.

PSEUDOCODE

BEGIN

  RECEIVE modificationResult

  SEND confirmationMessage TO Admin

END

INPUT PARAMETERS

- modificationResult
  (Outcome of menu modification operation)

OUTPUT PARAMETERS

- confirmationMessage
  (Success or error response sent to Admin)

GLOBAL VARIABLES

- None

LOCAL VARIABLES

- statusMessage

CALLS

- None

CALLED BY

P4.3 – Modify Menu Records

Member 4

Student Name: Sudip Giri

LMU ID: 24046651

Function Chosen: Set Beverage Preference

Context Level Diaram

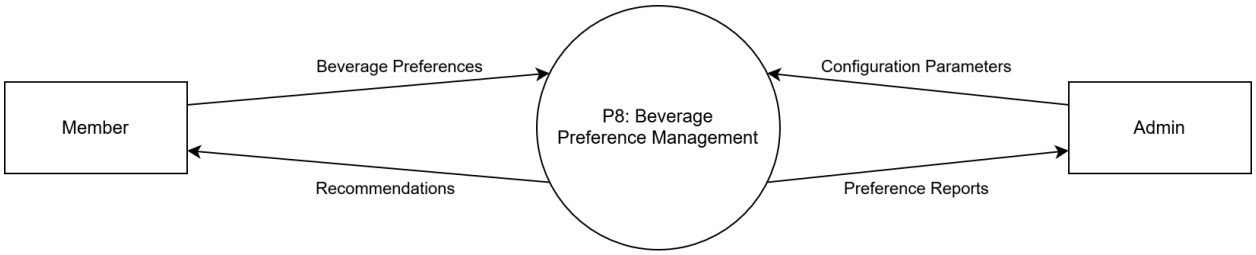

Figure 20: Member 4 DFD Level 0

Description
The Level 0 DFD shows the overall flow of the Beverage Preference Management process. A member provides their beverage preferences to the system, and based on these inputs, the system generates suitable beverage recommendations for the member. At the same time, the admin interacts with the system by providing configuration parameters and receiving preference reports. This level gives a high-level view of how members and admins communicate with the system without showing internal details.

DFD Level 1

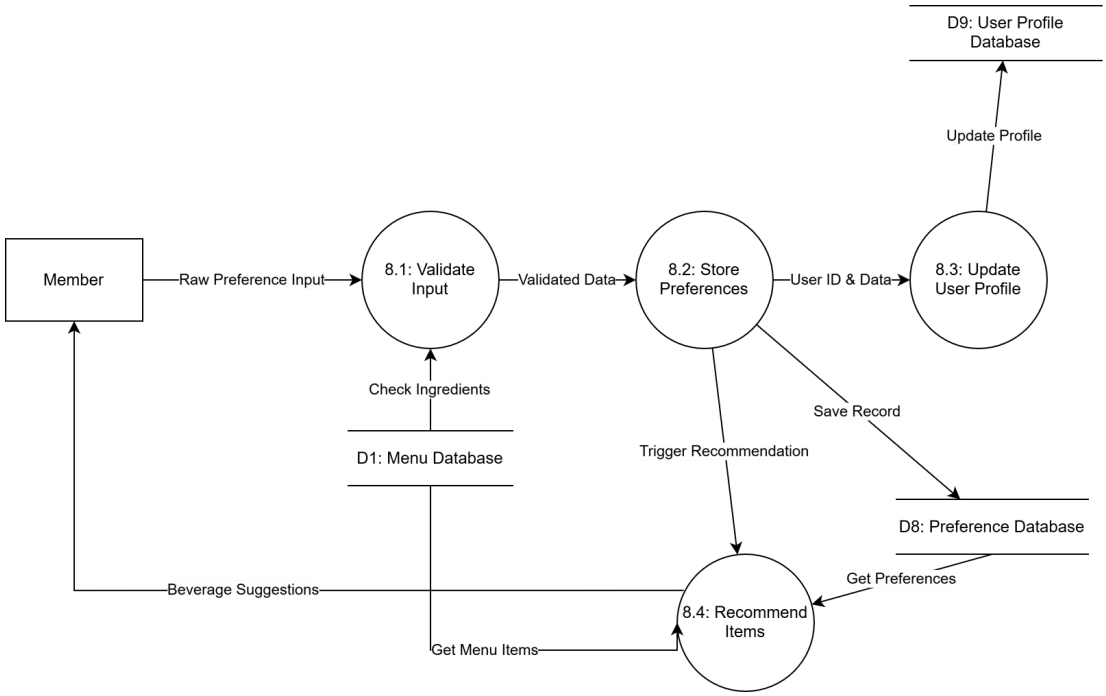

Figure 21: Member 4 DFD Level 1

Description

The Level 1 DFD explains the internal working of the Beverage Preference Management process in more detail. First, the member submits raw preference inputs, which are validated by checking ingredients against the menu database. Once validated, the preferences are stored and linked with the user's ID. The system then updates the user profile and saves the data in the preference database. Based on the stored preferences and available menu items, the system recommends suitable beverages to the member. This level clearly shows how data flows between processes and databases to produce accurate beverage suggestions.

DFD Level 2

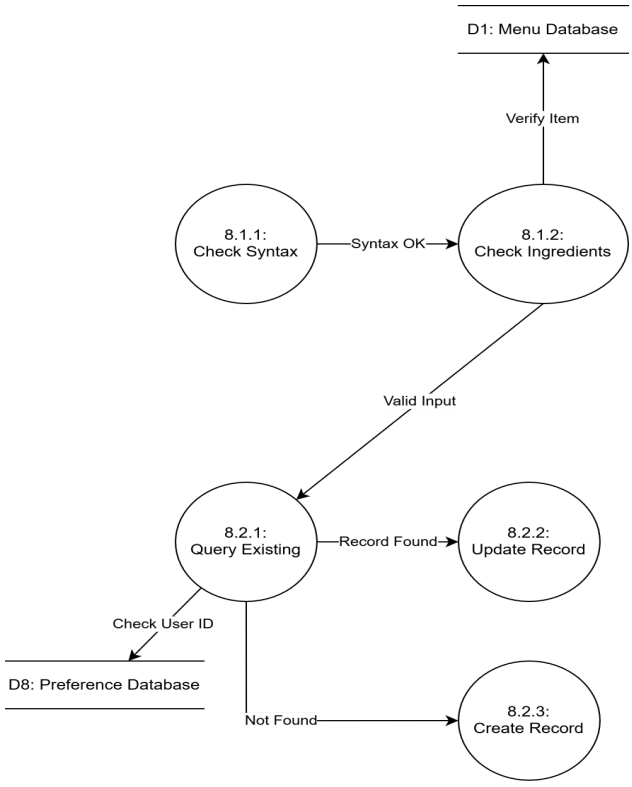

Figure 22: Member 4 DFD Level 2

Description

The Level 2 DFD breaks down the validation and preference storage processes in greater detail. First, the system checks the syntax of the member's input to ensure it is correctly formatted. Once the syntax is approved, the ingredients are verified against the menu database to confirm that the selected items are valid. After validation, the system checks the preference database to see if a record already exists for the user. If a record is found, the existing data is updated; if not, a new preference record is created. This detailed level shows how the system ensures data accuracy and properly manages user preference records.

Structure Chart

Figure 23: Member 4 Structure chart

Module Specifications (MSpecs)

MODULE NAME: BEVERAGE PREFERENCE CONTROLLER

PURPOSE
Main controller module that manages user beverage preferences, from capturing inputs to generating personalized recommendations.

PSEUDOCODE

```
BEGIN BeveragePreferenceController
   CALL ReceivePreferences(userInput)
   IF ValidateInput(userInput) THEN
      preferences = CALL FormatPreferences(userInput)
      CALL StorePreferences(preferences, userID)
```

```
      CALL UpdateProfile(userID, preferences)
      recommendations = CALL GenerateRecommendation(preferences)
      CALL ConfirmUpdate(recommendations)
      RETURN update_success
    ELSE
      RETURN invalid_input
    END IF
END BeveragePreferenceController
```

INPUT PARAMETERS

- userInput: RawPreferenceData (type, sugar_level, temperature, base)
- userID: String

OUTPUT PARAMETERS

- updateResult: StatusMessage

GLOBAL VARIABLES

- sessionData: UserSession

LOCAL VARIABLES

- preferences: FormattedData
- recommendations: List

CALLS

- ReceivePreferences()
- ValidateInput()
- FormatPreferences()
- StorePreferences()
- UpdateProfile()

- GenerateRecommendation()
- ConfirmUpdate()

CALLED BY

- Main Application Controller

MODULE NAME: VALIDATE INPUT

PURPOSE
Ensures selected beverage options are valid (e.g., sugar level within range, available ingredients).

PSEUDOCODE

```
BEGIN ValidateInput
  INPUT rawData
  IF IsValidType(rawData.type) AND IsValidSugar(rawData.sugar) THEN
    IF CheckIngredients(rawData.type) THEN
      RETURN valid
    ELSE
      RETURN ingredients_unavailable
    END IF
  ELSE
    RETURN invalid_parameters
  END IF
END ValidateInput
```

INPUT PARAMETERS

- rawData: RawPreferenceData

OUTPUT PARAMETERS

- isValid: Boolean

GLOBAL VARIABLES

- menuDB: DBConnection

LOCAL VARIABLES

- stockStatus: Boolean

CALLS

- IsValidType()
- IsValidSugar()
- CheckIngredients()

CALLED BY

- BeveragePreferenceController()

MODULE NAME: STORE PREFERENCES

PURPOSE
Saves the formatted preferences into the dedicated beverage preference database.

PSEUDOCODE

```
BEGIN StorePreferences
   INPUT prefs, userID
   existingPrefs = QueryPreferenceDB(userID)
   IF existingPrefs EXISTS THEN
      CALL UpdateRecord(userID, prefs)
   ELSE
      CALL CreateRecord(userID, prefs)
   END IF
   RETURN storage_success
END StorePreferences
```

INPUT PARAMETERS

- prefs: FormattedData

- userID: String

OUTPUT PARAMETERS

- status: Boolean

GLOBAL VARIABLES

- preferenceDB: DBConnection

LOCAL VARIABLES

- existingPrefs: Record

CALLS

- QueryPreferenceDB()
- UpdateRecord()
- CreateRecord()

CALLED BY

- BeveragePreferenceController()

MODULE NAME: GENERATE RECOMMENDATION

PURPOSE
Analyzes user preferences to suggest specific menu items.

PSEUDOCODE

```
BEGIN GenerateRecommendation
   INPUT userPrefs
   menuItems = QueryMenuDB("CATEGORY='BEVERAGE'")
   recommendedProperties = MatchAlgorithm(userPrefs, menuItems)
```

```
    RETURN recommendedProperties
END GenerateRecommendation
```

INPUT PARAMETERS

- userPrefs: FormattedData

OUTPUT PARAMETERS

- suggestions: List<MenuItem>

GLOBAL VARIABLES

- menuDB: DBConnection

LOCAL VARIABLES

- menuItems: List
- matches: List

CALLS

- QueryMenuDB()
- MatchAlgorithm()

CALLED BY

- BeveragePreferenceController()

MODULE NAME: UPDATE PROFILE

PURPOSE
Updates the main user profile with a summary of beverage habits for marketing.

PSEUDOCODE

```
BEGIN UpdateProfile
   INPUT userID, prefs
   profileData = GetUserProfile(userID)
   profileData.favorites = Merge(profileData.favorites, prefs)
   CALL SaveUserProfile(profileData)
   RETURN profile_updated
END UpdateProfile
```

INPUT PARAMETERS

- userID: String
- prefs: FormattedData

OUTPUT PARAMETERS

- status: Boolean

GLOBAL VARIABLES

- userDB: DBConnection

LOCAL VARIABLES

- profileData: UserObject

CALLS

- GetUserProfile()
- SaveUserProfile()

CALLED BY

- BeveragePreferenceController()

MODULE NAME: CONFIRM UPDATE

PURPOSE
Notifies the user that their preferences are saved and displays recommendations.

PSEUDOCODE

```
BEGIN ConfirmUpdate
   INPUT recommendations
   msg = "Preferences Saved!"
   displayData = FormatDisplay(msg, recommendations)
   CALL UIService.Show(displayData)
   RETURN confirmed
END ConfirmUpdate
```

INPUT PARAMETERS

- Recommendations: List

OUTPUT PARAMETERS

- uiStatus: Boolean

GLOBAL VARIABLES

- uiService: UIHandler

LOCAL VARIABLES

- msg: String
- displayData: UIObject

CALLS

- UIService.Show()

CALLED BY

- BeveragePreferenceController()

Member 5

Student Name: Yakendra Bam Rajawar

LMU ID: 24046664

Function Chosen: Order For Home Delivery

Context Level Diagram

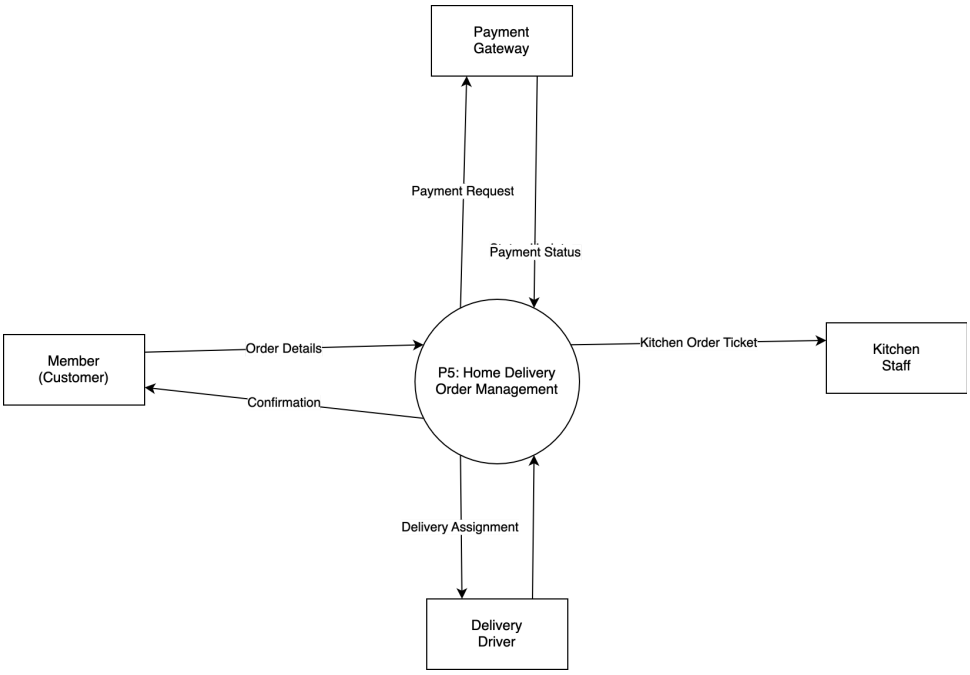

Figure 24: Member 5 DFD Level 0

Description

The Level 0 DFD represents the entire home delivery order management system as a single process (P5: Home Delivery Order Management). This high-level view shows the system's interaction with four external entities: customers who place orders and receive confirmations and status updates, kitchen staff who receive order tickets for food preparation, delivery drivers who get delivery assignments and provide status updates, and the payment gateway that processes payment requests and returns payment status. The system handles the complete flow from order placement through payment processing, food preparation, and delivery coordination, serving as the central hub that orchestrates all activities in the home delivery service.

Level 1 DFD

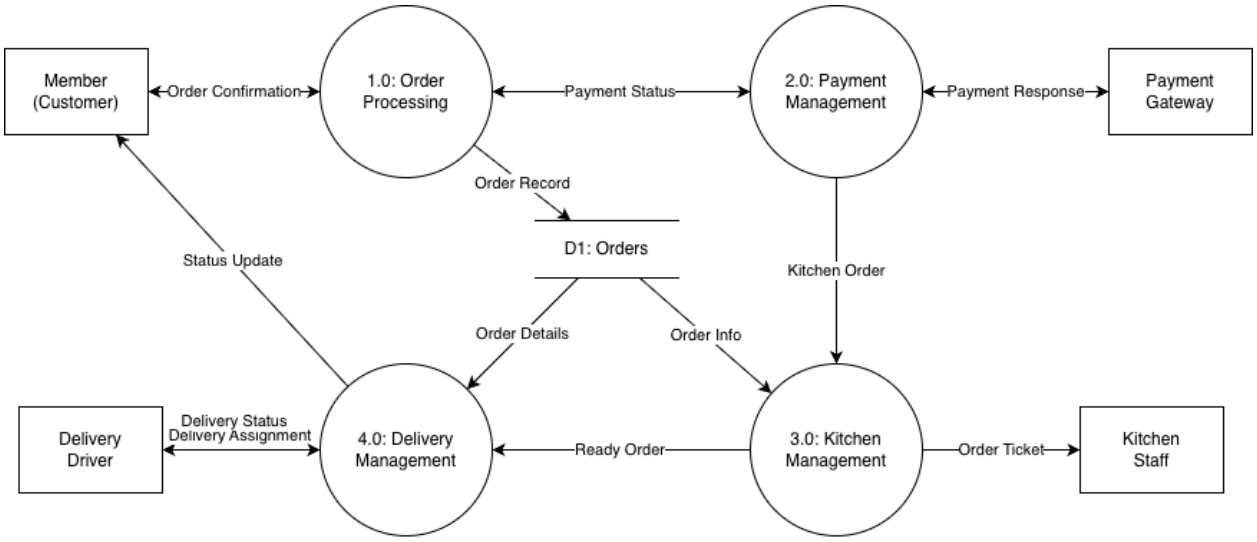

Figure 25: Member 5 DFD Level 1

Description

The Level 1 DFD decomposes the main process P5 into four interconnected sub-processes that represent the major functional areas of the home delivery system. Process 5.1 (Process Order) handles initial order intake from customers, validates order details, stores order records in the Orders database, and sends confirmations back to customers. Process 5.2 (Handle Payment) manages all payment-related activities by receiving payment information from Process Order, communicating with the external Payment Gateway for transaction processing, and coordinating with Process 5.3 upon successful payment completion. Process 5.3 (Prepare Food) retrieves order information from the Orders database, sends order tickets to Kitchen Staff for food preparation, and notifies Process 5.4 when orders are ready for delivery. Process 5.4 (Manage Delivery) coordinates the final delivery phase by assigning orders to Delivery Drivers, tracking delivery status, and providing real-time updates to customers throughout the delivery process.

Level 2 DFD

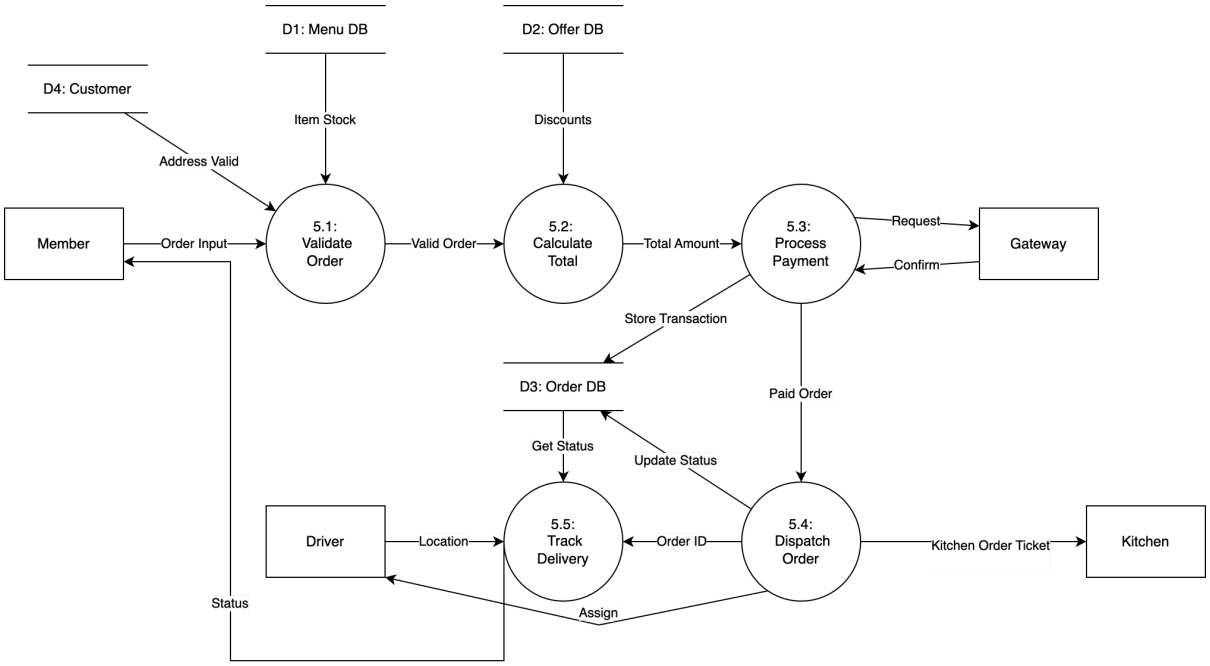

Figure 26: Member 5 DFD Level 2

Description

The Level 2 DFD would further decompose each Level 1 process into more detailed sub-processes. For example, Process 5.1 (Process Order) could be broken down into 5.1.1 (Validate Order Items) which checks item availability against menu databases and verifies customer information, 5.1.2 (Calculate Order Total) which applies pricing, discounts, taxes, and delivery charges to determine the final order amount, and 5.1.3 (Generate Order Confirmation) which creates detailed order confirmations with assigned order numbers and estimated delivery times. Similarly, other Level 1 processes would be decomposed into their constituent activities, such as breaking down Handle Payment into sub-processes for payment validation, transaction processing, and payment confirmation, or decomposing Manage Delivery into driver assignment, route optimization, and delivery tracking components.

Structure Chart

Figure 27: Member 5 Structure Chart

Module Specifications (MSpecs)

MODULE NAME: DELIVERY ORDER CONTROLLER

PURPOSE

Main controller module that coordinates the complete home delivery order process from validation to tracking.

PSEUDOCODE

```
BEGIN DeliveryOrderController
    CALL ValidateOrder(orderInput)
    IF order is valid THEN
        orderTotal = CALL CalculateTotal(orderInput)
        paymentStatus = CALL ProcessPayment(orderTotal)
        IF paymentStatus = "SUCCESS" THEN
            CALL DispatchOrder(paidOrder)
            CALL TrackDelivery(orderID)
            RETURN order_placed
        ELSE
            RETURN payment_failed
        END IF
    ELSE
        RETURN invalid_order
    END IF
END DeliveryOrderController
```

INPUT PARAMETERS

- orderInput: OrderData (items, address, customer_id)

OUTPUT PARAMETERS

- orderResult: OrderStatus (success, failed, reason)

GLOBAL VARIABLES

- sessionData: UserSession

LOCAL VARIABLES

- orderTotal: Currency
- paymentStatus: String
- orderID: String

CALLS

- Process: Validate Order
- Process: Calculate Total
- Process: Process Payment
- Process: Dispatch Order
- Process: Track Delivery

CALLED BY

- Entity: Member

MODULE NAME: VALIDATE ORDER

PURPOSE

Checks availability of items in stock and verifies if the delivery address is within the service area.

PSEUDOCODE

```
BEGIN ValidateOrder
    INPUT orderInput
    FOR EACH item IN orderInput DO
        CALL CheckStock(item)
    END FOR
    CALL CheckServiceArea(orderInput.address)
    IF all_items_available AND address_in_range THEN
        RETURN valid
    ELSE
        RETURN invalid_reason
    END IF
END ValidateOrder
```

INPUT PARAMETERS

- orderInput: OrderData

OUTPUT PARAMETERS

- validationResult: Boolean

GLOBAL VARIABLES

- menuDatabase: DBConnection

LOCAL VARIABLES

- itemAvailability: Boolean
- serviceAreaValid: Boolean

CALLS

- Process: Check Stock
- Process: Check Service Area
- Data Store: Menu Database

CALLED BY

- Process: Delivery Order Controller

MODULE NAME: CHECK STOCK

PURPOSE

Verifies quantity of a single item against current inventory.

PSEUDOCODE

```
BEGIN CheckStock
    INPUT itemID, qty
    currentStock = QueryMenuDB(itemID)
    IF currentStock >= qty THEN
        RETURN available
    ELSE
        RETURN out_of_stock
    END IF
END CheckStock
```

INPUT PARAMETERS

- itemID: String
- qty: Integer

OUTPUT PARAMETERS

- status: Boolean

GLOBAL VARIABLES

- menuDatabase: DBConnection

LOCAL VARIABLES

- currentStock: Integer

CALLS

- Data Store: Menu Database

CALLED BY

- Process: Validate Order

MODULE NAME: CHECK SERVICE AREA

PURPOSE

Validates if the provided zip code or coordinates are within the delivery radius.

PSEUDOCODE

```
BEGIN CheckServiceArea
    INPUT address
    distance = CalculateDistance(restaurantLoc, address)
    IF distance <= MAX_DELIVERY_RADIUS THEN
        RETURN true
    ELSE
        RETURN false
```

END IF

END CheckServiceArea

INPUT PARAMETERS

- address: AddressData

OUTPUT PARAMETERS

- isServicable: Boolean

GLOBAL VARIABLES

- config: SystemConfig

LOCAL VARIABLES

- distance: Float

CALLS

- System: GIS Service

CALLED BY

- Process: Validate Order

MODULE NAME: CALCULATE TOTAL

PURPOSE

Computes the final payable amount including subtotal, discounts, delivery fees, and taxes.

PSEUDOCODE

```
BEGIN CalculateTotal
    INPUT validatedOrder
    subtotal = CALL SumSubtotal(validatedOrder.items)
    discount = CALL ApplyDiscount(subtotal, validatedOrder.coupon)
    fee = CALL AddDeliveryFee(validatedOrder.address)
    total = subtotal - discount + fee + TAX
    RETURN total
END CalculateTotal
```

INPUT PARAMETERS

- validatedOrder: OrderData

OUTPUT PARAMETERS

- finalTotal: Currency

GLOBAL VARIABLES

- taxRate: Float

LOCAL VARIABLES

- subtotal: Currency
- discount: Currency
- fee: Currency

CALLS

- Process: Sum Subtotal

- Process: Apply Discount
- Process: Add Delivery Fee

CALLED BY

- Process: Delivery Order Controller

MODULE NAME: PROCESS PAYMENT

PURPOSE

Handles the transaction logic for the order via the payment gateway.

PSEUDOCODE

```
BEGIN ProcessPayment
    INPUT amount, cardDetails
    CALL GatewayInterface(amount, cardDetails)
    response = GetGatewayResult()
    IF response = "APPROVED" THEN
        CALL RecordTransaction(response)
        RETURN "SUCCESS"
    ELSE
        RETURN "FAILED"
    END IF
END ProcessPayment
```

INPUT PARAMETERS

- amount: Currency
- cardDetails: PaymentInfo

OUTPUT PARAMETERS

- status: String

GLOBAL VARIABLES

- paymentGateway: ExternalConnection

LOCAL VARIABLES

- response: GatewayResponse

CALLS

- Entity: Payment Gateway
- Process: Record Transaction

CALLED BY

- Process: Delivery Order Controller

MODULE NAME: DISPATCH ORDER

PURPOSE

Manages the post-payment logistics: sending order to kitchen and assigning a driver.

PSEUDOCODE

```
BEGIN DispatchOrder
    INPUT paidOrder
    CALL GenerateKOT(paidOrder)
    driverID = CALL AssignDriver()
    CALL NotifyDriver(driverID, paidOrder)
    RETURN dispatch_success
```

END DispatchOrder

INPUT PARAMETERS

- paidOrder: OrderData

OUTPUT PARAMETERS

- dispatchStatus: Boolean

GLOBAL VARIABLES

- orderDatabase: DBConnection

LOCAL VARIABLES

- kotID: String
- driverID: String

CALLS

- Process: Generate KOT
- Process: Assign Driver
- Process: Notify Driver
- Data Store: Order Database

CALLED BY

- Process: Delivery Order Controller

MODULE NAME: ASSIGN DRIVER

PURPOSE

Finds an available driver near the restaurant and assigns them to the order.

PSEUDOCODE

```
BEGIN AssignDriver
    availableDrivers = QueryDriverDB("STATUS='AVAILABLE'")
    selectedDriver = FindNearest(availableDrivers)
    RETURN selectedDriver.ID
END AssignDriver
```

INPUT PARAMETERS

None

OUTPUT PARAMETERS

- driverID: String

GLOBAL VARIABLES

- driverDatabase: DBConnection

LOCAL VARIABLES

- availableDrivers: List
- selectedDriver: DriverObject

CALLS

- Data Store: Driver Database

CALLED BY

- Process: Dispatch Order

MODULE NAME: TRACK DELIVERY

PURPOSE

Provides real-time status updates to the customer.

PSEUDOCODE

```
BEGIN TrackDelivery
    INPUT orderID
    currentStatus = GetOrderStatus(orderID)
    CALL NotifyCustomer(currentStatus)
    RETURN tracking_active
END TrackDelivery
```

INPUT PARAMETERS

- orderID: String

OUTPUT PARAMETERS

- status: String

GLOBAL VARIABLES

- orderDatabase: DBConnection

LOCAL VARIABLES

- currentStatus: String

CALLS

- Data Store: Order Database
- Process: Notify Customer

CALLED BY

- Process: Delivery Order Controller

Conclusion

Gokyo Bistro Online management system is a web based system aimed at eliminating the shortcomings of the manual management system of running restaurants by providing a centralized and automated system running on the web. The system provides necessary restaurant operations like user authentication, reservation of tables with advance payment and cancellation terms, home delivery services exclusive to the members, menu and offers management, billing and receipt creation, customer feedback, and administrative reporting.

The project highly emphasizes on usability, security, and performance. The role-based access control, integration of secure payment gateway, data validation and encrypted communication are features that make sure that user data and financial transactions are managed safely and effectively. Modular and maintainable principles are also based in the system design and the system can be updated and scaled in the future without substantial input of the structure.

Comprehensively, the Gokyo Bistro Online Management System will increase the efficiency of operations, customer experience, and enhance informed decision-making by referring to the correct data handling and reporting. The experienced knowledge acquired during the present project, such as requirement analysis, system design, and compliance with the principles of software engineering, has a good ground to be used in the creation of the real-world and scalable information systems in the future.

References

GeeksforGeeks (2024) *Structure Charts in Software Engineering*. Available at: https://www.geeksforgeeks.org/structure-charts-in-software-engineering/ (Accessed: 2 January 2026).

IEEE (1998) *IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications*. New York: Institute of Electrical and Electronics Engineers. (Accessed: 20 December 2025).

Lucidchart (2024) *What is a Data Flow Diagram*. Available at: https://www.lucidchart.com/pages/data-flow-diagram (Accessed: 21December 2025).

Perforce Software (2024) *How to Write a Software Requirements Specification (SRS) Document*. Available at: https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document (Accessed: 21 December 2026).