Task 4(b): Architectural Design of the Whole System

Overview of the System Architecture

The CloudTables system consists of multiple subsystems designed to fulfill the needs of different user groups:

1. **Customer Subsystem:** Manages features that interact with customers, including ordering food, making payments, scheduling tables, and writing reviews.
2. **Service Subsystem:** Helps restaurant employees keep track of table and order statuses in real time while managing reservations, processing orders, and processing payments.
3. **Manager Subsystem:** Gives restaurant managers the means to create reports on operational performance, monitor employees, and change menus.
4. **Operation Subsystem:** Helps system operators with event planning, multi-tenant operations management, and system health monitoring.

An API Gateway serves as the central point of entry for all requests connecting the subsystems. To guarantee safe access, offer data insights, and transmit real-time updates, the system makes use of shared services including authentication, analytics, and notification.

Each subsystem's distributed database is part of the Database Layer, which guarantees data scalability and integrity.

Detailed Explanation of Subsystems

1. CloudTables-Customer Subsystem

- **Purpose:**
    - AAllows users to communicate with the system for orders, payments, reviews, and bookings.
- **Components:**
    - Reservation: Allows customers to book tables.
    - Order Food: Allows customers to order food and drinks.
    - Bill Payment: Handles all of customer payments.
    - Customer Review: Collects any reviews and ratings.
- **Interactions:**
    - Enables real-time booking status retrieval, order processing, and payment handling by connecting to the Service Subsystem.
    - Sends customer reviews to the **Analytics Service** for insights.
- **APIs:**
    - createBooking(customerID, tableID, timeSlot)
    - placeOrder(orderDetails)

- submitReview(reviewDetails)

2. CloudTables-Service Subsystem

- **Purpose:**
  - Helps restaurant servers oversee operations and interactions with patrons.
- **Components:**
  - Booking Service: Manages table bookings and updates statuses.
  - Order Service: Handles the food and drink orders.
  - Payment Service: Processes all payments securely.
  - State Tracker Service: Tracks the real-time status of tables and orders.
- **Interactions:**
  - Updates the database with the status of booking requests received from the Customer Subsystem.
  - Exchanges operational data and retrieves menu updates via communication with the Manager Subsystem.
  - Sends data to the **Analytics Service** for generating reports.
  - Sends information to employees and clients via the Notification Service.
- **APIs:**
  - updateTableStatus(tableID, newStatus)
  - processOrder(orderDetails)
  - generateBill(orderID)

3. CloudTables-Manager Subsystem

- **Purpose:**
  - Gives restaurant managers the resources they need to keep an eye on and update restaurant operations.
- **Components:**
  - Menu Management: Allows managers to update food menus.
  - Staff Management: Tracks staff schedules and roles.
  - Reports Service: Generates performance reports.
- **Interactions:**
  - Retrieves booking and order data from the **Service Subsystem**.
  - Coordinates ordering features by sending menu updates to the Service Subsystem.
  - Utilizes the **Analytics Service** to generate reports.
- **APIs:**
  - updateMenu(menuDetails)
  - viewReports(criteria)

4. CloudTables-Operation Subsystem

- **Purpose:**

  Helps system operators oversee events and operations involving multiple tenants.

- **Components:**
  - Tenant Management: Activates or deactivates restaurant accounts.
  - Promotions Service: Organizes any cross-restaurant promotional events.
  - System Monitoring: Tracks system health and performance.
- **Interactions:**
  - Retrieves order and booking information for event planning from the Service Subsystem.
  - Sends data to the **Analytics Service** for generating system-wide insights.
- **APIs:**
  - activateTenant(tenantID)
  - createPromotion(promotionDetails)

5. Shared Services

Authentication Service

- **Purpose:**
  - Provides secure access to all of the subsystems and users.
- **APIs:**
  - login(userID, password)
  - validateToken(token)

Analytics Service

- **Purpose:**
  - Aggregates data from all the subsystems for insights and reports.
- **APIs:**
  - collectData(dataType)
  - generateReport(criteria)

Notification Service

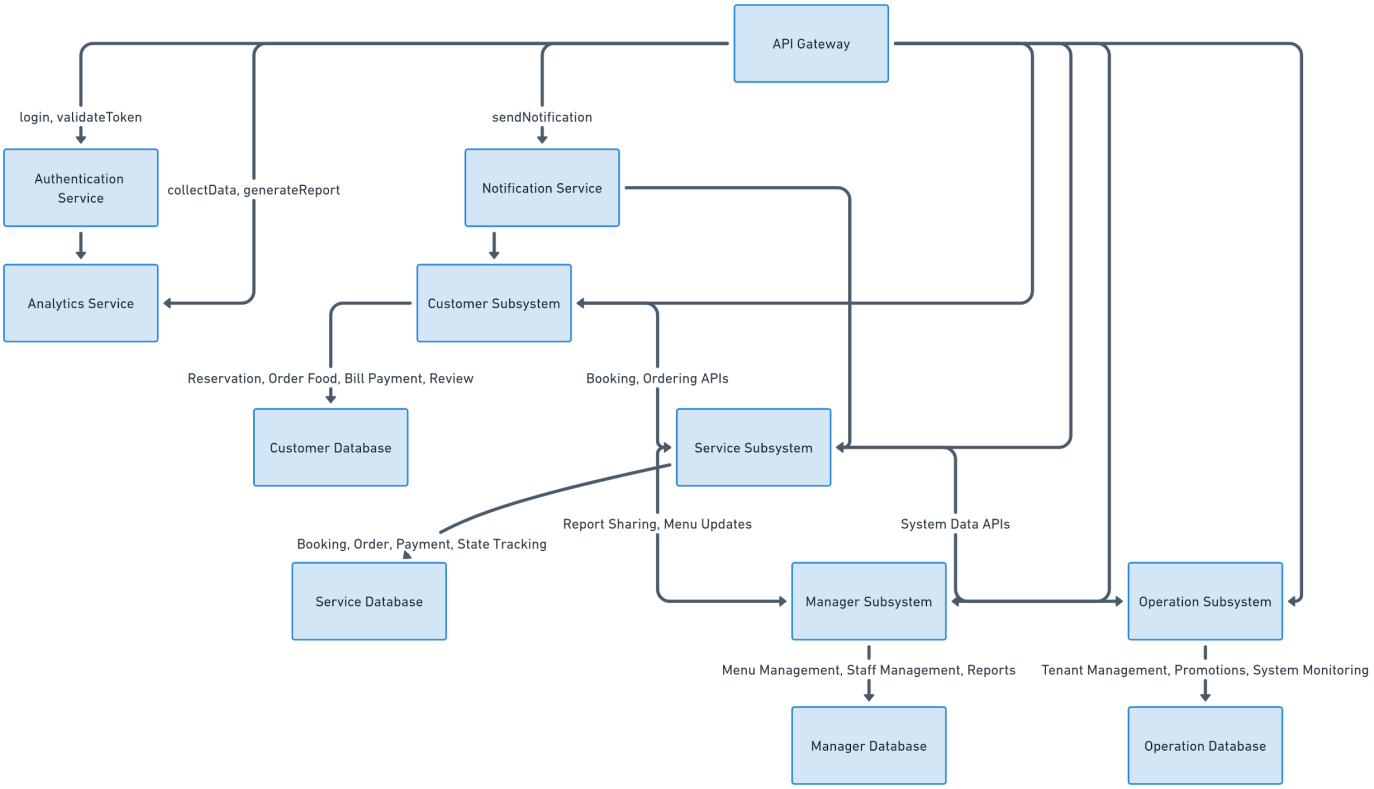
- **Purpose:**
    - Sends alerts and updates to customers and staff.
- **APIs:**
    - sendNotification(eventType, message)

6. API Gateway

- **Purpose:**
    - Acts as a centralized router for all requests.
- **Responsibilities:**
    - Directs requests to the appropriate subsystem or service.
    - Handles load balancing and security.

7. Database Layer

- **Purpose:**
    - Stores data for each subsystem while maintaining real-time synchronization.
- **Databases:**
    - Customer DB: Stores customer profiles, bookings, and reviews.
    - Service DB: Tracks table statuses, orders, and payments.
    - Manager DB: Stores menu and staff data.
    - Operation DB: Maintains tenant and system-level configurations.



Advantages of the Whole System Architecture

1. Modularity:
    - Because each component functions independently, scalability and maintenance are made simpler.
2. Scalability:
    - Depending on usage demands, shared services and subsystems can be scaled.
3. Fault Tolerance:
    - The operation of other subsystems is unaffected when one fails.
4. Security:
    - All subsystems are subject to stringent access controls enforced by the Authentication Service.
5. Flexibility:
    - It is possible to introduce new shared services or subsystems without interfering with the current system.
6. Efficiency:
    - Shared services that are centralised decrease redundancy and enhance the system performance in general.