# YUMMY – ONLINE FOOD ORDERING AND DELIVERY SYSTEM

## LAB 4 REPORT – MICROSERVICES DECOMPOSITION & COMMUNICATION

## I. Summary

L Lab 4 marks the structural transformation of the Yummy system from a monolithic Layered Architecture (implemented in Lab 3) to a Microservices Architecture. The main content of the lab focuses on decomposition by business capability, thereby identifying independent microservices, clear service boundaries, and communication mechanisms between services throughout the system.

The objective of Lab 4 is to help the group understand and apply the core principles of microservices architecture, including correctly defining the scope of responsibility for each service, designing service contracts via RESTful APIs, and selecting appropriate inter-service communication strategies (synchronous and asynchronous communication). The primary business domains of Yummy, such as User Management, Food Catalog, Order Management, Cart, Payment, Delivery/Driver Tracking, and Notification, are decomposed into separate microservices, with each service owning its own data to ensure independence and scalability.

Additionally, Lab 4 focuses on high-level architectural modeling through the C4 Model – Level 1 (System Context), which clearly illustrates the Yummy system boundaries, external actors (users, drivers, restaurants), and third-party systems such as payment gateways or notification services. Through this model, the communication strategy between system components is clearly analyzed, distinguishing between flows requiring immediate response and tasks that can be processed asynchronously to optimize performance and user experience.

Overall, Lab 4 provides a critical design foundation for developing the Yummy system towards microservices, creating a premise for flexible scalability, independent deployment, and efficient operation in subsequent development phases.

## II. Technology and tool installation

| Tool | Purpose | Installation/Setup Instructions |
|---|---|---|
| Google Docs<br>Microsoft Words | Drafting and storing requirement documents and ASRs. | Standard word processing software. Google Docs is used online; Microsoft Word requires local installation. |
| draw.io | Creating System Context Diagram and Microservice Decomposition Diagram (Logical View) | Accessed online via web browser; no local installation required. |
| Whiteboard/Pen & Paper | Initial brainstorming and sketching of layers/components | Standard brainstorming tools |

*Table 4.1. List of supporting tools used in Lab 4*

## III. Lab specific section: Microservices Decomposition & Communication

### 1. Decomposition by Business Capability

| External systems | Purpose | Interactions | Reason |
|---|---|---|---|
| Payment Gateway System (e.g: VNPay, MoMo,..) | Process online payments for food orders. | - Send payment requests.<br>- Receive payment status (success/failure) | Yummy does not handle sensitive payment data directly. |
| Mapping & Location Service (e.g: Google Maps API) | Provide location-related services. | - Calculate distance between restaurant and customer | Accurate delivery estimation and map visualization. |

| | | | |
|---|---|---|---|
| | | - Display restaurant and delivery locations | |
| Notification Service (e.g: SMS Gateway, Email Service) | Send notifications to users and drivers. | - Order status updates<br>- Delivery notifications | Improve user experience and real-time communication. |

*Bảng 1Table 4.2: External Systems and Their Roles in the Yummy System*

| **Business Capability** | **Proposed Microservice** | **Data Owned (Entities)** |
|---|---|---|
| User Management | User Service | User Profiles, Authentication Credentials. |
| Catalog Management | Food Service | Product Details, Pricing, Image URLs. |
| Order Management | Order Service | Orders, Order Items, Business Status. |
| Pre-Purchase | Cart Service | Cart Content, Session Status. |
| Fulfillment & Tracking | Delivery Service | Delivery Address, Delivery Status, Driver Location. |
| Payment Processing | Payment Service | Payment Transactions, Payment Status, Transaction Codes. |
| Notification Management | Notification Service | Notification Content, Delivery Status, Notification History. |

*Bảng 2Table 4.3: Business Capabilities and Proposed Microservices.*

### 2. Define Service Contracts (API Endpoints) for key services.

| **Endpoint** | **Method HTTP** | **Description** | **Response/ Return Data** |
|---|---|---|---|
| **Food Service** | | | |
| `/api/foods/{id}` | GET | Get detailed information of a food item | Food object (name, price, description) |
| `/api/foods` | GET | Search/List food items by criteria | List of Food (summary form) |
| `/api/foods` | POST | Add a new food item | Result notification |
| `/api/foods/{id}` | PUT | Update food information | Updated Food object |
| `/api/foods/{id}` | DELETE | Delete a food item | Result notification |
| **Order Service** | | | |
| `/api/orders/{id}` | GET | Get detailed information of an order | Order object |
| `/api/orders` | POST | Add a new food item | Result notification |
| `/api/orders/{id}` | PUT | Update order information | Updated Order object |
| `/api/orders/{id}` | DELETE | Delete an order | Result notification |
| **Cart Service** | | | |

| | | | |
|---|---|---|---|
| /api/cart | GET | Get the current cart of the user | Cart object (list of food items, price, total, ..) |
| /api/cart/items | POST | Add a food item to the cart | Result notification |
| /api/cart/items/{itemId} | PUT | Update item quantity in the cart | Updated Cart object |
| /api/cart/items/{itemId} | DELETE | Remove an item from the cart | Updated Cart object |
| **User Service** | | | |
| /api/users | POST | User registration | User profile |
| /api/users/login | POST | Login | Access token / user info |
| /api/users/{id} | GET | Get user information | User profile |
| /api/users/{id} | PUT | Update profile | Updated profile |
| /api/users/{id}/addresses | GET | Get delivery addresses | Address list |
| **Delivery Service** | | | |
| /api/deliveries | POST | Create a delivery request for an order | Delivery information |
| /api/deliveries/{orderId} | GET | Get delivery status | Status + ETA |
| /api/deliveries/{orderId}/tracking | GET | Track delivery location | Current location |
| **Payment Service** | | | |
| /api/payments | POST | Initialize payment for an order | Payment URL / token |
| /api/payments/{id} | GET | Check payment status | Payment status |
| /api/payments/callback | POST | Receive callback from Payment Gateway | Payment processing result |
| | | | |
| | | | |
| | | | |
| | | | |
| **Notification Service** | | | |
| /api/notifications | POST | Send notification | Notification result |
| /api/notifications/{userId} | GET | Get user notification history | Notification list |

*Bảng 3Table 4.4: API Endpoints Specification.*

## 3. Design the system's High-Level Communication Strategy (Synchronous vs. Asynchronous).

### 3.1. Synchronous communication Needs

In Yummy App, synchronous communication is employed for critical, user-driven operations where the calling component must wait for a result before proceeding. This model ensures data accuracy and provides the immediate feedback necessary for a smooth, uninterrupted user flow.

| Interaction | Service/Component | Communication Type | Rationale |
|---|---|---|---|
| User authentication and profile retrieval | API Gateway → User Service | Synchronous (HTTP) | Immediate feedback is required to authenticate users and display personal information during login and account management. |
| Browsing restaurants and menus | API Gateway → Food Service / Restaurant Service | Synchronous (HTTP) | Users expect real-time data when viewing available restaurants, menus, and food details. |
| Cart validation and inventory checking | Cart Service → Inventory Service | Synchronous (HTTP) | The system must verify product availability instantly before allowing users to proceed to checkout. |
| Order creation and confirmation | Client → Order Service | Synchronous (HTTP) | Users must receive immediate confirmation that their order has been successfully created. |
| Price calculation and promotion validation | Order Service → Pricing/Promotion Service | Synchronous (HTTP) | Accurate pricing information is required in real time before completing payment. |

*Table 4.5: Inter-Service Communication Strategy for Yummy System.*

### 3.2. Asynchronous Communication Needs

Asynchronous communication decouples services to handle operations that do not require immediate feedback. By allowing background processing independent of the main user action, this approach significantly enhances system scalability, fault tolerance, and the ability to handle high traffic.

| Interaction | Service/Component | Rationale |
|---|---|---|
| Order event propagation | Order event propagation | Order event propagation |
| Order Service → Event Bus → Other Services | Order Service → Event Bus → Other Services | Order Service → Event Bus → Other Services |
| Asynchronous | Asynchronous | Asynchronous |
| After an order is created, multiple services (Notification, Analytics, Restaurant Dashboard) are notified without blocking the order flow. | After an order is created, multiple services (Notification, Analytics, Restaurant Dashboard) are notified without blocking the order flow. | After an order is created, multiple services (Notification, Analytics, Restaurant Dashboard) are notified without blocking the order flow. |
| Centralized Notification Handling Various Services | Centralized Notification Handling Various Services | Centralized Notification Handling Various Services |

*Table 4.6: Asynchronous Communication Strategy Between Services.*

## 4. Model the system using the C4 Model (Level 1: System Context).

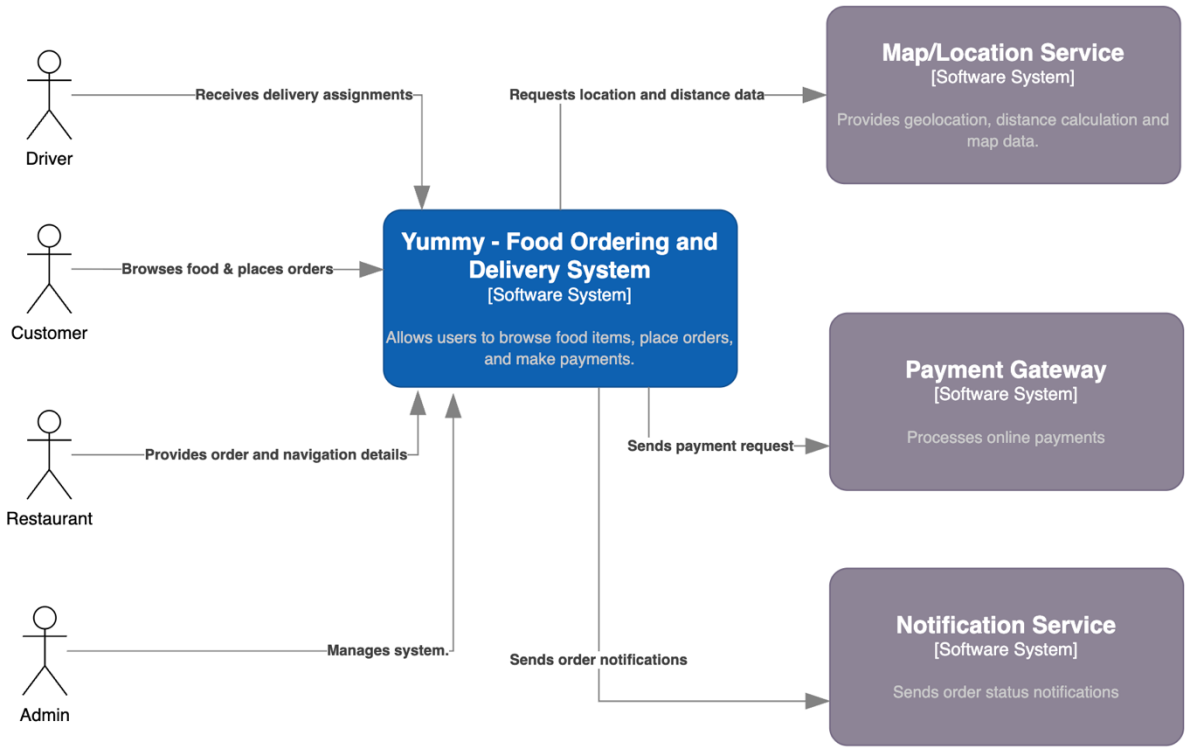


*Figure 4.1: System Context Diagram for Yummy (C4 - Level 1).*