YUMMY – ONLINE FOOD ORDERING AND DELIVERY SYSTEM
LAB 2 REPORT – LAYERED ARCHITECTURE DESIGN (LOGICAL VIEW)

I. Summary

Lab 2 focuses on designing the Layered Architecture for the Yummy online food ordering and delivery system, aiming to describe the system's static structure and the relationships between its key components. This is a crucial intermediate step before proceeding to detailed design and system implementation in subsequent phases.

In this lab, the Yummy system is organized into four main layers following the Layered Architecture model, including: Presentation Layer, Business Logic Layer, Persistence Layer, and Data Layer. Each layer has distinct responsibilities and adheres to strict layering principles, where each layer is only permitted to interact directly with the layer immediately below it. The Presentation Layer is responsible for receiving user requests via the mobile application, handling authentication, and rendering the user interface. The Business Logic Layer contains core business rules such as order processing, dish availability checks, and order coordination. The Persistence Layer handles data retrieval and manipulation through repositories, while the Data Layer represents the physical database management system.

Besides defining the layers, the lab also focuses on analyzing the data flow for a typical scenario in the Yummy system, such as a user viewing food details. The processing flow is described sequentially from the Presentation Layer down to the Data Layer and vice versa, clarifying how the system processes requests and returns results to the end-user.

Furthermore, Lab 2 requires identifying specific components for a representative system function, typically the "View Food Details" feature. Components such as FoodController, FoodService, and FoodRepository are allocated respectively to the Presentation, Business Logic, and Persistence layers. Finally, the entire logical architecture of the system is modeled using a UML Component Diagram, clearly illustrating the components, provided/required interfaces, and dependency relationships between the layers within the Yummy system.

II. Technology and tool installation

Tool	Purpose	Installation/Setup Instructions
Google Docs Microsoft Words	Drafting and storing requirement documents and ASRs.	Standard word processing software. Google Docs is used online; Microsoft Word requires local installation.
PlantUML	Creating professional UML diagrams (Use Case Diagram, Component Diagram, etc.).	Accessed online via web browser; no local installation required.
Whiteboard/Pen & Paper	Initial brainstorming and sketching of layers/components	Standard brainstorming tools

Table 2.1. List of supporting tools used in Lab 2.

III. Defining Layers and Responsibilities

1. Defind the Four Layers

The system is designed following the Layered Architecture pattern, which organizes the application into distinct layers, each with a clear responsibility and well-defined boundaries. This architectural approach helps improve maintainability, scalability, and separation of concerns, allowing each layer to evolve independently with minimal impact on others. In this design, the system is structured into four main layers: the Presentation Layer, Business Logic Layer, Persistence Layer, and Data Layer. Each layer plays a specific role in handling user interaction, business processing, data access, and data

storage. The following table summarizes the purpose, responsibilities, and main artifacts produced by each layer in the system.

Layer	Purpose / Responsibility	Output / Artifact
1. Presentation Layer (Client & Server)	Client (Flutter): Handles user interactions, UI navigation, state management, and rendering the user interface.	Flutter UI Components (Screens, Widgets, State Management)
	Server (API): Receives HTTP requests from client applications, performs request validation and routing, invokes corresponding business services, and returns JSON responses.	REST API Controllers / Endpoints (JSON-based APIs)
2. Business Logic Layer (Service/Domain)	Handling core operations of the system; check business rules; coordinate data access through Persistence Layer; independent of interface and database.	Services / Managers (e.g., OrderService, FoodService)
3. Persistence Layer (Data Access)	Performs data retrieval, maps business objects to stored data, hides database details, and provides CRUD operations for the Business Logic Layer.	Repositories / DAOs (OrderRepository, FoodRepository)
4. Data Layer	Store physical data, manage data structures, and ensure data integrity.	Database Schema (Tables, Indexes)

Table 2.2: Description of Layers in the Yummy Layered Architecture.

2. Define Data Flow

Scenario: Customers select an item to view details.

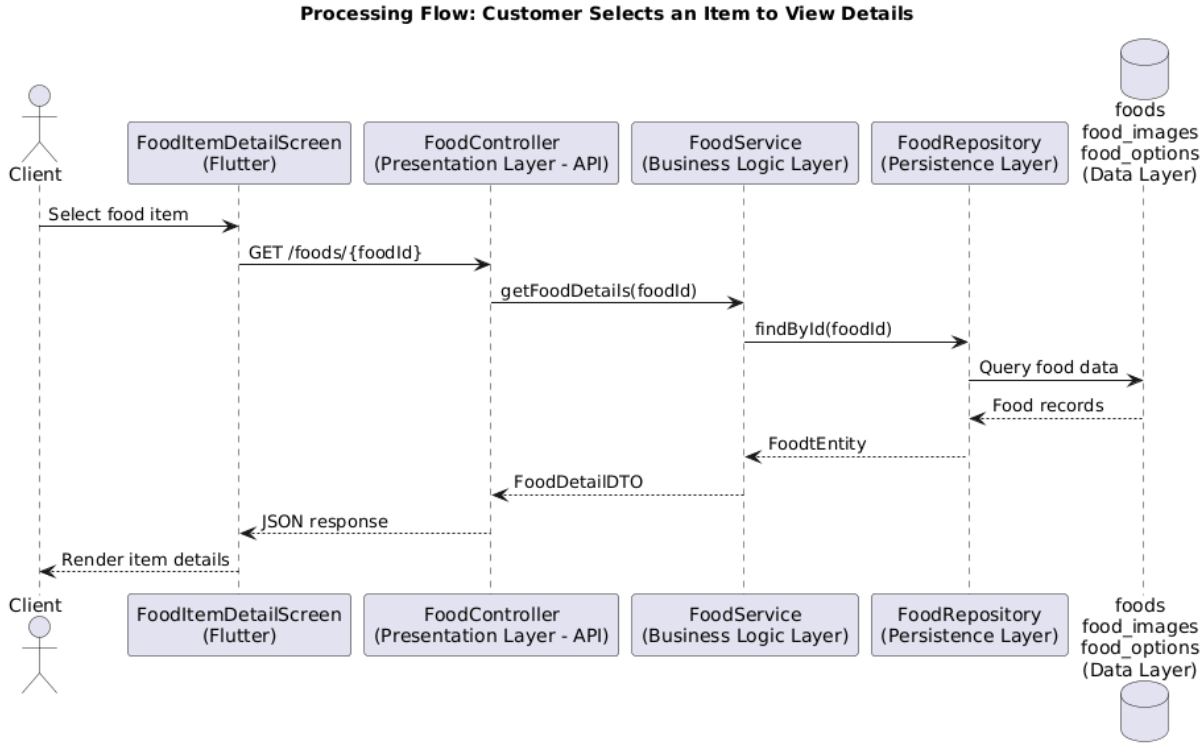

Figure 2.1: Sequence diagram for viewing food item details

Detailed Description:

Step	Actor/Component	Layer	Action/Description	Input	Output
1	Client	Client	Customer selects a food item from the list on the user interface	Food item selected	Food ID
2	FoodItemDetailScreen (Flutter)	Presentation Layer (Client)	Sends a request to retrieve detailed information of the selected food item	Food ID	HTTP GET request
3	FoodController	Presentation Layer (Server – API)	Receives HTTP GET request and routes it to the corresponding business service	GET `/foods/{foodID}`	Method call
4	FoodService	Business Logic Layer	Handles business logic to retrieve food details	`foodID`	Request to repository
5	FoodRepository	Persistence Layer	Queries the database to retrieve food information by ID	`foodID`	`FoodEntity`
6	Database	Data Layer	Executes query and returns matching food records	SQL/Query	Food records
7	FoodRepository	Persistence Layer	Maps database records to domain entity	Food records	`FoodEntity`
8	FoodService	Business Logic Layer	Converts FoodEntity into FoodDetailDTO	`FoodEntity`	`FoodDetailDTO`
9	FoodController	Presentation Layer (Server – API)	Wraps DTO into JSON response and sends it back to client	`FoodDetailDTO`	JSON response
10	FoodItemDetailScreen	Presentation Layer (Client)	Receives JSON response and renders food detail information on UI	JSON response	Rendered food details

Table 2.3: Processing Flow Description for Viewing Food items details.

IV. Identification Components (Food Item Detail Feature)

1. Identify Components

The objective of this section is to decompose the Food Item Detail function into specific components, clearly distributed across the three main layers of the layered architecture: Presentation Layer, Business Logic Layer, and Persistence Layer. Correctly identifying the role of each component ensures the Separation of Concerns and strict adherence to the principles of Layered Architecture.

1.1. Layer 1 – Presentation Layer (Client)

Component: FoodItemDetailScreen

FoodItemDetailScreen is the component responsible for displaying the food detail interface to the user. This component receives the `foodId` via a route parameter from the URL (e.g., `/foods/{id}`), then triggers the data retrieval process through a state management mechanism (Riverpod).

This component handles the rendering of UI elements such as food images, names, descriptions, prices, options (size, toppings, sugar level, ice level), and availability status. Additionally, FoodItemDetailScreen processes user interactions such as selecting product options, changing quantities, and adding items to the cart. Communication with the Business Logic Layer is performed indirectly through providers, ensuring that the Presentation Layer does not directly depend on business logic implementation details.

1.2. Layer 2 – Business Logic Layer (Service/Domain)

Component: FoodService

FoodService is responsible for handling the entire business logic related to the food detail view function. This component receives requests from the Presentation Layer via a `foodId`, then executes business rules such as checking the dish's availability, validating prices and accompanying options, as well as calculating the final price based on user selections.

Furthermore, FoodService plays an orchestration role, coordinating calls to the Persistence Layer to retrieve the necessary data, while simultaneously preparing the data in a suitable format to be returned to the Presentation Layer without exposing the internal data structure of the system.

1.3. Layer 3 – Persistence Layer (Data Access)

Component: FoodRepository

FoodRepository is the component responsible for data access. This component translates requests from the Business Logic Layer into corresponding database queries, performs CRUD operations, and maps stored data into domain entities.
In addition to the primary food item data, FoodRepository is also responsible for retrieving related data, such as product images and options, ensuring that the Business Logic Layer does not need to concern itself with data storage details.

2. Define Interface

2.1. Interface between Presentation Layer and Business Logic Layer

```
public Food getFoodDetails(String productId)
```

This method is provided by the Business Logic Layer for the Presentation Layer to query detailed information of a product based on `productId`. The Presentation Layer only interacts with the Service through this interface and does not care about how data is stored or retrieved.

2.2. Interface between Business Logic Layer andPersistence Layer

```
public FoodEntity findById(String foodId)
```

This method is defined in the Persistence Layer to retrieve product data from a relational database based on the unique identifier `productId`. The data retrieval is performed through data access

mechanisms (such as an ORM or query statements), which are fully encapsulated within the Persistence Layer.

V. Component Diagram Modeling

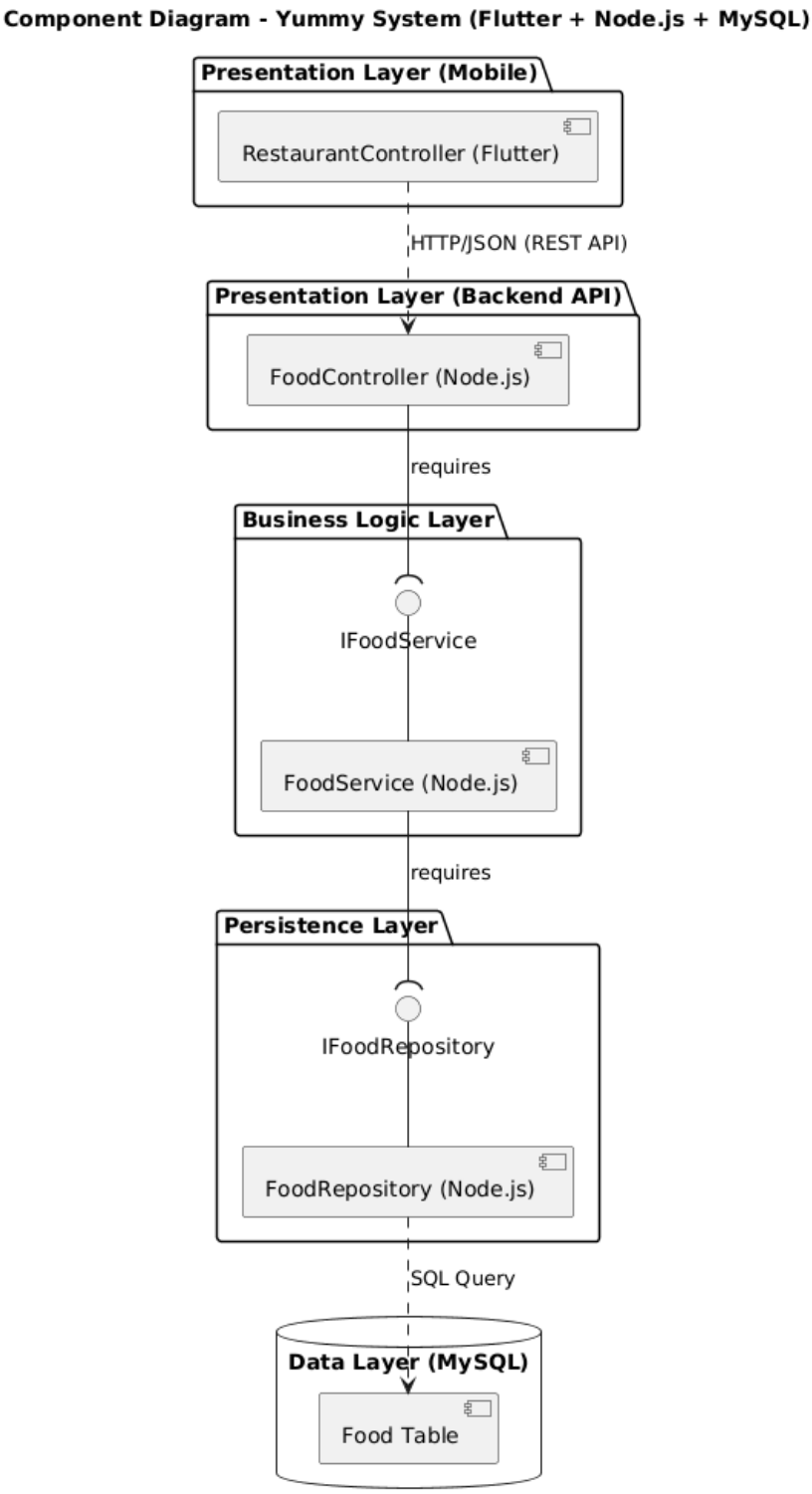

Figure 2.2: Layered Architecture of the Food Service system

VI. Conclusion

In this lab, the Layered Architecture of the Yummy system was designed and analyzed from a logical perspective. The system was decomposed into four main layers: Presentation Layer, Business Logic Layer, Persistence Layer, and Data Layer, each with clearly defined responsibilities and strict dependency rules. This structure ensures a clear separation of concerns, helping to reduce system complexity and improve maintainability.

Through the identification of key components for the Food Item Detail feature, the interactions between layers were clearly specified, along with the interfaces that connect them. The processing flow was also described in detail, illustrating how a user request is propagated through the layers, from the client interface to data storage and back to the presentation layer. This analysis demonstrates how business rules are isolated from user interface and data access logic.

Overall, Lab 2 provides a solid architectural foundation for the Yummy application. The layered design supports scalability, modifiability, and ease of testing, and serves as an essential basis for the subsequent implementation phase in Lab 3.