ARCHITECTURAL PATTERN & VIEW

E-FARM

TEAM MATES

MD ASIF MAHMUD
MUH2025004M
Year -3 Term-2

IMTIAZ CHOWDHURY
MUH2025027M
Year -3 Term-2

MAHAMUDUL HASAN TALUKDER
MUH2025028M
Year -3 Term-2

RUBYA RASHED
MUH2025014M
Year -3 Term-2

MEHEDI HASAN RAFI
MUH2025032M
Year -3 Term-2

07-12-2023

IIT, NSTU, E-Farm

PRESENTED TO

Dipok Chandra Das

Assistant Professor
Institute of Information Technology
Noakhali Science and Technology University

07-12-2023

IIT, NSTU, E-Farm

TABLE OF CONTENT

What's ARCHITECTURAL DESCRIPTION?

An **architectural description** in software engineering refers to a comprehensive representation of a system's architecture.

This description is crucial for understanding, communicating, and maintaining the system's design throughout its lifecycle.

E-FARM

VIEW VS VIEWPOINT

- A **view** is a representation of all or part of an architecture, from the perspective of one or more concerns which are held by one or more of its stakeholders.

- A **viewpoint** is *a collection of patterns*, templates and conventions for constructing one type of view.

E-FARM

EXAMPLE

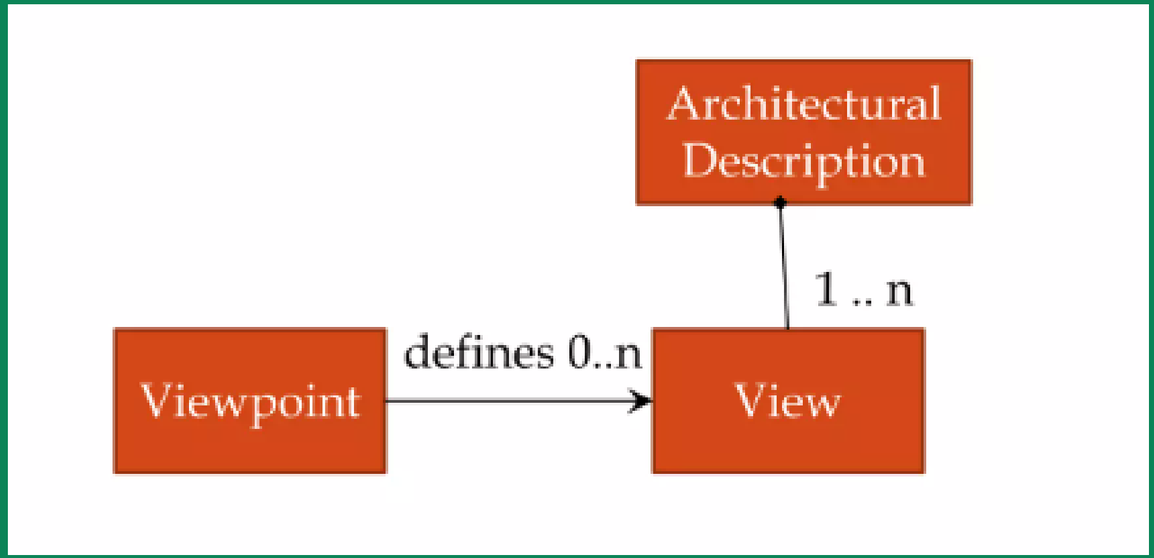
- **Viewpoint**: *Security*
  - **View**: Security view of the system's architecture showing access control, encryption methods, and authentication mechanisms.

- **Viewpoint**: *Deployment*
  - **View**: Deployment view illustrating how components/modules are distributed across servers or hardware, depicting the physical layout of the system.

view vs viewpoint

E-FARM

VIEWPOINT, VIEW & ARCHITECTURAL DESCRIPTION



- An architectural description is a collection of one or more views.
- A viewpoint is used to define (the structure and content of) zero or more views.
- The structure and content of a particular view is defined by exactly one viewpoint.

PROJECT ARCHITECTURE VIEWS E-FARM

Functional/Logic View:

- Purpose: Describes the functional behavior of the system and its components.

- Components:
  - User Interface (UI)
  - Marketplace
  - Equipment Management
  - Agro Solution and Consultancy
  - Data Management
  - Security

PROJECT ARCHITECTURE VIEWS E-FARM

Module/Code View:

Purpose: Shows the system's software modules and their relationships.

- Components:
  - Core modules: User management, product management, order management, equipment management, agro solution management, etc.
  - Utility modules: Authentication, authorization, logging, etc.
  - Data access layer: Interfaces with the data store.
  - External interfaces: APIs for interacting with external systems.

PROJECT ARCHITECTURE VIEWS E-FARM

Development/Structural View:

- Purpose: · Depicts the system's development structure and dependencies.
- Components:
  - Development tools (IDEs, version control systems, testing frameworks)
  - Programming languages and libraries
  - Build scripts and deployment process

PROJECT ARCHITECTURE VIEWS E-FARM

Physical/Deployment/Install View:

- Purpose: · Shows the physical infrastructure and deployment configuration.
- Components:
  - Hardware: Servers, network devices, storage devices.
  - Software: Operating systems, application software, databases.
  - Deployment environment: Cloud, on-premises, hybrid.

PROJECT ARCHITECTURE VIEWS E-FARM

Data View/Data Model:

- Purpose: · Defines the structure and organization of data within the system.
- Components:
  - Entities: Users, products, orders, equipment, etc.
  - Relationships between entities.
  - Data types and attributes.
  - Data storage and access methods.

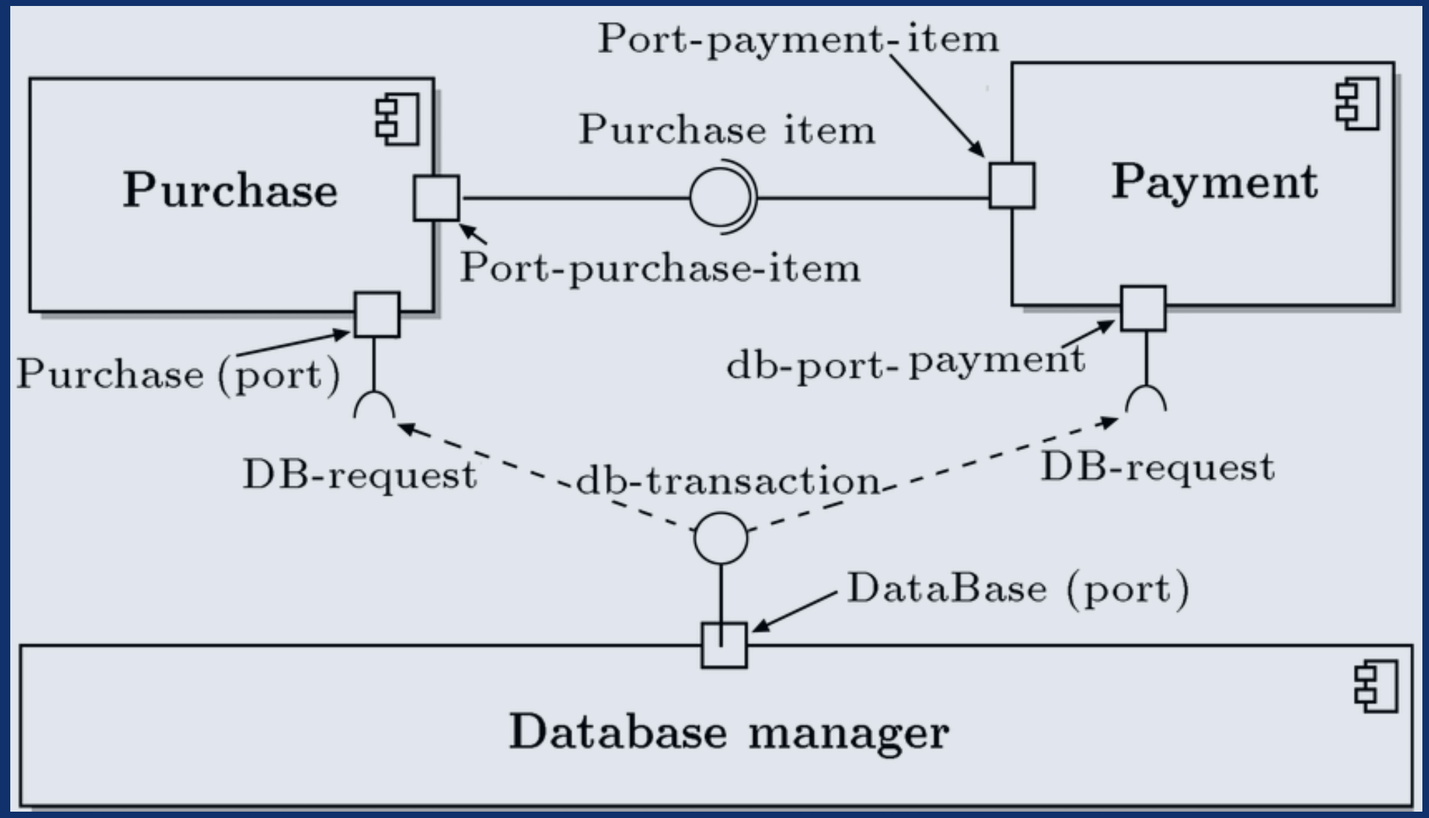
4+1 MODEL WITHIN OUR PROJECT

let's break down each architectural view **(Logical, Development, Process, Physical, +1 Use Case)** for E-Farm

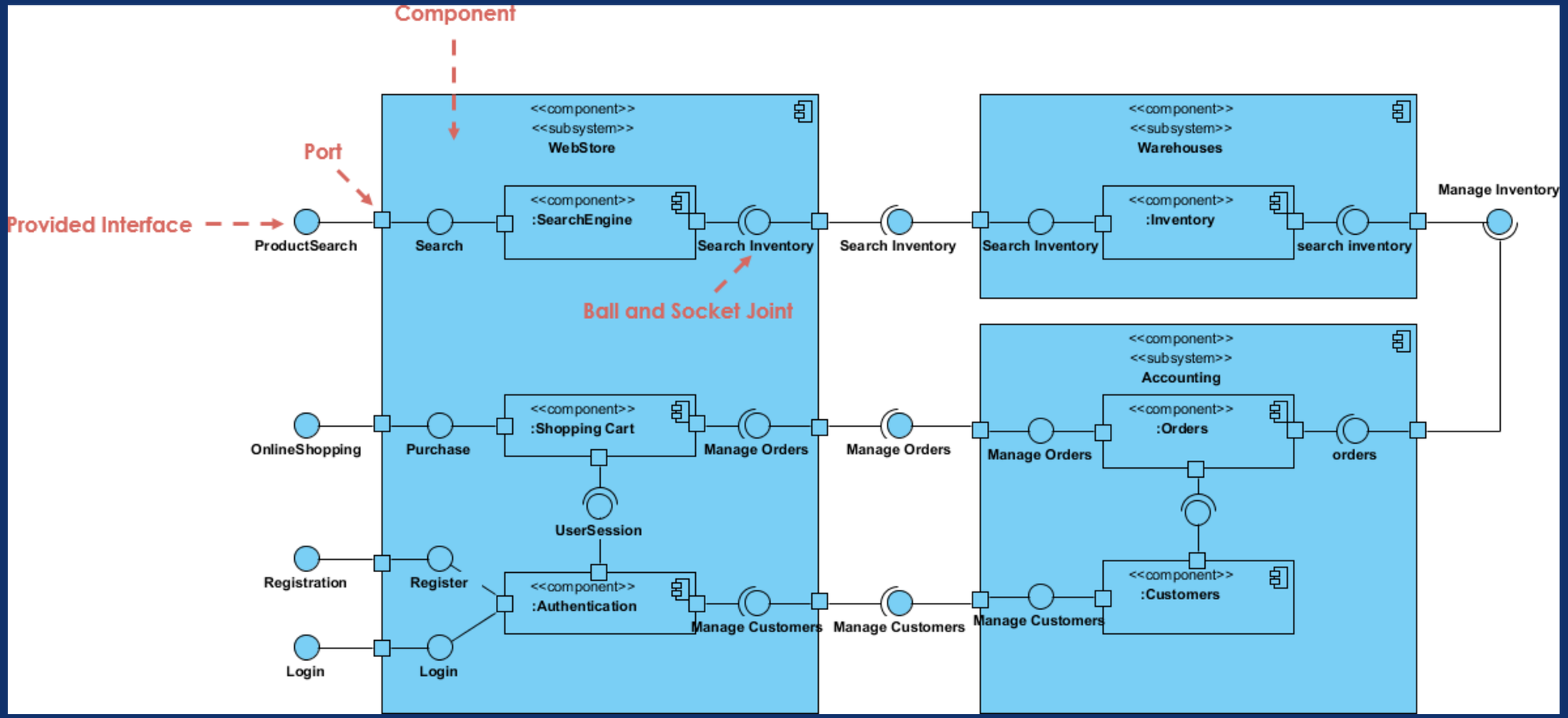
Logical View:

- **Focus**: Functionality without implementation details.
- **Viewpoint Type**: Functional Viewpoint.
- **Viewers**: Developers, Architects, Business Analysts.
- **Viewpoint Style**: **Class diagrams**, Package diagrams, **Component diagrams**.
- **Components**: User interfaces, databases, marketplace, equipment services, agro solutions.
- **Considerations**: Emphasize the functional aspects only, without delving into the technical implementation details.

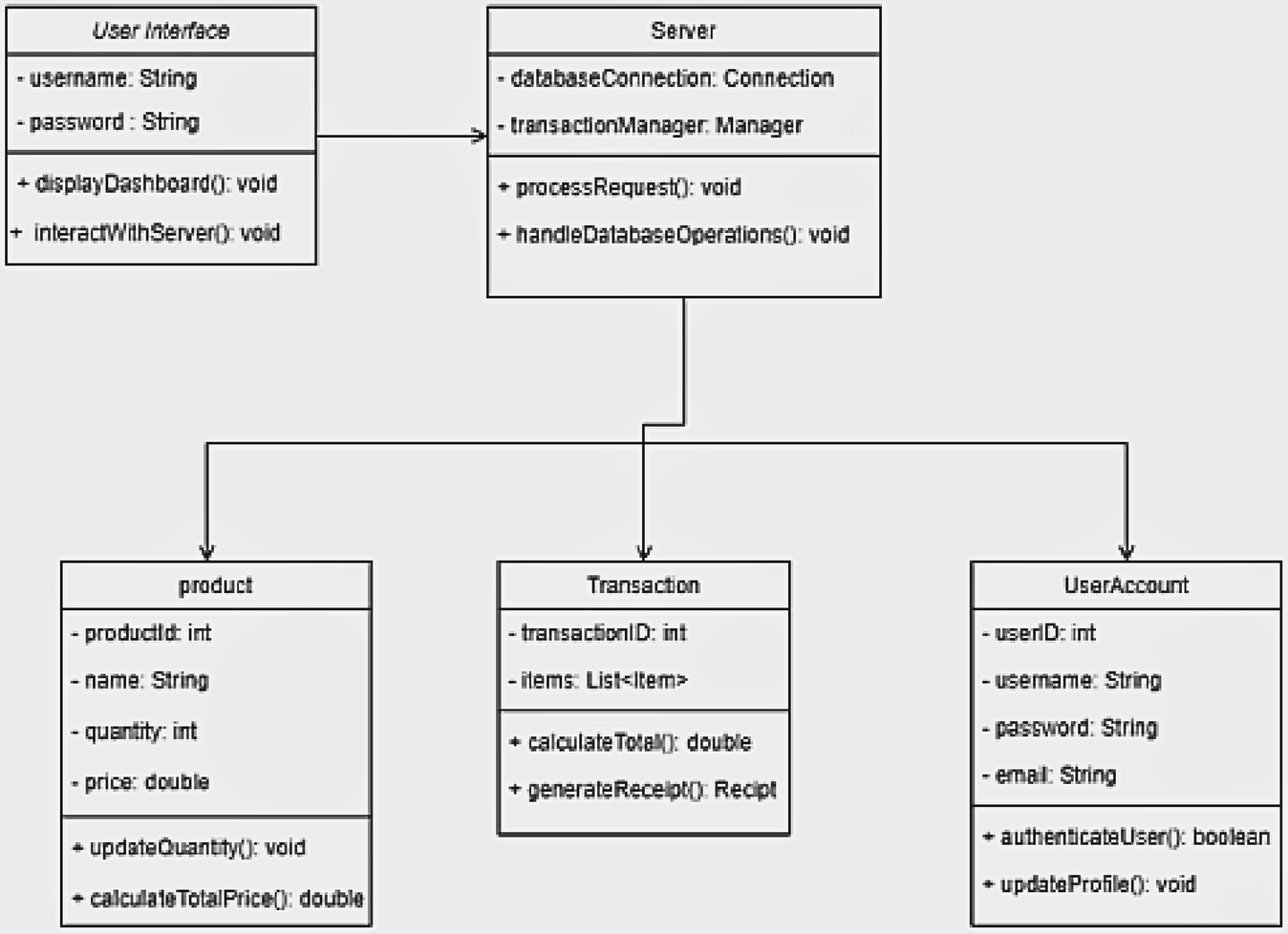
COMPONENT DIAGRAMS



COMPONENT DIAGRAMS-2



UML-CLASS DIAGRAMS

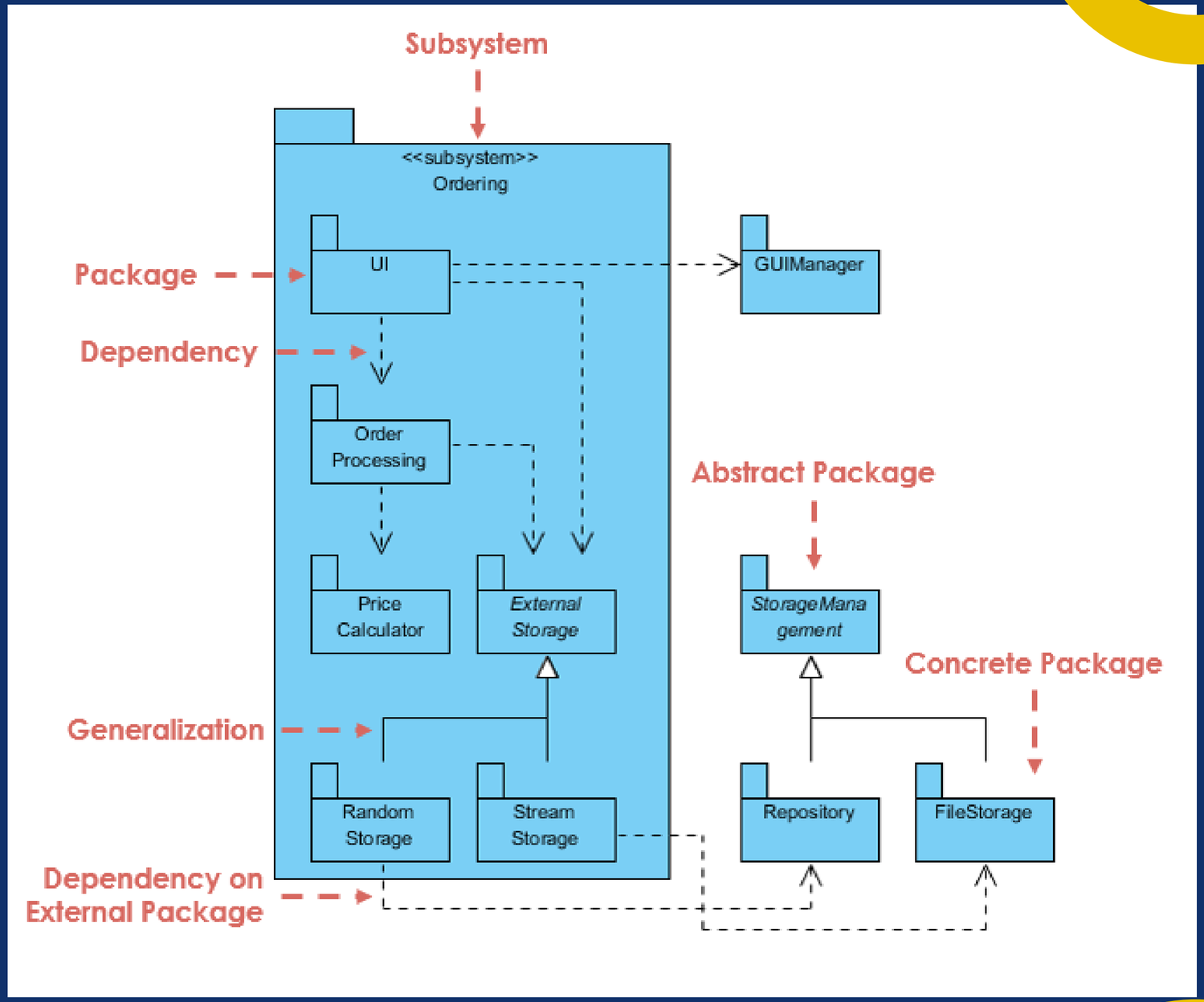


4+1 MODEL WITHIN OUR PROJECT

Development View:

- **Focus**: Software's development, modules, and organization.
- **Viewpoint Type**: Module Viewpoint.
- **Viewers**: Developers, DevOps, Project Managers.
- **Viewpoint Style**: Package diagrams, Dependency diagrams, Module breakdowns.
- **Components**: Modules/libraries for user authentication, data processing, interface components.
- **Considerations**: Highlight the software structure for development, showcasing modules and dependencies.

PACKAGE DIAGRAM



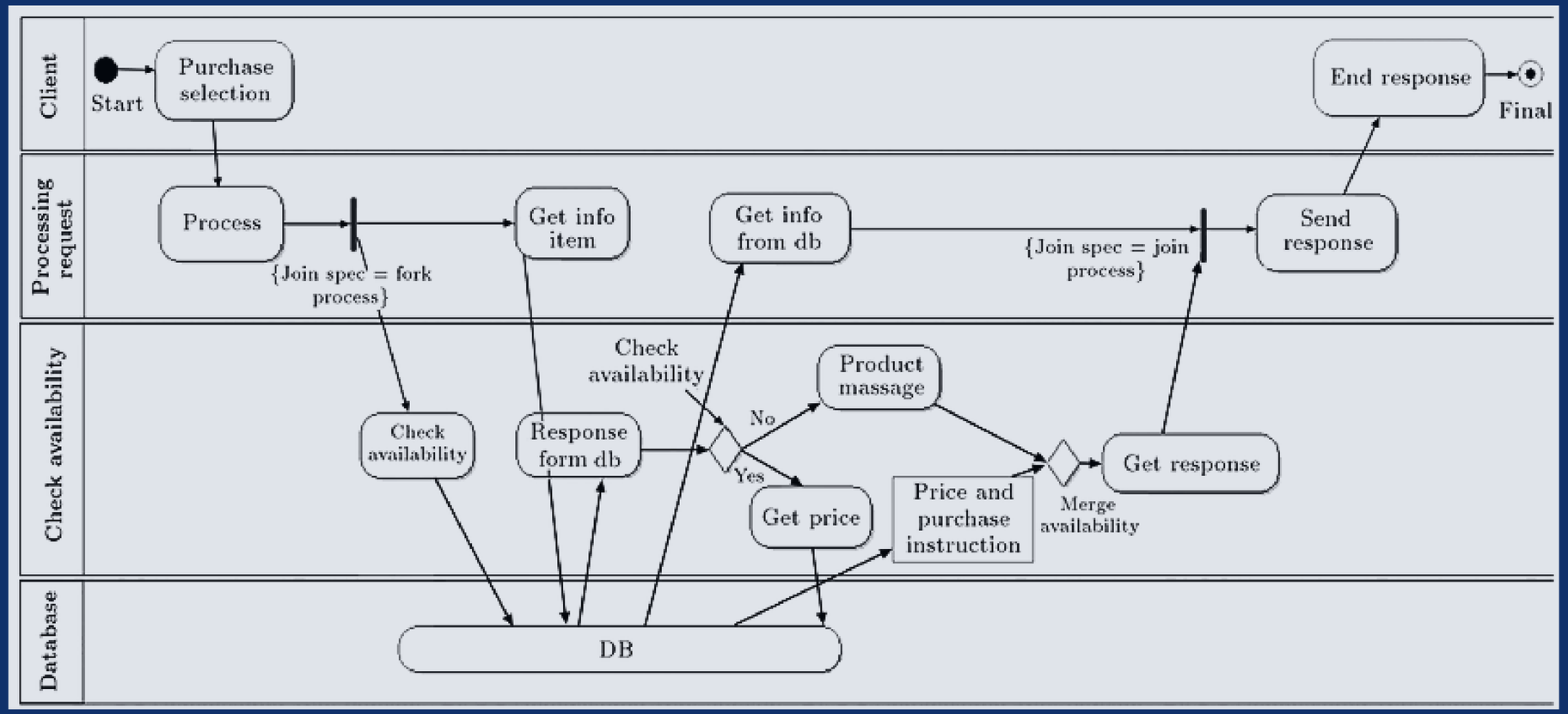
ORDER SUBSYSTEM

4+1 MODEL WITHIN OUR PROJECT

Process View:

- **Focus**: System's dynamic behavior and interactions.
- **Viewpoint Type**: Dynamic Viewpoint.
- **Viewers**: Developers, System Analysts, QA Engineers.
- **Viewpoint Style**: Sequence diagrams, **Activity diagrams**, State diagrams.
- **Components**: Interaction flows (e.g., order placement, equipment rental initiation, consultation service).
- **Considerations**: Illustrate system behaviors and interactions among components/processes.

ACTIVITY DIAGRAMS



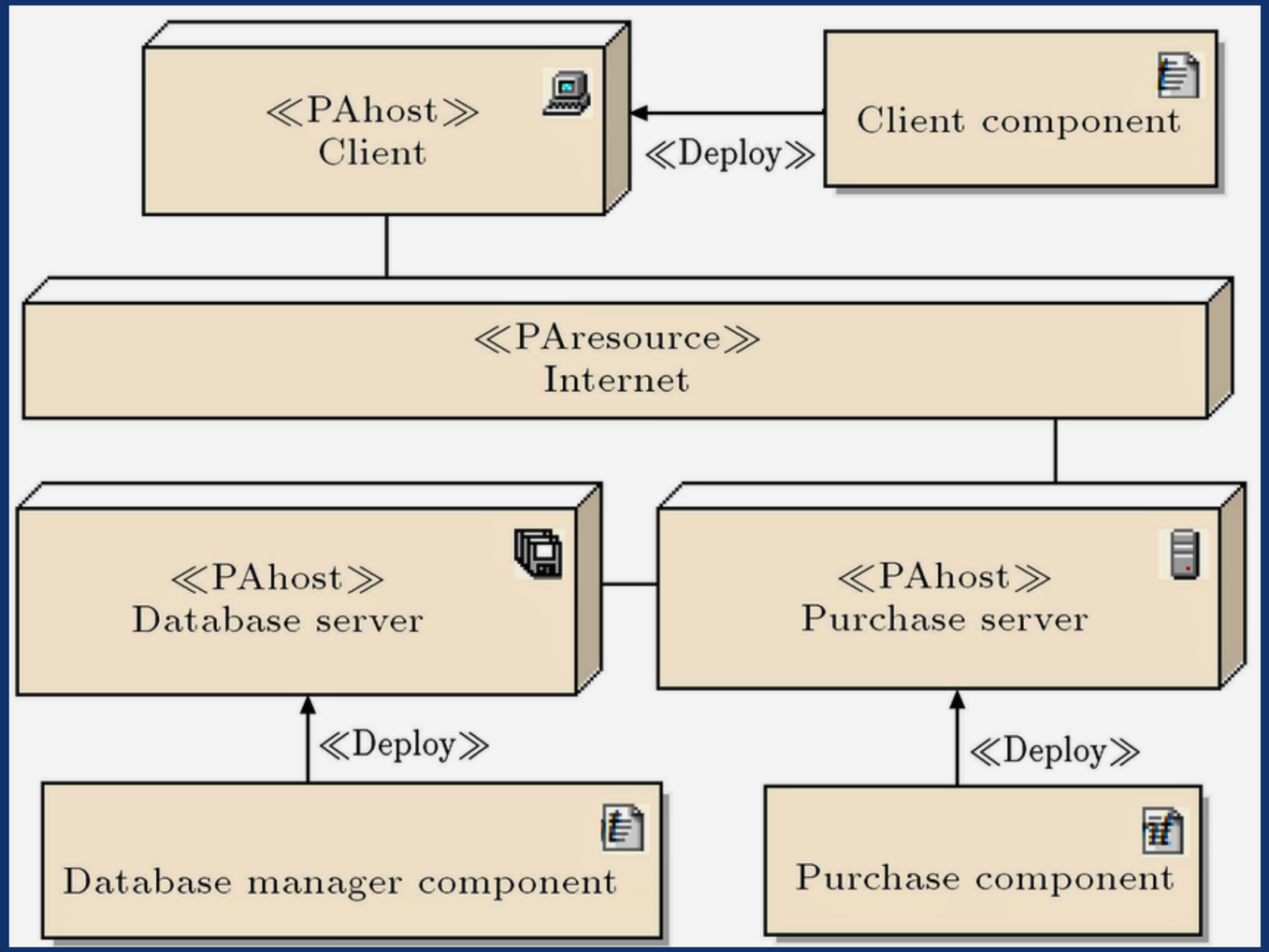
ORDER PLACEMENT

4+1 MODEL WITHIN OUR PROJECT

Physical View:

- **Focus**: Physical architecture and deployment.
- **Viewpoint Type**: Deployment Viewpoint.
- **Viewers**: System Administrators, Network Engineers, Cloud Architects.
- **Viewpoint Style**: **Deployment diagrams**, Infrastructure diagrams.
- **Components**: Servers, databases, network configurations, infrastructure layout.
- **Considerations**: Show physical deployment details, including server locations, networking setups, and distribution of components.

DEPLOYMENT DIAGRAMS



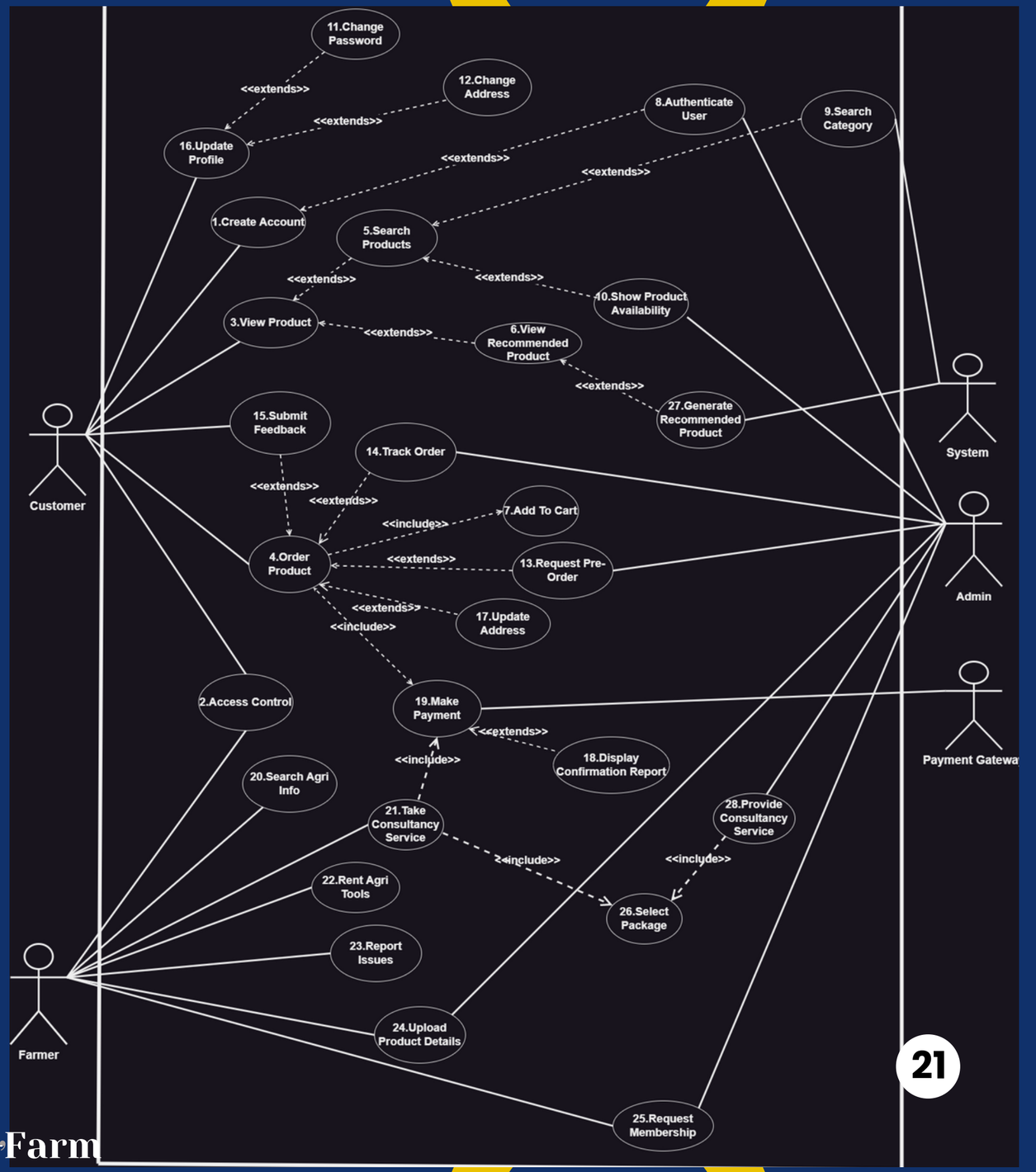
4+1 MODEL WITHIN OUR PROJECT

+1 Use Case or Scenarios:

- **Focus**: Specific scenarios to understand system functionality.
- **Viewpoint Type**: Scenario Viewpoint.
- **Viewers**: End-users, Business Stakeholders, UX Designers.
- **Viewpoint Style**: **Use case diagrams**, Sequence diagrams.
- **Components**: Farmer listing a product, consumer purchasing, equipment rental, consultation request.
- **Considerations**: Illustrate user interactions and system responses within specific scenarios.

USE-CASE DIAGRAM

E-Farm



RELIABLE PATTERN FOR E-FARM

Client-Server

- **User login and register**: The client and the server (authentication and user management).
- **Password Recovery**: The client (user) and the server (authentication) for password recovery.
- **Search Products**: The client interacts with the server to search and retrieve product data.
- **Track Order**: The client interacts with the server to track order status.
- **Feedback Submission**: Involves communication between the client and server.
- **Makes Payment**: communicate with client and server for payment processing.
- **Display Agri Information**: The client interacts with the server to display agricultural information.
- **User Logout**: Involves user interface (client) interacting with the server to log out.

RELIABLE PATTERN FOR E-FARM

Model-View-Controller (MVC)

- **Update Profile**: MVC separates the user interface, business logic, and data storage, making it suitable for handling profile updates.
- **Display Products Availability**: MVC can handle the separation of data, business logic, and user interface.

RELIABLE PATTERN FOR E-FARM

Pipe-Filter

- **Receive confirmation mail**: This pattern can be used to process and send confirmation emails as a series of filters.
- **Filter Products**: Filters can be applied on the server to process and filter product data based on user preferences.
- **Providing Ratings and Reviews**: Filters can process and manage ratings and reviews before displaying them.

RELIABLE PATTERN FOR E-FARM

Blackboard Pattern

- **Display Product Recommendation**: The Blackboard pattern can be used for collaborative decision-making, which is suitable for generating and displaying recommendations.
- **Pre-Order Request**: The Blackboard pattern can be used for collaborative decision-making, which is suitable for handling pre-order requests.

RELIABLE PATTERN FOR E-FARM

Layered Pattern

- **Order Multiple Address**: Layered pattern allows for a clear separation of concerns, suitable for managing multiple addresses in the order process.
- **Upload product details**: Layered pattern allows for clear separation of concerns, suitable for managing the upload and storage of product details.

THANK YOU!

E-FARM