# Case Study 1 :

# A customer can make a payment either by Card or by Wallet or by Cash or by Net banking.

# Q1. Draw a Use Case Diagram - 4 Marks



## System Boundary:

View payment

## Use cases:

"View payment"

"Pay by Card"

"Pay by Wallet"

"Pay by Cash"

"Pay by Net Banking"

## Include Relationships

Include (Shows mandatory with «Include» Label):

Connect "View payment" → "Pay by Card".

Connect "View payment" → "Pay by Wallet".

Connect "View payment" → "Pay by Cash".

Connect "View payment" → "Pay by Net Banking".

This shows that different payment methods include the base use case.

## Define 1-to-Many

1-to-Many:

One Customer → Many Payment Methods.

# Q2. Derive Boundary Classes, Controller classes, Entity Classes. - 4 Marks

## Boundary Classes

Boundary Class used to handle interactions between the system and external actors. All use cases will be boundary class.

Ex: PaymentScreen

## Controller Class

Controller Class - act as intermediaries between boundary and entity classes. The actor or the use relation where there is no third party involved are controller class.

Ex: PayementintiatedController, TransactionController, CardPaymentController

## Entity Class

Entity Class - represents the core data and business logic of the application. All actor will be the entity class.

. Ex: Customer, Payment, Transaction

# Q3. Place these classes on a three tier Architecture. - 4 Marks

## 1. User Layer (UI Layer - Boundary Classes)

PaymentScreen → Displays payment options and collects user input.

PaymentGatewayAPI → Connects to external payment services.

## 2. Business Logic Layer (Application Layer - Controller Classes)

PaymentController → Handles payment validation, processing, and selection of payment methods.

TransactionController → Manages transaction processing, verification, and updates status.

## 3. Data Layer (Persistence Layer - Entity Classes)

Payment → Stores payment details (amount, method, status).

Transaction → Logs transaction history (timestamp, transaction ID).

Customer → Stores user details (name, payment history, contact).

# Q4. Explain Domain Model for Customer making payment through Net Banking - 4 Marks

A domain model is a conceptual representation that define the structure relationship, and behavior of entity within a specific problem domain. It focuses on entities, their attributes, and relationships without considering implementation details.



# Q5. Draw a sequence diagram for payment done by Customer Net Banking - 4 Marks

## Sequence Diagram

It is a type of interaction between objects in a system over time, It used in System design to illustrate how a process operate with another and in what order. Sequence Diagram show how data is flowing in more sequential manner.



# Q6. Explain Conceptual Model for this Case - 4 Marks

## Conceptual Model

Conceptual model is high level representation of a system that helps in understanding visualizing and communicating the essentials aspects of a domain focusing on the key entities and their relationships without technical details. It helps understand the system from a business perspective.

### Important Entities in the Conceptual Model

Customer → Initiates the payment.

Order → Represents the purchase transaction.

Payment → Records the payment details.

Bank → Processes the net banking payment.

Net Banking Transaction → Logs details of the net banking process.

### Relationships Between Entities

A Customer can place multiple Orders (1:M).

Each Order has one associated Payment (1:1).

A Payment is processed using a Payment Method (Card/Wallet/Cash/Net Banking).

If the payment is via Net Banking, a Net Banking Transaction is recorded (1:1).

A Bank facilitates multiple Net Banking Transactions (1:M).

# Q7. What is MVC architecture? Explain MVC rules to derive classes from use case diagram and

# guidelines to place classes in 3-tier architecture - 8 Marks

## MVC Architecture

MVC (Model-View-Controller) is a software architectural pattern that separates an application into three main components:

Model → Represent the data and the business logic of the application.

View → Represents the UI (User Interface) presentation layer of application.

Controller → Handles user inputs and updates Model & View accordingly. Acts as an intermediary between model and view.

Model, View, and Controller—to promote modularity, maintainability, and scalability. This separation allows for independent development, testing, and modification of each component.

## From a Use Case Diagram, we can derive three types of classes:

Boundary Classes (View) → Represent interaction points with users (e.g., UI screens, forms).

Control Classes (Controller) → Manage the flow of data between the View and Model.

Entity Classes (Model) → Represent business objects (e.g., Customer, Order, Payment).

## Guidelines to Place Classes in a 3-Tier Architecture

|  |  |  |
| --- | --- | --- |
| **Tier** | **Purpose** | **Derived Classes** |
| **Presentation Layer (UI Layer)** | Handles user interactions and displays data | PaymentForm (Boundary Class) |
| **Business Logic Layer** | Contains business rules & processing logic | PaymentProcessor (Control Class) |
| **Data Access Layer** | Manages database operations and data storage | Payment, Customer, Bank (Entity Classes) |

# Q8. Explain BA contributions in project (Waterfall Model – all Stages) – 8 Marks

The Waterfall Model is a linear and sequential software development methodology where each phase depends on the completion of the previous one. A Business Analyst (BA) plays a crucial role in each stage by ensuring clear requirements, stakeholder alignment, and business value delivery. BA contributes to a project's success by minimizing risks and improving efficiency.

## Requirement Gathering & Analysis

BA Conducts stakeholder interviews, focus groups, and workshops to gather requirements. Documents Business Requirements Document (BRD) and Functional Specification Document (FSD) are being prepared. Defines Use Cases, Process Flow Diagrams, and Wireframes for better understanding. Ensures all business rules are captured correctly before proceeding to design.

## System Design

BA Assists in converting business requirements into technical specifications. Works with UI/UX designers to refine user interface mockups. They ensures the design meets business goals and user needs. Also participates in Data Modeling & Database Schema Design discussions.

## Implementation

BA works closely with developers to clarify business logic and resolve ambiguities. They ensures the scope is controlled to minimize requirement changes. Supports the project team in adapting to unforeseen challenges.

## Testing

BA Defines User Acceptance Testing (UAT) scenarios and test cases. BA reviews test results to ensure they align with the original requirements. BA works with QA teams to ensure business processes are tested. BA helps in identifying gaps, defects, and missing functionalities.

## Deployment

BA Assists in preparing training materials and conducting end-user training. Supports the Go-Live process by validating if all business needs are met. Monitors early feedback from users and provides post-implementation support.

## Maintenance

BA collects feedback for future improvements. Assists in analyzing change requests and their business impact. Works with teams to enhance system features and optimize performance.

# Q9. What is conflict management? Explain using Thomas – Kilmann technique – 6 Marks

Conflict management is the process of resolving conflicts or disagreements between individuals or groups in a constructive manner. Thomas Kilmann technique is a widely used tool for assessing conflict resolution styles & guiding individuals in selecting appropriate strategies to manage conflicts.

## 5 steps of conflict management -

- Identify the conflict.

- Discuss the details.

- Agree with the root problem.

- Check for every possible solution for the conflict.

- Negotiate the solution to avoid future conflicts.

Thomas-Kilmann Conflict Management Model is often represented as a matrix that visually maps the five conflict-handling styles based on two dimensions:

**Assertiveness** (Y-Axis) – The extent to which an individual tries to satisfy their own concerns.

**Cooperativeness** (X-Axis) – The extent to which an individual tries to satisfy the concerns of others.

# Q10. List down the reasons for project failure – 6 Marks

## • Poor Planning

 Without a well-defined project plan, teams may face missed deadlines, cost overruns, and misalignment with business goals.

## • Unclear Objectives and Requirements Inadequate Risk Management

 If project goals and requirements are not well-defined, the team may deliver the wrong product or face continuous changes.

## • Poor Communication

Miscommunication between stakeholders, teams, or departments can lead to misunderstandings, errors, and inefficiencies.

## • Scope Creep

Uncontrolled changes or additions to project scope lead to increased costs, delays, and resource exhaustion.

## • Lack of Stakeholder Engagement

 If stakeholders are not actively involved, the project may fail to meet their needs, causing dissatisfaction.

## • Resource Constraints

 Insufficient resources (budget, manpower, tools) can lead to delays, burnout, and compromised quality.

## • Technical Challenges

Unforeseen technical issues or reliance on outdated technology can cause system failures, security risks, or poor performance.

# Q11. List the Challenges faced in projects for BA – 6 Marks

## - Unclear or Changing Requirements

Unclear requirements can lead to misaligned deliverables, rework, and dissatisfaction. Frequent changes cause delays and budget overruns.

## - Managing Stakeholder Expectations.

Misalignment between stakeholders and the project team can result in conflicts, unrealistic demands, and low acceptance of deliverables.

## - Scope Creep and Scope Management

Uncontrolled expansion of project scope results in cost overruns, extended timelines, and exhausted resources.

## - Time and Resource Constraints

 Limited time or resources can lead to missed deadlines, team burnout, and compromised quality.

## - Quality Assurance and Testing

Poor testing can result in bugs, system failures, and customer dissatisfaction.

## - Documentation and Knowledge Management

Poor documentation leads to miscommunication, knowledge loss, and inefficiency in onboarding new team members.

## - Technology Constraints and Complexity

Choosing the wrong technology or working with outdated systems can lead to integration issues, security risks, and project failure.

# Q12. Write about Document Naming Standards

Document naming standards describe the structured way in which files and documents are named within an organization. These standards provide clarity, consistency, and ease of retrieval of documents, avoiding confusion and duplication.

**Clarity** – The name should be clear and meaningful to all users.

**Consistency** – Follow a uniform format across all documents.

**Version** **Control** – Indicate different versions to track changes.

**Date** **Format** – Use a standardized date format (e.g., YYYY-MM-DD).

**No** **Special** **Characters** – Avoid using # % & \* ? / \ to prevent compatibility issues.

**Avoid** **Long** **Names** – Use concise yet descriptive names.

|  |  |
| --- | --- |
| **Category** | **Good Example** |
| **Requirement Document** | FOODEX\_ReqDoc\_PaymentModule\_2025-02-10\_v1.0.docx |
| **Meeting Minutes** | FOODEX\_MoM\_StakeholderMeeting\_2025-02-10\_v1.1.docx |
| **Test Case** | FOODEX\_TestCase\_LoginModule\_2025-02-10\_v2.0.xlsx |
| **Wireframe** | FOODEX\_Wireframe\_HomePage\_2025-02-10.png |

# Q13. What are the Do’s and Don’ts of a Business analyst – 6 Marks

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Do's** | **Don’t** |
| **1** | Consult an SME for clarifications in requirements. | Never say NO to the client. |
| **2** | Go to the client with a plain mind with no assumptions. Listen carefully and completely until the client is done, and then you can ask queries. | There is no word as “By default”. |
| **3** | Try to extract maximum leads to the solution from the client himself. | Never imagine anything in terms of GUI. |
| **4** | Concentrate on the important requirements. | Don’t interrupt the client when he is giving you the problem. |
| **5** | Question the existence of existence./ Question everything. | Never try to give solutions to the client straight away with your previous experience and assumptions. |

# Q14. Write the difference between packages and sub-systems – 4 Marks

## - Packages:

Collection of components which are not reusable in nature. A package is primarily for organizing code. Packages help in categorizing and managing functionalities at a detailed level.

Ex: Application development companies work on Packages.

## - Sub systems:

Collection of components which are reusable in nature. A sub-system represents a functional unit within a larger system and may interact with other sub-systems. Sub-Systems represent large-scale components that require coordination between multiple stakeholders.

Ex: Product development companies work on Sub systems.

|  |  |
| --- | --- |
| **Package** | **Sub-System** |
| A **logical grouping** of related business processes or requirements. | A **self-contained functional unit** within a larger system that performs a specific business function. |
| Helps organize and categorize **requirements, use cases, or features** for better traceability. | Represents a **major functional area** of the system, handling specific business processes. |
| Used for structuring **requirements** within a project. | Covers a broader **business function** and may involve multiple **interacting components**. |
| Groups related **business rules, processes, or requirements** but does not enforce strict modularity. | Encapsulates **business logic, data flow, and interactions** with other sub-systems. |
| Packages may have **dependencies on other requirements or features**. | Sub-systems **interact through APIs, integrations, or interfaces** to function within the system. |
| Used in **BRDs (Business Requirement Documents), FSDs (Functional Specification Documents), and UML diagrams**. | Represented in **high-level architecture, process flows, system interaction diagrams**. |

# Q15. What is camel-casing and explain where it will be used- 6 Marks

When working with requirement documents, APIs, databases, and system specifications. Naming conventions like camelCase ensure consistency in software development and help in better collaboration with developers and stakeholders.

**Uses:**

|  |  |
| --- | --- |
| **Area** | **BA's Role** |
| **Requirement Documentation (BRD, FSD, SRS)** | Ensures **consistent naming conventions** in system specifications and documentation. |
| **Use Case & Data Flow Diagrams** | Helps in clearly defining **business processes and system actions**. |
| **Database & Data Mapping** | Works with DBAs to **align field names** in data dictionaries. |
| **API Specifications (REST, SOAP)** | Ensures **API endpoint names** are clear and follow standards. |
| **UI/UX Field Naming** | Collaborates with UX designers and developers to maintain **consistent field naming** in forms. |
| **Test Case Documentation** | Helps QA teams define **test scenarios and automation scripts** using structured names. |

# Q16. Illustrate Development server and what are the accesses does business analyst has?

A Development Server is an environment where software developers write, test, and debug code before it moves to testing or production. It allows teams to integrate and validate new features without affecting the live system. A Business Analyst (BA) has limited but important access in the development server to support requirement validation, issue tracking, and collaboration with developers.

software development lifecycle (SDLC) includes multiple environments:

Development Server (DEV) – Where developers write and test code.

Testing Server (QA/UAT) – Where testers validate functionality.

Staging Server (Pre-Production) – A replica of production for final validation.

Production Server (LIVE) – The final environment accessible to end-users.

BA Need Development Server Access to:

Requirement Validation – Ensuring business logic is implemented correctly.

Early Issue Detection – Identifying gaps before UAT (User Acceptance Testing).

Better Collaboration – Working closely with developers and testers for issue resolution.

Stakeholder Communication – Providing early feedback to business teams.

# Q17. What is Data Mapping 6 Marks

As a Business Analyst (BA), data mapping is essential when defining data flows, integrations, and migrations between different systems. A BA ensures that business rules and data formats are correctly aligned across various platforms.

Data mapping is the process of connecting data from one source to another.

It's like creating a guide or map that shows how data in one place corresponds to data in another place.

This is especially important when you're moving data between different systems or databases to ensure that the data stays consistent and accurate.

## BA's Responsibilities in Data Mapping

Identify Source & Target Data

Define Field Mapping

Understand Data Formats

Apply Business Rules

Document Data Mapping

Validate & Test Mappings

Data Mapping is used in Data Migration, System Integration, Data Warehousing, ETL Processes

# Q18. What is API. Explain how you would use API integration in the case of your application Date format is dd-mm-yyyy and it is accepting some data from Other Application from US whose Date Format is mm-dd-yyyy

An API (Application Programming Interface) is a set of rules that allows different software applications to communicate with each other. APIs enable data exchange between systems by defining request and response formats.

In context to Online Agriculture Products Store project, suppose an farmers purchase agricultural products through an online platform. considering application (Indian System) needs to accept order data from a US-based supplier’s system via an API. However, there is a date format mismatch:

## Scenario:

Our System's Date Format: dd-mm-yyyy (e.g., 25-12-2025)

US System's Date Format: mm-dd-yyyy (e.g., 12-25-2025)

## Steps:

1. US Supplier's API will sends Data as:

API Request: { "order\_id": 12345, "order\_date": "12-25-2025" }

2. Our System Receives Data & Processes Date Format:

Extract the order\_date field (12-25-2025).

3. Convert mm-dd-yyyy to dd-mm-yyyy before storing.

Use Backend Logic for Date Transformation:

4. Convert "12-25-2025" → "25-12-2025".

Store in the database in the correct format.