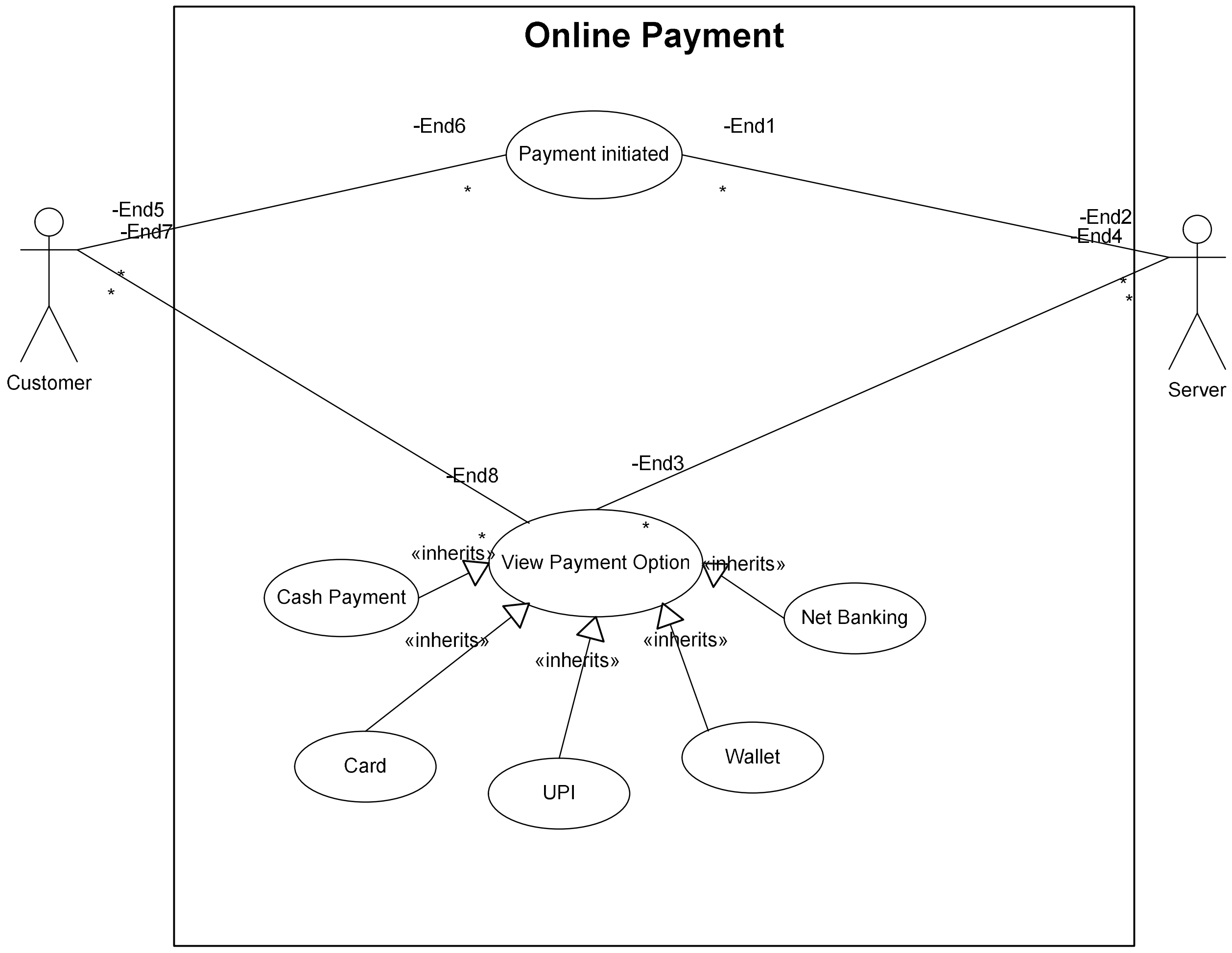
Q1. Draw a Use Case Diagram

Answer – 1



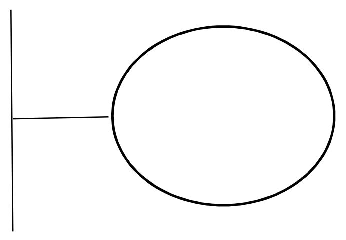
Q2. Derive Boundary Classes, Controller classes, Entity Class.

Answer 2-

UML Class Diagram Stereotypes

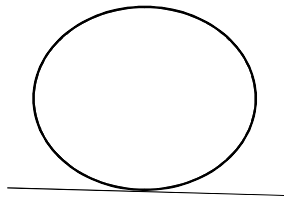
1. Boundary Classes:

- Definition: Represent the interaction between the system and external actors.
- Purpose: To define the boundaries of the system's interface with users or external systems.
- Features:
  - Often associated with UI elements or system interfaces.
  - Help in defining how external entities interact with the system.


Boundary Class

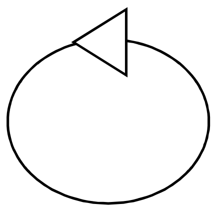
2. Controller Classe.

- Definition: Represent the logic that manages use cases in the system.
- Purpose: To handle the flow of control and interaction between boundary and entity classes.
- Features:
  - Typically have few attributes.
  - Focus on behavior and interaction logic.
  - Represented by a specific icon in UML diagrams.
  - Achieve the logic of use cases as defined in the Use Case Diagram.


Controller Class

3. Entity Classes :

- Definition: Represent the data or persistent objects in the system.
- Purpose: To encapsulate data and its associated behavior, often corresponding to database tables or records.
- Features:
  - Store and manage system data.
  - Frequently mapped to database entities (e.g., tables).
  - Extracted similarly to entities in ER (Entity-Relationship) diagrams.
  - Related to the DOA (Data-Oriented Approach).
  - High module cohesion ensures stability and minimizes changes.
  - Represented by a specific icon in UML diagrams.


Entity Class

Q3. Place these classes on a three tier Architecture.

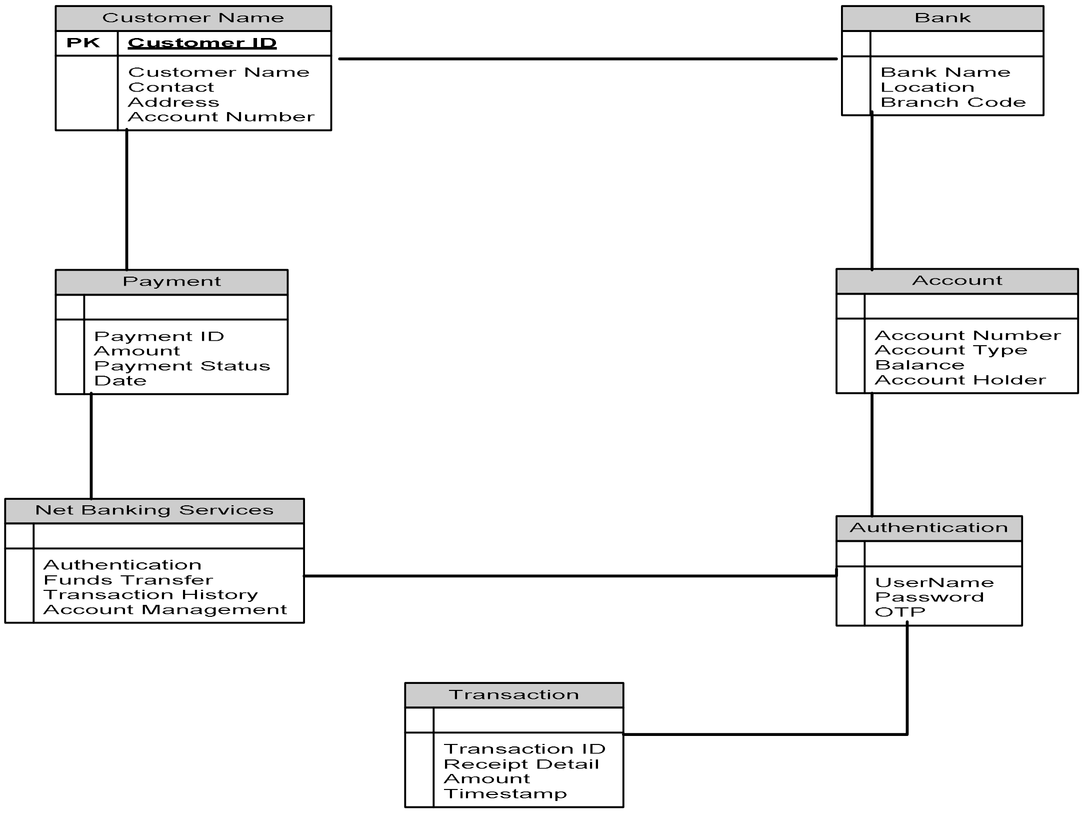
Answer-

User Layer
Payment Method Selection Boundary
Card Payment Boundary
Wallet Payment Boundary
Net Banking Boundary
Business Logic
Payment Controller
Card Payment Controller
Wallet Payment Controller
Net Banking Controller
Data Tier
Customer (Entity)
Card (Entity)
Wallet (Entity)
Net Banking (Entity

Q4. Explain Domain Model for Customer making payment through Net Banking.

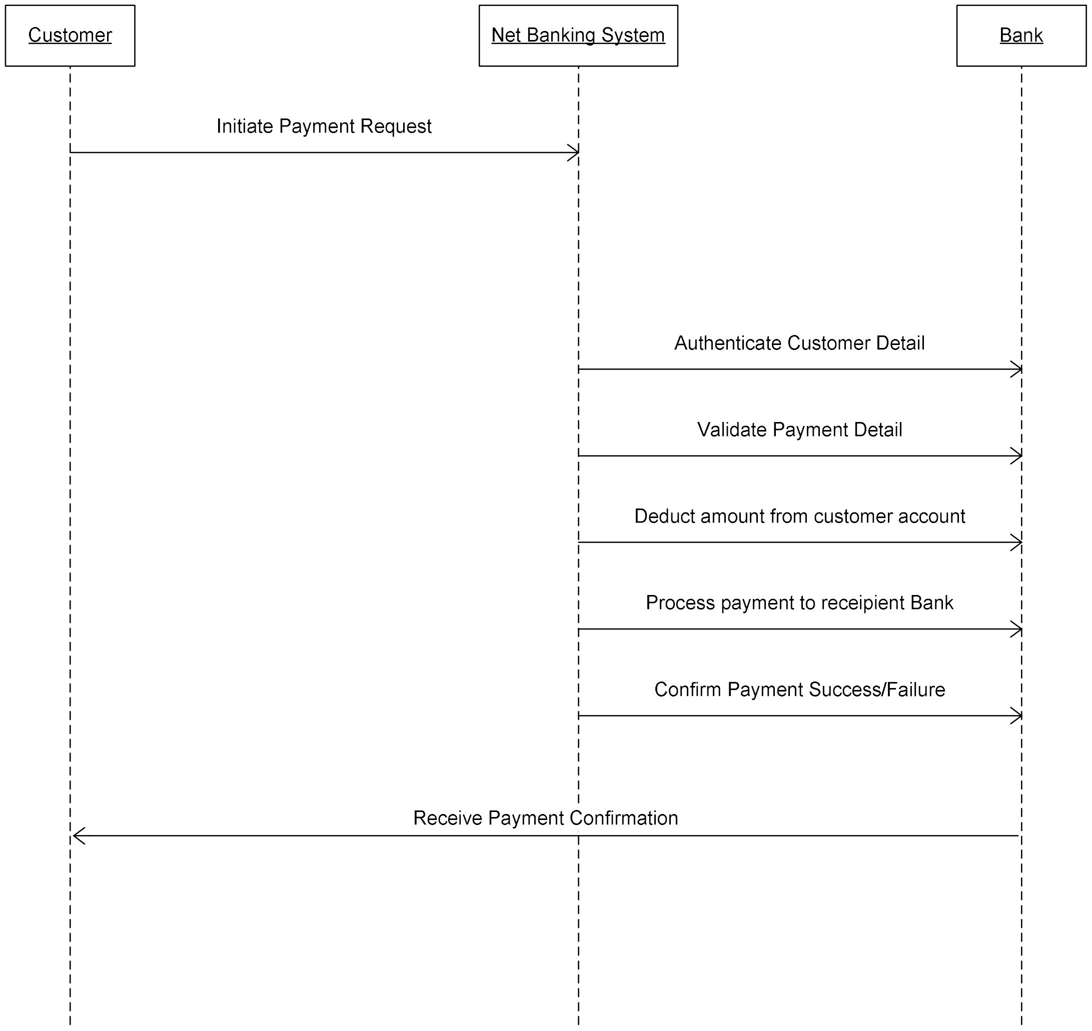
Answer-

A domain model represents the real-world entities, their attributes, and their relationships within a specific business context. For the scenario of a customer making a payment through net banking, the domain mdel identifies key entities and their interactions.



Q5. Draw a sequence diagram for payment done by Customer Net Banking .

Answer:-


Customer
Net Banking System
Bank
Initiate Payment Request
Authenticate Customer Detail
Validate Payment Detail
Deduct amount from customer account
Process payment to receipient Bank
Confirm Payment Success/Failure
Receive Payment Confirmation

Q6. Explain Conceptual Model for this Case.

Answer-

Conceptual modeling is a representation of the business model we have. There are entities, and their relationships among them based on which this is created. This is a Representation of a system which will be created, and this is not a language used to communicate with technical team.

•Entities In our case study, since customer is initiating online payment through net banking following entities will be there. Bank, Customer, Net banking system of a Bank, Final beneficiary (Customer's customer).

•Relationships Customer will be initiating payment with multiple beneficiaries, so it will be One to Many Relationships. Bank System will interact with Customer, so it will be one to one relationship. Bank will be facilitating payment for multiple customers; hence it will be One to May Relationship. Based on above

Entities and Relationships we will be creating a Entity Relationship diagram, which will be used as a representation to take sign off from the client.

Q7. What is MVC architecture? Explain MVC rules to derive classes from use case diagram and guidelines to place classes in 3-tier architecture.

Answer:

To identify Classes from use case Diagram, we apply MVC rules on each use case to derive Classes.

Model: The model class knows about all the data that need to be displayed. It is model who is aware about all the operations that can be applied to transform that class. It only represents the data of an application. The model represents enterprise data and the business rules that govern access to and updates of this data. This represents Database (Tables in DB). All Model Classes are represented as Entity Classes.

View: The view represents the presentation of the application. The view class refers to the model. It uses the query methods of the model to obtain the contents and renders it. The view is not dependent on the application logic. It remains same if there is any modification in the business logic. View Class is the data required by the query. View Class is represented as Boundary Class or Form Class.

Controller: Whenever the user sends a request for something then it always go through the controller. The controller is responsible for intercepting the requests from view and passes it to the model for the appropriate action. After the actior' taken on the data, the controller is responsible for directing the appropriate view to the user. In GUls, the views and the controllers often work very closely together.

MVC Architecture Rules

1. Combination of One Actor and an use case results in one Boundary class

2. Combination of Two Actors and an use case results in two Boundary classes

3. Combination of Three Actors and an use case results in Three Boundary classes and so on....

Note: only one primary actor is to be considered with a use case.

4. Use case will result in a controller class 5. Each Actor will result in one entity class.

Guidelines to place identified MVC Classes in a 3 Tier Architecture :

- Place all Entity Classes in DB Layer
- Place Primary Actor associated Boundary Class in Application Layer
- Place Controller Class in Application Layer
- If governing Body influence or Reusability is there with any of remaining Boundary Classes , place them in Business Logic Layer else place them in Application Layer

Q8. Explain BA contributions in project (Waterfall Model – all Stage)

Answer:

A BA is a facilitator for a project. The roles and responsibilities of business analyst are fundamental in satisfying stakeholder expectations and delivering a viable solution. Here are thecritical roles and responsibilities of a business analyst in project management.

Pre Project- This involves performing Enterprise Analysis using various methods like SWOT Analysis, GAP Analysis, Market Research, Feasibility Study, Root Cause Analysis. These methods help in preparing a Business Case, Project Scope and understanding the risks involved.

Planning, Estimations & Assessments- Planning an approach strategy for the project plan is extremely important for the success of the project. A BA conducts stakeholder analysis and understands the assumptions and constraints along with business rules and business goals.

Requirements Gathering- Requirements are an essential ingredient in any project because they form a foundation upon which projects are built. The requirements gathering process is pretty much a partnership between the business analyst, stakeholders, and the development team. This includes identifying stakeholders and using elicitation techniques to gather requirements. And prepare BRD and make prototype for client understanding, sort the requirements and prioritize and validate them.

Requirement Analysis- preparing UML diagrams and functional requirements from business requirements for better understanding of the requirements. Take signoff in SRS document which is legal binding document between technical team and business, prepare RTM. Track the progress during sdlc till testing phase .

Design- A BA communicates with clients on the design and solution documents to make them understand how the solution would look. A BA prepares prototypes and mock-ups along with the design teams for helping the client understand about the final product and also take feedback on the same for any changes and improvements.

Coding- BA organizes JAD sessions, clarifies queries of technical team during coding. A developer refers to the diagrams and documents prepared by a BA to code their unit. Regular status meetings are conducted to update the client on the progress.

Testing- A BA prepares test cases from the use cases or assist testing team to do so. A BA performs high level testing and prepares client for UAT, updates end user manuals, updates RTM and takes sign off on UAT.

Deployment and Implementation- BA forwards the RTM to the client or PM to be attached to the project closure document, coordinates to complete and share end user manuals, plans and organizes training sessions for end users. BA prepares lessons learned from this project to take precautions for upcoming project.

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

Answer:

Conflict Management

Conflict management is the process of identifying, addressing, and resolving disagreements or disputes in a constructive manner. It aims to minimize the negative impacts of conflicts and enhance collaboration, ensuring productive outcomes while maintaining healthy relationships among individuals or teams.

The Thomas-Kilmann Conflict Management Model is a widely used framework that identifies five strategies for managing conflict, based on two dimensions:

1. Assertiveness (the extent to which one tries to satisfy their own needs)
2. Cooperativeness (the extent to which one tries to satisfy others' needs)

The Five Strategies in Thomas-Kilmann Technique

1. Competing (High Assertiveness, Low Cooperativeness)
    - Description: This approach is power-oriented and involves pursuing one's goals at the expense of others.
    - When to Use:
        - Quick decisions are needed.
        - Issues are critical and non-negotiable.
    - Example: Enforcing a policy in a time-sensitive crisis despite opposition.
2. Collaborating (High Assertiveness, High Cooperativeness)
    - Description: Focuses on finding a win-win solution where both parties' concerns are fully addressed.
    - When to Use:
        - When the objective is to achieve the best outcome for all.
        - To resolve complex conflicts requiring mutual input.
    - Example: Two departments working together to integrate their processes for mutual benefit.
3. Compromising (Moderate Assertiveness, Moderate Cooperativeness)
    - Description: Involves finding a middle ground where both parties give up something to reach a mutually acceptable solution.
    - When to Use:
        - Time constraints exist.
        - The solution is temporary or of moderate importance.
    - Example: Splitting a budget equally between two competing projects.
4. Avoiding (Low Assertiveness, Low Cooperativeness)
    - Description: Involves sidestepping the conflict without resolving it.
    - When to Use:
        - The issue is trivial or not worth the time.
        - To allow emotions to cool down before addressing the conflict.
    - Example: Delaying a heated discussion until both parties calm down.
5. Accommodating (Low Assertiveness, High Cooperativeness)
    - Description: Involves prioritizing the other party's needs over one's own.
    - When to Use:
        - Maintaining harmony is more important than the outcome.
        - The issue is more critical to the other party.
    - Example: Agreeing to a colleague's proposal to maintain a good working relationship.

Q10. List down the reasons for project failure.

Answer

Project failure can occur due to various factors, often stemming from poor planning, communication, or execution. Below are some common reasons for project failure:

1. Lack of Clear Objectives and Goals

- Projects without well-defined objectives or goals can lack direction, leading to confusion among stakeholders and teams.
- Example: A project proceeds without a clear understanding of deliverables, resulting in unmet expectations.

2. Poor Planning and Estimation

- Insufficient planning or unrealistic timelines and budget estimations can derail a project.
- Example: Miscalculating resource requirements, leading to delays and budget overruns.

3. Inadequate Stakeholder Engagement

- Failure to involve key stakeholders or understand their expectations can lead to misaligned project goals.
- Example: Ignoring input from end users results in a product that doesn't meet their needs.

4. Communication Breakdown

- Ineffective communication among team members, stakeholders, or between management and the team can result in misunderstandings and delays.
- Example: Critical updates are not shared, causing teams to work on outdated information.

5. Scope Creep

- Uncontrolled changes or additions to the project scope can overwhelm resources and delay timelines.
- Example: Additional features requested late in the project without revisiting the impact on time and budget.

6. Lack of Risk Management

- Failure to identify, assess, and mitigate risks can lead to unforeseen obstacles.
- Example: Ignoring potential supply chain issues causes critical delays.

7. Inadequate Resources

- Insufficient allocation of human, financial, or technological resources can prevent project completion.
- Example: Assigning inexperienced team members to handle complex tasks.

8. Leadership Failures

- Weak leadership, lack of vision, or poor decision-making can demoralize teams and derail projects.
- Example: A project manager who fails to address conflicts, leading to team dysfunction.

Q11. List the Challenges faced in projects for BA

Answer

Challenges Faced by a Business Analyst (BA) in Projects

A Business Analyst (BA) plays a pivotal role in ensuring project success, but they often encounter several challenges during the project lifecycle. Below is a list of common challenges faced by BAs:

1. Ambiguous or Unclear Requirements

- Stakeholders may provide vague or conflicting requirements, making it challenging to define the scope clearly.

- Example: Different departments have varying expectations from the project, leading to confusion.

2. Stakeholder Management

- Difficulty in identifying and managing all relevant stakeholders or addressing their conflicting interests.
- Example: Stakeholders with differing priorities may resist proposed solutions or changes.

3. Communication Barriers

- Miscommunication or lack of effective communication between the BA, stakeholders, and technical teams.
- Example: Technical jargon misunderstood by business users, or vice versa, causing delays or errors.

4. Time Constraints

- Limited time for requirements gathering, analysis, or documentation can compromise the quality of deliverables.
- Example: Tight deadlines force the BA to rush through requirement analysis, leading to errors.

5. Resistance to Change

- Teams or stakeholders may resist adopting new processes, tools, or systems.
- Example: Employees are reluctant to shift to a new software system, fearing disruption.

6. Complexity of Requirements

- Highly complex or technical requirements may be difficult to document and communicate effectively.
- Example: Translating intricate technical specifications into business-friendly terms.

7. Insufficient Tools or Resources

- Lack of adequate tools, training, or resources to perform analysis and documentation effectively.
- Example: No access to proper requirements management software, forcing the BA to rely on manual processes.

8. Coordination with Cross-Functional Teams

- Collaborating with multiple teams across different domains, often with conflicting schedules and priorities.
- Example: Technical teams and business users fail to align their objectives during the SDLC.

Q12. Write about Document Naming Standards.

Answer:

Document Naming Standards

Document naming standards are predefined conventions used to create consistent, clear, and structured names for documents. These standards help in organizing, identifying, and retrieving documents efficiently, reducing confusion and errors. They are essential for maintaining uniformity across teams and projects.

Benefits of Naming Standards

- Simplifies document search and retrieval.
- Enhances collaboration by reducing ambiguity.
- Supports version control and traceability.

Document Naming Standards All documents will be named using some standards like [ProjectID][Document Type]V[X]D[Y].ext

Q13. What are the Do's and Don'ts of a Business analyst

Answer:

1. Never say NO to Client
2. There is NO word called as "BY DEFAULT"
3. Never imagine anything in terms of GUI ex: what client gives is not always correct
4. Question the existence of existence / question everything in the world Consult an SME for Clarifications in Requirements.
5. Every Problem of Client is unique. No two problems of different Client are same. May be the approach, technology, place of use, local laws may be varied to make them (Problems) to be different.
6. Go to Client with a plain mind with no assumptions. Listen carefully and completely until Client is done and then you can ask your Queries. Please do not interrupt the Client, when he/ She is giving you the problem.
7. Maximum Try to extract the leads to Solution from the Client itself. Never try to give Solutions to Client straight away with your previous experience and assumptions.
8. Try to concentrate on the important and truly required Requirements. Don't be washed away by add on Functionalities or don't imagine solutions on Screen basis.

Q14. Write the difference between packages and sub-systems

Answer:

Aspect	**Packages**	**Sub-Systems**
Definition	A logical grouping of related classes, interfaces, or components to organize code or functionality.	A self-contained module or system within a larger system, performing specific functions.
Purpose	Primarily used for organizing and modularizing code within software development.	Used to divide a large system into manageable, independent units.
Scope	Operates at the programming level to organize classes and interfaces.	Operates at the system design level to organize system components.
Example in Software	Java java.util package groups utility classes like ArrayList, HashMap, etc.	In an e-commerce application, a Payment Processing Sub-system handles payments independently.
Granularity	Finer-grained and focuses on grouping specific classes or code components.	Coarser-grained, representing a significant portion of a system with distinct functionality.
Dependency	Packages can depend on other packages but are typically smaller and less complex.	Sub-systems may depend on or interact with other sub-systems but are more complex.

Implementation	Defined explicitly in programming languages (e.g., package keyword in Java).	Defined during system architecture and design phases, often as part of the overall system structure.
Communication	Minimal, typically limited to importing classes from other packages.	Sub-systems communicate through well-defined interfaces or APIs.

Q15. What is camel-casing and explain where it will be use.

Answer:

Camel casing:-

Camel Case is a way to separate the words in a phrase by making the first letter of each word capitalized and not using spaces. It is commonly used in web URLs, programming and computer naming conventions. It is named after camels because the capital letters resemble the humps on a camel's back. Entire first word will be in lowercase and subsequent word's first letter should be in Upper Case. There will be no gap in between words.

Example: goThere(), completeTheProject(), getEmpId(), turnLeftAndSlowDown()Entire first word will be in lowercase and subsequent words first letter should be in Upper Case. There will be no gap in between words

Benefits of Camel-Casing

1. Readability: Improves the clarity of code by making compound words more distinguishable.
2. Consistency: Promotes uniformity in naming conventions, making code easier to understand and maintain.
3. Standardization: Aligns with best practices and conventions in many programming languages.

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

Development Server in Software Development

A development server is an environment where the initial development of a software application takes place. It is a safe and isolated space for developers to write, test, and debug code before it is moved to staging or production environments. It ensures that any issues are identified and resolved during the early stages of the Software Development Life Cycle (SDLC).

Key Features of a Development Server

1. Code Testing and Debugging
    - Allows developers to test new features or updates without impacting live systems.
2. Version Control Integration
    - Typically integrates with version control systems like Git for collaborative coding and version management.
3. Sandboxed Environment
    - Fully isolated from production systems, ensuring no unintended interference with live applications.
4. Simulated Data
    - Uses mock or test data to validate application functionality without risking sensitive real-world data.

Q17. What is Data Mapping.

Answer:

Data Mapping

Data Mapping is the process of creating a relationship between two distinct data models. It involves defining how data from one system, database, or format will be transformed and loaded into another system or format. The goal is to ensure that the data is accurately and consistently transferred, transformed, and understood between different systems.

Purpose of Data Mapping

1. Data Integration:
    - Facilitates seamless integration between different applications, systems, or databases by mapping corresponding fields or structures.
2. Data Transformation:
    - Ensures that data is transformed into the required format or structure to meet the needs of the target system or process.
3. Data Quality:
    - Helps in cleaning, validating, and ensuring the accuracy of the data during the migration or integration process.
4. Data Migration:
    - Used when migrating data from one platform to another, ensuring the integrity of data during the move.
5. ETL Process (Extract, Transform, Load):
    - Involves mapping data during the ETL process to extract data from source systems, transform it as needed, and load it into the target system.

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 10 Marks.

Answer:

An API (Application Programming Interface) is a set of rules and protocols that allows different software applications to communicate and interact with each other.

Establish API Communication: Set up API communication between your application and the other application to exchange data

Data Formatting: When sending date data from your application to the other application, convert the date from the dd-mm-yyyy format to the mm-dd-yyyy format. This can be achieved by extracting the day, month, and year components from the date and rearranging them according to the target format.

Data Parsing: When receiving date data from the other application, parse the mm-dd-yyyy formatted date into your application's dd-mm- yyyy format. Again, you will need to extract the day, month, and year components and rearrange them accordingly.

Data Validation: Perform data validation and ensure that the converted date remains valid after the format conversion. Check for edge cases, such as invalid dates or date ranges that might be affected by the format conversion, and handle them appropriately.