**Q.1**



**2. Boundary Classes:** Login Interface, Payment Method Selection Interface, Card Payment Details Interface, OTP Entry Interface, Payment Status Display, Logout Interface, Database Interface, Bank Interface. **Controller Classes:** Authentication Controller, Payment Initiation Controller, Card Payment Processing Controller, OTP Validation Controller, Payment Confirmation Controller, Session Management Controller.  
**Entity Classes:** User, Card, Wallet, BankAccount, Transaction.

**Q3. Place these classes on a three tier Architecture  
1. Presentation Tier (User Interface):**This tier is responsible for displaying information to the user and capturing user input. The Boundary Classes primarily reside in this tier.

Login Interface: Presents the login screen with "Email ID" and "Password" fields.  
Payment Method Selection Interface: Displays the options for "Card," "Wallet," "Net Banking," and "Cash."  
Card Payment Details Interface: Shows the form for entering card details.  
OTP Entry Interface: Presents the field for entering the OTP.  
Payment Status Display: Shows whether the payment was successful, failed, or is pending.  
Logout Interface: Provides the option for the user to log out.

2. Business Logic Tier (Application Tier):This tier contains the core logic of the application, processing user requests and interacting with the data tier. The Controller Classes belong to this tier.  
Authentication Controller: Handles the login process, verifying user credentials.  
Payment Initiation Controller: Receives the payment request and orchestrates the subsequent steps.  
Card Payment Processing Controller: Manages the specific logic for card payments (e.g., validating card details, interacting with the bank interface).  
OTP Validation Controller: Handles the verification of the One-Time Password received from the user.  
Payment Confirmation Controller: Finalizes the payment transaction and updates the system's records.  
Session Management Controller: Manages user sessions and their lifecycle.  
Database Interface: While it acts as a bridge, the logic for interacting with the database (queries, data manipulation) often resides within the business logic tier or is invoked by it.  
Bank Interface: The logic for communicating with the external banking system for authorization and processing resides here.

**3. Data Tier (Data Access Tier):**

This tier is responsible for storing and retrieving persistent data. The Entity Classes are representations of the data managed by this tier.

User: Represents the user data stored in the database.  
Card: Represents stored card details (potentially tokenized or encrypted for security).  
Wallet: Represents user wallet information.  
Bank Account: Represents stored bank account details (again, securely managed).  
Transaction: Represents the history and details of payment transactions.

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

A **Domain Model** is a conceptual representation of entities and relationships in a system that helps to understand the business context. In this case, the domain is **Customer Payment via Net Banking**.

**Difference between ER diagram and domain model-**

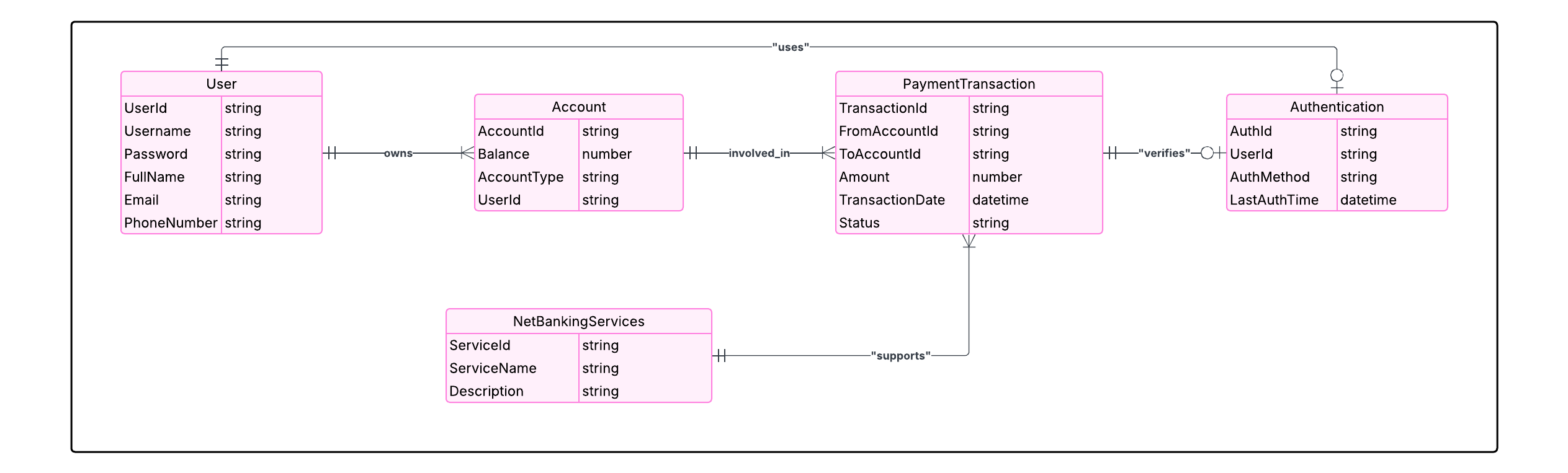
ER Model – do not have attributes inside the box

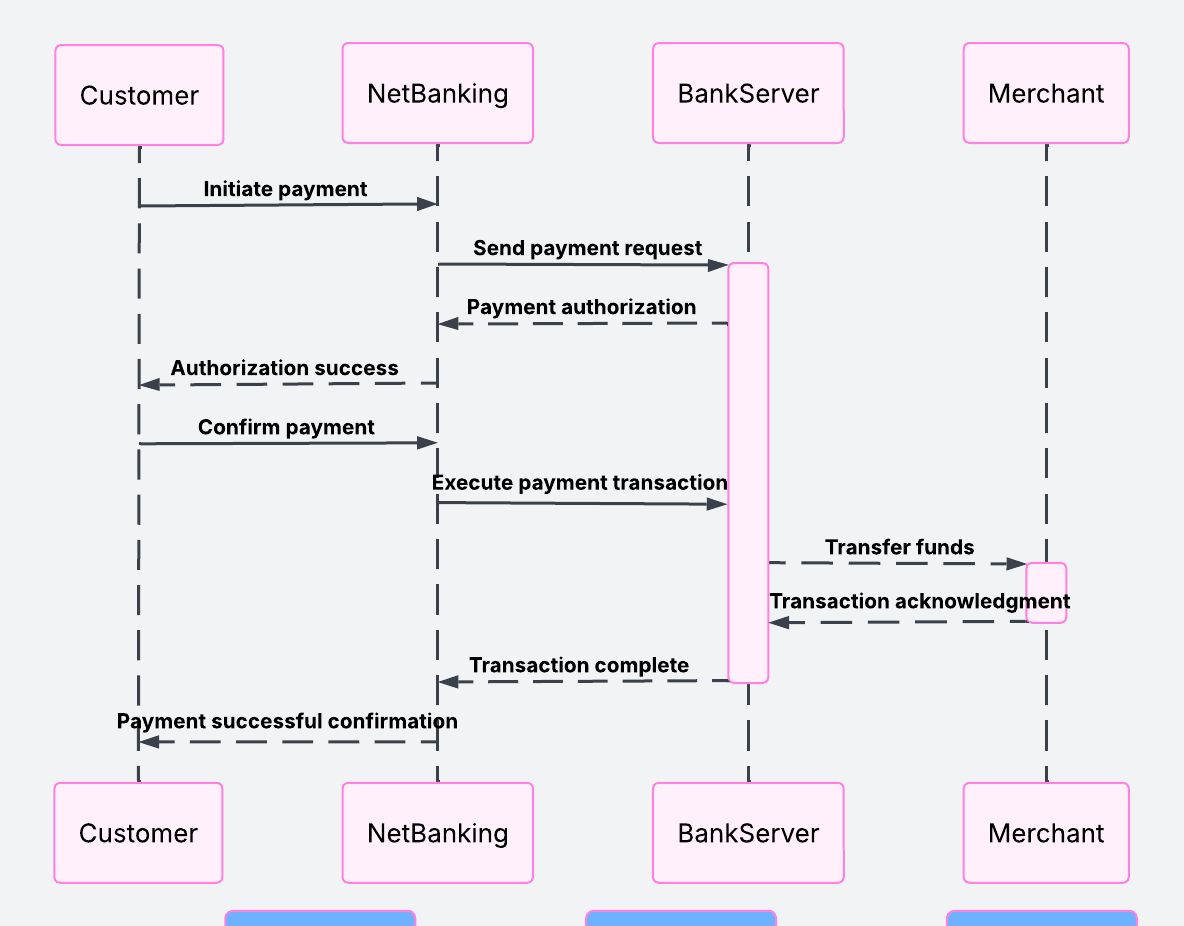
Domain Model- do have attributes mentioned inside the box.

ER Model – it is a data modelling technique used in database design to represent tables.

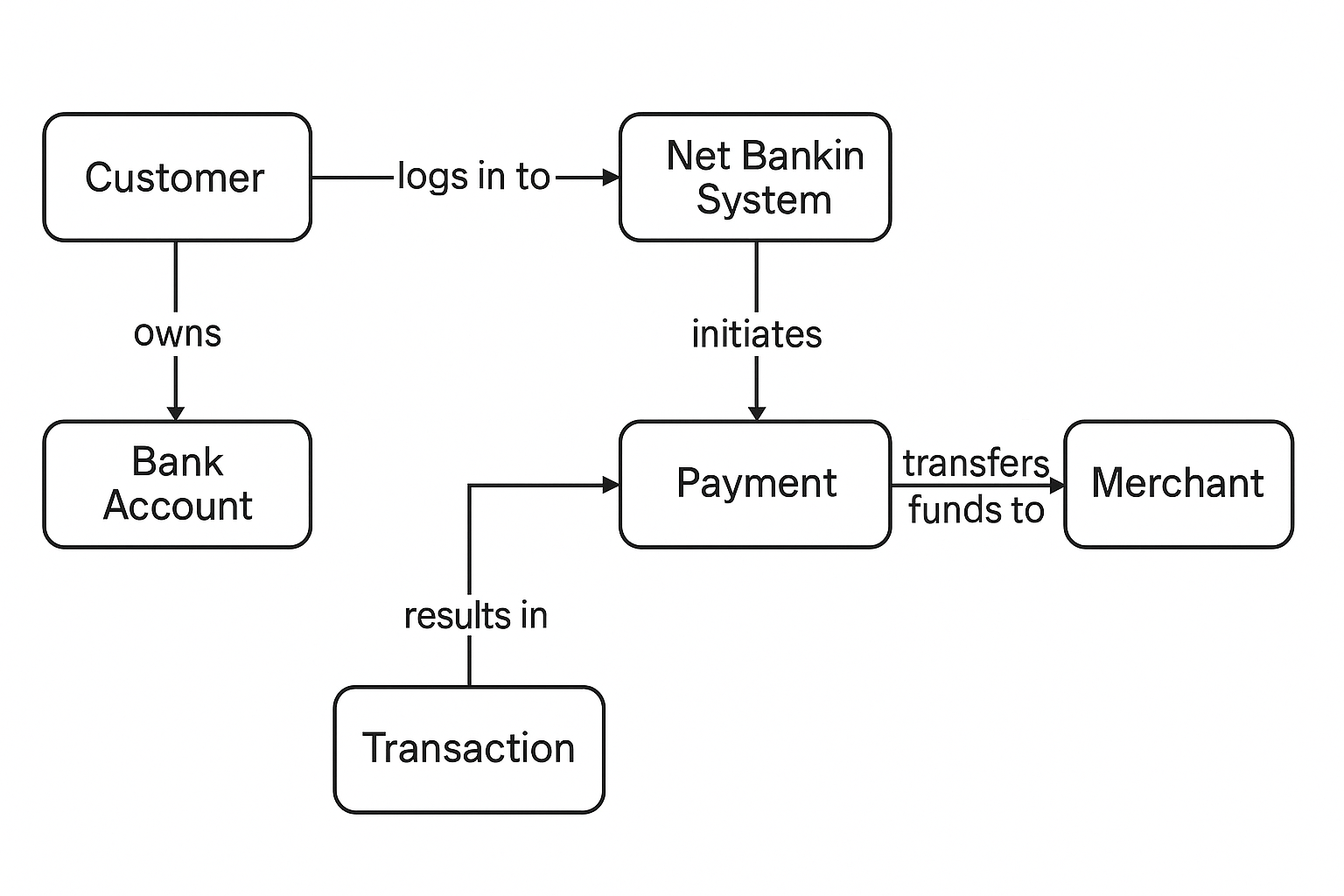
Domain Model- it is a conceptual model that represents real world entities.ER Model – focuses on relationships required for storing and retrieving the data  
Domain Model- It focuses on capturing the behaviour of application  
ER Model –primarily used in database design

Domain Model-used throughout the software development lifecycle

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



**Q6. Explain Conceptual Model for this Case**



### **What is MVC architecture?**

**MVC** stands for **Model-View-Controller**. It is a software design pattern used for developing user interfaces that separates an application into three interconnected components:

### **Components of MVC:**

1. **Model.**Interact with the Data base and Executive business logic in the form of Raw Data and given back to Controller
2. **Controller**Takes input from the client as a request parameter and transferred to the Model and takes back raw data from the model and process to the view
3. **View**It convert the raw data into readable format.  
   what user sees on the screen  
   Generate User interface for user
4. **MVC Rules to Derive Classes from Use Case Diagram:  
   Basic Rules   
   Identity Actor and Use case** – This helps to understand who interact with the system   
   Extract Nouns – Potential use case  
   **Potential Classes**: -“Card”, “Wallet”, “Payment”, “Customer” are all nouns → These can be Model classes.  
   **Extract Verbs → Operations or Methods:-** “Make Payment”, “Select Method”, etc. → Become methods in Controller or Model.

From Your Case Study:  
From the use case**:**

**Actors:**

* Customer

**Use Cases:**

* Make Payment by Card
* Make Payment by Wallet
* Make Payment by Cash
* Make Payment by Net Banking

**Derived Classes Using MVC**

**Model (Business Logic & Data)**

* Payment (Base Class)
* CardPayment (Subclass)
* WalletPayment (Subclass)
* CashPayment (Subclass)
* NetBankingPayment (Subclass)
* Customer

**View (UI Elements)**

* PaymentPageView
* PaymentSuccessView
* PaymentFailureView

**Controller (Handles Input & Logic)**

* PaymentController

**Guidelines for Placing Classes in 3-Tier Architecture**

| **Layer** | **Description** | **Includes** |
| --- | --- | --- |
| **Presentation Tier** | UI shown to the user | Views (e.g., HTML, UI Screens) |
| **Business Logic Tier** | Core application logic | Controllers, Services |
| **Data Access Tier** | Handles data storage/retrieval | Models interacting with DB |

**How to Place Our Classes:**

| **Class** | **Tier** |
| --- | --- |
| Customer | Data Access Tier |
| Payment | Data Access Tier |
| CardPayment | Data Access Tier |
| WalletPayment | Data Access Tier |
| PaymentController | Business Logic Tier |
| PaymentPageView | Presentation Tier |
| PaymentSuccessView | Presentation Tier |
| PaymentFailureView | Presentation Tier |

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

**Requirements Gathering: -** BA will gather requirements from Client by using elicitation techniques results BRD Doc.

**Requirements Analysis: -** Analyze and Document the business needs in the Form of BRD and SRS Documents

* BA will prepare the FRS and gather NFS from the technical team results SSD Doc
* BA will combine both FRS and SSD and SRS
* Doc Sign off From the client i.e SRS
* BA Will prepare RTM according to SRS

**Design:-**

Collaborate with solution architects and designers to ensure requirements are accurately interpreted.

**HDD,ADD** and solution document will be generated here done by the tech team.

**Highlevel Design Doc.: -**BA Collaborate with designers, architects, and developers to translate requirements into system design.  
BA Ensure that the design aligns with the documented requirements and addresses stakeholder needs.

**Development**

The Development phase include implementation. It involves coding the software based on the design specifications.  
Programmers or developer are involved in this phase.  
Here BA acts as a mediator between the development team and the stakeholders.  
BA clarifies the requirements, check if the development is going on right track or not.  
BA also participates in scrum meeting

**Testing: -** In the testing phase, the software is tested as that it meets the requirements and is free from defects.   
Testers are involved in this phase.  
Test Case Doc is generated   
BA Works with testing team to ensure that the solution meets requirements  
BA facilitate UAT  
BA helps the users to know the functionality of the system and also helps them to use the system

**Deployment:-**

Once the software has been tested and approved, it is deployed to the production environment.

BA ensures that there is smooth transition from development phase to the production phase.

**Implementation-**

This is the final stage of waterfall model. It involves running the code for the very first time in production phase. Release manager handles this phase.

BA will Update documentation and requirements specifications to reflect changes in the system over time

**Maintenance-**  
Running the code for second time in the production phase is called maintenance. This is done by support team

### **9 What is Conflict Management?**

**Conflict management** means handling problems or fights between people in a smart and peaceful way. The goal is to solve the issue without making things worse.  
**Thomas-Kilmann Technique: -** This technique says there are **5 ways** people usually deal with conflict. It’s like choosing a style based on how much you care about:

* **Your own needs** (called *Assertiveness*)
* **The other person’s needs** (called *Cooperativeness*)

**The 5 Conflict Styles (with examples):**

**Competing**

* + **Focus:** Only on yourself
  + **Like:** “I’m right. I want to win!”
  + **Example:** You insist your idea is best in a team meeting and don’t listen to others.
  + **Use when:** It's urgent or you need to make a strong decision fast.

### **Collaborating**

* + **Focus:** You and the other person
  + **Like:** “Let’s find a win-win solution!”
  + **Example:** You and your teammate work together to create a plan that makes both happy.
  + **Use when:** Both sides have important needs, and you want the best solution.

### **Compromising**

* + **Focus:** Both, but a little give and take
  + **Like:** “Let’s meet halfway.”
  + **Example:** You and your friend decide to split time between two ideas.
  + **Use when:** You need a quick and fair solution.

### **Avoiding**

* + **Focus:** Neither
  + **Like:** “Let’s not talk about it.”
  + **Example:** You stay quiet and don’t join the discussion to avoid conflict.
  + **Use when:** The issue is small or not worth your time right now.

### **Accommodating**

* + **Focus:** Only on the other person
  + **Like:** “It’s okay, we’ll do it your way.”
  + **Example:** You let your coworker lead the project because it means more to them.
  + **Use when:** You want to keep peace or when it’s not a big deal to you.

**10. List down the reasons for project failure**

* Improper Req Gathering
* Continuous change in requirements from Clint
* Lack of user involvement
* Lack of Executive support
* Improper planning
* Poor communication between Ba and stakeholder and Ba and Tech Team
* Lack of risk management

**11. List the Challenges faced in projects for BA**

* Lack of training
* Obtaining Sign off on Doc
* Change Management
* Coordination between developers and Testers
* Conducting Meetings
* Driving clients for UAT
* People Management

**12. Write about Document Naming Standards**

As per IEEE:- Institute of electrical and electronics Engineering

[ProjectID][Document Type]V[x]D[y].extension

Ex:- SF1FRSV1D1.Doc

**13. What are the Do’s and Don’ts of a Business analyst**

* Never say no to client
* There is no word called by default
* Never imagine anything in terms of GUI
* Question the existence of existence
* Requirement hurried project buried.
* Never criticize the stakeholder.
* Always appreciate the stakeholder, even for small efforts.
* Be like a lotus in mud
* Never try to give solutions to the client right away

**14.** **Write the difference between packages and sub-systems**

|  |  |
| --- | --- |
| **Packages** | **Subsystems** |
| Collections of components which are not reused in nature | Collections of components which are reused in nature |
| They are application based | They are Product Based |

**15. What is camel-casing and explain where it will be used**

Camel casing is a naming convention used in programming where

* Multiple words are joined without spaces, and
* Each word (except the first, in lowerCamelCase) starts with a capital letter.

This Camel Casing is a method used as a communication technique or transfer of info between two class

Programming (Code)

OOP

Database

**16. Illustrate the development server, and what are the accesses that the business analyst has?**

**Development server:** It’s a platform for developers and tester where they will develop and test the code and Fix issue

BA will have access to public Doc and Code and Testing in Development server

**17.What is Data Mapping ?**

Data Mapping is a process of connecting data from one place to another place. So that you can you properly in another system or process

Just like Translating from one language to another language, here we transfer data from one format to another.

**Why**

To ensure that the data is correctly transferred or transform and understood when moving from one system to another

**Where it is used**

Data migration   
Data Integration   
ETL process   
API

**Example**

You have data in Excel like this:

| First Name | Last Name | Phone |
| --- | --- | --- |
| John | Smith | 12345 |

You want to send it to a CRM tool that uses this format:

Full Name Contact Number

John Smith 12345

**Data Mapping:**

Full Name = First Name + Last Name  
Contact Number = Phone

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

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

Think of it as a messenger that takes your request to a system, tells it what you want, and then brings the response back.

API integration is the process of connecting two or more applications via their APIs to share data or functionalities seamlessly.

**Example: API Integration in My Application (Date Format Handling)**

Let’s say we’re building an Indian food delivery app that accepts date input in dd-mm-yyyy format (Indian format). Now, we’re integrating with a partner app from the US that sends date in mm-dd-yyyy format.