**PROJECT 3 - PART 1**

**Prepared by: Nilansh Kohli**

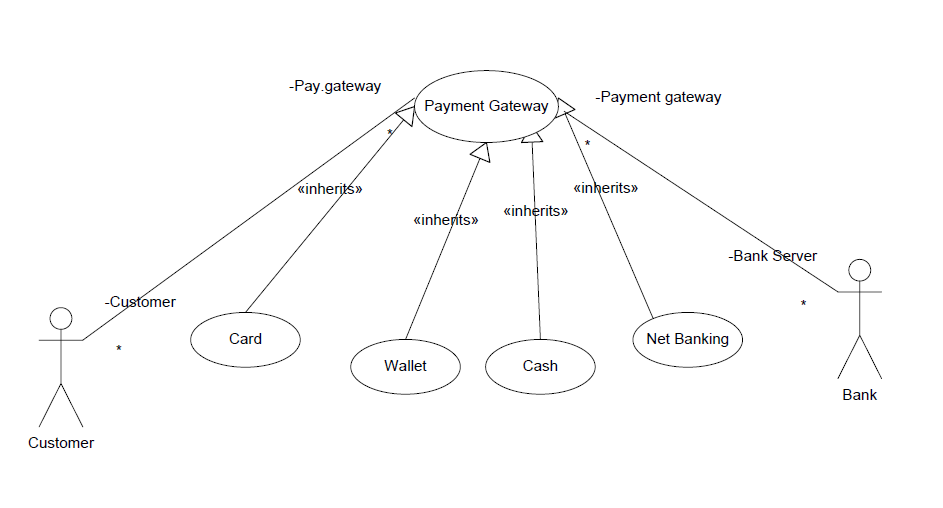
A customer can make a payment either by Card or by Wallet or by Cash or by Net banking.

**Q.1 - Draw a Use Case Diagram**

**Ans –** Use case diagrams are a vital part of UML (Unified Modelling Language) used to model the functional requirements of a system. They showcase the interactions between users (actors) and the system, capturing the system's intended behaviour. Here's an overview of the main symbols and their uses:

**Symbols:**

* Actor: Represented as a stick figure. It signifies any entity that interacts with the system (e.g., users, other systems).
* Use Case: Depicted as an ellipse or an oval. It denotes a specific functionality or behaviour the system must provide.
* System Boundary: Illustrated as a rectangle that encloses the relevant use cases. It marks the scope of the system.
* Association: Shown as a solid line connecting actors to use cases, indicating the interaction.
* Include: Represented as a dashed arrow with the label <<include>>. It signifies that one use case always uses the behaviour of another use case.
* Extend: Depicted as a dashed arrow with the label <<extend>>. It indicates that a use case conditionally includes the behaviour of another use case.
* Generalization: Illustrated as a solid line with a hollow arrowhead pointing to the parent actor or use case. It shows inheritance between actors or use cases**.**



**Q.2 - Derive Boundary Classes, Controller classes, Entity Classes.**

**Answer:**

**Boundary Classes –** It handles all the interactions between external actors and system. It represents the view part of the application.

Here in this case the Boundary Classes are as follows:

* Customer Initiating Payment Boundary
* Card Payment Boundary
* Cash Payment Boundary
* Wallet Payment Boundary
* Net Banking Payment Boundary

**Controller Class –** It acts as intermediary between Boundary Class and Entity Class. Whenever a user send a request for something then it always go through Controller class.

In this case the controller class is

* Payment initiation Controller
* Payment option Controller

**Entity Class –** It represents the core data and Business logic of the application. Here in this case the Entity class are as follows –

* Customer Entity Class
* Card entity Class
* Cash Entity Class
* Net Banking Entity Class
* Wallet Entity Class.

**Q.3 - Place these classes on a three tier Architecture.**

**Ans –** There are certain guidelines that need to follow to place identified classes in three tier Architecture which are as follows –

* All Entity Class should be in Database Layer
* Primary Actor associated Boundary class in Application Layer.
* Controller class should be placed in Application Layer.
* Governing body influenced Boundary class should be placed in Business Logic Layer, else in Application Layer.

Following the above guidelines let’s place theses classes in 3-Tier Architecture –

**Presentation Layer**

This layer handles the user interface for different payment methods.

**Classes:**

* **PaymentUI:** Manages the user interface for selecting and processing payments.
* **CardPaymentUI:** Handles card payment input and processing.
* **WalletPaymentUI:** Handles wallet payment input and processing.
* **CashPaymentUI:** Manages the input for cash payments.
* **NetBankingPaymentUI:** Manages net banking payment input and processing.

**Business Logic Layer**

This layer handles the core functionality and rules for processing payments.

**Classes:**

* **PaymentProcessor:** Base class for processing payments.
* **CardPaymentProcessor:** Implements card payment processing logic.
* **WalletPaymentProcessor:** Implements wallet payment processing logic.
* **CashPaymentProcessor:** Implements cash payment processing logic.
* **NetBankingPaymentProcessor:** Implements net banking payment processing logic.

**Data Access Layer**

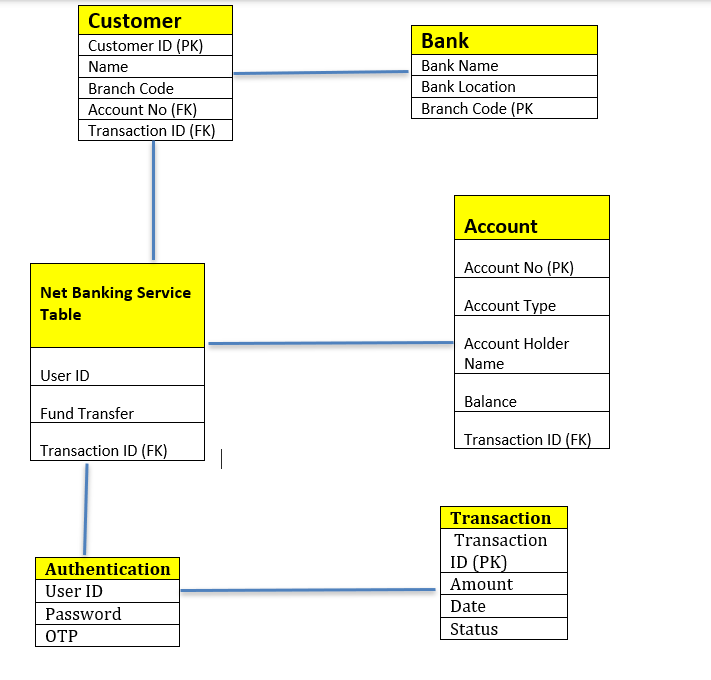
This layer is responsible for interacting with the database to store and retrieve payment-related data.

**Classes:**

* **PaymentDataAccess:** Manages CRUD operations for payment data.
* **CardPaymentDataAccess:** Handles card payment-specific data operations.
* **WalletPaymentDataAccess:** Handles wallet payment-specific data operations.
* **CashPaymentDataAccess:** Handles cash payment-specific data operations.
* **NetBankingPaymentDataAccess:** Handles net banking payment-specific data operations.

**Q.4 -** Explain Domain Model for Customer making payment through Net Banking

**Ans –** A domain model is a conceptual model that represents the entities and relationships within a particular domain - in this case, a customer making a payment through net banking. Here's a simplified example to illustrate the key components involved:

****

Domain Model for Net Banking Payment:

1. **Entities:**

* **Customer:** Attributes: Customer ID, Name, Email, Phone Number, Address, etc.

Relationships: Has one or more Bank Accounts.

* **Bank Account:** Attributes: Account Number, Account Type, Balance, Bank Name, etc.

Relationships: Belongs to a Customer, is linked to one or more Transactions.

* **Transaction:** Attributes: Transaction ID, Amount, Date, Time, Status, etc.

Relationships: Linked to a Bank Account, involves a Payment Method.

* **Payment Method**: Attributes: Method ID, Method Type (Net Banking, Credit Card, Debit Card, etc.), Details, etc. Relationships: Used in one or more Transactions.
* **Net Banking:** Attributes: Net-Banking ID, Bank ID, Username, Password, etc.

Relationships: Is a type of Payment Method.

1. **Relationships:**

* **Customer to Bank Account:** One-to-Many (A customer can have multiple bank accounts)
* **Bank Account to Transaction:** One-to-Many (A bank account can have multiple transactions)
* **Transaction to Payment Method:** Many-to-One (A transaction can involve one payment method)
* **Payment Method to Net Banking**: One-to-One (Net banking is a type of payment method.

**Q.5 -** Draw a sequence diagram for payment done by Customer Net Banking

**Ans –** A **sequence diagram** is a type of **UML diagram** that illustrates how different actors interact in a system over time. It represents the sequence of messages exchanged between objects to accomplish a specific process.

**Purpose of the Sequence Diagram for Net Banking Payment:**

The sequence diagram for **customer net banking payment** details how a customer initiates a payment on a merchant's website using net banking, how the payment is processed through a payment gateway, and how the bank validates and completes the transaction.



**Q.6 -** Explain Conceptual Model for this Case

**Ans –** A Conceptual Model is a high level representation of a system that helps in understanding, visualizing and communicating the essential aspects of a domain. It represents how the concepts are related to other. It provides a clear and simple view of the domain.

The key elements of Conceptual Model are –

* Entities – Customer, payment
* Attributes – Customer Id, Payment ID.
* Relationships – Ex – Customer making a payment.

So in this model it will show like customer is making payment either through Cash, Net Banking, Wallet or cash. All the possibilities of doing these transactions will be represented in this model like frequency, volumes, value and geographical distribution. This entire information together is called as Conceptual Model.

NET BANKING

**Q.7 -** What is MVC architecture? Explain MVC rules to derive classes from use case diagram and guidelines to place classes in 3-tier architecture.

**Ans –** The Model-View-Controller framework is an architectural pattern that divides the application in 3 logical layers i.e. Model, View and Controller.

1. **Model –** It represents the business layer of application. It is a model who is aware of all operations that can be applied to transform that class. It represents enterprise data and business rules that govern access to and updates of this data. All Model class are represented as Entity Class which also includes Database classes and persistent Class (Backend designer).
2. **View -** It represents the presentation of the application. Actors speak to the system and vice-versa through View. View class is represented as Boundary Class.
3. **Controller –** It act as an intermediary between View and Model Class. Whenever a user send a request to the system it always o through the controller. The controller then passes on the request to Model for processing. The processed request is then roll back to controller and the controller redirect it to View to present it to the user in appropriate format.

**MVC Architecture Rules –**

1. Combination of one Actor and one use case results in one Boundary Class.
2. Combination of two actors and one use case results in two Boundary Classes and so on. Primary Actor need to be considered.
3. Use case will result in Controller Class.
4. Each actor will result in one Entity Class.

**The advantages of MVC architecture are as follows:**

* **MVC has the feature of scalability that in turn helps the growth of the application.**
* **The components are easy to maintain because there is less dependency.**
* **A model can be reused by multiple views that provide reusability of code.**
* **The developers can work with the three layers (Model, View, and Controller) simultaneously.**
* **Using MVC, the application becomes more understandable.**
* **Using MVC, each layer is maintained separately therefore we do not require to deal with massive code.**
* **The extending and testing of applications is easier.**

**\*Use case will result in controller class.**

**\*Each actor will result in one entity class.**

**Ex: Model- Customer, Payment, Net banking, Card, cash.**

View- Loginview, paymentoptionview, Netbankingview, Bankselectionview, Credentialsview, Paymentamountview, Paymentconfirmationview, logoutview.

Controller- Logincontroller, paymentoptioncontroller, Netbankingcontroller, Bankselectioncontroller, Credentialscontroller, Paymentamountcontroller, Paymentconfirmationcontroller, logoutcontroller.

**Guidelines to place classes in 3-tier architecture.**

**Presentation layer- This layer is nothing but a user interface. The view is inside this layer. These classes interact directly with the user. The presentation layer is responsible for displaying information and also receiving input from the user.**

**Application layer- This layer is nothing but business logic. model and controller are inside this layer. The controller handles the user input, processes the request, and coordinates the interaction between the model and the view.**

**Data layer- Classes that are responsible for data access and storage are in this layer. It contains the classes which interact with the database, files, and other data sources.**

**Q.8 -** Explain BA contributions in project.

Answer: The Contribution of BA in project are as follows:

Below is table to view BA role in various lifecycle of Project:

**Role of a Business Analyst in a Project**

**Requirements Gathering**

The first step in a project is identifying stakeholders and gathering requirements. During this phase, a **Business Analyst (BA)** and a **Project Manager (PM)** collaborate to collect and document business needs. Once requirements are finalized, a **Business Requirements Document (BRD)** is created.

**Business Requirements Document (BRD)**

A **BA** plays a key role in developing the **BRD** by gathering, analyzing, and documenting business requirements. They conduct stakeholder interviews, define functional and non-functional requirements, and ensure clarity. BAs validate and refine requirements, prioritize them based on business value, and act as a bridge between business and technical teams. They also assess risks, manage scope, and prevent scope creep. Their structured approach ensures the **BRD** is comprehensive, aligned with organizational goals, and serves as a foundation for project success. Ultimately, a **BA** ensures business needs are effectively translated into actionable project deliverables.

**Functional Specification Document (FSD)**

**Design Phase**

Once requirements are finalized, the **design phase** begins. This phase includes a **Detailed Design Document (DDD)** that outlines software architecture, user interface, and system components. Additional documents such as the **Architectural Design Document (ADD)** and solution documents are generated. The **BA** collaborates with designers, architects, and developers to translate requirements into a structured system design, ensuring alignment with business needs and technical feasibility.

**Design Documents and UML Diagrams**

* **Design documents** define the system architecture and structure.
* **UML diagrams** (Unified Modeling Language) visually represent system components, workflows, and interactions.

**Development Phase**

During the **development phase**, programmers or developers code the software based on design specifications. A **BA** serves as a mediator between the **development team and stakeholders**, clarifying requirements and ensuring the development follows the intended path. BAs also participate in **scrum meetings** to track progress and address any issues.

**Testing Phase**

In the **testing phase**, the software is tested to ensure it meets the specified requirements and is free from defects. Testers execute test cases and generate test documents. A **BA** collaborates with the testing team to validate that the solution meets business requirements and facilitates **User Acceptance Testing (UAT)**.

**Test Cases and Test Results**

* **Test cases** ensure functionality meets requirements.
* **Test results** confirm system performance and defect resolution.

**Deployment Phase**

Once testing is complete and the software is approved, it is deployed into the **production environment**. A **BA** ensures a smooth transition from development to production by coordinating with stakeholders and addressing any concerns.

**Implementation Review Document**

* Captures feedback from stakeholders on system performance post-deployment.

**Implementation Phase**

This is the final stage of the **Waterfall model**, where the software is officially launched. The **Release Manager** oversees this phase, while the **BA** updates documentation and requirement specifications to reflect any changes in the system.

**Maintenance Phase**

Once the system is in production, it requires ongoing **maintenance**. Running the system for the second time in production and making necessary modifications is handled by the **support team**. A **BA** plays a role in reviewing user satisfaction and processing **change requests** as needed.

**User Satisfaction Review & Change Request Review**

* Ensures system enhancements align with business needs.
* Facilitates future improvements based on user feedback.

|  |  |  |
| --- | --- | --- |
| **Stages** | **Activities** | **Artifacts & Resources** |
| **Pre project** | Enterprise Analysis - SWOT Analysis, GAP Analysis, Market Research, Feasibility Study, Root Cause Analysis, Decision Analysis, Strategy Analysis, Project Scope and Business Case writing, Risk Analysis | Business Case, Statement of Work, Purchase Order Sr. BA, Business Architects, Pre sales consultants |
| **Planning & Estimation & Assessment** | 1. Understand Assumptions and constraints along with business rules and goals. | PM, BA |
| 2. Plan packages for Big picture |
| 3. Understand Project plan from PM |
| 4. BA conduct stakeholder analysis |
| **Project Initiation** | Plan BA approach Strategy (Req. gathering techniques, communication, Req. management, documents to follow, tools to use, change request handling methodology) | PM, BA |
| **Requirement gathering** | 1. Identifying stakeholders and document | Business Requirement Document |
| 2. Use elicitation techniques | BA, PM |
| 3. gather Requirements |  |
| 4. BRD is Prepared which has business object, scope, business requirements) |  |
| 5. Taking Sin-off on BRD from stakeholders |  |
| **Requirement Analysis** | 1. Validating and Prioritizing the requirements | Functional Requirement Document |
| 2. Draw UML diagrams | SSD (Supplementary Support Document |
| 3. Prepares functional requirements from Business requirement | SRS (Software Requirement Specification |
| Prepare SSD | RTM (Requirement Traceability Matrix |
| SRS will have functional and Technical requirements |  |
| Taking sign-off on SRS, the first legal binding document | BA, PM, Solution Architect, DB Architect, BW architect |
| Preparing RTM from SRS |  |
| **Design** | 1. From Use case BA or Test Manager will prepare Test cases | Solution Documents |
| 2. Communicate with client on the design and solution document | Design Document- HDD - ADD |
| Initiates the preparation of End User Manual |  |
| Updates RTM |  |
| Solution Architecture recommends Architecture of the It-solution | BA, PM, Solution Architect, DB Architect, BW architect, GUI-Designer, Test Manager |
| DB architect comes up with DB schema |  |
| GUI Designer designs all possible screens |  |
| **Development** | 1. BA organizes JAD Session | LDD (Low Level Design Document) |
| 2. BA qualifies queries of Technical Team | CDD(Component Design Document) |
| 3. Developers refers diagram and starts coding |  |
| 4. Updates RTM | Development Team, BA, PM |
| 5. Update end user Manual |  |
| 6.Conduct regular status meeting with technical team |  |
| **Testing** | 1. BA prepares performs high level testing | Test concerning documents |
| 2. BA prepares client for UAT | Application with less errors |
| 3. Updates End User manual |  |
| 4. Updates RTM |  |
| 5. Take sign-off from client on Client project acceptance form | Testing team, BA, PM, Client |
| **Deployment** | 1. Forwards RTM to client to be attached with Project closure document. |  |
| 2. Co-ordinates to complete and share end user Manuals |  |
| 3. Organising Training session for end user |  |
| 4. Prepares Lessons learned from the projects |  |

**Q.9 -** What is conflict management? Explain using Thomas – Kilmann technique

**Ans-** Conflict management is the process of resolving conflicts or disagreements between groups or individuals in a constructive manner. During the project phase there may be chances that stakeholders may have different views and approach and which can result in conflict among them which will ultimately have a negative impact on project. So, these conflicts should be managed in constructive manner and for this the most common technique used is Thomas – Kilmann Technique.

5 steps of Conflict Management -

1. Identify the conflict.
2. Discuss the details
3. Agree with the root problem
4. Check for every possible solution for the conflict
5. Negotiate the solution to avoid future conflict.

Thomas-Kilmann Technique gives 5 options of Conflict Management. In this technique the stakeholders are mapped in terms of Assertiveness on Y- Axis and their level of Co-operation on X- Axis.

1. Competing – people will low on Co-operation and high on Assertiveness. It means standing up for your individual beliefs and simply trying to win.
2. Avoiding – People with level on both co-operation and assertiveness. The individual neither want to address their own problem and neither want ot address the problem of others. In this situation one should avoid the discussion until the time is favourable or simply step back from hazardous situation. Expert meeting required.
3. Accommodating – When people are high on level of co-operation and low on assertiveness, this technique should be used. This involves selfless understanding, generosity or charity. While accommodating individual neglect their problem and address other part’s problems.
4. Collaborating – this is the most beneficial stage where people are both high level in co-operation and assertiveness. It involves deep diving into an issue and locate the critical demands of the concerned individuals or parties. People try to understand the root cause of the problem in this technique.
5. Compromising – When people are average on both assertiveness and co-operation this technique is used. They find a mutually acceptable solution that satisfies both the individuals.

**Q.10 -** List down the reasons for project failure?

**Ans-** There could be many reasons for a project failure which are explained below:

1. Improper Requirement gathering – If the requirements are not properly gathered this could result in deriving wrong functionalities and ultimately will result in project failure. It is the responsibility of business stakeholders and BA to ensure proper requirements are jotted down.
2. Continuous change in requirement – If the client keeps on changing the requirement, this could result over budgeting and exceed in time frame which ultimately will result in project failure.
3. Lack of User involvement – The user involvement is required in every phase of project as the user has to provide sign-off on documents at all the stages and subsequently the team moves to another stage. Also user has to participate actively in req. gathering and analysis. If lack of user involvement is their then this would result in project failure.
4. Lack of executive/management support – The team requires various approvals and support at regular intervals from executives/management. If they do not cooperate properly this could result in project failure.
5. Inadequate Risk management – if the risk management process is defined properly then this can result in project failure.
6. Resources constraint –if the right amount of resources with required skill is not available then this could lead to project failure.
7. Scope creep – if the project is not completed within budget and time then this is also a case of project failure.
8. Unrealistic expectations – if the client comes up with any unrealistic expectation then it can lead to unnecessary usage of time and budget and can result in project failure.

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

The challenges A BA can face are explained below –

1. Lack of training – When the BA is new to industry it requires proper training. It is team member’s responsibility. Lack of training will result improper execution of project.
2. Obtaining sign-off on requirements- It is necessary to obtain sign-off on various documents from the client before proceeding to the next stage. So identifying the right stakeholder for the sign-off could be challenging if the roles and responsibilities of the stakeholders is not clear.
3. Change Management – If there is continuous change in the requirements then a BA has to be very careful as it will involve re-do activities and can affect the time and budget of the project. So proper change management technique should be there.
4. Co-ordination between developers and testers - BA has to establish proper communication between developers and tester as these both are integral part of product development. If the co-ordination is not proper then the built product may not be acceptable to the user.
5. Conducting meetings - A BA has to conduct lot of meetings in every phase of the project to collect and also to provide inputs. So conducting meeting could be a challenge as the required stakeholders may not be available at the same date and time.
6. Scope creep and Scope management - A BA has to ensure that the project gets completed within time and budget and also the objective of the project should be achieved.
7. Documentation - A BA has to prepare many documents and the correctness of these documents is essential to make a record of the same.
8. Time and resources constraint - If the correct amount and right quality resources are not available then this could be challenge to BA as it will lead to more utilisation of time and can result in project failure.

**Q.12 -** Write about Document Naming Standards.

**Ans –** A Document Naming Standard is a systematic approach to assign unique identifiers to various document created and used through the development process.

Ex – Suppose we have a project with ID “ALPHA123” and we are working with Business Requirement Document (BRD). Then to name this document we follow the below steps –

Project ID – BUSINESS123

Document Type – BRD

Version – 1.1

Date – 2025-02-17

Then in the above case the document identifier can be – BUSINESS123-BRD-1.1-2025-02-17.

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

**Ans –**

|  |  |  |
| --- | --- | --- |
| **Sr.No** | **Do's** | **Don'ts** |
| 1 | Consult a SME for clarification | Never Say NO to client |
| 2 | Listen to the client with plain mind with no assumptions | There is no word called 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 they are giving the problem |
| 5 | Considers every problem of client as Unique | Never give solution based on your experience |
| 6 | Question the existence of existence | Don't be washed away by add on functionalities |

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

**Answer:**

**Package**- Collection of components that are not reusable in nature. A package is a grouping and organizing element in which other elements reside, which must be uniquely named. In the UML, packages are used in a manner similar to the way directories and folders in an operating system group and organize files. For example, the project management system may be decomposed into a collection of classes organized into packages as follows.

Ex- Application development companies work on packages.

**Subsystem**- Collection of components that are reusable in nature. Recall that a system is an organized collection of elements that may be recursively decomposed into smaller subsystems and eventually into non-decomposable primitive elements. For example, the project management system may be decomposed into the following: A user interface subsystem responsible for providing a user interface through which users may interact with the system business processing subsystem responsible for implementing business functionality. A data subsystem is responsible for implementing data storage functionality.

Ex- Product development companies work on sub-systems.

While a package simply groups elements, a subsystem groups elements that together provide services such that other elements may access only those services and none of the elements themselves. A subsystem is shown as a package marked with the Subsystem keyword.

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

**Ans –** Camel-casing is a naming convention used in computer programming. It is used in programming language to name different files, functions and identifiers without violating the naming standards. It doesn’t allow spaces in file names or while defining any functions.

Ex- firstName, lastName

In BA, camel-casing is used in requirements documentation. In requirement documentation, BA often uses camel-casing to name the entities like use cases, features, user stories like validateCustomerDetails, calculateInterestRate, and business rules, which should be satisfied by the system use of camel-casing. While documenting business processes or workflows, camel-casing can be used to individual in steps. This will help maintain consistency in the document. The database table name also uses camel-casing. Requirement naming- camel casing is used in the requirement document also, to name the functional and non-functional requirements. Using camel casing in the documents, it helps to maintain consistency in the entire document and also increases readability.

CamelCase 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.

**Q.16 -** Illustrate Development server and what are the accesses does business analyst has?

**Ans -** A development server is a type of server that is designed to facilitate the development process of websites, software or applications for software programmers. It provides a platform to build, test and debug applications before they are deployed to a production environment.

It is divided into 2 parts viz. **Documents and Technology**.

The Documents has again 2 logical partitions – Public and Protected.

Public Document area will have documents like Requirements document, Features documents, BRD, Process Document Areas.

Protected area will have like billing information, escalation raised on associates and any sensitive information

The Technology has 3 logical partitions – Code, Test and Database.

In code area all the coding related server information will be available.

In Test area all the testing related documents or information will be available.

In Database all the databases related to project will be available.

BA will have the access to Public area, Test area and Code Area.

**Q.17 -** What is Data Mapping?

**Ans –** Data Mapping is the process of connecting data from once source to another. Its like creating a map or guide that shows how data in one place corresponds to data in other place. It bridges the differences between two systems or data models. So that when data is moved from one source, it is accurate and usable at destination.

Here are some steps for data mapping:

* Decide which data fields to include
* Establish standard naming conventions.
* Define schema logic or transformation rules.
* Test the logic on a small sample.
* Complete the data map

Data mapping is important for a number of data management tasks, including:

* **Data migration**: Moving data from one source to a new destination.
* **Data integration**: Combining data from multiple sources into a single view.
* **Data transformation**: Converting source data from one format to another.
* **Data warehousing**: Storing data in a single source for analysis and access.

Data mapping connects different data sources so they can work together and speak the same language. For example, if one system calls a customer's age "Age" and another system label it "Birth Year," data mapping would bridge the gap between the two.

**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.

**Ans-** Application Programming Interface is a set of rules and tools that allows different software application to communicate with each other. It acts as an intermediary layer that processes data transfers between systems. It defines the method and data formats that applications can use to request and exchange information.

There are 4 type of API’s – Public, Partner, Private and Composite.

1. Public - A public API is open and available for use by any outside developer or business. It involves moderate authentications.
2. Partner - A partner API, only available to specifically selected and authorized outside developers or API consumers, is a means to facilitate business-to-business activities.
3. Private – A private API is intended only for use within the enterprise to connect systems and data within the business. Ex – Internal HRMS system.
4. Composite - Composite APIs generally combine two or more APIs to craft a sequence of related or interdependent operations.

When our system has date format in dd-mm-yyyy and it is accepting some data from other application having date format in mm-dd-yyyy, then in this scenario API will integration will help to solve the problem so that there is no confusion. When we are receiving such information it will be processed through API and API will convert this format into our system format before displaying it to us. API will work at backend and will process the information internally.