**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   
Answer 🡪  


Q2. Derive Boundary Classes, Controller classes, Entity Classes.   
Answer 🡪

Boundary classes, entity classes, and control classes are types of classes in the Entity-Control-Boundary (ECB) pattern.   
The ECB pattern is used in object-oriented programming to structure classes based on their responsibilities in a use case.

* **Boundary class**

A "boundary class" is a class that lies on the periphery of a system, but within it.   
It interacts with actors outside the system as well as objects of all three kinds of analysis classes within system.  
Example – PaymentOptionBoundary  
 CardPaymentBoundary  
A black background with a black square

Description automatically generated with medium confidence

* **Entity class**

An entity class is a class that is passive; that is, it does not initiate interactions on its own.   
 In business modelling entities represent objects that workers access, inspect, manipulate, produce, and so on. Entity objects provide the basis for sharing among workers participating in different Use Case realizations.  
Example – Customer, Payment  
A black background with a black square

Description automatically generated with medium confidence

* **Control class**

A control class is a class that contains an object which denotes an entity that controls interactions between a collection of objects. A control class usually has behavior specific for one Use case and a control object usually does not outlive the Use Case realizations in which it participates.  
  
A screenshot of a computer program

Description automatically generated

**Q3.** Place these classes on a three tier Architecture.  1. **User Layer:**   
 PaymentMethodSelectionBoundary  
 CardPaymentBoundary  
 2. **Business Logic:**  
 PaymentController   
 WalletController  
 3. **Data Tier:**  
 Customer (Entity Class)  
 Payment(Entity Class)

A screenshot of a computer

Description automatically generated

Q4. Explain Domain Model for Customer making payment through Net Banking  
  
Answer 🡪  
A domain model for a customer making a payment through net banking would include the following:

* **Entities**

The objects in the domain, such as accounts, banks, and transactions

* **Behaviours**

The interactions between the objects, such as debiting an account or issuing a statement

* **Vocabulary**

he terms and phrases used in the domain, such as debit, credit, and portfolio

* **Context**

The assumptions and constraints that apply to the domain, such as only allowing a living person or entity to open a bank account

* **Connections**

The connections between the entities, such as the customer table being connected to the bank table and the payment table

A domain model is a blueprint that shows the relationships between the entities in a problem domain. When creating a domain model, you can start by defining the subject area, which involves identifying the scope, focus, and limits of the domain.

The tables are connected to each other   
In the below diagram   
The Customer table is connected to bank table, which is why the customer is able to make payment.  
The Customer table is connected to payment table, because he should be able to make payment.  
Now the payment is done by net-banking, so payment table is connected to net-banking table.

The Account is in bank, so the account table is connected to bank table.  
The authentication is connected to both net banking table and bank table because authentication is to be performed here.  
Also authentication table is connected to net banking table and bank table, because authentication will be done while transaction.  
Difference between ER Diagram and Domain Model -   
ER Model --- do not have any attributes inside the box  
Domain Model – do have attributes mentioned in 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 – Primarily used in Database design  
Domain Model – used throughout the software development life cycle  
ER Model – focuses on relationships required for storing and retrieving the data  
Domain Model – It focuses on capturing the behaviour of application.  
  
A diagram of a bank

Description automatically generated

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

Description automatically generated  
  
this diagram shows how the objects in the system interact and communicate with each other with time to achieve specific task.  
Developer will draw this.  
It is used to show the Show of messages, events or actions between the objects of the system.  
This diagram helps to visualize the behaviour of the system .This diagram shows the process in detail.

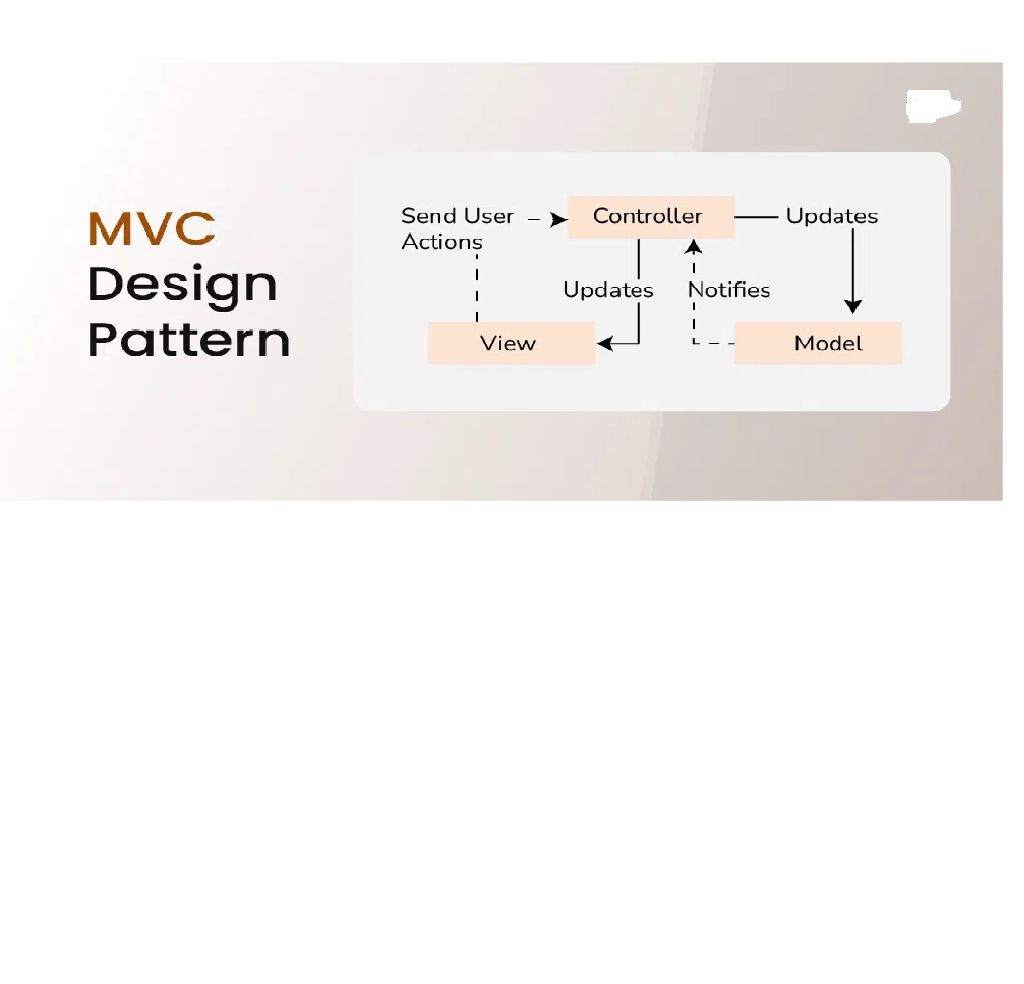
Q6. Explain Conceptual Model for this Case   
  
Answer 🡪  
The conceptual model helps in understanding the key concepts, their relationships, and overall, the structure of the net banking system.  
It serves as a foundation for designing the database schema, defining the application architecture, and implementing the necessary functionalities within the system.  
The relationship between these entities can be described as follows:  
1. Customer: This Node represent the customers or users of net banking services.  
2. Service awareness: Customer should be aware of the available net banking services and their features.  
3. Privacy of Data: the importance/significance of this node is to protect the privacy and confidentiality of customer data in the context of net banking.  
4. Technology awareness: the significance of this node is that customers should be aware and comfortable with the underlying technology used in net banking services  
5. Trust and Support: This node indicates that the bank provide such good services that it will help to enhance the customer trust.  
6. Bank: This node represents a service provider responsible for offering net banking services.  
7. Online Information: This aspect highlights the importance of providing accurate and up-to date information about net banking services to customers.  
8. Security & Privacy: the bank should adapt the security policies which will help the customers to keep their data related to their transaction secure and private.  
9. Infrastructure: This component suggests that the underlying technological infrastructure, including hardware and software systems play an important role in enabling net banking services.  
10. Policies: This node represents the various policies and regulations that govern the implementation and operation of net banking services ensuring compliance and customer protection.  
  
 A diagram of a network

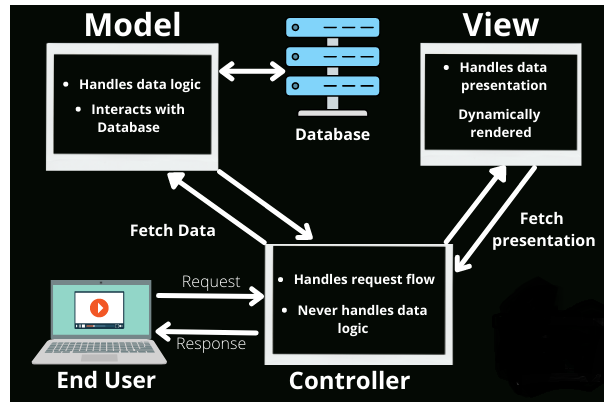
Description automatically generated

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 🡪

MVC, or Model-View-Controller, is a software design pattern that separates an application into three components to make it easier to manage:

* **Model**: Manages the application's data and business logic
* **View**: Presents the model's information to the user
* **Controller**: Manages user input, data operations, and the coordination between the model and view

The **Model-View-Controller (MVC)** framework is an architectural/design pattern that separates an application into three main logical components **Model**, **View**, and **Controller**. Each architectural component is built to handle specific development aspects of an application. It isolates the business logic and presentation layer from each other. 



**MVC Architecture Design:  
Controller:**

The controller is the component that enables the interconnection between the views and the model, so it acts as an intermediary. The controller doesn’t have to worry about handling data logic, it just tells the model what to do. It processes all the business logic and incoming requests, manipulates data using the **Model**component, and interact with the **View**to render the final output.

**Responsibilities:**

* Receiving user input and interpreting it.
* Updating the Model based on user actions.
* Selecting and displaying the appropriate View.

**Example:**In a bookstore application, the Controller would handle actions such as searching for a book, adding a book to the cart, or checking out.

**View:**

The **View**component is used for all the UI logic of the application. It generates a user interface for the user. Views are created by the data, which is collected by the model component, but these data aren’t taken directly but through the controller. It only interacts with the controller.

**Responsibilities:**

* Rendering data to the user in a specific format.
* Displaying the user interface elements.
* Updating the display when the Model changes.

**Example:** In a bookstore application, the View would display the list of books, book details, and provide input fields for searching or filtering books.

**Model:**

The **Model**component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. It can add or retrieve data from the database. It responds to the controller’s request because the controller can’t interact with the database by itself. The model interacts with the database and gives the required data back to the controller.

**Responsibilities:**

* Managing data: CRUD (Create, Read, Update, Delete) operations.
* Enforcing business rules.
* Notifying the View and Controller of state changes.

**Example:**In a bookstore application, the Model would handle data related to books, such as the book title, author, price, and stock level.

**MVC Rules for Identifying Classes:**

1. Combination of one actor and a use case results in one boundary class.
2. Combination of two actors and a use case result in two boundaries class.
3. Combination of three actors and a use case result in three boundaries class

NOTE: Only one Primary actor is to be considered with a use case

1. Use case will result in a controller class.
2. Each Actor will result in one entity class.

**Features of MVC:**

1. It Provides a clear separation of business logic, UI Logic, Input Logic
2. It offers full control over your HTML, and URLs which makes it easy to design web applications architecture.
3. It is a powerful URL-mapping component using which we can build applications that have comprehensible and searchable URLs
4. It Supports Test Driven Development (TDD)

**Guidelines to place identified MVC Classes in a 3 tier Architecture:**

1. Place all the entity classes in DB Layer
2. Place Primary Actor associated with boundary class in Application Layer.
3. Place Controller Class in Application Layer.
4. If Governing Body influence or reusability is there with any of the remaining boundary classes, place them in the business layer.

Q8. Explain BA contributions in project (Waterfall Model – all Stages)   
Answer 🡪  
Software Development is taking place in phases under waterfall Model.

1. Requirement Gathering and Analysis
2. Designing
3. Coding
4. Testing
5. Deployment
6. Testing

A BA is a Facilitator for a Project. The Roles and Responsibilities of business analysts are fundamental in satisfying stakeholder expectations and delivering a viable solution. Here are the critical 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, & Assessment – 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.

1. **Requirement Gathering and Analysis**

Requirements are essential ingredients in any project because they form a foundation upon which project are built. Requirement gathering and analysis is a process that helps define a project's scope, goals, and constraints. It's a crucial part of project planning and can lead to a successful project outcome

Here are some steps in the requirement gathering and analysis process:

* **Identify stakeholders**

Identify the people who have a say in the project, such as business owners, investors, customers, and employees.

* **Gather requirements**

Use techniques like user stories and use cases to identify the project's requirements.

* **Categorize and prioritize requirements**

Group requirements into categories like functional, technical, transitional, and operational.

* **Analyse requirements**

Ensure that the project team can deliver the requirements.   
Requirement gathering and analysis in SDLC is a vital step in the software development process. It involves understanding the client's needs and identifying their problems. It also involves designing solutions. This phase is important to ensure that the final product is perfect. The final software must meet the client's expectations.

In the analysis phase, the development team works closely with the client. They conduct workshops to understand the business needs of the client. This helps them understand the objectives and requirements of the business developers to get into the technical specifications. This helps them in the development process.

* **Document requirements**

Record the results of the requirement gathering and analysis in a document

* **Design**  
  After gathering all the requirements, it's time to move on to the design stage. Here, designers develop solutions that meet the requirements. In this stage, designers:
* create schedules and project milestones.
* determine the exact deliverables.
* create designs and/or blueprints for deliverables.
* Deliverables could include software, or they could consist of a physical product. For instance, designers determine the system architecture and use cases for software. For a physical product, they figure out its exact specifications for production.
* **Implementation/Coding**

Once the design is finalized and approved, it's time to implement it. Design hands off their specifications to developers to build.

To accomplish this, developers:

create an implementation plan.

collect any data or research needed for the build.

assign specific tasks and allocate resources among the team.

Here is where you might even find out that parts of the design that can't be implemented. If it's a huge issue, you must step back and re-enter the design phase.

* **Testing**

After the developers code the design, it’s time for quality assurance. It’s important to test for all use cases to ensure a good user experience. That's because you don't want to release a buggy product to customers.

QA also:

writes test cases.

documents any bugs and errors to be fixed.

tests one aspect at a time.

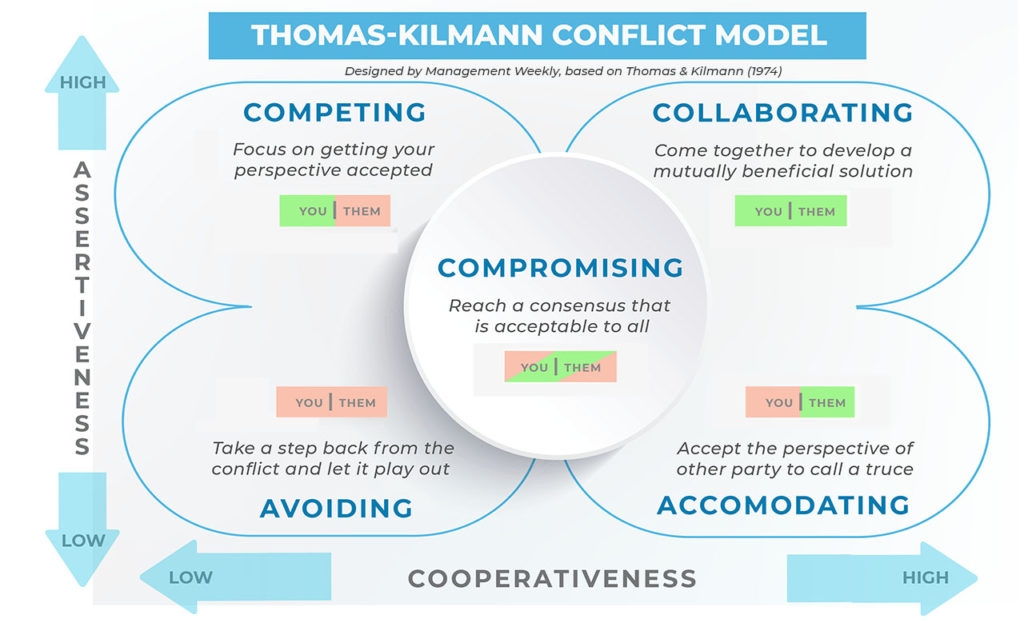
determines which QA metrics to track.

covers a variety of use case scenarios and environments.

* **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 manual, plans and organizes training sessions for end users. BA prepares lessons learned from this project to take precautions for upcoming projects.  
    
    
    
  Q9. What is conflict management? Explain using Thomas – Kilmann technique

Answer 🡪 Conflict management is the practice of being able to identify and handle conflicts sensibly, fairly, and efficiently. It is a process of dealing with disagreements arising from. For example, diverging opinions, objectives and needs.  
The aim of conflict management is to enhance learning and group outcomes including effectiveness or performance in an organisational setting.  
Thomas-Killman Technique  
Researchers Kenneth Thomas and Ralph Kilmann developed a model for resolving conflicts. This model is known as the Thomas – Killman model.

Conflicts occurs whenever people disagree. The disagreement could be over their perceptions, ideas, values, motivations, or desires. This model is based on two dimensions of conflict management: assertiveness and empathy.  
Based on these two dimensions, there are five conflict resolution strategies:

Competing, Avoiding, Accommodating, Collaborating, and Compromising.  


**Competing**

At the top left end of the chart, we have high assertiveness and low cooperation. We find competing right at this spot. This means that we use ‘competing’ as a conflict resolution strategy when we resort to being assertive. Additionally, we also become uncooperative with the opposing party. This mode may be more appropriate when we need speedy resolution, and we are a higher position of power. It is also imperative to use this mode when we must take tough calls.

**Avoiding**

The avoiding mode reflects passivity in conflict management. Although, from an ideal perspective, we would agree that we should face the problems, head on. However, the most confident of us would have been in situations where confrontation was best avoided. Also, some problems are so trivial, they are not worth your time.

This approach involves shunning the conflict and withholding one’s views and opinions. People also employ this approach when the cost of confrontation and resolution is far more than the cost of living with the conflict. However, avoiding does not resolve the conflict. On the downside, it just buries it below the surface and may potentially lead to future conflicts.

**Accommodating**

Now, we are gradually moving towards more amicable means of resolving conflicts. Perhaps, accommodating mode is so considerate of the opposing party that you are ready to call it a truce at the cost of giving up your position. It may be worthwhile in situations when the conflict is a total waste of your time.

The Thomas and Kilmann Conflict Model positions this mode as high in cooperativeness and low in assertiveness. We sacrifice our own stance to make the other party happy.

**Collaboration**

A combination of being high assertive and cooperative, those who collaborate attempt to work with others to identify a solution that fully satisfies everyone’s concerns. In this style, which is the opposite of avoiding, both sides can get what they want, and negative feelings are minimized. “Collaborating works best when the long-term relationship and outcome are important

**Compromising**

Finally, we arrive at the literal middle ground in the Thomas Kilmann Conflict Model. The centre of the chart shows us a point where we are moderately assertive and moderately cooperative. In certain situations, we may not want to have an extended resolution. Collaborating takes time. Also, in some situations, we don’t want to be too confrontational but at the same time, we want to take a stand as well.  
  
  
Q10. List down the reasons for project failure

Answer 🡪  
  
**1. Unclear objectives**

**Problem:** Your team isn’t aligned on project goals, and there’s no way to measure success.

Project objectives are the things you plan to achieve by the end of your project. They should be specific, time-bound goals you can measure when your project is finished. Without clear objectives, it’s hard to keep your team aligned or even know if your project was a success or a failure.

For example, imagine your team is designing a new checkout page for your mobile app. Without a clear objective (such as “reduce average checkout time for end users by 30% in Q2"), it’s hard to know which new features will make the page a success. And after the project is over, you’ll be hard-pressed to measure performance without a concrete goal to compare with.

**2. Scope creep**

**Problem:** Your project deliverables change as work progresses.

Scope creep is hard to spot because it often comes on slowly—you could even say it *creeps up on you*. It’s what happens when project deliverables exceed the project scope, and you end up with more work than you bargained for.

For example, imagine you planned to publish 10 blog articles this month as part of a new product launch. However, you got a request from a stakeholder to add two additional posts to support a different product. With that new ask, your resources are stretched thin and you need to delay publishing deadlines across the board.

**3. Unrealistic expectations**

**Problem: Success isn’t attainable.**

Inspiring goals can help spur forward momentum, but they should still be attainable. If your project objectives are too ambitious, stressed teammates and missed deadlines can easily ensue.

For example, imagine your sales team has a stretch goal of 100 commissions this month. But two team members are on PTO, so the rest of the team will need to work overtime to achieve the goal. That means there aren’t enough resources to achieve the objective, and success is likely out of reach

**4. Limited resources**

**Problem:** You don’t have the resources you need to get the job done.

Resources are anything you need to complete a project—like budget, staff, time, space, or tools. A lack of resources can delay a project or even stop it in its tracks.

For example, imagine you’re working on an ad campaign for a new product. The deadline is approaching, but your budget for freelance video editing has run dry. With only one in-house editor to help, you need to delay the campaign’s launch. In this case, you’ve run out of both the budget and manpower you need to deliver work on time.

Solution: Make a resource management plan.

**5. Poor communication**

**Problem:** Team members don’t understand how and when to communicate updates.

Nowadays, communication is more complex than ever.   
Solution: Create—and share—a communication plan.

A communication plan typically includes the frequency, channel, audience, and owner for each type of communication you’ll be using. For example, you might include these details for weekly project status updates:

* Communication type: project status updates
* Frequency: weekly
* Channel: email
* Audience: project team
* Owner: project manager

**7. Lack of transparency**

**Problem**: Team members can’t find important project documentation.

So, you’ve crafted a killer project plan complete with a project schedule, communication plan, resource management plan, and SMART objectives. Now what?

If those documents are static and not easily accessible to your team, it can be hard to communicate updates without time-consuming status meetings. And even so, information might get lost in the mix.

Solution: Use a work management tool to house project info in one place.

Q11. List the Challenges faced in projects for BA

Answer 🡪  
Here are some challenges that business analysts (BAs) may face in projects:

* **Communication**: Poor communication can lead to project issues. BAs should ensure effective communication with everyone involved in the project.
* **Conflicts with stakeholders**: Stakeholders may have conflicting expectations of what the project will deliver. BAs can help by ensuring that all stakeholders have a shared understanding.
* **Inadequate resources**: A lack of resources can lead to project delays and compromised quality. This can include inadequate staffing, limited budget, insufficient equipment, or a lack of required expertise.
* **Absence of clear objectives**: Undefined goals can be a common challenge. Stakeholders may not have a clear idea of what they want, or they may not agree.
* **Lack of domain knowledge**: BAs need to collaborate with business users to understand the requirements.
* **Changing business needs or requirements**: Business needs or requirements may change.
* **Unrealistic timelines**: Timelines may be unrealistic.
* **Conflict with users**: BAs may face conflict with users
* **Lack of Training**: It includes all those areas which are directly or indirectly related to the project by efficient training to the people involved in the project we can produce a good and efficient result for the organisation.
* **Obtaining Sign off on Requirements**: This is one of the biggest major areas where BA struggles for while taking sign off on requirements. It is necessary for him/her to convince them to the best that all the requirements will be satisfied for taking sign off they have to keep on approaching the clients and be in touch, book an appointment to have the sign off on requirements as well.
* **Coordination between Developer and Testers:** In any project if the coordination between developers and testers is not well managed then the project success could not be assumed in any project it is must to create a coordination between both these teams in an efficient wat

Q12. Write about Document Naming Standards

Answer 🡪

1. Keep the file names short but meaningful
2. Include any unique identifiers. E.g. case number, project title.
3. Be consistent
4. Indicate version name where appropriate.
5. Ensure the purpose of the document is Quickly and easily identifiable
6. All Documents will be named using some standards like [Project ID][Document Type]VD[Y].ext.  
   For Ex: PQ298BRDV1D1.docx

PQ298BRDV1D1.2.docx  
 PQ536BRDV1D1.docx

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

Answer 🡪

|  |  |
| --- | --- |
| **DO’s** | **Don’ts** |
| Listen carefully | Do not interrupt the client, when he/she is giving you the problem |
| Be data-driven | Never try to give solutions to client straight away with your previous experience and assumptions Make assumptions |
| Question Everything | Should not be hurry. |
| Consult an SME for clarifications in Requirements | Should not miss any requirement |
| Every problem of client is unique | Should know what the Scope of the Project is. |
| Maximum try to extract the leads to solution from the client itself. | Ignore feedback |
| BA should focus on “what” and “when” to develop rather than focus on “how” to develop | Use jargon |
| Communicate clearly | Never say No to client |
| Be adaptable | Never imagine anything in terms of GUI |
|  |  |

Q14. Write the difference between packages and sub-systems

Answer 🡪  
 **Packages:**  
A Package is a grouping and organising element reside which must be uniquely named. In the UML, packages are used in a manner like the way directories and folders in an OS group and organize files.  
It is a collection of components which are not reusable in nature.  
  
**Sub-Systems:**Collection of components which are reusable in nature.  
system is an organized collection of elements that may be 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.  
A Business processing subsystem responsible for implementing business functionality.  
A Data subsystem responsible for implementing data storage functionality.  
While a package simply group of elements, a subsystem groups element that together provide services such that other elements may access only those services and none of the elements themselves  
E.g. Tally Software is one of the examples of subsystem.  
  
Packages and subsystems are both organizational units in software, but they have different characteristics:

* **Packages**

A collection of source files and headers that provide related functionality. Packages are model organizational units that only exist during design time.

* **Subsystems**

A collection of one or more packages that are part of an overall system. Subsystems are organizational units for parts that will be implemented, so they exist as real-world elements at run-time.

Here are some other differences between packages and subsystems:

* **Representation in UML**

In the UML, packages are represented as folders, and subsystems are stereotyped packages with the stereotype of <<subsystem>>.

* **Subsystem model**

A subsystem model is a systems-model-in-the-small that includes packages for subsystem requirements and internal components.

* **Subsystem design**

The contents and internal behaviours of a subsystem can be changed, as long as the subsystem's interfaces remain constant.

* **Subsystem benefits**

Subsystems provide a "replaceable design" element, and they emphasize the intent to encapsulate behaviour and hide internal details.

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

Answer 🡪  
  
Camel case is a writing style that combines words without spaces or punctuation, with the first letter of each word capitalized. It's commonly used in computer programming, web development, and other areas, such as text messaging and instant messaging.

Here are some reasons why camel case is used:

* **Code readability**: Camel case makes code easier to read and understand by combining words into one term.
* **Error detection**: Capital letters throughout the code make it easier to spot typos or errors.
* **Domain name legibility**: Camel case can make multi-word domain names more legible.
* **Online usernames**: Camel case is sometimes used in online usernames.

Some examples of camel case include:

goThere(), getEmpId(),

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

Answer 🡪

A Development Server is a type of server used for creating and testing software applications before they are deployed to production. It is part of a set of servers that also includes production servers, test servers, and other specialized servers.  
  
**The accesses a BA has are**-  
Read-only- BA’s may be granted with the read-only access to the development server.   
This will allow them to view the user interface of the Application, navigate through the features and also they will be able to observe the behaviour of the application.  
Limited Access-Depending upon the project needs, the BA’s will be granted limited access to the specific modules in the Application.  
Limited Configuration Access- Means BA have the authority to make changes in certain areas of application where they have the access.  
Ex: Confluence, Jira, Project Folders, etc

Q17. What is Data Mapping?  
Answer 🡪

Data mapping is the process of matching fields from one database to another. It's the first step to facilitate data migration, data integration, and other data management tasks.

Before data can be analysed for business insights, it must be homogenized in a way that makes it accessible to decision makers. Data now comes from many sources, and each source can define similar data points in different ways.

Data mapping bridges the differences between two systems, or data models, so that when data is moved from a source, it is accurate and usable at the destination.

Data mapping is an essential part of many data management processes. If not properly mapped, data may become corrupted as it moves to its destination. Quality in data mapping is key in getting the most out of your data in data migrations, integrations, transformations, and in populating a data warehouse

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 1  
Answer 🡪  
  
**An API, or application programming interface**, is a set of rules or protocols that enables software applications to communicate with each other to exchange data, features and functionality.  
**API is the acronym for application programming interface** — a software intermediary that allows two applications to talk to each other. APIs are an accessible way to extract and share data within and across organizations. APIs are all around us.  
It’s useful to think about API communication in terms of a request and response between a client and server.   
The application submitting the request is the client, and the server provides the response. The API is the bridge establishing the connection between them.  
  
**Why API is used?**  
APIs are needed to bring applications together to perform a designed function built around sharing data and executing pre-defined processes. They work as the middleman, allowing developers to build new programmatic interactions between the various applications people and businesses use daily.  
  
To use API integration when your application's date format is dd-mm-yyyy and you're accepting data from another application with a date format of mm-dd-yyyy, you can use the ISO 8601 format:

* **ISO 8601 format**

This is a well-known and widely used format that can be handled across many different languages.

* **REST API date input**

The accepted ISO 8601 UTC formats are:

* + YYYY-MM-DD
  + YYYY-MM-DDThh:mm <TZDSuffix>
  + YYYY-MM-DDThh:mm:ss <TZDSuffix>

Here are some other date formatting details:

* **API date formatting**: YYYY is the 4-digit year, MM is the 2-digit month (zero-padded), and DD is the 2-digit day (zero-padded).
* **Salesforce**: The valid date and Date Time format is YYYY-MM-dd.
* **Automation Anywhere**: The dd-MMM-YYYY custom datetime format might show incorrectly on computers where the display language is set to Simplified Chinese or Traditional Chinese

Let’s understand it with an Example:  
You want to book your flight ticket using amazon/Paytm of indigo flight for Mumbai to Goa destination, but the amazon/Paytm have not access of indigo company, in this case via API the application show the require details like available data/seat etc. API is the intermediate of two application using API we can go for the booking

An API used to integrate new applications with existing software system. It serves as the connection between two applications, letting them exchange data. This increase development speed because each functionality doesn’t have to be written from scratch.