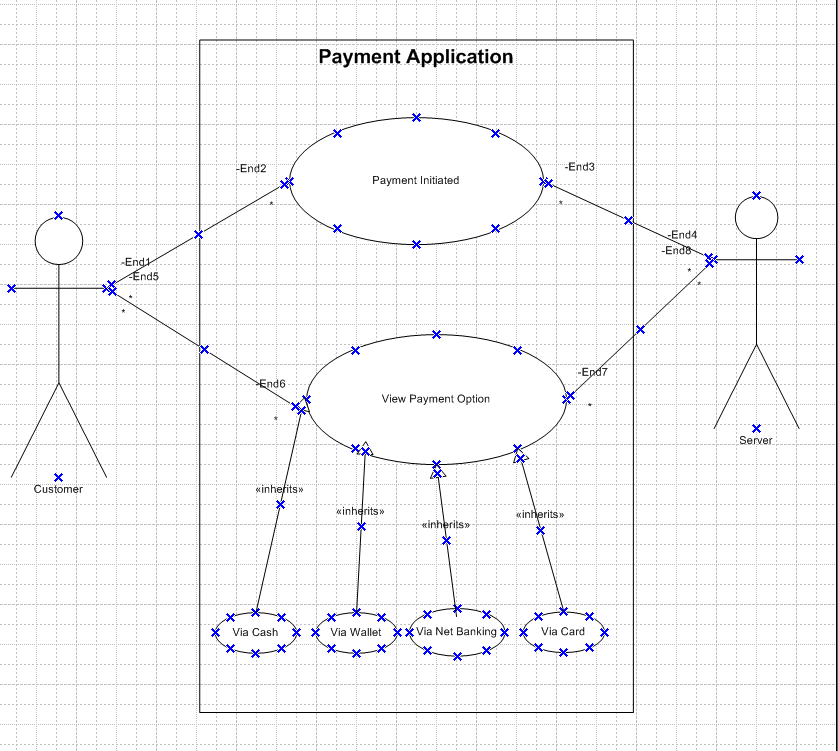
* + - 1. **Use Case Diagram**



1. **Derive Boundary Classes, Controller classes, Entity Classes**

**Boundary Classes**

Boundary classes are a type of class in software design, particularly in Object-Oriented Analysis and Design, which act as intermediaries between the system and external entities. These external entities could be users, external systems, or devices. The role of a boundary class is to handle the interaction between the system and the outside world.

Boundary classes are denoted by below symbol –

Characteristics of boundary classes –

1. Interface Handlers: They manage inputs to and outputs from the system, such as user interface elements, API endpoints, or file I/O.
2. Interaction Focus: They focus solely on the interaction logic, leaving business logic to other classes (usually entity classes).
3. Transient: Typically, short-lived and only active during specific interactions.4. Decoupling: They decouple external systems or user interfaces from the internal business logic.

Examples – Payment Option Boundary, Card Payment Boundary

**Controller Classes**

Controller classes are a type of class in Object-Oriented Analysis and Design (OOAD) that handle the flow of control and manage the interactions between Boundary classes (responsible for system interfaces) and Entity classes (responsible for data and business logic). Their primary role is to process incoming requests, coordinate activities, and direct data between other components of the system.

Controller classes are denoted by below symbol –

Characteristics of Controller Classes

1. Orchestrators: They do not perform business logic directly but delegate it to entity classes and manage the overall process flow.
2. Interaction Coordinators: Facilitate communication between boundary classes(e.g., user interfaces, APIs) and entity classes.
3. Independent of UI and Data: They are designed to be decoupled from specificuser interfaces or data structures, making them reusable and adaptable.
4. Transient: Typically instantiated during specific use-case executions anddisposed of afterward.

Examples – Payment Initiated Controller, Card Payment Controller

**Entity Classes**

Entity classes are a type of class in software design, particularly in Object-Oriented Analysis and Design (OOAD), that represent the core business objects of a system. These classes encapsulate data and the logic (business rules) that operate on that data Entity.

Classes are denoted by below symbol –

Characteristics of Entity Classes –

1. Core Domain Representation: Represent real-world concepts, business processes, or system components.
2. Data-Centric: Store data attributes that describe the business object.
3. Business Logic: Contain methods to enforce business rules and ensure data integrity.
4. Persistent: Typically persist in a database or other storage system.
5. Reusable: Can be used across various use cases and workflows because they model core system concepts.

Examples – Customer, Payment

1. **Place these classes on a three-tier Architecture.**

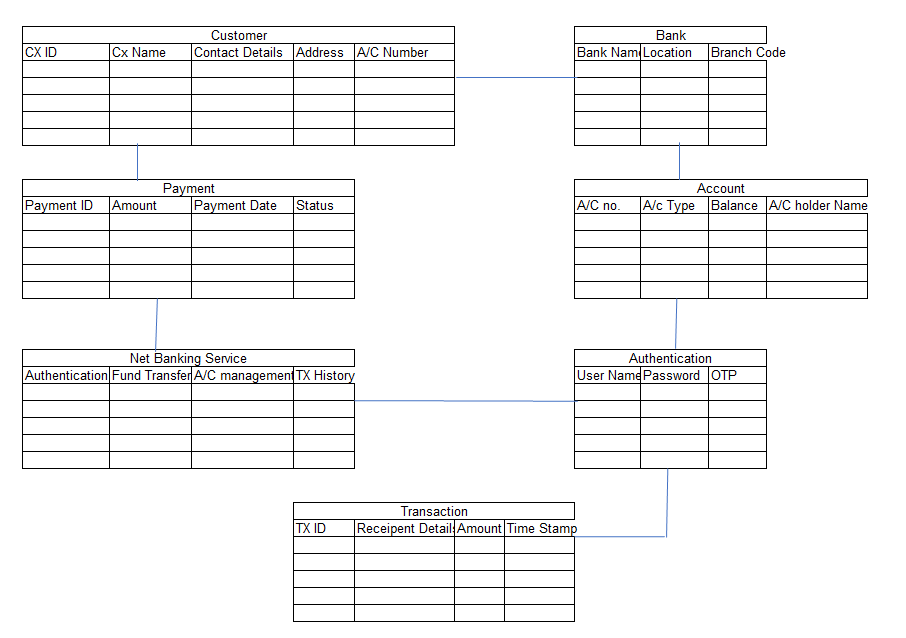
3-Tier Architecture is a software design pattern that separates an application into three logical layers: Presentation Tier (user interface), Application Tier (business logic), and Data Tier (database). The presentation tier handles user interaction, the application tier processes business rules and workflows, and the data tier stores and retrieves information. This separation improves scalability, maintainability, flexibility, and security, allowing each tier to be developed, managed, and scaled independently.

We can place the classes derived from Boundary classes, Controller classes and Entity classes into 3 levels of 3 tier architecture-

1. Presentation Tier (User Interface) – Presentation layer will contain boundary classes i.e. Payment Option Boundary and Card Payment Boundary classes.
2. Application Tier (Business Logic) – Application layer will contain controller classes i.e. Payment Initiated Controller, Card Payment Controller classes.
3. Data Tier (Database) – Data layer will contain the entity classes i.e. Customer and Payment classes
4. **Domain Model for Customer making payment through Net Banking**

A domain model is a conceptual representation of the key entities, their attributes, relationships, and behaviors within a specific problem domain. It serves as a blueprint to understand and design the core functionality of the system by modelling real-world objects and their interactions. The domain model is typically visualized using UML (Unified Modelling Language) class diagrams or similar tools and is central to object-oriented software design. It bridges the gap between the business requirements and the technical implementation.

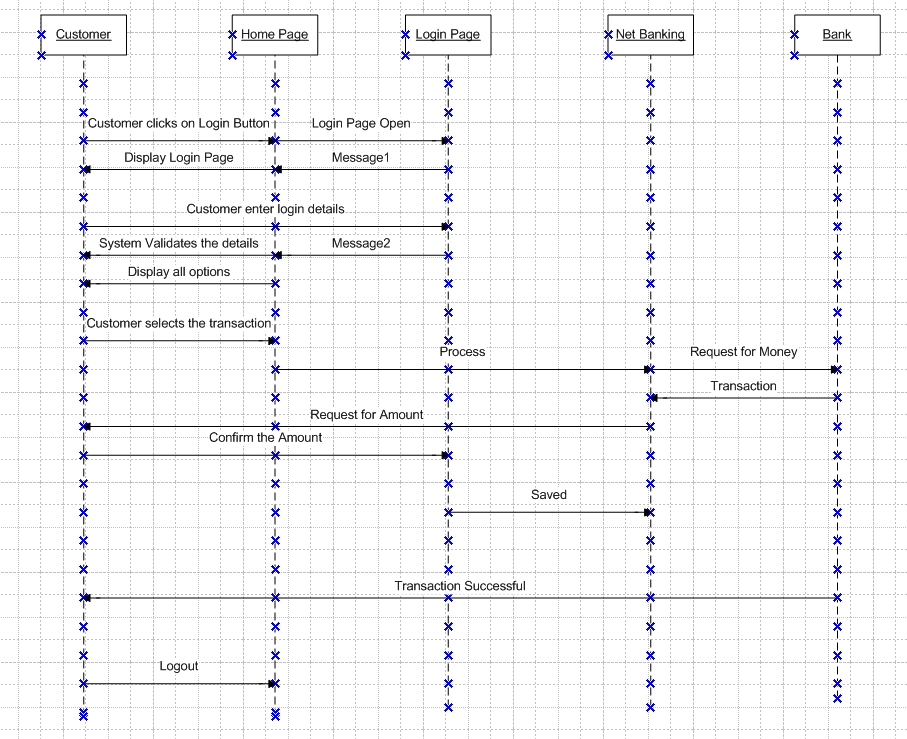
Domain model diagram of Customer making payment through net banking is below



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

A Sequence Diagram is a type of UML (Unified Modeling Language) diagram that visually represents the interaction between different components or objects in a system over time. It focuses on the sequence of messages or events exchanged between participants (e.g., users, systems, or software components) to achieve a specific functionality or use case.

Sequence diagram for Customer Net Banking is drawn below -



1. **Conceptual Model for this Case**

A Conceptual Model is a high-level, abstract representation of the system or domain that illustrates the key concepts, entities, relationships, and overall structure of the problem space without focusing on technical details or implementation specifics. It serves as a bridge between the business perspective and the technical design.

The relationships between these entities can be described as follows with the help of diagram-

A diagram of a company

Description automatically generated

* Customer - This node represents the customers or users of net banking services.
* Service Awareness - Customers should be aware of the available net banking services and their features.
* Privacy of Data - The importance/significance of this node is to protect privacy and confidentiality of customer data in the context of net banking.
* Technology Awareness - The significance of this node is that customers should be aware and comfortable with the underlying technology used in net banking services.
* Trust & Support - This node indicates that the bank provides such good services that it will help to enhance the customer’s trust.
* Bank - This node represents a service provider responsible for offering net banking services.
* Online Information - This aspect highlights the importance of providing accurate and up-to-date online information about net banking services to customers.
* Security & Privacy - the bank should adopt the security policies which will help the customers to keep their data related to their transaction secure and private.
* Infrastructure - This component suggests that the underlying technological infrastructure, including hardware and software systems, plays an important role in enabling net banking services.
* Policies - This node represents the various policies and regulations that govern the implementation and operation of net banking services, ensuring compliance and customer protection.

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

MVC Architecture - MVC (Model-View-Controller) Architecture is a software design pattern that separates an application into three interconnected components: the Model, which manages the application’s data and business logic; the View, which handles the user interface and presentation of data; and Controller, which processes user input, updates the model, and determines how information is displayed in the view. This separation of concerns improves modularity, scalability, and maintainability by allowing each component to be developed, tested, and modified independently while working cohesively to handle user interactions and system workflows.

3 Parts of MVC architecture are described below –

1. Model-

* Represents the application's core logic and data.
* Responsible for retrieving, storing, and processing data (e.g., through databases or APIs).
* Independent of the user interface.

1. View-

* Represents the presentation layer or the user interface.
* Displays data from the model to the user and sends user input to the controller.
* Examples: Web pages, mobile app screens, or desktop GUIs

1. Controller-

* Acts as the intermediary between the model and the view.
* Handles user input, processes it, and determines the appropriate response.
* Updates the model or view as needed.

Rules to derive the classes from use case diagram-

1. Combination of One actor and a use case results in one boundary class.
2. Combination of Two actors and a use case results in two boundary class.
3. Combination of Three actors and a use case results in three boundary class.
4. Use case will result in a controller class.
5. Each actor will result in one entity class.

For example, we take scenario of customer making payment either by card or wallet or by cash or net banking –

1. Model Classes – Customer, Payment, Net Banking, Card, Cash
2. View Classes - Login View, Payment Option View, Net Banking View, Bank Selection View, Credentials View, Payment Amount View, Payment Confirmation View, Logout View
3. Controller Classes – Login Controller, Payment Option Controller, Net Banking Controller, Bank Selection Controller, Credentials Controller, Payment Amount Controller, Payment Confirmation Controller, Logout Controller

Guidelines to place identified MVC classes in a 3 Tier Architecture –

1. Place all entity classes on DB layer.
2. Place Primary actor associated Boundary class in Application Layer
3. Place Controller class in application layer.
4. If Governing body influence or reusability is any of the remaining Boundary class place them in Business Logic layer or else place them in Application layer
5. **Explain BA contributions in project (Waterfall Model – all Stages)**

The Waterfall Model is a traditional software development methodology that follows a linear and sequential approach. The process is divided into distinct phases: Requirements Gathering, Requirement Analysis, Design, Development, Testing, Deployment, Implementation and Maintenance, with each phase completed before moving to the next. Progress flows in one direction, like a waterfall, making it best suited for projects with well-defined and unchanging requirements. While the Waterfall Model provides structure and simplicity, it is less flexible in accommodating changes, which can make it less effective for dynamic or complex projects.

Waterfall model has following stages and corresponding Business Analyst responsibilities have also been described –

1. Requirements Gathering

* First, the stakeholders are identified.
* In this phase, all the requirements are gathered from the stakeholder.
* BA and Project Manager participate in this phase.
* After completing this phase, BRD will be generated.

1. Requirements Analysis

* The requirements are analyzed to understand the scope of the project.
* Analyzing means the BA will check all the requirements, if he finds conflicting requirements then the BA will talk to the concerned stakeholder to clear it, remove the ambiguous requirements.
* BA will prepare functional requirements.
* The document which contains the functional requirements is called FRS (Functional Requirement Specifications)
* Technical team will prepare non-functional requirements.
* The document which contains the non-functional requirements is called SSD (Supplementary Support Document).
* BA will combine FRS and SSD to form SRS (Solution Requirement Specifications).
* BA will prepare RTM by referring SRS.

1. Design

* After the requirements are cleared, Design phase starts.
* This has a detailed design document that outlines the software architecture, user interface, and system components.
* HDD and solution documents will be generated here.
* 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.

1. Development

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

1. Testing

* In the Testing phase, the software is tested to ensure that it meets the requirements and is free from defects.
* Testers are involved in this phase.
* Test documents are generated here.
* BA works with the testing team to ensure that the solution meets the requirements.
* BA facilitates UAT.
* BA helps the users to know the functionality of the system and helps them to use the system.

1. 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.

1. Implementation

* This is the final stage of Waterfall.
* It involves running the code for the very first time in the production phase.
* Release manager handles this phase.
* BA will Update documentation and requirements specifications to reflect changes in the system over time.

1. Maintenance

* +Maintenance is provided once after the whole deployment for any unforeseen issues.
* This is done by a support team.

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

Conflict Management is the process of identifying, addressing, and resolving disagreements or disputes in a constructive manner to prevent escalation and maintain productive relationships. It involves techniques and strategies to handle conflicts in a way that minimizes negative outcomes while fostering collaboration, understanding, and growth among individuals or teams.

Key Objectives of Conflict Management are:

* Resolve Disputes Constructively: Focus on solutions that satisfy all parties involved.
* Maintain Relationships: Preserve trust and respect between individuals or teams.
* Improve Collaboration: Use conflict as an opportunity to generate new ideas and better understanding.
* Enhance Productivity: Minimize disruptions caused by unresolved conflicts.
* Promote Healthy Communication: Encourage open dialogue and mutual respect

Thomas - Kilmann approach –

The Thomas-Kilmann Conflict Management Model is a framework developed by Kenneth W. Thomas and Ralph H. Kilmann that identifies five primary styles of handling conflict based on two dimensions: assertiveness (the degree to which a person seeks to satisfy their own concerns) and cooperativeness (the degree to which a person seeks to satisfy the concerns of others). The model emphasizes that there is no one-size-fits-all approach, and the choice of style depends on the situation and goals.

There are 5 Conflict Management Styles as per the Thomas Kilmann technique -

1. Competing (High Assertiveness, Low Cooperativeness) –

* Description: The individual pursues their own interests at the expense of others.
* Use Case: Best for quick, decisive action or when the outcome is critical, such as during emergencies.
* Example: Enforcing a strict deadline despite objections

1. Accommodating (Low Assertiveness, High Cooperativeness)

* Description: The individual puts the other party's needs above their own
* Use Case: Best when maintaining relationships is more important than the issue at hand or when the issue is trivial to you.
* Example: Agreeing to a colleague's suggestion to avoid conflict

1. Avoiding (Low Assertiveness, Low Cooperativeness) –

* Description: The individual sidesteps or postpones the conflict without addressing it directly.
* Use Case: Best when the issue is minor, emotions are high, or there’s a need to gather more information.
* Example: Delaying a heated discussion until emotions cool down.

1. Compromising (Moderate Assertiveness, Moderate Cooperativeness) –

* Description: Both parties make concessions to reaching a mutually acceptable solution.
* Use Case: Best when a quick, temporary solution is needed or when both parties have equally strong positions.
* Example: Splitting resources evenly between two departments.

1. Collaborating (High Assertiveness, High Cooperativeness) –

* Description: The individual works with the other party to find a win-win solution that fully satisfies both sides.
* Use Case: Best when the issue is important to both parties and requires a creative, long-term solution.
* Example: Jointly designing a new project strategy that incorporates everyone's ideas

Benefits of Thomas Kilmann Technique –

* Flexibility - Encourages using different styles depending on the situation.
* Awareness - Helps individuals understand their natural conflict managementtendencies.3. Resolution Oriented - Provides a structured approach to handle conflicts effectively

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

Projects can fail for a variety of reasons, often due to poor planning, mismanagement, or unforeseen circumstances. Below is a list of common factors that contribute to project failure:

1. Poor Project Planning

* Lack of clear goals and objectives.
* Insufficient detail in the project plan.
* Unrealistic timelines and resource estimates.
* Failure to identify and address potential risks.

1. Inadequate Requirements Management

* Misunderstanding or incomplete requirements.
* Frequent changes in requirements without proper change control.
* Lack of stakeholder involvement in defining requirements.

1. Weak Leadership and Governance

* Inexperienced or unengaged project managers.
* Lack of executive sponsorship or support.
* Poor decision-making or lack of accountability.

1. Ineffective Communication

* Breakdown in communication between stakeholders and team members.
* Unclear or inconsistent instructions and updates.
* Failure to share critical information in a timely manner

1. Resource Constraints

* Insufficient funding or budget overruns.
* Shortage of skilled personnel or key resources.
* Overloading team members with unrealistic workloads

1. Scope Creep

* Uncontrolled expansion of project scope without adjusting resources, time, orbudget.
* Lack of a formal process to manage changes in scope

1. Poor Risk Management

* Failure to identify, assess, or mitigate risks.
* Ignoring potential threats to the project’s success.
* Underestimating the impact of external factors like market changes or economic conditions.

1. Unrealistic Expectations

* Overpromising to stakeholders or customers.
* Setting unachievable goals without proper analysis.

1. Lack of Stakeholder Engagement

* Failure to involve key stakeholders in decision-making.
* Misalignment between stakeholder expectations and project deliverables.
* Resistance to change from stakeholders or team members

1. Inadequate Quality Management

* Insufficient testing or quality assurance.
* Deliverables that do not meet the agreed-upon standards.
* Neglecting customer or end-user feedback.

1. Technology Issues

* Reliance on outdated or incompatible technologies.
* Technical failures or bugs that disrupt progress.
* Underestimating the complexity of technical requirements.

1. Organizational Challenges

* Lack of alignment between project goals and organizational strategy.
* Internal politics or power struggles affect decision-making.
* Resistance to new processes or systems within the organization

1. External Factors

* Changes in regulatory requirements or compliance issues.
* Economic downturns, market shifts, or supply chain disruptions.
* Natural disasters or other unforeseen events.

1. Failure to Monitor and Control

* Inadequate tracking of project progress against the plan.
* Ineffective use of tools and metrics for project management.

1. Cultural and Team Issues

* Lack of collaboration or teamwork.
* Conflicts within the team that are not resolved effectively.
* Cultural differences lead to miscommunication or misunderstandings.

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

Business Analysts (BAs) play a critical role in bridging the gap between stakeholders and technical teams. However, they often face challenges that can impact project success. Here is a list of common challenges faced by BAs in projects:

1. Unclear or Evolving Requirements -

Stakeholders may not fully understand or articulate their need, Frequent changes in requirements disrupt planning and development, Lack of clarity in requirements documentation.

1. Managing Stakeholders –

Identifying and engaging all relevant stakeholders, handling conflicting stakeholder priorities or expectations, gaining stakeholder buy-in for decisions or deliverables.

1. Communication Barriers –

Bridging the gap between technical teams and non-technical stakeholders, Overcoming misunderstandings due to jargon or differing perspectives, Lack of timely feedback from stakeholders or team members.

1. Time Constraints - Tight deadlines to gather, document, and validate requirements, Pressure to complete tasks quickly, compromising quality, balancing multiple projects or competing priorities.
2. Lack of Stakeholder Involvement –

Stakeholders not dedicating enough time to the project, Delayed decisions or inputs from stakeholders, Resistance from stakeholders to change existing processes or systems.

1. Handling Scope Creep –

Managing unplanned changes or additions to project scope, Lack of proper change control mechanisms, Difficulty maintaining focus on core project objectives.

1. Insufficient Domain Knowledge -

Difficulty understanding complex or unfamiliar business domains, Limited access to Subject Matter Experts (SMEs) for clarification, need to quickly learn and adapt to industry-specific terminologies or practices.

1. Conflict Resolution –

Mediating conflicts between stakeholders with differing priorities, resolving disputes within the project team, such as developers vs. Testers, Balancing the interests of multiple departments or business units.

1. Technical Constraints -

Aligning business needs with existing technology limitations, understanding technical jargon and constraints imposed by development teams, Ensuring compatibility with legacy systems or third-party integrations.

1. Unrealistic Expectations –

Stakeholders expect quick solutions for complex problems, Pressure to deliver beyond the project’s scope or capabilities, Unrealistic assumptions about technology capabilities or timelines.

1. **Write about Document Naming Standards**

File Naming Standards are used to save the file with particular name or format. This is important in sharing and keeping track of data files.

The following are the best standards in Naming Convention –

1. It should be Named Consistently.
2. File names should be short (<25 characters)
3. Avoid special characters or spaces in a file name.
4. Use Capital and Underscores instead of spaces or slashes.
5. Use date format as ISO 8601: YYMMDD
6. Include a version number.
7. Write down naming convention.

We must consider following naming conventions –

* Date of Creation
* Short Description
* Work
* Location
* Project name or number
* Sample
* Analysis
* Version Number

For example – We have a project with ID “PROJ456BANK” and we are working with Requirement Specification Document then –

Project ID - PROJ456BANK

Document Type – REQ

Version – 1.0

Date – 2025-02-14

Then the naming convention of the document will be “PROJ456BANK-REQ-1.0-2025-02-14”.

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

|  |  |
| --- | --- |
| **DO's** | **DONT's** |
| Consult an SME for clarifications in requirements | Never say ‘NO’ to clients. |
| Go to the client with plain mind with no assumptions, listen carefully and completely until the client is done and then you can ask queries. | There is no word as “By Default”. |
| Try to extract maximum leads to the solution from the client himself. | Never imagine anything in terms of GUI. |
| Concentrate on important requirements. | Don’t interrupt the client when he is giving you the problem. |
| Question the existence of existence or question everything. | Never try to give solutions to the client straight away with your previous experience & assumptions. |

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

Packages – Packages are the collection of components which are not reusable in nature.

Example: Application development companies work on Packages.

Sub Systems – Sub Systems are the collection of components which are reusable in nature.

Example: Product development companies work on Sub Systems.

Difference between Packages and Sub Systems are described below –

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Package** | **Sub-system** |
| **Definition** | A collection of related functionalities or components bundled together for a specific purpose. | A self-contained unit within a larger system, consisting of multiple components or packages. |
| **Scope** | Focused on a specific, narrow functionality. | It covers broader and more integrated business functions. |
| **Complexity** | Generally simpler and less complex. | Larger and more complex due to integration of multiple components. |
| **Purpose** | Provides specific features or services. | Represents a significant part of a larger system's functionality. |
| **Size** | Smaller in size and scope. | Larger, often consisting of several packages or modules. |
| **Components** | May consist of a single module or component. | Comprised of multiple components, potentially including packages. |
| **Integration** | Typically needs to be integrated with other packages or systems. | Operates as part of a larger system, often with complex interdependencies |
| **Example** | Payment processing module, Reporting package | Inventory Management System, CRM System |

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

Camel casing is a style of writing where multiple words are combined without spaces, and each word after the first starts with an upper case letter. The first word is written in lowercase. It’s called "camel casing" because the capital letters in the middle of the words resemble the humps of a camel.

Example – firstName, userProfile, calculateTotalAmount, isValidInput

Where camel casing is used –

* Programming (Variable/Function Names): Camel casing is commonly used in programming for naming variables, functions, methods, and objects, particularly in languages like Java, JavaScript, and C#. It helps improvere adability while adhering to the language's naming conventions.

Example: totalPrice, getUserInfo(), isValidUser().

* API Endpoints: Camel casing is often used in the naming of API endpoints or URLs to maintain consistency and improve readability.

Example: /getUserDetails, /createNewProduct.

* Class/Method Naming (in certain languages like Java): In some programming anguages, camel casing is used for method names and class names, though class names may also use Pascal casing (a variation where the first letter is also capitalized).

Example: calculateTax(), setUserPreferences()

* CSS/HTML Class Names (in some conventions): While hyphenated names (kebab case) are more common in CSS, some developers use camel casing for class names, especially in JavaScript frameworks like React.

Example: myComponent, navBarItem

* File and Folder Naming: Some development environments and frameworks use camel casing for file and folder names.

Example: userProfile.js, createUserFolder

Benefits of using Camel Casing-

* Readability – Helps in distinguishing words easily when spaces are not allowed.
* Consistency – Adhering to a consistent naming convention across a code base improves maintainability.
* Convention – Many programming languages and frameworks have adopted camel casing as a standard, making it a widely accepted practice.

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

A Development Server is a computer or environment where developers build, test, and deploy software applications or services. It is a dedicated machine, or instance used for development purposes, often running on a local or remote server. A development server allows developers to code and test their applications before moving them to production.

A Business Analyst can have below types of access based on the needs –

1. Read Only Access – BA may be granted with 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 behavior of the application.
2. Limited Access – Depending upon the project needs, the BA’s will be granted limited access to the specific modules in the application.
3. Limited Configuration Access – It means BA has the authority to make changes in certain areas of application where they have access.
4. **What is Data Mapping**

Data Mapping is the process of establishing relationships between two distinct data models. It is a technique used to link data from one system, database, or format to another, ensuring that the data is correctly interpreted and transferred. This process is crucial when integrating different systems, migrating data, or transforming data between formats for analysis or reporting.

Types of Data Mapping –

* One-to-One Mapping:

Each field in the source corresponds to a single field in the target.

Example: A field "FirstName" in the source database maps directly to the "First\_Name" field in the target database.

* One-to-Many Mapping:

One field in the source maps to multiple fields in the target.

Example: A "Full Address" field in the source could map to multiple fields such as "Street", "City", "State", and "Zip Code" in the target.

* Many-to-One Mapping:

Multiple fields in the source are combined or consolidated into a single field in the target. Example: First name and last name fields from the source system could be combined into a single "Full Name" field in the target system.

* Many-to-Many Mapping:

Multiple fields in the source are mapped to multiple fields in the target. This is often used when integrating systems with complex data relationships.

Example: Multiple product categories in the source system could map to various sub-categories in the target system.

Uses of Data Mapping –

* Data Integration:

When combining data from different systems, applications, or databases, data mapping ensures that the data from multiple sources is merged into a single, coherent dataset.

* Data Migration:

When moving data from one system to another (e.g. Upgrading databases, moving to cloud-based platforms), data mapping ensures the data fits the new system’s structure.

* ETL (Extract, Transform, Load) Processes:

In ETL, data is extracted from various sources, transformed (via mapping), and loaded into a data warehouse or reporting system.

* API Data Transfers: Data mapping is used when data is transferred between different APIs, ensuring that the formats and structures between sending and receiving systems align.
* Reporting and Analytics:

When preparing data for reporting or analysis, data mapping helps align the source data with the required formats for dashboards, visualizations, or business intelligence tools.

Benefits of Data Mapping –

* Improved Data Accuracy: Ensures that data is consistently transferred and interpreted correctly between systems.
* Efficient Data Integration: Facilitates the seamless combination of data from different sources, making it easier to generate reports and insights.
* Time and Cost Savings: Automates the process of transforming and mapping data, reducing manual work and errors.
* Data Quality Assurance: Identifies discrepancies and inconsistencies in data early in the process, improving the quality of the data being transferred.

1. **What is API. Explain how you would use API integration in the case of your application**

Application Programming Interface or API - Application Programming Interface (API) is a set of rules, protocols, and tools that allow different software applications to communicate and interact with each other. APIs define how different software components should interact, enabling one system or service to access features or data from another without needing to understand the internal workings of the other system.

Key components of API are described below –

1. API Endpoint:

A specific URL or URI (Uniform Resource Identifier) that represents a specific function or resource available via the API.

Example: https://api.example.com/users (Endpoint for fetching user data)

1. HTTP Methods (Verbs):

These methods define the type of operation to be performed on the resource:

GET: Retrieve data from the API

POST: Send data to the API, usually to create a new resource

PUT: Update an existing resource

DELETE: Remove a resource.

1. Request Headers:

Metadata sent along with the request, such as authentication tokens, content types, or session IDs.

1. Request Body:

Data sent along with the request, usually in JSON or XML format, contains the necessary information for the API to process.

1. Response Body:

The data returned by the API after processing the request, typically in a structured format like JSON or XML.

1. Authentication and Authorization: Many APIs require security mechanisms to verify the identity of the user or system making the request (e.g., via API keys, OAuth, or JWT tokens).

APIs are used in Third-Party Integrations, Mobile and Web Applications, Cloud Services, IoT (Internet of Things), Automation and Workflow.

Benefits of APIs –

1. Efficiency:

APIs allow businesses and developers to reuse existing software, components, or services, speeding up development time.

1. Scalability:

APIs enable systems to scale by allowing new services or components to be added easily without disrupting existing systems.

1. Integration:

APIs allow for the seamless integration of external services, platforms, or data sources, enabling systems to communicate and share information across different environments.

1. Security:

APIs can offer controlled access to services or data, enabling security features like authentication, rate limiting, and encryption to ensure safe data transmission.

1. Modularity:

APIs promote modular design by allowing applications to bebroken down into smaller, independent services, which can be updated or replaced without affecting the whole system.

For the above scenario, we can follow below procedure –

1. Establish API communication - set up API communication between your application and other applications to exchange data.
2. Do Data formatting- while sending the data from one application to another, convert the date format from dd-mm-yyyy to mm-dd-yyyy.
3. While receiving the data from another application, parse the data and extract the date, month and year and re-arrange them accordingly.
4. Perform Data Validation and ensure that the converted date remains in a valid format.