Project 3 Part 1

Question 1:

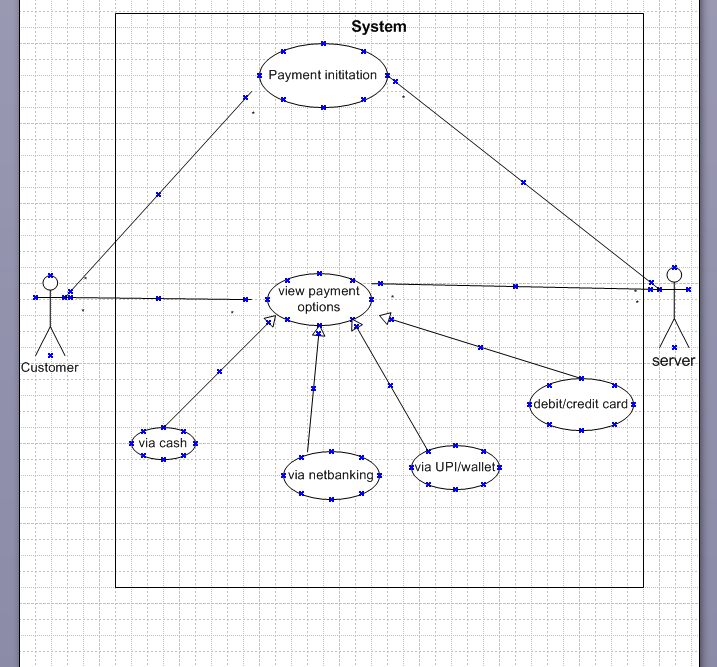
A use case diagram is a visual representation that shows how users (actors) interact with a system to achieve specific goals (use cases). It highlights the main functions of the system and the roles of the users, making it easier to understand the system's purpose.

Essential use case- An **essential use case** is a basic description of what a user wants to do and how the system should respond, without any technical or design details. It focuses on the main goal and interaction

Support use case- A **support use case** is a secondary or auxiliary use case that helps or enhances the functionality of essential use case.

A **use case relationship** defines how different use cases interact or relate to each other within a system. Common relationships include:

1. Include- One use case includes the behavior of another use case to achieve its goal. For e.g.-"Make Payment" includes "Validate Payment Details."
2. Extend-A use case extends another when it adds optional or conditional behavior. For e.g.- "Track Order" extends "Place Order" when the user chooses to check their order status.
3. Generalization- A child use case inherits behaviors from a parent use case, allowing specialization. For e.g.- "Online Payment" and "Cash on Delivery" are specialized use cases of "Make Payment."
4. Dependency-A use case depends on another use case to execute successfully. For e.g.- "Login" must occur before "Place Order."



Question 2:

Boundary Class-

A **boundary class** in software design represents the interaction between the system and external entities (like users or other systems). It typically handles input/output and acts as an interface.

**Simple Example:**

Consider an online shopping system:

**Boundary Class:** A class like Login Page or ShoppingCartUI that handles user inputs and displays outputs.

The following are the feature of the boundary class

1. This class is easier to be changed than the entity and control class
2. The attribute of this class and screen layout are defined at the basic design
3. In a class diagram, there are cases that is shown by the following icon

Controller class-

A **controller class** is responsible for managing the flow of data and handling the logic between the user interface (boundary class) and the system's data (entity classes). It acts as a bridge between what the user interacts with and how the system processes information.

Example:

In an online shopping system: It handles the logic of checking if a username and password are correct.

1. This class has a few attribute
2. This class is used to achieve use cases in the use case diagram
3. In a class diagram, there are cases that is shown by the following icon

Entity class-

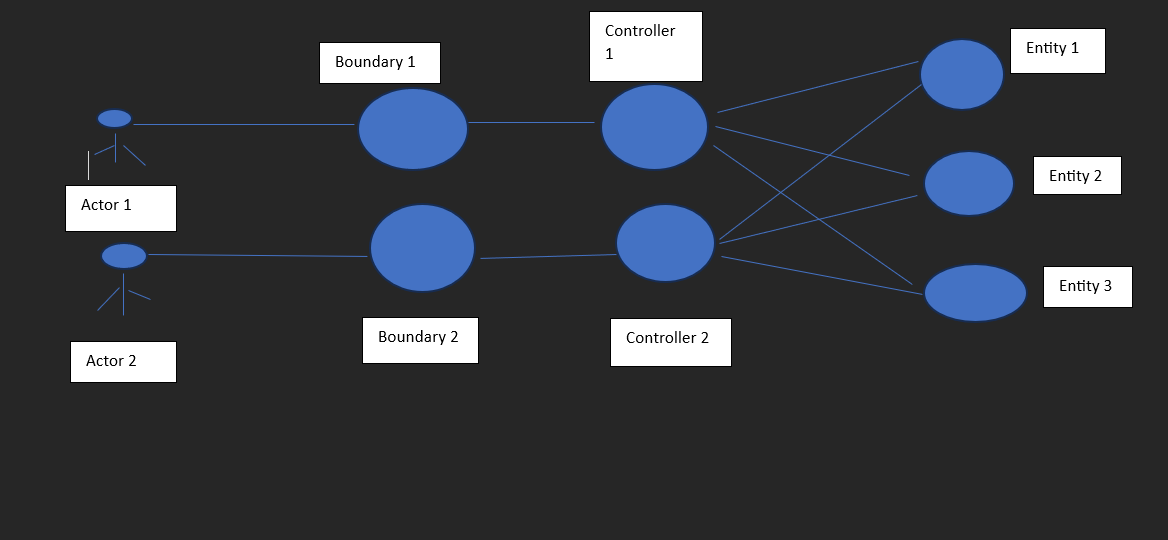
An **entity class** represents the core data and business logic of a system. It usually corresponds to real-world concepts or objects, like a user, product or order. These classes store data and define how that data can be manipulated.

For example:

It holds user information like name, email, and password.

1. The module cohesion of this class is high and is not easy to be changed
2. In a class diagram, there are cases that is shown by the following icon

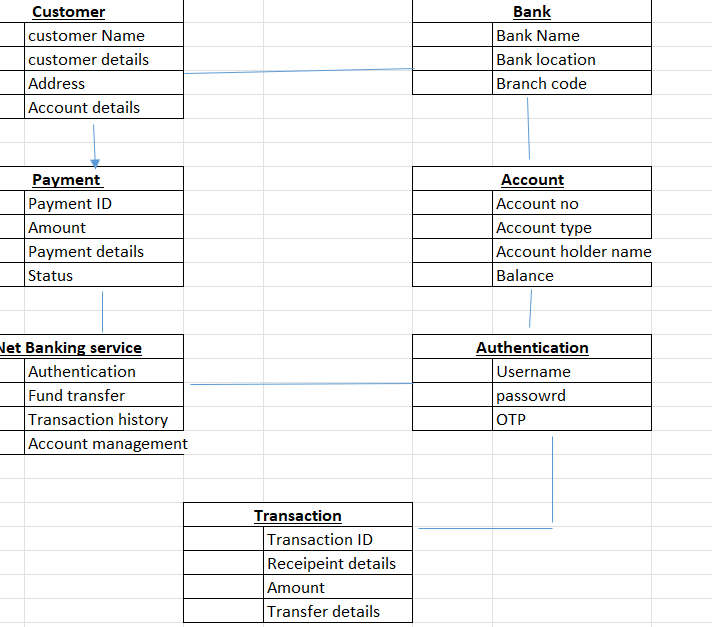
Question 3:



Question 4:

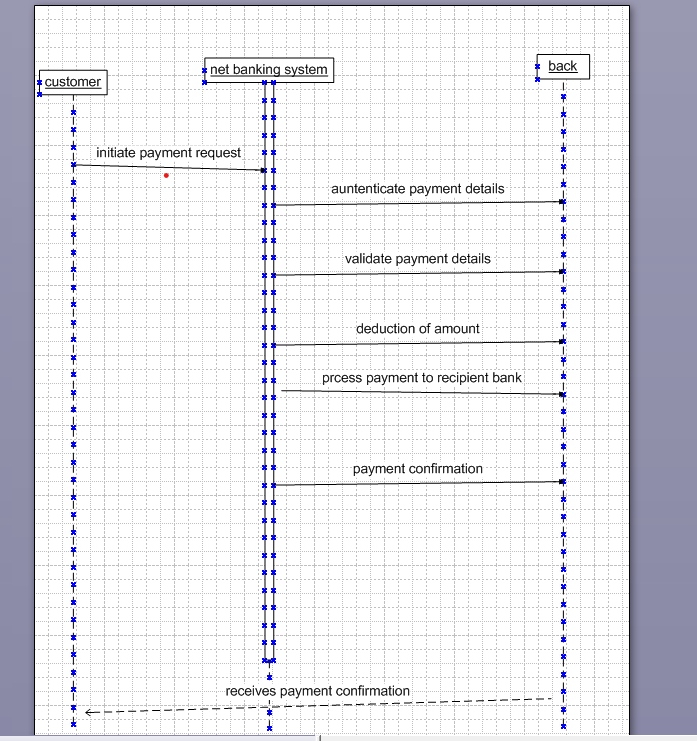
domain model is a visual representation of the key concepts, objects, or entities in a specific area (domain) of a system and how they relate to each other. It helps explain the real-world things the system deals with, like customers, products, or orders, in a simple and structured way.

A domain model is a conceptual representation that defines the structure , relationships, and behaviors of entities within a specific problem domain



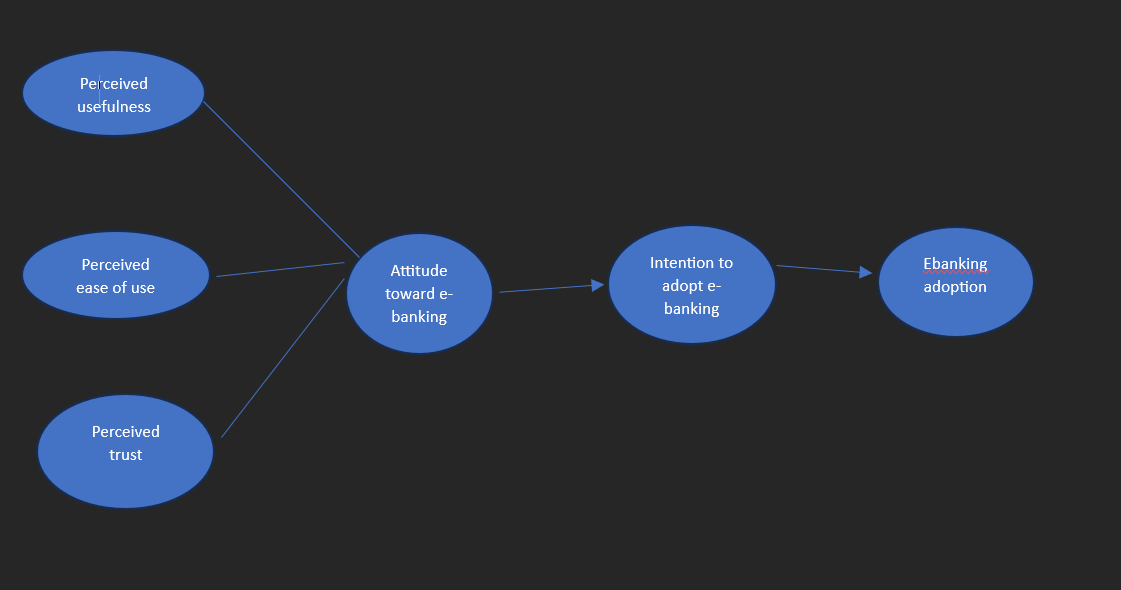
Question 5:

A sequence diagram is a visual chart that shows how objects in a system interact with each other over time. It displays the sequence of messages or actions between them to accomplish a specific task, helping to understand the flow of communication step by step



Question 6:

A **conceptual model** is a high-level representation of how a system or process works. It focuses on key concepts, their relationships, and the overall structure, without going into technical or implementation details. It's used to understand, communicate, or plan a system.



Question 7:

**MVC architecture** (Model-View-Controller) is a design pattern that separates an application into three interconnected components. This structure makes the application easier to manage, scale (can handle growth or increased demand), and test

1. Model- it represents the business layer of application. It is an object to carry the data that can also contain the logic to update controller if data is changed
2. View- it represents the presentation layer of application. It is used to visualize the data that the model contains
3. Controller- it works on both the model and view. It is used to manage the flow of application i.e. data flow in the model object and to update the view whenever data is changed

To identify classes from use case diagram, we apply MVC rules on each use case to derive classes.

1. Model: The model class knows about all the data that need to be displayed. it is a model who is aware about all the operations that can be applied to transform that class. It only represents the data of an application. The model represents enterprise data and the business rules that govern access to and updates of this data. All model classes are represented as entity class.
2. View: The view represents the presentation of the application. The view class refers to the model. it uses the query methods of the model to obtain contents and renders it. The view is not dependent on the application logic. It remains same if there is any modification in the business logic. view class is the data required by the query. view class is represented as boundary class or form class. Actor speaks to system and vice versa through boundary. Authenticating information between boundary and entity class.
3. Controller- Whenever the user sends a request for something then it always go through the controller. The controller is responsible for intercepting the requests from view and passes it to the model for the appropriate action. After the appropriate action has been taken place, the controller is responsible for directing the appropriate view to the user. In GUIs the views and controllers often work very closely together. Controller class is working based on the user command. Understands the command/ request given by user through boundary/form class.

MVC architecture rules:

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 classes.
3. Combination of three actors and a use case results in three boundary class and so on
4. Use case will result in a controller class
5. Each actor will result in one entity class

guidelines to place classes in 3-tier architecture

1. Place all entity classes in 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 there with any of the remaining boundary classes, place them in business logic layer else place them in application layer.

Question 8:

A waterfall model is very old and traditional method in IT industries. It is a progressive implementation of the projects which is divided into different phases of SDLC

The business analyst will verify the product is delivered as per the requirements and it is meeting the business need. Maintenance-once the implementation is done the team has to give support by installing patches, handling change requests etc.

Stages in waterfall model.

|  |  |  |
| --- | --- | --- |
| Stage | Activities | Artifacts& resources |
| Pre Project | **Requirement Gathering**: Collecting detailed requirements from stakeholders. | Artifacts- BRD, feasibility reports, risk assessment report, initial project plan |
|  | **Feasibility Analysis**: Assessing technical, financial, and operational feasibility. | Resources-Project Manager, SME, Team members, financial resource and Market data |
|  | **Project Planning**: Creating a roadmap, timelines, and resource allocation. |  |
|  | **Approval**: Gaining stakeholder and management sign-off to proceed. |  |
| Planning | **Project Scope Definition**: Clearly defining what is included and excluded from the project. | Artifacts-Project Plan, Gantt chart, risk management plan, resource allocation plan |
|  | **Work Breakdown Structure (WBS)**: Breaking the project into smaller, manageable tasks. | Resources-Project Manager, SME, Team members, financial resource, |
|  | **Timeline Creation**: Establishing deadlines and milestones for each phase. |  |
|  | **Resource Allocation**: Identifying the required resources (human, hardware, software). |  |
|  | **Budget Planning**: Estimating costs and creating a budget for the entire project. |  |
|  | **Risk Management Plan**: Identifying potential risks and defining mitigation strategies. |  |
| Project initiation | **Defining the Project**: Identifying the purpose, objectives, and expected outcomes of the project. | Artifacts-Project Charter, Stakeholder Register, Feasibility Report, Business Case |
|  | **Feasibility Study**: Ensuring the project is viable technically, financially, and operationally. | Resources-Project Sponsor, Project Manager, Business Analysts, Subject Matter Experts (SMEs) |
|  | **Stakeholder Identification**: Identifying all stakeholders and their roles and responsibilities. | Documentation Tools |
|  | **Approval to Proceed**: Getting formal approval from stakeholders or project sponsors to begin the project. |  |
|  | **High-level Risk Identification**: Preliminary identification of major risks that could impact the project. |  |
| Requirement gathering | **Stakeholder Interviews**: Engaging with stakeholders to gather detailed needs and expectations. | Artifacts- -BRD, use case documents, RTM, Stakeholder Analysis Document: |
|  | **Documenting Requirements**: Recording all functional and non-functional requirements clearly. | resources-Business Analysts, Project Manager, Subject Matter Experts (SMEs)Stakeholders |
|  | **Analyzing Requirements**: Ensuring all gathered requirements are feasible, complete, and clear. |  |
|  | **Requirement Validation**: Reviewing requirements with stakeholders for approval and adjustments. |  |
|  | **Creating Requirement Traceability Matrix**: Ensuring that all requirements can be traced through each phase of the project |  |
| Requirement analysis | **Requirement Review**: Analyzing the requirements to ensure they align with business goals and are feasible. | Artifacts-Requirements Specification Document, Use Case Diagrams/Models, Requirements Traceability Matrix (RTM)Prioritized Requirements, Gap Analysis Document: |
|  | **Clarifying Ambiguities**: Addressing any unclear or conflicting requirements by consulting stakeholders. |  |
|  | **Prioritizing Requirements**: Determining which requirements are most critical to the project's success. | resources-Business Analysts, Project Manager, Subject Matter Experts (SMEs)Stakeholders |
|  | **Creating Detailed Specifications**: Defining functional and non-functional requirements in detail. |  |
|  | **Creating a Requirements Traceability Matrix (RTM)**: Mapping requirements to design, development, and testing phases to ensure they are met. |  |
|  | **Feasibility Assessment**: Verifying the feasibility of implementing the requirements within technical and budget constraints. |  |
| Design | **High-Level Design (System Design)**: Defining the system architecture, components, and interactions between modules. This is an overview of how the system will work. | Artifacts-System Design Document, Detailed Design Specifications, UI/UX Design, Interface Design Specifications |
|  | **Detailed Design (Component Design)**: Breaking down the high-level design into detailed specifications for each system component, including algorithms, data structures, and interfaces. | Resources- System Architects, Designers, Developers, Project Manager, Stakeholders, Design Tools, Documentation Tools |
|  | **Database Design**: Structuring the database schema, including tables, relationships, and data flow. |  |
|  | **Interface Design**: Designing user interfaces (UI) and system interfaces (APIs, communication protocols). |  |
|  | **Design Reviews**: Reviewing the designs with stakeholders to ensure they align with the requirements. |  |
| Development | **Coding**: Writing the actual code for each component, module, and system functionality. | Artifacts- code, code review reports |
|  | **Component Integration**: Integrating individual modules or components into a cohesive system. | Resources-Developers, Development Tools, Testers, project managers |
|  | **Unit Testing**: Testing individual components or modules to ensure they function correctly. |  |
|  | **Documentation**: Writing technical documentation to describe the system and code for future reference. |  |
|  | **Version Control**: Managing code changes through version control systems (e.g., Git). |  |
| Testing | **Test Planning**: Creating a test strategy, defining test cases, and identifying the scope of testing. | Artifacts-Test Cases, Bug/Defect Reports, Test Summary Report |
|  | **Test Case Development**: Writing detailed test cases based on the requirements and design documents. |  |
|  | **Unit Testing**: Verifying individual components or modules. | Resources-Testers, Test Environment, Development Team, Project Manager, End Users (for UAT) |
|  | **Integration Testing**: Testing the interaction between integrated components or systems. |  |
|  | **System Testing**: Conducting end-to-end testing of the entire system to ensure it meets the functional and non-functional requirements. |  |
|  | **User Acceptance Testing (UAT)**: Verifying the system with actual users to ensure it meets their needs and expectations. |  |
|  | **Defect Reporting and Resolution**: Identifying, documenting, and fixing defects or issues discovered during testing. |  |
| UAT | **Test Planning**: Defining the scope, criteria, and test scenarios for UAT based on business requirements. | Artifacts-UAT Test Plan, UAT Test Cases, UAT Test Scripts, Defect Reports, UAT Test Results, UAT Sign-off Document |
|  | **Test Case Preparation**: Creating test cases that reflect real-world scenarios and user workflows. | Resources-End Users, Testers (QA Team), Business Analysts, Developers, Documentation Tool |
|  | **UAT Execution**: Actual users or stakeholders perform the tests to validate that the system works as expected in a live-like environment. |  |
|  | **Defect Reporting**: Any issues or discrepancies discovered during UAT are logged, and the development team fixes them. |  |
|  | **Test Results Review**: Reviewing the outcomes of UAT, confirming that all business requirements have been met. |  |
|  | **Approval for Release**: Once the system passes UAT, the stakeholders provide formal approval to proceed with deployment. |  |

Question 9:

In the 1970s, researchers Kenneth Thomas and ralph Kilmann developed a model for conflict resolution known as this model.

This model describes the two core dimensions while choosing a mode of conduct in a situation of conflict assertiveness and cooperativeness.

Assertiveness is the extent to which you try to solve and resolve for your preferred outcomes. Think of this as the factor on the y axis of a graph. On the other hand, cooperativeness is the level to which you try to resolve the other party problems. This is the factor on the x axis of the graph.

Identify the conflict, discuss the details, agree with the root problem, check for every possible solution for the conflict, negotiate the solution to avoid further conflicts.

The five steps of conflict management are:

1. **Identify the Conflict**: Recognize and define the issue that is causing the conflict. Understand the root cause and the parties involved.
2. **Understand Different Perspectives**: Listen to all parties involved to understand their views, needs, and concerns. Encourage open communication to foster understanding.
3. **Find Common Ground**: Identify areas of agreement or shared goals to help reduce tension. This helps in focusing on mutual interests rather than differences.
4. **Develop Solutions**: Collaborate to come up with possible solutions that address the concerns of all parties. This could involve compromise, negotiation, or finding creative alternatives.
5. **Implement and Follow Up**: Once a solution is agreed upon, implement it and monitor the situation. Ensure that the conflict is resolved and take steps to prevent future issues.

5 models for handling conflicts.

1. Competing- Competing is a conflict resolution method where one party prioritizes winning the conflict over others' needs or concerns. It’s assertive and uncooperative, often used when quick decisions are needed, or the outcome is critical. However, it can strain relationships if overused.
2. Accommodating- Accommodating is a conflict resolution method where one party puts the other’s needs or concerns above their own. It’s cooperative but unassertive, often used to maintain harmony or preserve relationships. While it helps in the short term, overuse may lead to resentment or imbalance.
3. Avoiding- Avoiding is a conflict resolution method where a person neither pursues their own concerns nor confronts the conflict. They choose to ignore or postpone dealing with the issue. It’s useful for minor conflicts or when emotions need to cool down but can lead to unresolved problems if overused.
4. Collaborating- Collaborating is a conflict resolution method where both parties work together to find a win-win solution that satisfies everyone's needs. It’s assertive and cooperative, aiming for mutual benefit and long-term solutions. While it takes time and effort, it builds trust and strong relationships.
5. Compromising- Compromising is a conflict resolution method where both parties give up something to reach a mutually acceptable solution. It’s a middle-ground approach, balancing assertiveness and cooperation. While it resolves conflicts quickly, it may not fully satisfy either party’s needs

Question 10:

Reasons for project failure

1. Poor planning- Poor planning is a common reason for project failure because it leads to unclear goals, unrealistic timelines, inadequate resource allocation, and poor risk management. Without a solid plan, the project lacks direction, causing delays, budget overruns, and unmet expectations.
2. Inconsistently defined resources- Inconsistently defined resources can cause project failure because unclear roles, responsibilities, or insufficient resources lead to confusion, inefficiencies, and missed deadlines. Without proper resource planning, the team cannot work effectively to achieve project goals.
3. Unclear objective and requirements- Unclear objectives can lead to project failure because the team lacks a clear direction, making it difficult to prioritize tasks and measure progress. Without well-defined goals, there is a higher risk of scope creep, misalignment, and wasted effort, resulting in the project not meeting its intended outcomes.
4. Lack of detailed control- Lack of detailed control can cause project failure because it leads to poor tracking of progress, timelines, and budgets. Without proper oversight, issues may go unnoticed, resulting in delays, cost overruns, and quality problems. Effective control ensures the project stays on track and meets its goals.
5. Poor communication- Lack of communication is a major reason for project failure because it can cause misunderstandings, delays, and a lack of coordination among team members and stakeholders. Without regular updates and clear communication, issues may go unnoticed, and the project can veer off track, leading to unmet goals and missed deadlines.
6. Unrealistic expectations- Unrealistic expectations can lead to project failure because they set goals that are difficult or impossible to achieve within the given time, budget, or resources. This often results in frustration, burnout, and subpar outcomes, as the team struggles to meet expectations that were not properly aligned with the project's capabilities.
7. Inadequate risk management-Inadequate risk management is a significant risk factor that can lead to project failure by coming up of unforeseen issues, Lack of Preparedness, Resource Drain, increased Costs etc.
8. Lack of stakeholders engagement- Lack of stakeholder engagement can lead to project failure by causing misalignment between project objectives and stakeholder expectations. It can result in missed requirements, reduced support, and delays due to unaddressed concerns. Without active involvement, stakeholders may lose trust and confidence, leading to a lack of necessary resources or approvals. Effective engagement ensures clear communication and alignment throughout the project lifecycle.
9. Scope creep-scope creep occurs when uncontrolled changes or additions are made to the project scope, leading to delays, budget overruns, and resource strain. It diverts focus from original objectives, resulting in a project that fails to meet its initial goals.
10. Technical challenges-Technical challenges can lead to project failure when the technology or infrastructure used is inadequate, incompatible, or overly complex for the project’s needs. These issues can cause delays, cost overruns, and the inability to deliver the expected product or system.
11. Resource constraints-Resource constraints, such as insufficient budget, personnel, or equipment, can hinder project progress and quality. These limitations lead to delays, increased costs, and the inability to meet project goals.

Question 11:

A BA is responsible for multiple tasks at the same time. From handling the projects, maintain client relationships, interacting with stakeholders and managing project deadlines

Below are the challenges faced in projects for BA.

1. Managing stakeholder expectations is challenging for a Business Analyst as it requires balancing conflicting interests and aligning diverse stakeholder goals. Failure to communicate effectively can lead to misunderstandings, scope creep, and dissatisfaction
2. Scope creep is a challenge for Business Analysts as it involves managing changing requirements and ensuring that new demands align with project objectives. Uncontrolled changes can disrupt timelines, budgets, and resource allocation.
3. Changing business needs or requirements- Changing business needs or requirements are a challenge for a Business Analyst (BA) because they can cause scope creep, delays, and confusion. Frequent changes require the BA to continuously adjust and re-prioritize, which can lead to miscommunication, difficulty in delivering solutions on time, and a lack of alignment between stakeholders.
4. Time and resource constraints are a challenge for Business Analysts as they must prioritize tasks and manage expectations within limited budgets and schedules. These constraints can lead to compromised quality or missed opportunities to gather complete requirements.
5. Quality assurance and testing present a challenge for Business Analysts as they need to ensure that the requirements are correctly understood and implemented. Inadequate testing or unclear requirements can result in defects, impacting the system’s functionality and user satisfaction.
6. Documentation and knowledge management are challenges for Business Analysts as they must maintain accurate, up-to-date records of requirements, decisions, and changes. Inadequate documentation can lead to miscommunication, confusion, and difficulty in tracking project progress and decisions.
7. Technology constraints and complexity are challenges for Business Analysts as they must understand and work within the limitations of available tools and platforms. These factors can affect the feasibility of proposed solutions, requiring careful consideration of technical limitations and the ability to bridge the gap between business needs and technical capabilities.

Conflict with users- Conflict with users is a challenge for a Business Analyst (BA) because it can create tension, hinder collaboration, and delay decision-making. Disagreements may arise over requirements, priorities, or expectations. The BA must navigate these conflicts by facilitating open communication, managing differing viewpoints, and ensuring that the users' needs are understood and addressed in the solution

Question 12:

A document numbering standard is a systematic approach to assigning unique identifiers to various documents created and used throughout the development process.

Common Components of a Document Naming Standard:

1. **Project or Department Name**

Example: HR, IT, Marketing.

2. **Document Type**

Example: Invoice, Report, Proposal.

3. **Date**

Format: YYYY-MM-DD or DD-MM-YYYY.

4. **Version Number**

Example: v1.0, v2.1.

5. **Author or Team**

Initials or name of the creator.

6. **Description or Keywords**

Short description of the content.

Below standards should be used while naming the documents.

1. Keep file name short, but meaningful-
2. Avoid unnecessary repetition and redundancy in file names and folder names/file paths
3. The most preferred is title case- less preferable are no separation, underscores, dashes or spaces.
4. When including a number, use leading zeros to ensure files sort properly
5. Date format should be YYYYMMDD or YYMMDD so years of files sort in chronological order
6. When including a personal name in a file name give the family name first followed by the initials.
7. Avoid using common words such as draft or letter at the start of file names.
8. Order the elements in a file name in the most appropriate way to retrieve the record
9. The file names of records relating to recurring events should include the date and a description of the event, except where the inclusion of either of these elements would be incompatible with rule 2
10. The file names of correspondence should include the name of the correspondent an indication of the subject, the date of the correspondence and whether it is incoming or outgoing correspondence except where the inclusion of any of these elements would be incompatible with rule 2
11. The version number of a record should be indicated in its file name by the inclusion of V followed by the version number and where applicable draft
12. Avoid using special characters.

Question 13:

Do’s and Don’t’s for a BA:

|  |  |  |
| --- | --- | --- |
| Sr no | Dos | DONTs |
| 1 | Consult an SME for clarification in requirements | Never say no to clients |
| 2 | GO to the client with a plain mind with no assumptions. Listen carefully and completely, until the client id done and then you ask queries | There is no word 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 he is giving you the problem |
| 5 | Question the existence of existence/question everything | Never try to give solutions to the client straight away with your previous experience and assumptions |

Question 14:

Packages- collection of components which are not reusable in nature- application development companies work on packages

A package is a collection of related files, code, or tools bundled together to perform specific tasks in programming.

Sub-systems- collection of components which are reusable in nature- It is used by product company

A **subsystem** is a smaller, self-contained part of a larger system that performs a specific function or task. It works independently but also interacts with other subsystems to contribute to the overall functionality of the main system.

For eg:

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 storge functionality

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.

Question 15:

Camel Casing is a naming contention for writing file or object names using compounded or joined words with at least of those words beginning in a capital letter.

CamelCase is used in programming language to name different files and functions without violating the naming laws of the underlying language

CamelCase is also known as medical capitals and pascal case

It is used in many programming languages that doesn’t allow spaces in the file names camelCase enables the creation of names that are more unique and have more meaning for the developer.

It is a away to separate the words in a phrase by making the first letter of each word capitalized and not using spaces. It is used in **web URLs, programming and computer naming conventions**

Question 16:

A development server is a type of a server that is designed to facilitate the development and testing of programs, websites, software or applications for software programmers. It provided a run time environment as well as all hardware/software utilities that are essential to program debugging and development.

A development server is the core tier in a software development environment where software developers test code directly. It is comprised of the essential hardware, software and other components used to deploy and test the software under development including bulk storage development platform tools and utilities, network access and high-end processor. Upon testing completion, the application is moved either to a staging server or production/live server.

BA has the visualizing access in development server.

He has the access to all the functional servers and not to the technical servers.

Development server are of two types:

Technical and non-technical

Non-technical types ( BA has access to this type of server)

In some servers the BA has only view access however in some servers they also have other accesses.

* 1. Testing server- BA also has access to input data(not only view)
  2. Demo server- can also demonstrate
  3. Documentation Server-only view access
  4. Reporting Server- can generate and customize reports
  5. Feedback Server- Modify configurations like UI settings, populate mock data

Question 17:

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 management tasks

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

It has been a common business function for some time, but as the amount of data and source increase, the process of data mapping has become more complex, requiring automated tools to make it feasible for large data sets.

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 ware house

Data mapping is an essential part of ensuring that in the process of moving data from a source to a destination, data accuracy is maintained. Good data mapping ensures good data quality in the data warehouse.

Question 18:

An API, is application programming interface, is a software-to-software interface. API provide a secure and standardized way for applications to work with each other and to deliver the information or functionality requested without user intervention.

An API, or application programming interface is a set of defined rules that enables different applications to communicate with each other. It acts as an intermediary layer that processes data transfers between systems, letting companies open their application data and functionality to external third-party developers, business partners and internal departments within their companies.

To handle API integration when dealing with different date formats (e.g., your application uses **dd-mm-yyyy** while the external US application uses **mm-dd-yyyy**), you need to standardize the date format during the data exchange process. Here's how you can manage it:

1. **Receive Data from the US Application:**

Fetch the data from the API, which includes dates in **mm-dd-yyyy** format.

1. **Parse the Date:**

Use a date-parsing library or logic to interpret the **mm-dd-yyyy** format

1. **Convert Before Saving or Displaying:**

Store or display the date in **dd-mm-yyyy** format.

1. **Send Data Back to the US Application (if needed):**

Reverse the process: Convert the date from **dd-mm-yyyy** to **mm-dd-yyyy** before sending.