**Question 1 - Draw a Use Case Diagram - 4 Marks**A Use Case Diagram in UML is a visual representation that illustrates the interactions between users (actors) and a system. It captures the functional requirements of a system, showing how different users engage with various use cases, or specific functionalities, within the system.

**

Question 2 - Derive Boundary Classes, Controller classes, Entity Classes - 4 Marks**\*\* Boundary Class – Used to handle interactions between the system and external actors
Example – Payment Option Boundary, Card Payment Boundary

 Boundary Class

\*\* Controller Class – Acts as intermediaries between boundary and entity class.
Example – Payment initiated controller, Card Payment controller

 Controller Class

\*\* Entity Class – Represents the core data and business logic of the application.
Example – Customer Payment

 Entity Class

\*\* Note -- all actors would become entity class // all use cases can be boundary class // the actor or the use case relations where there is no third party involved becomes the controller class.

**Question 3 - Place these classes on a three tier Architecture - 4 Marks**

Application Layer- Customer Registration, Customer Login, Bank Server Login – All the boundary come to the application/user layer.

Business Logic Layer- Customer, Bank Server – All the controller class comes to the business logic layer.

Data Layer- Customer, Bank server, Cash, Card, Net banking – All entity class will come under data layer.

In this three-tier architecture, the application tier handles the user interface, the business logic layer manages the business logic and coordinates between the other tiers and the data tier handles data storage and retrieval.

**Question 4 - Explain Domain Model for Customer making payment through Net Banking- 4 Marks**A domain Model is a conceptual representation that defines the structure, relationships and behaviors of entities within a specific problem domain.

A domain model is similar to the entity-relationship model. The tables are connected to each other. In the below diagram, The customer table is connected to the bank table, which is why the customer is able to make payment. The customer table is also connected to the payment table because he should make the payment. Now the payment is done by net banking, so the payment table is connected to the net banking table. The account is in the bank, so the account table is connected to the bank table. The authentication table is connected to both the net banking table and the bank table because authentication is to be performed there. Also, the authentication table is connected to the transaction table, because authentication will be done during the transaction.



**Question 5 - Draw a sequence diagram for payment done by Customer Net Banking - 4 Marks**

A sequence diagram is a type of interaction diagram used in software engineering and systems design to illustrate how processes operate with one another and in what order. It is interacting between the actor and the object.

This diagram shows how the objects in the system interact and communicate with each other to achieve specific tasks. The developer will draw this. It is used to show the flow of messages, events, or actions between the objects of the system. This diagram helps to visualize the behavior of the system. This diagram shows the process in detail.



**Question 6 - Explain Conceptual Model for this Case - 4 Marks**A conceptual model is a high-level representation of a system that helps in understanding visualizing, and communicating the essential aspects of a domain.

It provides a clear and simplified view of the domain, making it easier to understand.

Key Elements are Entities- Customer, Product, and Order & Payment.
Attributes: Customer ID, name, email, phone number.
Relationships: For example, a customer places an order.

The conceptual model helps in understanding the key concepts, their relationships, and the overall structure of the net banking payment system. It serves as a foundation for designing the database schema, defining the application architecture, and implementing the necessary functionalities within the system.

**Question 7 - What is MVC architecture? Explain MVC rules to derive classes from use case diagram and guidelines to place classes in 3-tier architecture - 8 Marks**MVC is a design pattern where, the application is divided into 3 Logical parts- Model, View, and Controller.

Each of these parts will have a specific responsibility -

Model- Model represents the data and the business logic of the application. The model is responsible for multiple tasks like managing the application's data, performing data validation, implementing business rules, and handling data access operations. The model does not depend on how the data is presented or how the user interacts with the application.

The model class knows about all the data that is needed to be displayed. This layer corresponds to the data-related logic that the user works with. It represents the data that is being transferred between View and Controller. It can add or retrieve data from the database.

It responds to the controller’s request because the controller cannot interact with the database by itself. The model interacts with the database and gives the requested data. All the model classes are nothing but the entities. Model classes are represented as entity classes.

View-View is responsible for presenting the data to the user for handling the user interface. The View can be a web page, a desktop application window, or any other form of user interface. It receives input from the user and passes it to the Controller for processing.

It represents the presentation of the application. View refers to the model.

It takes the data from the Model and renders it in a way that is suitable for the user's display or interaction.

For rendering the data, it uses the query method. View does not depend upon application logic. View classes are represented as boundary classes.

Controller-Controller acts as an intermediary between the Model and the View. It receives input from the user (via the View), processes the input by invoking the appropriate methods in the Model, and then updates the View with the new data or state. The Controller handles user interactions, interprets user input, and translates it into instructions for the Model or the View. It coordinates the flow of data between the Model and the View, ensuring that they remain separate and independent of each other. Whenever the user requests anything, that request first goes to the controller. The controller works on the user's request.

Takes input from the user/ client. It interacts with the model and view. The controller class represents a use case. The controller acts as a mediator between the model and the database. A controller cannot directly get the data from the database. So controller interacts with the model.

The advantages of MVC architecture are as follows:

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

Rules to derive the classes from the use case diagram-

\*\* A combination of one actor and one use case results in one boundary class.
\*\* A combination of two actors and one use case results in two boundary classes.
\*\* A combination of three actors and one use case results in three boundary classes.
\*\* Use case will result in controller class.
\*\* Each actor will result in one entity class.

Ex: Model- Customer, Payment, Net banking, Card, Cash.

View- Login view, Payment option view, Net banking view, Bank selection view, Credentials view, Payment amount view, Payment confirmation view, logout view

Controller- Login controller, Payment option controller, Net banking controller, Bank selection controller, Credentials controller, Payment amount controller, Payment confirmation controller, logout controller.

Guidelines to place classes in 3-tier architecture

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

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

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

**Question 8 - Explain BA contributions in project (Waterfall Model – all Stages)- 8 Marks**The waterfall model is useful in situations where the project requirements are well- defined and the project goals are clear. It follows a sequential approach. In this model, each phase is completed entirely and then only moved to the next phase. The waterfall model relies on documentation to ensure that the project is well- defined and project team is working toward clear goals. Once that particular phase has been completed then we move onto the next phase, and we cannot make changes in the previous phase. This model is stable for the projects when the requirements are understood and it is clear.

Requirements Gathering- First, the stakeholders are identified. In this phase, all the requirements are gathered from the stakeholders.BA and PM participate in this phase. After completing this phase, BRD will be generated.

Artifacts-FSD functional specification doc

Design- After the requirements are cleared, the Design phase starts. This has a detailed design document that outlines the software architecture, user interface, and system components.HDD, ADD, and solution documents will be generated here.BA collaborates with designers, architects, and developers to translate requirements into system design.Artifacts- Design documents and UML diagrams

Development- The development phase includes 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, and checks if the development is going on the right track or not.BA also participates in scrum meeting.
Artifacts-code

Testing- In the testing phase, the software is tested as a whole 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.

Artifacts-test cases and test results

Deployment- Once the software has been tested and approved, it is deployed to the production environment.BA ensures that there is a smooth transition from the development phase to the production phase.

Artifacts- Implementation review document

Implementation- This is the final stage of the waterfall model. It involves running the code for the very first time in the production phase. The release manager handles this phase.BA will Update documentation and requirements specifications to reflect changes in the system over time.

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

Artifacts- User satisfaction review and change request review.

**Question 9 - What is conflict management? Explain using Thomas – Kilmann technique – 6 Marks**In the 1970s, researchers Kenneth Thomas and Ralph Kilmann developed a model for conflict resolution. It was called the Thomas-Kilmann model after them. Under this model, the term ‘conflict’ is described as the condition in which people’s concerns can’t be compared with others. If two or more people or groups care about things that are contradictory to each other, then the outcome is conflict. Conflict can occur due to various reasons, such as differences in goals, values, personalities, resources, or communication breakdowns.

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’s problems. This is the factor on the X-axis of the graph.

The Thomas-Kilmann approach is widely used to recognize the approaches of conflict management.

High Assertiveness and High Cooperativeness- Collaboration- means working together to find a solution.
High Assertiveness and Low Cooperativeness-Competition- means defensive that is standing for your individual beliefs and trying to win.
Low Assertiveness and High Cooperativeness– Accommodation- stakeholders will prioritize their needs over others.
Low Assertiveness and Low Cooperativeness– Avoidance- mean ignoring the conflict.

Assertiveness- the extent to which the person attempts to satisfy his own concerns. Cooperativeness- the extent to which the person attempts to satisfy the other persons concerns.

The Thomas-Kilmann Model is based on two dimensions: assertiveness and empathy. There are 5 conflict resolution strategies - Compete, Avoid, Accommodate, Collaborate and Compromise.

Compete-Competing, the first Thomas-Kilmann conflict mode is assertive and non- cooperative. It refers to addressing only one’s own concerns at the cost of the concerns of the other. Competing is defensive—it strictly means standing up for your individual beliefs and simply trying to win.

Avoid-In the Thomas-Kilmann model; avoiding is both unassertive and uncooperative. The individual wants to neither address their own problems nor the problems of others. This ultimately means that they do not want to engage in the conflict at all.

Accommodate- According to the Thomas-Kilmann model, the Accommodating mode is both accepting and cooperative. It is the opposite of competing. While accommodating, the individual in question neglects their own problems or beliefs to address the problems of the other party, self sacrifice is highlighted in this mode.

Collaborate- Collaborating, the most beneficial outcome in the Thomas-Kilmann conflict model is both assertive and cooperative. This mode is the complete opposite of avoiding. Collaborating includes a voluntary effort to work alongside the opposition to find a perfect solution that wholly addresses the collective problem. Collaborating involves deep-diving into an issue to locate the critical demands of the concerned individuals or parties.

Compromising-The last outcome in the Thomas-Kilmann conflict model falls on the average point on both the assertiveness and cooperativeness scales. The goal here is to find a mutually acceptable and robust solution that, in some ways, satisfies both the individuals. It comes midway between competing and accommodating. It addresses an issue more directly than avoiding but falls short of investigating it with as much depth and rigor as collaborating.

Each strategy has its benefits and disadvantages. Choose the appropriate one according to the situation.

**Question 10 - List down the reasons for project failure – 6 Marks**Reasons for project failure are:

I. Improper Requirement gathering- If the requirements of the project are not gathered properly, it will lead to project failure.
II. Lack of stakeholder involvement- A project can fail if the stakeholders are not participating in the process. The stakeholder's input and feedback play very important role in meeting the goals.
III. Less Communication- If there are communication issues between stakeholders and team members then this will lead to misunderstanding and delay in the project and the project will fail.
IV. Poor risk management- It can also lead to project failure if the team fails to identify the risks and do the risk mitigation, which can lead to unexpected challenges or delays in the project.
V. Improper planning- The project can fail if the planning is not done properly. The milestones and goals should be discussed. If there is no proper planning, then the team may face difficulties in addressing the issues or to track the progress.
VI. Insufficient resources- It can lead to project failure. The project may fail due to a lack of technology knowledge or lack of finances.
VII. Change of direction- Among the ways projects fail, a very common one is scope creep. This concept refers to changes requested when the project has already started which had not been planned before. This is very common when projects are not appropriately documented and defined beforehand.
VIII. Poorly assigned roles- When each team member receives their responsibilities clearly, they will know what, when, and how to perform their activities without someone needing to constantly ask for them.
IX. Unrealistic due dates- Planning unrealistic due dates for complex tasks for short due dates is definitely one of the causes for project failure. It is vitally important to carefully consider how long each project phase will take, in addition to extra time for unexpected events. This is the only way to develop a quality project.
X. Lack of monitoring- Providing a schedule to the team is not enough for a project to be successful. You should also make sure everything goes as planned. This means having frequent progress checks or meetings, as well as making adaptations, when necessary.

**Question 11 - List the Challenges faced in projects for BA – 6 Marks**\*\* Lack of training
\*\* Obtaining sign-off on the requirement
\*\* Change management
\*\* Coordination between developers and testers
\*\* Conducting meeting
\*\* Making sure the status report is effective
\*\* Driving clients for UAT completion
\*\* Making sure that the project is going on the right track and delivered as per the timelines without any issues
\*\* Gathering clear and unambiguous requirements can be challenging
\*\* Unable to understand what the stakeholder is trying to convey
\*\* Lack of domain knowledge
\*\* BA may face difficulties in understanding the requirements of the domain is not familiar to him
\*\* Poor communication between stakeholders and BA can affect the process of gathering the information
\*\* Changing business needs or requirements.

**Question 12 - Write about Document Naming Standards – 4 Marks**A document numbering standard is a systematic approach to assigning unique identifiers to various documents created and used throughout the development process.
EX-Suppose we have a project with ID “PROJ123”, we are working with a requirements specification document
Project ID- Proj123
Document type-REQ
Version-1.0
Date-2025-02-04
\*The document identifier could be Proj123-REQ-1.0-2025-02-04

\*Keep file name short and meaningful ex-/…/orientation/20250204capstone1.pdf
\*Avoid unnecessary repetition and redundancy in file names and folder names
Ex-/…/Project/20250204capstone1.pdf
\*Simple title case- no separate file name, underscore dashes, or spaces
\*When including a number, use leading zeros to ensure less sort properly, i.e., “001, 002…101” instead of “1, 2…101”
\*Date format should be YYYYMMDD (or YYMMDD) so years of files are sorted in chronological order
\*When including the personal name add surnames instead of first name. Ex-Banerjee2025.jpg
\*Order the elements in a file name in the most appropriate way to retrieve the record
\*Avoid using special characters.

**Question 13 - What are the Do’s and Don’ts of a Business analyst – 6 Marks**


**Question 14 - Write the difference between packages and sub-systems – 4 Marks**Package- Collection of components that are not reusable in nature. A package is a grouping and organizing element in which other elements reside, which must be uniquely named. In the UML, packages are used in a manner similar to the way directories and folders in an operating system group and organize files. For example, the project management system may be decomposed into a collection of classes organized into packages as follows -

Ex- Application development companies work on packages.Subsystem- Collection of components those are reusable in nature. Recall that a system is an organized collection of elements that may be recursively decomposed into smaller subsystems and eventually into non-decomposable primitive elements. For example, the project management system may be decomposed into the following: A user interface subsystem responsible for providing a user interface through which users may interact with the system business processing subsystem responsible for implementing business functionality. A data subsystem is responsible for implementing data storage functionality.Ex- Product development companies work on sub-systems**.**While a package simply groups’ elements, a subsystem groups elements that together provide services such that other elements may access only those services and none of the elements themselves. A subsystem is shown as a package marked with the Subsystem keyword**.**

**Question 15 - What is camel-casing and explain where it will be used- 6 Marks**Camel-casing refers to the naming convention of variables, parameters, or properties. Here, multiple words are combined together. In camel-casing, the starting letter of the first word starts with a small letter, and in other words first letter starts with capital letters**.**Camel case is used in a programming language to name different files and functions without violating the naming laws of the underlying language. Camel case is also known as medial capitals and Pascal case. The term Camel case is derived from its appearance, which can resemble a camel's back. It is used in many programming languages that don't allow spaces in file names. Camel case enables the creation of names that are more unique and have more meaning for the developers.

Ex- first Name, last Name

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

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

**Question 16 - Illustrate Development server and what are the accesses does business analyst has? -6 Marks**A development server refers to a dedicated environment or server that is used during the software development process. It provides a platform for developers and testers to build, test, and debug applications before they are deployed to a production environment.

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 underdevelopment, including bulk storage, development platform tools and utilities, network access, and a high-end processor. Upon testing completion, the application is moved either to a staging server or a production/live server.

The accesses a BA has are -

Read only- BAs may be granted 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.

Limited Access- Depending upon the project needs, the BAs will be granted limited access to the specific modules in the application. Limited Configuration Access means BA has the authority to make changes in certain areas of applications where they have access.

**Question 17 - What is Data Mapping 6 Marks**Data mapping is the process of connecting one source to another. It’s like creating a guide or map that shows how data in one place corresponds to data in another place. This is especially important when you are moving data between different systems or databases to ensure that the data stays consistent and accurate.

The main purpose of Data mapping is -

Data integration- Combining the data from different sources, ensures that the data is properly matched

Data Migration-While migrating the data from the legacy system(source) to the new system(destination), the data elements are mapped accurately into the new system. Required techniques are applied to convert the data into the format that is required by the new system

Data Transformation- Data transformation means converting the data from one format to another. In data mapping, data transformation plays a very important role which ensuring that the data of the legacy system(source) is mapped correctly to the data in the new system(destination)

**Question 18 - 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 who’s Date Format is mm-dd-yyyy 10 Marks**An API or application programming interface is a set of rules and tools that allows different software applications to communicate with each other. It defines the method and data formats that applications can use to request and exchange information. APIs provide a secure and standardized way for applications to work with each other and deliver the information or functionality requested without user intervention.

An API, or application programming interface, is a set of defined rules that enable 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.

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