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

**Q1. Draw a Use Case Diagram - 4 Marks**

**Answer-**

**Q2. Derive Boundary Classes, Controller classes, Entity Classes - 4 Marks**

**Answer-** As per rule of MVC we get-

Total 5 classes

1) for 2 Actor and a use case we get 2 BC

2) 1 Use case we get 1 CC

3) for 2 actors we get 2 EC



 **Q3. Place these classes on a three tier Architecture. - 4 Marks**

**Answer:** 

**Q4. Explain Domain Model for Customer making payment through Net Banking - 4 Marks**

**Answer:** A **domain model** is a conceptual representation of the key entities, their attributes, relationships, and business rules within a specific problem domain. It serves as a **blueprint** for understanding how real-world objects and their interactions are structured in a system. Domain Modelling is a method to describe and model entities and the relationships between them, which collectively represent the problem domain space.


****

**Process Explanation**:

* Customer-before the system -Screen-He will enter

1-Customer ID 2) Password 3) Select account and select/enter amount 4) Confirm amount and enter MPIN and send this to Payment controller class

* On the screen itself Customer making payment boundary class will be there, he will capture 1,2 through (a)
* Payment controller class will add 3) Account 4) Amount 5) Confirmation 6) Enter MPIN and sent 1,2,3,4,5,6 to Pay through Net banking boundary class through (b)
* Pay through Net Banking Boundary class once get transaction update through FCH, that transaction info is sent to Payment Controller class through (c).
* Payment Controller class will update its own DB i.e. NB entity class through (d) and Customer entity class through (e)
* Then Payment Controller Class will send transaction details to Customer making payment Boundary Class through(f)
* Customer making payment Boundary class will show transaction successful details the Customer.

**Q5. Draw a sequence diagram for payment done by Customer Net Banking - 4 Marks**

**Answer:** Sequence diagram is used primarily to show the interactions between classes in the sequential order in which those interactions occur. The major components of a sequence diagram are:

* **Lifelines:** Vertical lines representing the participating objects in the interaction.
* **Messages:** Arrows between lifelines indicating the communication between objects, including the message name and parameters.
* **Activation boxes:** Rectangles on a lifeline showing the duration of a method call.

Following is the sequence diagram for Customer doing payment by Net Banking:



Here Boundary Class CustomerMakingPayment BC will take care of this action and pass on the information to Controller Class Payment CC. Payment CC will perform following actions:

1. Initiate method shareDetails() through which ID, Password, Account Details and Amount is shared and response received.
2. It performs self validation.
3. Payment is processed and it sends paymentComfirmation() to CustomerMakingPayment BC.
4. Customer EC and NetBanking EC database is updated through updateDatabase() method.

**Q6. Explain Conceptual Model for this Case - 4 Marks**

**Answer-** A **Conceptual Model** is a high-level representation of how entities interact in a system. It focuses on defining the key entities, their attributes, and relationships without going into technical details. Entire information together we call it as conceptual model.

**Process Explanation**:

* Customer-before the system -Screen-He will enter

1-Customer ID 2) Password 3) Select account and select/enter amount 4) Confirm amount and enter MPIN and send this to Payment controller class

* On the screen itself Customer making payment boundary class will be there, he will capture 1,2 through (a)
* Payment controller class will add 3) Account 4) Amount 5) Confirmation 6) Enter MPIN and sent 1,2,3,4,5,6 to Pay through Net banking boundary class through (b)
* Pay through Net Banking Boundary class once get transaction update through FCH, that transaction info is sent to Payment Controller class through (c).
* Payment Controller class will update its own DB i.e. NB entity class through (d) and Customer entity class through (e)
* Then Payment Controller Class will send transaction details to Customer making payment Boundary Class through(f)
* Customer making payment Boundary class will show transaction successful details the Customer.

**Q7. 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**

**Answer-**The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. MVC architecture is used to identify classes from use case diagram. We apply MVC rules on each Use Case.

**1) Model:** The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data**. All model class are represented as entity class.**

**2) View:** View represents the presentation of the application. The view class refers to the model. It uses the query method of the model to obtain the contents and renders it. **View class is represented as boundary class**. Actor speaks to system through boundary class authenticating information between boundary class and entity class.

**3) Controller:** Whenever user send a request for something it always goes through the controller. The controller is responsible for intercepting the request from from view and pass it to the model for the appropriate action.

**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 classes.

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.**

* Place all Entity Classes in DB Layer
* Place Primary Actor associated Boundary Class in Application Layer
* Place Controller Class in Application Layer
* If governing Body influence or Reusability is there with any of the remaining Boundary Classes, place them in the Business Logic Layer else place them in Application Layer

**Q8. Explain BA contributions in project (Waterfall Model – all Stages) – 8 Marks**

**Answers**-Initially BA will carry out the Pre project activities in which he will be performing –

Enterprise Analysis, SWOT Analysis, GAP Analysis, Market Research, Feasibility Study, Root Cause Analysis, Decision Analysis, Strategy Analysis, Enterprise Architectural Frameworks, Project Scope and Business case writing and Risk analysis.

**1) Planning, Estimations & Assessment:**

* Understand Assumptions and Constraints along with Business Rules and Business Goals
* Plan Packages for Big Projects
* Understands the project plan from PM
* BA conducts stakeholders Analysis
* Plan BA approach strategy (Req. gathering techniques, communication, Requirement management, Documents
* to follow, Tools to use, Change Request Handling methodology) for this Project

**2) Requirements Gathering:**

* Stakeholders identify and document
* Client gives BRD or BA prepares BRD by interacting with Client Brainstorming, Document Analysis, Reverse engineering, Interviews, workshops, Focus Groups, Observation, Questionnaires.
* Prototyping can be used by BA to make the Client to give more specific requirements
* Sort the gathered Requirements
* Prioritize requirements (MoSCoW)
* Validate Requirements (FURPS)

**3) Requirements Analysis:**

* Draws UML Diagrams (Use case and Activity Diagrams)
* Prepares Functional Requirements from Business Requirements
* All Architects comes up with Technical Requirements (SSD)
* SRS will have Functional Requirements and Technical Requirements
* Takes Signoff on SRS from Client. SRS is the first legal binding Doc between the Business and the technical
* Team
* BA prepared RTM from SRS before Design phase starts.
* BA traces how requirements are dealt in each phase of development life cycle from Design till UAT

**4) Design:**

* From Use Case Diagram, Test Manager or BA will prepare Test Cases
* Communicates with Client on the design and Solution documents (updates Status to Client and make them
* understand how the solution would look like to prepare them to drive UAT)
* BA will initiate the preparation of End user manuals
* updates RTM
* From Use Case Diagram Solution Architect recommends Architecture of the IT solution
* DB Architect uses Persistence Classes (Entity Classes) and comes up with ER Diagrams or DB Schema.
* GUI Designer will look into Transient Classes (Boundary Classes) and designs all possible Screens for the IT Solution

**5) Coding:**

* BA organizes JAD Sessions
* BA clarifies queries of Technical Team during Coding
* Developers refer Diagrams and Transient (Controller Classes) of BA and code their unit
* Update End user manuals
* Update RTM
* Conducts regular Status meetings with technical team and the Client and tuning Client for participation in UAT.

**7) Testing:**

* BA Prepares Test Cases from Use Cases or assists Test Manager to do so
* BA performs high-level testing
* BA prepares Client for UAT
* Test Data is requested by BA from the Client
* Updates End User Manuals
* Updates RTM
* Take signoff from Client on Client Project Acceptance form

**8) Deployment and Implementation:**

* Forwards RTM to Client or the PM which should be attached to the Project Closure Document
* Coordinates to complete and share End User Manuals
* Plans and Organizes Training Sessions for End Users
* Prepares Lessons learned from this project (to take precautions for coming projects)

**Q9. What is conflict management? Explain using Thomas – Kilmann technique – 6 Marks**

**Answer-**This model is based on two dimensions of conflict management: assertiveness and Co-operative.

Based on these two dimensions, there are five conflict resolution strategies: Competing, Avoiding, Accommodating, Collaborating and Compromising.

**1. Competing**

At the top left end of the chart, we have high assertiveness and low cooperative. We use competing as a conflict resolution strategy whenever we resort to being aggressive. In addition, we also get uncooperative with the opponent.

The first strategy for dealing with conflict is competing.

**2. Avoiding**

At the bottom left end of the diagram, we see low assertiveness and low co-operative. That means you neither asset your position nor do you consider or emphasize the other party's point of view.

The second strategy for dealing with conflict involves avoiding it.

**3. Accommodating**

In the bottom right part of the diagram, we find low assertiveness and high empathy.

Accommodating is a third strategy for dealing with conflict.

**4. Collaboration**

Up above on the top right, we see high assertiveness and high empathy. When you collaborate, you take a moderate approach to solving problems. You attempt to balance power between yourself and another person. You also try to find common ground and work together towards achieving a shared goal.

It's an ideal choice if both parties are committed to reaching a mutual agreement. The goal here is to reach an agreement with each other.

**5. Compromising**

In the middle of the model is the fifth strategy, which is compromising. This is the middle of the assertiveness and empathy scales. If you compromise, you take the middle road between opposing views. This means agreeing to specific terms and giving up on certain other items. And since you're trying to resolve a disagreement, it's important that you show flexibility.

This strategy usually gives the feeling of a win-win scenario, where both parties feel that they gained something out of the conflict.



**Q10. List down the reasons for project failure - 6 Marks**

**Answer**-

1) Improper requirement gathering

2) continuous change in requirement.

3) lack of user involvement.

4) lack of executive support.

5) unrealistic expectations.

6) Improper planning.

**Q11. List the Challenges faced in projects for BA - 6 Marks**

**Answer-** Challenges faced in projects for BA are mentioned below:

1. Lack of training.
2. Obtaining sigh off on requirements.
3. Change management.
4. Co-ordination between developers and testers.
5. Conducting meetings.
6. Making sure status reporting is effective.
7. Driving clients for UAT
8. People management
9. Making sure the overall health of the project and project is delivered as per the timelines and without any issues.

**Q12. Write about Document Naming Standards - 4 Marks**

**Answer-** Document Naming Standards define a structured way to name files and documents to ensure **consistency, clarity, and easy retrieval** across teams or organizations. A well-defined naming convention improves collaboration, prevents confusion, and ensures version control. All documents should be named as per the following standards.

like [ProjectID][Document Type]V[X]D[ext]

Implementing a **document naming standard** improves file organization, enhances collaboration, and prevents errors in version control. By following a structured format with **project names, document types, version numbers, dates, and statuses**, teams can maintain a clear and efficient document management system.

**Q13. What are the Do’s and Don’ts of a Business analyst - 6 Marks**

**Answer-**

1. Never say NO to the client
2. There is no word called as By default
3. Never imagine anything in the terms of GUI
4. Question the existence of existence.
5. Consult an SME for clarifications in requirements.
6. Every problem of client is unique. No two problems of different client are same. May be the approach, technology, place of use, local laws may be varied to make them to be different.
7. Go to the client with plain mind with no assumptions.
8. Listen carefully and completely until client is done and then you may ask your queries.
9. Do not interrupt client when they are sharing their problems.
10. Maximum try to extract the leads to solution from the client itself.
11. Never try to give solutions to the client straight away with your previous experience and assumptions.
12. Try to concentrate on the important and truly required requirements.
13. Don’t be washed away by add on functionalities and don’t imagine solutions on screen basis.

**Q14. Write the difference between packages and sub-systems?- 4 Marks**

**Answer**-

* Packages are collections of components that are not reusable in nature. Application development companies work on packages. Organizes code for better modularity, reusability, and maintainability.
* Subsystems are collections of components that are reusable in nature. Product development companies work on sub-system. Represents an independent functional unit that interacts with other parts of the system.

**Q15. What is camel-casing and explain where it will be used- 6 Marks**

**Answer- Camel Casing** is a naming convention in which multiple words are combined into a single word without spaces, and each word (except the first) starts with an uppercase letter. Entire first word will be in lowercase and subsequent words first letter should be in Upper Case. There will be no gap in between words.

**Example:** getEmpId(); turnLeftAndSlowDown();

Camel casing is widely used in programming, especially in naming conventions for variables, functions, object properties, and more. Used by developers to increase understanding and readability ,widely used when creating certain diagrams like Sequence Diagram, writing methods,functions,naming variables.

**Q16. Illustrate Development server and what are the accesses does business analyst has? 6 Marks**

**Answer**-Server is a space where we store our data. From IT perspective we have 3 different servers-

* Production (owned by client)
* UAT (owned by client)
* Development (owned by IT company)

A development server is a type of server that is designed to facilitate the development and testing of programs, websites, software or applications for software programmers. As a BA we will have access to public document area and code and test areas we won’t be having any access to the Protected area and DB area.



Above is illustrated logical partitions of server where if PQ786 is a project allocated to an IT company and a piece of server is given to it. Here BA will have access to Public documents area, coding area and testing area.

**Q17. What is Data Mapping? 6 Marks**

**Answer-** **Data mapping** is the process of connecting and transforming data from one format, structure, or system to another. It ensures that data from different sources is correctly aligned so that it can be used efficiently in an application, database, or API integration. Data mapping ensures that information from different systems is accurately interpreted and processed.

**Data Mapping Process:**

* **Source Data Identification** – Identify the structure, format, and type of the incoming data.
* **Target Data Definition** – Define how the data should be structured in the receiving system.
* **Transformation & Conversion** – Apply necessary changes such as renaming fields, changing date formats, or converting data types.
* **Data Mapping Rules** – Define rules for handling missing, incorrect, or additional data.
* **Loading & Integration** – Save or send the transformed data to the destination system.

**Types of Data Mapping**

* **Manual Mapping** – Manually defining data transformations in code (as shown above).
* **Schema Mapping** – Matching database or API field names to a predefined schema.
* **Automated Mapping** – Using tools or middleware to map and transform data dynamically.

**Q18. What is API. Explain how you would use API integration in the case of your application Date format is dd-mm-yyyy and it is accepting some data from other application from US whose Date format is mm-dd-yyyy- 10 Marks**

**Answer-** An API (Application Programming Interface) is a set of rules and protocols that allows different software applications to communicate with each other. It acts as an intermediary that enables systems to exchange data or functionalities without exposing internal details.

**Steps for API Integration in case of accepting date in US format from other application where our app date format is dd-mm-yyyy:**

* **Establish API communication:** Setup API communication between your application and other application to exchange data.
* **Data Formatting**: When sending date data from your application to the other application convert the date from dd-mm-yyyy format to the mm-dd-yyyy format. This can be achieved by extracting the day month and year components and rearranging them according to target format.
* **Data Parsing:** When receiving date data from other application parse the mm-dd-yyyy formatted date into our application’s dd-mm-yyyy format. Again you will need to extract the day, month and year components and rearrange them accordingly.
* **Data Validation**: Perform data validation and ensure that the converted date remains valid after the format conversion. Check for edge cases such as invalid dates or date ranges that might be affected by the format conversion and handle them appropriately.

**Process:**

**1)**API Integration in our Application

API integration involves sending and receiving data between your application and the external system. In this case:

* Your app receives API responses with dates in mm-dd-yyyy format.
* You need to convert the date to dd-mm-yyyy before storing or displaying it.
* If your app sends date data back to the external system, you must convert it back to mm-dd-yyyy.

**2)**Handling Date Conversion in Angular (Frontend Approach)

Convert MM-DD-YYYY to DD-MM-YYYY and Apply the Conversion to API Data.

**3)**Handling Date Conversion in Backend (Node.js/Express Approach)

If the API response is processed in the backend before sending data to the frontend, you can use JavaScript to transform the date format.

Convert MM-DD-YYYY to DD-MM-YYYY in Node.js

**4)Converting Dates Before Sending Data to US Application**

If you need to **send data back** to the US-based application in mm-dd-yyyy format, reverse the conversion and apply it when making an API call.

By implementing **date format conversion** in either the **frontend (Angular)** or **backend (Node.js)**, our application can seamlessly integrate with the US-based system, ensuring accurate date handling.