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

**Q1. Draw a Use Case Diagram**

**Answer:**

****

**Q2. Derive Boundary Classes, Controller classes, Entity Classes.**

**Answer:**

|  |  |
| --- | --- |
| **Boundary Class (All use cases)**  [combination of 1 actor and a use case is one boundary class]  [combination of 2 actor and a use case is two boundary class]  [combination of 3 actor and a use case is three boundary class] and so on  These actors should be primary actors  Primary actors are the ones who directly interact with the system boundary and initiate use case  These classes handle the logic for different payment methods and control the payment process | Customer Registration  Customer Login  Bank Server Login  Customer Logout  Bank Server Logout |
| **Controller class(handles user input and process the data)**  Use case will be considered as the controller classes  These classes handle the logic for different payment methods and control the payment process | Registration Controller  Login Controller  Payment Controller  Credentials Controller  Net Banking Controller  Email Controller  Logout Controller |
| **Entity Class (all actors)**  Each actor will be considered as one entity  These classes represent the main data used within the system | Customer  Bank Server  Cash  Card  Net banking |

**Q3. Place these classes on a three tier Architecture.**

**Answer:**

|  |  |
| --- | --- |
| **Application Layer** | Customer Registration  Customer Login  Bank Server Login |
| **Business Logic Layer (**Primary actors associated with the Boundary class**)** | Customer  Bank Server |
| **Data Layer (**All the entity classes**)** | Customer  Bank Server  Cash  Card  Net banking |

**Application Layer:**

* The customer selects a payment method through the Customer Interface.
* Payment details are entered via the respective interface (e.g., Net Banking Interface).

**Business Logic Layer:**

* The Payment Controller receives the request and routes it to the appropriate payment controller.
* The payment controller (e.g., Net Banking Controller) validates the data and processes the transaction.

**Data Layer:**

* The controller interacts with Entity Classes like Bank Account or Wallet to check balances and update transactions.
* The Transaction entity is updated with the transaction status.

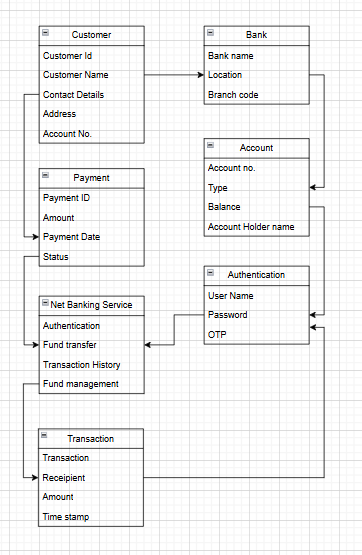
**Q4. Explain Domain Model for Customer making payment through Net Banking**

**Answer:** A domain model represents the entities involved, their attributes, and relationships within a business process. For a customer making a payment through Net Banking, the domain model includes entities such as Customer, Bank, Transaction, and Order.

The diagram represents the interaction between the entities like the customer table connected with the bank table, due to which the payment is possible. Similarly, the payment is done by net banking thus the table are connected. In this aspect, all the tables are connected with each other for the relationship between them for the successful working of the payment.

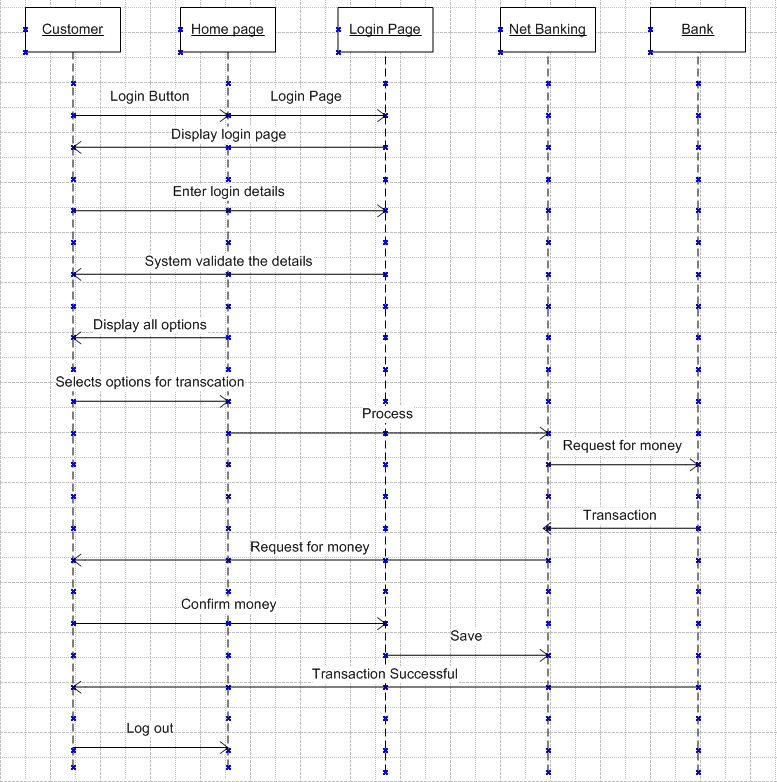
Difference between ER diagram and Domain model

|  |  |  |
| --- | --- | --- |
|  | **ER Model (Entity-Relationship Model)** | **Domain Model** |
| **Purpose** | Represents the database structure and relationships between data entities. | Represents the business concepts and relationships in the application. |
| **Focus** | Data storage, attributes, and relationships (database-centric). | Business process, system behavior, and interactions (business-centric). |
| **Entities** | Physical entities like tables, columns, and keys. | Conceptual entities like objects, actors, and processes. |
| **Attributes** | Detailed attributes and data types of each entity. | High-level attributes relevant to business logic. |
| **Relationships** | Explicit relationships using lines, cardinality (1:1, 1:M, M:N). | Represents associations between business objects (often implicit). |



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

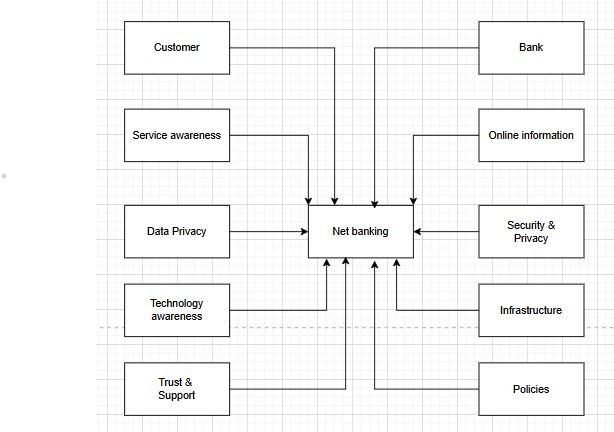
**Answer:**



**Q6. Explain Conceptual Model for this Case**

**Answer**: A Conceptual Model represents the high-level structure of a system, focusing on key concepts, their attributes, and relationships. The relationship between these entities can be described as follows:

1. **Customer:** This node represents the customer or users of net banking services
2. **Service awareness:** Customer should be aware of the available net banking services and their features
3. **Privacy of data:** The importance of this node is to protect the privacy and confidentiality of the customer data in the context of net banking
4. **Technology awareness**: The importance of this node is that customers should be aware and comfortable with the underlying technology used in net banking services
5. **Trust & Support:** This node indicates that the bank provide such good services that it helps to enhance the customer trust
6. **Bank**: This node represents a service provider responsible for offering net banking services
7. **Online information:** This aspect highlights the importance of providing accurate and up-to date online information about net banking services to customers
8. **Security & privacy:** The bank should adapt the security policies which will help the customers to keep their data related to their transactions secure and private
9. **Infrastructure:** This component suggests that the underlying technological infrastructure, including hardware and software systems, plays a important role in enabling net banking services
10. **Policies:** This node represents the various policies and regulations that govern the implementation and operation of net banking services, ensuring compliance and customer protection



**Q7. What is MVC architecture? Explain MVC rules to derive classes from use case diagram and guidelines to place classes in 3-tier architecture**

**Answer:** MVC isa design pattern used to separate an application into three interconnected components – Model, View and Controller. Each of these parts are discussed further:

**Model**:

It represents the data and the business logic of the application. It is responsible for multiple tasks like managing the application's data, performing data validation, implementing business rules, and handling data access operations.

A Model does not depend on how the data is presented or how the user interacts with the application. 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 the 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 give the requested data. Model classes are represented as entity class.

**View**:

The View is responsible for presenting the data to the user for handling the user interface. It 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. It takes the data from the Model and renders it in a way that is suitable for the user's display or interaction. View class are represented as boundary class.

**Controller:**

The 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 separated and independent of each other. Whenever the user requests for anything, that request first goes to the controller.

Controller works on the users request. Takes input from the user/ client. Controller cannot directly get the data from the database. Controller class represents as use case.

**Advantages of MVC-**

* MVC has the feature of scalability, which in turn helps the growth of application.
* The components are easy to maintain.
* A model can be used by multiple views that provide reusability of code.
* By using MVC, the application becomes more manageable.
* As all the three layers are different and independent, they are maintained separately.

**Consider the example of Online shopping application with the following use case**:

Model Classes-

Customer, Payment, Net Banking, Card, Cash

View Classes-

Login View, Payment Option View, Net Banking View, Bank Selection View, Credentials View, Payment Amount View, Payment Confirmation View, Logout View

Controller Classes-

Login Controller, Payment Option Controller, Net Banking Controller, Bank Selection Controller, Credentials Controller, Payment Amount Controller, Payment Confirmation Controller, Logout Controller

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

**Answer:** The Waterfall Model is a sequential development process with distinct phases, where each phase must be completed before the next begins. This model follows a sequential approach. This model is used only when the requirements are well-defined and clear. The Business Analyst (BA) plays a crucial role in each phase by bridging the gap between stakeholders and the development team. Below is a detailed explanation of BA contributions in each stage:

* Requirements gathering:
  + Identify all the Stakeholders
  + Gather all the necessary requirements
  + Document functional and non-functional requirements in a clear, structured format (e.g., Business Requirements Document - BRD)
* Requirement Analysis:
  + The requirements are analysed to understand the scope of project
  + Prepare the functional requirements documents called as FRS (functional requirement specs)
  + The technical team will develop the non-functional requirements called as SSD   
    (supplementary support document)
  + A BA will combine the FRS and SSD documents to form SRS document (Software require specifications
* 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.
  + 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.
* Development-
  + The Development phase includes the implementation.
  + It involves coding the software based on the design specifications.
  + Programmers or developer are involved in this phase.
  + Here BA acts as a mediator between the development team and the stakeholders.
  + BA clarifies the requirements, check if the development is going on right track or not.
  + BA also participates in scrum meetings.
* 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 helps the users to know the functionality of the system and also helps them to use the system.
* 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.
* Implementation-
  + This is the final stage of waterfall model. It involves running the code for the very first time in production phase.
  + BA will update documentation and requirements specifications to reflect changes in the system over time
* Maintenance-
  + Running the code for second time in the production phase is called maintenance.
  + This is done by support team.

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

**Answer:** Conflict management is the process of identifying, addressing, and resolving disagreements in a constructive manner to minimize negative impact while maximizing positive outcomes. In a business setting, conflicts can arise due to differences in goals, priorities, communication gaps, or resource constraints. A structured approach to managing conflicts ensures team collaboration, productivity, and a positive work environment identifying the conflicts efficiently, it will in turn be helpful to reduce negative impact and increase positive impact.

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

A diagram of conflict model

AI-generated content may be incorrect.

* **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- stakeholder will prioritize their needs over others.
* **Low Assertiveness and Low Cooperativeness** - Avoidance- means 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.

**Q10. List down the reasons for project failure**

**Answer-**

Project failure can occur due to multiple factors, ranging from poor planning to external risks. Below are some of the key reasons:

1. **Poor Requirements Gathering:** Unclear or incomplete requirements lead to scope creep and misalignment between stakeholders and the development team. Business Analysts must ensure thorough requirement elicitation and documentation.
2. **Scope Creep (Uncontrolled Changes in Scope):** Frequent addition of new features beyond the original project plan can lead to delays and budget overruns Lack of a proper Change Control Process contributes to scope creep.
3. **Lack of Proper Planning:** Inadequate project roadmap, resource allocation, and risk assessment lead to inefficiencies. Failing to define a clear Project Scope, Milestones, and Timelines causes confusion.
4. **Poor Communication Among Stakeholders:** Lack of regular updates and unclear roles/responsibilities result in misunderstandings and misaligned expectations. Agile projects require constant stakeholder engagement through meetings like Daily Stand-ups, Sprint Reviews, and Retrospectives.
5. **Insufficient Risk Management:** Ignoring technical, financial, and operational risks increases the chances of project delays and failures. Risk mitigation strategies should be incorporated in project planning.
6. **Inadequate Resource Allocation:** Lack of skilled resources or overburdening the team leads to burnout and poor-quality deliverables. Proper workload distribution and resource planning are necessary.
7. **Unrealistic Timelines and Deadlines:** Setting unachievable deadlines without considering the complexity of tasks results in stress and low-quality work. Projects should follow realistic estimations using techniques like Story Points and Velocity Calculation in Agile.
8. **Lack of Executive Support:** If project sponsors or senior management are not involved or engaged, decision-making is delayed. Strong leadership and clear decision-making processes are required.
9. **Poor Team Collaboration:** Conflicts within the team, lack of motivation, and ineffective leadership affect project execution. Encouraging teamwork and conflict resolution methods like the Thomas-Kilmann Model can help.
10. **Technical Challenges and Poor Technology Selection:** Using outdated or incompatible technology can create integration issues and increase maintenance costs. Proper feasibility analysis should be conducted before finalizing the technology stack.
11. **Lack of Testing and Quality Assurance:** Insufficient testing leads to defects in the final product, increasing rework costs. Implementing Continuous Testing and Automated QA ensures software reliability.
12. **External Factors (Market & Economic Conditions):** Regulatory changes, economic downturns, and competitive disruptions can impact project viability. Conducting SWOT and PESTLE Analysis helps in anticipating external risks.

**Q11. List the Challenges faced in projects for BA**

**Answer:**

1. **Unclear or Changing Requirements**: Stakeholders may not have a clear vision of the requirements. Continuous changes in requirements lead to scope creep and delays.
2. **Stakeholder Conflicts:** Different stakeholders may have conflicting interests, making it difficult to finalize requirements.
3. **Lack of Stakeholder Engagement**: Some stakeholders may be unresponsive or unavailable, delaying decision-making.
4. **Communication Gaps:** Misinterpretation of requirements can lead to incorrect development.
5. **Managing Expectations**: Stakeholders may have unrealistic expectations regarding timelines, budgets, or features.
6. **Technical Constraints:** Some requirements may not be technically feasible or may need additional development time.
7. **Resistance to Change:** Users may be reluctant to adopt a new system or process.
8. **Documentation Overload:** In traditional methodologies like Waterfall, extensive documentation is required, which can be time-consuming.
9. **Limited Access to End-Users:** A BA may not have direct interaction with the actual end-users, making it difficult to capture their true needs.
10. **Time Constraints:** Tight project deadlines may not allow thorough analysis, leading to missing details.
11. **Poor Requirement Traceability:** Difficulty in tracking changes and ensuring all requirements are addressed.
12. **Cultural and Geographical Barriers**: Global teams may face communication and collaboration challenges due to language and time zone differences.

**Q12. Write about Document Naming Standards**

**Answer:**

**Key Principles of Document Naming Standards**

1. **Clarity & Consistency**
   1. Use clear, descriptive names that indicate the document’s purpose.
   2. Maintain a consistent format across all documents.
2. **Avoid Special Characters & Spaces**
   1. Use underscores (\_) or hyphens (-) instead of spaces (e.g., Project\_Requirements\_v1.0.docx).
   2. Avoid special characters like #, $, %, &, \* as they may cause issues in systems.
3. **Include Relevant Information**
   1. A standard document name should include key details such as:
      1. Project Name/Code
      2. Document Type (e.g., BRD for Business Requirements Document)
      3. Version Number
      4. Date (YYYY-MM-DD format)
      5. Author (if needed)
4. **Version Control**
   1. Use a versioning system to track changes.
   2. Example formats:
      1. Project\_X\_BRD\_v1.0.docx (Major version)
      2. Project\_X\_BRD\_v1.1.docx (Minor update)
      3. Project\_X\_BRD\_v1.1\_DRAFT.docx (Draft version)

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

**Answer:**

* Never say "NO" to the client.
* There is no word called as "BY DEFAULT".
* Never imagine anything in terms of GUI.
* Question everything in the world.
* Go to the client with plain mind i.e. with no assumptions.
* Listen to the client very carefully and after he is done, then ask question.
* Don't interrupt the client.
* Never try to give solutions to the client right away.
* Try to concentrate only on important and required things.
* Be like a lotus in mud- if a client comes with a fancy requirement, then talk to the project manager first.
* Requirement hurried is project buried.
* Never criticize the stakeholder.
* Always appreciate the stakeholder even for small efforts.

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

**Answer:**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Packages** | **Sub-Systems** |
| **Definition** | A **package** is a logical grouping of related classes, interfaces, or components within a system to improve organization and modularity. | A **sub-system** is a self-contained module or component of a larger system that performs a specific function. |
| **Scope** | Focuses on organizing classes and interfaces within a system. | Represents a higher-level component that may include multiple packages. |
| **Purpose** | Used for code organization, modularity, and reusability. | Provides a functional unit within a system, often independently developed and tested. |
| **Dependency** | Packages depend on other packages but do not necessarily function independently. | Sub-systems can operate independently or interact with other sub-systems. |
| **Representation** | Represented using UML package diagrams. | Represented using UML component or sub-system diagrams. |
| **Example** | In a food delivery system, a Payment Package may contain classes like Payment Processor, Transaction, and Invoice. | The Payment Sub-System may include multiple packages like Payment Processing, Fraud Detection, and Bank Integration. |

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

**Answer:** Camel casing is a naming convention used in programming and system design where multiple words are joined together without spaces, and each word (except the first in lower camel case) starts with a capital letter.Ex- firstName, lastName

In BA, camel-casing is used in requirements documentation. In requirement documentation, BA often use camel-casing to name the entities like use case, features, user stories like validateCustomerDetails, calculateInterestRate, etc

Business rules, which should be satisfied by the system use camel-casing. While documenting business process or workflows, camel-casing can be used to individual in steps. This will help maintain consistency in the document. The database tables name also uses camel-casing.

**Camel casing is used for the following reasons:**

* Improves readability and clarity.
* Ensures consistency in naming conventions.
* Helps in maintaining clean code standards.
* Widely accepted in many programming languages (Java, C#, JavaScript, Python, etc.).

By using camel casing in the documents, it helps to maintain consistency in the entire document and increases readability.

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

**Answer:** A Development Server is an environment where software applications are built, tested, and modified before being deployed to production. It allows developers and testers to work on the application without affecting live users.

**Key Features of a Development Server:**

* Isolated from production systems.
* Contains test data instead of real user data.
* Used for coding, debugging, and unit testing.
* Supports continuous integration and version control.
* Allows developers to deploy new features without impacting end users.

The accesses a BA has are-

**ReadOnly**-

BA's may be granted with the readonly 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 BA's will be granted limited access to the specific modules in the application. Limited Configuration Access- Means BA have the authority to make changes in certain areas of application where they have the access.

**Q17. What is Data Mapping**

**Answer:** Data Mapping is the process of linking data fields from one source to a corresponding field in another destination. The database contains multiple tables in it. There may come a scenario, where we need to map the data from one table to another. Data mapping is necessary in cases where we want quick manner.

Data mapping is nothing but a process to establish connection between multiple data sources. The purpose of data mapping is to ensure that the data is accurately transferred or converted into different format.

The main purpose of data mapping is-

**Data integration-** While combining the data from different sources, it ensures that the data is properly matched.

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

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

**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**

**Answer:** API (Application Programming Interface) is a set of rules that allows different software applications to communicate with each other. It defines how requests and responses should be structured between different systems. APIs enable seamless integration between applications, making them work together efficiently.

It is the set of rules, protocols and tools that define how different software application should interact with each other.

API allows sharing of only necessary information and keeps the internal system details hidden, which helps the system security.

For the above scenario, establish API communication- set up API communication between your application and other application to exchange data.

Do Data formatting- while sending the data from one application to other, convert the date format from dd-mm-yyyy to mm-dd-yyyy.

While receiving the data from other application, 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.