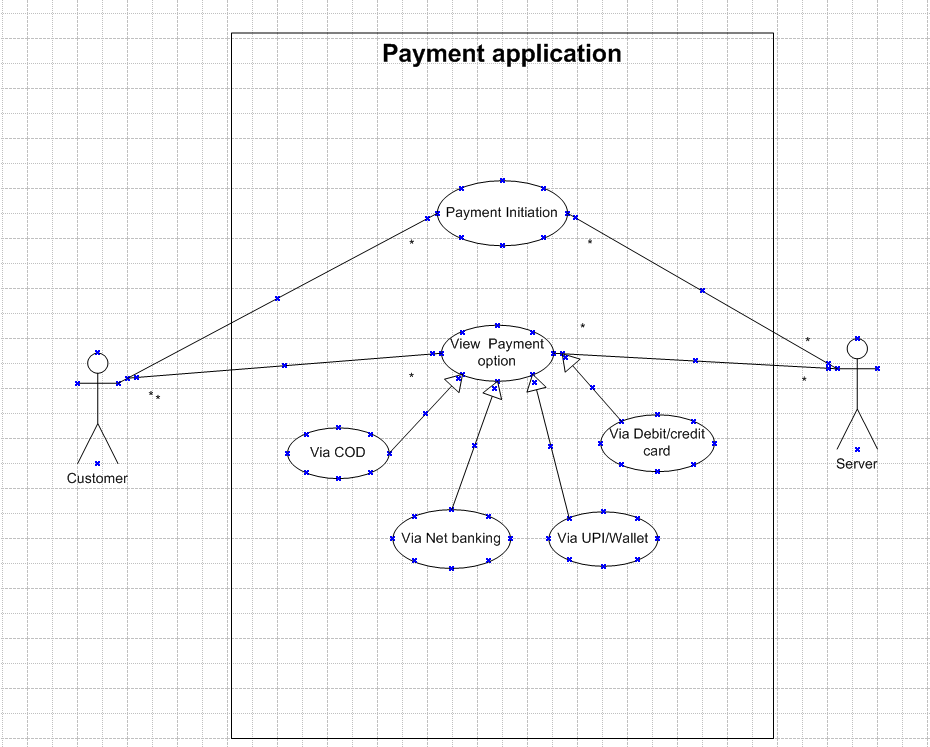
Capstone Project 3 Part 1

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

A Use Case Diagram represents the interaction between the user and the system. It visually describes how a user (actor) interacts with the system's functionalities



Q2. Derive Boundary Classes, Controller classes, Entity Classes

To design a well-structured system, we need to classify elements into three types of classes:

* Boundary Classes: These act as intermediaries between the system and the user. They handle user interactions and pass requests to the controller. In this case, the boundary classes are:

Payment Interface: This class manages how users interact with the payment system.

User Input Interface: It validates user inputs before sending them for processing.

* Controller Classes: These classes handle the logic between boundary and entity classes. They process inputs and communicate with entity classes. Here, we have:

Payment Controller: Ensures payment requests are properly processed.

Payment Processor: Manages the step-by-step execution of transactions.

* Entity Classes: These classes store and manage data relevant to the system. They ensure that data integrity is maintained. Examples include:

Customer: Stores customer details such as name and payment history.

Payment: Represents a payment instance with details like amount, date, and method.

Transaction: Maintains records of completed transactions.

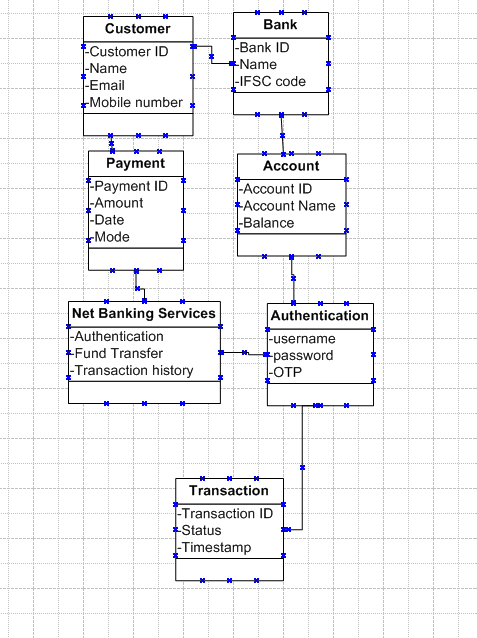
Q3. Place these classes on a three tier Architecture

A three-tier architecture divides the application into three layers:

* Presentation Layer: This layer includes all user-facing components and manages user interactions. The boundary classes (Payment Interface, User Input Interface)
* Business Logic Layer: This layer handles the core functionality of the system. It processes data from the presentation layer and interacts with the data layer. The controller classes (Payment Controller, Payment Processor)
* Data Layer: This layer is responsible for data storage and retrieval. It contains the entity classes (Customer, Payment, Transaction), which store and manage the data.

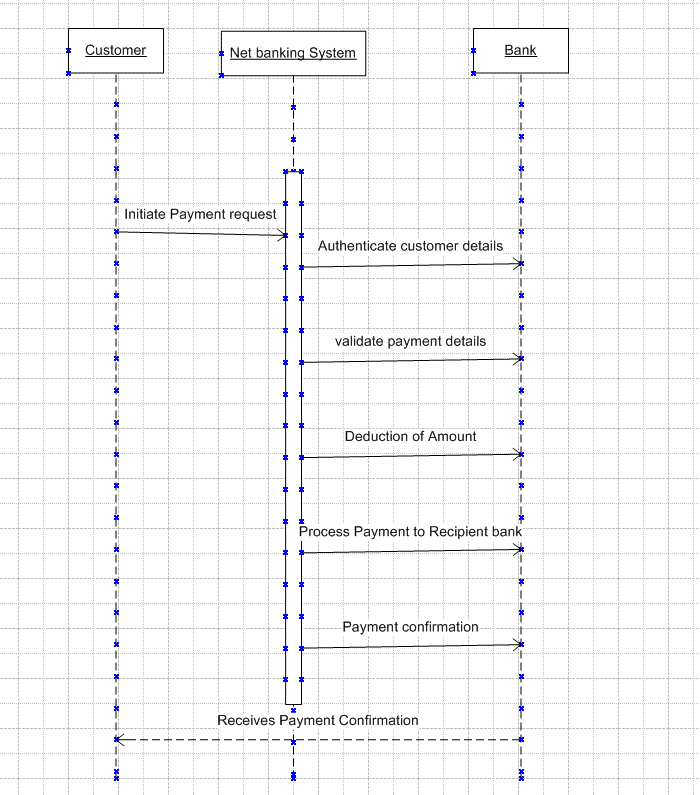
Q4. Explain Domain Model for Customer making payment through Net Banking

A domain model represents real-world entities and their relationships within the system. It provides a structured overview of how different components interact.



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

A sequence diagram illustrates the flow of interactions between objects over time. It shows the step-by-step communication between system components.



Q6. Explain Conceptual Model for this Case

The conceptual model is a high-level representation of the different entities, attributes, and relationships within the system. It is essential for providing a clear understanding of how the different components interact without delving into implementation details.

Components of Conceptual Model:

* Entities: These are the main objects involved, such as Customer, Payment System, Bank, and Transaction.
* Attributes: Each entity has specific characteristics, such as: Customer: ID, Name, Contact Details, Transaction: Transaction ID, Amount, Payment Mode
* Associations:

The Customer interacts with the Payment System.

The Payment System communicates with the Bank for processing.

The Transaction records are updated based on the bank's response.

This conceptual model provides a broad understanding of how different entities function within the system.

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

MVC (Model-View-Controller) is an architectural pattern used in software design that separates an application into three interconnected components:

* Model: Represents the data and business logic of the application. It is responsible for managing the state and retrieving or updating data in response to requests.
* View: Handles the presentation layer and user interface. It is responsible for displaying data to the user and capturing input.
* Controller: Acts as an intermediary between the Model and View. It processes user input, interacts with the Model, and updates the View accordingly.

Rules to Derive Classes from a Use Case Diagram:

* Identify system components and their interactions from the use case diagram.
* Categorize objects into Model, View, or Controller classes based on their responsibilities.
* Ensure controllers manage the workflow by communicating between Model and View layers.

Guidelines for Placing Classes in a Three-Tier Architecture:

* Presentation Layer: Contains UI components and View classes.
* Business Logic Layer: Implements business rules and Controller classes.
* Data Layer: Stores and retrieves data through Model classes.

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

The Waterfall Model is a linear and sequential approach to software development. Each phase must be completed before moving on to the next one. The role of a Business Analyst (BA) is crucial in ensuring that business requirements are properly captured, analyzed, and implemented correctly at each stage. Below is a breakdown of the Waterfall Model and the BA’s contributions at every stage:

* Pre-Project PhaseBefore the actual project starts, businesses analyze the need for the project and determine whether it aligns with their objectives.

BA's Role:

* Identifies business needs and problems.
* Conducts feasibility studies.
* Prepares a high-level business case document.
* Identifies key stakeholders and gathers their expectations.
* Planning PhaseThis phase involves defining the project scope, timeline, budget, and resources needed.

BA's Role:

* Works with project managers to define project scope and deliverables.
* Helps in risk assessment and mitigation strategies.
* Identifies dependencies and constraints.
* Assists in preparing project documentation like a project charter.
* Project Initiation PhaseAt this stage, the project officially begins with resource allocation and setting up the required infrastructure.

BA's Role:

* Works with stakeholders to finalize project goals and objectives.
* Conducts stakeholder meetings to define high-level requirements.
* Helps in drafting the initial requirement documents.
* Ensures stakeholder alignment and approval before proceeding.
* Requirement Gathering PhaseThis phase focuses on collecting detailed functional and non-functional requirements from users and stakeholders.

BA's Role:

* Conducts requirement-gathering workshops and interviews.
* Uses techniques like brainstorming, surveys, and document analysis to capture requirements.
* Creates Business Requirement Documents (BRD) and Functional Requirement Documents (FRD).
* Ensures all business needs are well-documented and agreed upon by stakeholders.
* Requirement Analysis PhaseAfter gathering requirements, BAs analyze them for feasibility, clarity, and completeness.

BA's Role:

* Reviews and prioritizes requirements based on business value.
* Identifies gaps, inconsistencies, and conflicts in requirements.
* Creates process flow diagrams and data models.
* Works closely with technical teams to ensure requirements are achievable.
* Design PhaseIn this phase, the system architecture and design specifications are developed based on the gathered requirements.

BA's Role:

* Collaborates with architects and developers to translate requirements into system design.
* Reviews wireframes, mockups, and UI/UX designs.
* Ensures all business logic is reflected correctly in the design.
* Validates whether the design aligns with the documented requirements.
* Development PhaseThis phase involves coding and building the actual system based on the approved design.

BA's Role:

* Works closely with developers to clarify business logic.
* Addresses any requirement changes or clarifications needed.
* Ensures that the development follows the defined requirements.
* Participates in sprint planning and review meetings (if Agile practices are followed within the Waterfall framework).
* Testing PhaseOnce development is complete, the system undergoes rigorous testing to identify and fix bugs.

BA's Role:

* Assists in defining test scenarios and acceptance criteria.
* Reviews test cases to ensure all business requirements are covered.
* Works with QA teams to validate system behavior.
* Participates in defect triaging and resolution discussions.
* User Acceptance Testing (UAT) PhaseUAT is the final stage before deployment, where end-users validate that the system meets business needs.

BA's Role:

* Facilitates UAT sessions with business users.
* Collects feedback and ensures all critical issues are resolved before go-live.
* Provides user training and documentation.
* Ensures sign-off from key stakeholders for production deployment.
* Deployment & Maintenance Phase Once the system is tested and approved, it is deployed to the live environment. The maintenance phase ensures smooth operation post-deployment.

BA's Role:

* Assists in change management and system rollouts.
* Gathers user feedback and suggests improvements.
* Monitors system performance and identifies enhancement opportunities.
* Provides support for any post-implementation changes.

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

Conflict management is the process of identifying, addressing, and resolving disputes efficiently and fairly. It ensures that disagreements do not disrupt productivity and team dynamics. One of the most widely used conflict resolution models is the Thomas-Kilmann Conflict Mode Instrument (TKI), which categorizes conflict-handling styles into five approaches:

* Competing: A high-assertiveness and low-cooperativeness style where one party pursues their concerns at the expense of others. Used when quick, decisive actions are required.
* Collaborating: A high-assertiveness and high-cooperativeness style that seeks a win-win solution. It involves discussing and integrating multiple perspectives.
* Compromising: A moderate approach where both parties give up something to reach a mutually acceptable solution.
* Avoiding: A low-assertiveness and low-cooperativeness style where the issue is ignored or postponed. Used when conflicts are trivial or when more information is needed.
* Accommodating: A low-assertiveness and high-cooperativeness style where one party gives in to maintain harmony. Used when preserving relationships is more important than personal gain.

Q10. List down the reasons for project failure

Projects fail due to various reasons, including:

* Unclear Requirements: Poorly defined or constantly changing requirements lead to confusion and misalignment.
* Lack of Stakeholder Engagement: If key stakeholders are not involved, the project may not align with business needs.
* Poor Project Planning: Inadequate planning, unrealistic timelines, and incorrect resource allocation can lead to failure.
* Scope Creep: Uncontrolled changes in project scope result in delays and increased costs.
* Ineffective Communication: Miscommunication between team members, stakeholders, or clients leads to misunderstandings and conflicts.
* Lack of Risk Management: Failure to anticipate risks and create mitigation strategies can cause unexpected disruptions.

Q11. List the Challenges faced in projects for BA

A Business Analyst (BA) encounters several challenges while working on projects, including:

* Changing Requirements: Stakeholders may modify requirements frequently, leading to scope changes and rework.
* Lack of Stakeholder Availability: Decision-makers may be busy, delaying approvals and impacting project timelines.
* Limited Domain Knowledge: BAs often work across industries and may require time to understand specific business processes.
* Managing Conflicting Interests: Different stakeholders may have varying priorities, requiring negotiation and alignment.
* Technology Constraints: Legacy systems or incompatible technologies may limit implementation possibilities.
* Ensuring Requirement Clarity: Misinterpretations of requirements can result in incorrect implementations, requiring detailed documentation and validation.

Q12. Write about Document Naming Standards

* Use a Consistent Format: Follow a structured naming convention like ProjectName\_DocumentType\_Date\_Version (e.g., Agricultureplatform\_Requirements\_2024\_V1).
* Avoid Special Characters: Use only letters, numbers, and underscores to prevent system compatibility issues.
* Include Version Numbers: Maintain versioning (e.g., V1, V2) to track document revisions.
* Specify Date Format: Use YYYY-MM-DD to maintain chronological order.
* Describe Content Clearly: Ensure the file name conveys its purpose without needing to open the document.

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

|  |  |  |
| --- | --- | --- |
| Sr.No | Dos | Don’ts |
| 1 | Consult an SME for clarifications in requirement | Never say no to the client |
| 2 | Go to the client with a plain mind with no assumptions. Listen carefully and completely until the client is done, and then you can ask queries | There is no word as By default |
| 3 | Try to extract maximum leads to the solutions from the client himself | Never imagine anything in terms of GUI |
| 4 | Concentrate on the important requirement | Don’t interrupt the client when he is giving you the problem |
| 5 | Question the existence of the existence. Question everything | Never try to give solutions to the client straight away with your previous experience and assumptions |

Q14. Write the difference between packages and sub-systems

Package: A package is a grouping of related classes, interfaces, and sub-packages that help in organizing code efficiently. It is used to manage complexity and modularize large projects.

Sub-System: A sub-system is a self-contained module within a system that performs a specific function. It can have multiple packages and components interacting within it.

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

Camel casing is a naming convention where multiple words are joined without spaces, and each word starts with an uppercase letter except the first one.

Examples:

customerPayment (lower camel case)

CustomerPaymentDetails (upper camel case or Pascal case)

It is used for

Programming Languages: Used in Java, C#, JavaScript for naming variables and functions.

Database Naming: Applied in column and table names for better readability.

API Development: Used for naming methods and parameters.

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

A Development Server is an environment where software is built, tested, and modified before moving to production.

BA’s Access Rights:

* Read-Only Access: View logs, reports, and configurations to verify implementation.
* Requirement Validation: Check if the developed features align with business requirements.
* User Acceptance Testing (UAT): Assist in UAT execution by verifying system functionality.
* Collaboration with Developers: Discuss changes and clarify requirements if needed.

BAs usually do not have administrative or coding access but play a crucial role in ensuring the system functions as intended.

Q17. What is Data Mapping

Data Mapping is the process of defining relationships between different data models to enable seamless data integration, transformation, and migration. It is used when data needs to be transferred between different sources and destinations while maintaining accuracy and consistency.

Importance of Data Mapping:

* Ensures Data Consistency: Helps in maintaining uniformity when transferring data between systems.
* Enables Data Transformation: Converts data into a format suitable for the target system.
* Aids Data Migration: Facilitates smooth transition when moving data from one database to another.
* Improves Data Integration: Helps in merging data from multiple sources for analytical or operational use.

Types of Data Mapping:

* Manual Data Mapping: Analysts manually match fields between source and destination databases.
* Automated Data Mapping: Software tools use algorithms to map fields automatically.
* Schema Mapping: Aligns data fields from different databases with varying structures.
* Semantic Mapping: Ensures meaning consistency between different dataset

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

An API (Application Programming Interface) is a set of rules and protocols that allow different software applications to communicate with each other. APIs enable systems to exchange data and functionalities seamlessly without requiring direct user interaction.

Types of APIs:

* REST (Representational State Transfer): Uses HTTP requests for data exchange.
* SOAP (Simple Object Access Protocol): A protocol-based API used for secure communication.
* GraphQL: Allows clients to request specific data instead of retrieving a full dataset.
* Webhooks: Sends real-time data updates when specific events occur.

Example Use Case: Handling Different Date Formats in API Integration

Consider a scenario where an API is used to fetch data from an application in the US (which uses MM-DD-YYYY format), and the receiving application follows the DD-MM-YYYY format.

Steps to Handle Date Format Differences:

* Data Extraction: The API fetches data from the source system (US application).
* Format Conversion: Before sending data to the target system, a middleware layer or script is used to transform the date format from MM-DD-YYYY to DD-MM-YYYY.
* Validation: The converted date is checked to ensure accuracy before inserting it into the target database.
* Data Transmission: The formatted data is sent to the destination application using an API request.
* Error Handling: If an invalid date is detected, the system triggers an error response with necessary logs.

Using APIs efficiently in such scenarios ensures seamless data exchange, reduces manual errors, and enhances interoperability between applications.