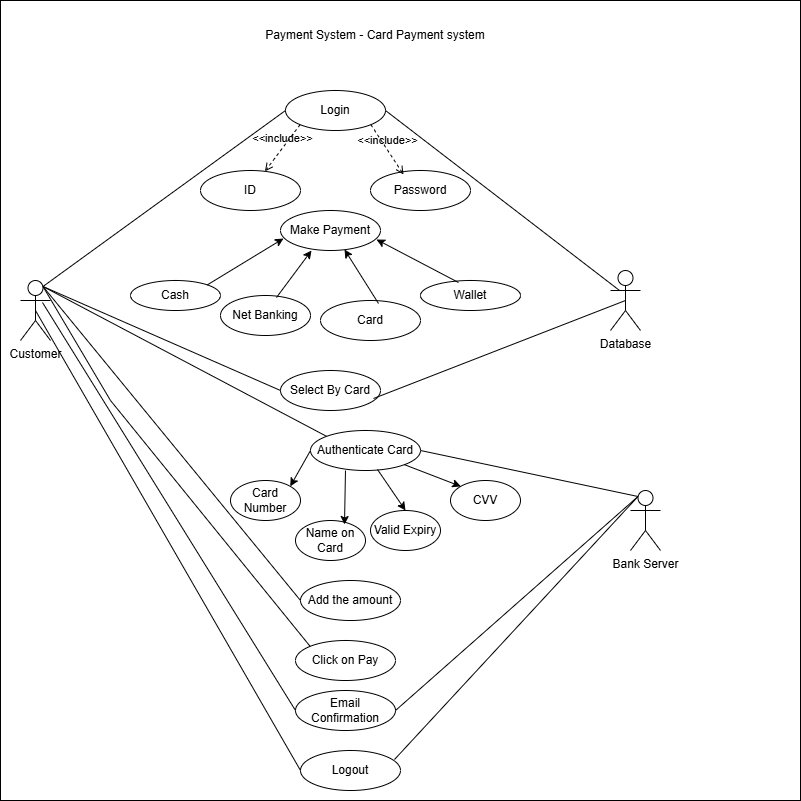
A customer can make a payment either by Card or by Wallet or by Cash or by Net banking.

**Q1. Draw a Use Case Diagram**



**Q.2 Derive Boundary Classes, Controller classes, Entity Classes**

Boundary Classes –

These represent the interface between the system and its actors.

Combination of one actor and a use case is one boundary class.

Combination of 2 actors and a use case is two boundary class.

Combination of 3 actors and a use case is three boundary class and so on. And those actors should be primary actors.

|  |  |
| --- | --- |
| Customer Login | Takes the customer ID and password to log them into the system |
| Customer logout | Logs out the customer. |
| Bank Server Login | Handles card authentication details like Card Number, Name, CVV, and Expiry Date. |
| Bank Server Logout | Logs out the bank server. |
| Make Payment Options | Lets the customer choose how they want to pay (Wallet, Cash, Net Banking, or Card). |

Controller Classes –

These act as coordinators, controlling the flow of data and decisions.

|  |  |
| --- | --- |
| Login Controller | Handles the process of verifying user credentials (ID &Password). |
| Payment controller | Coordinates the payment process, including the selection of payment methods and verification. |
| Card Authentication Controller | Validates card details (Card Number, CVV, Expiry Date with the Bank Server. |
| Confirmation Controller | Sends payment confirmation and Email confirmation after successful payment. |
| Logout Controller | Manages customer and bank server logout processes. |

Entity Classes –

These represent the core and business logic of the system. Each Actor will be considered as one entity.

|  |  |
| --- | --- |
| Customer Information | Stores user data like ID and Password. |
| Payment Information | Manages payment methods (Wallet, Net Banking, Card details.) |
| Card Details | Stores and validates Card Number, Name, CVV and Expiry Date. |
| Database | Managers all system data storage and retrieval. |

**Q.3 Place these classes on a three tier Architecture.**

1) Application Layer -

This layer is responsible for interacting with the user or external actors (e.g., customers, bank servers). It contains the **Boundary Classes**.

2)Business Layer -

This layer contains the core logic and decision-making processes of the system. It processes the data received from the application layer and interacts with the data layer. It includes **Controller Classes**.

3) Data Layer -

This layer handles the storage and retrieval of data. It includes the **Entity Classes**, which represent the data and its related operations.

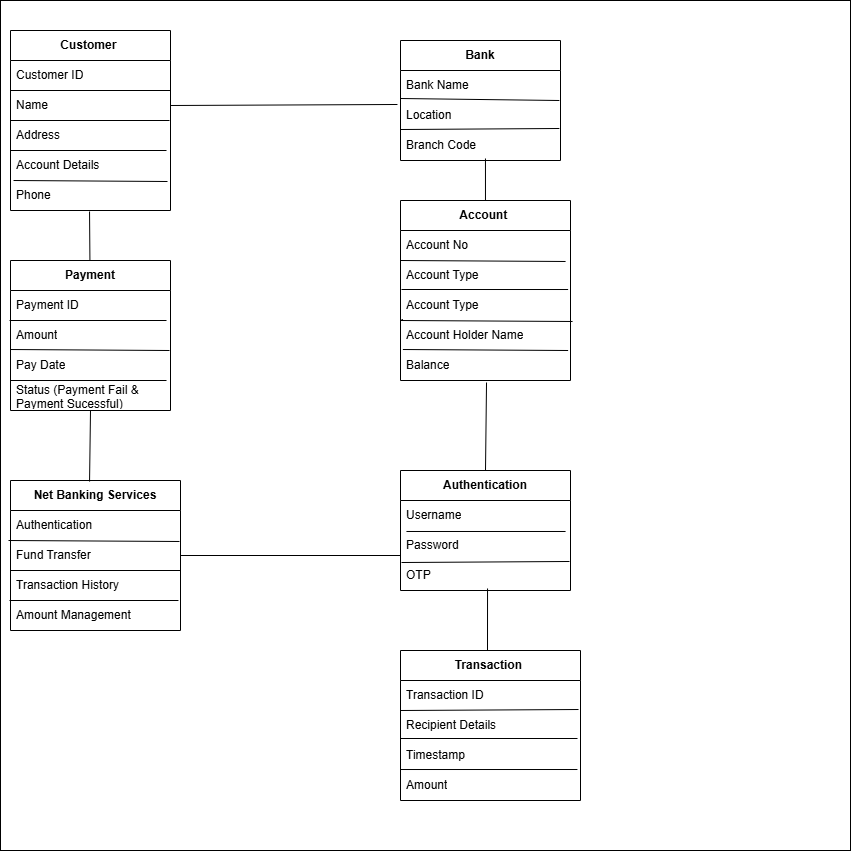
Entity Classes

* Customer Information – Stores user credentials like ID and Password)
* Payment Information – Manages payment Methods.
* Card Details – Validates card data.
* Database – Handles storage and retrieval of system data.

|  |  |
| --- | --- |
| Tier | Classes |
| Application Layer | Customer Login, Customer Logout, Bank Server Login, Bank Server Logout, Make Payment Options. |
| Business Layer | Login Controller, Payment Controller, Card Authentication Controller, Confirmation Controller, Logout Controller. |
| Data Layer | Customer Information, Payment Information, Card Details, Database. |

**Q.4 Explain Domain Model for Customer making payment through Net Banking**

A **Domain Model** is a visual representation of the main concepts, entities (things), and their relationships in a specific problem area or system. It focuses on **what exists** in the system rather than how it works. It helps you understand and explain the **business logic** or **real-world concepts** related to the project.



**Q. 5 Draw a sequence diagram for payment done by Customer Net Banking**

A screenshot of a diagram

AI-generated content may be incorrect.

**Q.6 Explain Conceptual Model for this Case**

A Conceptual Model provides an abstract and high-level view of a system, focusing on the key entities, their relationships, and business rules, without technical details like database structures or implementation.

1. **Customer -**

The customer opens the login page, enters their username and password, and then types in the payment amount. After entering the details, the customer submits the payment amount and logs out once the process is complete.

**Business Role** - Payment Initiation

1. **Login Page –**

The login page displays where the customer enters their login credentials. It then validates the customer’s username and password. If the authentication is successful, the customer is redirected to the payment page.

**Business Role -** Authentication Interface

1. **Net Banking –**

Net Banking verifies the customer’s credentials and starts the payment process. It then directs the customer to the payment page. After the payment is completed, it confirms the transaction and ensures the customer has logged out after finishing the process.

**Business Role –** Payment Processor

1. **Bank –**

The bank receives the payment request from the net banking system andprocesses the payment. It deducts the required amount from the customer’s account and sends a payment confirmation back to the net banking system.

**Business Role –** Financial Institution

**Q.7** **What is MVC architecture? Explain MVC rules to derive classes from use case diagram and guidelines to place classes in 3-tier architecture**

MVC is a design paern where, the application is divided into three logical parts- Model, View and Controller.

**Model –**

The Model represents the data and the business logic of the application. Model is responsible for multiple tasks like managing the application's data, performing data validation, implementing business rules, and handling data access operations**.** Model does not depend on how the data is presented or how the user interacts with the application. All the model classes are nothing but the entities. Model classes are represented as entity class.

Ex. Customer, Payment, Net Banking, Card, Cash.

**View -**

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

**Controller -**

The controller acts as an intermediary between the Model and the View. It receives input from the user, 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. Whenever the user requests for anything, that request first goes to the controller. Controller works on the user's request.

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

**Advantage 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.

**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 and so on.…
4. Use case will result in a controller class.
5. Each Actor will result in one entity class.

**Guidelines to place classes in 3-tier Architecture**

1. **Application Layer -**

This is the topmost layer that interacts with the user. It includes UI components (web pages, mobile app screens, dashboards, forms etc.) It sends user inputs to the Business Logic Layer and displays results. Handles UI and user interactions.

Login Form – Handles user login interface. Dashboard View – Displays reports and charts. User Controller – Handles user-related requests.

1. **Business Logic Layer –**

This layer contains all the business rules and application logic. It processes requests from the Presentation Layer, applies rules and interacts with the Data Layer. Ensure data integrity and security. Processes logic and applies business rules. User Service – Handles authentication and user management, Order Processor – Validates and processes orders. Payment Manager - Calculates payments and applies business rules.

1. **Database Layer -**

This layer is responsible for storing and retrieving data from the database. It communicates with databases, file systems or external data sources. Ensures that data is accessed securely and efficiently. Manages database interactions.

Payment Gateway – interacts with external payment APIS.

**Q. 8** **Explain BA contributions in project (Waterfall Model – All Stages**)

Waterfall model follows sequential approach. In this model, each phase is completed entirely and then only moved to the next phase.

|  |  |  |  |
| --- | --- | --- | --- |
| **Stages of Waterfall Model** | **Activities** | **Artifacts** | **Resources** |
| Requirement Gathering | Identify Business needs. Conduct Stakeholder meetings. Gather and document requirements. | BRD, Meeting Notes, Stakeholder approvals. | BA, Stakeholders, Project Manager. |
| Requirements Analysis | Analyze requirements. Validate and prioritize needs. Identify dependencies and risks. | FRD, SRS, Requirements Traceability Matrix | BA, PM, Tech Team |
| Design | Create system architecture. Define data flow and UI/UX design. Prepare technical specifications. | System Design Document (SDD), Data Flow Diagrams (DFD) Wireframes. | Development Team,  UI/UX Designer, Solution Architect. |
| Implementation | Develop code based on design. Integrate system components. Perform unit testing. | Source code, Development Logs, API Documentation. | Programmers Developers. Tech Leads. Database Engineers. |
| Testing | Perform functional and system testing.  Conduct UAT with stakeholders.  Fix defects and retest. | Test Cases, UAT Plan, Defect Logs | QA Testers, BA, End Users. |
| Deployment | Deploy system to production.  Monitor for early issues.  Train end users. | User Manuals, Training Materials, Checklist | IT Team, End Users. |
| Maintenance | Monitor system performance. Implement change requests. | Change Request Documents  Maintenance Reports | Developers,  Business Analysts. |

**Q. 9 What is conflict management? Explain using Thomas – Kilmann technique?**

**Conflict Management –**

Conflict management refers to the process of identifying, addressing, and resolving conflicts in a way that minimizes negative outcomes and enhances the team's ability to work effectively. In project management or organizational settings, conflicts often arise due to differing opinions, priorities, or resource constraints. Effective conflict management ensures collaboration, maintains relationships, and fosters a productive work environment.

**Thomas-Kilmann Conflict Management Technique**

**The Thomas-Kilmann Conflict Mode Instrument (TKI)** is a widely used framework for understanding and resolving conflicts. It identifies five conflict resolution styles based on two dimensions:

**X-Axis – Co-operation/ Cooperativeness** - Represents how much a person tries to satisfy the needs, concerns or goals of others.

**Y-Axis – Assertiveness –** Represents how much a person tries to satisfy their own needs, concerns or goals.

**Types of conflict management –**

1. **Competing - (High Assertiveness, Low Cooperativeness)**

Definition: A "win-lose" approach where one party prioritizes their own goals over others, often at their expense.

When to Use: When a quick decision is needed, or the issue is too important to compromise on. It works best in emergencies or when enforcing rules.

Example - A project manager sticks to a strict deadline, even if the team feels pressured, to make sure the project is completed on time.

1. **Avoiding** - **(Low Assertiveness, Low Cooperativeness)**

Definition - It's a "lose-lose" approach, This happens when a person ignores or postpones a conflict instead of addressing it.

When to Use: When the issue is minor, when emotions are running high, or when more time or information is needed before making a decision.

Example: A manager delaying a discussion about budget cuts until the annual review to avoid immediate tension.

1. **Accommodating (Low Assertiveness, High Co-operativeness) -**

Definition: A "lose-win" approach where one person prioritizes the other’s needs over their own, often to maintain harmony or avoid conflict.

When to Use: When the issue is more important to the other person, to maintain relationships, or when peace is more valuable than winning.

Example: A team member agrees to take on extra work to avoid conflict with a colleague.

1. **Collaborating (High Assertiveness, High Cooperativeness)**

Definition: A "win-win" approach where both sides actively work together to find a solution that benefits everyone.

When to Use: Best for complex issues that require creativity, long-term problem-solving, or when maintaining strong relationships is important.

Example: A team discusses and brainstorms different ways to redistribute tasks so that everyone has a fair workload while still meeting deadlines.

1. **Compromising (Moderate Assertiveness, Moderate cooperativeness)**

Definition: A middle-ground approach where both sides make concessions to reach a mutually acceptable solution.

When to Use: Useful when a quick resolution is needed, when both parties have equal power, or when a temporary or partial solution is acceptable.

Example: Two departments agree to share resources equally so both can meet their goals without conflicts.

5 Steps to Conflict Management –

* Identify the conflict
* Discuss the details
* Agree with the root problem
* Check for every possible solution for the conflict negotiate. The solution to avoid future conflicts.

**Q.10. List down the reasons for project failure.**

**1) Improper Requirements Gathering –**

If the project requirements are not well understood or documented, the final product may not meet user expectations. This leads to rework, delays, and sometimes complete project failure.

**2) Continuous change in requirements –**

Frequent changes in project scope without proper planning can cause confusion, increase costs, and delay delivery. It also affects resource allocation and project stability.

3)**Lack of user involvement –**

If end-users are not involved during the development process, the final product may not align with their needs. This results in poor adoption and dissatisfaction.

4)**Lack of executive support –**

Without strong leadership and support from top management, projects may lack funding, resources, and strategic direction, making them difficult to complete successfully**.**

**5) Unrealistic expectations –**

Setting deadlines, budgets, or goals that are not achievable can create pressure on teams, leading to burnout, poor-quality work, and eventual project failure**.**

**6) Improper planning**

Poor planning leads to mismanagement of time, resources, and budget. If a project lacks a clear roadmap, risk assessment, or contingency plans, it can face delays, cost overruns, and failure to meet objectives. Proper planning ensures smooth execution and minimizes uncertainties.

**Q.11 List the Challenges faced in projects for BA.**

**1) Lack of Training –** BA may not always receive proper training on tools, processes or domain knowledge, making it difficult to perform effectively.

**2) Change Management with respect to cost and timelines-** Managing changes in requirements while keeping the project within budget and deadlines can be challenging. Uncontrolled changes can lead to scope creep and delays.

**3) Coordination between developers & testers-** Ensuring smooth communication between development and testing teams is crucial. Misalignment can lead to defects, rework, and project delays.

**4) Conducting Meetings** - Organizing meetings effectively, ensuring stakeholder participation, and keeping discussions on track can be difficult, especially when dealing with busy stakeholders.

**5) Making sure status reporting is effective -** Providing clear, concise, and accurate updates on project progress to stakeholders and management is essential to avoid miscommunication.

**6) Driving clients for UAT completion -** Clients may delay User Acceptance Testing (UAT) due to their own workload or lack of clarity, which can impact project delivery timelines.

**7) People Management (coordinating with different people and different teams)**

Coordinating with multiple teams, handling different working styles, and resolving conflicts between team members can be a major challenge.

**8) Ensuring Project Health and Timely Delivery** – Keeping the project on track, managing risks, and ensuring it is completed without major issues is a key responsibility of a BA.

**Q.12 Write about Document Naming Standards.**

**Document Naming Standards**

Document naming standards help keep files organized and easy to find. A good naming system makes sure that everyone in a project or team can quickly understand what a document is about. These standards usually include details like the document’s purpose, date, version number, and other important information.

* Project ID – Identifies the specific project.
* Document Type – Specifies the document category (e.g. BRD)
* Version Number V (x) – Indicates the version of the document.
* Draft or Revision Number D (y) – Specifies the draft or revision stage.
* File Extension – Represents the file type (e.g. docx. Pdf)

All documents will be named using some standards like..

(Project ID) (Document Type) V (x) D (y). extension

Example:

**PQ786BRDV1D2.docx** (Project ID: PQ786, Document Type: BRD, Version 1, Draft 2)

**PQ786BRD1.2.docx** (Project ID: PQ786, Document Type: BRD, Version 1.2)

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

|  |  |
| --- | --- |
| **Do’s** | **Don’ts** |
| Listen to the client carefully. | Don’t interrupt the client while they speak. |
| Ask questions after the client is done talking. | Don’t assume anything. |
| Go to the client with a clear mind. (no assumptions). | Don’t say “No” to the client directly. |
| Focus only on important and required things. | Don’t imagine things in terms of GUI. |
| Always question and clarify doubts. | Don’t give solutions immediately. |
| Talk to the project manager if the client asks for fancy features. | Don’t criticize stakeholders. |
| Appreciate stakeholders for their efforts. | Don’t use the term “By default” in requirements. |
| Be patient and ensure clarity in requirements. | Don’t rush through requirements or the project may fail. |

**Q.14 Write the difference between packages and sub-systems**

|  |  |
| --- | --- |
| **Packages** | **Sub-systems** |
| Package can be termed as a collection of components which are not reusable in nature. | Sub-system can be termed as a collection of components which are reusable in nature. |
| Not reusable in different systems. | Can be reused across multiple systems. |
| Application development companies work on packages. | Product Development companies work on subsystems. |
| Packages are smaller and more focused in scope. | Sub-systems are larger and encompass multiple packages or modules. |
| They manage dependencies at class and component level. | They manage dependencies at a higher level, defining boundaries and interfaces between different parts of the system. |
| Limited in scalability, built for a specific solution. | Highly scalable and used in various applications. |

**Q.15 What is camel-casing and explain where it will be used.**

Camel-casing refers to the naming convention of variables, parameters or properties. Here, multiple words are combined without space. There will be no gap between words.

* The first letter of the first word is lowercase.
* The first letter of each subsequent word is in uppercase.

Ex. First Name, Last Name

Where it will be used:

* Requirement Documentation**—BA often uses came-casing to name entities such as use cases, features, and user stories, such as validating Customer Details and calculating Interest rates**.
* Business Rules **–** Rules that need to be followed by the system can use camel casing for consistency.
* Business Process Documentation – While documenting process flows or workflows, camel casing can be used to maintain clarity and readability in steps.
* Database Naming Conventions- Column names, table names and attributes often follow camel casing.
* Requirement Naming – Functional and non-functional requirements in BRD/FRD use camel casing for uniformity.
* Consistency and Readability – Using camel casing throughout documents ensures consistency, better readability and standardization.

**Q.16 Illustrate Development server and what are the accesses does business analyst has.**

A Development Server is an environment where developers write, test, and debug code before deploying it to production. It is an isolated space that allows teams to experiment with new features and ensure functionality without affecting live users.

BA’s may have different levels of access to the Development Server depending on project requirements.

**Business Analyst Access Levels -**

**1) Read-Only Access** –

* BAs may be granted read-only access to the development server.
* They can view the user interface (UI), navigate through features and observe the application’s behavior.
* No ability to modify or configure the system.

2) Limited Access –

* BA’s may receive access to specific modules based on project needs.
* Allows them to review workflows and validate that development aligns with business requirements.

3) Limited Configuration Access –

* In some cases, BA’s may have authority to make changes in specific areas of the application.
* Typically applies to configurable settings, such as updating business rules, modifying workflow settings or adjusting UI labels.

**Q.17 What is Data Mapping**

Data mapping is the process of **establishing connections between multiple data sources** to ensure **accurate transfer, integration, or conversion** of data. It helps in structuring data efficiently when moving it from one system or format to another.

The main purpose of data mapping is –

1. Data Integration –

* A continuous process of transferring data between systems.
* Typically occurs in real-time or on a scheduled basis, triggered by events or system workflows.

2. Data Migration -

* A one-time transfer of data from a legacy system to a new system.
* Once the migration is complete, the original data source is usually retried.

3. Data Transformation –

* Converts data from one format to another (e.g. Date formats, number formats or database structures) to ensure compatibility with the new system.
* May involve removing duplicates, handling null values, enriching data, or changing data types.

4. Data Warehousing –

* Consolidates data from multiple sources into a central repository for reporting, analysis and queries.
* Data in a warehouse has already undergone migration, integration and transformation.

**Q.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 whose Date Format is mm-dd-yyyy.**

API stands for Application Programming Interface.

It is a set of rules and protocols that allow different software applications to communicate with each other. It defines the methods and data formats that programs can use to request and exchange information. APIs enable applications to interact with each other in a standardized way without needing to understand the internal workings of the other systems.

Application expects dates in the dd-mm-yyyy format, but the external system from the US sends dates in mm-dd-yyyy format. Using API integration, your app will send a request to the US. Then, you can convert the date to dd-mm-yyyy using simple code (like in Python or Java Script). After the conversion, your app can display or store the date in the correct format. This ensures smooth data exchange between the 2 systems without errors.