Nurturing Process - **Capstone Project 3** – 1/2

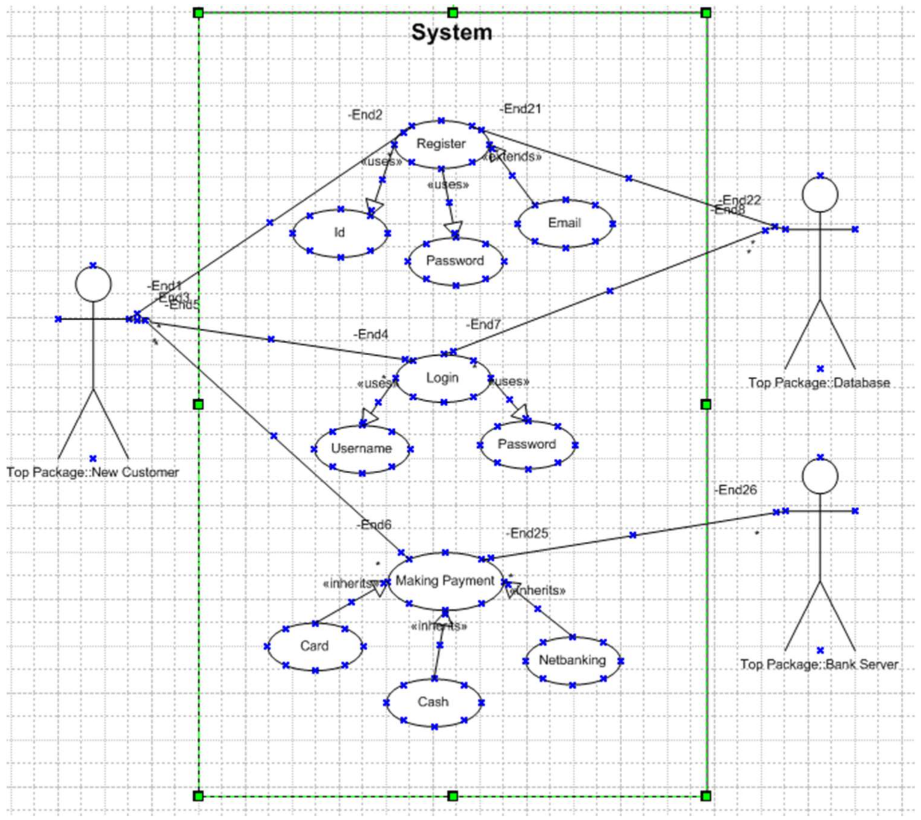
COEPD – Traditional Development

Project Name: **Case Study – 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

Answer:



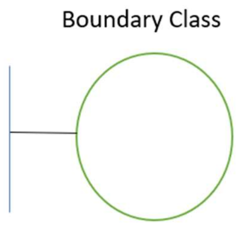
Q2. Derive Boundary Classes, Controller classes, Entity Classes.

Answer:

Boundary Classes, Controller classes, Entity Classes are derived as below,

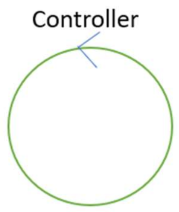
Boundary Classes: A Boundary Class is a type of class used in software design, particularly in the context of object-oriented design and the Model-View-Controller (MVC) architecture. It acts as an interface between the system and external factors, such as users, external systems, or devices. Combination of 1 actor and a use case is one boundary class, Combination of 2 actors and a use case is two boundary class.

Example: Customer Registration, Customer Login, Bank Server Login.



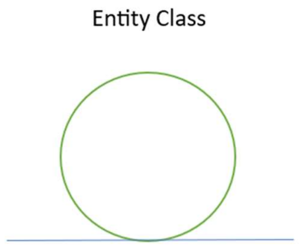
Controller Classes: A Controller Class is a key component in software design, particularly in the Model-View-Controller (MVC) architectural pattern. Its primary role is to manage the flow of data between the Model (business logic and data) and the View (user interface). It processes incoming requests, handles user input, and updates the model and view accordingly.

Example: LoginController, PaymentController, EmailController.



Entity Classes: An Entity Class represents core business objects or concepts in a software application. It typically encapsulates data and business logic related to that data. Entity classes are often used in the Model layer of the Model-View-Controller (MVC) design pattern and are central to the Domain Model in domain-driven design (DDD).

Example: Customer, Bank Server.



Q3. Place these classes on a three tier Architecture.

Answer:

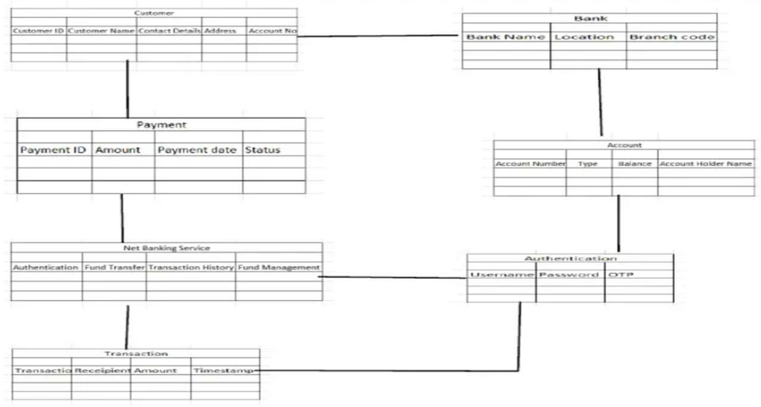
Application Layer	• CustomerRegistration • CustomerLogin • BankServerLogin
Business Logic Layer (Primary Actor associated with the Boundary Class)	• Customer • Bank Server
Data Layer (All the Entity Classes)	• Customer • Bank Server • Cash • Card • Net banking

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

Answer:

Domain Modelling: Domain Modelling is the process of creating a conceptual representation of the key elements, relationships, and rules within a specific domain or problem space in software development. It helps define how data and processes are structured in a system by modelling real-world entities and their interactions. Key Elements can be mentioned as below,

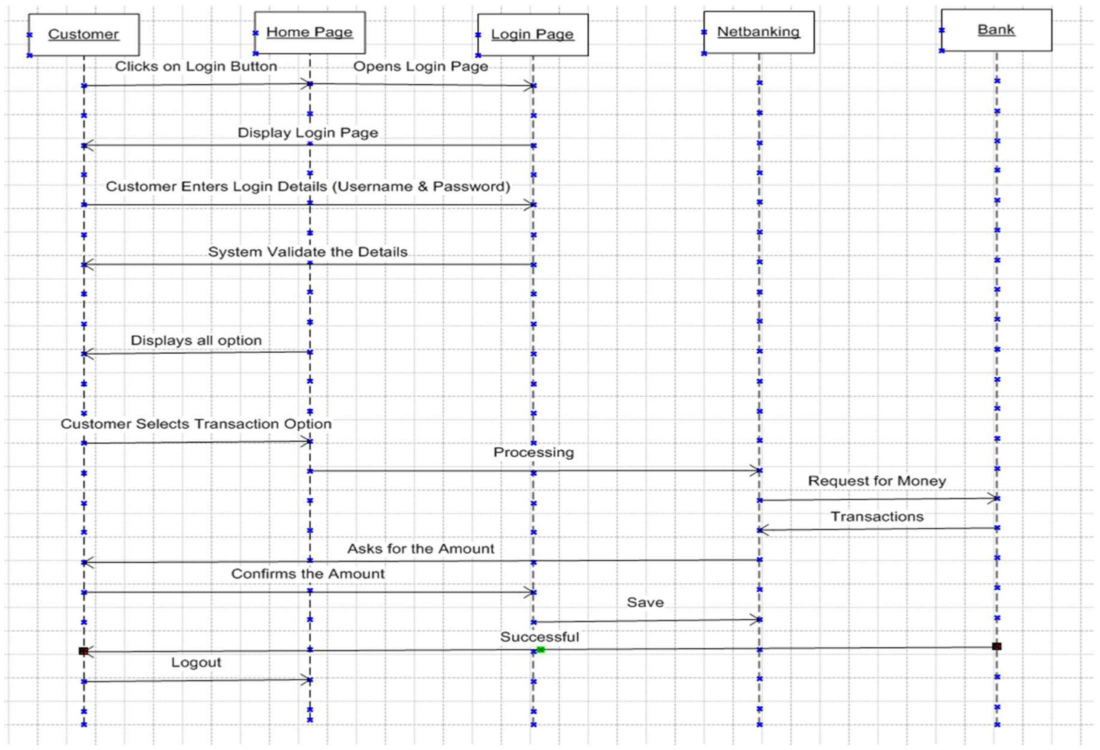
- **Entities**: Represents, the key or the object within the domain. Example: Customer, Payments, Order.
- **Attributes**: Properties of a particular entity. Example: Name, Email, Phone, Address, DOB.
- **Relationships**: Connections or Interaction between the Entities. Example: CUSTOMER places ORDER & ORDER contains multiple PRODUCTS.
- **Aggregates**: Entities, which are grouped in a single unit. Example: ORDER contains ORDER ITEM & SHIPPING DETAILS.
- **Business Rules**: These can also be called as the Constraints which defines the Domain. Example: Payment is the pre-requisite for the Delivery of the Product.



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

Answer:

Sequence Diagram: A Sequence Diagram is a type of UML (Unified Modelling Language) diagram that visually represents the interaction between objects or components in a system over time. It shows how objects communicate through method calls or messages in a sequential order.



Q6. Explain Conceptual Model for this Case

Answer: A Conceptual Model is a high-level representation of how a system or process works. It focuses on abstract concepts and relationships between them, providing a simplified overview of the system's structure without diving into technical details. Here are some key components below,

- **Entities:** The main objects or concepts in the system, Example: Customer, Product.
- **Attributes:** Descriptive properties of the entities, Example: Customer Name, Product Price.
- **Relationships:** Connections between entities, Example: A Customer places an Order.
- **Constraints:** Rules governing how entities interact, Example: Each order must have at least one product.

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 Architecture: MVC Architecture stands for Model-View-Controller, a design pattern used to develop user interfaces by separating application logic into three interconnected components. This separation makes applications more modular, scalable, and easier to maintain.

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. The model class is known 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 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. All the model classes are nothing but the entities. Model classes are represented as entity class.

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. 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 query method. View does not depend upon application logic. 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 method 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 user's request.

<u>Rules to derive from the Use case</u>:

1. Combination of one actor and one-use case results in one boundary class. Combination of two actor and one-use case results in two boundary class. Combination of three actor and one-use case results in three boundary class.
2. Use case will result in controller class.
3. Each Actor will result in one entity class.

<u>In 3-Tier Architecture</u>:

Model Classes	Customer, Payment, Net-Banking, Card, Cash.
View Classes	Login, View, PaymentOptionView, NetBankingView, BankSelectionView, CredentialView, PaymentAmountView, PaymentConfirmationView, Logout View
Controller Classes	LoginController, PaymentOptionController, NetBankingController, BankSelectionController, CredentialsController, PaymentAmountController, PaymentConfirmationController, LogoutController.

<u>Guidelines to place in 3-Tier Architecture</u>: There are 3 Layers stated as below,

- **Presentation Layer**: This layer is nothing but a user interface. View is inside this layer.
- **Application Layer**: This layer is nothing but business logic. Model and controller are inside this layer.
- **Data Layer**: Classes which are responsible for data access and storage are in this layer. It contains the classes which interacts with the database, Files, and other data sources.

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

<u>Answer</u>:

<u>Waterfall Model</u>: Waterfall model relies on documentation to ensure that the project is well defined and project team is working toward clear goals. Ones that phase has been completed and ones we move to the next phase, we cannot go back to the previous phase to make changes. This model is stable for the projects when the requirements are clear.

BA Contribution in the project is as mentioned below,

Sr. No	Stage	Activities	Resources	Artifacts
1.	Requirement Gathering	Stakeholders are identified. All the requirements are gathered from the stakeholder. BA and Project Manager participates in this phase. After completing this phase, BRD will be generated.	BA, Project Manager.	BRD.
2.	Requirement Analysis	The requirements are analysed to understand the scope of the project. Analysing means the BA will check all the requirements, if he founds conflicting requirements then the BA will talk to the concerned stakeholder to clear it, remove the ambiguous requirements.	BA, Project Manager, Tech Team – Sol Arch, NW Arch, DB Arch.	FS/FRS, SSD, SRS, RTM.
3.	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.	Tech Team – Solu Arch, NW Arch, DB Arch, GUI Designer	HDD, ADD, Solution Document.
4.	Development (Coding)	The Development phase include 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.	Programmers, Developers, Software Engineers.	LDD/CDD, Applications
5.	Testing	In the testing phase, the software is tested to ensure that it meets the requirements and is free from defects. Testers are involved in this phase.	Testers, QA, QC.	Test Documents, Application with less Errors.
6.	Process (Configuration Management)	Here, BA will Contribute with the Project Manager for configuring the documentation according to the need and requirements.	Project Manager	RTM
7.	Deployment & Implementation.	BA, ensures that there is smooth transition from development phase to the production phase. **Implementation** is the final stage of waterfall model. It involves running the code for the very first time in production phase. Release manager handles this phase.	Release Engineers	Solution Document.
8.	Maintenance	Running the code for second me in the production phase is called maintenance. BA, contribute with the support Engineers.	Support Engineers, Software Engineers.	Test Documents.

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

Answer:

Conflict Management: "Conflict Management" refers to the process of identifying, addressing, and resolving disputes or disagreements constructively to maintain a productive and collaborative environment. It involves strategies, techniques, and practices aimed at minimizing the negative impact of conflict while fostering positive outcomes. Types of conflict in the Organization can be mentioned as below,

- **Task Conflict:** Disagreements about tasks, goals, or project direction.
- **Relationship Conflict:** Interpersonal disputes arising from personal differences.
- **Process Conflict:** Issues related to how work is done, including procedures and responsibilities.

There are 5 steps for identifying the conflicts which are stated below,

- **Identifying the Conflict:** To Identify the Problem Statement or the Challenge.
- **Discussing in Detail:** Elaboration of the Conflict with respect to the Problem Statement within the Team Players.
- **Agree with the Root Problem:** Satisfying the Problem Statement with specific solution.
- **Check for every possible solution for the conflict:** Review of the Conflict and Solutions.
- **Negotiate the solution to avoid future conflict:** Solution to the futuristic Problems.

Thomas – Kilmann technique: Is a framework used to address and manage conflict effectively. It identifies five conflict-handling styles based on two dimensions mentioned below,

- **Assertiveness** - The extent to which a person tries to satisfy their own needs.
- **Cooperativeness** - The extent to which a person tries to satisfy the needs of others.

The five conflict handling style technique is mentioned below,

- **Competing (High Assertiveness, Low Cooperativeness)**
  Focus: Winning the conflict at the other party's expense.
  Use When: Quick decisions are needed, or vital issues are at stake.
  Risk: Can damage relationships if overused.

- **Collaborating (High Assertiveness, High Cooperativeness)**
  Focus: Finding a win-win solution that satisfies both parties.
  Use When: Long-term solutions are needed, or the issue is critical.
  Benefit: Builds trust and strengthens relationships.

- **Compromising (Moderate Assertiveness, Moderate Cooperativeness)**
  Focus: Finding a middle ground where both parties give up something.
  Use When: Time is limited, and mutually acceptable solutions are required.
  Risk: Solutions may be temporary or less satisfying.

- **Avoiding (Low Assertiveness, Low Cooperativeness)**
  Focus: Ignoring or sidestepping the conflict.
  Use When: The issue is trivial, or emotions need to cool down.
  Risk: Problems can escalate if avoidance is overused.

- **Accommodating (Low Assertiveness, High Cooperativeness)**
  Focus: Satisfying the other party's needs at your expense.
  Use When: Preserving relationships or showing goodwill is more important.
  Risk: May cause resentment if personal needs are consistently neglected.

Q10. List down the reasons for project failure.

Answer:

Project Failure: Project Failure occurs when a project does not meet its defined objectives, deliverables, or expectations. It happens when the project's outcomes deviate significantly from the original goals in terms of scope, time, cost, or quality.

Reasons are to be followed,

- **Improper Requirement Gathering**: If the requirements of the project are not gathered correctly, then this can lead to project failure.
- **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 to meet the goals.
- **Ineffective or less communication:** If there are communication issues between stakeholders, team members then this can lead to misunderstandings or delays in project or even can lead to project failure.
- **Poor Risk Management**: Poor risk management can also lead to project failure. The team fails to identify the risks and do the risk mitigation, which can lead to unexpected challenges or delays in project. Lack of user involvement. Lack of executive support.
- **Unrealistic Expectation**: Means the goals that cannot be achieved or the goals that are out of scope.
- **Improper planning:** The project can fail if the planning is not done properly. The milestones, goals should be discussed. If there is no proper planning, then team may face difficulties in addressing the issues or to track the progress.
- **Insufficient Resources:** Insufficient resources can also lead to project failure. The project may fail due to lack of technology knowledge or lack of finances.

Q11. List the Challenges faced in projects by BA?

Answer:

Therefore, Challenges faced by the BA are to be followed,

1. Lack of Training.
2. Obtaining signoff on the requirement.
3. Change Management.
4. Co-ordination between developers and testers.
5. Conducting Meetings.
6. Making sure the Status report is effective.
7. Driving clients for UAT completion.
8. Making sure that the project is going on the right track and deliver as per the timelines without any issues.

9. Gathering clear and Unambiguous requirement.
10. Unable to understand what stakeholder is conveying
11. Scope Creep, change in the requirement or the scope of the project during the project lifecycle can lead to scope creep.
12. Managing the stakeholder with conflicting interest, can be a difficult task for BA.
13. Poor communication between stakeholder and BA can affect the process of gathering the information.
14. Technical Complexity.

Q12. Write about Document Naming Standards.

Answer:

Document Naming Standard: A Document Naming Standard is a set of guidelines that define how documents should be named within an organization or project. This standard ensures consistency, clarity, and ease of document retrieval and management.

The use of the Document Naming Standard can be done as follows,

- **Consistency:** Ensures uniform naming across all files.
- **Organization:** Helps categorize and sort files logically.
- **Searchability:** Makes locating documents faster and easier.
- **Version Control:** Tracks revisions and updates effectively.
- **Collaboration:** Enhances team collaboration by reducing confusion.

Key Elements include for the Document Naming Standard can be mentioned as below,

- **Prefix or Identifier:** Indicates the project, department, or file type (e.g., HR_, FIN_, PRJ_).
- **Descriptive Name:** Describes the content or purpose of the document (e.g., Budget Report, Meeting Minutes).
- **Date Format:** Standardized date format, typically YYYY-MM-DD (e.g., 2024-12-17).
- **Version Number:** Indicates revisions (e.g., v1.0, v2.1).
- **Document Status:** Specifies the stage of the document (e.g., DRAFT, FINAL, APPROVED). This can be Optional.
- **Author or Creator:** Initials or names of contributors (e.g., JDoe). This can be Optional.

Here is the Example for the for the Document,

Format: [ProjectID] [Document Type] V[x]D[y]. extension

Example: PQ777FRDV1D1.docx

Q13. What are the Do's and Don'ts of a Business Analyst.

Answer:

Sr. No	Do's	Don'ts
1.	Take time to understand the business objectives and problems.	Do not make assumptions without validating them with stakeholders.
2.	Facilitate clear and effective communication among stakeholders and project teams.	Do not overwhelm stakeholders with excessive technical details or unnecessary information.
3.	Maintain clear, concise, and well-organized documentation (e.g., use cases, user stories).	Do not skip important documentation or skip validating requirements with stakeholders.
4.	Keep the end-user in mind when defining requirements.	Do not ignore non-functional requirements (e.g., performance, security).
5.	Prioritize requirements based on business value and urgency (e.g., MoSCoW).	Do not prioritize incorrectly or neglect important requirements for short-term gains.
6.	Regularly engage with stakeholders to ensure alignment.	Do not work in isolation or avoid difficult conversations about project direction.

Q14. Write the difference between packages and sub-systems

Answer:

Sr. No	Package	Subsystem
1.	A Package can be defined as a logical grouping of related classes or modules.	A subsystem can be defined as a self-contained system or component performing a specific function.
2.	The purpose of the Package is to organizes and manages code structure.	The purpose of the Package is to implements major features or functions.
3.	Abstraction level for the package is Low-level, closer to the code base.	Abstraction level for the subsystem is high-level, representing a functional system.
4.	Packages are dependent on the internal dependencies between classes.	Subsystem are dependent on the managing between different systems or external services.
5.	Visibility of the packages is used within a single system or application.	It may be integrated with the other systems.
6.	Example: Libraries or packages in Python or Java.	Example: Payment Processing, Authentication, Data Storage.

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

Answer:

Camel Casing: Camel Casing is a naming convention used in programming where multiple words are combined into a single identifier without spaces, and each word after the first begins with a capital letter. This convention improves readability and distinguishes words within a name.

Types of Camel Casing are mentioned below,

<u>Lower Camel Case (camelCase)</u>: The first word is in lowercase, and each subsequent word starts with a capital letter.

Usage: Commonly used for variable names, function names, and object properties.

Examples:

- firstName
- totalAmount
- calculateSum

<u>Upper Camel Case (PascalCase)</u>: Every word starts with a capital letter, including the first one.

Usage: Typically used for class names, types, and namespaces.

Examples:

- CustomerDetails
- OrderProcessor
- StudentInfo

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

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

Answer:

Development Server: A Development Server is a local or remote server environment used by developers to build, test, and debug applications during the development process. It simulates a real server but is configured specifically for development needs, offering flexibility and tools that aid in writing and testing code. It provides platform for the developers and the testers to build, test, develop and debug the application.

So, as an access for BA are follows,

- **Read Only**: BA's may be granted with the read only access to the development server. This will allow them to view the user interface of the application, navigate through the features and, they will be able to observe the behaviour 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**- Here, 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: Data Mapping is the process of matching data elements from one data source to corresponding elements in another. It defines how data fields from a source system correspond to fields in a target system, enabling data integration, migration, and transformation. Its key aspects include,

- **Source Data:** The original data location (e.g., a database, file, or API).
- **Target Data:** The destination where the data is moved or transformed.
- **Mapping Rules:** Rules that define how each data element from the source relates to the target, including any transformations, conversions, or calculations.

Different Types of Data Mapping includes,

- **Manual Mapping:** Done manually by developers or data analysts. Best for small or simple projects.
- **Automated Mapping:** Performed using specialized data integration tools. Suitable for complex or large-scale projects.
- **Schema Mapping:** Involves matching entire data structures or schemas between two databases.

Purpose Include,

- **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.

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: Stands for “Application Programming Interface.” It is a set of rules, protocols, and tools that allow different software applications to communicate and interact with each other. It defines how requests are made and responses are received between systems, enabling seamless data exchange and functionality sharing.

The working of the API is based on the below process mentioned,

- **Client Request:** A client (e.g., a web browser or mobile app) sends a request to the API with specific instructions.
- **Server Processing:** The API processes the request, interacts with the underlying database or service, and retrieves the required information.
- **Response:** The API sends back a response, often in formats like JSON or XML, containing the requested data or action result.

Types of API is stated below,

- **Web APIs:** Accessible over the internet using protocols like HTTP/HTTPS. Examples: REST, GraphQL, SOAP.
- **Operating System APIs:** Allow applications to access operating system features (e.g., file management). Example: Windows API.
- **Library/Framework APIs:** Expose functionalities from a software library or framework. Example: NumPy API in Python.
- **Database APIs:** Enable communication between applications and databases. Example: SQL queries through a database driver.

To use the Application Integration, to change the format of the date below steps can be followed,

Step 1: **API Request/Response Identification**

- Identify the API request and response structure.
- Determine where the date field exists and its format in the incoming data.

Step 2: **Data Transformation Logic**

- Implement date format conversion using a programming language or middleware.

Step 3: **Integration Implementation**

- **Data Validation:** Validate the incoming date format using the correct pattern.
- **Data Conversion:** Apply the conversion logic before saving or processing data.
- **Data Storage or Forwarding:** Save the converted date in the application's database or forward it to another service.

Step 4: API Response (if applicable)

- When sending data back to the US application, convert the date from dd-mm-yyyy to mm-dd-yyyy if required.

Step 5: Testing & Error Handling

- **Unit Tests:** Test various date scenarios, including edge cases like leap years and invalid dates.
- **Error Handling:** Implement try-except blocks or API response validation to handle incorrect formats gracefully.