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

**Question 1**. Draw a Use Case Diagram.

**Answer**- A Use Case Diagram is a type of UML diagram that visually represents the interactions between users (actors) and a system. It helps in understanding the system's functionality by showing different use cases (functionalities) and how users interact with them.



**Question2.** Derive Boundary Classes, Controller classes, Entity Classes.

**Answer-** To design the system, we need to classify the *Boundary Classes, Controller Classes, and Entity Classes* based on their roles in software architecture.

**What are these, Classes?**

1. Boundary (View) Classes:
	* These handle user interaction (UI elements, API endpoints, external system communication).
	* They act as intermediaries between users and the system.

Example: Web pages, forms, mobile screens, API request handlers.

1. Controller Classes:
	* These handle application logic and arrange interactions between the UI and entities.
	* Controllers process user requests, validate inputs, and invoke entity operations.

Example: Payment processing logic, order management, authentication handling.

1. Entity (Model) Classes:
	* These represent the core business objects and store data.
	* Entities encapsulate attributes and behaviors related to the system.

Example: Customer details, order data, payment transactions.

Boundary Classes interact with the user (Payment UI, Order UI).

Controller Classes handle business logic (Payment Controller, Order Controller).

Entity Classes store core data (Customer, Order, Payment, Payment Method and its subclasses).



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

**Answer**- In a three-tier architecture, the system is typically divided into:

* Presentation Layer (Boundary layer) – Handles user interactions.
* Business Logic Layer (Controller Layer) – Contains business rules and logic.
* Data Access Layer (Entity Layer) – Manages database interactions.

Class Breakdown for the Payment System-

1**.** *Boundary Classes (Presentation Layer)*

These handle user interactions and UI logic.

* Payment UI – Provides UI for customers to make payments.
* Payment Form Validator – Validates payment inputs before processing.

2**.** *Controller Classes (Business Logic Layer)*

These process requests and coordinate between UI and database.

* Payment Controller – Manages payment requests, validates them, and forwards them to the service layer.
* Payment Service – Implements business rules for processing payments.
	1. *Entity Classes (Data Access Layer)*

These represent the data and handle database operations.

* Payment – Represents a payment transaction with attributes like amount, payment Mode, status, etc.
* Card Payment – Inherits from Payment and contains card-specific attributes.
* Wallet Payment – Inherits from Payment and contains wallet details.
* Cash Payment – Represents cash transactions.
* Net Banking Payment – Inherits from Payment and contains bank transaction details.
* Payment DAO – Handles CRUD operations for the payment data in the database.

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

**Answer**- A Domain Model represents the key entities, their attributes, and the relationships involved in a particular business process. In this case, we are modeling the scenario where a customer makes a Payment through Net Banking.

|  |
| --- |
| Customer |
| PK |  **Customer ID** |
|   | Customer NameContact DetailsAddressAccount Details |

|  |
| --- |
| Bank |
|  |  |
|   | Bank NameLocation Branch code |
|  |

|  |
| --- |
| Payment |
|  |  |
|  | Payment IDAmountPayment DateStatus |

|  |
| --- |
| Account |
|  |  |
|   | Account NoAccount typeAccount holder NameBalance |
|  |

|  |
| --- |
| Net Banking service |
|  |  |
|  | AuthenticationFund TransferTransaction historyAccount management |

|  |
| --- |
| Authentication |
|  |  |
|   | User name PasswordOTP |
|   |

|  |
| --- |
| Transaction  |
|  |  |
|   | Transaction IDReceptionist DetailsAmountTimestamp |

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

**Answer**- A Sequence Diagram is a type of UML (Unified Modeling Language) diagram that helps visualize how different components in a system interact step by step over time. It is used in system design, software development, and business process modeling to understand and document the flow of actions, messages, and decisions between users and system elements.

****

**Question 6**- Explain Conceptual Model for this Case,

**Answer-**

A Conceptual Model is a high-level representation of how different entities interact in a system, focusing on key concepts rather than implementation details. It helps in understanding the system’s structure, relationships, and flow of data before diving into technical details like database schemas or coding.

Key Components of the Conceptual Model

For the Net Banking Payment Process, the conceptual model includes:

1. Entities (Key Participants)

* Customer – Initiates the transaction.
* E-commerce Website – Provides the platform for shopping and payment selection.
* Bank Net Banking System – Handles authentication, payment processing, and transaction validation.
* Bank Database – Stores customer account details and transaction records.
* Merchant – Receives payment confirmation and processes the order.

2. Relationships Between Entities

* Customer interacts with the E-commerce Website to select products and a payment method.
* E-commerce Website communicates with the Bank’s Net Banking System to redirect the customer for login.
* Customer logs into the Bank Net Banking System and provides payment details.
* Bank Net Banking System validates the payment by checking the Bank Database for sufficient funds.
* Bank confirms the payment to the E-commerce Website and the Customer.
* E-commerce Website updates the Merchant, who then processes the order and confirms to the Customer.

Why is a Conceptual Model Important?

✔ Provides a Clear Understanding – Helps stakeholders visualize how the system works.
✔ Defines the Scope – Identifies key entities and interactions without technical complexity.
✔ Helps in System Design – Serves as a foundation before moving to a Logical or Physical Model.
✔ Improves Communication – Allows developers, business analysts, and non-technical teams to align their understanding.

**Question 7-** 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 (Model-View-Controller) is a software design pattern that helps organize code by separating concerns into three parts:

Model → Handles the data and business logic.
View → Manages the user interface (UI) and displays information.
Controller → Acts as a middleman between the Model and View, processing user inputs and updating the system.

This architecture is commonly used in web and application development because it makes applications more organized, maintainable, and scalable.

MVC Components in Detail-

1. Model (M) – Data & Business Logic

* Represents the data of the application.
* Handles database operations, calculations, and rules for business logic.
* Example: In an online shopping system, the "Product" and "User" data are part of the Model.

2. View (V) – User Interface (UI)

* Displays information to the user.
* Receives updates from the Model and presents them in a user-friendly way.
* Example: A webpage that shows the list of products in an e-commerce website.

3. Controller (C) – Request Handler

* Accepts user inputs, processes them, and updates the Model or View accordingly.
* Acts as a bridge between the View and Model.
* Example: When a user clicks "Add to Cart", the Controller updates the Model (Cart) and refreshes the View.

Rules to Derive Classes from a Use Case Diagram (MVC Perspective)

A Use Case Diagram shows interactions between users (actors) and the system.
To derive classes for MVC architecture, follow these steps:

1. Identify Key Elements from the Use Case Diagram

* Actors (Users, Systems) → Helps identify Controller classes.
* Use Cases (Functionalities) → Helps derive Model classes.
* Interactions & Data Flows → Helps define View components.

2. Create Classes Based on These Elements

-Model Classes (Data & Logic)

* Derived from objects in use cases that store data.
* Example: If a use case is "User Registers", the User class is created.

- View Classes (UI Representation)

* Created for user interactions & outputs in the system.
* Example: If a use case involves "Show Product Details", a Product View class is made.

- Controller Classes (Processing User Actions)

* Derived from actions in use cases.
* Example: If a use case is "Process Payment", a Payment Controller class is required.

-Guidelines to Place Classes in a 3-Tier Architecture-

A 3-Tier Architecture further divides MVC components into three layers for better organization:

1️- Presentation Layer (UI / View)

* Includes View classes (HTML, React, Angular, JSP, etc.).
* Displays data to the user and handles user interactions.

2️- Business Logic Layer (Application / Controller)

* Includes Controller classes that process requests.
* Applies business logic (e.g., calculations, workflows, decisions).

3️-Data Layer (Database / Model)

* Includes Model classes that handle database interactions.
* Stores and retrieves data (e.g., using MySQL, MongoDB, etc.).

**Question 8-** Explain BA contributions in project (Waterfall Model – all Stages).

**Answer-** The Waterfall Model is a step-by-step linear approach to software development. Each phase must be fully completed before moving to the next phase. A Business Analyst (BA) plays a crucial role in ensuring project success by gathering requirements, documenting processes, and facilitating communication between stakeholders.

|  |  |  |
| --- | --- | --- |
| **Stage** | **Activities** | **Artifacts & Resources** |
| Pre-Project | -Understand business needs -Conduct feasibility study -Identify stakeholders | Business Analyst, Project Sponsor, Senior Management |
| Planning | -Define project scope -Identify risks- Create project roadmap | Business Analyst, Project Manager, Stakeholders |
| Project Initiation | - Identify key business objectives - Define success criteria - Conduct meetings | Business Analyst, Project Sponsor, Clients, Project Manager |
| Requirements Gathering | - Conduct interviews, surveys, and workshops - Document business needs | Business Analyst, Clients, Stakeholders, SMEs (Subject Matter Experts) |
| Requirements Analysis | - Analyze requirements - Prioritize and validate requirements - Get approvals | Business Analyst, Product Owner, Developers, QA Team |
| Design | - Create wireframes and process flows - Review with stakeholders | Business Analyst, UI/UX Designers, System Architects |
| Testing | - Prepare test cases - Validate requirements through User Acceptance Testing (UAT) | Business Analyst, QA Testers, End Users, Clients |
| UAT (User Acceptance Testing) | - Ensure system meets business needs - Gather feedback from end users | Business Analyst, Clients, End Users, UAT Testers |

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

**Answer-** Conflict management is the process of handling disputes or disagreements between individuals or groups in a way that minimizes negative impact and leads to a positive resolution. Effective conflict management helps maintain healthy relationships, increases productivity, and improves decision-making in both personal and professional settings,

 Conflict management is the process of identifying and resolving disagreements effectively. It begins by recognizing the existence of a conflict and understanding the perspectives of those involved. Once the issue is acknowledged, open discussions take place to gather details and ensure that all parties feel heard. The next step is to analyze the root cause of the conflict rather than focusing on surface-level disagreements. After identifying the main problem, various possible solutions are explored, weighing their advantages and drawbacks. Finally, a mutually acceptable resolution is negotiated and implemented to ensure long-term harmony and prevent similar conflicts in the future.

The Thomas-Kilmann Conflict Mode Instrument (TKI) is a model used to understand how people handle conflicts. It helps individuals choose the best way to resolve disagreements,the Thomas-Kilmann Conflict Mode Instrument (TKI) is a widely used model for understanding how people manage conflicts.

 It is based on two key factors: assertiveness, which is the extent to which a person tries to satisfy their own needs, and cooperativeness, which is the extent to which a person tries to satisfy the needs of others. Depending on how much a person emphasizes these two factors, the TKI model identifies five conflict resolution styles: Competing, Collaborating, Compromising, Avoiding, and Accommodating.

The 5 steps of conflict management from the image are:

1️. identify the conflict – Recognize that a problem exists.
2️. Discuss the details – Understand the viewpoints of all involved parties.
3️. Agree with the root problem – Find the actual cause of the conflict.
4️. Check for every possible solution – Explore different ways to resolve the issue.
5️. Negotiate the solution – Implement a resolution to prevent future conflicts.

Conflicts arise when two or more people have different opinions, needs, or goals. Proper conflict management helps in resolving disagreements peacefully and maintaining healthy relationships in workplaces and personal life.

 Key Takeaways:

 -Conflict is normal but must be handled professionally.
-Good conflict resolution involves understanding, communication, and compromise.
 -Following these 5 steps ensures smoother teamwork and better outcomes.

**Question 10**- List down the reasons for project failure.

**Answer-**

Project failure can occur due to various reasons, often starting with unclear requirements. When the project scope is not properly defined or documented, teams may misunderstand stakeholder expectations, leading to misaligned deliverables and multiple revisions. Poor planning is another major factor, as unrealistic deadlines, missing milestones, and lack of risk assessment can result in inefficiencies and missed objectives**.**

 Unclear Requirements – If project goals and requirements are not properly defined, the team may develop features that don’t match stakeholder expectations. This can lead to multiple revisions, increasing project time and cost.

Poor Planning – A project plan should define clear milestones, realistic deadlines, and assigned responsibilities. If planning is weak, tasks may take longer than expected, and the project can run over budget or fail entirely.

Lack of Stakeholder Engagement – Stakeholders (clients, sponsors, or users) need to provide regular input and approvals. Without their involvement, the project might not meet business needs, leading to rejection or major changes later.

Inadequate Risk Management – Every project faces risks such as budget constraints, resource unavailability, or technical failures. If these are not identified and managed early, they can escalate into bigger problems and cause delays or failure.

Scope Creep – When additional features or changes are introduced without proper evaluation, it increases workload, budget, and timelines. This can put pressure on teams and lead to project failure.

Poor Communication – Miscommunication between teams, stakeholders, and management can result in misunderstandings, duplicate work, and rework. Clear and regular updates help ensure alignment.

Insufficient Resources – A project requires skilled personnel, adequate budget, and proper tools. If any of these are lacking, productivity drops, and quality suffers, leading to an unsuccessful project.

Weak Leadership & Governance – A strong project manager is needed to provide direction, resolve conflicts, and keep the team motivated. Poor leadership can lead to missed deadlines and confusion within the team.

Technology Issues – If the wrong technology is chosen, it can cause integration problems, slow performance, or security risks. This can lead to delays and extra costs for fixing issues.

Failure to Meet Deadlines – Delays in project milestones due to mismanagement, scope creep, or poor coordination can impact the final delivery. This may result in financial losses and loss of customer trust.

**Question 11**- List the Challenges faced in projects for BA.

**Answer-** Challenges Faced by a Business Analyst in Projects

Unclear Requirements – One of the biggest challenges for a BA is dealing with vague, incomplete, or constantly changing requirements. If stakeholders are unsure of what they need, it becomes difficult to document precise business needs, leading to rework and delays.

Managing Stakeholder Expectations – Stakeholders often have different perspectives and priorities, making it challenging to align everyone’s expectations. Some may want fast delivery, while others may focus on quality, creating conflicts that a BA needs to resolve.

Scope Creep – Frequent additions or changes to project requirements without proper evaluation can expand the project scope beyond initial plans. This leads to increased workload, stretched timelines, and resource constraints, making it difficult for the BA to maintain project stability.

Communication Gaps – A BA acts as a bridge between business teams and technical teams. If communication is not clear or frequent, misunderstandings can arise, resulting in incorrect deliverables or delays in decision-making.

Limited Business Knowledge – When working in a new industry or domain, a BA may struggle with understanding business processes, regulations, and market trends. This lack of knowledge can affect requirement gathering and solution design.

Conflicts Among Stakeholders – Different stakeholders may have conflicting needs, priorities, or opinions about the project. A BA must use negotiation and conflict resolution skills to balance these differences and reach a consensus.

Lack of User Engagement – End-users are the ultimate consumers of the system, but if they are not involved in the requirement-gathering process, the final product may not meet their actual needs, leading to low adoption rates.

Technology Constraints – Sometimes, business requirements are difficult to implement due to limitations in existing technology or infrastructure. A BA must work with technical teams to find feasible solutions without compromising business objectives.

Tight Deadlines – Projects often have aggressive timelines, leaving little time for proper requirement gathering and analysis. This pressure can lead to rushed decisions and incomplete requirements, affecting project quality.

Changing Market Conditions – Business priorities and industry trends can shift during the project lifecycle. If a BA does not stay updated with these changes, the final solution may become irrelevant or outdated by the time it is implemented.

**Question 12**- Write about Document Naming Standards

**Answer**-

Document naming standards are a set of guidelines that help ensure consistency, clarity, and organization when naming files and documents in a project or business environment. These standards make it easier to locate, manage, and track documents efficiently, reducing confusion and errors.

A well-structured document name should be clear, descriptive, and follow a consistent format that includes important details like project name, document type, version number, and date. A common naming convention follows a structured pattern such as:

[Project/Department] *[Document Type]* [Description]*[Version]*[Date]

For example:

* BA\_ProjectPlan\_V1\_2025-02-20 (Business Analysis Project Plan, Version 1, created on February 20, 2025)
* HR\_Policy\_LeaveManagement\_V2\_2024-12-15 (HR Leave Policy, Version 2, updated on December 15, 2024)

Best Practices for Document Naming

* Use a Consistent Format – Maintain a uniform naming pattern across all documents to avoid confusion.
* Include a Version Number – Versioning (e.g., V1, V2, V3) helps track updates and changes.
* Use Dates in a Standard Format – Preferably YYYY-MM-DD (e.g., 2025-02-20) to ensure chronological sorting.
* Avoid Special Characters & Spaces – Use underscores (\_) or hyphens (-) instead to prevent system errors.
* Keep Names Short but Meaningful – Avoid overly long names while ensuring clarity.
* Indicate Document Type – Specify whether it is a proposal, report, requirements document, or policy.

**Question 13**- What are the Do’s and Don’ts of a Business analyst.

**Answer-**

|  |  |  |
| --- | --- | --- |
| Sr No. | DO'S | DON’TS |
| 1. | Consult an SME (Subject Matter Expert) for clarifications in requirements. | Never say NO to the client; always provide alternatives. |
| 2. | Go to the client with an open mind and no assumptions. Listen carefully until the client finishes, then ask questions | Never assume anything is "By default"; always confirm requirements. |
| 3. | Try to extract the maximum possible solution details from the client. | Never imagine the system in terms of GUI; focus on functionalities first. |
| 4. | Concentrate on important and prioritized requirements. | Don’t interrupt the client while they explain their problem. |
| 5. | Question the existence of each requirement and validate its necessity. | Never propose a solution immediately based on past experiences and assumptions. |
| 6. | Document every requirement in a structured way and get it validated by stakeholders. | Do not ignore requirement changes; manage them through a proper change control process. |
| 7. | Always confirm requirements with multiple stakeholders to avoid misunderstandings. | Never assume that all stakeholders have the same understanding of the requirement. |
| 8. | Use visual aids like flowcharts and diagrams to clarify complex requirements. | Don’t depend only on verbal communication; misinterpretations can occur. |
| 9. | Ensure all requirements are testable and measurable. | Never finalize requirements without considering their impact on other modules. |
| 10. | Maintain a proper version history of all requirement documents. | Don’t forget to align business needs with technical feasibility. |
| 11. | Engage with end-users to understand real-world challenges and ensure usability. | Don’t assume what end-users need without discussing it with them. |
| 12. | Keep communication transparent between business and technical teams to avoid misalignment. | Never act as a messenger; ensure you understand and analyze the requirements before passing them. |
| 13. | Align requirements with business goals and ensure they provide value. | Don’t document irrelevant or vague requirements that do not contribute to business objectives. |
| 14.  | Conduct requirement walkthroughs with stakeholders to ensure clarity and avoid gaps. | Don’t finalize requirements without stakeholder approval, as misinterpretations can lead to rework. |

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

**Answer**- Packages and Sub-Systems are both used in UML (Unified Modeling Language) to organize and structure complex systems, but they serve different purposes.

A Package is a grouping mechanism that helps in organizing related elements like classes, use cases, or components. It acts as a container to logically group similar functionalities together. Packages do not define system behavior; instead, they help manage complexity by organizing the model into smaller, manageable sections.

On the other hand, a Sub-System is a self-contained, independent module within a larger system that has a specific function. It consists of multiple interacting components, including classes, interfaces, and packages, that work together to achieve a well-defined goal. A sub-system can function independently or as part of a bigger system.

For example, in an e-commerce system, a Package may group together all payment-related classes (like Payment, Transaction, and Invoice), while a Sub-System would be the entire Payment Processing module, which includes interactions with banks, validation mechanisms, and confirmation systems.

The key difference is that Packages are used for organizing elements conceptually, while Sub-Systems represent functional components that contribute to the overall working of the system. Packages help in structuring models, while sub-systems define actual system behavior and interactions.

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

**Answer**- Camel-casing is a naming convention widely used in computer programming to make variable names, function names, and identifiers more readable without using spaces. Instead of separating words with underscores (\_), camel case joins them together while capitalizing the first letter of each new word (except the first word in some cases).

Why is Camel Casing Important?

* It improves readability by making long names easier to understand.
* It follows consistent naming conventions, making it easier for developers to recognize variables and functions.
* It eliminates spaces, which are not allowed in programming language identifiers.

Types of Camel Casing

1. Lower Camel Case (camelCase)
	* The first word starts with a lowercase letter, and all following words start with an uppercase letter.
	* Example: customerDetails, orderNumber, calculateTotalAmount
	* This is commonly used for variables and function names in programming.
2. Upper Camel Case (PascalCase)
	* Every word starts with a capital letter, including the first one.
	* Example: CustomerDetails, OrderNumber, CalculateTotalAmount
	* This is often used for class names and database table names.

Where is Camel Casing Used?

* Variables: userName, accountBalance
* Functions: getCustomerDetails(), calculateTax()
* Class Names: EmployeeDetails, OrderManager
* File Names: customerProfile.json, orderHistory.xml
* Database Columns: productPrice, orderDate

Using camel casing ensures that code is structured, clear, and professional, making it a widely accepted standard in modern programming languages like Java, JavaScript, Python, and C#.

**Question 16-** illustrate Development server and what are the accesses does business analyst has.

**Answer**- A development server is a specialized environment where software applications are built, tested, and modified before being deployed to production. It is mainly used by developers, testers, and business analysts to work on new features, fix bugs, and verify functionality before release.

Key Characteristics of a Development Server:

* Not accessible to end users.
* Allows frequent changes, testing, and debugging.
* May have sample or dummy data instead of real user data.
* Helps in collaboration between developers, testers, and analysts.

Accesses Business Analysts Have in the Development Server-

1. Read-Only Access to Database
	* Business Analysts (BAs) might need access to the database to check if the data structure aligns with business needs.
	* They usually do not have permission to modify or delete data but can view and extract reports.
2. Application Testing Access
	* BAs often have login credentials to the development environment to test workflows, validate requirements, and ensure the system behaves as expected.
	* They may conduct User Acceptance Testing (UAT) before moving changes to production.
3. Requirement Verification & UI/UX Review
	* BAs check if the UI/UX design matches the business requirements.
	* They ensure that the screens, fields, and flows meet user expectations.
4. Access to Logs & Error Reports
	* Some BAs get access to system error logs or audit logs to track failures and report them to developers.
	* This helps in debugging issues and improving system reliability.
5. Limited Configuration Access
	* In some cases, BAs may be allowed to update configurations, such as workflow settings, roles, or permissions, in low-risk areas.
	* However, code-level changes are strictly handled by developers.
6. API & Integration Testing Access (Optional)
	* If the BA is involved in API-based projects, they may have tools like Postman or Swagger to verify API responses.
	* They check if integrations between different systems work properly.
7. Business Intelligence & Reporting Tools
	* BAs may use tools like Power BI, Tableau, or SQL queries to extract reports and analyze system performance.
	* This helps in making data-driven business decisions.

What Business Analysts *Do Not* Have Access To?

 *Production Environment* – BAs usually don’t have access to live production systems where real users interact.

 *Codebase* – BAs do not modify or deploy code; developers handle coding tasks.
 *System Administration Controls* – They cannot change security settings or access configurations that affect infrastructure.

**Question 17**- What is Data Mapping.

**Answer-** Data mapping is an essential process in data management that involves linking fields from one database or system to another. It plays a critical role in ensuring that data is transferred, integrated, or transformed correctly between different platforms. When organizations migrate data from an old system to a new one, they must ensure that data fields in both systems correspond accurately to maintain consistency and usability.

This process is commonly used in data migration, data integration, and ETL (Extract, Transform, Load) operations. For example, when a company upgrades its customer management software, the data stored in the old system needs to be mapped to the new system. If the old system uses the field name "Cust\_Name" and the new system uses "CustomerFullName," a mapping process aligns these two fields so that data is transferred correctly without loss or duplication.

Key Uses-

1. Data Migration: Moving data between old and new systems.
2. Data Integration: Combining data from multiple sources for analysis.
3. ETL (Extract, Transform, Load) Processes: Preparing data for business intelligence and reporting.

Example:
A company moves customer records from an old database to a new system. The "Cust\_Name" field in the old system may be mapped to "CustomerFullName" in the new system to ensure correct data transfer.

Data mapping is crucial for ensuring data accuracy, consistency, and smooth system operations.

**Question 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.

**Answer**- An API (Application Programming Interface) is like a messenger that helps two different applications talk to each other. It defines rules on how data should be sent and received between systems. For example, when you use an app to check the weather, the app requests data from a weather server using an API, and the server sends the weather details back.

API Integration in My Application

In my case, I am working on an online agriculture product store. Suppose my application needs to accept order details from another application in the United States, but there is a problem—our date formats are different.

= My application expects dates in DD-MM-YYYY format.

= The US-based application sends dates in MM-DD-YYYY format.

How to Handle This in API Integration?

When the external application sends data, my API will first convert the date format before saving it into my system. Here’s how it would work:

1. Receive Data: My API will receive order details from the US application, where dates are in MM-DD-YYYY format.
2. Extract Date Information: The API will separate the month, day, and year.
3. Rearrange Format: My system will swap the month and day values to convert it to DD-MM-YYYY format.
4. Save Correctly: After formatting, the API will store the data in my system.

Example

If the US application sends this date:
📩 12-25-2025 (MM-DD-YYYY)

My API will convert it to:
📩 25-12-2025 (DD-MM-YYYY)

Now, the data is correctly formatted and can be used without errors. This process ensures that different applications can communicate smoothly despite using different formats.