**CAPSTONE PROJECT PREP 3 PART 1**

Case Study 1 (Q1-Q6): A customer can make a payment either by Card or by Wallet or by Cash or by Net banking.

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

**Ans:**

* A **Use Case Diagram** is high level diagram, so the main focus of the diagram is to how external interfaces will be interacting with proposed IT systems.
* It can be used to describe the functionality of a system in a horizontal way and UCD represents only in positive flow
* In this diagram, we deal with several elements such as,

Actor - System you are describing interacts with it. Also it’s noun phrase which is living or non-living thing to perform a task

Primary Actor – Who directly interacts and initiates with all process within the system

Secondary Actor – Who are those actors will supports and contributes the system to fulfil the primary actor.

Use Case- Which the system know how to perform, which is an individual functionality.

System – Itself, the rectangular box.

Here is the use case diagram for the case study,

 

1. **Derive Boundary Classes, Controller classes, Entity Classes?**

**Ans:**

* 1. **Boundary Classes:**
* It is the type of classes which is also known as **FORM Class** who acts the interface between the system and the external factors such as users, mobile/laptop or external systems and they handle interaction such as user input or response from other APIs
* These classes are used to validate and transfer the data from the system without encouraging the business logic layer.
* In our case study, these classes handles payment interface between customers and servers to make payments. So the classes communicates with external payment services such as wallet, card etc. and captures payment details data entered by the customer and sends payment requests to the controller class.
* **Example:** Login of Customer registration or bank server such as CustLogin Boundary ServerLogin Boundary
	1. **Controller Classes:**
* It is the type of classes which is also known as **Transient Class** , who manages the system’ s workflow (incoming process requests), validation and logic (handles user input and updates model) of the process, and have interaction between boundary classes and entity classes.
* **Example:** In payment method, the controller validates payment request and call the respective entity class for transaction creation and finally routes the request to appropriate payment gateway. Termed such as Login Controller, Payment controller.
	1. **Entity Classes:**
* It is type of classes also known as **Database classes or Persistent Class** which represents the object data in system and they encapsulate data and business logic often corresponding to tangible objects in real world. It used to store and manipulate data related to that entity.
* **Example:** In payment method. Store business data such as customer details and payment transactions. Each actor will be considered as entity such as customer, cash, card etc.

1. **Place these classes on a three tier Architecture?**

 **Ans:** A **Three tier architecture** is the software design architecture which organises the applications into three layers which helps to improve security, scalability and maintenance such as,

* 1. **Presentation/Application Layer (UI Layer) :** Handles user Interactions specially Front End Developers
	2. **Business Logic Layer :** Manages and handles workflows, business logic and processing
	3. **Data Layer (DB layer):** Manages data retrieval and storage.

|  |  |  |
| --- | --- | --- |
|  **Application Layer** |  **Business Logic Layer** |  **Data Base Layer** |
|   PaymentBoundary CustLoginBoundary ServerBoundary CashBoundary CardBoundary NetBankingBoundary BankBoundary  |   PaymentController CustLoginController ServerController CashController CardController NetBankingController BankController  |  Customer Server Cash Card Net banking Bank |

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

 **Ans:** A **Domain Model** is the conceptual representation of the system which mainly focuses on business logic and how entities interact with each other to achieve a particular use cases which includes main entities or objects, attributes, their relationship and behaviours. It is typically visualized using UML class diagram and bridges gap between business requirements and technical Implementation.

**Example: Customer Making Payment through Net Banking**

 **Entities:** Represent Key object within domain.Customer, Payment, Transaction, Bank, Net Banking

 **Relationships:** Interactions between the Entities. A Customer makes a Payment and the Transaction is processed by Bank.

 **Attributes:** Columns or Properties of Particular entities. Ex: Name, Email ID, Address



### **In this diagram, Customer initiates the payment with its PayID, select the PayMethod, enters the Amount**

### **Customer choose the Netbanking options, enters AccID, TrasnAmt, AccDet**

### **Bank triggers the customer AccNo, AccName, AccType, Balance with their appropriate BankName, enters IFSC, Location from payment gateway**

### **Later, Payment Authentication validates the Username, Password and OTP to generate the payment.**

### **Transaction delivers and updates the payment status and stores in the payment history of the customer.**

### Draw a sequence diagram for payment done by Customer Net Banking?

### Ans:

### A Sequence Diagram is used to design, document and validate the architecture, interface and system logic by describing the sequence of actions that need to perform to complete a task as it show the interactions between classes in the sequential order.

### UML sequence Diagram are used to represent the flow the messages, events and actions between classes or components of the system. It can be explained to draw in many variation by using objects or actors instead of classes and single dimension times space where there is no need to model same classes/objects multiple times.

### In this diagram, Time is represented in vertical direction and the sequence of interactions of the header elements, which are represented in horizontal direction.

### The sequence diagram which will be discussed and formed by the classes that are discovered from MVC architecture and then mapped to 3 Tier architecture.

### Here is the sequence diagram for payment done by customer net banking

###

1. **Explain Conceptual Model for this Case?**

**Ans:** A **Conceptual model** is a high level design or representation of real world concept without focusing on technical implementation. It helps the stakeholders including BA and developers to understand easily and also in defining the relationship between key entities before moving to system design.

Here the **Key Entities** are **Customer, Transaction, Netbanking, and Bank**

**Here are the conceptual model** for the customer making payment through net banking

* 1. **Customer –** This objects represents the payment services users who initiate the payment using secure credentials and logs to choose a payment method such as net banking.
	2. **Data Privacy &awareness –** Customers should awareness for the online payment service and to protect their data confidentially.
	3. **Authentication –** System verifies the user credentials and performs additional security by sending the OTP to their respective Email/ph no
	4. **Payment Transaction Details –** Customer will add their info about the account name, id, amount and purpose of description.
	5. **Bank Trust and support-** Bank are the responsible for net banking services also should adapt the security policies to ensure customer data and transition will flow smoothly and trust on the bank service and customer will aware the net banking technology
	6. **Authorization –** The payment gateway initiates the payment through Netbanking and validate the customer details, and performs additional authenticate services by sending the confirmation of payment through OTP. Additionally it communicate to bank for request the money to ensure customer have sufficient amount
	7. **Transaction Process & Records –** Once the valid OTP enter by the customer, the gateway authorise the payment and debit the amount from the customer account while the amount transfer to other account which reflects and updated status to bank server, customer and the merchant. Both the customer and merchant will have the transaction history of the payment and receives notification through the sms/mail
1. **What is MVC architecture? Explain MVC rules to derive classes from use case diagram and guidelines to place classes in 3-tier architecture?**

 **Ans:** A **MVC architecture** elaborates with **Model-View-Controller** which is a software design pattern that organises an application logic into three interconnected components.

* 1. **Model:**
* Model class represents the data of an application and shows the enterprise data and performs task like managing the data, or data validation, implementing the business rules that govern access to and updates of this data.
* This model class also called as **Entity Class** or **Database Classes** or **Persistent Class.**
* It knows about all the data that need to be displaced, and responds to the controller request whether to add or retrieve the data from the database, also who is aware about the all operations that can be applied to transform that class.
	1. **View:**
* View represents the presentation of the application. It refers to the model as it uses the query methods of the model to obtain the contents and renders it. So it receives input from the user and passes to the controller for processing and later it takes the data from the model and renders it to user interaction
* It remains same if there is any modification in the business logic as it not dependent on the application logic.
* It class of data which is required by the query which is also known as **Boundary Class** or **Form Class**
	1. **Controller:**
* It as an intermediate between model and view class which is responsible for intercepting the requests from view when the user sends a request for something and passes to the model for the appropriate action.
* After the action has been taken on the data, the controller is responsible for directing the appropriate view to the user.
* These class is working based on the user’s command by understanding the command or request given by user through boundary or Form class.
* This Class is also called as **Transient Class.**
	1. **MVC Architecture Rules to derive classes from use case diagram:**
* Primary actor is considered to be only one in use cases.
* Combination of one actor and an use case results in one boundary class
* Combination of two actor and an use case results in two boundary class
* Combination of three actor and an use case results in three boundary class and so on
* Use case will result in an controller class and each Actor will result in one entity class
	1. **Guidelines to place classes in 3 Tier Architecture:**
* Place all Entity classes in Database Layer like customer, employee, cash, card, etc
* Place primary actor associated boundary class in application layer like CustLogin boundary
* Place controller class in application layer like Login controller
* If governing body influence or reusability is there with any of remaining boundary classes, place them in business logic layer else place them in application layer.

**Skills: HTML, CSS, JavaScript**

**Application Layer (Front End)**

**Business Logic Layer Business Logic Layer**

**Skills: Java, C,C++,python**

**Data Base Layer (Back End)**

**Skills: MySQL, mangodb, oracle**

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

 **Ans:** A **Business Analyst (BA)** plays a role around 12 to 16% in whole team project. From pre project initiation to post deployment maintenance we have a role in every phase of the project.

Especially, in waterfall model, it called as linear sequential life cycle model where each stage must be complete fully and mover to another stage of life cycle. They are different stages of the SDLC which are Requirement Gathering and Analysis, Design, Development, Testing, Deployment and Implementation.

 .

|  |  |  |
| --- | --- | --- |
|  **Stages** |  **Activities** |  **Artifacts & Resources** |
|  **Pre Project Initiation** | * To identify business problems, goals and opportunities.
* Conduct enterprise analysis such as Feasibility study, SWOT, Gap, root cause analysis
* Perform Enterprise architectural frameworks, project scope and business case writing, risk analysis.
* Collaborate with stakeholders to define project objectives
 | * Business Case Document
* Feasibility Study Report
* Cost Benefit Analysis Report
* SOW, Purchase order
* Project Charter
* **Resource:** Sr.BA, Business Architects
 |
|  **Requirements Gathering &Analysis** | * Stakeholders Identify by conduct stakeholder interviews, workshop and surveys
* Gather ,and prepare business and functional requirement like BRD by interacting with client by performing brainstorming, document analysis
* Prototyping can be used by BA by define use case, business rules and process flows to make the client to give more specific Requirements.
* Ensure and sort the gathered requirements by avoiding duplicate reqs, grouping into similar functionality
* Prioritize the requirements by MOSCOW technique and Validate the requirements by FURPS technique
* All Architects come with technical requirements (SSD)
* SRS will have functional requirements and technical requirements
* Takes sign off on SRS from client. SRS is the legal bond doc between business and technical team
* BA Prepared RTM from SRS before design phase
* BA traces how requirements are dealt in each phase of development life cycle design till UAT
 | * BRD
* FRS
* Use case diagrams
* Process flow diagrams
* Solution requirements Specification (SRS)
* RTM
* SSD (Supplementary Support Document)
* **Resource:** BA,PM,(DB,NW, Solution)
 |
|  **Design** | * From use case diagram, prepare the test case. Work with these teams to design wireframes and mockups
* From use cases diagram, Solution architect will recommend architect of the IT solution
* DB architect uses entity class and comes up with ER diagram or DB schema
* Front end developers will look into boundary classes and design all possible screens.
* Communicates with client on the design and solution documents for status update to client.
* BA Initiate the preparation of End user manual and update with RTM
 | * Solution Document
* Wireframes &mockups.
* Data flow diagram
* ER diagram,
* Design document (HDD,ADD)
* System architecture document
* **Resource:** BA,PM,(DB,NW, Solution),GUI designer, Test manager
 |
|  **Development (Code)** | * BA Organize JAD sessions and clarify queries on requirements, diagrams and controller classes for technical team during coding
* Act as liaison between business and development teams
* Monitor scope creep and ensure adherence to project scope.
* Conducts regular status meetings with technical team and client and tuning client for participation for UAT
* Update End user manual and RTM
 | * RTM
* Change Request Document if any
* LDD
* CDD application
* **Resource:** Development team, BA ,PM
 |
|  **Testing** | * BA Prepares test cases from use cases or assist test manager to work based on business requirements
* BA performs high level testing and prepares client for UAT
* Test data is requested by BA from client
* Update End user manuals and RTM
* Identify and report defects ensure resolution, later take signoff from client on client project acceptance form
 | * Test cases and test scenarios document
* UAT plan and report
* Defects logs and report
* **Resource:** Testing Team ,BA,PM, client
 |
| **Deployment &Post Implementation** | * Forward RTM to client or PM with attachment of project closure document
* Assist in training end users and support teams
* Coordinates to complete and share end user manuals
* Ensure smooth transition and go live readiness
* Suggest enhancements for system optimization
 | * User training manuals
* Release notes and Deployment checklist
* Enhancement Requests & Change Logs
* **Resource:** PM, Client, BA
 |

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

**Ans:** A **Conflict Management** is the management process of identifying, addressing and resolving unhealthy arguments that arise due to difference in opinions, interests, values or priorities in workplace or personal setting between two individuals or group to promote learning and growth to achieve a positive outcome.

A **Thomas Kilmann Technique** helps to recognise the approach of conflict management in order to help to reduce negative effects and to promote positive growth so that it ensures improved teamwork and collaboration, productivity increase and reduced stress or tension.

This Techniques is kind of framework that establishes the relationship between **Assertiveness** (Which satisfy your own needs) and **Cooperativeness (**To extent which satisfy other needs). There are five factors which describes the relationship between them,

* 1. **Collaboration:**
* It is kind of **High – High** situation and it can be occurred both groups concern are important in order to build long term relationships during creative problem solving is needed.
* **Example:** Both technical team and PM agree on some deadline that balance product quality of work and speed.
	1. **Competing:**
* It is kind of **High – Low** situation and it can be occurred when we want to get quick decisions in case of critical situations or emergencies, so it enforces some rules or policies.
* **Example:** The product manager forces the team to complete without compromising.
	1. **Avoiding:**

* It is kind of **Low- Low** situation, it can be occurred when the issue is temporary due to high emotions and we need to have relax minded since it requires lot of information.
* **Example:** Due to miscommunication from client or avoid the issue.
	1. **Accommodating:**
* It is kind of  **Low – High** situation, it can be occurred when we consider the issue is very important to other people. When we want to have good relationship other people by maintaining harmony which is more important than winning.
* **Example:** Product manager fixed the short deadline ensure to risk with quality.
	1. **Compromising:**
* It is kind of **Medium – Medium** Situation, it can be occurred when we consider the quick, mutually acceptable solution is needed, when both parties have equal importance whose preserving good relationship between them.
* **Example:** Deadline can be extended, with few features were included.

**HIGH**

**COMPETING**

**COLLABORATING**

ASSERTIVENESS

**COMPROMISING**

COOPERATIVENESS

**ACCOMODATING**

**AVOIDING**

**LOW**

**HIGH**

1. **List down the reasons for project failure?**

 **Ans:** Here, are the reasons for project failure due to following reasons,

* 1. **Improper Requirement Gathering:**
* Due to unclear or incomplete requirement gathering and it will occur because of stakeholder will have different interpretation of the project goals
* **Example:** when delivery app company, has forget to interpret the requirements regarding the real time delivery track of the system.
	1. **Continuous Change in Requirements:**
* It occurs due to frequent changes done by business users to keep changing the requirements to alter project scope after the project work initiated, due to which it may causes scope creep leading to increased costs and resource overload.
* **Example:** The delivery app initiate the payment to support only for cash mode only but later it demands integration with other payment methods

* 1. **Lack of User Involvement:**
* Here, end users were not involved during requirement gathering or testing stage. So it cause the final product that may not meet actual business needs and causes expensive rework after implementation.
* **Example:** A delivery app end users were not involved during the testing stage for the app and they finds lack of load time during usage of app.
	1. **Lack of Executive Support:**
* It happens due to lack of budget, resources and decision making authority where no clear with strategic alignment with business goals. It cause struggle to get approvals or priority, no escalation path for resolving major roadblocks.
* **Example:** A CRM system, were does not allocate necessary resources.
	1. **Unrealistic Expectations:**
* When Project deadlines and budges are too optimistic and impractical and stakeholders underestimate the complexity of task which cause team experience stress and leads to poor quality of needs.
* **Example:** When a delivery app customer want to develop an online platform within 3 months but the requirement gathering will take around 2 months.
	1. **Improper Planning:**
* It can occurs due to lack of detailed project roadmap with less estimation of resources, costs and risk. Which causes budget overruns due to poor forecasting
* **Example:** A delivery system lack a payment gateway integration, halfway through the team realises the app request to order for the product without the payment which is not planned earlier
1. **List the Challenges faced in projects for BA**

 **Ans:**

* 1. **Lack of Training:** Inadequate training leads to inefficiency, errors and poor adoption of new system.
	2. **Obtaining Sign off Requirements:**  Delays in approval causes project hold ups and frequent changes leads to rework
	3. **Change Management –** with respect to cost and timelines : Frequent scope changes increase cost and delay project delivery
	4. **Coordination between developers and testers:** Poor collaboration results in late bug detection and rework
	5. **Conducting Meeting:** Unstructured meetings waste time and leads to miscommunication
	6. **Making sure status reporting is effective:** Poorly structured reports cause misalignment and ineffective decision making
	7. **Driving client for UAT completion –** Delays in UAT impacts go live timelines and software quality
	8. **People Management (**Coordinating with different people and different teams**):** Misalignment among teams creates conflicts and project delays
	9. **Overall making sure project health is in good shape and delivered as per the time lines without any issues:** Poor risk management and resource planning result in missed deadlines and cost overruns
1. **Write about Document Naming Standards?**

 **Ans:** The **Document Naming Standards** is a well-established naming convention represents an optimised way to name files and documents to ensure easy retrieval, clarity and enterprises. Also it helps team locate files efficiently, maintain version control and prevent duplication.

There are some of the rules along with elements specified for naming the document standards,

1. A unique name or code to represent the project (Proj\_ABC)
2. Specifies the nature of the document or the document type (BRD, FRD, SOW)
3. Keep the version control by showing the document version (V1.2.1)
4. Creation or modification date with Specified format ( YYYYMMDD)
5. Use Creator name or department team (Michael, Sales)
6. Defines document sensitivity or confidentially level of the document ( Public, Internal)
7. Follow a predefined structure for all files
8. Use Underscores ( ) or hyphens (-) instead of spaces.
9. Keep name short yet meaningful
10. Use \_DRAFT and \_FINAL tags to separate draft and final versions

**Format of Document Naming Standards:**

[Project ID]\_[Document Type]\_[Date]\_[Version]\_[Dept/author].ext.

 **Example:** PQ\_7394BRD\_20250120\_V2\_sales.ext

**Benefits of DNS:**

* It is more efficiency by simplifying the document search or faster up the document retrieval
* Ensures team collaboration by identifying the documents quickly which reduce ambiguity.
* Supports for large scale project documentations
1. **What are the Do’s and Don’ts of a Business analyst?**

 **Ans:**

* Never say No to client – Whenever client request for some requirements we don’t have authority to say No in decision making instead we will discuss with team
* There is No word called as “BY DEFAULT” – We should not assume anything on our own before discussing with the client or team
* Never Imagine anything in terms of GUI
* Question the existence of existence, question everything in the world

Let’s take some of the examples,

|  |  |  |
| --- | --- | --- |
|  **Function** |  **Do’s** |  **Don’t** |
|  **Stakeholder Management** | Ensure stakeholder alignment and regular communication | Don’t neglect stakeholder delay feedback collection |
| **Collaboration** | Work closely with technical and business teams | Don’t work in isolated or overlook dependencies between teams |
|  **Project Risk Management** | List out Potential risk early and document them | Don’t ignore risk or wait until issues escalate |
| **Domain Knowledge** | Continuously improve your understanding of the industry and business processes | Don’t rely only on IT knowledge without understanding business needs |
| **Requirement Gathering** | Ask the right questions, and involve stakeholder early | Don’t assume requirements or ignore stakeholders |

1. **Write the difference between packages and sub-systems?**

**Ans:**

|  |  |
| --- | --- |
|  **Packages** |  **Sub Systems** |
| It is collection of logical grouping of related classes, int grouped together for making the code base more maintainable and easier to navigate for better organization and modularity within a system  | It is distinct parts of larger system and can often function semi independently. Which has its own specific functionality and responsibilities. |
| Operates within the single application organizingcodes which share common functionality  | Operates at a higher level of abstraction with multiple packages or components |
| It usually contains classes, functions or interfaces | It usually contains multiple packages or components that work together |
| Interacts with other packages within the same application , which are not reusable in nature | Interacts with other sub systems to form a solution, which are reusable in in nature |
| Handles a specific aspect or feature within amodule like I/O | Handles a broader area of functionality like data processing or business logic. |
| **Ex:** Logging, string manipulation containing utility functions. Java.io collection of input and output operations. | **Ex:** it is responsible for user authentication in Web application. Also it includes all modules of payment processing. |

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

 **Ans:**

* A **Camel Casing** is a text formatting style where compound words are joined together and entire first word in lowercase and subsequent words first letter should be upper case letter.
* It is commonly used in programming for naming variable, functions, classes and methods to enhance readability without using spaces or underscores.
* Also, it practices in some of the programming languages like Python, JavaScript and Java.
* We have two different of camel casing exists,

**LowerCamelCase:** Where the first letter in lowercase and other words with uppercase.

 **Ex:** lastName, selectUserDetails()

**UpperCamelCase:** Where every word start with an uppercase letter

 **Ex:** LastName, SelectUserDetails()

* Let us Consider these camel casing other usage areas ,

|  |  |
| --- | --- |
|  Class Name | It is written UpperCamelCase Ex: EmployeeDetails() |
|  Function Name | selectUserInfo(),getPassword() |
|  Object Properties | turnRightAndThenSecondLeft(); |
|  Variable Name | userBio, sumPrice |

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

 **Ans:**

* A **Development Server** is a dedicated environment used to developers to build, initial test of new features and modify applications before they are deployed to production.
* It provides a controlled space where code changes can be written, debugged and reviewed without affecting live users.
* It includes a copy of the application, a mock test database, and necessary dependencies to simulate real word conditions while ensuring a smooth development process.
* In this server, **Developers, Product Manager and Business analyst** will get a access but this server enhance less security as compared to production server.
* **As a BA we have access for limited area,** which are based
1. We have limited access to UAT servers as to validate features that confirm business needs.
2. We can participate in sprint demos and review meeting during which to facilitate and review new features to provide feedback
3. We have access to reporting tools like JIRA or Trello to track project progress and report bugs or issues.
4. We have access to project requirements and specs to ensure the development aligns with requirements and clarify them
5. We can use tools like slack or confluence for development team communication and collaborate with stakeholders

**Source Code**

 **Data Store**

**Logs & Review**

**Mock Services**

**a**

**Web app**

**Web app**

**Web app**

**API’s (Test version**

**RUNTIME ENVIRONMENT**

 **RUNTIME** **RUNTIME**

1. **What is Data Mapping ?**

 **Ans:**

* A **Data Mapping** is the process of linking data fields from a source system to corresponding fields in a target system to ensure accurate data transfer, transformation or integration.
* It helps to maintain a data consistency and usability by aligning different formats, structures.
* **Types of Data Mapping:**
1. **Direct Mapping –** It maps one to one where fields from source match to destination. Eg: first\_name = first\_name
2. **Transformation Mapping -**  Before mapping, data to be modified for format changes or concatenation eg: jone\_micheal + raj = jone\_micheal\_raj
3. **Multi source mapping -** Data comes from different sources and combined to destination. Eg: cust\_id + prod\_id = ord\_info
4. **Lookup Mapping -** User reference data to transform values eg: state\_code(KE) = state\_name (Kerala)
* This mapping used in different concepts such as,
1. **Data Integration :** combines data from multiple sources to single system such as integrating of CRM and ERP data
2. **ETL Processes:** Extracting, transforming and loading data into a data warehouse. Such as loading sales data into an analytics
3. **API data Exchange:** Mapping fields between different API’s for communication which used for connecting an ecommerce platform to a payment gateway
4. **Database Migration:** Moving data from one database to another while keeping consistency, transferring data from MySQL to PostgreSQL.
* Challenges outcomes:Handling large datasets efficiently without performance issue.
1. **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?**

**Ans:**

* An **API – Application Programming Interface** which is system that allows different application to communicate with each other by exchanging data. It acts a connector that enables software application to send and receive information in structured way
* An **API Integration** is the process of connecting two or more application using API’s to enable seamless data exchange, which ensures that information flows smoothly between systems without manual intervention making process more efficient.
* **Usage of API’s:**
1. It ensure data consistency across different applications.
2. It reduces manual work by automating data conversions
3. It enhances user experience by maintaining accuracy
4. It improves system compatibility by standardizing data formats
5. It prevents incorrect data storage due to mismatched formats
* **Based on Case study on data format:**

Since our application stores data in the format DD-MM-YYYY but we got US based application who sends data in MM-DD-YYYY so this may cause confusion or errors. We can convert the required date to our format before storing it by following the steps.

1. **API Request –** Implement the API calls within your application to send the data to this conversion endpoint before storing it.
2. **Date Formatting –** Use formatting functions from the data libraries from date strings to convert from one format to other format.

 **Eg:** YYYY-MM-DD to DD-MM-YYYY

1. **Data Parsing –** It involves interpreting and extracting meaningful information from date string. By using parsing functionalities to interpret date string and extract components such as DD-MM-YYYY into DD- date, MM- month, YYYY- year
2. **Data Validation & API Response:** It ensure the converted data info meets specified criteria by ensuring zero errors. Once validated dates are converted ensure that they correspond to calendar dates. ApI response to send data for another system request if further