COEPD – Prep Exam 3 –Part 1/2

**DINESH KULKARNI**

Case Study 1 (Q1-Q6 24 Marks)

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 - 4 Marks Answer

***Answer***

**Use case diagram: -**A use case diagram is visual representation in the unified modelling language UML that depicts how a system interacts with actors in various scenarios

**Actor: -**

1.Primary Actor: - Customer

2.Secondary Actor: -Server

**Use cases: -**

1. Payment initiated
2. View Payment option

1.By Card

2.By Wallet

3.BY Cash

4.By Net banking



 **Q2.** Derive Boundary Classes, Controller classes, Entity Classes. - 4 Marks

***Answer***

We can use MVC (Model-View-Controller) used in UML and object-oriented analysis for deriving the below classes

**1.Boundary classes (View / UI Layer)**

These represents the interface between the system and external world (users or other system). They are often associated with UI or input / output interaction.

 1.Paymentoptionboundary

 2.Cardpaymentboundary

 3.Walletpaymentboundary

 4.Cashpaymentboundary

 5.Netbankingpaymentboundary

**2.Controller classes (Logic layer)**

These manage the flow of data between boundary and entity classes, handling user action and controlling system operations

 1.Paymentinitiatedcontroller

 2.Cardpaymentcontroller

 3.Wallerpaymentcontroller

 4.Cashpaymentcontroller

 5.Netbankingpaymentcontroller

**3.Entity classes (Model / Data layer)**

These represents business / domain data logic typically store in database

 1.Customer-Stores customer data like customer ID, Customer name

 2.Payment-General payment data like amount, date, status

 3.Card-Stores card information like card number, CVV, expiry

 4.Wallet-Holds wallet balance and transaction history

 5.Server-To make transaction

 6.Bankaccount-Netbanking information

 7.Transaction-Record of each payment

 **Q3.** Place these classes on a three tier Architecture. - 4 Marks

***Answer***

3 tier architecture separates the system in to three logical layers

1.Presentation Layer-User interface layer UI

2.Application / Business logic Layer-Controls and Process the logic

3.Data Layer-Manages the data usually database

|  |
| --- |
| Application Layer / Presentation Layer / User Interface |
| 1.Paymentoptionboundary2.Cardpaymentboundary3.Walletpaymentboundary4.Cashpaymentboundary5.Netbankingpaymentboundary |
| Business Logic Layer  |
| 1.Paymentinitiatedcontroller2.Cardpaymentcontroller3.Wallerpaymentcontroller4.Cashpaymentcontroller5.Netbankingpaymentcontroller |
| Data Layer |
| 1.Customer-Entity Class 2.Payment- Entity Class3.Card- Entity Class4.Wallet- Entity Class5.Server- Entity Class6.Bankaccount- Entity Class7.Transaction- Entity Class |

 **Q4.** Explain Domain Model for Customer making payment through Net Banking - 4 Marks

***Answer***

Domain model is like entity relationship diagram. It is a visual representation between tables, components in database. It includes classes, attributes, and its relationships

**Importance of Domain model**

1.Bridges the gap between business stakeholders and developers

2.Ensures common understanding of business rules

3.Assists in designing the database schema

4.Supports API design, UI logic, and backend development

|  |  |  |
| --- | --- | --- |
| Aspect | Domain Model | ER Diagram |
| Purpose | Represents real-world business concepts and relationships | Represents data structure and how data entities are related |
| Focus Area | Focuses on business logic, objects, and behaviour | Focuses on database design and data relationships |
| Used By | Business Analysts, Developers | Database Designers, Developers |
| Includes | Entities, attributes, methods, and relationships | Entities, attributes, and primary/foreign keys |
| Level of Detail | High-level, conceptual | Logical or physical data structure |

**1. Domain Model (Conceptual View – Business Level)**

The Domain Model represents real-world objects and their interactions involved in the payment process. It helps stakeholders and developers understand the business logic.

 **Key Entities:**

1.Customer: Has attributes like CustomerID, Name, Email

2.Order: Linked to Customer; attributes like OrderID, OrderAmount, OrderDate

3.Payment: Linked to Order; attributes like PaymentID, PaymentMode, PaymentStatus

4.NetBankingDetails: Linked to Payment; includes BankName, AccountNumber, IFSCCode, TransactionID

**Relationships:**

1.A Customer places multiple Orders

2.Each Order has one Payment

3.If payment mode is Net Banking, then NetBankingDetails are used

 **Purpose:** Helps define how objects interact in the business process of payment.

**2. ER Diagram (Technical View – Data Level)**

The ER Diagram is used for database design, showing how tables relate in a structured format.

**Entities and Attributes:**

1.Customer (CustomerID [PK], Name, Email)

2.Order (OrderID [PK], CustomerID [FK], OrderDate, Amount)

3.Payment (PaymentID [PK], OrderID [FK], Mode, Status)

4.NetBankingDetails (TransactionID [PK], PaymentID [FK], BankName, AccountNumber, IFSCCode)

**Relationships:**

1.One-to-Many: Customer → Order

2.One-to-One: Order → Payment

3.One-to-One: Payment → NetBankingDetails

**Purpose:** Helps in building the actual database structure to store data efficiently.



 **Q5.** Draw a sequence diagram for payment done by Customer Net Banking - 4 Marks

***Answer***

**Sequence Diagram: -**It is UML diagram and it shows how object interact over time in a specific scenario of a system

1.It focuses on time-order of messages between objects

2.Shows how and in what order objects or components in the system communicate with each other

3.Represents the flow of a use case or functionality step by step

**Main components**

1.Object: -Participants in the interaction

2.Lifeline: -Vertical dash line showing object’s presence during the interaction

3.Messages: -Arrow showing communication between lifelines

4.Activation bars: -Narrow rectangles on lifelines showing when the object is active



 Q6. Explain Conceptual Model for this Case - 4 Marks

***Answer***

A conceptual model for the payment process done by customer using net banking provides high level understanding of the key concept and their relationship involved in the payment transaction.

It helps in visualizing the overall structure and flow of payment process.

**A conceptual model is high-level, real-world view of the system.**

It identifies and describes the main object class, their attributes, and their relationships without getting into the implementation details.

Think of it as a blueprint of business concepts involved in the use case. It is used to understand and communicate what the system is about at a domain level.

Conceptual model

Concepts classes

|  |  |
| --- | --- |
| Class  | Description |
| Customer | The person makes the payment |
| Payment | Represents the act of making payment |
| Net Banking | Mode of payment selected by customer |
| Bank account | Customer’s account use for net banking  |
| Transaction | Records the detail of completed payment |

Relationships

1.A customer makes a payment

2.A payment is made through net banking

3.Net banking uses bank account

4.A payment results in transaction



All possibilities of doing these transactions frequency, Volumes, Value, Geographical distribution entire information together called conceptual model

|  |  |  |  |
| --- | --- | --- | --- |
| Aspect | Domain Model | ER Diagram | Conceptual Model |
| Definition | Represents real-world business objects and interactions | Represents entities, attributes, and relationships in a DB | High-level view of system structure and business requirements |
| Purpose | Understand business logic and object relationships | Design the database schema | Define what the system should do, not how |
| Focus Area | Business objects, behaviour, and associations | Data storage, primary/foreign keys, and cardinality | System’s scope, actors, high-level requirements |
| Used By | Business Analysts, Developers | Database Designers, Developers | Stakeholders, BAs, Architects |
| Includes | Classes, attributes, methods, relationships | Entities, attributes, keys, relationships | Actors, processes, data flow, requirements |
| Used By | Business Analysts, Developers | Database Designers, Developers | Stakeholders, BAs, Architects |
| Level of Detail | Mid-level (business logic) | Detailed (data-centric) | High-level (abstract/system overview) |
| Example Entities | Customer, Order, Payment | CUSTOMER table, ORDER table, foreign keys | Actor: Customer, Use Case: Make Payment |
| Output Used For | Object design, business logic discussions | Database schema implementation | Project scoping and requirement gathering |

**Q7.** What is MVC architecture? Explain MVC rules to derive classes from use case diagram and guidelines to place classes in 3-tier architecture - 8 Marks

***Answer***

**MVC stands for Model-View-Controller**

It is a software design pattern that separates an application into three interconnected components

|  |  |
| --- | --- |
| Component | Description |
| Model  | Represents data and business logic |
| View  | Handles the user interface and presentation |
| Controller  | Manages user inputs and updates the model and view accordingly |

MVC promotes separation of concerns, Making system more maintainable, testable, and scalable

**Rules to derive Classes from use case diagram**

When analysing a use case diagram, the following rules help identify the MVC related classes

**1.Boundary classes View layer**

 1.Derived from actor interaction with the system

 2.Represents user interfaces / screens / forms

 3.Named like Loginform, PaymentUI, OrderScreen

**2.Control classes Controller layer**

 1.Derived from main flows and alternative flows in use cases

 2.Represents control logic, coordinating between view and model

 3. Named like Paymentcontroller, Ordercontroller

**3. Entity classes Model layer**

 1.Derived from nouns in the use cases description

 2.Represents business / domain data

 3. Name like customer, Payment, Transaction

**MVC Architecture rules**

1.Combination of one actor and a use case results in one boundary class

2.Combination of two actor and a use case result in two boundary class and so no

3.Only primary actor is considered with a use case

4.Use case will result in controller class

5.Each actor will result in one entity class

**Guidelines to place Classes in 3 tire architecture**

After deriving the MVC rules we organise them into 3-Tire architecture

|  |  |  |
| --- | --- | --- |
| Tire | What to place  | Example |
| Presentation | Boundary classes UI / Input / Output  | PaymentUI / NetbankingForm  |
| Business logic  | Controller classes process logic  | Paymentcontroller |
| Data | Entity classes Business data / DB related | Customer, Payment, Bank account |

 Q8. Explain BA contributions in project (Waterfall Model – all Stages) – 8 Marks

***Answer***

**Summary of role of BA in waterfall SDLC model**

|  |  |
| --- | --- |
| Waterfall Stage | BA Contributions / Responsibilities |
| 1. Requirement Gathering | - Conduct stakeholder interviews, workshops, and surveys-Elicit functional & non-functional requirements- Create BRD (Business Requirement Document) and Use Cases |
| 2. System Design | - Assist in converting business requirements into system-level specifications- Collaborate with architects/UI/UX teams- Validate design against business needs |
| 3. Implementation | - Clarify requirements for developers during coding- Respond to requirement-related queries- Support change requests (CRs) |
| 4. Integration & Testing | - Prepare test scenarios & test cases- Assist QA team in validation of functionality- Support User Acceptance Testing (UAT) |
| 5. Deployment | - Coordinate with stakeholders for go-live readiness- Support training and user manuals- Validate that solution meets business goals |
| 6. Maintenance | - Collect and analyze feedback- Propose enhancements/improvements- Track and manage Production Issues (PIRs) or change requests |

**Role of BA in Projects in details**

|  |  |  |
| --- | --- | --- |
| Stages  | Activities | Artifacts and Resources |
| -Pre-Project | Enterprise analysis-SWOT Analysis, GAP Analysis, Market research, Feasibility study, Root cause analysis, Decision analysis, Strategy analysis, Enterprise architectural frameworks, project scope and business case writing, Risk analysis | -Business case-SOW (Statement of work)-PO (Purchase order)-Sr. BA, Business Architects -Pre-sales Consultants |
| -Planning and Estimation and Assessment-Project Kick off-Big picture plan | 1.Understand assumptions and constraints along with business rules and business goals2.Plan packages for big projects3.Understand the project plan from PM4.BA conducts stakeholder analysis5.Plan BA approach strategy (Req gathering tech, communication, Req management, Documents to follow, Tools to use, Change request handling methodology) for this project | -PM-Sr BA |
| -Requirements Gathering | 1.Stakeholder identify and document2.Client give BRD or BA prepares BRD by interacting with client-Brainstorming, Document analysis, reverse engineering, interviews, workshops, focus groups, observations, Questionnaires3.Prototyping can be used by BA to make the client give more specific requirement4.Sort the gathered requirements (Avoiding duplicates reqs, grouping into similar functionalities or into modules)5.Prioritize requirements- MoSCoW6.Validates requirements-FURPS | -BRD (Business requirements documents)-BA-PM |
| -Requirements Analysis | 1.Draws UML Diagrams (Use case and Activity diagram)2.Prepare functional requirements from business requirements3.All architects comes up with technical requirements (SSD)4.SRS will have functional requirements and technical requirements5.Takes sign off on SRS from client. SRS is the first legal binding document between the business and the technical team6.BA prepared RTM from SRS before design phase starts (BA is the owner of RTM)7.BA traces how requirements are dealt in each phase of development life cycle from design till UAT | -Functional requirements specification-SSD (Supplementary Support Document)-SRS (Software Requirement Specification)RTM (Requirement Traceability Matrix)-BA-PM-Solution Architect-DB Architect-NW Architect |
| Design | 1.From Use case diagram, Test Manager or BA will prepare Test Cases2.Communicates with client on the design and solution documents (updates status to client and make them understand how the solution would look like to prepare them to drive UAT3.BA will initiate the preparation of End user manuals4.Update RTM5.From Use case diagram solution architect recommends architecture of the IT solution6.DB architect uses persistence classes entity classes and come up with ER diagram or DB schema7.GUI designer will investigate transient classes (Boundary classes) and design all possible screens for the IT solution | -Solution document-Design document-HDD-ADD-BA-PM-Solution Architect-DB Architect-NW Architect-GUI Designer-Test Manager |
| Coding | 1.BA organises JAD sessions2.BA clarifies queries of Technical Team during coding3.Developers refer Diagrams and Transient Controller classes of BA and code their unit 4.Update End user manuals5.Update RTM6.Conducts regular status meetings with technical teams and the client and tuning client for participation in UAT | -LDD-CDD Application-Development team-BA-PM |
| Testing | 1.BA prepares test cases from use cases or assists test manager to do so2.BA performs high level testing3.BA prepares client for UAT4.Test data is requested by AB from client5.Updates End user manuals6.Update RTM7.Take sign off from client-on-client project acceptance form | -Test concerning documents-Application with less errors-Testing team-BA-PM-Client |
| Deployment and Implementation | 1.Forwards RTM to client or the PM which should be attached to the project Clouser document2.Coordinate to complete and share end user manuals3.Plans and organise training session for end user4.Prepare lesson learned from this project to take precautions for coming projects |  |

 **Q9.** What is conflict management? Explain using Thomas – Kilmann technique – 6 Marks

***Answer***

**Conflict management** refers to the process of handling and resolving the conflicts and disagreements that arise between individuals or group within an organisation.

**The Thomas-Kilmann** conflict mode instrument TKI is a widely used technique for understanding and managing conflict.

Thomas-Kilmann technique helps individuals understand their preferred conflict handling styles and provides insights into when each mode might be appropriate.

Effective conflict management involves recognizing the existence of conflicts, actively listening to the concerns of all parties involved, seeking common ground, and working towards collaborative solutions that meets the needs of everyone.

**5 options of conflict management**

1.Competing

2.Avoiding

3.Accomodating

4.Collabrating

5.Comprising

**Managing the conflict**

1.Conflicts require high level of energy for resolution

2.Managing the conflict is all about maintaining the relations

3.Habits are limiting factors in managing the conflicts

4.Before engaging in conflicts, think about expected outcomes

5 Steps to conflict management

1.Identify the conflict

2.Discuss in details

3.Agree with the root problem

4.Check for every possible solution for the conflict negotiate the solution to avoid the future conflicts

Conflict management, utilizing the Thomas-Kilmann Conflict Mode Instrument (TKI), involves understanding and addressing conflicts through five primary modes:

**Competing:** This mode involves asserting one's own concerns at the expense of others. It is suitable when quick, decisive action is necessary, such as in emergencies or when unpopular decisions need to be made.

**Collaborating:** This mode seeks to find mutually beneficial solutions where both parties' concerns are addressed. It involves exploring concerns and finding alternatives through open communication and creativity.

**Compromising:** In this mode, individuals seek a middle ground where both parties give up something to reach a mutually acceptable solution. It is useful when time is limited, and partial satisfaction is acceptable.

**Avoiding:** This mode involves sidestepping or postponing conflict, which may be appropriate when issues are trivial, emotions are high, or when more important issues need attention.

**Accommodating:** This mode emphasizes satisfying the concerns of the other party, often at one's own expense. It is suitable when preserving harmony is crucial or when the issue is less important to one party.

Each mode addresses different aspects of conflict depending on the situation and the parties involved. Effective conflict management involves selecting the appropriate mode based on factors like the importance of the issue, relationship dynamics, time constraints, and the desired outcomes. Flexibility in applying these modes can lead to constructive resolutions that benefit all parties involved.



 **Q10.** List down the reasons for project failure – 6 Marks

***Answer***

**Reasons for project failure**

Here are some common reasons for project failure, especially relevant in IT and software development projects:

1.Unclear, improper requirement gathering or Changing Requirements: Poorly defined project requirements or frequent changes without proper impact analysis can derail a project.

2.Lack of Proper Planning: Inadequate project planning leads to missed deadlines, cost overruns, and resource mismanagement.

3.Poor Communication: Miscommunication between stakeholders, team members, or clients can lead to misunderstandings and delays.

4.Inadequate Risk Management: Failure to identify, assess, and mitigate risks can cause unexpected issues during project execution.

5.Scope Creep: Uncontrolled changes or continuous growth in the project's scope can lead to overburdened teams and missed deadlines.

6.Lack of Stakeholder / User Involvement: Insufficient engagement or support from stakeholders can result in misaligned goals and lack of commitment.

7.Lack of Executive Support is another critical reason for project failure.

 **Here is how it impacts a project:**

1.No Clear Direction: Without executive backing, the project may lack strategic alignment with business goals.

2.Limited Resources: Executives play a key role in allocating budgets, tools, and manpower. Lack of support can result in resource constraints.

3.Low Morale and Motivation: Teams may feel the project is not valued if leadership does not show interest or involvement.

4.Slow Decision Making: Executive-level decisions may be delayed, causing bottlenecks.

5.Reduced Influence: Without top-level support, project managers might struggle to get cooperation across departments. Each of these factors can individually or collectively contribute to a project not meeting its objectives in terms of time, budget, or quality.

8.Unrealistic Expectations is also a major reason for project failure.

1.Overly Ambitious Deadlines: Trying to complete too much in too little time leads to rushed work, quality issues, and burnout.

2.Underestimated Costs and Resources: Setting expectations without understanding the actual needs results in budget overruns and resource shortages.

3.Pressure on Team: Unrealistic goals create stress, lower morale, and increase turnover.

4.Misalignment with Reality: Stakeholders may expect features or outcomes that are technically or financially unfeasible, leading to dissatisfaction.

 **Q11.** List the Challenges faced in projects for BA – 6 Marks

1.Ambigious, Unclear or changing Requirements: -Stakeholders may not clearly define what they want, or change their expectations mid-project, leading to confusion and rework.

2.Project Communication: -Misunderstandings between stakeholders, developers, and testers can result in incorrect or incomplete requirements.

3.Time and Resource constraints: -Continuous addition of new features or changes without proper evaluation impacts timelines and deliverables and scope creep can occur.

4.Lack of Stakeholder involvement: -If stakeholders are not actively involved, it becomes difficult to gather accurate requirements and validate solutions.

5.Managing the conflicts and Negotiations: -Different departments may have conflicting goals, making it hard to prioritize and finalize requirements.

6.Limited Domain Knowledge: -If the BA is unfamiliar with the industry or product, it may lead to gaps in understanding and misalignment with business needs.

7.Stakeholder Management: -Managing stakeholders with different priorities, availability, and opinions can be difficult, especially in large or cross-functional teams.

8.Unclear Project Objectives: - When the overall goals of the project are not clearly defined, it becomes difficult for the BA to align requirements with business needs.

9.Resistance to Change: -End users or stakeholders may be reluctant to adopt new systems or processes, slowing down implementation and acceptance.

 **Q12.** Write about Document Naming Standards – 4 Marks

***Answer***

Document Naming Standards refer to a consistent and structured approach to naming files and documents within a project or organization. These standards help ensure clarity, uniformity, easy retrieval, version control, and collaboration among team members.

**Key Elements of Document Naming Standards:**

1.Consistency

Use a fixed structure, such as [ProjectName]\_[DocumentType]\_[Date]\_v[Version] (e.g., ScrumFoods\_BRD\_2025-04-22\_v1.0).

2.Descriptive Naming

Document names should clearly reflect the content or purpose (e.g., Requirements\_Specification, Test\_Case\_Suite).

3.Version Control

Include version numbers or status in the filename to track changes (e.g., v1.0, v2.1, Final, Draft).

4.Date Format

Use a consistent and sortable date format, such as YYYY-MM-DD, for chronological organization.

**Benefits:**

1.Avoids confusion

2.Enhances collaboration

3.Makes searching and organizing files easier

4.Supports audit trails and document traceability

All Documents will be named using some standards below is example

Like [ProjectID][DocumentType]V[x]D[y]. ext

Example

1.PQ786BRDV1D2.docx

2.PQ786BRD1.2.docx

 **Q13.** What are the Do’s and Don’ts of a Business analyst – 6 Marks

***Answer***

**Dos and Don’ts as BA**

1.Never say NO to client

2.There is NO word called as “BY DEFAULT”

3.Never imagine anything in terms of GUI

4.Question the existence of existence / questions everything in the world

Example: -What client gives is not always correct

Consult an SME for clarifications in requirements

Every problem of client is unique. No two problems of different client are same. May be the approach, technology, place of use, local laws, may be varied to make problems to be different.

Go to client with a plain mind with no assumptions. Listen carefully and completely until client is done and then you can ask your queries. Please do not interrupt the client when he / she is giving you the problem. Maximum try to extract the leads to solution from the client itself. Never try to give solutions to client straight away with your previous experience and assumptions. Try to concentrate on the important and truly required requirements. Don’t be washed away by add on functionalities or don’t imagine solutions on screen basis.

**In short**

Do’s of a Business Analyst

1.Understand Business Needs Clearly: -Always gather complete and accurate requirements aligned with business objectives.

2.Communicate Effectively: -Maintain clear and continuous communication with stakeholders, developers, and testers.

3.Document Thoroughly: -Create and maintain well-structured documentation like BRDs, FRDs, and user stories.

4.Ask the Right Questions: -Clarify ambiguities and dig deep to uncover hidden requirements.

Don’ts of a Business Analyst

1.Don’t Make Assumptions: -Avoid assuming requirements or processes without validation from stakeholders.

2.Don’t Ignore Stakeholder Inputs: -All relevant feedback must be considered to avoid gaps and ensure solution alignment.

**Do: Clarify, Communicate, Document**

**Don’t: Assume, Overlook, Ignore**

 **Q14.** Write the difference between packages and sub-systems – 4 Marks

***Answer***

1.Package: -A package is a grouping of related classes or components that share common functionality, organization, or purpose. It helps to logically organize the system and manage large projects by keeping the model modular.

Example: A package could be "UserManagement", which contains classes like User, Admin, UserProfile, etc.

2.Component: -A component is a modular, self-contained unit of functionality within a system that has a defined interface. It interacts with other components but is independent in terms of its implementation. Components are typically part of a larger system and often correspond to distinct functionalities, like data access or reporting.

Example: A PaymentProcessing component could handle all aspects of payment-related operations.

3.Sub-system: -A sub-system is a larger, more complex module that represents a distinct functional part of a system. It consists of several components or packages and communicates with other sub-systems through interfaces.

Example: A Customer Relationship Management (CRM) sub-system could include components like User Management, Sales Tracking, and Customer Support.

4.Module: -A module is a self-contained unit of functionality within a software system that is logically separated from others, often corresponding to a specific feature or business function. A module may consist of one or more components or packages.

Example: An Inventory Management Module could include components for stock tracking, inventory reporting, and order management.

**Difference between packages and sub-system**

|  |  |  |
| --- | --- | --- |
| Aspects | Packages | Sub-system |
| Definition | A package is a grouping of related classes or components to organise the model | A sub-system is larger, self-contained module representing a major functional part of a system  |
| Purpose | Mainly use for logical organisation and modular design | Represents the functional division within the system often with its own interfaces  |
| Complexity level | Usually smaller and less complex than sub system | Larger and more complex may include multiple packages |
| Representations | Represented as folder icon in UML diagrams | Represented as a component or composite structure in UML |
| Dependency management | Mange dependency at class / component level  | Manage dependencies at higher level, defining boundaries and interfaces between different parts of the system  |
| Usages | Application development companies work on packages | Product Development companies work on Sub-System |

|  |  |
| --- | --- |
| Object | An object is instance of class |
| Class | Collection of similar objects it is blue print |
| Component | Collection of classes is a component |
| Package | Collection of components which are not reusable in nature |
| Sub-System | Collection of Components which are reusable in nature |

 **Q15.** What is camel-casing and explain where it will be used- 6 Marks

***Answer***

**Camel-Casing** is a popular naming convention in programming and other technical fields where multiple words are combined into a single string, and each word starts with an uppercase letter, except for the first word. This makes it easier to read and understand the name, particularly when it is used for variables, functions, or methods.

**Example of Camel-Casing:**

1.userName

2.calculateTotalAmount

3.getUserData

The term "camel-case" comes from the hump-like appearance of the capital letters in the middle of the words.

**Where Camel-Casing is Used:**

1.Variable Names

It is commonly used for variable names in programming languages like Java, JavaScript, Python (for variables), C#, and more.

Example: totalAmount, userProfile.

2.Function/Method Names

Functions and methods often follow camel-casing, especially in languages like JavaScript, Java, and C#.

Example: calculateTotal(), getUserDetails().

3.Class Names

In some coding standards, class names are written in Pascal Case, which is like camel-case but starts with an uppercase letter.

Example: CustomerService, InvoiceManager.

4.Event Handlers

In event-driven programming, especially with JavaScript and front-end frameworks, camel-case is used for event handler names.

Example: onClick, onSubmit.

5.API Endpoints

For RESTful API endpoints, camel-casing is often used for naming parameters or JSON keys.

Example: { "userId": 123, "orderDate": "2025-04-22" }.

**Benefits of Using Camel-Casing:**

1.Readability: Camel-casing helps improve readability, as the capitalization at the start of each word creates a clear distinction between them

2.Consistency: It provides consistency in naming across different programming languages, especially when working with object-oriented code

3.Avoids Special Characters: By using only letters and avoiding spaces or special characters, camel-casing makes code clean and readable

**How to write in camel case?**

1.First word lowercase: Always start the first word in lowercase (userProfile, not UserProfile)

2.Use it for simple words: Avoid using camel-case for overly long or complex names; consider using underscores (\_) in such cases.

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

***Answer***

A Development Server is an environment where software developers build, test, and debug the application before it moves to higher environments like Testing (UAT) or Production.

It is used for:

1.Writing and modifying code

2.Unit testing

3.Integrating new features

4.Fixing bugs

5.Testing APIs

The development server is often unstable and not exposed to end-users. It supports frequent changes and is usually refreshed or deployed with each new feature or bug fix.

**Key Features of a Development Server:**

1.Supports early-stage testing

2.Allows real-time changes in code

3.Connects to a mock or test database

4.Can be accessed by developers, testers, and sometimes BAs

The access level for a Business Analyst (BA) may vary depending on the organization, but generally includes:

1.Read-Only Access to Application: -BAs can log in to the development environment to validate features against requirements or user stories.

2.Access to Logs or Error Messages (if permitted): -BAs may view logs or exception messages to help analyze bugs or verify acceptance criteria.

3.Access to Development Builds: -BAs can interact with the latest builds to conduct informal walkthroughs or internal demos.

4.View Configuration or Master Data (limited): -BAs might check configuration screens or master tables to ensure business logic is implemented correctly.

5.Collaboration with Developers: -BAs often work closely with developers on the dev server to explain requirements or clarify doubts in real-time.

**What BAs Typically Do NOT Have:**

1.Code-level access

2.Write access to server settings

3.Deployment permissions

4.Direct access to databases (unless granted specifically for testing/troubleshooting)

Example Scenario:

In a food delivery app project Scrum Foods, BA might log into the development server to:

1.Check if the Promo Code logic is working as per the BRD

2.Confirm that the Agent Commission calculation shows correct output

3.Validate screen flow before it is moved to the QA/UAT environment

**Q17.** What is Data Mapping 6 Marks

***Answer***

**Data Mapping** is the process of matching fields from one data source to another. It defines how data from the source system corresponds to data in the target system. This is crucial when integrating systems, migrating data, or transforming data for reporting and analytics.

**Key Components of Data Mapping:**

1.Source Field: -The original data field from the source system.

Example: Cust\_Name in legacy database.

2.Target Field: -The destination field in the target system.

Example: CustomerFullName in the new CRM system.

3.Transformation Logic: -Rules for converting or formatting data between source and target.

Example: Combining FirstName and LastName into a single FullName.

**Where Data Mapping is Used:**

1.Data Migration Projects: -When moving data from old systems to new ones.

2.System Integration: -Connecting different systems (e.g., CRM to ERP) so that data flows correctly.

3.ETL (Extract, Transform, Load): -In data warehousing and analytics, mapping defines how raw data is transformed into usable insights.

4.API Integrations: -Ensuring data sent from one system matches the expected structure of another via API.

**Example of Data mapping**

|  |  |  |
| --- | --- | --- |
| Source Field | Target field | Transformation rule |
| First\_Name | CustomerFullNames | First\_Name+ Last\_Name |
| DOB | Birthday | Covert format to YYYY-MM-DD |
| Phone Number | ContactNumber | Remove country code if exists |

**Importance of Data Mapping:**

1.Ensures data accuracy and integrity

2.Helps in smooth data migration and system integration

3.Minimizes data loss or duplication

4.Supports business continuity during system upgrades

**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 10 Marks

***Answer***

An API (**Application Programming Interface**) is a set of rules and protocols that allows two software applications to communicate and exchange data with each other.

It acts like a messenger that delivers a request from one system to another and brings back the response.

API Integration is the process of connecting different systems through APIs to allow data sharing, feature access, and system collaboration. It enables seamless interaction between applications, even if they are built on different technologies.

**Use Case Scenario (Date Format Conversion)**

Let us consider Scrum Foods accepts date in dd-mm-yyyy format.

You are receiving data from a US-based system through an API, where the date format is mm-dd-yyyy.

How to Use API Integration in This Case:

1.Established API communication

2.Set up API communication between your application and other application

3.API request received: -US based application sends data

4.Middleware or backend processing: -On receiving the data your backend must parse the incoming date called data parsing

5.Date format conversion logic: -The application logic should convert the data format before storing or processing

6.Data saved and validated: -Once converted the date is saved in database and displayed through GUI as required

 **Points in Implementation:**

1.Use a Date Parser in your backend that can interpret and convert date formats.

2.Validation: Ensure the date is valid before converting.

3.Error Handling: Handle cases like invalid formats or missing values.

4.Consistency: Always convert incoming data to a standard internal format used by your system.

---------------------------------------------------------------END-----------------------------------------------------------------