CAPSTONE PROJECT 3

Case Study 1 (Q1-Q6 → 12 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 - 2 Marks**



**Q2. Derive Boundary Classes, Controller classes, Entity Classes. - 2 Marks**

**Boundary Classes**: Boundary classes in software design represent the system's interface with external entities or actors, such as users or external systems. These classes handle input and output interactions, facilitating communication between the system and its environment.

**Controller Classes**: Controller classes orchestrate and manage the flow of operations within a system. They interpret user inputs received from boundary classes, process them, and direct appropriate actions or updates to the system's entities or components.

**Entity Classes**: Entity classes represent core business objects or data within a system. They encapsulate data and business logic, often corresponding to tangible objects or concepts in the real world. These classes primarily focus on data storage and manipulation without being concerned with the system's operations or interface.

|  |  |
| --- | --- |
| **Boundary classes** | PaymentModeBoundaryCardPaymentBoundaryCashPaymentBoundaryNetBankingPaymentBoundaryWalletPaymentBoundary |
| **Controller classes** | PaymentModeControllerCardPaymentControllerCashPaymentControllerNetBankingPaymentControllerWalletPaymentController |
| **Entity classes** | CustomerPaymentCardWalletServer |

**Q3. Place these classes on a three tier Architecture. - 2 Marks**

PaymentModeBoundary

CardPaymentBoundary

CashPaymentBoundary

NetBankingPaymentBoundary

WalletPaymentBoundary

CashPaymentController

Customer

Payment

Card

Wallet

Server Wallet

Database Layer

PaymentModeController

CardPaymentController

NetBankingPaymentController

WalletPaymentController

Application Layer

Business Logic Layer

**Q4. Explain Domain Model for Customer making payment through Net Banking - 2 Marks**

A domain model is a visual representation or abstraction of a specific business area or subject matter. It captures the key concepts, entities, their attributes, and the relationships among them within a particular domain or business context. The goal of creating a domain model is to gain a clear understanding of the business domain's structure, behavior, and rules.



**Q5. Draw a sequence diagram for payment done by Customer Net Banking - 2 Marks**

A sequence diagram is a visual representation in Unified Modeling Language (UML) showcasing the interaction sequence between various system components or actors. It illustrates the flow of messages exchanged over time, emphasizing the order and dependencies of interactions. Lifelines represent the existence of objects, arrows depict message exchanges between them, and activation boxes indicate the duration of an object's activity during the interaction. These diagrams help understand system behavior, design architecture, and validate scenarios or use cases in software development or system analysis.



**Q6. Explain Conceptual Model for this Case - 2 Mark**

A conceptual model represents abstract ideas or principles that help understand a system, process, or concept. It's a high-level representation, often using diagrams, descriptions, or visualizations to illustrate relationships, components, and interactions within a system. Conceptual models simplify complex concepts, making them easier to comprehend by focusing on key elements and their interconnections without delving into detailed specifics. These models serve as a blueprint or guide to conceptualize, design, or communicate ideas in various fields such as software engineering, psychology, or business analysis.

The payment process through NetBanking begins with customers accessing a secure and user-friendly interface provided by their bank. Authentication, a crucial step, ensures the safety of their accounts before initiating any transaction. Upon selecting the NetBanking option, customers proceed to input payment details, prompting the system to verify these transactions through the bank's secure backend systems. This verification ensures accuracy and security in the transfer of funds or bill payments. Once validated, the transaction receives confirmation, providing both the customer and the recipient with comprehensive details for their respective records. This meticulously designed and efficient process guarantees a seamless and secure experience for users conducting financial transactions via NetBanking.

**Q7. What is MVC architecture? Explain MVC rules to derive classes from use case diagram and guidelines to place classes in 3-tier architecture - 4 Mark**

MVC (Model-View-Controller) architecture is a design pattern used in software development to organize and structure code in a modular and efficient way. It consists of three main components:

1. **Model**: Represents the application's data and business logic. It encapsulates the data and the rules for manipulating or processing that data. Changes in the model trigger updates in the user interface.
2. **View**: Represents the user interface or the presentation layer of the application. It is responsible for displaying the data from the model and sending user inputs back to the controller. Views are kept separate from the application's logic.
3. **Controller**: Acts as an intermediary between the model and the view. It receives user inputs from the view, processes them (often updating the model), and then updates the view accordingly. The controller manages the flow of data and the overall application behavior.

**MVC rules rules to derive classes from use case diagram.**

1. Combination of one actor and a use case results in one Boundary class
2. combination of two actors and a use case results in two Boundary classes
3. combination of three actors and a use case results in 3 Boundary classes and so on

Note: only one primary actor is to be consider with a use case.

1. Use case will result in a controller class.
2. Each actor will result in one entity.

**MVC guidelines to place classes in 3-tier architecture.**

1. Place all Entity Classes in DB Layer.
2. Place Primary Actor associated Boundary Class in Application Layer
3. Place Controller Class in Application Layer

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

**Q8. Explain BA contributions in project (Waterfall Model – all Stages) – 4marks**

|  |  |  |
| --- | --- | --- |
| 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) |
| Planning and Estimations and Assessment | 1. Understand Assumptions and Constraints along with Business Rules and Business Goals 2. Plan Packages for Big Projects 3. Understands the project plan from PM 4. BA conducts stakeholders Analysis 5. Plan BA approach strategy (Req. gathering techniques, communication, Req. mgmt., Documents to follow, Tools to use, Change Request Handling methodology) for this Project.  | PM, Sr.BA |
| Requirement Gathering | 1. Stakeholders identify and document 2. Client gives BRD or BA prepares BRD by interacting with Client - Brainstorming, Document Analysis, Reverse engineering, Interviews, workshops, Focus Groups, Observation, Questionnaires. 3. Prototyping can be used by BA to make the Client to give more specific requirements 4. Sort the gathered Requirements (avoiding duplicate Reqs, grouping into similar functionality or into modules) 5. Prioritize requirements - MOSCOW 6. Validate Requirements - FURPS | BRD (Business Requirements Document), BA PM |
| Requirement Analysis | 1. Draws UML Diagrams (Use case and Activity Diagrams) 2. Prepares Functional Requirements from Business Requirements 3. All Architects comes up with Technical Requirements (SSD) 4. SRS will have Functional Requirements and Technical Requirements 5. Takes Signoff on SRS from Client. SRS is the first legal binding Doc between the Business and the technical Team 6. 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 Requirements Specification) RTM(Requirements Traceability Matrix) BA ,PM, Solution- Architect DB-Architect NQ - Architect  |
| Design | 1. From Use case Diagram, Test Manager or BA will prepare Test Cases 2. 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 UAT) 3. BA will initiate the preparation of End user manuals 4. updates RTM 5. From Use case Diagram Solution-Architect recommends Architecture of the IT solution 6. DB Architect uses Persistence Classes (Entity Classes) and comes up with ER Diagrams or DB Schema. 7. GUI Designer will look into Transient Classes (Boundary Classes) and designs all possible Screens for the IT Solution  | Solution Document Design Document- HDD- ADD |
| Coding |  1.BA organizes JAD Sessions 2. BA clarifies queries of Technical Team during Coding en 3. Developers refer Diagrams and Transient (Controller Classes) of BA and code their unit 4. Update End user manuals 5. Update RTM 6. Conducts regular Status meetings with technical team and the Client and tuning Client for participation in UAT | BA PM Solution-Architect DB-Architect NW-Architect GUI-Designer Test Manger Test Concerning Documents Application with less errors Testing Team BA PM Client |
| Testing | 1.BA- Prepares Test Cases from Use Cases or assists Test Manager to do so 2. BA performs high level testing 3. BA prepares Client for UAT 4. Test Data is requested by BA from Client 5. Updates End User Manuals 6. Updates RTM 7. Take signoff from Client on Client Project Acceptance form | LDD - CDD Application Development Team BA PM  |
| Deployment and implementation | 1.Forwards RTM to Client or the PM which should be attached to the Project Closure Document 2. Coordinates to complete and share End User Manuals 3. Plans and Organizes Training Sessions for End Users 4. Prepares Lessons learned from this project (to take precautions for coming projects) | Test Concerning Documents Application with less errors Testing Team BA PM Client |

**Q9.What is conflict management?Explain using Thomas – Kilmann technique – 3Mark**

Conflict management refers to the process of handling disputes, disagreements, or differing opinions between individuals or groups within an organization. The Thomas-Kilmann Conflict Mode Instrument (TKI) is a technique used to understand and manage conflicts based on five conflict-handling styles:

1. **Competing**: This style is characterized by high assertiveness and low cooperativeness. It involves pursuing one's own concerns at the expense of others, often focusing on winning or achieving personal goals.
2. **Collaborating**: This style exhibits high assertiveness and high cooperativeness. It emphasizes finding solutions that satisfy the concerns of all parties involved, aiming for a win-win outcome through open communication and teamwork.
3. **Compromising**: This style balances moderate assertiveness and moderate cooperativeness. It seeks a middle ground where both parties make concessions, finding a solution that partially satisfies everyone's interests.
4. **Avoiding**: This style involves low assertiveness and low cooperativeness. It includes sidestepping or postponing the conflict, often to maintain harmony or because the issue is perceived as less significant.
5. **Accommodating**: This style displays low assertiveness and high cooperativeness. It prioritizes the concerns and needs of others over one's own, often used to preserve relationships or when the issue at hand is not of personal importance.

These conflict-handling styles provide a framework for understanding how individuals approach and manage conflicts in various situations. Understanding these styles helps in choosing the most appropriate approach to address conflicts effectively.

**Q10. List down the reasons for project failure – 3 Marks**

Common reasons behind project failures:

1. **Unclear Objectives and Goals**: Projects without well-defined, achievable, and communicated goals often struggle. Lack of clarity on what needs to be achieved can lead to confusion and misalignment among team members.
2. **Inadequate Planning and Monitoring**: Poorly structured plans or insufficient monitoring mechanisms contribute to project failure. Without proper oversight, it's challenging to detect issues early or make timely adjustments.
3. **Scope Creep and Changes**: Constantly changing requirements or expanding the project scope without proper evaluation can strain resources, timelines, and budgets. This can lead to a misalignment between expectations and deliverables.
4. **Insufficient Resources Allocation**: Inadequate staffing, funding, or technology resources can hinder project progress. Projects often fail when there's a lack of necessary resources to execute tasks efficiently.
5. **Ineffective Communication and Collaboration**: Poor communication among team members, stakeholders, or departments can lead to misunderstandings, delays, and errors. Misaligned expectations or a lack of transparency can impede progress.
6. **Risk Mismanagement**: Failure to anticipate, identify, or mitigate risks can lead to project setbacks or failures. Ignoring potential risks or not having contingency plans in place can result in significant issues during project execution.

**Q11. List the Challenges faced in projects for BA – 3 Marks**

Challenges faced by Business Analysts in projects are

1. **Changing Requirements**: Constantly evolving business needs create challenges in capturing and prioritizing requirements, demanding agile adaptation and frequent stakeholder communication to maintain alignment.
2. **Scope Creep Management**: Ensuring project scope adherence amid expanding requirements necessitates vigilant monitoring, effective change control, and clear communication with stakeholders to avoid project drift.
3. **Stakeholder Alignment**: Managing diverse stakeholders with conflicting interests requires adept negotiation skills and effective communication to achieve consensus and maintain project focus.
4. **Technology Complexity**: Understanding intricate technological landscapes and integrating diverse systems demands a blend of technical acumen and business understanding for seamless solution design.
5. **Data Quality and Availability**: Dealing with inconsistent or inadequate data sources poses challenges in analysis and decision-making, requiring meticulous data governance and collaboration with data experts.
6. **Time and Resource Constraints**: Balancing tight timelines and limited resources necessitates prioritization, efficient task management, and resource optimization to meet project milestones effectively.
7. **Resistance to Change**: Overcoming resistance to new processes or systems requires effective change management strategies, fostering a culture of openness, and addressing stakeholder concerns.
8. **Documentation and Traceability**: Maintaining comprehensive documentation and ensuring traceability of requirements throughout the project lifecycle demands meticulous record-keeping and robust tracking mechanisms.

**Q12. Write about Document Naming Standards – 2 Marks**

Document naming standards in business analysis are essential for maintaining organization, clarity, and consistency in the plethora of documents generated throughout the analytical process. These standards serve as guidelines for naming files, reports, and other documentation, ensuring that they are easily identifiable, searchable, and comprehensible for stakeholders and team members involved in the analysis. Documents should have names that clearly describe their content. This includes using keywords or phrases that succinctly summarize the document's purpose or contents.

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

**Example:** PQ786BRDV1D2.docx

 PQ786BRD1.2.docx

Breaking down the components of the filename

 "ABCcompany\_ProjectCharter\_v1\_20230105.docx ":

represents the initial version (v1) of the project charter document for the "ABC Company" project, created or modified on January 5, 2023, in Microsoft Word format (.docx).

**Q13. What are the Do’s and Don’ts of a Business analyst – 3 Marks**

Some of the Do’s and Don’ts of a Business analyst are

* Never say NO to Client.
* There is NO word called as "BY DEFAULT"
* Never imagine anything in terms of GUI Question the existence of existence / question everything in the world .
* ex: 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 them (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.

**Q14. Write the difference between packages and sub-systems – 2Marks**

Difference between packages and subsystems:

1. **Packages**:
	* **Function**: Packages organize related classes or resources within programming, aiding in code structuring and management of dependencies at a finer level.
	* **Scope**: They operate within modules, handling smaller code elements for maintainability and reusability.
	* **Granularity**: Packages group closely related components within a system, offering a more detailed organizational structure.
	* **Use**: Primarily employed within a project or application for modular development.
2. **Subsystems**:
	* **Function**: Subsystems encapsulate larger portions of a system, comprising multiple packages or components to provide higher-level abstraction.
	* **Scope**: They manage broader functionalities within the system architecture, defining specific system parts.
	* **Granularity**: Subsystems handle larger chunks of functionality, offering a broader and more comprehensive organizational view.
	* **Use**: Used to divide complex systems into manageable units, aiding in maintenance, scalability, and system flexibility.

**Q15. What is camel-casing and explain where it will be used- 2 Marks**

Camel-casing is a naming convention used in programming and writing where compound words or phrases are joined together without spaces, and each word or abbreviation within the phrase begins with a capital letter except for the first word, which starts in lowercase.

1. **Artifact Naming**: When naming documents or artifacts like reports, diagrams, or requirement files, camel-casing can be employed for readability and consistency, such as "stakeholderAnalysisReport" or "businessRequirementsDocument."
2. **Field or Attribute Naming**: In requirement documents or specifications, camel-casing might be used to label attributes or fields for clarity, like "customerID" or "productDescription."
3. **Diagram Labels**: Applying camel-casing to label elements in diagrams (e.g., use case diagrams, process flows) can enhance readability, such as "placeOrder" or "validateUser."
4. **Consistency in File Naming**: Utilizing camel-casing in naming directories or folders for categorizing documents or project phases helps maintain an organized repository, such as "userAcceptanceTesting" or "systemIntegration."
5. **Database Entity Names**: In cases where business analysts interact with database design or data models, camel-casing might be employed for naming tables or entities, like "customerInformation" or "orderDetails."
6. **Interface or System Naming**: When specifying interfaces or system components, camel-casing can be used for clear identification, like "paymentGateway" or "inventoryManagement."

**Q16. Illustrate Development server and what are the accesses does business analyst has? -3 marks**

A development server is a dedicated environment used by software development teams to create, test, and refine applications before deploying them to production. It mimics the production environment but is separate and isolated, allowing developers to experiment without affecting the live system. It typically includes tools and resources needed for coding, testing, and debugging software.

1. **Read-Only Access**: This level of access allows BAs to view and review the contents, functionalities, and data on the development server but restricts their ability to make any changes. They can analyze the system, verify requirements, and conduct assessments without altering or modifying the code or configurations. This access is mainly for observation and validation purposes.
2. **Collaborative Access**: BAs with collaborative access have the ability to interact with the development team, providing feedback, suggestions, and requirements clarification. They can communicate findings, report issues, and collaborate with developers or testers by sharing insights gained from their analysis. This access level might involve discussions, document sharing, or participation in collaborative tools.
3. **Limited Configuration Access**: In some cases, BAs might have limited access to configure certain aspects of the system, such as user interfaces or business rules, within predefined boundaries. This access allows them to fine-tune specific configurations relevant to business needs without delving into the technical intricacies or altering critical system functionalities.

**Q17. What is Data Mapping 2 Marks**

Data mapping is the process of establishing a relationship or connetion between data elements in two or more different data sources or data format. It involves defining how data from one source corresponds to or transforms into data in another source. Data mapping is commonly used in data integration, data migration and data transformation process .

The purpose of data mapping is to ensure that data can be accurately and effectively transferred converted or transformed between databases or formats it involves identifying the source data elements determining their meaning and structure and mapping them to the corresponding target data elements.

**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 3 Marks**

An API (Application Programming Interface) is a set of rules, protocols, and tools that allows different software applications to communicate and interact with each other. It defines how different software components should interact, making it possible for them to access each other's functionalities and share data.

In the context of handling date formats (dd-mm-yyyy and mm-dd-yyyy) and ensuring data integrity, three critical processes come into play:

1. **Establish API communication:**

Implement the necessary API calls within your application to send the date data to this conversion endpoint before processing or storing it.

1. **Data Formatting**:
	1. **Purpose**: Data formatting involves transforming date data from one format to another. It ensures that dates are represented consistently and uniformly across systems or applications.
	2. **Implementation**: Use formatting functions or methods available in programming languages or date libraries to convert date strings from one format (e.g., dd-mm-yyyy) to another (e.g., mm-dd-yyyy) based on the application's requirements.
	3. **Example**: Transforming "15-03-2023" (dd-mm-yyyy) to "03-15-2023" (mm-dd-yyyy) using date formatting functions before sending data to the US application.
2. **Data Parsing**:
	1. **Purpose**: Data parsing involves interpreting and extracting meaningful information from date strings, ensuring accuracy and validity during conversion processes.
	2. **Implementation**: Utilize parsing functionalities to interpret date strings correctly and extract components such as day, month, and year, ensuring proper conversion between different formats.
	3. **Example**: Parsing "03-15-2023" (mm-dd-yyyy) to identify components as month: 03, day: 15, year: 2023 for correct processing within the application.
3. **Data Validation**:
	1. **Purpose**: Data validation ensures that the converted or parsed date information meets specified criteria or rules, preventing errors or inconsistencies in the application.
	2. **Implementation**: Apply validation checks to confirm that the converted date is logical and valid within the defined context (e.g., checking for valid month/day ranges, leap years, or unexpected characters).
	3. **Example**: Validating converted dates to ensure they correspond to valid calendar dates and avoiding logical inconsistencies like February 30th.