**Capstone Project 3 – Part 1**

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

**Answer.** Use Case Diagram is a visual representation of the interactions between users (actors) and a system, showing how users achieve their goals through the system’s functionalities (use cases).

They have four major elements:

* Actors –who interact with the system.
* System boundary- (rectangular box) describes the scope of the system.
* Usecases or services- specific things a system performs.
* Relationships- interactions between actors and usecases.



**Question 2.** Derive Boundary Classes, Controller classes, Entity Classes

**Answer.** MVC is a design pattern used to separate an application into three interconnected components:

* Model: Represents the data and business logic. It directly manages the data, logic, and rules of the application.
* View: The user interface (UI) part that displays data (from the Model) to the user and sends user inputs to the Controller.
* Controller: Acts as an intermediary between the Model and the View. It processes user input, updates the Model, and renders the final output.

To identify classes from usecase diagram we apply MVC rules from which the classes we get are:

|  |  |
| --- | --- |
| Boundary Class | Payment System InterfacePayment Confirmation InterfaceCard, Wallet, Net Banking and Cash Payment |
| Entity Class | CustomerPayment MethodTransactionMerchantBank |
| Controller Class | Payment ControllerTransaction ControllerPayment GatewayPayment ValidationTransaction Management |

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

**Answer.**

3-tier architecture is a software design pattern that separates an application into three distinct layers:

* Presentation Layer (Application Tier): This is the user interface (UI) layer that interacts with the user and handles user input and display of information. Boundary classes form a part of presentation tier, they are placed in this tier.
* Business Logic Layer (Business Tier): This layer contains the core functionality, processing, and business rules of the application. It manages the logic and operations. This tier comprises of controller classes as it helps in managing the business logic of application.
* Data Layer (Data Tier): This layer is responsible for data storage, retrieval, and management, often interacting with a database or other data sources. Entity classes form a part of data tier or physical tier as they contain models and databases.

The three tiers are independent of each other, promoting better scalability, maintainability, and separation of concerns. Now the derived classes within the 3 tier architecture will fit as follows:

|  |
| --- |
| **Application Tier** |
| Payment InterfaceCard Payment Wallet Payment Cash Payment Net banking Payment Confirm Interface |
| **Business Tier** |
| Payment Gateway Payment Validation Transaction management  |
| **Data Tier** |
| CustomerPayment methodTransactionMerchantBank |

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

**Answer.** A domain model is a conceptual representation that defines the structure, relationship and behaviours of entities within a specific problem domain. It is similar to entity relationship diagram where relationships are determined between multiple tables and components.



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

**Answer.**  Sequence diagram shows how the objects in the system interact and communicate with each other with time to achieve specific task. Developer will draw this. It is used to show the flow of messages, events or actions between the objects of the system. This diagram helps to visualize the behaviour of the system.



**Question 6.** Explain Conceptual model for the case

**Answer.** A Conceptual Model is a high-level representation of the system or process. It is a representation of the business model we have. it helps in understanding the key concepts, their relationships, and the overall structure of the net banking payment system. It serves as a foundation for designing the database schema, defining the application architecture, and implementing the necessary functionalities within the system.

The aim of a conceptual data model is to provide a data centric perspective of the organisation for documenting how different business entities relate to one another. One of the key benefits of conceptual data models is that they can be quickly comprehended and communicated to stakeholders irrespective of technical jargon.

In the Conceptual Model for this payment system:

1. Entities: Represent the key objects like Customer, Payment Method, Transaction, Merchant, and Bank.
2. Attributes: Define the key properties of these entities, such as the type of payment method (Card, Wallet, etc.), the amount in a transaction, and the customer’s payment preferences.
3. Relationships: Describe how entities are connected:
* A Customer can use multiple Payment Methods.
* A Transaction is linked to one Customer, one Payment Method, and one Merchant.
* A Payment Gateway processes certain payment methods, like Card or Net Banking, involving a Bank for payment processing.

This model shows the flow between the customer, payment methods, and transactions without any technical information like how it is to be done. So showing all the respective entities and associations gives us the idea about the concepts involved within the domain of payment and help stakeholder understand the need of necessitating certain functionalities and also helping the development team in working with some functionalities.

**Question 7.** What is MVC architecture? Explain MVC rules to derive classes from use case diagram and guidelines to place classes in 3-tier architecture.

**Answer.** The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application.

* Model - The model class/component corresponds to all the data-related logic that the user works with. Model is responsible for multiple tasks like managing the application's data, performing data validation, implementing business rules, and handling data access operations. It basically knows about all the data that needs to displayed. It can add or retrieve data from the database. It works with the data that is being transferred between view and controller components and responds to the controller requests by retrieving and sending information from the databases. It interacts with the database because controller cannot interact with the database. All Model classes are represented as Entity classes.
* View - The view component or class represents the presentation aspect of the

application. It basically displays data from the model to the user and sends user inputs to the controller. The View can be a web page, a desktop application window, or any other form of user interface. It receives input from the user and passes it to the Controller for processing. View refers to the model. It takes the data from the Model and renders it in a way that is suitable for the user's display or interaction. For rendering the data, it uses query method. It is represented as boundary class.

* Controller - The controller acts as an intermediary between the model and the view

class or component. It handles user input and updates the model accordingly and

updates the view to reflect changes in the model. It is responsible for intercepting the requests from view and passes it to the model for appropriate action. So basically Whenever the user requests for anything, that request first goes to the controller because controller cannot directly get the data from the database. So controller interacts with the model.

MVC Architecture Rules:

1. Combination of one actor and an usecase results in one boundary class.
2. Combination of two actors and an usecase results in two boundary classes and in the same way, three actors and a usecase would result in three boundary classes and so on.
3. Usecase results or corresponds to a controller class.
4. Each actor would result in one entity class.
5. One primary actor is to be considered with a usecase.

Guidelines to place identified MVC classes into 3 tier architecture:

* Place all entity classes in data tier or physical tier.
* Place all primary actors associated boundary class in Application layer.
* Place controller class in Application layer.
* If governing body influence or reusability is existing with remaining boundary classes, place them in Business tier or else Application tier.

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

**Answer.** In the Waterfall Model, which is a traditional and linear approach to project management, the Business Analyst (BA) plays an essential role throughout each phase. Below is a breakdown of BA contributions in each stage:

|  |  |  |
| --- | --- | --- |
| Stage | Activities | Artifacts and Resources |
| Pre Project | Enterprise Analysis-Swot Analysis, Gap Analysis, Marketresearch, feasibility study, Root Cause Analysis, DecisionAnalysis, Strategy Analysis, Risk Analysis, Enterprise ArchitecturalFrameworks, Project Scope, Business Case Writing. | Artifacts - Business Case Statement Of work (SOW) Purchase Order (PO)Resources - Sr.BA, Business Architects, Pre salesconsultants. |
| Planning | Collaborate with the project manager to define the project scope and objectives.- Identify stakeholders and define their roles and responsibilities.- Contribute to project planning activities like risk identification and resource estimation.- Support in setting timelines and milestones. | - Scope Definition Document- Risk Log- Stakeholder MatrixResources: - Project Manager (PM), - Sr.BA (senior Business Analyst) |
| Project Initiation | Conduct stakeholder analysis to understand key priorities and expectations.- Assist in defining measurable business objectives and project goals.- Draft an initial outline of requirements and share with stakeholders for alignment | Initial Requirements Document- Stakeholder Analysis Document- Communication Plan- RACI Matrix |
| Requirements Gathering | - Organize workshops, focus groups, interviews, and surveys to gather requirements.- Document functional and non-functional requirements.- Prioritize requirements and Validate Requirements. | - BRD (Business RequirementDocument- Business Requirements Document (BRD)- Use Case DiagramsResources – BA/PM and Business stakeholders |
| Requirement Analysis | -Draw UML diagrams (Usecase and Activity diagrams)-Prepares functional Requirements from Business Requirements- Analyze requirements to ensure feasibility and alignment with business goals.- All architects in the team would come up with Technical Requirements (SSD).- SRS will have functional Requirements and technical requirements.- Takes Signoff on SRS from the client- BA prepared RTM from SRS before design phase starts. | - Functional Requirements Specification (FRS)- Process Flow Diagrams- SSD (Supplementary Support Document).- SRS (Software Requirement Specification).- RTM (Requirements Traceability Matrix.Resources: BA- PM- Solution- Architect- DB architect- NW architect |
| Design | -Work with the design and architecture teams to ensure alignment with requirements.- Validate system design and wireframes against user needs.- BA will intiate the preparation of end user manuals- From UCD, Solution architect recommends architecture of the IT solution.- DB Architect uses entity classes and comes up with the ER diagrams or Db schema.- GUI designer will look into boundary classes and designs all possible screens for the IT solution. | - Design Document- Wireframes or PrototypesResources:- BA- PM- Solution architects- DB architect- GUI designer |
| Development | Act as a liaison between developers and stakeholders to clarify any ambiguities in requirements.- Track development progress and ensure adherence to documented requirements.- Resolve requirement-related queries and support the development team with domain expertise.- Update and maintain the traceability matrix for requirements | - Updated Requirements Traceability Matrix- Updated BRDResources:- Development team- PM- BA |
| Testing | - BA prepares test cases from use cases- Assist the QA team in designing test cases and scenarios to validate functionality against requirements.- Ensure comprehensive test coverage for both functional and non-functional requirements.- Validate system readiness for UAT. | - Test Cases- Test Plans |
| UAT (User Acceptance Testing) | - Coordinate UAT activities- Ensure acceptance criteria are met- Collect feedback from stakeholders | - UAT Plan- UAT Feedback Report- Sign-Off Document |

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

**Answer**. In a group or team conflicts can occur due to multiple reasons, due to differences opinions, goals, values, or communication breakdowns. To resolve conflict becomes crucial to work together which can be done with conflict management.

Conflict management is the process of handling conflicts and disagreements in a healthy and constructive manner to lead to positive outcomes. For which various strategies and techniques are used. A usual approach used for conflict management typically has 5 steps:

* Identify the conflict
* Discuss the details
* Agree with the root problem
* Check for every possible solution
* Negotiate the solution to avoid future conflict.

Among other methods and techniques one of the most widely used techniques is the Thomas-Kilmann Model. Which is based on two dimensions namely Assertiveness and Cooperativeness/Empathy. Based on these 5 primary types of conflict management styles can be identified which are:

These 5 styles can be graphically depicted where X-axis represents Cooperativeness and Y-axis represents Assertiveness.



* Competing: These people are highly assertive and non-cooperative which makes them aggressive and authoritative. It is a defensive style where it strictly means standing up for your individual beliefs and simply trying to win.
* Accommodating: These people are highly cooperative and non-assertive and usually give in to others demands. You may have to follow other people’s orders when you would not like to do so.
* Avoiding: These are non-cooperative and non-assertive where they avoid or dodge conflict. They do not address anyone’s problems, not theirs and not other’s.
* Collaboration: These are highly assertive and highly cooperative where working together to find a solution for both sides is aimed. It's used when integrating solutions, learning, merging perspectives, gaining commitment, and improving relationships.
* Compromising: These people have moderate levels of assertiveness and cooperativeness. This style aims to find a quick, mutually acceptable solution that partially satisfies both parties. it addresses an issue more directly than avoiding but falls short of investigating it with as much depth and rigor as collaborating.

**Question 10.** List down the reasons for Project Failure

**Answer.**  There can be various reasons behind a project failure, sometimes multiple reasons can contribute to product failure or sometimes even a few can lead. Some of the reasons which can lead to project failure are as follows:

* Improper Requirement Gathering: Failing to collect all necessary details or understanding the requirements inadequately can lead to a project that does not meet the intended goals.
* Lack of Stakeholder Involvement: Without regular input and engagement from stakeholders, projects can veer off track or fail to meet their critical needs.
* Ineffective or Insufficient Communication: Poor communication among team members or between the team and stakeholders can result in misunderstandings and misaligned expectations.
* Continuous Change in Requirements: Frequent changes or scope creep can disrupt project timelines and budgets, making it difficult to deliver the intended outcome.
* Poor Risk Management: Not identifying, assessing, or preparing for potential risks can lead to unexpected challenges that the project is not equipped to handle.
* Unrealistic Expectations: Setting goals that are not achievable with the available resources or within the timeline can doom a project from the start.
* Improper Planning: Inadequate or inaccurate planning can result in timelines and budget estimations that are not feasible, leading to project overruns.
* Insufficient Resources: Lack of necessary resources, whether it be funds, personnel, or technology, can impede the progress and completion of a project.
* Lack of End-User Involvement: Not involving the actual users of the product or service in the development process can result in a final product that does not meet the user’s needs.
* Scope Creep: Uncontrolled changes or continuous growth in a project’s scope, often leading to budget overruns and delays.
* Technical Challenges: Issues related to technology complexities, which can delay projects and increase costs.

**Question 11.** List the challenges faced in projects for BA.

**Answer.**

* Unclear or Changing Requirements: Handling frequent changes to requirements can disrupt project scope and timelines. Constantly evolving requirements make it hard to finalize designs and can lead to increased costs and delays.
* Managing Stakeholder Expectations: Balancing diverse expectations from various stakeholders and aligning them with project goals can be challenging. Different stakeholders may have conflicting priorities, making it difficult to please everyone and stay on track.
* Scope Creep or Scope Management: Controlling scope to prevent unapproved or uncontrolled growth in project boundaries is vital. Scope creep can lead to projects going over budget and timelines extending beyond initial estimates.
* Time and Resource Constraints: Dealing with limited timelines and resources while trying to deliver on project objectives. Insufficient resources or tight deadlines can compromise the quality of the final product.
* Quality Assurance and Testing: Ensuring the product meets required standards and effectively managing the testing processes is crucial. Poor quality and inadequate testing can result in a product that fails to meet user needs or contains significant bugs.
* Documentation and Domain Knowledge: Maintaining comprehensive documentation and acquiring sufficient domain knowledge to support project needs. Inadequate documentation or lack of understanding can lead to misinterpretations and errors in the project deliverables.
* Technology Constraints and Complexity: Navigating limitations and complexities associated with the project’s technology. Technical constraints can limit the solution options available and affect the overall feasibility of the project.
* Obtaining Sign-Off on Requirements: Securing formal approval on requirements ensures all stakeholders are on the same page. Delays in obtaining sign-off can halt project progress and affect timelines.
* Coordinating Between Developers and Testers: Facilitating communication and collaboration between development and testing teams to ensure alignment. Poor coordination can lead to misunderstandings, errors, and a product that does not fully meet requirements.
* Preparing Effective Reports: Creating reports that accurately convey project status, insights, and risks to stakeholders. Ineffective reports can lead to misinformed decisions and overlooked risks.
* Driving Clients for UAT (User Acceptance Testing) Completion: Overseeing the UAT process to ensure the final product meets user needs and is ready for deployment. Ineffective UAT management can result in a product that doesn’t truly solve the user’s problems or has usability issues. Driving clients for UAT completion is challenging as it requires coordinating busy stakeholders, providing them with the necessary training and support, and motivating timely feedback.
* Overall making sure project health is in good shape and delivered as per the time-lines without any issues.

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

**Answer.** Naming a document is also one of the key tasks though it is often overlooked by importance. Naming documents in a standardised, logical, intuitive way and ensures that team members and collaborators can discover, manage and access documents when required. Having a standard format also helps in reducing informal protocols or procedures.

A good naming standard includes:

1. [Project ID] [Document Type] [V ] [D].ext- this states that a project must have a project identification number (ID), the type of document ,its version (V), date (D) and extension.

For example, [PQ777-FRD-1.0-2024-12-24.docx]

Where, Project name – PQ777

Document Type – FRD

Version – 1.0

Date – 24/12/2024

Extension – Word document

**Question 13.**  Dos and Don’ts of a BA

**Answer.**

|  |  |
| --- | --- |
| Do’s | Don’ts |
| * Consult an SME for Clarifications in Requirements.
* 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.
* Try to extract the leads to Solution from the Client itself.
* Try to concentrate on the important and truly required Requirements.
* Question the existence of existence / question everything in the world
* Be like a lotus in mud- if a client comes with a fancy requirement, then talk to the project manager first.
* Always appreciate the stakeholder even for small efforts.
 | * Never say NO to Client
* There is NO word called as "BY DEFAULT"
* Never imagine anything in terms of GUI
* Never interrupt the Client, when he/ she is giving you the problem.
* Never try to give Solutions to Client straight away with your previous experience and assumptions.
* Don’t be washed away by add on Functionalities or don’t imagine solutions on Screen basis.
* Never criticize the stakeholder.
 |

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

**Answer.** Packages

In UML models and Object Oriented Analysis, a package is a organised group of elements. It can be termed as the UML mechanism for grouping things. It may contain many structural things like classes, components and other packages in it. It can be used to:

* Group semantically related elements.
* Define a semantic boundary in the model.
* Provide units for parallel working and configuration management.
* It is used to provide encapsulated namespace within which all names must be unique.

Subsystem

In UML models, subsystems are a type of stereotyped components that represent independent, behavioural units of a system. They are widely used in class, component, use case diagrams to represent large-scale components that are to be modelled. An entire system can be modelled by a hierarchy of subsystems. The behaviour of each subsystem can be defined by specifying the interfaces and operations that support interfaces in accordance with subsystems.

Differences between Packages and Subsystems are as follows:

|  |  |
| --- | --- |
| Packages | Subsystems |
| A package can be termed as a container which tends to organise, group elements present in the system into a more manageable unit. | A subsystem is a stereotyped components which represents individual behavioural units in a system hierarchy. |
| Package can be termed as a collection of components which are not reusable in nature. | Subsystem can be termed as a collection of components which are reusable in nature  |
| Application development companies work onpackages. | Product development companies work on subsystems. |
| Package is represented as a rectangle with tab in upper left corner. The rectangle contains name of the package and icon | Subsystem is displayed as a rectangle that contains the name of the subsystem and icon along with<<subsystem>> keyword |
| Packages are smaller and more focused in scope. | Subsystems are larger and encompass multiple packages or modules |
| They manage dependencies at class and component level | They manage dependencies at a higher level, defining boundaries and interfaces between different parts of the system |

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

**Answer.** Camel casing is a way of writing names in programming or documentation where each word is written without spaces, and the first letter of each word (except the first one) is capitalized. It looks like the humps of a camel.

It is a type of writing where each word is capitalized so that it stands out from the rest of the text. Some examples of CamelCase include HTML tags, Java Script Variables, SQL connections, JSON data and CSS selectors. All these words are created by combining multiple words into one single term. This type of writing helps keep code easier to read and understand for those familiar with the syntax.

Camel Casing has a wide range of utility of which some are:

* Used in requirements documentation to name entities like usecase, features, user stories, etc.
* Database tables also uses camel-casing.
* Programming and web development: Camel case is a common naming convention for variables, functions, and other elements. It's popular because it's easy to read and understand.
* Company names: Some companies use camel case in their names or for their products and systems, such as FedEx, PlayStation, PayPal, MasterCard, and PowerPoint to be registered as new trademarks.
* It is also used in web URLs.

By using camel casing in the documents, it helps to maintain consistency in the entire document and also increases readability

**Question 16.** Illustrate Development server and what are the accesses does business analyst has?

**Answer.** A server is a hardware system or computer program dedicated to managing access of centralised resources or services supplied to users of the computers on the network.

A development server refers to a dedicated environment that is used during the software development process. It is designed to facilitate the development and testing of programs and websites, software applications for software programmers. It provides a quick run time environment as well as hardware and software utilities that are essential for program debugging and development.

As a BA we are provided very limited access we only have permission for viewing the development server. Which is basically read only access. Checking the progress with the development team but not involved in the process of development.

Collaborative Access- this access refers to group access where out of multiple functions, limited scope is provided.

Limited Configuration Access- limited access to the functions of the server.

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

**Answer.** Data Mapping is the process of matching fields from one database to another. It’s the first step to facilitate data migration, data integration and other data management tasks. It can be basically described as the process of connecting a data field in another source. .This reduces potential of errors, helps standardise the data and makes it easier to understand the data.

Data mapping also enables smooth flow of data through different systems, applications and services and also a critical element of data privacy.

There are steps which needs to be taken for data mapping which are as follows:

1. Define- Define the data to be moved, including tables, fields and format within each field.
2. Map the data- Match the source fields of target data to their respective destination fields and identifying their relationships by determined set of instructions.
3. Transformation- After the data is integrated, if a field requires any transformation or changes, the transformation rule is coded.
4. Test- Using a test system and a sample data from source, run the transfer to see how it works and adjust if necessary.
5. Deploy- Once it is determined that the data transformation is working as per planned, the data is deployed.

The main purpose of data mapping is to:

* Data Integration - While combining the data from different sources, it ensures that the data is properly matched.
* Data Migration - While migrating the data from source system to the new system (destination), the data elements are mapped accurately into the new system.
* Data Transformation - Data transformation means converting the data from one format to other. In data mapping, data transformation plays very important role which ensures that the data of source system is mapped correctly to the data in new system.

**Question 18.** What is API. Explain how you would use API integration in the case of your application Date format is dd-mm-yyyy and it is accepting some data from Other Application from US whose Date Format is mm-dd-yyyy.

**Answer.** API stands for Application Programming Interface. It is a software intermediary that allows the two application to interact and exchange data with each other without any user intervention.

It is the set of rules, protocols and tools that define how different software application should interact with each other. API allows sharing of only necessary information and keeps the internal system details hidden, which helps the system security.

Importance of API:

* It helps you immerse content from any site or application efficiently.
* Content generated through API can be published automatically.
* It allows the user or a company to customize the content and services which they use the most.

Using API integration from accepting date format mm-dd-yyy from other app while it is dd-mm-yyyy:

API integration is the connection between two or more applications via API’s that enables those systems exchange data. So here are the following steps:

1. Establish API Connection: Set up API communication between your application and other US based application to exchange data.

2. Data Formatting: When sending date data from your application to the other application, change the format from dd-mm-yyyy to mm-dd-yyyy format

3. Data Parsing: When receiving date data from other application, parse the mm-dd-yyyy formatted date into you application’s date format (dd-mm-yyyy) by extracting and rearranging components of date.

4. Perform Data Validation and ensure that the converted date remains in a valid format.