**CAPSTONE PROJECT 3 - PART 1/2**

**Question 1**

Draw a Use Case Diagram

**Answer**



**Question 2**

Derive Boundary Classes, Controller classes, Entity Classes.

**Answer**

**Boundary Classes:** These classes handle interactions between the system and its actors (users, external systems, etc.). In the context of payment, boundary classes could include interfaces, screens, or APIs through which users interact with the system.

* Payment option Boundary
* Card Payment Boundary
* Wallet Payment Boundary
* Cash Payment Boundary
* Net Banking Payment Boundary

**Controller Classes:** These classes coordinate interactions between the boundary classes and the entity classes. They contain business logic and handle requests from the boundary classes.

* Payment initiated Controller
* Card Payment Controller
* Wallet Payment Controller
* Cash Payment Controller
* Net Banking Payment Controller

**Entity Classes:** These classes represent the core data or objects manipulated by the system. In the context of payment, entity classes could include objects representing transactions, payment methods, accounts, etc.

* Customer
* Payment
* Card
* Wallet
* Server

**Question 3**

**Place these classes on a three-tier Architecture.**

**Answer**

|  |
| --- |
| **User layer** |
| Payment methods selection boundary |
| Card payment boundary |
| Wallet payment boundary |
| Cash payment boundary |
| Net banking payment boundary |
| **Business logic layer** |
| Payment controller |
| Card payment Controller |
| Wallet payment controller |
| Cash payment controller |
| Net banking payment controller |
| **Data layer** |
| Customer entity class |
| Payment entity class |
| Card entity class |
| Wallet entity class |
| Bank account entity class |

**Question 4**

Explain Domain Model for Customer making payment through Net Banking

**Answer**

 ****

****

 ****

** **

 ****

Question 5

Draw a sequence diagram for payment done by Customer Net Banking

**Answer**

****

**Question 6**

Explain Conceptual Model for this Case

**Answer**

A conceptual model for the payment process done by a customer using net banking provides a high- level understanding of the key concepts and their relationships involved in the payment transaction. It helps in visualizing the overall structure and flow of the payment process.

* User Registration & Authentication: Users register and authenticate through secure credentials.
* Dashboard & Navigation: Upon login, users access a dashboard for account overviews and intuitive navigation.
* Account Management: Users view balances, transaction history, and perform account-related tasks like transfers and bill payments.
* Transaction History & Statements: Detailed transaction histories and downloadable statements aid in financial tracking.
* Fund Transfers & Payments: Users transfer funds between accounts, pay bills, and schedule transactions.
* Mobile & Online Banking: Accessible through mobile apps and online portals for convenient banking.
* Security Features: Robust security measures, including encryption and authentication methods, protect user data.
* Customer Support: Resources like FAQs, tutorials, and live support assist users with account queries and technical issues.
* Privacy & Compliance: Adherence to data privacy regulations ensures user privacy and compliance with legal standards.

**Question 7**

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

**Answer**

**MVC Architecture**

MVC stands for Model-View-Controller, which is a software architectural pattern commonly used for developing user interfaces. It divides the application into three interconnected components:

Model: The Model represents the application's data and business logic. It manages the data, logic, and rules of the application. In an MVC architecture, the model is independent of the user interface. It communicates with the database, processes data, and responds to queries from the controller.

View: The View represents the user interface components such as screens, forms, buttons, and other elements that users interact with. It displays the data from the model to the users and sends user actions (like button clicks or form submissions) to the controller for processing.

Controller: The Controller acts as an intermediary between the Model and the View components. It receives user input from the View, processes it (often by interacting with the Model), and updates the View accordingly. The Controller essentially handles the application's logic, decides how to handle user actions, and updates the Model and View as necessary.

**MVC Architecture Rules**

1. Combination of One Actor and an use case results in one Boundary class

2. Combination of Two Actors and an use case results in two Boundary classes

3. Combination of Three Actors and an use case results in Three Boundary classes and so on....

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

4. Use case will result in a controller class

5. Each Actor will result in one entity class

**Guidelines to place identified MVC Classes in a 3 Tier Architecture**

➢Place all Entity Classes in DB Layer

➢Place Primary Actor associated Boundary Class in Application Layer

➢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

**Question 8**

Explain BA contributions in project

**Answer**

|  |  |  |
| --- | --- | --- |
| **Stages** | **Activities** | **Artifacts & Resources** |
| **Pre Project** | Enterprise Analysis- SWOT Analysis, GAP Analysis, Market Research, Feasibility Study, Root Cause Analysis, Decision Analysis, Strategy Analysis, Enterprise Architectural Framework, Project Scope and Business case writing, Risk analysis | Business case SOW (Statement of work)PO (Purchase Order)Sr. BA, Business ArchitectsPre sales Consultant |
| **Planning & Estimations & 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 PM4. BA conducts stakeholders Analysis5. Plan BA approach strategy (Requirements gathering techniques, communication, Requirements management, Documents to follow, Tools to use, Change Request Handling methodology )for this Project | PMSr.BA |
| **Requirements 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 requirements4. Sort the gathered Requirements (avoiding duplicate Reqs , grouping into similar functionality or into modules)5. Prioritize requirements – MoSCoW6. Validate Requirements - FURPS | BRD(Business Requirements Document) |
| **Requirements Analysis** | 1. Draws UML Diagrams ( Usecase and Activity Diagrams)2. Prepares Functional Requirements from Business Requirements3. All Architects comes up with Technical Requirements (SSD)4. SRS will have Functional Requirements and Technical Requirements5. Takes Signoff on SRS from Client. SRS is the first legal binding Doc 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 SpecificationSSD(Supplementary Support Document)SRS (Software Requirements Specification)RTM (Requirements Traceability Matrix) |
| **Design** | 1. From Usecase 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 UAT)3. BA will initiate the preparation of End user manuals4. updates RTM 5. From Use case Diagram Solution-Architect recommends Architecture of the IT solution6. 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 DocumentDesign Document – HDD – ADD |
| **Coding** | 1.BA organizes JAD Sessions2. BA clarifies queries of Technical Team during Coding3. Developers refer Diagrams and Transient (Controller Classes) of BA and code their unit4. Update End user manuals5. Update RTM6. Conducts regular Status meetings with technical team and the Client and tuning Client for participation in UAT | LDD – CDDApplication |
| **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 BA from Client5. Updates End User Manuals6. Updates RTM7. Take signoff from Client on Client Project Acceptance form | Test Concerning DocumentsApplication with less errors |
| **Deployment and Implementation** | 1.Forwards RTM to Client or the PM which should be attached to the Project Closure Document2. Coordinates to complete and share End User Manuals3. Plans and Organizes Training Sessions for End Users4. Prepares Lessons learned from this project (to take precautions for coming projects) |  |

**Question - 9**

What is conflict management? Explain using Thomas – Kilmann technique

**Answer**

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

The Thomas-Kilmann Conflict Mode Instrument (TKI) is a widely used technique for understanding and managing conflict

* The 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 meet the needs of everyone.



The Thomas Kilmann Conflict Mode Instrument begins by identifying the two basic dimensions of Conflict Behavior:

•Assertiveness: The degree to which you try to satisfy your own concerns during a conflict. This is related to how you might try to meet your needs or receive support for your ideas.

•Cooperativeness: The degree to which you try to satisfy the other individuals’ concerns. It is related to how you might try to help the other individual meet his or her needs or how you can be receptive to the other individuals’ ideas

**5 Thomas Kilmann Conflicts Modes**

* Competing: In this mode, individuals pursue their own concerns at the expense of others. They assert their needs, opinions, and positions forcefully, often ignoring the needs and interests of others.
* Collaborating: Collaboration involves working together with others to find a mutually beneficial solution. Individuals in this mode try to satisfy the concerns of all parties involved. They invest time and effort in understanding others' perspectives and work toward creating win-win outcomes.
* Compromising: Compromising entails seeking a middle ground in which both parties give up something to reach a mutually acceptable solution. It involves moderate assertiveness and cooperativeness and may not fully satisfy either party's concerns.
* Avoiding: Avoidance involves sidestepping or postponing conflict situations altogether. Individuals in this mode often withdraw from the conflict, ignoring the issues or denying that a problem exists. They may choose to avoid confrontation to maintain harmony or reduce tension.
* Accommodating: Accommodating involves self-sacrifice, where individuals prioritize the concerns and needs of others over their own. They yield to others' viewpoints, showing empathy and support, even if it means neglecting their own interests.

**Question -10**

List down the reasons for project failure

**Answer**

* Improper requirement gathering
* Continuous change in requirements
* Lack of user involvement
* Lack of executive support
* Unrealistic expectation
* Improper planning

**Question -11.**

List the Challenges faced in projects for BA

**Answer**

* Changing Requirements
* Stakeholder Management
* Lack of Stakeholder Involvement
* Unclear Project Objectives
* Managing Conflicts and Negotiations
* Project Communication
* Time and Resource Constraints
* Resistance to Change

**Question – 12**

Write about Document Naming Standards

**Answer**

Document Naming Standards

All documents will be named using some standards like

ProjID-Doc Name- V[X]-D[Y].ext

PQ786-BRD-V1-D1.docx 1.1.

PQ786-BRD-V1-D2.docx 1.2.

PQ786-BRD-V1-D3.docx 1.3.

PQ786-BRD-V2-D1.docx 2.1.

**Question-13**

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

**Answer**

* 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**.**

**Question – 14**

Write the difference between packages and sub-systems

**Answer**

|  |  |  |
| --- | --- | --- |
| **Features** | **Packages** | **Subsystems** |
| Definition | Logical grouping of related classes/object | Collection of packages or components |
| Size | Smaller units of organization | Larger units of organization |
| Depencency | Manage dependencies at a class/component level. | 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 it will be used

**Answer**

Camel casing is a naming convention used in computer programming and is characterized by removing spaces between words and capitalizing the first letter of each word except for the first word. The name "camel casing" is derived from the appearance of the resulting string, which resembles the humps of a camel.

By using camel casing, developers can create meaningful and readable names that are easier to understand and follow coding standards. It promotes consistency within the codebase and improves collaboration among team members.

**Question -16**

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

**Answer**

A development server is a specialized computer or software environment dedicated to the development and testing of software applications or websites before they are deployed to a production environment. The purpose of a development server is to provide a controlled and isolated environment where developers can work on code changes, test new features, and identify and fix bugs without impacting the live or production environment.

Accesses for Business Analysts:

Business analysts (BAs) typically interact with the development server in several ways:

* Requirements Gathering: BAs collaborate with developers and other stakeholders to gather and document requirements for the software being developed. They may review user stories, use cases, and other documentation stored on the development server.
* Testing and Validation: BAs may access the development server to test new features and changes in the application against the documented requirements. They verify that the software meets the specified criteria and report any discrepancies or issues to the development team.
* Feedback and Collaboration: BAs provide feedback to developers based on their analysis and testing of the application. They may communicate through project management tools, issue trackers, or directly within the development environment.
* Access to Documentation: BAs may need access to design documents, technical specifications, and other documentation stored on the development server to understand the system architecture and functionality.
* Demonstrations and Reviews: BAs participate in demonstrations and reviews of the software on the development server to ensure that it aligns with business objectives and requirements.

**Question-17**

 What is Data Mapping

**Answer**

Data mapping is the process of establishing a relationship or connection between data elements in two or more different data sources or data formats. 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 processes.

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

**Question-18**

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

**Answer**

An API, or Application Programming Interface, is a set of protocols, tools, and definitions that allows different software applications to communicate and interact with each other. APIs define the methods and data formats that applications can use to request and exchange information, enabling seamless integration and interoperability between systems.

In the context of software development, APIs serve as intermediaries that enable developers to access the functionality and data of other applications or services without needing to understand the underlying implementation details. APIs abstract away the complexity of interacting with external systems, providing developers with standardized interfaces to perform specific tasks or access resources.

* Establish API Communication: Set up API communication between your application and the other application to exchange data
* Data Formatting: When sending date data from your application to the other application, convert the date from the dd-mm-yyyy format to the mm-dd-yyyy format. This can be achieved by extracting the day, month, and year components from the date and rearranging them according to the target format.
* Data Parsing: When receiving date data from the other application, parse the mm-dd-yyyy formatted date into your application's dd-mm- yyyy format. Again, you will need to extract the day, month, and year components and rearrange them accordingly.
* Data Validation: Perform data validation and ensure that the converted date remains valid after the format conversion. Check for edge cases, such as invalid dates or date ranges that might be affected by the format conversion, and handle them appropriately.