Capstone Project 3 Part 1

A customer can make a payment either by Card or by Wallet or by Cash or by Net banking.

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

**Answer**



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

**Answer**

Boundary Classes:

* Interface directly with external actors (users, systems)
* Named with suffixes like Form, Page, Interface, UI
* Examples: PaymentOptionBoundary, CardPaymentBoundary

Controller Classes:

* Coordinate between boundary and entity classes
* Handle business logic and workflows
* Named with suffixes like Controller, Manager, Handler
* Examples: PaymentInitiatedController, CardPaymentController

Entity Classes:

* Represent core business objects and data
* Mirror database tables/documents
* Named as nouns without suffixes
* Examples: Customer, Payment

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

**Answer**

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

User Layer

Business Logic Layer

Data Tier

Each layer operates independently and communicates only with adjacent layers, promoting modularity, scalability, and maintainability.

User Layer:

PaymentMethodSelectionBoundary

CardPaymentBoundary

Business Logic Layer:

PaymentController

WalletController

Data Tier:

Customer (Entity Class)

Payment (Entity Class)

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

**Answer**

A Domain Model is a conceptual representation that defines the structure, relationships, and behaviours of entities within a specific problem domain.

A domain model for customer net banking payments would include:

Core Entities:

- Customer (with attributes like ID, name, account details)

- Bank Account (balance, account number, type)

- Transaction (amount, timestamp, status)

- Payment Gateway (handles secure communication)

Key Relationships:

- Customer has one or more Bank Accounts

- Customer initiates Transactions

- Transaction is processed through Payment Gateway

- Bank Account is debited/credited through Transaction

Business Rules:

- Customer must authenticate before payment

- Account must have sufficient balance

- Transaction requires 2-factor verification

- Payment gateway validates transaction details

- System maintains audit trail

This represents the key objects, their relationships, and rules governing online banking payments.



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

**Answer**

A sequence diagram is a visual representation of how a group of objects interact with each other over time. It's a type of Unified Modelling Language (UML) diagram. Software developers and business professionals use sequence diagrams to understand requirements for new systems or document existing processes

**Question 6 Explain Conceptual Model for this Case**

**Answer**

A conceptual model is a high-level representation of a system that helps in understanding, visualising, and communicating the essential aspects of a domain.

It provides a clear and simplified view of the domain, making it easier to understand.

Key elements of a conceptual model are:

* Entities: Customer, Product, Order and Payment
* Attributes: Customer ID, Name, Email, Phone number
* Relationships:

Customer initiates Payment

Payment is processed through Payment Method

Net Banking links to Bank Account

Bank Account belongs to Customer

* Business Rules:

Each payment requires customer authentication

Payment amount must be valid

Customer must have sufficient funds

System records transaction details

Payment status tracked throughout process

This model provides a high-level view of the domain's key elements and how they interact in the payment flow.

**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) framework is an architectural pattern that separates an application into three main logical components Model, View, and Controller.

View- Represents the presentation layer of the application.

Model- Represents The data and the business logic of the application.

Controller- Acts as an intermediary between Model and View.

MVC Architecture rules guidelines to place identified MVC Classes in a 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.

4. If governing body influence or reusability is there with any of remaining

3-Tier Architecture Placement:

Presentation Tier (View):

* UI components (Forms, Pages)
* Input validation
* User interaction handlers
* Display formatting

Business Tier (Controller + Some Model):

* Business logic implementation
* Transaction management
* Workflow orchestration
* Data validation rules
* Service interfaces

Data Tier (Model):

* Data access objects
* Database interactions
* Data persistence
* Entity definitions

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

**Answer**

|  |  |  |
| --- | --- | --- |
| **Stages** | **Activities** | **Artifacts and Resources** |
| **Pre-project** | -Feasibility analysis-Stakeholder identification-Initial scope assessment-Business case development | -Business case-Project charter-Feasibility reports-Stakeholder matrix |
| **Planning** | -Project charter review-Resource planning-Timeline estimation-Risk assessment documentation |
| **Project Initiation** | -Kick-off meeting coordination-Stakeholder analysis-Communication plan-Project scope document | -Scope document-Communication plan-Project schedule-Risk register |
| **Requirement Gathering** | -Stakeholder interviews-Workshops facilitation-Document review-Process mapping-Requirements documentation |
| **Requirement Analysis** | -Use case development-Business rules documentation-Gap analysis-Requirements validation-Functional specifications | -BRD/FRD-Use cases-Process flows-Requirements matrix |
| **Design** | -Review technical design-Interface specifications-Data mapping documents-Workflow diagrams-Process flows | -Interface specifications-Data models-Wireframes-System architecture |
| **Development** | -Clarify requirements-Review prototypes-Change request analysis-Documentation updates-Status reporting | -Change requests-Status reports-Technical specifications-Configuration docs |
| **Testing** | -Test case review-Defect analysis-Requirements traceability-Test scenario validation-UAT planning | -Test cases-Test plans-Defect logs-Sign-off documentsPeople:-Stakeholders-SMEs-Technical team-Project manager-End usersTools:-JIRA/Rally-Modelling tools-Documentation tools-Testing tools-Collaboration platforms |
| **UAT** | -Test plan creation-User training materials-UAT execution support-Defect tracking-Sign-off coordination |

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

**Answer**

Conflict Management is the process of resolving conflicts or disagreements between individuals or groups in a constructive manner.

Thomas-Kilmann technique is a widely used tool for assessing conflict resolution styles and guiding individuals in selecting appropriate strategies to manage conflicts.

5 steps of management are:

1. Identify the conflict

2. Discuss the details

3. Agree with the root problem

4. Check for every possible solution for the conflict

5. Negotiate the solution to avoid future conflicts

Thomas-Kilmann Conflict Management Technique identifies 5 approaches:

1. Competing (Assertive & Uncooperative)

- Standing firm on position

- Used when quick decisions needed

- Best for emergencies/critical issues

2. Accommodating (Unassertive & Cooperative)

- Yielding to others' views

- Maintains relationships

- Used when issue matters more to others

3. Avoiding (Unassertive & Uncooperative)

- Postponing or withdrawing

- Used for trivial issues

- When confrontation cost exceeds benefit

4. Collaborating (Assertive & Cooperative)

- Finding win-win solutions

- Best for complex issues

- When relationships and outcome both matter

5. Compromising (Moderate Assertiveness & Cooperation)

- Finding middle ground

- Used when time is limited

- When collaboration fails

Choice depends on:

- Importance of issue

- Time constraints

- Relationship value

- Power dynamics

**Question 10 List down the reasons for project failure**

**Answer**

- Poor planning: Unrealistic timelines, Inadequate resource allocation, Poor risk management

- Unclear objectives and requirements: Changing requirements

- Inadequate risk management: Lack of leadership support, Poor change management, Inadequate monitoring

- Poor communication: Stakeholder misalignment, Unclear expectations, Information silos

- Scope creep

- Lack of stakeholder engagement

- Resource constraints: Skill gaps, Budget shortfalls, Staff turnover

- Technical challenges: Technology limitations, Integration issues, Technical debt

- External Factors: Market changes, Regulatory compliance, Vendor dependencies

- Quality Problems: Insufficient testing, Poor documentation, Defect management issues

**Question 11 List the Challenges faced in projects for BA**

**Answer**

* Unclear or changing requirements
* Managing stakeholder expectations
* Scope creep and scope management
* Time and resource constraints
* Quality assurance and testing
* Documentation and knowledge management
* Technology constraints and complexity

**Question 12 List the Challenges faced in projects for BA**

**Answer**

A document numbering standard is a systematic approach to assigning unique identifiers to various documents created and used throughout the development process.

Example:

Suppose we have a project with the ID ‘’PROJ123’’, and we are working with a requirements specification document.

Project ID: PROJ123

Document type: REQ

Version: 1.0

Date: 2025-02-07

The document identifier could be: PROJ123-REQ-1.0-2025-02-07

**Question 13 What are the Do’s and Don’ts of a Business analyst?**

**Answer**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **DO’S** | **DON’TS** |
| **1** | Consult a SME for clarifications in requirements. | Never say NO to the client. |
| **2** | Go to the client with a plain mind with no assumptions. Listen carefully and completely until the client is done, and then you can ask questions. | There is no word as ‘’By Default’’. |
| **3** | Try to extract maximum leads to the solution form the client himself. | Never imagine anything in terms of GUI. |
| **4** | Concentrate on the important requirements. | Don’t interrupt the client when he is giving you the problem. |
| **5** | Question the existence of existence/ question everything. | Never try to give solutions to the client straight away with your previous experience and assumptions. |

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

**Answer**

Packages serve as organizational units that group related elements together within a codebase. They provide a way to manage code dependencies and avoid naming conflicts, much like organizing files into folders.

Subsystems, in contrast, function as larger, more comprehensive components that deliver complete business functionality. They operate with a higher degree of independence and typically encompass multiple packages working together.

Key Differences:

* Scope: Packages organize code, subsystems provide complete functionality
* Independence: Packages are dependent, subsystems can work independently
* Complexity: Packages are simpler, subsystems are more complex
* Structure: Packages group elements, subsystems contain multiple packages
* Interface: Packages have loose coupling, subsystems have defined interfaces

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

**Answer**

Camel casing is a naming convention in programming and documentation where compound words or phrases are written without spaces, and each word after the first begins with a capital letter.

There are two main types of camel casing:

Lower Camel Case (camelCase):

* First word starts with lowercase
* Used for:

 - Variable names (firstName, customerAddress)

 - Method names (calculateTotal, getUserDetails)

 - Property names (emailAddress, phoneNumber)

Upper Camel Case/Pascal Case (PascalCase):

* First word starts with uppercase
* Used for:

 - Class names (CustomerOrder, PaymentGateway)

 - Interface names (IPayable, IDisposable)

 - Type names (DataStructure, BusinessLogic)

This convention enhances code readability and maintainability by making compound names easily readable while eliminating spaces that aren't allowed in most programming contexts. The consistent use of camel casing helps maintain clean, professional, and standardized code across development teams.

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

**Answer**

A development server is a specialized environment where software applications are built and tested before deployment to production.

Development Server Environment:

The development server provides a controlled environment that mirrors production settings while allowing for testing and modification.

 Business Analyst Access Levels:

Business Analysts generally receive restricted access to the development server, tailored to their role requirements.

Key Access Areas for BAs:

The BA usually has access to test environments where they can validate requirements implementation, review user interfaces, and verify business logic.

Access Restrictions:

BAs generally cannot modify source code, change server configurations, or access sensitive production data. Their access is limited to areas necessary for requirements validation and testing activities.

**Question 17 What is Data Mapping?**

**Answer**

Data mapping is the process of creating relationships and correlations between different data elements across various systems, databases, or data models. It serves as a crucial foundation for data integration, migration, and transformation activities within organizations.

The process encompasses several key components. First, it requires understanding the source and target data structures thoroughly. Second, it involves establishing clear rules for how data should be transformed or modified during transfer. Third, it includes defining validation rules to ensure data quality and consistency.

Business Analysts often create data mapping documents that serve as blueprints for technical teams implementing data integrations or migrations. These documents typically include source and target field names, data types, transformation rules, and validation requirements, ensuring clear communication between business and technical stakeholders about data handling requirements.

**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 (Application Programming Interface) is a set of rules and protocols that allows different software applications to communicate and share data with each other. It defines the methods and data formats that applications can use to request and exchange information.

For handling date format differences between applications, here's how the API integration would work:

Data Flow and Transformation:

When receiving data from the US application (mm-dd-yyyy) to your application (dd-mm-yyyy), the API implementation would include a date format conversion process. This ensures data consistency and accuracy across systems.

Implementation Approach:

The API would include date format handling logic in the integration layer. When receiving a date like "02-14-2024" from the US system, it would parse this date and transform it to "14-02-2024" before storing it in your system. Similarly, when sending data back, it would convert dates from your format to the US format.

This approach ensures reliable data exchange while maintaining the integrity of date information across different regional formats.