**PREP 2**

Question 1 – Write Agile Manifesto – 8 Marks

The Agile Manifesto is a formal proclamation of four key values and twelve principles to guide an iterative and people-centric approach to software development. It states:

Values:

1. Individuals and interactions over processes and tools: This value emphasizes the importance of people working together and communicating effectively, rather than relying solely on formal processes and tools.

2. Working software over comprehensive documentation: While documentation is important, the primary focus should be on producing functioning software that provides value to the customer.

3. Customer collaboration over contract negotiation: Agile teams prioritize working closely with customers to understand and meet their needs, rather than focusing on strict contractual agreements.

4. Responding to change over following a plan: Agile teams embrace change and are adaptable, recognizing that requirements and priorities may evolve over the time.

Principles:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

4. Business people and developers must work together daily throughout the project.

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

7. Working software is the primary measure of progress.

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9. Continuous attention to technical excellence and good design enhances agility.

10. Simplicity—the art of maximizing the amount of work not done—is essential.

11. The best architectures, requirements, and designs emerge from self-organizing teams.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Question 2: User Stories- Acceptance Criteria-BV-CP – 40 Marks

Write a minimum of 40 User stories and their Acceptance Criteria along with their BV and CP

Sprint 1:

| User Story No: 1 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A CUSTOMER I WANT TO REGISTER AN ACCOUNT SO THAT I CAN START ORDERING FOOD |
| BV: 300 | CP: 2 |
| Acceptance Criteria: Registration screen with fields for username, password, email, and phone number. Successful registration notification |

Sprint 2:

| User Story No: 2 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A CUSTOMER I WANT TO LOG IN TO MY ACCOUNT SO THAT I CAN ACCESS MY ORDERS AND PROFILE |
| BV: 300 | CP: 2 |
| Acceptance Criteria: Login screen with fields for username and password. Successful login redirects to the dashboard |

Sprint 3:

| User Story No: 3 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A CUSTOMER I WANT TO SEARCH FOR RESTAURANTS SO THAT I CAN FIND MY PREFERRED DINING OPTIONS |
| BV: 250 | CP: 3 |
| Acceptance Criteria: Search bar functionality with autocomplete and filters (cuisine, rating, distance) |

Sprint 4:

| User Story No: 4 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A CUSTOMER I WANT TO VIEW A RESTAURANT'S MENU SO THAT I CAN DECIDE WHAT TO ORDER |
| BV: 250 | CP: 2 |
| Acceptance Criteria: Menu page displaying categories and items with prices and descriptions |

Sprint 5:

| User Story No: 5 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A CUSTOMER I WANT TO ADD ITEMS TO MY CART SO THAT I CAN PLACE AN ORDER |
| BV: 200 | CP: 3 |
| Acceptance Criteria: Add to cart button on menu items. Cart page displays selected items with quantities and total price |

Sprint 6:

| User Story No: 6 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A CUSTOMER I WANT TO PLACE AN ORDER SO THAT I CAN RECEIVE FOOD DELIVERY |
| BV: 400 | CP: 5 |
| Acceptance Criteria: Checkout process with payment options (credit card, UPI). Order confirmation message |

Sprint 7:

| User Story No: 7 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A CUSTOMER I WANT TO TRACK MY ORDER SO THAT I KNOW ITS STATUS |
| BV: 200 | CP: 3 |
| Acceptance Criteria: Order tracking page displaying status updates (preparing, out for delivery, delivered) |

Sprint 8:

| User Story No: 8 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A CUSTOMER I WANT TO RATE AND REVIEW MY ORDER SO THAT I CAN SHARE MY EXPERIENCE |
| BV: 100 | CP: 2 |
| Acceptance Criteria: Rating and review form linked to past orders. Submit button with a confirmation message |

Sprint 9:

| User Story No: 9 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A CUSTOMER I WANT TO CANCEL MY ORDER SO THAT I CAN CHANGE MY MIND OR RECTIFY A MISTAKE |
| BV: 150 | CP: 2 |
| Acceptance Criteria: Cancel button on active orders. Confirmation prompt and cancellation notification |

Sprint 10:

| User Story No: 10 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A CUSTOMER I WANT TO SAVE MY FAVORITE RESTAURANTS SO THAT I CAN ACCESS THEM QUICKLY |
| BV: 100 | CP: 2 |
| Acceptance Criteria: Favorite button on restaurant pages. Favorites list on user profile |

Sprint 11:

| User Story No: 11 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A CUSTOMER I WANT TO SAVE MY DELIVERY ADDRESSES SO THAT I CAN REUSE THEM FOR FUTURE ORDERS |
| BV: 150 | CP: 3 |
| Acceptance Criteria: An address book in the user profile. Add, edit, and delete address functionality |

Sprint 12:

| User Story No: 12 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A CUSTOMER I WANT TO CHAT WITH CUSTOMER SUPPORT SO THAT I CAN RESOLVE ISSUES WITH MY ORDER |
| BV: 200 | CP: 4 |
| Acceptance Criteria: Chat window accessible from the help section. Real-time messaging with customer support agents |

Sprint 13:

| User Story No: 13 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A RESTAURANT OWNER I WANT TO REGISTER MY RESTAURANT SO THAT I CAN START RECEIVING ORDERS |
| BV: 300 | CP: 3 |
| Acceptance Criteria: Registration form with fields for restaurant name, address, cuisine type, and contact details.Approval process by admin |

Sprint 14:

| User Story No: 14 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A RESTAURANT OWNER I WANT TO LOG IN TO MY ACCOUNT SO THAT I CAN MANAGE MY RESTAURANT'S PROFILE AND ORDERS |
| BV: 200 | CP: 2 |
| Acceptance Criteria: Login screen with fields for username and password. Successful login redirects to the restaurant dashboard. |

Sprint 15:

| User Story No: 15 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A RESTAURANT OWNER I WANT TO MANAGE MY MENU SO THAT I CAN UPDATE ITEMS AND PRICES |
| BV: 250 | CP: 4 |
| Acceptance Criteria: Menu management page with add, edit, and delete functionality for items. |

Sprint 16:

| User Story No: 16 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A RESTAURANT OWNER I WANT TO VIEW INCOMING ORDERS SO THAT I CAN PREPARE AND DELIVER THEM |
| BV: 300 | CP: 3 |
| Acceptance Criteria: Orders list with status (new, in progress, completed). Details page for each order. |

Sprint 17:

| User Story No: 17 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A RESTAURANT OWNER I WANT TO UPDATE THE STATUS OF AN ORDER SO THAT CUSTOMERS KNOW ITS PROGRESS |
| BV: 200 | CP: 2 |
| Acceptance Criteria: Status update buttons (preparing, out for delivery, delivered) on the order details page |

Sprint 18:

| User Story No: 18 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A RESTAURANT OWNER I WANT TO MANAGE MY RESTAURANT'S PROFILE SO THAT I CAN KEEP MY INFORMATION UP TO DATE |
| BV: 150 | CP: 2 |
| Acceptance Criteria: Profile management page with fields for contact information, address, and operating hours |

Sprint 19:

| User Story No: 19 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A DELIVERY BOY I WANT TO REGISTER SO THAT I CAN START DELIVERING ORDERS |
| BV: 200 | CP: 2 |
| Acceptance Criteria: Registration form with fields for name, contact information, and vehicle details. Approval process by admin |

Sprint 20:

| User Story No: 20 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A DELIVERY BOY I WANT TO LOG IN TO MY ACCOUNT SO THAT I CAN VIEW AND ACCEPT DELIVERY TASKS |
| BV: 150 | CP: 2 |
| Acceptance Criteria: Login screen with fields for username and password. Successful login redirects to the delivery dashboard |

Sprint 21:

| User Story No: 21 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A DELIVERY BOY I WANT TO VIEW AVAILABLE DELIVERY TASKS SO THAT I CAN SELECT AND COMPLETE THEM |
| BV: 250 | CP: 3 |
| Acceptance Criteria: List of available deliveries with details (pickup location, destination, order contents). Accept button for tasks |

Sprint 22:

| User Story No: 22 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A DELIVERY BOY I WANT TO UPDATE THE STATUS OF A DELIVERY SO THAT CUSTOMERS AND RESTAURANTS ARE INFORMED |
| BV: 200 | CP: 2 |
| Acceptance Criteria: Status update buttons (picked up, in transit, delivered) on delivery details page |

Sprint 23:

| User Story No: 23 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A DELIVERY BOY I WANT TO VIEW MY PAST DELIVERIES SO THAT I CAN TRACK MY PERFORMANCE AND EARNINGS |
| BV: 100 | CP: 2 |
| Acceptance Criteria:  List of completed deliveries with details (date, earnings, feedback). Filter by date |

Sprint 24:

| User Story No: 24 | Tasks: 2 | Priority: Low |
| --- | --- | --- |
| AS A DELIVERY BOY I WANT TO UPDATE MY PROFILE SO THAT MY CONTACT AND VEHICLE INFORMATION ARE CURRENT |
| BV: 100` | CP: 1 |
| Acceptance Criteria: Profile management page with fields for name, contact information, and vehicle details |

Sprint 25:

| User Story No: 25 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A REGIONAL ADMIN I WANT TO MANAGE RESTAURANTS IN MY REGION SO THAT I CAN ENSURE QUALITY AND COMPLIANCE |
| BV: 300 | CP: 3 |
| Acceptance Criteria: List of restaurants with approve/reject buttons for new registrations. Status update functionality. |

Sprint 26:

| User Story No: 26 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A REGIONAL ADMIN I WANT TO MANAGE DELIVERY BOYS IN MY REGION SO THAT I CAN ENSURE ADEQUATE SERVICE COVERAGE |
| BV: 250 | CP: 3 |
| Acceptance Criteria: List of delivery boys with approve/reject buttons for new registrations. Status update functionality |

Sprint 27:

| User Story No: 27 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A REGIONAL ADMIN I WANT TO VIEW REGIONAL PERFORMANCE REPORTS SO THAT I CAN MONITOR THE BUSINESS METRICS |
| BV: 400 | CP: 4 |
| Acceptance Criteria: Reports dashboard with filters for date range and metrics (orders, revenue, customer feedback) |

Sprint 28:

| User Story No: 28 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A SUPER ADMIN I WANT TO MANAGE ALL USERS IN THE SYSTEM SO THAT I CAN ENSURE SMOOTH OPERATIONS |
| BV: 500 | CP: 5 |
| Acceptance Criteria: List of users (customers, restaurant owners, delivery boys, regional admins) with search and filter options. Status update functionality |

Sprint 29:

| User Story No: 29 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A SUPER ADMIN I WANT TO MANAGE SYSTEM SETTINGS SO THAT I CAN CONFIGURE APPLICATION PARAMETERS |
| BV: 300 | CP: 4 |
| Acceptance Criteria: Settings page with options for configuration (notification settings, payment gateways, API keys) |

Sprint 30:

| User Story No: 30 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A SUPER ADMIN I WANT TO VIEW GLOBAL PERFORMANCE REPORTS SO THAT I CAN ASSESS THE OVERALL BUSINESS HEALTH |
| BV: 500 | CP: 4 |
| Acceptance Criteria: Global reports dashboard with filters for date range and metrics (orders, revenue, customer feedback) |

Sprint 31:

| User Story No: 31 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A SUPER ADMIN I WANT TO SET REGIONAL TARGETS SO THAT REGIONAL ADMINS CAN WORK TOWARD SPECIFIC GOALS |
| BV: 300 | CP: 3 |
| Acceptance Criteria: Target setting functionality with fields for region, metric, and target value. Notification system for target updates |

Sprint 32:

| User Story No: 32 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A CUSTOMER I WANT TO RECEIVE NOTIFICATIONS ABOUT MY ORDER STATUS SO THAT I AM UPDATED IN REAL-TIME |
| BV: 200 | CP: 3 |
| Acceptance Criteria: Notification system that sends alerts (email, SMS, app notification) for order status changes |

Sprint 33:

| User Story No: 33 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A CUSTOMER I WANT TO APPLY DISCOUNT CODES SO THAT I CAN SAVE MONEY ON MY ORDERS |
| BV: 150 | CP: 2 |
| Acceptance Criteria: Discount code field on the checkout page. Validation of codes and application of discounts |

Sprint 34:

| User Story No: 34 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A RESTAURANT OWNER I WANT TO VIEW CUSTOMER FEEDBACK SO THAT I CAN IMPROVE MY SERVICE |
| BV: 200 | CP: 2 |
| Acceptance Criteria: Feedback section in the restaurant dashboard with customer ratings and comments |

Sprint 35:

| User Story No: 35 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A DELIVERY BOY I WANT TO RECEIVE ROUTE GUIDANCE SO THAT I CAN DELIVER ORDERS EFFICIENTLY |
| BV: 300 | CP: 3 |
| Acceptance Criteria: Integration with maps for route guidance. Display of pickup and delivery addresses with route options |

Sprint 36:

| User Story No: 36 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A REGIONAL ADMIN I WANT TO COMMUNICATE WITH RESTAURANT OWNERS AND DELIVERY BOYS IN MY REGION SO THAT I CAN COORDINATE OPERATIONS |
| BV: 250 | CP: 3 |
| Acceptance Criteria: Messaging system integrated into the admin dashboard. Contact lists for restaurants and delivery boys |

Sprint 37:

| User Story No: 37 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A SUPER ADMIN I WANT TO GENERATE CUSTOM REPORTS SO THAT I CAN ANALYZE SPECIFIC BUSINESS METRICS |
| BV: 400 | CP: 4 |
| Acceptance Criteria: Report generation tool with customizable parameters (date range, region, metrics). Export functionality (PDF, CSV) |

Sprint 38:

| User Story No: 38 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A CUSTOMER I WANT TO SAVE MY PAYMENT METHODS SO THAT I CAN CHECKOUT QUICKLY IN THE FUTURE |
| BV: 200 | CP: 3 |
| Acceptance Criteria: Payment methods section in the user profile. Add, edit, and delete functionality for payment options |

Sprint 39:

| User Story No: 39 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A CUSTOMER I WANT TO VIEW ORDER HISTORY SO THAT I CAN REORDER MY FAVORITE MEALS |
| BV: 200 | CP: 2 |
| Acceptance Criteria: Order history page with details of past orders and reorder button |

Sprint 40:

| User Story No: 40 | Tasks: 2 | Priority: Highest |
| --- | --- | --- |
| AS A CUSTOMER I WANT TO RECEIVE RECOMMENDATIONS BASED ON MY ORDER HISTORY SO THAT I CAN DISCOVER NEW DISHES AND RESTAURANTS |
| BV: 250 | CP: 3 |
| Acceptance Criteria: Recommendation section on the dashboard with personalized suggestions |

Question 3 – What is epic? Write 2 epics – 5 Marks

In agile software development, an "Epic" is a large body of work that can be broken down into smaller, more manageable user stories that are used to group related user stories together, and they help in planning and prioritizing work for the development team.

The idea is to break the work down into shippable pieces so that software products or any other type of product can get done and you can continue to ship value to your customer, regularly broken down epics into smaller user stories allows for more detailed planning and implementation, as well as providing a clearer understanding of the work.

For example, if you were developing a food delivery application, an Epic could be something like a "Payment Processing System" which would encompass all the user stories related to handling payments, including features like selecting payment methods, processing transactions, and generating payment reports.

Examples of Epics:

Epic 1: User Account Management

1. AS A USER

I WANT TO BE ABLE TO REGISTER ON THE SCRUM FOODS APP

SO THAT I CAN CREATE A PERSONALIZED ACCOUNT.

1. AS A USER

I WANT TO LOG IN SECURELY

SO THAT I CAN ACCESS MY ACCOUNT AND THE APP'S FEATURES.

1. AS A USER

I WANT THE ABILITY TO RESET MY PASSWORD

SO THAT IN CASE I FORGET IT I SHALL BE ABLE TO RETRIEVE IT.

Acceptance criteria:

1. The registration form includes fields for username, password, email, and other necessary details. The system generates a confirmation message upon successful registration.
2. The user is directed to the home page of the app on successful login.
3. The user receives an email with a reset password link.

Epic 2: Order Processing

1. AS A USER

I WANT TO BROWSE AVAILABLE RESTAURANTS AND VIEW THEIR MENUS

SO THAT I CAN CHOOSE WHAT TO ORDER.

1. AS A USER

I WANT TO ADD ITEMS TO MY CART

SO THAT I CAN PLACE ORDERS EASILY.

1. AS A USER

I WANT REAL-TIME TRACKING OF THE ORDER

SO THAT I’M AWARE OF THE ORDER STATUS.

Acceptance criteria:

1. The user can see all the available restaurants and their menus.
2. The user can choose food items from the menu and can see them in the cart upon selection.
3. Upon successful placement of the order, the user can track the order in real-time for its status.

Question 4 – What is the difference between BV and CP – 2 Marks

Business Value (BV):

It is a measure of how important a feature is to the business. It reflects the potential benefit or value that the feature will bring to the business once implemented.

Complexity Points (CP):

It is also known as Story Points (SP), representing the effort required to implement a feature. It includes the time and complexity involved in developing the feature.

Here are the key differences between BV and CP:

Business Value (BV) - It is a measure of the importance and effort required for achieving or implementing a user story from the perspective of the stakeholders or business. It indicates how much the feature or functionality is valued by the users or how much it aligns with the business goals and objectives. Business Value is typically measured in abstract units that represent the importance or priority of a user story relative to other stories. The units are often specific to the project or organization and may not have a standardized scale and will vary from company to company.

Business Value is often determined through discussions with stakeholders, product owners, and other relevant parties.

Techniques - Moscow, Currency notes (denominations like 500,200,100 where 500 being the highest)

Complexity Points (CP) - CP, or Story Points, represent the relative effort or complexity involved in implementing a user story. It is a measure of the work required to develop a feature. Complexity Points are also abstract units but represent the effort or complexity involved in implementing a user story. They are relative and specific to the development team's understanding of the work.

Complexity Points are assigned by the development team during the sprint planning process.

Techniques - Planning Poker(Fibonacci series for initial numbers - where the sum of two consecutive numbers will be the next number, 1+2=3 or 2+3=5) or other consensus-based methods to collectively estimate the effort required.

Question 5 – Explain about Sprint – 5 Marks

A Sprint is a time-boxed period, typically 2-4 weeks, during which a Scrum team works to complete a set of user stories, features, or tasks from the product backlog. The goal of each sprint is to create a potentially shippable product increment.

Sprint Characteristics:

* Duration: A Sprint typically lasts 2-4 weeks, with 2 weeks being the most common duration. The length of a Sprint is determined by the team based on factors like the complexity of the work, the level of uncertainty, and the organization's preferences.
* Sprint Goal: Each sprint has a specific goal or objective. This goal is set by the Product Owner and provides a clear focus for the team's work during the sprint. It helps guide decision-making and prioritization.
* Sprint Planning: At the beginning of a sprint, the Product Owner presents the highest-priority user stories from the product backlog to the development team. The team selects a subset of these stories to work on during the sprint based on their capacity and the Sprint Goal.
* Daily Scrum Meetings: Also known as stand-ups, these are daily 15-minute meetings where team members discuss what they did the previous day, what they plan to do today, and any obstacles they are facing.
* Sprint Review: At the end of the sprint, the team presents the work they've completed to stakeholders for feedback and to provide transparency on progress.
* Sprint Retrospective: After the review, the team discusses what went well, what didn't, and how processes can be improved for the next sprint.

Question 6 – Explain Product backlog and sprint backlog – 5 Marks

Product Backlog:

The product backlog is a dynamic, prioritized list of all the features, enhancements, bug fixes, tasks, and technical debt required for a product. It is essentially the master to-do list for the product, managed and owned by the Product Owner. The product backlog is constantly evolving based on feedback from stakeholders, changes in market conditions, and the team’s progress.

Key characteristics of the product backlog:

* Prioritized: Items are ordered by priority, with the most valuable or urgent items at the top.
* Detailed appropriately: Higher priority items are more detailed and ready for implementation, while lower priority items are less detailed.
* Dynamic: The backlog is continuously refined and updated.
* Visible: It is accessible to all stakeholders and team members to ensure transparency.

Sprint Backlog:

The sprint backlog is a subset of the product backlog that the development team commits to completing during a specific sprint. It includes selected product backlog items and a detailed plan for delivering the product increment. The sprint backlog is owned by the development team and is their working plan for the sprint.

Key characteristics of the sprint backlog:

* Committed: The team commits to completing the items in the sprint backlog during the sprint.
* Detailed: Each item is broken down into tasks that are estimated and assigned to team members.
* Fixed for the sprint: Once the sprint begins, the sprint backlog is not changed unless absolutely necessary.
* Progress tracked: The team tracks progress through daily stand-ups and updates the sprint backlog accordingly.

Differences and Relationship:

* Scope: The product backlog encompasses the entire scope of the project, while the sprint backlog focuses on a specific sprint.
* Ownership: The product backlog is managed by the Product Owner, while the sprint backlog is owned by the development team.
* Detail level: Product backlog items can vary in detail, with high-priority items being more refined. In contrast, sprint backlog items are fully detailed with clear tasks.
* Flexibility: The product backlog is continuously refined and updated, while the sprint backlog remains fixed for the duration of the sprint unless changes are agreed upon by the team.

Question 7 – What is impediments log? Write 2 impediments – 5 Marks

An Impediments Log is a tool used in Scrum and other agile frameworks to track and manage obstacles that hinder the progress of the development team. These impediments can be anything that slows down or blocks the team's ability to complete their tasks and deliver product increments. The Scrum Master is primarily responsible for identifying, documenting, and addressing these impediments to ensure that the team can maintain its productivity and velocity.

Characteristics of an Impediments Log:

* Visibility: The impediments log should be visible to the entire team and relevant stakeholders to ensure transparency.
* Regularly Updated: The log should be updated frequently, typically during daily stand-ups or whenever new impediments are identified.
* Prioritized: Impediments should be prioritized based on their impact on the team's progress. High-impact impediments should be addressed first.
* Action-Oriented: Each impediment should have an associated action plan or strategy for resolution.

Example 1: Technical Impediment

* Description: Insufficient documentation for third-party API integration.
* Impact: Integration tasks are stalled, affecting the sprint timeline.
* Priority: High
* Assigned to: Scrum Master
* Status: In Progress
* Action Taken: The Scrum Master has contacted the third-party provider to request detailed documentation and has scheduled a support call to resolve the issues.
* Resolution: Pending support call with third-party provider.

Example 2: Resource Impediment

* Description: Key team member on unexpected leave.
* Impact: Tasks assigned to this team member are delayed, risking the completion of sprint goals.
* Priority: Medium
* Assigned to: Scrum Master and Team Lead
* Status: Resolved
* Action Taken: The Scrum Master reallocated tasks among available team members, and the Team Lead provided additional support for critical tasks.
* Resolution: Tasks were redistributed and completed on time, with temporary support from the Team Lead.

Question 8: Explain Velocity of the Team (1 Mark)

Velocity refers to the measure of the amount of work a development team can complete during a sprint. The number of hours the development team will allocate to this sprint. This is calculated by scrum master.

It is a metric used to understand the team's pace of work and helps in predicting how much work can be completed in future sprints. The development team has more than 1 sprint under them so to calculate the velocity of the team we need to check the number of developers working on it.

For eg - Lets consider the Login feature for the app where the development team is working for the past sprints.

Each sprint will have a story point for completing user stories and will learn from the previous sprints if any challenges are faced. To calculate the velocity - each story point from each sprint will be taken and will be divided by the number of developers.

Question 9: Draw Sprint Burn Charts and Product Burn Down Charts (3 Marks)

Sprint Burnup Chart



Sprint Burndown Chart



Question 10: Explain about Product Grooming (2 Marks)

Product Grooming:

Also known as Backlog Grooming or Refinement, is an important activity in Agile and Scrum methodologies. It involves reviewing, prioritizing, and refining items in the product backlog to ensure they are well-prepared for upcoming sprints. The primary goal of Product Grooming is to have a clear, well-defined set of user stories or product backlog items (PBIs) that are ready for the development team to work on in the next

sprint.

Once the sprint starts, Product grooming activity can happen when there is a change request, enhancements,user stories are not well defined.

How is it conducted?

* Frequency: Product Grooming is an ongoing process that occurs regularly throughout the project. It's typically scheduled as a recurring meeting, often held once per sprint, and involves relevant stakeholders, including the Product Owner and development team. It is a collaborative effort of the scrum team and stakeholders.
* Participants:
* Product Owner: The Product Owner is central to the grooming process. They lead the session, provide guidance on priorities, and clarify user stories.
* Development Team: The team members participate to gain a deep understanding of upcoming work and to estimate the complexity of user stories.
* Scrum Master: The Scrum Master may facilitate the grooming session, ensuring that it remains focused and time-boxed.
* Agenda:
* Review of User Stories: The Product Owner presents user stories from the product backlog to the team. These can be newly added items or ones that need further refinement.
* Clarification and Details: The team discusses each user story in detail, seeking clarification from the Product Owner. They may ask questions about acceptance criteria, business rules, and dependencies.
* Estimation: The development team provides complexity point estimates for user stories. This helps in understanding the level of effort required for implementation.
* Prioritization: The Product Owner and team collaborate to prioritize user stories based on factors like business value, dependencies, and stakeholder needs.
* Breaking Down Epics: If a user story is too large or complex (an Epic), it may be broken down into smaller, more manageable pieces.
* Acceptance Criteria:

Clear acceptance criteria are defined for each user story. These criteria outline the conditions that must be met for the user story to be considered complete.

* Outcome:

The outcome of Product Grooming is a refined and prioritized product backlog. The top-priority user stories are well-understood by the team, and any ambiguities or uncertainties have been addressed.

Question 11: Explain the Roles of Scrum Master and Product Owner (3 Marks)

Product Owners and Scrum Masters are two integral roles in Scrum, an Agile project methodology While both roles are essential for successful Scrum teams, their responsibilities are distinct. When talking about Product Owner vs Scrum Master responsibilities, it is necessary to note that the Product Owner looks after the vision

and content of the product.

| Areas  | Scrum Master (PM) | Product Owner (BA) |
| --- | --- | --- |
| Roles  | Works as a leader to facilitate team growth. | Works as the customer’s representative and delivers the customer's needs. |
| Responsibilities | Ensures Scrum processes are abided by the team. | Strikes a balance between the clients and the team in terms of understanding the needs and delivering them. |
| Authority | Scrum Masters can schedule and carry out meetings while ensuring team productivity. | Product Owners make decisions regarding product backlogs. |
| Interactions | Scrum Masters generally interact with the team to work on strengthening the Scrumprocess. | The Product Owner works as a bridge between the team, stakeholders, and clients. |
| Communication | Scrum Masters offers product updates to clients. | The Product Owner analyzes the available information to project how the product meets the client's requirements. |
| TimeCommitment | Scrum Masters are committed to facilitating the Scrum process. | Product Owners are committed to prioritizing product backlog and aligning it with the client'svision. |

Question 12: Explain all Meetings Conducted in Scrum Project (8 Marks)

In a Scrum project, several meetings are conducted to facilitate effective communication, collaboration, and progress tracking. Here are the key meetings conducted in a Scrum project:

1. Sprint Planning:
* Purpose: To plan the work for the beginning of each sprint.
* Agenda:
* Review and clarify the items in the product backlog.
* Select and commit to a set of backlog items (user stories) for the sprint.
* Define the tasks required to complete the selected items.
* Estimate the effort (complexity points) for each task.
1. Daily Stand-up (Daily Scrum):
* Purpose: Identify any potential roadblock
* Agenda: Each team member answers three questions: What did I accomplish yesterday? What will I do today? Are there any impediments or roadblocks?
1. Sprint Review:
* Purpose: To review and demonstrate the completed work from the sprint
* Agenda:
* Demonstrate the completed user stories.
* Gather feedback from stakeholders.
* Review and adjust the product backlog based on feedback.
1. Sprint Retrospective:
* Purpose: To reflect on the sprint and identify areas for improvement.
* Agenda:
* Review what went well, what didn't, and what could be improved in the next sprint.
* Identify action items to implement improvements in the next sprint.

(Optional or as-needed meetings in Scrum)

* Backlog Grooming (Refinement): This involves the product owner and development team and focuses on refining and estimating backlog items
* Release Planning: This occurs at the start of the project and discusses timelines and goals
* Ad-hoc meetings: These may be scheduled as per the need for specific topics such as resolving impediments or collaboration sessions.

Question 13: Explain Sprint Size and Scrum Size (2 Marks)

"Sprint Size" and "Scrum Size" are not standard terms in Agile or Scrum methodologies. It's possible that these terms are specific to your organization or context. However, I can provide some general explanations based on how these terms might be interpreted:

1. Sprint Size:

In Scrum, a "Sprint" is a time-boxed period during which the development team works to complete a set of prioritized tasks or user stories from the product backlog. The duration of a Sprint is typically fixed (e.g., 2 weeks) and remains consistent throughout the project.

"Sprint Size" could refer to the length or duration of a Sprint. For example, a 2-week Sprint has a different "size" compared to a 4-week Sprint.

1. Scrum Size:(Scrum Team size)

"Scrum" refers to the framework used to implement Agile principles and practices. It includes roles (Product Owner, Scrum Master, Development Team), "Scrum Size" might refer to the scale or scope of a Scrum implementation within an organization. For instance, an organization might have multiple Scrum teams working on different projects, and the collective scale of these teams could be referred to as the "Scrum Size."

For large projects - 10 members; Small projects - 5 members

Question 14: Explain DOR and DOD (2 Marks)

DOR (Definition of Ready) and DOD (Definition of Done) are two important concepts in Agile and Scrum methodologies that help ensure clarity and shared understanding within a development team and is decided by product owner.

Definition of Ready (DoR): Functional

* The Definition of Ready outlines the criteria that a user story or product backlog item must meet before it is considered ready for development in a sprint. It helps ensure that the team has all the necessary information and context to effectively work on a particular item.
* Clear and Understandable: The user story or backlog item should be written in a clear and understandable manner, with well-defined acceptance criteria.
* Independent: The item should be independent and not reliant on other tasks or dependencies that are not yet completed.
* Estimable: The team should be able to estimate the complexity or effort required to complete the item.
* Testable: There should be clear acceptance criteria that can be used to verify that the item is complete and meets the specified requirements.
* Prioritized: The item should be appropriately prioritized in the product backlog, aligning with the overall goals and objectives.
* Detailed Enough: The item should contain enough information for the development team to understand what needs to be done.
* Dependencies Identified: Any dependencies or external factors that may impact the item's completion should be identified.

Definition of Done (DoD): Technical

The Definition of Done outlines the criteria that a user story or task must meet in order to be considered complete and ready for release. It serves as a checklist to ensure that the team has met all necessary criteria before marking an item as done. Key components of a well-defined Definition of Done may include:

* Code Complete: The code for the user story has been written, reviewed, and tested. Unit Tests Passed: All unit tests related to the user story have passed successfully.
* Integration Tests Passed: Any integration tests for the user story have passed successfully.
* Code Review Conducted: The code has been reviewed and approved by other team members.
* Acceptance Criteria Met: The user story meets all specified acceptance criteria.
* Documentation Updated: Any necessary documentation, such as user manuals or technical documentation, has been updated.
* Demo Ready: The user story is ready to be demonstrated to stakeholders during the sprint review.
* No Critical Defects: There are no critical defects or issues associated with the user story.
* Product Owner Approval: The Product Owner has reviewed and approved the completed user story.

Question 15: Explain Prioritization Techniques and MVP (3 Marks)

Prioritization techniques and Minimum Viable Product (MVP) are critical concepts in product development and project management. They help teams focus on what matters most and ensure that resources are allocated efficiently.

Prioritization Techniques:

* MoSCoW Method:

DefinitiOn: MoSCoW is an acronym that stands for Must-haves, Should-haves, Could-haves, and Won't-haves. It is used to categorize requirements or features based on their importance and criticality to the project.

Usage: Teams classify each requirement into one of these four categories, helping them understand what is essential for the success of the project and what can be deferred.

* 100 Dollars Test: Top 10 requirements = Numerical Assignment-Mandatory, very important, rather important, not important; does not matter.
* “FURPS” Technique: This technique is used to validate must requirement.

F-Functionality

U-Usability

R-Reliability

P-Performance

S-Supportability (Extendable, Testable & Enhance-able)

* Minimum Viable Product (MVP):

Definition: The MVP is the simplest version of a product that delivers just enough value to meet the needs of early adopters or target customers. It focuses on core features that demonstrate the product's viability and value proposition.

Question 16: Difference between Business Analyst and Product Owner (3 Marks)

| Aspect | Business Analyst (BA) | Product Owner (PO) |
| --- | --- | --- |
| Focus and Scope | Analyzes and documents business needs and processes. | Defines product features, prioritizes backlog, andensures the delivery of a valuable product. |
| StakeholderInteraction | Interacts with a wide range ofstakeholders for requirements. | Closely collaborates with stakeholders and thedevelopment team. |
| Responsibility forProject Success | Contributes to success by defining requirements. | Crucial for project success, owning the productvision and making decisions that impact the product's value. |
| Decision-MakingAuthority | Provides recommendations, and may not have final decision authority. | Has decision-making authority over the product backlog and features. |
| Involvement inAgile Frameworks | May work in Agile environments, role can vary. | Central role in Agile methodologies, especially in Scrum. |
| Involvement inDevelopmentProcess | May not be directly involved in the development process. | Actively participates in the development process,working closely with the development team. |

Question 17: Prepare a Sample Resume of 3 Years Experience Product Owner (3 Marks)

**Swati Kurwade**

**Email: k\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*@gmail.com | Phone: +91XXXXXXXXXX | LinkedIn: www.linkedin.com/in/\*\*\*\*\*\*\*\*\*\*\*\*\***

**Professional Summary:**

Experienced Product Owner with a demonstrated history of successfully leading cross-functional teams in delivering high-quality software products. With over 3 years in agile environments, I excel in backlog management, stakeholder engagement, and driving product development from concept to launch. Proven track record of defining product vision, prioritizing features, and delivering complex projects on time and within budget. Seeking an opportunity to leverage my skills in a dynamic and collaborative environment.

**Key Skills:**

- Product Management

- Backlog Management

- Agile/Scrum Methodologies

- User Story Creation

- Product Roadmapping

- Requirements Gathering

- Sprint Planning

- Stakeholder Communication

- Team Collaboration

- User Research and Testing

- Market Analysis

- Release Planning

- KPI Tracking

**Professional Experience:**

Senior Product Owner

Microsoft, Redmond, WA

May 2022 – Present

- Understands customer needs and drives the vision to engineering teams.

- Exhibits customer focus and user empathy in product development.

- Thrives in a fast-paced, technologically forward-leaning environment.

- Conducts market trend research and competitor analysis to propose actionable solutions.

- Leads agile meetings and maintains a strong connection with the scrum team.

- Fosters collaborative, open working relationships across teams.

- Proactively anticipates customer needs and resolves issues.

- Adapts to change quickly and iterates on products efficiently.

Product Owner

Microsoft, Redmond, WA

June 2021 – April 2022

- Managed a product backlog of over 100 user stories, prioritizing based on business value.

- Collaborated with cross-functional teams to deliver features on time and within budget.

- Conducted sprint planning, review, and retrospective meetings.

- Engaged with stakeholders to gather requirements and ensure alignment with business goals.

Junior Manager Product Owner

Infosys, Pune, MH

November 2020 – May 2021

- Supported marketing teams by planning, executing, and improving selected activities.

- Contributed to marketing events, price and product analyses, and communications.

- Managed reports, trackers, and dashboards; implemented process improvements.

- Coordinated responses to operational inquiries from stakeholders, channel partners, or vendors.

- Conducted primary and secondary research to gather and analyze information.

- Outlined presentations and communications, collaborating with team members.

Junior Product Owner

Infosys, Pune, MH

July 2019 – October 2020

- Assisted in backlog refinement and user story creation.

- Coordinated with development teams to ensure alignment on sprint goals.

- Gathered and analyzed customer feedback to inform product decisions.

- Supported the Scrum Master in daily stand-ups and sprint reviews.

**Projects:**

Project: New Feature Development

Role: Product Owner

- Led the development and launch of a new feature that increased user engagement by 20%.

- Facilitated workshops to gather requirements and prioritize features based on customer feedback.

Project: Payment Gateway Implementation

Role: Junior Product Owner

- Collaborated with the development team to implement a new payment gateway, reducing transaction failures by 15%.

- Conducted user acceptance testing to ensure the new feature met quality standards.

**Education:**

Master of Information Systems Management

Carnegie Mellon University, Pittsburgh, PA

2021

**Certifications:**

- Certified Scrum Product Owner (CSPO)

- Agile Certified Practitioner (PMI-ACP)

**Computer Languages:**

- Javascript

- Python

- SQL

- C#

- R

**Software Tools:**

* PM

- Azure DevOps

- JIRA

- Trello

- Asana

* Data analysis and vizualization

- Power BI

- Tableau

* Design & prototyping

- Figma

- Sketch

- Adobe XD

* Collaboration & Communication

- Microsoft Teams

- Slack

- Confluence

* Customer feedback & analytics

- Google Analytics

- Hotjar

**Interests:**

- Hiking

- Swimming

- Volunteering for community good

- Traveling and exploring

**PREP 3**

Q1. Draw a Use Case Diagram

A Use Case Diagram for a customer making a payment by Card, Wallet, Cash, or Net Banking should include the following actors and use cases:



Q2. Derive Boundary Classes, Controller Classes, Entity Classes

Boundary Classes- Used to handle interactions between the systems and external actors: **a boundary represents a user interface screen or some kind of input-output device:** use cases become boundary class:

* PaymentUI
* CardPaymentUI
* WalletPaymentUI
* CashPaymentUI
* NetBankingPaymentUI

Controller Classes- Acts as an intermediate between boundary and entity classes: **a controller is defined as what controls the work that's done as well as when and how that work is done so you might think of a controller as an overseer that decides for example how the system should handle a user's request:** use case relationship where there is no third party involved becomes controller class

* PaymentController
* CardPaymentController
* WalletPaymentController
* CashPaymentController
* NetBankingPaymentController

Entity Classes- Represents the core data and business logic of the application: **an entity is some persistent element that is something that gets stored typically in a file so an entity in a sequence diagram is typically implemented as a table or an element in a database**: all the actors become entity class

* Customer
* Payment
* CardDetails
* WalletDetails
* BankDetails
* Transaction

Q3. Place These Classes on a Three-Tier Architecture

Presentation Tier:

* PaymentUI
* CardPaymentUI
* WalletPaymentUI
* CashPaymentUI
* NetBankingPaymentUI

Business Logic Tier:

* PaymentController
* CardPaymentController
* WalletPaymentController
* CashPaymentController
* NetBankingPaymentController

Data Access Tier:

* Customer
* Payment
* CardDetails
* WalletDetails
* BankDetails
* Transaction

Q4. Explain Domain Model for Customer Making Payment through Net Banking

A domain model is a conceptual representation that defines the structure, relationships, and behaviors of entities within a specific problem domain. Following is the domain model for a customer making a payment through Net Banking including entities and their relationships. **Visual representation of the relation between different component tables and entities**

Q5. Draw a Sequence Diagram for Payment Done by Customer Net Banking

A sequence diagram is a type of interaction diagram used in software engineering and systems design to illustrate how processes operate with one another and in what order, following is a diagram representing the same:



Q6. Explain the Conceptual Model for this Case

A conceptual model for the payment process done by a customer using net banking 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:

1. Entities: Customer, Product, Order & Payment
2. Attributes: CustomerID, Name, Email, Phone Number
3. Relationship: For example, a customer places an order.

Q7. What is MVC Architecture? Explain MVC Rules to Derive Classes from Use Case Diagram and Guidelines to Place Classes in 3-Tier Architecture

Model–view–controller (MVC) is a software design pattern commonly used for developing user interfaces that divide the related program logic into three interconnected elements. These elements are the internal representations of information (the model), the interface (the view) that presents information to and accepts it from the user, and the controller software linking the two.

It is an architecture or a software design pattern that makes creating huge applications easy. It does not belong to a specific programming language or framework, however, it is a concept that you can use in creating any kind of application or software in any programming language.

* Model: The central component of the pattern. It is the application's dynamic data structure, independent of the user interface. It directly manages the data, logic, and rules of the application. The model works directly with the database. It does not have to deal with user interface or data processing. In day-to-day scenarios, we will simply use the model to fetch, insert, update, and delete data from your database. For example, imagine we are creating a task management application that will simply allow users to organize tasks based on date and time. It means that we will have users and tasks to manage in our database. In the language of MVC, User and Task are models in our application.
* View: View is the User Interface on which our customer/user can perform some actions. It contains HTML, CSS, JS, XML, or any other markup language that we can use to create a beautiful user interface. It also contains code to show the data that it receives from our application. View means any representation of information such as a chart, diagram, or table. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants. The only two things that a View has to do are to show data to the customer/user on the User Interface and to respond to the events.

For example, what to do when a user clicks on the Update or Delete button? The answer is, that the user should be redirected to the Update form or the delete confirmation popup.

* Controller: The controller is the part in which we process the data after we get a request from View and before updating anything in our database with our Model. Accepts input and converts it to commands for the model or view.
* 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.

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

Q8. Explain BA Contributions in Project (Waterfall Model – All Stages) – 8 Marks

| Stages | Activities | Artifacts & Resources |
| --- | --- | --- |
| Pre Project  | Enterprise Analysis - SWOT Analysis, GAP Analysis, 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, Presale consultants |
| Planning, Estimation & AssessmentProject Kick-Off(Big Picture Plan) | 1. Understand Assumptions and constraints along with business rules and business goals
2. Plan packages for Big Projects
3. Understand the project plan from PM
4. BA conducts stakeholder analysis
5. Plan BA approach strategy (Requirement gathering techniques, communication, requirement management, documents to follow, tools to use, change request handling methodology) for this project
 |  |
| PM, Sr. BA |
| Requirement Gathering | 1. Stakeholder identification and document
2. Client gives BRD, or BA prepares BRD by interacting with the Client- Brainstorming, Document Analysis, Reverse Engineering, Interviews, workshops, Focus Groups, Observation, and Questionnaires.
3. Prototyping can be used by BA to make the Client give more specific requirements
4. Sort the gathered requirements (avoiding duplicate requirements, grouping into similar functionality or into modules)
5. Prioritize requirements- MoSCoW
6. Validate Requirements- FURPS
 | BRD (Business Requirement Document) |
| BA, PM |
| Requirement Analysis | 1. Draw UML Diagrams (Use Case and Activity Diagrams)
2. Prepare Functional requirements from Business requirements
3. All Architects come up with Technical Requirements (SSD)
4. SRS will have Functional Requirements and Technical Requirements
5. Take Sign-Off on SRS from the Client. SRS is the first legally binding document between the Business and the Technical team.
6. BA prepares RTM from SRS before the Design phase starts. (BA is the owner of RTM)
7. BA traces how the requirements are dealt in each phase of the development life cycle from Design to UAT.
 | Functional Requirement Specification,SSD (Supplementary Support Document),SRS (Software Requirement Specification),RTM (Requirement Traceability Matrix) |
| BA, PM, Solution Architect, DB Architect, NW Architect |
| Design | 1. From the Use Case Diagram, the Test Manager or BA will prepare Test Cases.
2. Communicate with the Client on the Design and Solution Documents (Update 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 the end-user manuals
4. Update RTM
5. From Use Case Diagrams, the Solution Architect recommends the Architecture of the IT Solution.
6. DB Architect uses Persistence Classes (Entry Classes) and comes up with the ER Diagrams or DB Schema.
7. GUI Designer will look into 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 organizes JAD Sessions
2. BA clarifies queries of the Technical Team during Coding
3. Developers refer to Diagrams and Transient(Controller Classes) of BA and code their unit.
4. Update end-user manuals
5. Update RTM
6. Conduct regular Status meetings with the Technical team and the Client and turn the Client for participation in UAT
 | LDD-CDD Application |
| Development Team, BA, PM |
| Testing | 1. BA prepares Test Cases from the Use Cases or assists the Test Manager in doing so
2. BA Performs high-level testing
3. BA prepares Client for UAT
4. Test Data is requested by BA from the Client
5. Update end-user manual
6. Update RTM
7. Take Sign-Offs from Client on Client Project Acceptance form
 | Test Concerning Documents Application with less errors |
| Testing Team, BA, PM, Client |
| Deployment & Implementation | 1. Forward RTM to the Client or the PM which should be attached to the Project Closure Document
2. Coordinate to complete and share end-user manuals
3. Plan and organize Training Sessions for end users
4. Prepare lessons learned from this project (to take precautions for the coming project)
 |  |

Q9. What is Conflict Management? Explain Using Thomas-Kilmann Technique

Conflict management is the approach and strategies designed towards achieving a positive outcome and resolution amongst the parties involved in matters relating to conflicts. These strategies and approaches are dependent on the type of conflict that exist in the organizations or institutions involved. It could be a systematic or unordered method that is task-specific, research-oriented, and requires proper attention. Conflict management is the process of limiting the negative aspects of conflict while increasing the positive aspects of conflict. Conflict management aims to enhance learning and group outcomes, including effectiveness or performance in an organizational setting.

It's human to deal with conflict by defaulting to what's comfortable. According to Thomas and Kilmann, most people take one of two approaches to conflict management, assertiveness or cooperativeness. From these approaches come five modes or styles of conflict management

* Accommodating:

An accommodating mode of conflict management tends to be high in cooperation but low in assertiveness. When you use this style, you resolve the disagreement by sacrificing your own needs and desires for those of the other party. This kind of conflict management requires that the parties involved accept the views of one another. Both or one is ready to give the other party/side what is needed, for instance, in a business environment where employees are required to dress formally throughout the working days of the week. An employee can approach the organization to allow the wearing of casuals on Fridays. This can cause conflicts between the company and the employee because that may be a core organizational principle. However the company can apply an accommodation strategy by allowing the employees to wear casuals on Fridays. People are happy and feel among when this strategy is employed

* Avoiding:

When avoiding, you try to bypass a conflict. This style of managing conflicts is low in assertiveness and cooperativeness. Avoidance is unproductive for handling most disputes because it may leave the other party feeling like you don't care. Also, if left unresolved, some conflicts become much more troublesome.

* Compromising:

Compromising demands moderate assertiveness and cooperation from all parties involved. With this type of resolution, everyone gets something they want or need. This style of managing conflict works well when time is limited. Because of time constraints, compromising isn't always as creative as collaborating, and some parties may come away less satisfied than others. Most businesses would include this strategy to resolve a dispute when parties observe that there is a risk at the end, as any of them could lose something of value in matters that caused the conflict. Contract negotiation, customer service, and big investors are aspects you will see businesses compromising their standard to work with the other party.

* Collaborating:

A collaborating conflict management style demands a high-level of cooperation from all parties involved. Individuals in a dispute come together to find a respectful resolution that benefits everyone. Collaborating works best if you have plenty of time and are on the same power level as the other parties involved. If not, you may be better off choosing another style.

* Competing:

This type of conflict management strategy often involves individuals who are assertive and have the will power to fight till they achieve their goal and the other losing. For instance, a company might hold on to this competitive strategy by conflicting and holding on until they win on issues that will damage the organization's reputation based on an employee trying to defraud the company.

Q10. List Down the Reasons for Project Failure

There are no guarantees that starting a project will ensure that it’s delivered successfully. There are many ways a project can fail. To avoid failure, it’s important to understand failure and know the signs that your project is in danger. Project failure can be delivering a project after its deadline, over its budget, or not meeting its quality expectations. It could also be not delivering the project for any number of reasons. However, if the project fails, the client or stakeholder won’t get a return on their investment.

Reasons for Project Failures:

* Poor Planning:

Insufficient planning can lead to unrealistic timelines, inadequate resource allocation, and overlooked dependencies. For example, launching a software product without a detailed development and testing schedule can cause missed deadlines and budget overruns, resulting in project failure.

* Unclear Objectives and Requirements:

When project goals and requirements are vague or poorly communicated, team members may work towards different objectives, causing misalignment and confusion. For instance, developing a new app without clear specifications can result in features that don't meet user needs, leading to project failure.

* Inadequate Risk Management:

Failing to identify and mitigate potential risks means the project can be derailed by unforeseen issues. For example, not planning for potential supplier delays in a construction project can cause significant delays and increased costs, ultimately halting the project.

* Poor Communication:

Effective communication is essential for coordination and clarity. Poor communication leads to misunderstandings, errors, and a lack of cohesion among team members. For instance, if a marketing team isn't kept in the loop about product changes, their campaigns may become irrelevant, impacting project success.

* Scope Creep:

Uncontrolled additions of new features or requirements without proper change management can stretch resources thin and disrupt the project timeline. For example, continually adding new features to a software development project without adjusting the schedule or budget can lead to delays and increased costs, overwhelming the team.

* Lack of Stakeholder Engagement:

Engaging stakeholders ensures that their needs and expectations are understood and met. Without their involvement, the project may miss critical requirements and face resistance. Also, when stakeholders don't put sufficient time and effort into giving inputs or clarifying doubts, it can hinder project progress. For example, not involving key users in the development of a new software system can result in a product that doesn't meet their needs, causing project failure.

* Resource Constraints:

Adequate resources are crucial for project success. Constraints in time, budget, or personnel can impede progress and reduce the quality of work. For instance, attempting to develop a complex mobile application with a small, inexperienced team can lead to missed milestones and project failure.

* Technical Challenges:

Projects involving new or complex technologies may encounter significant technical issues. For example, implementing an advanced AI system without sufficient expertise can lead to persistent technical difficulties, increasing costs and delaying progress, ultimately resulting in project failure.

Q11. List the Challenges Faced in Projects for BA

Following are the challenges faced by a BA during a project:

* Unclear or Changing Requirements:

BAs struggle with vague or frequently changing requirements, causing confusion and misalignment among team members. This leads to increased workload, rework, and delays, jeopardizing project success. For example, a client requests a simple reporting tool but later adds advanced analytics, causing the team to repeatedly revise the project scope and timeline, leading to rework and delays.

* Managing Stakeholder Expectations:

Balancing diverse stakeholder expectations is challenging, as different priorities can cause conflicts. Effective communication, negotiation, and stakeholder engagement are crucial. Ensuring a shared understanding of goals and limitations is vital. For example, the marketing team wants user-friendly features, while finance prioritizes cost savings, leading to misaligned project goals and difficulties in meeting everyone’s needs.

* Scope Creep and Scope Management:

Uncontrolled scope changes (scope creep) significantly impact project timelines and budgets. BAs must implement strict change management to control scope and evaluate new requirements' impact. For example, during a website redesign, stakeholders continuously add features like additional payment gateways without extending deadlines or budgets, overwhelming the team and delaying the project.

* Time and Resource Management:

Effective time and resource management keeps projects on track and within budget. Challenges arise with limited resources, unexpected issues, or conflicting priorities. BAs must monitor progress and adjust plans. For example, a software development project faces delays due to an insufficient number of developers or unexpected technical issues like bugs that require more time to resolve.

* Quality Assurance and Testing:

Ensuring final deliverables meet quality standards and user requirements involves thorough testing and validation. Challenges include coordinating testing activities and managing defects. BAs must work closely with QA teams. For example, failing to properly test the checkout process in an e-commerce platform can result in a non-functional payment system, leading to lost sales and customer dissatisfaction.

* Documentation and Knowledge Management:

Creating and maintaining comprehensive documentation for all project aspects is essential for clarity and continuity. Challenges include ensuring accuracy and managing versions. Poor documentation leads to misunderstandings and errors. For example, a BA needs to document all functional requirements, user stories, and design specifications for a new CRM system, ensuring they are up-to-date and accessible.

* Technology Constraints:

BAs face challenges with technology limitations and integration issues, affecting feasibility and development efforts. Managing these constraints involves planning, risk assessment, and communication. For example, implementing a new ERP system is constrained by existing legacy systems that are not easily compatible, requiring additional resources and time to develop custom integration solutions or migrate data.

Q12. Write About Document Naming Standards

Establishing electronic file-naming standards will:

Improve access to documents and make sure that the right documents are available at the right time to support the daily business operations of each unit; facilitate more effective collaborative work between employees; and ensure that electronic files are easy to locate for records retention purposes. All documents will be named using some standards like:

[ProjectID ][Document Type]V[X]D[Y].ext

Q13. What Are the Do’s and Don’ts of a Business Analyst

Being a successful business analyst requires a combination of technical skills, communication skills, and a deep understanding of the business.

* Do's of BA:
1. Understand the business:

To be an effective business analyst, you need to have a deep understanding of the business you're working in. This means understanding its goals, objectives, and operations. By understanding the business, you'll be able to identify the right data to analyse and provide insights that are relevant to the organization.

1. Listen carefully:

As a business analyst, you'll be working with different stakeholders in the organization, including business leaders, developers, and project managers. It's important to listen carefully to their needs, concerns, and feedback. This will help you understand their requirements and provide solutions that meet their needs.

1. Be data-driven:

As a business analyst, you'll be working with a lot of data. It's important to be data-driven in your approach, which means using data to support your recommendations and decisions. This will help you provide insights that are based on facts and evidence rather than opinions.

1. Communicate clearly:

To be an effective business analyst, you need to be able to communicate clearly and concisely. This means presenting data in a way that is easy to understand and using clear and jargon-free language. This will help you to ensure that your recommendations are understood and acted upon.

1. Be adaptable:

The business environment is constantly changing, and as a business analyst, you need to be adaptable. This means being able to adjust your approach and methods as needed to meet changing business needs.

* **Don'ts of BA:**
1. Never say NO to the client.
2. There is NO word called “By Default”
3. Never imagine anything in terms of GUI.
4. Question the existence/ question everything in the world. Eg: what the client gives is not always correct
5. Consult an SME for clarifications in Requirements. Every problem of the Client is unique. No two problems of different clients are the same. Maybe the approach, technology, place of use, and local laws may vary making them different.
6. Make assumptions: As a business analyst, it's important to avoid making assumptions. Assumptions can lead to incorrect conclusions and recommendations that don't meet the needs of the organization. Instead, focus on gathering and analyzing data to support your recommendations.
7. Ignore feedback: Feedback is important in any business environment, and as a business analyst, you need to be open to feedback from stakeholders. Ignoring feedback can lead to missed opportunities and solutions that don't meet the needs of the organization.
8. Use jargon: Using technical jargon can be confusing for stakeholders who may not be familiar with the terminology. Avoid using jargon and instead, focus on communicating in a language that is easy to understand.
9. Overcomplicate things: As a business analyst, it's important to keep things simple. Overcomplicating things can make it difficult for stakeholders to understand your recommendations and may lead to too complex or difficult-to-implement solutions.
10. Work in isolation: Business analysis is a collaborative effort, and as a business analyst, you need to work with different stakeholders in the organization. Avoid working in isolation and instead collaborate with others to ensure that your recommendations meet the needs of the organization.

Q14. Write the Difference Between Packages and Sub-systems

|  | Packages | Subsystems |
| --- | --- | --- |
| Definition | Collection of components that are not reusable in nature | Collection of components that are reusable in nature |
| Granularity | Smaller and more focused in scope | Larger and encompasses multiple packages or modules |
| Dependency Management | Manage dependencies at a class/component level  | Manage dependencies at a higher level, defining boundaries and interfaces between different parts of the system. |
| Example: | Application development companies work on packages. | Product development companies work on sub-systems. |

Q15. What is Camel-Casing and Explain Where It Will Be Used

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

Example: camelCasingExample

Q16. Illustrate Development Server and What Are the Accesses Does Business Analyst Have?

Development Server:

A development server refers to a dedicated environment or server that is used during the software development process. It provides a platform for developers and testers to build, test, and debug applications before they are deployed to a production environment. The development server typically replicates the target production environment to ensure compatibility and accurate testing.

BA Accesses:

- Read-only access to requirements and design documents.

- Limited access to test environments.

- Access to project management tools and collaboration platforms.

Q17. What is Data Mapping?

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 from another source. Data mapping is commonly used in data interpretation, 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 and determining their corresponding target data elements. This is especially important when you are moving data between different systems or databases to ensure that the data stays consistent and accurate.

Q18. What is API? Explain How You Would Use API Integration in the Case of Your Application

API (Application Programming Interface):

An API is a set of rules and protocols that allow different software applications to communicate with each other. It defines the methods and data formats that applications can use to request and exchange information.

Using API Integration:

1. Establish API Communication: Set up API communication between your application and the other application to exchange data.
2. 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.
3. Data Parsing: When receiving the date 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.
4. 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.