In Sprint 5 of the analytics dashboard project, the team delivers multiple visualizations. During the sprint review, the sales leadership team expresses dissatisfaction, stating that key performance indicators (KPIs) were not calculated as expected. The stories were marked "done" by the team but failed to meet user expectations.

Q1. How should the team define and verify stakeholder expectations more effectively?

The team should ensure that acceptance criteria are written in stakeholder-friendly terms (e.g., "include rolling 30-day average with comparison to last quarter") and validated using Examples or Given-When-Then formats. Use collaborative specification sessions during refinement to involve business users directly. Stakeholder demos and mid-sprint reviews can also help realign expectations before the sprint ends.

Q.2 What mistake might have occurred in defining the Definition of Done (DoD)? The DoD may have focused on technical completeness (e.g., "query runs without error," "chart renders correctly") without incorporating business validation (e.g., KPIs reflect agreed logic). The team should enhance the DoD to include user validation, data correctness, and visual accuracy for analytics projects.

Q.3 What role can a Business Analyst or Data Steward play here?

In analytics projects, Business Analysts or Data Stewards bridge the gap between raw data and business interpretation. They can:

- Validate logic before development
- Provide business rules documentation
- Help write data story acceptance criteria
- Act as proxy Product Owners when the PO is unavailable

Q4. How can the team prevent such expectation mismatches in future sprints?

Implement the following:

- Story walkthroughs with stakeholders during backlog refinement
- Create mockups or low-fidelity wireframes to visualize expected charts early
- Use data dictionaries and sample outputs
- Involve stakeholders in mid-sprint checkpoints
- Enhance sprint reviews with business-focused demo scripts

During Sprint 6, the team is tasked with creating a revenue breakdown dashboard, but they discover that key data from a third-party provider is delayed. The external API endpoint needed for historical revenue data is not yet provisioned, and the sprint goal becomes unachievable.

Q.1 How should the team handle blocked stories due to external dependencies? The team should raise an impediment immediately via the Scrum Master, who can escalate to the external partner or Product Owner.

Stories that rely on unavailable components should be:

- Marked as blocked and moved to the next sprint if delay persists
- Split to allow partial progress on components that can be completed
- Paired with a spike or stub implementation to simulate functionality for continued development/testing

Q.2 How can external dependencies be better managed in backlog planning?

Backlog refinement should identify external data risks early, tagging stories with:

- Dependency flags
- Lead times for API availability
- Fallback plans (e.g., use mock data)

Dependencies should be tracked in a risk register or via dependency mapping tools.

Q.3 Should the sprint goal be changed mid-sprint due to this issue?

Yes — but only if the entire sprint goal becomes invalid. In such cases:

- The Scrum team and PO mutually agree to revise the goal.
- Remaining capacity is reallocated to other high-priority stories or technical debt.
- Stakeholders should be informed transparently about the cause and new sprint direction.

Q.4 What Agile techniques help manage uncertainty in external data delivery?

- Use spikes to explore API integration feasibility before committing a story to a sprint.
- Establish early contracts or SLAs with data providers.
- Use agile modelling to decouple backend API integration from front-end chart rendering.
- Implement progressive delivery, allowing parts of the dashboard to ship while waiting for external inputs.

Your AI chatbot MVP is deployed with limited scope. However, customer feedback reveals that the chatbot frequently misunderstands refund requests due to poor Natural Language Processing (NLP) detection. This impacts on customer trust, and the business team is considering reverting to human-only support.

Q.1 How can the team respond to user feedback and preserve project confidence? Initiate an emergency triage sprint to:

- Analyse misunderstood queries
- Improve training data coverage for key intents
- Raise the confidence threshold to reduce false positives
- Add fallback rules (e.g., "I'm transferring you to a human")
- Communicate transparently with stakeholders, showing metrics and remediation timelines.

Q2. What acceptance criteria should have been used to validate this intent?

Acceptance criteria should include:

- Minimum intent classification precision/recall (e.g., >85%)
- Successful handling of 5+ test phrases per intent (e.g., "I want a refund," "return my item")
- Fallback logic triggers if confidence is below threshold
- Success confirmed via end-user interaction logs or test sessions

Q3. How does model explainability help in AI project retrospectives?

Model explainability tools (e.g., LIME, SHAP) reveal why a model predicted a specific intent. This helps the team:

- Identify poor or biased training data
- Improve intent mapping logic
- Communicate technical issues clearly to stakeholders
- Make future retrospectives more data-informed, reducing trial-and-error fixes

Q4. What Agile practice supports continuous model improvement after deployment?

Agile embraces Continuous Delivery (CD) and Iterative Learning Loops:

- Use sprint cycles to iterate on training data and retrain models
- Monitor post-release analytics to track error types
- Treat AI as a living product rather than a fixed deliverable
- Integrate feedback collection mechanisms (like thumbs-up/thumbs-down) into the chatbot interface

The chatbot is technically functional and meets performance benchmarks, but stakeholders argue over its tone: Marketing wants it friendly and casual, while Legal demands a neutral tone to avoid compliance risks. Sprint reviews become tense and fragmented.

Q1. How can conflicting tone requirements be handled during chatbot development?

Tone should be treated as a non-functional requirement, captured in:

- A shared Chatbot Personality Guide or style document
- Examples of do's and don'ts for responses
- Sprint stories for tone adjustments (e.g., "Rephrase refund flow to meet tone guideline v2")

If needed, tone can be contextual — friendly for general queries, formal for policy-related ones.

Q2. What role does Definition of Done (DoD) play in resolving tone debates?

The DoD should include:

- Tone compliance checks during peer review
- Alignment with approved tone style guide
- Stakeholder sign-off on representative flows (e.g., greetings, escalations, sensitive topics)

Q3. How can you involve stakeholders constructively during sprint reviews?

Create structured demo sessions with:

- Scripted conversations across various use cases
- A tone rating matrix for stakeholder feedback
- Live voting on sample replies (e.g., "Which of these responses feels appropriate?") This encourages collaborative feedback rather than conflict.

Q4. What Agile tool or ceremony can reduce future tone-related conflicts?

Use Design Studios or Storyboarding Workshops early in each PI or epic to align on tone. Involve Marketing, Legal, and UX in:

- Co-creating sample dialogues
- Mapping tone to user emotions (e.g., anger, confusion)
- Building reusable **conversation components**

Also, use personas and empathy maps to align chatbot tone with real user expectations.

A financial institution is modernizing its core banking software by migrating it from on-premises legacy systems to a secure, cloud-based infrastructure. The migration includes several critical components: database migration, identity and access management (IAM) integration, performance testing, and staff training. The institution must maintain business continuity, ensure regulatory compliance, and minimize downtime — making accurate dependency management crucial.

Q1. What are the mandatory dependencies in this migration?

Mandatory dependencies, also called hard logic dependencies, are those that must occur in a specific order due to the nature of the tasks.

In this case:

- Database migration must be completed before performance testing can begin. Performance testing requires a working data environment with real or representative data to evaluate system behaviour, load capacity, and query performance.
- Similarly, the cloud infrastructure must be provisioned and configured before any legacy data upload or database transfer can occur. This includes setting up:
  - Virtual networks
  - Firewalls and IAM policies
  - Storage buckets or databases (e.g., AWS RDS, Azure SQL)
  - Encryption and backup mechanisms

These are non-negotiable sequences, dictated by technical constraints. Ignoring them would result in failed tests, corrupted data, or non-compliance with financial data handling policies.

Q2. What would be an example of a discretionary dependency related to training?

A discretionary dependency (also known as a soft dependency) is established based on best practices, team preferences, or scheduling decisions, not technical necessity.

In this project:

- User training for staff (e.g., customer service agents, back-office staff) could be scheduled after deployment, so it includes training on the final user interface and configuration. Alternatively, training could begin earlier using mock systems, wireframes, or partial functionality to accelerate onboarding.
- The decision to delay training until after UI stabilization is **not mandatory** — it's a discretionary choice aimed at enhancing training relevance and reducing rework.

Such flexibility enables parallel planning, though it comes with a trade-off between early readiness and training accuracy.

Q3. Identify one external dependency that may impact the cloud migration.

External dependencies involve tasks or approvals that originate outside the project team's control yet significantly impact its progress.

A critical example in this scenario:

- Regulatory audits and cloud compliance approval from financial oversight bodies or government regulators (e.g., APRA in Australia, or the SEC in the U.S.). The institution may need:
  a) Pre-deployment audits to confirm data residency, encryption standards, retention policies, and access controls.
  b) Certifications or compliance letters before systems go live (e.g., ISO 27001, PCI-DSS).

Delays in receiving regulatory feedback, or changes in compliance expectations, can cause substantial delays in the deployment phase, requiring the project to build **buffer time** and maintain an **escalation path**.

Q4. What internal dependencies should be noted between UI configuration and backend API readiness?

Internal dependencies are those that occur within the project team or system architecture — where one component must be in place before another can function correctly.

In this case:

- The **frontend UI (e.g., account summary screen, loan dashboard, transaction interface)** must connect to **backend APIs** to retrieve customer data, initiate transactions, or display account activity.
- Therefore, **UI development is dependent on backend API readiness** — specifically, on:
  - API endpoint availability
  - Stable data structures and responses
  - Secure authentication flows

If the UI is built first using mock data but APIs change later, it may lead to:

  - Data binding errors
  - User experience bugs
  - Costly rework

To minimize risk, teams should synchronize **UI and backend teams via interface contracts**, maintain **stub APIs**, and prioritize **integration testing** before full deployment.

A government IT team is tasked with building a nationwide COVID-19 vaccination tracker app to support public registration, real-time vaccine availability, hospital integration, and analytics dashboards. Ensuring public trust, data accuracy, and high availability requires precise dependency sequencing, given the system's scope and political visibility.

Q1. What are mandatory dependencies in developing this app?

Mandatory (hard) dependencies reflect the logical order in which tasks must be completed due to technical or functional requirements.

Key examples:

- Hospital database integration must be completed before displaying vaccine availability. Without a live data feed from hospitals or clinics, the app cannot show accurate appointment slots, doses available, or location-specific updates.
- User registration functionality must precede the appointment booking module. Users must first be authenticated, verified, and profiled before they can view or reserve time slots.

These dependencies are fundamental to core app functionality and must be prioritized in sequencing, testing, and release planning.

Q2. What is an example of a discretionary dependency in the development process?

Discretionary dependencies are based on planning preferences or team strategy rather than technical limitations.

In this case:

- The development team chooses to build dashboard analytics after the core registration and booking modules, even though technically these could be developed in parallel.

The reasons for this decision may include:

- Focusing on mission-critical functionality first to meet public health deadlines.
- Waiting for real user data to make analytics dashboards meaningful.
- Reducing team context-switching and complexity in early phases.

This dependency is not required by the system architecture—it's a strategic call to improve quality and delivery focus.

Q3. What external dependency could influence the release timeline?

An external dependency refers to any task or deliverable outside the project team's direct control, but which affects its timeline or success.

In this project:

- Timely access to vaccination data from regional health departments is a critical external dependency.

Challenges may include:

- Inconsistent data formats across states
- Delays in API development or batch data feeds
- Legal or bureaucratic hurdles in data-sharing agreements

Without this data, the app's core value proposition (e.g., showing vaccine availability by location) becomes unreliable. To mitigate this, the team should:

- Establish SLAs with data providers
- Create mock feeds or simulators to proceed with development
- Prioritize data integration tests before public launch

Q.4 What internal dependencies should be mapped between user verification and SMS gateway integration?

This scenario highlights an internal dependency — where one feature must precede another within the system's logic.

Specifically:

- Phone number verification must occur before the app sends SMS-based appointment confirmations or reminders.
- The SMS gateway must be integrated only after the verification logic is validated, ensuring:
  - The number belongs to the intended user
  - Consent has been recorded for message delivery
  - Messages won't be sent to the wrong recipient

Failing to observe this dependency could result in **data privacy violations**, miscommunication, or noncompliance with data protection regulations (e.g., GDPR, HIPAA).

Therefore, development must ensure:

- Sequenced implementation
- Testing of fallback flows (e.g., resend codes, update number)
- **SMS throttling mechanisms** to prevent abuse

A financial services firm is moving its legacy CRM, financial software, and document storage systems to the AWS cloud. The project includes phases like system assessment, infrastructure reconfiguration, data migration, testing, and compliance audits. Given tight budgets and regulatory oversight, the firm must produce highly accurate cost estimates using multiple estimation approaches.

Q1. How can parametric modeling be effectively used here?

Parametric modeling uses quantitative historical data and scalable variables to estimate costs. It is most effective when the cost factors have a predictable relationship with project parameters.

In this scenario, the team can use:

- **Cost per user**: If past projects showed a cost of $120 per user to migrate CRM functions, and the firm has 1,200 users, the CRM migration alone could cost $144,000.
- **Cost per server instance**: If historical migration projects cost $3,000 per server (including licensing, labor, and cloud provisioning), and this project involves 40 servers, the estimate would be $120,000.

Additional factors may include:

- Cost per GB for data transfer or storage
- Cost per compliance control (e.g., encryption, MFA enforcement)
- Estimated hours per audit requirement

Parametric modelling helps the firm:

- Establish quick budget forecasts
- Run what-if analyses for scaling
- Ensure cost justification in compliance documentation

What is the value of bottom-up estimation for this migration?

Bottom-up estimation involves breaking the project into detailed work packages, estimating each separately, and aggregating them into a total cost.

In this cloud migration:

Each major component is decomposed into tasks, such as:

  - **CRM Migration**: Data extraction, schema mapping, integration
  - **Finance System**: Security hardening, data compliance testing
  - **Document Storage**: Encryption setup, metadata tagging, access policies
  - **Testing**: Functional, performance, and regression testing
  - **Compliance Audits**: Internal reviews, documentation, submission costs

For each task, estimators calculate:

  - Required staff hours and skill level
  - Software or service licenses (e.g., AWS KMS, CloudTrail)
  - Tooling and testing environment costs

This approach yields:

  - High accuracy, especially for projects with many variable components
  - Greater stakeholder confidence
  - A roadmap for tracking actual vs. estimated costs during execution

How would three-point estimation support risk management in cost planning?

Three-point estimation uses three scenarios — Optimistic (O), Most Likely (M), and Pessimistic (P) — to calculate an expected cost while accounting for uncertainty and risk.

The formula:

Expected Cost (E) = (O + 4M + P) / 6

Example: For migrating legacy finance software:

  - O: $20,000 (if code is easily portable)
  - M: $28,000 (if minor rework is needed)
  - P: $45,000 (if major compatibility issues arise) Using the formula:

E = (20,000 + 4×28,000 + 45,000) / 6 = $29,500

This technique helps the firm:

  - Understand financial exposure to delays or unknowns
  - Prioritize contingency reserves for high-risk components
  - Justify risk buffers to regulators and internal audit

Q4: When might analogous estimating mislead cost planners?

Analogous estimation relies on past project data to provide quick estimates for similar projects. It becomes misleading when critical variables differ significantly.

For instance:

The last cloud migration project involved **a retail company** with minimal data privacy constraints. If the financial institution uses that project's $250,000 cost as a baseline, it might **ignore financial sector compliance costs**, such as:

- PCI-DSS and APRA compliance
- Data residency regulations
- Encryption at rest and in transit
- Incident response frameworks

This underestimation could result in:

- Budget overruns
- Audit failures
- Delays in go-live due to non-compliance

Analogous estimates are useful only when project environments, complexity, and compliance obligations are truly comparable.

A city council is deploying a **smart parking system** to improve traffic flow, payment efficiency, and real-time monitoring. The project includes installing **IoT parking sensors**, a **mobile application for users**, **central dashboards** for operators, and **payment integrations**. The budget is funded by public grants, requiring **accurate cost forecasting** and transparent financial reporting.

Q1: How can parametric modeling help the planning team?

In this project, parametric modeling is particularly useful because it's easy to scale known costs. Example:

- From a pilot project, it was found that installing and configuring a single parking spot costs **$420**, including sensor, networking, testing, and labor.
- With 1,000 spots planned citywide:
  - 1,000 × $420 = **$420,000 total baseline**

Other parametric variables may include:

- Cost per app user (based on projected adoption)
- Cost per dashboard view or traffic analytics endpoint
- Cost per payment integration for public-private payment providers

This helps:

- Provide fast forecasts
- Offer a range of budgets for different scope levels
- Guide grant applications with justifiable figures

Q2: What's a bottom-up approach for estimating total system cost?

Bottom-up estimation allows planners to account for all line items and indirect **costs** by breaking the project into detailed components:

- **Sensor Installation**: Unit cost, labour, roadwork adjustments, and wiring
- **Mobile App Development:** User flows, front-end dev, backend services, security
- **Central Dashboard:** GIS integration, live data visualization, admin features
- Payment Integration: API dev, testing, and legal compliance (e.g., PCI-DSS)
- **Testing and Calibration:** Field testing, acceptance checks, sensor calibration
- **Training and Documentation:** Staff workshops, SOPs, multilingual user manuals Summing individual estimates allows the council to:
  - Build a granular cost model
  - Isolate overruns to specific areas
  - Present detailed breakdowns for audit or funding review

Q3: When is three-point estimation most useful in this project?

Three-point estimation is highly effective in areas with physical construction variability, such as:

- Parking zones with narrow roads, underground infrastructure, or private ownership
- Delays from weather, contractor availability, or permit approvals
- Example for 1 zone:
- O: $30,000 M: $38,000 P: $55,000
- Expected = (30,000 + 4×38,000 + 55,000) / 6 = **$39,500**
- This allows the planning team to:
- Quantify cost variance risks
- Create contingency ranges for stakeholders
- Compare projected vs. actual costs post-implementation

Q4: Can analogous estimation work here? Detailed Answer:

Yes — if a nearby city has recently deployed a similar smart parking system, its total implementation cost can serve as a useful reference.

Example:

- City A deployed 800 sensors for $400,000
- With similar geography and pricing, City B (planning 1,000 sensors) might project around

$500,000–$520,000

Adjustments must be made to:

- Population size and expected usage
- Existing infrastructure (fibre network, power)
- Local labour rates and vendor pricing
- Differing payment systems or compliance rules

Analogous estimation works well in the feasibility phase or when time is limited but should later be replaced with bottom-up validation.

A university launches a new online learning platform, but students begin reporting a range of issues, including quiz submission errors, video buffering, and login failures. To address growing dissatisfaction and restore system trust, the project manager initiates a Plan Quality Management process, using basic quality tools to systematically identify root causes and performance bottlenecks.

Q1: How does a cause-and-effect diagram help with system issues?

A Cause-and-Effect Diagram (also known as a fishbone or Ishikawa diagram) helps visually map out potential causes of system failures, categorized under major headings like People, Process, Technology, and Environment.

In this context:

- Quiz submission failures might stem from:
  - **Technology**: Server overload, unoptimized database queries, software bugs
  - **Process**: Poor session timeout logic, lack of confirmation messages
  - **People**: Inadequate faculty training, student misuse
  - **Environment**: Weak internet infrastructure in certain regions by facilitating a structured brainstorming session, the team can:
- Move beyond guesswork
- Analyse multiple layers of interrelated causes
- Prioritize issues that are most fixable or impactful

This supports the development of targeted action plans (e.g., increase server load capacity, fix auto-save bugs, and improve user instructions).

Q2: How can a control chart be implemented in this context?

A control chart is used to monitor performance trends over time, highlighting whether variations are within acceptable limits or require intervention.

In this project:

- Metrics tracked could include:
  - Login success rate per hour or day
  - Video buffers start time (in seconds)
  - Quiz completion rate without error
- The chart includes:
  - A mean line (average value)
  - Upper and lower control limits (acceptable variation boundaries)
  - Data points plotted over time

If login success drops below the lower control limit or video buffering increases suddenly, it flags a special cause variation requiring immediate attention (e.g., CDN failure, update bug).

Using control charts, the team can:

- Detect performance degradation early
- Justify resource allocation to IT teams
- Validate whether system optimizations are effective

Q3: What purpose does a checksheet serve here?

A checksheet is a structured form for collecting real-time data on recurring problems or behaviors. For the learning platform:

- Students or support staff could log:
  - Type of issue (quiz error, login problem, video lag)
  - Device/browser used
  - Time of occurrence
  - Error messages seen

Over a few days or weeks, this data helps:

- Identify patterns — e.g., most quiz errors occur on Safari during peak hours
- Prioritize fixes — e.g., 40% of issues are login failures, suggesting authentication bottlenecks
- Quantify **impact across modules**, supporting a phased resolution plan Checksheet analysis feeds into **Pareto charts** or dashboards for quality decision-making.

Q4: How would a flowchart improve user navigation?

A flowchart maps the actual user journey through the system, highlighting touchpoints, decision paths, and error points.

Example: A student attempting to submit a quiz

1. Log into portal
2. Navigate to course → quiz
3. Attempt questions → auto-save answers
4. Submit → receive confirmation
5. Log activity is recorded Flowcharting this journey helps identify:

- **Redundant steps** (e.g., multiple confirmations)
- **Poor feedback loops** (e.g., unclear submission success message)
- **Navigation inconsistencies** between desktop and mobile Flowcharts also support:
  - UX audits
  - Improved help documentation
  - Testing protocol design for edge cases

A software development firm working on a healthcare application is encountering defects late in the development cycle. These issues slow down release cycles and raise concerns about QA practices. In response, the quality team adopts a structured Plan Quality Management framework, leveraging standard quality tools to detect and mitigate quality problems earlier in the SDLC (Software Development Life Cycle).

Q1: What role does a fishbone diagram play in this project?

A fishbone diagram is used to explore why defects are occurring and what their root causes might be. It's essential for quality control in complex, regulated projects like healthcare systems.

Categories for root causes may include:

- **Requirements**: Ambiguity, incomplete user stories, missing regulatory
- constraints
- **Design**: Poor architecture decisions, insecure data models
- **Testing**: Lack of unit or integration test coverage, incomplete scenarios
- **People**: Developer inexperience, unclear handoffs between teams
- **Tools**: Misconfigured CI pipelines, outdated test automation scripts

This tool allows the QA team to:

- Move from surface-level error correction to systemic problem solving
- Build a quality culture that emphasizes prevention over rework
- Create a prioritized list of corrective and preventive actions (CAPA)

Q2: How can control charts monitor development progress?

A control chart can visualize trends in defect detection rates, allowing the team to measure whether their quality efforts are improving or regressing.

For example:

- Track defects reported per sprint or per module
- Monitor code coverage percentage or test pass/fail ratios over time
- Use control limits to detect:
  - Sudden spikes in regression bugs (e.g., after a major merge)
  - Consistent patterns of poor test stability in one component

Benefits include:

- Data-driven decision-making
- Early identification of unstable code areas
- Supporting conversations with stakeholders about release readiness

Q3: How should checksheets be used?

In this development setting, checksheets provide structured documentation of defects by category, frequency, and severity. Usage:

- Developers and testers record every defect found during:
  - Unit testing
  - Integration testing
  - UAT
- Fields might include:
  - Module affected
  - Defect severity (low, medium, critical)
  - Type (UI, logic, security, performance) This allows teams to:
- Generate defect heatmaps (e.g., "most bugs come from the user registration module")
- Apply Pareto analysis — 80% of bugs may stem from 20% of modules
- Adjust sprint priorities to focus on high-defect components

Q4: How can flowcharts help with process improvement?

Flowcharts help visualize the software development lifecycle (SDLC) to identify inefficiencies, handoff delays, or gaps in quality checks.

For example: Map out the end-to-end flow: **Requirements → Design → Dev → Code Review → Test → Release → Feedback**

A review of this flow might reveal:

- Skipped test cases when sprint timelines are tight
- Delays in code review due to team overload
- Defects leaking into UAT because integration tests were incomplete

Flowcharts enable:

- Identification of non-value-adding activities
- Streamlining of QA workflows
- Better understanding of team bottlenecks and cross-functional coordination

A fintech startup forms a new Agile cross-functional team to develop a real-time budgeting app that connects to users' bank accounts and visualizes spending patterns. The team includes designers, developers, a Scrum Master, and a Product Owner. Over a 12-week period, they experience the full spectrum of Tuckman's team development stages — from initial bonding to peak performance.

Q1: What are the signs the team is in the Forming stage?

During the Forming stage, team members exhibit behaviors typical of newly created groups where structure and norms are not yet established.

Signs included:

- **Politeness and guarded communication**: Individuals were courteous but reluctant to challenge ideas or express concerns, particularly during initial sprint planning.
- **Uncertainty about Agile roles**: Designers weren't sure how their contributions fit within the sprint cycle; developers weren't familiar with the designer handoff process.
- **Dependency on leadership**: Team members looked to the Scrum Master or Product Owner to define direction, goals, and workflow structure.
- **Lack of shared vocabulary**: Misunderstandings arose around what "Definition of Done" meant, or how Jira tasks should be broken down.

At this stage, the team focused on relationship building, understanding roles, and establishing a psychological safe space.

Q2: How did the team handle the Storming phase?

The Storming stage surfaced as the team transitioned from planning to actual development. With real tasks underway, differences in style and expectations became evident. Key challenges:

- **Conflicts over coding standards**: Some developers favoured rapid iteration; others emphasized strict style rules and unit testing, causing friction during code reviews.
- **Backlog confusion**: The Product Owner was still refining priorities, leading to confusion during sprint planning and mid-sprint rework.
- **Uneven contributions**: Designers felt overwhelmed during UI prototyping, while backend developers had downtime waiting for requirements.

The Scrum Master played a vital role by:

- Facilitating retrospectives and alignment workshops
- Mediating value vs. effort discussions
- Helping establishes a shared Working Agreement (e.g., coding conventions, DoD, use of branches)

This structured conflict resolution helped the team move toward the next stage.

Q3: What defines the Norming phase in this case?

In the Norming stage, the team began developing a sense of rhythm, mutual respect, and shared ownership of the product.

Behaviors included:

- **Collaborative sprint planning**: The team used velocity metrics from previous sprints to realistically plan future work.
- **Tool adoption and process maturity**: They fully adopted Jira for backlog management, created burndown charts, and refined their estimation techniques.
- **Retrospective-driven improvements**: Sprint retros became proactive sessions for identifying bottlenecks (e.g., too many meetings on Mondays), rather than blaming sessions.
- **Mutual support**: When one team member faced deployment issues, others stepped in without being asked, showing collective accountability.

This phase marked the start of self-organization and continuous improvement.

Q4: What is an example of Performing in this team?

During the Performing stage, the team operated with autonomy, efficiency, and innovation. Example outcomes:

- Delivered a working prototype of the budgeting app two days ahead of schedule, with accurate real-time visualizations and a smooth user experience.
- Minimal supervision was required from the Scrum Master, who shifted focus to coaching rather than coordination.
- Developers and designers held cross-functional design reviews, merging their workflows for faster iteration.
- They introduced a "Friday Fixathon" to resolve bugs, mentor juniors, and experiment with small innovations like dark mode.

By this point, the team had built a high-trust, high-output culture focused on value delivery and user-centric outcomes.

A group of university students form a team to work on a smart traffic monitoring system for their final-year capstone. The project includes sensor data collection, real-time visualization, and web dashboard deployment. Like many student teams, they move through the Tuckman stages with limited initial structure and varying technical strengths.

Q1: How did the Forming stage appear in this team?

The Forming stage was marked by initial excitement but functional confusion. Team behavior included:

- **Enthusiasm and ambition:** Students were excited about working with real-world IoT data and creating a visible final product.
- **Unclear task distribution:** There was no clear plan for who would handle front-end development, data processing, or reporting.
- **Over-reliance on the team leader:** The most vocal or experienced member became the de facto lead, while others deferred decisions to them.
- **Low communication frequency:** Meetings were infrequent and unstructured, with minimal follow-up or documented outcomes.
- At this point, the team needed scaffolding in terms of project planning, tools, and role definition.

`Q2: What conflicts happened in Storming?

As tasks progressed, **Storming** emerged through disagreements and frustration over unequal workloads and unclear goals. Examples:

- **Task conflicts**: Two members both began coding the back end without

coordination, leading to duplicated effort.

- **Role dissatisfaction**: One student who preferred programming was assigned to report writing, leading to disengagement.
- **Communication gaps**: Decisions made informally outside meetings weren't shared, creating tension and mistrust.
- **Missed deadlines**: Misalignment on milestones caused delays in integrating sensors with the dashboard.

These conflicts were addressed through:

- An intervention by a faculty mentor
- A candid group meeting where everyone expressed their concerns
- Agreement on task reallocation based on strengths and interests

Q3: How did they enter Norming?

The Norming phase began once the team reset expectations and introduced more structure. Key practices:

- Created a shared project schedule using Google Sheets and Trello
- Delegated tasks based on each member's expertise (e.g., electrical engineering student handled sensor calibration, CS students took on API dev)
- Instituted weekly sync-ups and midweek updates on group chat to keep everyone aligned
- Documented decisions and created a shared GitHub repo with access controls

The team also developed informal rituals (like quick coffee check-ins) that helped reinforce team cohesion and psychological safety.

Q4: What did the team achieve during Performing?

By the Performing stage, the team was collaborating smoothly and independently to meet their objectives.

Achievements included:

- A fully functional smart traffic monitoring prototype, with live sensor data, analytics, and an intuitive dashboard.
- Seamless division of labour — while some finalized documentation, others demoed the system to faculty.
- High confidence during their final presentation, with live demos, technical Q&A handling, and clear articulation of project impact.
- The team even conducted a self-evaluation, offering each other feedback on both technical and soft skills growth.

WEEK 3

1. Waterfall – Handling Late Requirement Changes in an IT Infrastructure Project

Question: You are managing an IT infrastructure upgrade project using the Waterfall methodology. The client initially approved a plan to migrate their database servers to a private cloud. However, in the testing phase, the client requests switching to a hybrid cloud model instead. What challenges will you face? How will you handle the change within Waterfall methodology?

Challenges: Waterfall projects are rigid, and changes at the testing phase impact design, implementation, and documentation. Switching to a hybrid cloud model requires: (1) Reworking architecture design. (2) Rewriting data migration plans. (3) Delays in deployment and increased costs. (4) **Handling the Change:** (1) Initiate a Change Request (CR) with impact analysis. (2) Present the cost and timeline impact to the client. (3) If approved, revise project documents, conduct additional testing, and re-train teams. (4) If the timeline is fixed, recommend implementing the change in Phase 2 after deployment.

2. Agile – Mid-Sprint Priority Change in a Mobile App Development Project

Question: Your Agile team is working on a mobile banking app and is in the middle of a two-week sprint. The Product Owner (PO) suddenly asks to prioritize implementing biometric authentication instead of a chatbot feature. How should you handle this change without disrupting the sprint?

Answer: Assess the urgency: If it's business-critical (e.g., due to compliance), discuss the impact with the Scrum Master and team. **Check sprint progress:** If little work has been done on the chatbot, swap the backlog items. **Push to next sprint:** If biometric authentication requires significant changes, suggest implementing it in the next sprint. **Communicate with stakeholders:** Explain the impact to the PO and negotiate a feasible approach.

Outcome: Agile allows flexibility, but **protecting sprint focus** is key. Changes should be **prioritized for future sprints** when possible.

3. Scrum – Unavailable Product Owner in an E-Commerce Platform Development

Question: Your team is developing an e-commerce platform using Scrum. Midway through the sprint, the Product Owner (PO) becomes unavailable, leaving a critical user story (payment gateway integration) unclear. How do you proceed without delaying the sprint?

Answer: Refer to existing documentation: Check backlog items, acceptance criteria, and wireframes. **Leverage team expertise:** Developers and testers should make educated assumptions based on previous sprint decisions. **Engage a proxy PO:** Seek guidance from business analysts or senior stakeholders. **Work on other backlog items:** To prevent idle time, prioritize other tasks while waiting.

Outcome: Scrum relies on **self-organizing teams**. Clear documentation and alternative communication channels ensure progress continues.

4: Waterfall – Unexpected Downtime in an IT Network Upgrade Project

Question: You are managing a network infrastructure upgrade using the Waterfall model. During the deployment phase, a critical router upgrade fails, causing unexpected downtime for all office locations. How do you handle the situation within the Waterfall framework?

Answer: Activate risk management protocols: Use the pre-defined contingency plan to revert to the previous configuration. **Conduct root cause analysis:** Identify the failure point and document findings. **Escalate to leadership:** Inform key stakeholders about potential delays and solutions. **Update deployment plan:** Implement phased rollouts or introduce additional testing in future deployments.

Outcome: Unlike Agile, Waterfall requires thorough planning and fallback mechanisms to handle unexpected failures.

5: Agile – Handling a Difficult Stakeholder in a SaaS Development Project

Question: Your team is developing a SaaS-based HR management system using Agile. A key stakeholder demands daily status updates and extensive documentation, disrupting the team's productivity. How do you manage this situation while staying true to Agile principles?

Answer: Educate the stakeholder – Explain that Agile prioritizes working software over excessive documentation. **Offer Sprint Reviews** – Invite the stakeholder to sprint demos instead of daily reports. **Leverage Agile tracking tools** – Provide access to JIRA/Trello dashboards for real-time updates. **Balance expectations** – If reports are necessary, provide a brief automated summary without disrupting workflows.

Outcome: Agile emphasizes **transparency without micromanagement**. Encouraging participation in sprint ceremonies reduces unnecessary reporting efforts.

6: Scrum – Team Failing to Meet Sprint Commitments in a Web Application Project

Question: Your Scrum team is developing a web-based booking system, but each sprint ends with unfinished backlog items. This pattern repeats, affecting overall delivery. How do you improve the team's performance?

Analyse sprint velocity – Check past sprint data and **adjust commitments accordingly**. **Improve backlog refinement** – Ensure stories are broken into smaller, manageable tasks. **Reduce work-in-progress (WIP)** – Encourage the team to **finish tasks before** starting new ones. **Use sprint retrospectives effectively** – Identify blockers and **implement corrective actions**. **Outcome:** Sprint planning should be **data-driven**, and teams must be trained to **estimate realistically**.

7: Waterfall – Handling Fixed Budget Constraints in a Database Migration Project | Question: A company hires you to migrate their database to AWS using Waterfall with a fixed budget and strict timeline. During testing, unexpected compatibility issues require additional third-party tools, exceeding the budget.

How do you manage the situation?

1. **Reassess priorities** – Identify non-critical features that can be postponed.
2. **Optimize existing tools** – Explore **open-source alternatives** instead of purchasing new software.
3. **Negotiate with stakeholders** – If changes are critical, request additional
4. funding approval.
5. **Document a risk report** – Ensure similar issues are planned for in future projects.

Outcome: Waterfall requires **careful budgeting**. Stakeholder communication is key when constraints arise.

8: Agile – Handling Resistance to Change in an IT Service Desk Automation Project

Question: Your company is implementing an **AI-powered IT helpdesk system** using **Agile**, but employees resist using chatbots over human support.

How do you handle resistance to change?

1. **Educate users** – Conduct **workshops and training** to showcase chatbot
2. benefits.
3. **Gather feedback** – Identify user concerns and **incorporate their input into product development**.
4. **Introduce changes gradually** – Implement a **hybrid support model** (AI + human).
5. **Showcase success stories** – Demonstrate how automation improves efficiency
6. and reduces workload.

Outcome: Change management is essential in Agile. **Stakeholder involvement improves adoption rates**.

9: Scrum – Handling Sprint Interruptions in a Cybersecurity Monitoring Project

Question: Your team is building a real-time security monitoring system. Mid-sprint, a critical security vulnerability is discovered, requiring immediate fixes.

How do you balance sprint goals while addressing this urgent issue?

1. **Conduct an emergency backlog grooming** – Reprioritize tasks to accommodate the security fix.
2. **Work in parallel** – Dedicate some team members to the **vulnerability fix** while others continue the sprint.
3. **Update stakeholders** – Inform the PO about the **sprint scope adjustment**.

Outcome: Scrum allows flexibility, but **urgent security issues must be prioritized**.

10: Agile – Handling Scope Creep in a Cloud Migration Project

Question: Your Agile team is migrating a company's on-premises servers to Azure. The client continuously adds new feature requests, delaying project milestones.

How do you prevent scope creep while maintaining Agile flexibility?

1. **Reprioritize backlog items** – Move non-essential features to **future sprints**.
2. Educate the client on MVP (Minimum Viable Product).
3. **Use sprint planning to set realistic goals**.

Outcome: Agile allows **continuous feedback**, but maintaining scope is **essential for timely delivery**.

WEEK 4

1: Developing a Mobile Banking Application Scenario Overview:

A bank is launching a **new mobile banking app** to provide customers with seamless online transactions. The project team must define the project scope, gather requirements, validate deliverables, and manage changes.

Q1: What steps should the project team take to ensure the project scope is well-defined and managed throughout the lifecycle?

A1: The team should create a **Scope Management Plan**, which outlines how the scope will be defined, validated, and controlled, ensuring stakeholder expectations are met.

Q2: What techniques can the team use to collect accurate requirements for the mobile banking app?

A2: Techniques include **interviews, surveys, focus groups, and use case analysis** with customers and stakeholders to identify essential features (e.g., fund transfers, bill payments, biometric login).

Q3: What key documents should be reviewed before finalizing the project scope statement?

A3: The **Project Charter, Scope Management Plan, Requirement Documents, and Organizational Process Assets** should be reviewed to ensure a comprehensive and realistic scope definition.

Q4: How should the project team structure the Work Breakdown Structure (WBS) for this project?

A4: The WBS should be hierarchical, with **major components** like **User Interface, Security Features, Transaction Processing, and Reporting** further broken into **smaller deliverables**.

Validating Scope Q5: How can the project manager ensure stakeholders formally accept the app's deliverables?

A5: By conducting **User Acceptance Testing (UAT)**, obtaining feedback, and ensuring the customer or sponsor signs off before proceeding to deployment.

Controlling Scope Q6: If a stakeholder requests additional features (e.g., cryptocurrency transactions) during testing, how should the project manager handle it?

A6: The request should go through **Change Control**—assessing its **impact on time, cost, and resources** before approval or deferral to a future phase.

2: Implementing a Cloud-Based CRM System Scenario Overview:

A company is moving from a traditional on-premises CRM system to a **cloud-based solution** to improve sales and customer relationship management.

Planning Scope Management Q1: Why is it important to document how the project scope will be managed?

A1: Proper documentation ensures that all stakeholders have clarity on how scope changes, deliverable acceptance, and requirement adjustments will be handled.

Collecting Requirements Q2: How can the team identify must-have vs. nice-to-have features for the CRM system?

A2: By conducting stakeholder interviews and using MoSCoW analysis (Must-have, Should-have, Could-have, Won't-have) to prioritize requirements.

Defining Scope Q3: What should be included in the CRM project's scope statement?

A3: It should define objectives, deliverables (e.g., Customer Data Migration, Sales Reporting, Integration with ERP), constraints, exclusions, and acceptance criteria.

Creating WBS Q4: How should the WBS be structured for a cloud migration project?

A4: The WBS should include Planning, Data Migration, Feature Customization, Security Implementation, User Training, and Go-Live as key deliverables.

Validating Scope Q5: Who should formally accept the new CRM system before deployment?

A5: The executive sponsor, sales managers, and IT director should validate the deliverables through UAT and system testing sign-off.

Controlling Scope Q6: What should the project manager do if the sales team requests real-time AI- driven customer insights after the project scope has been finalized?

A6: The manager should **evaluate the request's impact** on schedule and budget before deciding to **approve, defer, or reject** the change.

3: Developing an E-Learning Platform for a University Scenario Overview:

A university is developing a **new e-learning platform** to provide online courses, student assessments, and virtual collaboration tools.

Planning Scope Management Q1: How can the project team prevent scope creep in this e-learning platform project?

A1: By defining a clear scope management plan that outlines change request procedures and stakeholder approvals before adding new features.

Collecting Requirements Q2: What tools can be used to collect detailed functional requirements for the e- learning system?

A2: Surveys, stakeholder interviews, focus groups, and prototype feedback from faculty and students.

Defining Scope Q3: What are some key elements to include in the scope statement for this project?

A3: Core deliverables (Learning Modules, Video Lectures, Interactive Quizzes, Student Dashboard), constraints (budget, technology limitations), and success criteria.

Creating WBS Q4: How should the WBS be structured for this project?

A4: The WBS should break down into Course Content Development, Learning Management System (LMS) Customization, Student Registration System, Testing, Deployment, and Training.

Validating Scope Q5: How will stakeholders validate the successful development of the platform?

A5: By conducting pilot testing with selected students and professors, ensuring features meet academic and accessibility requirements.

Controlling Scope Q6: Midway through development, a department requests a real-time plagiarism detection tool. What should the project team do?

A6: The team should evaluate feasibility through a Change Impact Analysis and decide whether to implement it in Phase 1 or a future update.

4: Upgrading an IT Security Infrastructure Scenario Overview:

A multinational company is upgrading its **IT security infrastructure** to prevent cyber threats, implementing **firewalls, intrusion detection systems, and employee cybersecurity training**.

Planning Scope Management Q1: What key factors should be considered when planning the scope for an IT security upgrade?

A1: Compliance regulations (GDPR, ISO 27001), risk assessments, security layers, and stakeholder concerns should be addressed in the Scope Management Plan.

Collecting Requirements Q2: What techniques should be used to gather security requirements?

A2: Risk assessments, penetration testing reports, compliance reviews, and cybersecurity workshops.

Defining Scope Q3: How should the security upgrade project's scope be clearly defined?

A3: The scope statement should include specific security enhancements (Firewall Upgrades, Employee Awareness Training, Threat Monitoring), exclusions, and risk mitigation strategies.

Creating WBS Q4: What are the key components in the WBS for this security project?

A4: Threat Analysis, System Hardening, Security Policy Development, Firewall Configuration, Intrusion Detection System Deployment, Training, and Monitoring.

Validating Scope Q5: What process should be followed to validate that the new security measures are effective?

A5: Conduct penetration testing, security audits, compliance checks, and obtain formal approval from the IT security team.

Controlling Scope Q6: If a new security vulnerability is discovered during implementation, what steps should be taken?

A6: Initiate a change request, conduct risk assessment, reallocate budget/resources, and update the implementation plan accordingly.

WEEK 5- Network Diagrams

Q1. What is the primary purpose of Activity on Node (AON) diagrams?

Answer: Activity on Node (AON) diagrams are used to visually represent project activities as nodes (rectangles), showing the sequence and dependencies between tasks using arrows. The primary purpose is to illustrate the flow of tasks and dependencies clearly.

Q.2 What is the significance of the Precedence Diagramming Method (PDM)?

Answer: PDM is an advanced form of AON diagrams that allows four types of logical relationships between activities:

1. **Finish-to-Start (FS)** – Task B starts after Task A finishes.
2. **Start-to-Start (SS)** – Task B starts at the same time as Task A.
3. **Finish-to-Finish (FF)** – Task B finishes at the same time as Task A.
4. **Start-to-Finish (SF)** – Task B must start before Task A can finish. PDM provides greater flexibility in scheduling complex projects.

Q.3 Which diagramming method is most commonly used in project management software and why?

Answer:

AON (including PDM) is the most used in modern project management software, such as Microsoft Project and Primavera, because it does not require dummy activities and provides a more intuitive visualization of task dependencies.

Q.4 What are the advantages of using AON over AOA? Answer:

- **Easier to Read:** Activities are represented as nodes, making the diagram more intuitive.
- **No Dummy Activities:** AON eliminates the need for dummy tasks.
- **Better Software Support:** Most project management tools support AON over AOA.
- **More Flexibility:** AON allows better representation of complex dependencies.

Q5. In which situations is the Precedence Diagramming Method (PDM) most useful?

Answer: PDM is most useful when a project involves overlapping tasks with different types of dependencies. It is especially effective in software development, construction, and large- scale engineering projects where multiple relationships (FS, SS, FF, SF) are necessary for accurate scheduling.

Q6. How does the choice of network diagram affect project scheduling?

Answer:

The choice of network diagram affects project scheduling in the following ways:

- AON (PDM) allows better visualization of dependencies and is easier to update in software tools.
- AOA requires dummy activities and is less flexible but is useful in simpler projects.
- PDM enables detailed scheduling with multiple types of task relationships, making it best for complex projects.

PERT Analysis

Q1. What is the primary purpose of PERT analysis?

Answer: The primary purpose of PERT analysis is to estimate project duration by incorporating uncertainty in task completion times. It helps in planning, scheduling, and coordinating project tasks by considering three-time estimates:

- **Optimistic time (O):** Best-case scenario
- **Most likely time (M):** Normal expected duration
- **Pessimistic time (P):** Worst-case scenario

The PERT formula calculates the expected time (TE) for each activity:

Expected E = (O+4M+P)/6

PERT also determines the **Critical Path** and helps in risk assessment.

Q2. How does PERT differ from the Critical Path Method (CPM)?

Answer: PERT is used for projects with high uncertainty, focusing on probabilistic time estimation using three estimates (O, M, P). **CPM** is used for projects with well-known activity durations, using **deterministic** (fixed) time estimates. **PERT** helps analyse **project risks and time variation**, while **CPM** focuses on cost optimization and scheduling.

Q3. Calculate the expected time for a task if its Optimistic time is 5 days, Most Likely time is 8 days, and Pessimistic time is 17 days.

Answer: Using the **PERT formula**: The expected duration for the task is **9 days**.

WEEK 6- Cost Management Plan

Q1. Why is it important to define a level of accuracy in cost estimates?

Answer: To ensure consistency and avoid unrealistic expectations. Defining the level of accuracy helps the team understand how precise cost estimates need to be. For example, rounding to the nearest $100 ensures uniformity and reduces the risk of over-engineering estimates that offer no practical benefit.

Q2. Why should units of measure be clearly defined in a cost management plan?

Answer: Clearly defining units of measure (e.g., hours, days, dollars, resources) ensures everyone interprets cost and effort in the same way. It prevents miscommunication, especially in global or multi-departmental teams where different standards might exist.

Q3. Why are organizational procedure links, like control accounts, necessary in cost planning?

Answer: Control Accounts (CAs) link cost tracking directly to the Work Breakdown Structure (WBS). They act as checkpoints where scope, budget, and performance are monitored together. This structure allows for better budget control, early detection of deviations, and accountability for specific project segments.

Q4. Why do we use control thresholds in cost management?

Answer: Control thresholds define how much variance is acceptable before action must be taken. For example, a 10% cost overrun might trigger a report or sponsor escalation. They serve as early warning indicators, helping prevent small issues from becoming major budget problems.

Q5. Why should the project sponsor be notified if costs exceed 10% of the budget?

Answer: Exceeding the budget by more than 10% signals a significant deviation from the project plan. Notifying the sponsor ensures transparency and enables executive-level decisions, such as approving additional funding or re-scoping the project to fit within remaining resources.

Q6. Why is a CPI threshold like 0.9 used as a performance trigger?

Answer: The Cost Performance Index (CPI) indicates cost efficiency. A CPI below 0.9 means you're getting less than 90 cents in value per dollar spent. This signals inefficiency and requires immediate action, such as resource reallocation or cost review, to bring the project back on track.

Q7. Why are rules of performance measurement included in the cost management plan?

Answer: Rules such as how to calculate CV (Cost Variance) or CPI ensure consistent, objective assessment of cost performance across teams and time. Without clear rules, metrics could be misinterpreted or manipulated, undermining decision-making.

Q7. Why is it important to define the reporting format and frequency?

Answer: Standardized reporting formats and schedules ensure stakeholders receive timely, clear, and consistent updates on project cost status. This improves communication, helps identify issues early, and maintains trust among stakeholders.

Q8. Why must the cost management plan describe how cost processes are performed?

Answer: Describing processes like estimating, budgeting, and forecasting ensures the entire team knows what to do, when, and how. It reduces ambiguity, improves coordination, and supports successful implementation of the cost management framework.

Q9. Why does the cost management plan support proactive decision-making?

Answer: By defining thresholds, performance metrics, and escalation procedures, the cost management plan enables the team to detect and act on potential issues before they grow. This minimizes surprises and improves the likelihood of project success within budget constraints.

WEEK 7- ISO9001 Quality Management Standard

Scenario 1: TechEdge Solutions – IT Services Company

TechEdge Solutions offers cloud-based IT services. As they grow, customer complaints about service delays and inconsistent experiences increase. To enhance their processes, they are seeking ISO 9001:2015 certification.

Scenario 2: BioPharm Devices – Medical Equipment Manufacturer

BioPharm Devices manufactures hospital-grade diagnostic equipment. Their expansion into international markets requires compliance with ISO 9001 to prove their commitment to quality, risk management, and continuous improvement.

Clause 4 – Context of the Organization Q1: BioPharm Devices is preparing for ISO 9001 certification. As part of Clause 4, what internal and external factors should they evaluate to define their quality management system?

Answer: According to Clause 4.1 and 4.2, the organization must determine external and internal issues relevant to its purpose and strategic direction. For BioPharm Devices, this includes regulatory compliance, customer expectations, technological changes (external), and internal aspects like staff expertise, organizational structure, and process capability. They must also identify the needs and expectations of interested parties such as customers, suppliers, regulatory authorities, and employees.

Clause 5 – Leadership Q2: TechEdge Solutions has documented a quality policy, but employees are unaware of it. Which leadership responsibility under Clause 5 is being neglected?

Answer: Clause 5.1.1 requires top management to demonstrate leadership and commitment to the QMS, including promoting awareness of the quality policy. In this case, the company failed to communicate the policy effectively, resulting in lack of awareness among employees. This impacts alignment with strategic objectives and undermines QMS effectiveness.

Clause 6–Planning Q3: During internal review, BioPharm identifies risk of product recalls due to design inconsistencies. How should they respond per Clause 6?

Answer: Under Clause 6.1, organizations must identify risks and opportunities and plan actions to address them. BioPharm should evaluate the severity and likelihood of design flaws, and implement measures such as enhanced design reviews, testing protocols, and training to mitigate these risks. These actions must be integrated into the QMS and reviewed for effectiveness.

Clause 7 – Support Q4: TechEdge lacks a proper document control system, leading to outdated SOPs in use. Which requirement of Clause 7 is this violating?

Answer: Clause 7.5 emphasizes the need to control documented information. Documents must be reviewed, approved for adequacy, and updated as necessary. TechEdge must establish procedures to ensure availability of current versions, prevent unintended use of obsolete documents, and protect information from loss or unauthorized changes.

Clause 8 – Operation Q5: BioPharm received a large urgent order but skipped quality checks to meet the deadline. Which part of Clause 8 was compromised?

Answer: Clause 8.5 mandates that production and service provision should occur under controlled conditions. This includes ensuring quality checks and inspections are completed to verify product conformity. Skipping these steps compromises product quality, increases risk of failure, and violates ISO requirements.

Clause 9 – Performance Evaluation Q6: TechEdge wants to track how well their customer service team is performing. Which Clause 9 requirement applies?

Answer: Clause 9.1.1 requires organizations to monitor, measure, analyze, and evaluate performance. This includes customer satisfaction, process performance, and QMS effectiveness. Key performance indicators (KPIs) should be established to track metrics like response time, resolution rate, and customer satisfaction scores.

Clause 10 – Improvement Q7: A customer complaint led BioPharm to discover recurring issues in testing process. What should they do according to Clause 10?

Answer: According to Clause 10.2, the organization must implement corrective actions to eliminate the cause of nonconformities and prevent recurrence. BioPharm should conduct
root cause analysis (e.g., 5 Whys, Fishbone Diagram), update procedures or training, and verify effectiveness of the corrective action through follow-up audits.

General Knowledge Q8: Who is responsible for ensuring compliance with the ISO 9001 QMS in both companies?

Answer: Clause 5.1 states that top management holds ultimate responsibility for the QMS. However, compliance is a shared responsibility. Each employee must follow procedures, contribute to quality objectives, and support continual improvement.

Clause 6 – Planning (Quality Objectives) Q9: TechEdge is drafting quality objectives. What characteristics must these objectives meet under ISO 9001?

Answer: Clause 6.2 outlines that quality objectives must be consistent with the quality policy, measurable, monitored, communicated, and updated as necessary. For example, 'Reduce customer complaints by 20% over the next year' is a valid quality objective.

Clause 9 – Internal Audit Q10: How can both organizations use internal audits to prepare for external ISO audits?

Answer: Internal audits (Clause 9.2) help verify conformity to QMS requirements and ISO 9001 standards. They identify areas of noncompliance, opportunities for improvement, and ensure processes are functioning effectively. Findings should be reviewed by management and corrective actions taken.

Clause 4 – Context of the Organization Q11: BioPharm Devices has entered new global markets with stricter compliance laws. According to ISO 9001 Clause 4.2, what must the organization do to address this change?

Answer: Clause 4.2 requires identification of relevant interested parties and their needs/expectations. BioPharm must assess how these new regulatory demands impact the QMS and ensure requirements are reflected in documentation and operations.

Clause 5 – Leadership Q12: TechEdge's CEO has delegated all QMS responsibilities to the quality manager. According to Clause 5.1.1, is this sufficient for ISO 9001?

Answer: No. While delegation is allowed, top management must still demonstrate leadership by being involved in strategic planning, communicating the importance of quality, and ensuring customer focus. Simply assigning tasks does not fulfill leadership obligations.

Clause 7 – Support (Competence) Q13: BioPharm hired new production staff but didn't provide formal training. After defect incident, what ISO 9001 clause was violated?

Answer: Clause 7.2 requires ensuring personnel are competent on the basis of education, training, and experience. BioPharm should have identified training needs and ensured staff were qualified before assigning responsibilities.

Clause 8 – Operation (Design and Development) Q14: TechEdge introduces a new client-facing platform without documenting design inputs or validating outputs. Which part of Clause 8 does this breach?

Answer: This breaches Clause 8.3, which requires planning and controlling design and development processes. This includes capturing design inputs, defining outputs, conducting reviews, and validating the product against customer needs.

Clause 9 – Performance Evaluation (Customer Satisfaction) Q15: TechEdge claims to be "customer focused" but doesn't analyze customer satisfaction data. What clause are they failing to meet?

Answer: Clause 9.1.2 requires organizations to monitor customers' perceptions of whether their needs and expectations have been met. Without data collection and analysis, TechEdge cannot demonstrate customer focus or make informed improvements.

WEEK 9- Risk Management Process

A technology company developing a mobile banking app follows a structured risk management process. They identify high-probability risks like developer turnover and medium-probability, high-impact risks like third-party API failures. Qualitative and quantitative analyses prioritize risks and estimate potential delays and cost increases. To mitigate these, they implement cross-training and knowledge-sharing for developers and secure vendor SLAs. Regular risk reviews help monitor morale and vendor updates. When a developer resigns, the impact is minimized due to prior planning, showcasing effective risk management.

Q1: Why were cross-training developers an effective risk response?

Cross-training developers effectively mitigates risks by reducing dependency on individual team members. It ensures project continuity during turnover, supports flexible task management, and fosters collaboration and innovation. This approach boosts team morale, lowers recruitment and onboarding costs, and aligns with resilience planning, helping the mobile banking app project stay on schedule and within budget despite unexpected staff changes..

Q2: How did quantitative analysis help in this scenario?

Quantitative risk analysis provided clear, data-driven insights into potential project impacts, helping prioritize mitigation strategies. By estimating a senior developer's departure could cause a three-month delay and $300,000 in added costs, the team could assess risk severity objectively. It enabled cost-benefit comparisons of mitigation options like cross-training, supported better resource and schedule planning, and turned subjective concerns into actionable insights. This approach justified preventive investments and facilitated proactive decision-making.

Q3: What role did monitoring team morale play in controlling risks?

Monitoring team morale was vital in controlling risk, as it helped identify early signs of dissatisfaction that could lead to turnover. Regular check-ins, surveys, and open communication allowed managers to address issues proactively, preventing disruptions from resignations. This approach supported workload balance, recognition, and targeted support, acting as an early warning system for people-related risks and enhancing the project's resilience.

Q4. Why was negotiating SLAs with third-party API vendors important?

Negotiating SLAs with third-party API vendors was crucial for managing external risks. SLAs set clear expectations for performance, security, and response times, holding vendors accountable and reducing the chance of integration failures. They also defined escalation procedures and aligned vendor deliverables with project timelines. By formalizing these terms, SLAs helped transfer and mitigate external risks, ensuring service reliability and protecting the mobile app project from major delays or disruptions.

Q5. How do holding bi-weekly risk reviews align with controlling risks?

Holding bi-weekly risk reviews supported the Controlling Risk phase by keeping the team continuously aware of both existing and new risks. These regular meetings allowed for timely reassessment of risks, updates to response plans, and verification of mitigation effectiveness. They promoted accountability, early risk detection, and adaptive planning. In this project, the reviews helped manage developer turnover and API integration challenges by enabling swift interventions and maintaining a culture of ongoing risk awareness and responsiveness.

A construction company building a shopping mall under a tight deadline uses risk management to address potential delays. They identify heavy rainfall as a medium-probability, high-impact risk and estimate it could cause a 45-day delay and $500,000 in extra costs. To mitigate this, they add buffer time, use waterproof materials, and rent water pumps in advance. The team monitors real-time weather updates and adjusts plans accordingly. When unexpected storms hit, the proactive measures keep the project on schedule, demonstrating the success of their risk management strategy.

Q1. Why was studying historical weather important during Planning Risk Management?

Studying historical weather was vital for creating realistic schedules and risk forecasts. By analyzing past patterns, the team identified rainfall as a likely disruption, allowing for early risk recognition, better resource planning, and accurate time buffers. This data-driven approach aligned with Planning Risk Management goals by incorporating real environmental factors into project planning. In construction, where weather can greatly impact progress, this proactive step ensured a more resilient and adaptable project delivery strategy.

Q2. How did qualitative analysis contribute in his scenario?

Qualitative risk analysis was essential in helping the project team prioritize weather-related risks early on. By classifying heavy rainfall as a medium-probability, high-impact risk, the team could focus on critical environmental threats without complex calculations. It also enabled effective risk categorization, clear ownership assignment, and better stakeholder communication. Using tools like risk matrices and expert judgment, the team quickly evaluated and addressed potential delays, improving planning and preventing costly surprises.

Q3. What specific quantitative data influenced the risk response plans?

Quantitative risk analysis provided concrete estimates—like a 45-day delay and $500,000 cost impact from heavy rainfall—enabling informed decisions on finances and scheduling. These data justified investments in preventive measures such as waterproof materials and pumps. It also supported realistic contingency planning and schedule buffering based on worst-case scenarios. Moreover, the analysis helped secure executive buy-in for upfront risk mitigation funding, turning hypothetical risks into actionable, cost-effective strategies.

Q4. How did real-time weather monitoring aid in controlling risks?

Real-time weather monitoring enhanced risk control by providing early warnings that allowed the project team to adjust operations promptly. Using alerts and forecasts, they rescheduled work, deployed protective gear, and activated water pumps to prevent flooding. This proactive approach minimized delays and resource waste, turning weather risks into manageable events. It also improved stakeholder communication and transparency. Overall, real-time monitoring was key to executing effective mitigation strategies, helping keep the construction project on schedule despite weather challenges.

Q5. Why were pre-renting water pumps considered a proactive risk response?

Pre-renting water pumps was a strategic risk response that ensured quick access to vital equipment for flood control. This proactive measure avoided delays and damage from scrambling for resources during emergencies. The modest pre-rental cost was a cost-effective investment compared to potential repair and delay expenses. It enabled swift action, protected construction progress, and boosted team confidence and morale. Overall, it exemplified strong risk management by anticipating and neutralizing a major environmental threat in advance.

An international corporation launching a digital marketing campaign identifies regional advertising regulations as a key risk. They assess regulatory issues as moderate in probability but high in impact, with potential fines of $1 million and $3 million in lost opportunities. To mitigate this, they involve legal experts early, create region-specific content, and set up regulatory approval workflows. Ongoing audits ensure compliance. When stricter regulations arise in two regions, the campaign adapts smoothly due to built-in flexibility, demonstrating the effectiveness of proactive risk management and legal integration.

Q1How did early involvement of legal counsel reduce risk?

Involving legal counsel early in Planning Risk Management reduced regulatory risks by integrating compliance into the campaign from the start. Legal experts ensured ads met regional standards and laws, preventing costly redesigns, fines, and delays. Early legal input streamlined regulatory approvals and supported creative yet compliant marketing strategies. It also helped interpret evolving rules, enabling faster, risk-free campaign launches. This proactive approach turned compliance from a burden into a strategic advantage for successful market entry.

Q2 What role did qualitative analysis play in prioritizing regulatory risks?

Qualitative risk analysis prioritized regulatory compliance risks by evaluating their likelihood and business impact using tools like probability-impact matrices and expert judgment. It identified compliance violations as high-impact, moderate-likelihood threats, ensuring focused attention and resource allocation. This process helped develop contingency plans, improved communication with stakeholders, and justified cautious campaign design choices. Overall, qualitative analysis laid the foundation for effective risk mitigation in a complex regulatory environment.

Q3 Why was creating region-specific materials an effective risk response?

Creating region-specific marketing materials was an effective risk response that ensured compliance with local laws and cultural norms. This tailored approach avoided regulatory violations and audience alienation, reducing legal and reputational risks. It also enhanced campaign effectiveness and allowed for quick adjustments to changing regulations. Strategically, it demonstrated cultural sensitivity and boosted brand reputation, serving as a preventive control that increased the marketing initiative's resilience and adaptability.

How did quantitative analysis strengthen executive buy-in for proactive measures?

Quantitative risk analysis provided clear financial estimates—$1 million in fines and $3 million in lost opportunities—that helped secure executive support for compliance investments. By translating regulatory risks into measurable business impacts, it demonstrated the ROI of spending on legal counsel, content customization, and approval processes. The analysis also simulated worst-case scenarios, highlighting reputational and financial risks. This data-driven approach reframed compliance as a strategic investment, enabling better resource allocation and cross-functional cooperation, making risk management integral to project success.

5 Why are regular compliance audits crucial during campaign execution?

Regular compliance audits are essential for ensuring ongoing adherence to regulations throughout the campaign. They help detect and address minor issues before they become serious violations, maintain accountability, and provide documentation of due diligence. Audits promote transparency and a proactive approach to emerging risks, enabling quick adjustments to regulatory changes and minimizing disruptions. Far from being mere formalities, they are vital tools for sustaining compliance and resilience throughout the campaign.

WEEK 11 - Tuckman Scenarios

A team of university students is working on a capstone project to develop an online mental health support chatbot. The team includes a mix of CS majors and psychology students. They meet weekly and follow Agile principles.

What are the signs the team is in the Forming stage?

Team members were polite but unsure how their roles fit in. Psychology students were confused about contributing to chatbot logic, and CS students needed more guidance. Everyone relied on the tutor to define the workflow.

How did the team handle the Storming phase?

There were disagreements on tone and content for responses. Developers wanted simplicity, but psychology students emphasized empathy. The team lead held a workshop to co-design the chatbot tone, resolving conflicts.

What defines the Norming phase in this case?

The team established shared vocabularies, created a communication schedule, and began peer reviewing chatbot responses before integration.

What is an example of Performing in this team?

They completed user testing ahead of schedule and integrated real-time feedback collection. Team members supported each other across roles.

Earn Value Management (EVM)

PV (Planned Value):	**CPI (Cost Performance Index):**
PV = Planned Completion % × BAC	**CPI = EV / AC**
EV (Earned Value):	CPI < 1: Over Budget
EV = Work Completed % × BAC	CPI > 1: Under Budget
AC (Actual Cost):	**SPI (Schedule Performance Index):**
(Actual cost incurred)	**SPI = EV / PV**
BAC (Budget at Completion):	SPI < 1: Behind Schedule
(Total budget)	SPI > 1: Ahead Schedule
CV (Cost Variance): CV = EV – AC	**EAC (Estimate at Completion):**
CV < 0: Over Budget	**EAC = BAC / CPI**
CV > 0: Under Budget	($ cost more than planned)
SV (Schedule Variance):SV = EV – PV	**VAC (Variance at Completion):**
SV < 0: Behind Schedule	**VAC = BAC – EAC**
SV > 0: Ahead Schedule	VAC < 0: Over Budget by $
	VAC > 0: Under Budget by $

1. Actual time taken = Effort / Productivity
2. Cost = Time taken × $/Hr
3. Available Time = Actual Time / Availability