Unit 2: Study: Why Projects Fail and Gathering Requirements Exercise

1. **Suggest mitigations for common project failures.**

While project failure may seem like a roadblock, it's a stepping stone towards improvement. The real key is in the post-mortem analysis, an integral part of any project plan, particularly when a project hits a snag (Malsam, 2018). This powerful analysis tool equips project managers to learn from mistakes, adapt strategies, and set the stage for future triumphs.

According to Kissflow (2020), a project can only be deemed successful if it is completed within the agreed-upon budget and timeline while meeting all the requirements. Ultimately, the stakeholders' satisfaction with the outcome determines whether the project was a success or failure. Moreover, it is crucial to meet financial forecasting and ROI targets. In addition, Asana (2022) has identified seven significant project management risks and provided recommendations on how to deal with them, as outlined below:

1. **Scope creep:**

Scope creep, also known as undefined project objectives, is a risk that emerges when the project's scope is not communicated to stakeholders. This lack of clarity can lead to changes in the project's direction and objectives, potentially derailing the project.

Mitigation: To prevent scope creep, it's crucial to establish clear project parameters from the start and communicate them effectively to stakeholders. Regular progress check-ins should be scheduled to ensure the project remains within the original scope, emphasising the urgency and importance of clear communication in project management.

2. **Low performance**

Performance risk refers to a project's performance below expectations. Potential project risks, such as time constraints and team miscommunication, can be identified, and measures can be implemented to prevent such issues.

Mitigation: Preventing low performance through early planning and project management software allows for real-time tracking, thorough planning, and open communication among team members.

3. **High costs**

Cost risk arises when a project exceeds the initial budget due to unrealistic or insufficient budgeting during the planning phase. Creating a detailed list of project elements and their costs can help anticipate project needs and mitigate high costs.

Mitigation: To reduce high costs, accurately estimate project elements and adhere to your budget. Create a project plan template aligning deliverables, scope, and schedule. Schedule regular check-ins during development to review your budget and pacing.

4. **Time crunch**

Time risk, also known as project schedule risk, refers to the possibility of project tasks taking longer than anticipated, potentially impacting the budget, delivery date, and overall performance. During the initial planning phase, project managers often underestimate the time it takes team members to complete a project.

Mitigation: To mitigate time risk, overestimate task completion time in the planning phase and build in time contingency. Create a project schedule using a timeline or Gantt chart, and understand work dependencies and delays to adapt to time risk. Understanding the project lifecycle helps determine task duration.

5. Stretched resources

Resource risk arises when a project requires insufficient resources, such as time, skills, money, or tools. Depending on the project size, project managers are responsible for procuring and communicating resources, typically 1-2 months before project execution.

Mitigation: To minimise resource risk, create a resource allocation plan that maximises team resources, supports team goals, and ensures they are used effectively. This will minimise the chance of running out of resources later.

6. Operational changes

Operational risk refers to unexpected changes in company or team processes, such as role shifts, management changes, or new methods, which can cause distractions, require workflow adjustments, and potentially affect project timelines.

To mitigate operational mishaps, prepare your team for upcoming team shifts or process changes by ensuring they are ready and have time to adjust through meetings, scheduling tools, or additional training.

7. Lack of clarity

Lack of clarity in project management can result from miscommunication, vague project scopes, or unclear deadlines. This can lead to budget overruns, missed deadlines, changes in requirements, pivoting project direction, or disappointing outcomes.

Mitigation: To avoid clarity in project planning, review and recheck requirements, ensure everyone is on the same page, developers are prepared for the next phase, and the scope is clearly defined. Keep project information accessible and updated to keep everyone updated.

2. Carry out requirements gathering using Gherkin.

Requirement gathering is crucial for collecting and documenting information to understand the client's needs and define project objectives. It also guides product or service development (Sire, 2023). Smartbear (2023) states that Gherkin is a simple, plain-text language that non-programmers can easily understand. It allows concise descriptions of test scenarios and examples to illustrate business rules in real-world domains. Furthermore, Nicieja (2017) explains that Gherkin is a language used for writing requirements in Specification by Example (SBE), which helps bridge the gap between stakeholders and technical teams. It provides a clear and structured way to express requirements.

Gherkin is a descriptive language used for writing requirements under Specification by Example (SBE), as explained by Carballo (2022). It is part of the Domain Specific Language (DSL) family and is easily understood by all roles involved in the software development process. The basic syntax of a file written in Gherkin language consists of a feature, business definitions, assumptions, background, scenario, scenario outline, and **When**.

Gherkin uses the **Given**, **When**, and **Then** keywords to establish preconditions for executing requirements. The Given keyword initialises data, creates instances, and sets the database state. The **When** keyword describes the user's essential action on the system, which triggers the test. The **When** command describes interaction with a web page, another user interface element, an API, or a library function, as stated by Carballo (2022). Moreover, the **And** and **But** words allow multiple **Givens**, **When's**, and **Then** statements within the same test scenario. The **And** word replaces the repetition of **Givens**, **When's**, and **Then's**, while the **But** word describes an adverse action or result, making the code more understandable to readers.

Collecting project requirements is a complex process that can be time-consuming and costly. To avoid communication failure, use the Gherkin language to simplify the process. Gherkin is a natural language that helps business analysts, and developers create scenarios for functional tests. Its specific syntax allows for easy understanding and discussion of requirements. The requirements collected in Gherkin are saved to a feature file, which can be used for automated tests in Behaviour-driven development using Cucumber (Nowacki, 2016).

Here is a specification written in Gherkin based on the examples of Nicieja (2017):

```
Feature: Setting starting points and destinations

  Scenario: Starting point should be set to current location

    Given a commuter that enabled location tracking
     When the commuter wants to plan a journey
     Then the starting point should be set to current location

  Scenario: Commuters should be able to choose bus stops and
locations

     Given a bus stop at Edison Street
       And a Edison Business Center building at Main Street
      When the commuter chooses a destination
      Then the commuter should be able to choose Edison Street
       But the commuter should be also able to choose Edison
Business Center
```

References:

- Malsam, W. (2018). *5 Notorious Failed Projects & What We Can Learn from Them*. [online] ProjectManager.com. Available at: https://www.projectmanager.com/blog/failed-projects.
- Kissflow (2020). *Project Failure | 6 Reasons Why Project Fails and How to Avoid It*. [online] Kissflow. Available at: https://kissflow.com/project/why-projects-fail/.
- Asana (2022). *7 Common Project Risks and How to Prevent Them*. [online] Asana. Available at: https://asana.com/resources/project-risks.
- Sire, T. (2023). *How to Gather Requirements as a Business Analyst (Guide)*. [online] Available at: https://www.requiment.com/how-to-gather-requirements-as-a-business-analyst/.
- Smartbear (2023). *Writing scenarios with Gherkin syntax | CucumberStudio Documentation*. [online] Available at: https://support.smartbear.com/cucumberstudio/docs/bdd/write-gherkin-scenarios.html#:~:text=To%20create%20Gherkin%20test%20scenarios [Accessed 24 Mar. 2024].
- Nicieja, K. (2017). *Chapter 1. Introduction to specification by example and Gherkin · Writing Great Specifications: Using Specification by Example and Gherkin*. [online] Available at: https://livebook.manning.com/book/writing-great-specifications/chapter-1/52 [Accessed 25 Mar. 2024].
- Carballo, P. (2022). *Writing Specification by Example Requirements with Gherkin*. [online] Available at: https://www.octobot.io/blog/gherkin-specification-by-example/.
- Nowacki, Ł. (2016). *Speak Gherkin and learn how to collect requirements for your project*. [online] Neoteric. Available at: https://neoteric.eu/blog/speak-gherkin-and-learn-how-to-collect-requirements-for-your-project/ [Accessed 25 Mar. 2024].