# PROG6001 MANAGING SOFTWARE DEVELOPMENT PROJECTS

## Assessment 1

**PART 1**

Considering the PayId implementation scenario, write a proposal following the general structure.

* Executive summary

Upon understanding the need to streamline the sharing of sender/receiver payment PayId details for the bank’s mobile app, the team is focused on implementing a QR code feature that would allow users to generate a code that would be scanned by the other party, minimising the possibility of an error on the payment information. Once scanned, the QR code would automatically fill in the required information, thus simplifying the payment process.

* Introduction

[Explain briefly about the real-time payment, why it is important/useful. Who it is benefited the financial section in other counters such as financial inclusion, low-cost payment etc ...... also explain how Australian credit card payment .....]

* Background

Historically, it’s known that the app receives payment via Osko and that there is a need for the implementation of a QR code feature aiming to facilitate the transfer of payment details amongst users.

User Story – [what is user story ...]

[General format of the user story.]

Project user story.

As a testing strategy, the team could apply the V Model. For that, the team needs to clearly outline the requirements for the new payment feature, its use, and integration; design the look and architecture of the interface; conduct unit testing followed by integration and system testing to ensure the new feature works well and cohesively with the rest of the application; run user acceptance testing to validate that the new feature meets user’s expectations and needs.

* Proposed System/Design

The goal of the project is to enhance the user experience and minimise the chance of human-introduced errors by promoting the transfer of payment details through a QR code. The new feature will be tested to ensure the feature is well accepted by end users, and bugs are found and corrected timely. As deliverables, the team aims to provide an upgraded version of the current app with the implemented feature; user documentation, and a built-in pop-up guide to instruct the user; on further documentation.

Considering that the team is experiencing issues with the consistency of their product deployments, and that the isolation with minimal collaboration. This has resulted in numerous integration issues and project delays. In order to streamline its development process and minimize conflicts, the company has decided to implement Agile methodology in its development. The developers now follow SCRUM and have regular SCRUM meetings, work in Sprints, and have regular Sprint Reviews. This has managed to set the development on the right track. Let us have a detailed look at how the SCRUM master and the team will address Change Management, Version Management, System Building, and Release Management.

1. Change Management:

In every scenario, encouraging open communication and collaboration can be a vital step in embracing changes to the development methods. This helps people involved in the project to share proposed changes and gather necessary feedback. Sprint reviews can be a good platform for these kinds of discussions. After the end of a sprint, developers can come together to discuss the progress and problems faced in the previous sprint and gather feedback from the project manager and the clients. Additionally, implementing the use of version tracking like Git to track changes and improve collaborations can help improve transparency among developers and project managers.

Changes can be difficult and transitioning to a new development methodology can be confusing. To counter this, training and support can be provided early on in the transition.

2. Version Management:

While making a transition, standardised version control practices should be set in place to facilitate the changes in the working structure and environment. Development works can be branched and parallel development can be performed while posting commits in Github to manage repositories, branches, and merges efficiently. It is essential to test the products developed in individual sprints. This can be repetitive and time-consuming. To combat this issue, automated testing can be implemented for numerous integrations to validate changes throughout various sprints.

3. System Building:

Developers have their own choice of IDEs, development platforms as well as operating systems. Collaborating across all these differences might be a hassle. In order to make collaborations seamless, technologies like Docker and containers in Dockers can be used to ensure efficient collaborations between development, production, and testing environments. Likewise setting up a centralized workstation and running off of virtual machines can be another alternative.

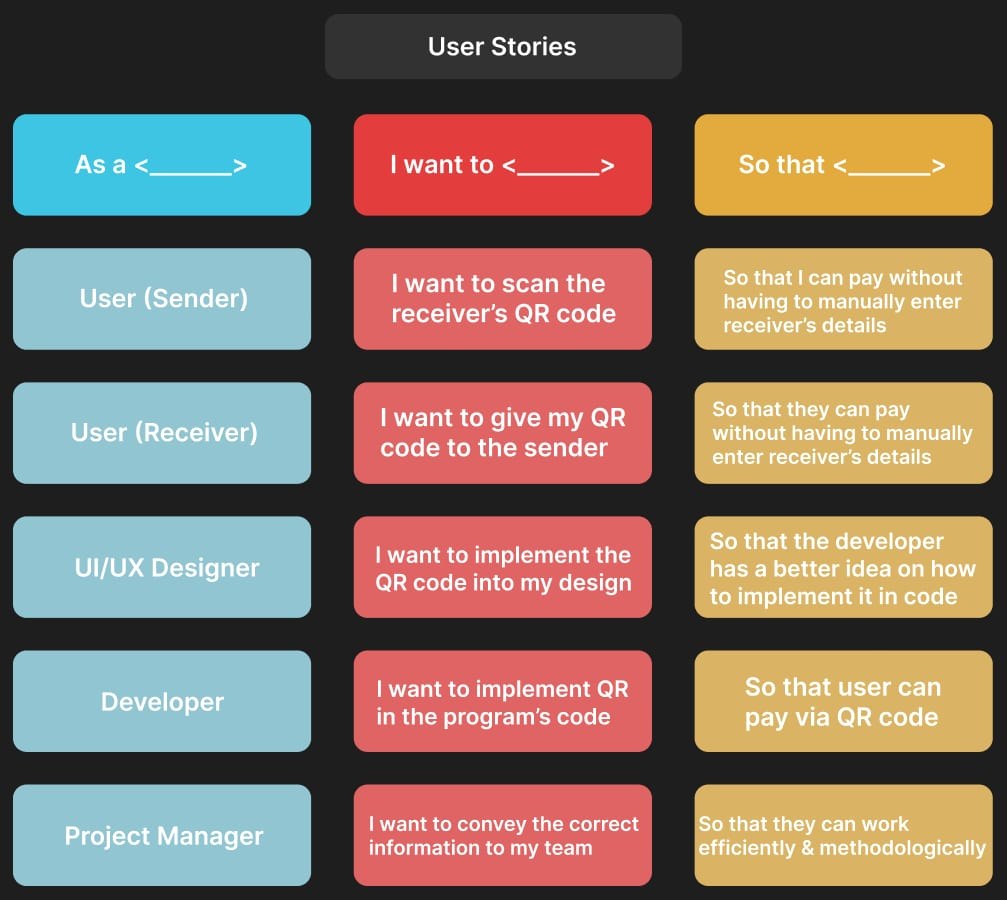
Likewise, it is also essential to establish a versioning scheme to prepare for possible unfixable errors in the code. In those situations, developers can just choose to roll back to a previous stable version and save a lot of time and workforce. It is also essential to be careful to ensure that such disasters can be avoided in the future.

Another way of ensuring efficiency is to monitor building performance and optimise the code and resource allocation. This can help reduce build times and improve overall software performance. When working on large projects and several builds of the program, a reduction in build time will improve the efficiency and productivity of the development team as a whole.

4. Release Management:

A clear management process should be laid out. Planning and scheduling will make a major improvement in meeting release deadlines and avoiding carryovers from one Sprint to another. Using planning tools like Trello or Jira can give the team a better idea of the project's current status and things that are to be worked on in order of urgency. Setting achievable milestones for Sprints is very important as it makes sure that tasks have the required amount of time to be worked on. Not doing so might lead to important tasks being left for another Sprint which in turn makes the next Sprint more crowded and meeting the deadlines for the next Sprint might also be difficult. If not taken care of; such carryovers can keep compiling over different Sprints leading to missed deadlines again.

Properly documenting and setting points to be rolled back to make proper contingency plans will help lower the risk of failed development or critical issues during development.



Considering that the team is experiencing issues with the consistency of their product deployments. The developers work in isolation with minimal collaboration. This has resulted in numerous integration issues and project delays. In order to streamline its development process and minimize conflicts, the company has decided to implement Agile methodology in its development. The developers now follow SCRUM and have regular SCRUM meetings, work in Sprints, and have regular Sprint Reviews. This has managed to set the development on the right track. Let us have a detailed look at how the SCRUM master and the team will address Change Management, Version Management, System Building, and Release Management.

**PART 2**

[ Give a small intro. about change, version management, system building and release Management]

1. Change Management:

In every scenario, encouraging open communication and collaboration can be a vital step in embracing changes to the development methods. This helps people involved in the project to share proposed changes and gather necessary feedback. Sprint reviews can be a good platform for these kinds of discussions. After the end of a sprint, developers can come together to discuss the progress and problems faced in the previous sprint and gather feedback from the project manager and the clients. Additionally, implementing the use of version tracking like Git to track changes and improve collaborations can help improve transparency among developers and project managers.

Changes can be difficult and transitioning to a new development methodology can be confusing. To counter this, training and support can be provided early on in the transition.

2. Version Management:

While making a transition, standardised version control practices should be set in place to facilitate the changes in the working structure and environment. Development works can be branched and parallel development can be performed while posting commits in Github to manage repositories, branches, and merges efficiently. It is essential to test the products developed in individual sprints. This can be repetitive and time-consuming. To combat this issue, automated testing can be implemented for numerous integrations to validate changes throughout various sprints.

3. System Building:

Developers have their own choice of IDEs, development platforms as well as operating systems. Collaborating across all these differences might be a hassle. In order to make collaborations seamless, technologies like Docker and containers in Dockers can be used to ensure efficient collaborations between development, production, and testing environments. Likewise setting up a centralized workstation and running off of virtual machines can be another alternative.

Likewise, it is also essential to establish a versioning scheme to prepare for possible unfixable errors in the code. In those situations, developers can just choose to roll back to a previous stable version and save a lot of time and workforce. It is also essential to be careful to ensure that such disasters can be avoided in the future.

Another way of ensuring efficiency is to monitor building performance and optimise the code and resource allocation. This can help reduce build times and improve overall software performance. When working on large projects and several builds of the program, a reduction in build time will improve the efficiency and productivity of the development team as a whole.

4. Release Management:

A clear management process should be laid out. Planning and scheduling will make a major improvement in meeting release deadlines and avoiding carryovers from one Sprint to another. Using planning tools like Trello or Jira can give the team a better idea of the project’s current status and things that are to be worked on in order of urgency. Setting achievable milestones for Sprints is very important as it makes sure that tasks have the required amount of time to be worked on. Not doing so might lead to important tasks being left for another Sprint which in turn makes the next Sprint more crowded and meeting the deadlines for the next Sprint might also be difficult. If not taken care of; such carryovers can keep compiling over different Sprints leading to missed deadlines again.

Properly documenting and setting points to be rolled back to make proper contingency plans will help lower the risk of failed development or critical issues during development.