PROG6001 Managing Software Development Projects

ASSESSMENT 1 COVER SHEET

For use with online submission of the report

Please complete all of the following details and then make this sheet the **first page of your report – do not send it as a separate document.**

Students Name: Muhammad Sheheryar Tariq
Student ID: 24819196
Unit Name: Managing Software Development Projects
Unit Code: PROG6001
Assignment No.: Assignment 1,
Assignment Title: Project Proposal Document
Due date: 23/03/2025
Date submitted: 23/03/2025

Assignment - Part A

Proposal

PayID QR Scanner Integrations

Table of Contents

Executive Summary

This initiative aims to broaden the scope of a bank's mobile payment application using the PayID feature by streamlining its features. At present, users need to manually input the PayID details which the system Osko works on. As with any human procedure, this one has a tendency of making mistakes, thus deteriorating the overall customer satisfaction. The new functionality will let users effortlessly scan a QR code that holds the PayID details and the payment fields will be completed automatically. In Australia, the New Payments Platform (NPP) is also looking for ways to make digital payment transaction easy, quick, and safe for users, and this improvement is in line with their wider objectives. This will be achieved through user story formulation, defining the development of testing methods and procedures, and a clear strategy for execution.

Background

Context and Rationale

In Australia, payments are done in real time, 24/7, due to the existence of The New Payments Platform (NPP). With PayID, users can effortlessly send and receive funds by using easy identifiers, such as emails or phone numbers, instead of their bank account numbers. Nonetheless, the manual PayID input currently in use tends to be problematic, often causing errors, delays, and even failed payments which lower a user's confidence.

Reference to Literature

Pillai (2023) discusses the efficacy of PayID and Osko in making payments seamless and instant, pointing out that the accuracy and overall experience of the payment system requires enhancement.

User Stories

User Type	User Story	Benefit
Busy Professional	As a busy professional, I want to scan a payee's PayID QR code using my mobile banking app to auto-fill their payment details for a fast Osko payment.	Enables a busy professional to complete transactions swiftly during a hectic workday, saving time and preventing data entry errors.
Freelancer	As a freelancer, I want to use my banking app to scan a vendor's PayID QR code on an invoice and automatically capture their account details for a quick Osko payment.	Helps the freelancer pay invoices quickly and accurately, so they can maintain good business relationships and focus on billable work instead of paperwork.
Small Business Owner	As a small business owner, I want to use the mobile app to scan my supplier's PayID QR code from their invoice, auto-populating the payment information for a quick Osko payment.	Streamlines the business's accounts payable by reducing manual data entry, ensuring suppliers are paid promptly and without mistakes.
Senior Customer (Less Tech-Savvy)	As a senior customer with limited tech skills, I want to scan a PayID QR code using the banking app to automatically fill in the payee's details instead of typing them myself.	Makes digital payments easier for an older or less tech-savvy user by eliminating the need to type out details, giving them confidence to send money without mistakes.

Busy Parent	As a busy parent, I want to use the mobile app to scan a PayID QR code when paying my child's tutor or babysitter and have their payment details auto-filled instantly.	Allows quick, on-the-go payments even while multitasking, ensuring the right person is paid and freeing up time for family priorities.
Bank Support Representative	As a bank support representative, I want to guide customers in using the PayID QR scanning feature so they can auto-fill payee details during Osko payments.	Reduces manual entry errors and related support calls, allowing staff to assist customers more efficiently and improving overall customer satisfaction.

Testing Strategies

To ensure the PayID QR code scanner functions as intended and improves the user experience, a structured testing approach will be adopted. The primary strategy will be User Acceptance Testing (UAT), supported by functional, integration, and usability testing.

Test Type	Purpose	Details
User Acceptance Testing (UAT)	Validate the feature with real users in realistic scenarios.	A select group of customers will scan QR codes from various sources (paper, screen, printouts) under different conditions (lighting, angles).
Functional Testing	Ensure the scanner correctly captures and parses PayID details.	Test against all supported PayID types (email, phone number, ABN). Confirm autofill and field validation.
Integration Testing	Verify system integration between the scanner, PayID lookup, and Osko payment modules.	Simulate end-to-end payment flow from scan to transaction completion, including failed lookup scenarios.

Usability Testing	Assess user experience, accessibility, and ease of use.	Measure time to complete a payment, error rates, and user satisfaction through surveys and observation.
Security Testing	Ensure scanned data is securely handled and not vulnerable to injection or spoofing.	Validate encryption of PayID data in transit, verify QR validation protocols, and check against spoofed QR codes.
Format Compatibility Testing	**JPEG:** Test QR detection from various quality levels; handle non-QR images with clear errors. **PNG:** Ensure accurate scanning of PNGs, including transparent or corrupted files. **PDF:** Verify QR extraction from embedded PDFs; show errors if no QR is found. **SVG:** Test decoding of SVG-based QR codes across devices and browsers. **HEIC:** Check support for HEIC files from iOS or convert them automatically.	Conduct QR test scans using files such as JPEG, PNG, PDF, SVG, and HEIC. Verify, for each format, that QR code image quality is accurately captured and confirm that there is valid error handling for unsupported or corrupted files. Ensure that the images are readable by various devices at different resolutions and compression rates.

Proposal

Goals

Improve the Payment Process: Allow users to start transactions using a PayID QR code for real-time payments to make the system more user-friendly and efficient.
Decrease Manual Entry and Errors: Reduce user PayID entry mistakes that can lead to failed or erroneous payments, either through the wrong direction or the wrong account.
Encourage Safe and User-Friendly Transactions: Facilitate the use of modern digital transaction systems that are quick, dependable, safe, and user-centric—meeting the expectations and capabilities of consumers and the New Payments Platform (NPP).

Deliverables

Module for QR Code Scanning: A fully integrated payment workflow within the application includes a working payment QR code scanner that can recognize and retrieve PayID data from QR codes.

System Validation on the Backend: Logic on the server verifies and validates the scanned PayID for secure data and issues payment through Osko after the PayID is processed.

Design Evolution of the UI/UX: A redesigned payment page that encompasses the fallback of manual detail input, retrieval verification, and QR code confirmation scanning.

Support Documentation Guides: Maintenance users are provided with explanatory technical documents while the general app user is provided with the guide regarding the new feature.

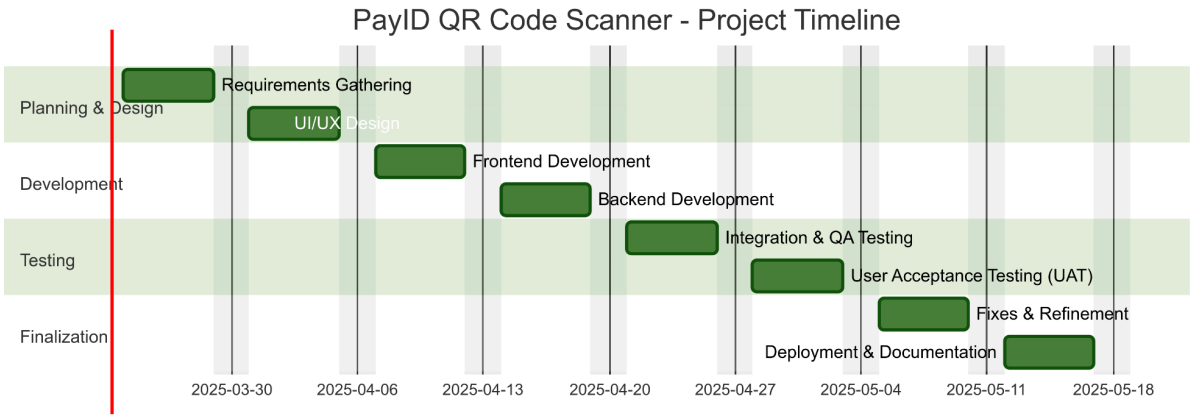
Report on Testing and Feedback: Summary after UAT (User Acceptance Testing): documents results of testing, feedback and comments, issues encountered, and suggestions made before the last delivery.

Timeline

Week	Activity
Week 1	Gather further details via meetings, look for appropriate stakeholders and check the running feasibility.
Week 2	Initial payment task workflow drawings for payment modification and features MVP design with QR scan integration.
Week 3	Construction of user interface components for the QR code scanning features and other related components from the payment UI.
Week 4	Validation and payment processes backend: PayID parsing validator and execution environment security processing workstation.

Week 5	Verification on all known platforms and regions; Internal QA for system integration. Make sure the system passes format compatibility testing.
Week 6	Selected users participate in User Acceptance Testing UAT.
Week 7	Resolve UAT issues, perform refinements, and finalize performance and security enhancements.
Week 8	Final deployment to production, prepare technical documentation and user training materials.

Gantt Chart



Assignment - Part B

Plan

Enhancing Team Collaboration and Deployment Efficiency

Problem

A lack of collaboration along with a deficiency in structured processes are the main causes of deployment inconsistency and failed integration within the team.

Siloed Development: Developers operate in isolation which causes a lack of necessary communication, causes duplication of effort and last minute integration conflicts.
Weak Version Control: There is no explicit strategy set for Git; this lack of clear workflows results frequent merge conflicts and change tracking becoming difficult.
Manual Builds: Builds are unreliable because they are triggered manually and are tailored to specific environments, fostering environment specific bugs.
Unstructured Releases: There is no adequate testing or rollback plan that accompanies the releases, causing the deployments to be unstable.

Impact: The overall effect leads to stakeholders to lose trust, the project is often delayed, the release is often filled with bugs, and productivity is heavily affected.

Plan

To solve these issues, a comprehensive strategy will be implemented across four critical domains of software engineering best practices:

1. **Change Management**
2. **Version Management**
3. **System Building**
4. **Release Management**

Change Management

Problem

Changes made by developers are not tracked, communicated, or assessed for impact.

Solution

Institute an organized formal Change Management Process where an Impact Assessment determines how and when changes are done to a project.

Key Actions:

- Use a **change request form** (via Jira/ClickUp) for all major updates.

- Assign a **Change Control Board (CCB)** or tech lead to review and approve changes.
- Require impact analysis and rollback plans before approval.
- Schedule regular **scrum meetings** to discuss ongoing or upcoming changes.

Version Management

Problem

Developers are either erasing code or changing it due to lack of a structured approach to versioning.

Solution

Implement a switching system of version control through Git and use a branching feature such as GitFlow.

Key Actions:

- Use **feature branches** for isolated development.
- Merge through **pull requests** and conduct **code reviews**.
- Tag releases using **semantic versioning** (e.g., v1.2.0).
- Maintain clear branch naming conventions (feature/, hotfix/, release/).

System Building

Problem

Manually building does not sit well, there is deviation in configuration and specific bugs to the environment cause issues.

Solution

Put in place automation of the build process through the use of CI/CD tools and containers.

Key Actions:

- Implement **CI pipelines** (GitHub Actions, Jenkins) to run builds on every push.
- Automate dependency installation and run tests after each build.
- Use **Docker** to containerize the application for consistent environments.
- Store artifacts in a central repository (e.g., GitHub Packages).

Release Management

Problem

There is a significant lack of communication in deployments often leads to errors sprouting at the last moment. Also, deployments are often done in an inconsistent manner.

Solution

Establish a release process that incorporates testing and documentation for better workflow control.

Key Actions:

- Define a **release schedule** (e.g., every 2 weeks).
- Use **checklists** to validate testing, approvals, and documentation.
- Implement **automated deployments** to staging and production environments.
- Maintain **rollback strategies** using versioned releases or infrastructure snapshots.

References

OpenAI. (2024). *ChatGPT* (Mar 2024 version) [Large language model]. https://chat.openai.com

Pillai, B. (2023). *Understanding real-time payment dynamics in Australia*. TechRxiv. https://www.researchgate.net/publication/374823771_Understanding_Realtime_Payment_Dynamics_in_Australia