Urmila_A1_ProjectProposal

by Urmila Urmila

Southern Cross University

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.**

Group No: 03

Students Details:
Unit Name: Managing Software Development Projects
Unit Code: PROG6001
Assignment No.: Assignment 1,
Assignment Title: Project Proposal Document
Due date:
Date submitted:

PART A: Project Proposal (QR Scanner Integration)

Executive Summary:

The project proposes enhancing QR code scanning functionality to our bank's PayID payment system that exists within the mobile application. Currently, PayID details exchange between users exists through manual input which produces errors that deteriorate the user experience. The integration of a PayID QR code scanner will optimize payments through a user-friendly system which supports Australia's shift towards NPP real-time payments (Pillai, 2023).

Background:

History: The PayID service enables more manageable payments through phone numbers, emails, and ABNs instead of demanding that users use complicated banking information. The current practice of manual PayID identifier sharing produces problems because users need to type or copy the details by hand. Through the New Payments Platform (NPP), Australia maintains a real-time payment system that supports efficient, secure, and seamless financial transactions. The proposed solution uses NPP's infrastructure through the "Osko" layer service, enabling users to accomplish instant secure transactions between banks.

User Stories:

Personal User: Quickly pay a friend using a QR Code

As a banking app user, I will scan my friend's PayID QR code, and payment details will automatically be entered into my banking app. Then, I can quickly transfer the amount without manually typing in their PayID details. The following are the steps:

- Select "Pay with QR code".
- Scans a PayID QR code using their phone camera.
- PayID details populate automatically, showing payee’s name.
- Input the amount and pay.

- The system confirms transaction completion and provides instant feedback via Osko notification.

Small Business/ Merchant: Easily accept payments through a QR code

As a small business owner, I will generate a PayID QR code linked to my bank account, displayed in-store, so customers can scan and pay instantly. The following are the steps:

- Generate a unique PayID QR code from the banking app.
- The QR code contains PayID details linked to the bank account.
- Merchants can download, print, and display.
- Customers scanning the QR code auto-fill the merchant's payment details.
- Merchants receive instant notifications of completed payments.

Testing Strategies (User Acceptance Testing - UAT):

Customers will test the PayID scanner feature in the real world. This will entail scanning many QR codes with varied PayID details to ensure the software accurately gathers and automatically fills the data. Users will also test the payment process after scanning to confirm that the transaction goes well. UAT will help detect usability issues or defects before the feature is formally deployed, ensuring it meets user needs and expectations.

Goals

- Develop a PayID scanner feature for the bank's mobile app.
- Enable users to scan QR codes to retrieve PayID information for ease of use.
- Streamline the payment process and reduce manual entry mistakes.

Deliverables

- Fully functional PayID scanner feature within the bank mobile app.
- Thorough testing documentation, including UAT results.
- User guides for the new feature.

Timeframe

- Weeks 1-4: Project planning and design.
- Weeks 5-8: Development and internal testing.
- Weeks 9-10: User Acceptance Testing.
- Weeks 11-12: Final adjustments and deployment.

Part B - Process Improvement Proposal for Software Deployment Consistency

Our development team has always struggled with product deliveries. Developers are now working in teams, which restricts collaboration and communication, leading to integration conflicts, software issues, and project delays. To address these issues and streamline our development process, we recommend making the following improvements to change management, version management, system construction, and release management.

Software changes can alter system functionality and alter control needs to step in for the implementation and auditing of the modifications. Moving over to Agile practices promotes high interaction, cooperation and openness for programmers in comparison to past isolated work that significantly minimizes integration disputes (Cockburn & Highsmith, 2001). Utilizing formal change request tracking tools like Jira or Azure DevOps requires us to formally assess any changes and authorize them which results in greater accountability and vagueness. Documentation with the utilization of centralized documentation tools (Confluence, Jira) also provides greater visibility and facilitates better documentation which facilitates independent decisions making and subsequent integration (Ambler & Lines, 2012).

Version control addresses the challenges of independent work and constant integration issues. The use of a robust, centralized version control system such as Git enhances collaborative software development, maintains a history of changes transparent, and resolves conflicts through branching and merging techniques such as GitFlow. Code reviews through pull requests enforce quality assurance, knowledge sharing, and early error detection, leading to more stable and integrated versions of software.

System construction ensures that software pieces are integrated nicely and that builds occur less often. Continuous Integration (CI) employs tools such as Jenkins or GitLab CI/CD to compile, test, and verify code automatically, finding and fixing integration bugs early in the development process. Automated testing methods, including unit tests, integration tests, and static code analysis tools such as SonarQube, provide real-time feedback on the quality of code, substantially reducing integration challenges and enhancing programme stability (Shahin, 2017). Normalizing development environments using container technologies such as Docker also ensures uniformity across development, test, and production environments, further reducing compatibility challenges (Bernstein, 2014).

Improved release management processes ensure reliability and pre-emptible software deployments. By automating deployment, Continuous distribution (CD) reduces manual error and time lag by ensuring timely and dependable distribution of software updates with less human intervention. Structured release planning and scheduling reduce the risk of problematic deployments, offering predictability and adequate time for rigorous testing and debugging. In addition, clearly established rollback procedures and monitoring tools such as Prometheus or Datadog provide real-time visibility into application performance, allowing for the immediate identification and fixing of deployment issues and overall system resilience.

Adoption of these recommendations is expected to heighten developer cooperation, reduce integration and deployment disputes, and increase the reliability of the software. These methods, backed by industry research, guarantee a unified, effective, and reliable software development process.

REFERENCES

Pillai, B. (2023, November). Understanding Real-time Payment Dynamics in Australia. In *International Symposium on Distributed Ledger Technology* (pp. 124-145). Singapore: Springer Nature Singapore.

Shahin, M., Babar, M. A., & Zhu, L. (2017). Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. *IEEE access*, *5*, 3909-3943.

Cockburn, A., & Highsmith, J. (2001). Agile software development, the people factor. *Computer*, *34*(11), 131-133.

Bernstein, D. (2014). Containers and cloud: From lxc to docker to kubernetes. *IEEE cloud computing*, *1*(3), 81-84.

Ambler, S. W., & Lines, M. (2012). *Disciplined agile delivery: A practitioner's guide to agile software delivery in the enterprise*. IBM press.

Urmila_A1_ProjectProposal

ORIGINALITY REPORT

12%	8%	9%	10%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	lutpub.lut.fi Internet Source	3%
2	Submitted to Flinders University Student Paper	2%
3	kau.diva-portal.org Internet Source	2%
4	rria.ici.ro Internet Source	2%
5	jyx.jyu.fi Internet Source	1%
6	fastercapital.com Internet Source	1%
7	Igwe, Hephzibah. "The Significance of Automating the Integration of Security and Infrastructure as Code in Software Development Life Cycle", Purdue University, 2024 Publication	1%
8	Submitted to Southern Cross University Student Paper	<1%

Exclude quotes	Off	Exclude assignment template	On
Exclude bibliography	Off	Exclude matches	Off