Fatima Mahnoor, Vilka Sergei, Vu Trung

Luku Library - Book reservation System

Software Engineering Project

Project Development Process

Metropolia University of Applied Sciences
Information and communication technologies
Software Engineering
Software project
07.05.2025

1. Introduction

This document covers the purpose, chosen methodologies, development process, and final outcomes of the “Luku Library Management System” software project, developed within the SP1 and SP2 courses in the Software Development program. The aim of the project was to design and develop a user-centric library book reservation platform that enhances accessibility, usability, and efficiency in managing academic library resources.

The aim of this document is to provide a detailed overview of the Luku project, covering its planning, development, and evaluation stages. It examines the full project lifecycle—from initial vision and design to implementation, testing, and final review—while highlighting the use of Agile methodology and DevOps practices. In addition, it reflects on the team’s collaborative workflow, the technologies and tools used, challenges faced along the way, and the key insights gained throughout the development process.

2. Applied Methodology

During the SP1 and SP2 courses, the Luku project was developed using modern software development practices to enhance both technical competence and teamwork. The project followed a practical, real-world approach by using iterative development, task management tools, version control systems, and code quality practices. This section describes the main methodologies and tools used throughout the project lifecycle.

2.1 Agile Methodology

The project was carried out using Agile principles to enable flexibility, regular feedback, and incremental delivery of working software. Agile practices were central to maintaining clear communication, quick adaptation to feedback, and structured teamwork despite a reduced team size. Iterative development cycles helped the team stay aligned with evolving technical goals.

2.1.1 Scrum

The Scrum framework was adopted, with the team working in structured sprints across multiple development phases. Each sprint lasted two weeks, during which new features were planned, implemented, and reviewed. The team operated with a rotating Scrum Master role - ensuring that each member had a chance to manage the sprint, lead meetings, and help to solve problems. Sprint Planning, Daily Stand-ups, Sprint Reviews, and Retrospectives were conducted regularly using tools such as Trello (initially) and Jira (from Sprint 5 onward). The teacher, Amir Dirin, acted as the Product Owner, offering feedback, setting priorities, and validating sprint outcomes during review sessions.

2.1 DevOps Practices

DevOps practices were integrated alongside the Agile approach to enhance automation, simplify deployment processes, and maintain code quality throughout the project. These practices also contributed to smoother team collaboration, faster identification of issues, and consistent development setups.

2.2.1 Continuous integration (CI)

Continuous integration was implemented using Jenkins, which automated the process of building and testing the application. Jenkins was configured to fetch the latest code from the GitHub repository and execute builds using Maven, helping the team identify integration issues early and maintain a stable codebase across development sprints.

2.2.2 Automated Testing and Code Quality

Automated testing was an integral part of the CI pipeline. The team used JUnit for writing unit tests and Mockito for mocking dependencies. Testcontainers were employed for integration testing in containerized environments. JaCoCo was used to measure test coverage, while SonarQube was integrated to analyse code quality, maintainability, and adherence to coding standards. Static code analysis with CheckStyle further ensured consistency and compliance with Java best practices.

2.2.3 Containerisation

The project utilised Docker to create consistent runtime environments for both the frontend and backend components. Docker containers simplified the setup and deployment process, making it easier to test and run the application across different systems. This also allowed the team to simulate production-like environments during development and testing phases.

3. Planning and Risk Analysis

3.1 Risk Identification and Planning

The planning phase of the project began with defining the product vision and identifying user stories to guide development goals, prioritise functionality, and set the scope for the upcoming sprints. During this stage, the team also assessed potential risks that could impact the progress or quality of the system. The following subsections summarise key risks identified early in the project and the strategies used to manage them.

3.1.1 Team Capacity and Coordination

One of the most significant challenges was working with a smaller team than initially planned. The project was completed by three members instead of four, which increased the individual workload and demanded efficient task allocation and time management. In addition, coordinating tasks and responsibilities among team members required clear communication and consistent use of collaboration tools such as Trello, Jira, and WhatsApp.

This risk was mitigated by holding regular Scrum meetings, rotating the Scrum Master role, and ensuring visibility of progress through a shared workspace.

3.1.2 Technical Risks

Although the project remained within the scope of technologies introduced in the course, several tools—such as Docker, Jenkins, SonarQube, and Testcontainers—were new to the team. This lack of prior experience presented a risk of configuration errors, delays due to learning curves, and potential integration issues. To manage this risk, the team conducted independent research, studied documentation and tutorials, and dedicated early sprint time to understanding and testing these tools.

3.1.3 Time Constraints

The project was carried out over a sixteen-week period, with several development stages and deliverables following each other closely. Because of the project's size and the smaller team, there was a risk that important features might not be delivered on time or that the quality could suffer. To manage this, the team focused on completing the most essential features first, used a clear sprint backlog for planning, and stayed flexible to adjust the project scope when needed.

4. Agile Implementation Approach

The development of the Luku project followed the Scrum framework, with a total of eight sprints, each lasting two weeks. Sprints 1 to 4 were completed during the SP1 course period, while Sprints 5 to 8 took place during SP2. Scrum practices were applied throughout to support iterative development, team coordination, and continuous improvement.

- **Sprints**: Structured two-week development cycles, each with specific goals and deliverables.
- **Daily Stand-ups**: Held regularly, both in-person and via WhatsApp or Zoom, to ensure visibility and resolve any blockers.
- **Sprint Reviews and Retrospectives**: Conducted at the end of each sprint to present completed work to the Product Owner, receive feedback, and reflect on team performance.

4.1 Sprint 1 - Project Planning and Initial Design

The first sprint focused on laying the foundation for the project. The team defined the product vision, created the project plan, and set up the Trello board. Initial deliverables included the Product Vision, Project Plan, and a user interface blueprint designed in Figma. The team also created the first version of the ER model and database schema.

4.2 Sprint 2 - Core Features and Prototype Development

During Sprint 2, the team began developing the first working version of the system. Key functionalities such as user registration, login, book search, reservation, and notifications were implemented. A connection to the MariaDB database was established, and unit testing with JUnit, Mockito, and Testcontainers was introduced. The backend was linked with JavaFX frontend components using a basic MVC structure.

4.3 Sprint 3 - Integration and UI Expansion

Sprint 3 focused on extending system functionality and improving the user interface. The frontend was expanded with multiple JavaFX views including pages for search, profile management, and reservations. Controllers were refactored and split for better maintainability. Docker was introduced to prepare for containerised deployment, and a Jenkins pipeline was set up for continuous integration. Code coverage and quality were further improved using JaCoCo and SonarQube.

4.4 Sprint 4 - SP1 Completion and Project Refinement

By the end of Sprint 4, all core features defined in the original backlog were implemented and tested. The backend and frontend were integrated, and the system was capable of handling user interactions, book reservations, and real-time availability updates. A project presentation was prepared, and documentation such as the User Manual and Developer Guide were drafted.

4.5 Sprint 5 - Localisation and UI Enhancements

The SP2 phase began with the addition of multilingual support. The application was localised into English, Urdu, and Russian, and the database was updated to store multilingual content. The header component was refactored into a reusable file, improving the structure of the frontend code. Project tracking was moved from Trello to Jira.

4.6 Sprint 6 - Static Analysis and Documentation

This sprint focused on improving code quality and preparing the project for final review. The team used CheckStyle to fix formatting and structural issues, resolving 98% of identified problems. SonarQube analysis confirmed high ratings in security, reliability, and maintainability. The team also created a Code Review Report and updated the project's README with diagrams and setup instructions.

4.7 Sprint 7 - User Acceptance Testing and Bug Fixing

Sprint 7 was dedicated to User Acceptance Testing (UAT) and heuristic evaluation. These activities aimed to assess the system's usability, identify potential issues, and gather user feedback to support final adjustments. A written summary of test results was also prepared.

4.8 Sprint 8 - Final Delivery and Presentation

The final sprint focused on wrapping up the project. The team completed the remaining documentation and delivered the final project presentation. All deliverables were submitted, including technical documentation, testing materials, and user-oriented documentation.

5. Scrum Team Roles and Responsibilities

The Luku Library Management System was developed by a three-member team: Sergei Vilka, Mahnoor Fatima, and Trung Vu. Due to the reduced team size, all members contributed to various aspects of the project, including design, development, testing, and documentation. Responsibilities were distributed flexibly based on individual strengths and sprint requirements.

To align with Scrum methodology, the role of Scrum Master was rotated among the team members during the project. Each member took on this role at least once, gaining experience in facilitating sprint meetings, removing impediments, and maintaining team coordination. The Product Owner role was fulfilled by the course instructor, Amir Dirin, who provided feedback, prioritised features, and validated sprint outcomes during review sessions.

The team collaborated through regular Scrum ceremonies, including Sprint Planning, Daily Stand-ups (held both in-person and online), Sprint Reviews, and Retrospectives. Communication and task management were handled using Trello during the early sprints and Jira from Sprint 5 onward.

6. Summary

The Luku Library Management System project successfully achieved its goal of delivering a functional, user-friendly library reservation system tailored for academic environments. Over the course of eight sprints, the team implemented a wide range of features, including user authentication, real-time book search and reservation, multilingual support, and automated notifications. The project also incorporated modern development tools and practices, such as Docker containerisation, continuous integration with Jenkins, and code quality analysis using SonarQube and CheckStyle.

Through the use of Agile and DevOps methodologies, the team was able to adapt to challenges, maintain a steady development pace, and deliver a complete system with supporting documentation and test coverage. Each sprint contributed to the system’s progress while offering opportunities for learning and process improvement.

6.1 Areas for Improvement

While the project achieved its core objectives, several areas for improvement were identified through heuristic evaluation and user testing:

- **Session Timeout Feedback**: One of the most critical usability issues was the lack of clear feedback when a session expired. Users remained on a non-functional page with no explanation. Introducing a popup message informing users of automatic logout, with options to log in again, would greatly improve usability.

- **Simultaneous Reservations**: Users attempting to reserve the same book at the same time may encounter errors that could result in double-bookings. Implementing a locking mechanism or transactional check on book availability would help prevent this issue and improve system reliability.

- **Navigation and Exit Options**: Several screens lacked clearly marked “Back” or “Cancel” buttons, making navigation cumbersome. Users who accidentally initiated an action, such as reserving or returning a book, were unable to easily reverse it. Adding consistent navigation elements and confirmation dialogs would help prevent errors and improve user control.

These usability findings present opportunities for further improvement in future iterations and demonstrate the value of user feedback in refining system quality.

Appendices

- **GitHub Repository**: [GitHub link]

- **Figma Mockups**: [Figma link]

- **Trello (Sprint 1-4):** [Trello link]

- **Jira (Sprint 5-8):** [Jira link]