Monami Kirjavainen,

Riku Toivanen,

Victoria Vavulina

LibraryHub - Library application

Metropolia University of Applied Sciences

Software Engineering Project

12.12.2025

1. Introduction

This document covers the methodologies, development technologies, and final outcomes of the LibraryHub software project, developed during OTP1 and OTP2 software engineering courses. The aim of the LibraryHub project was to design and develop a Library application for students.

The planning, development, and assessment of the finished product are all covered in detail in this report.

2. Methodology

LibraryHub was developed using common best practices to support product iteration and completion. The project used an iterative way of working, with a focus on clean code and proper version control. This section outlines the key methodologies applied.

2.1 Agile Methodology

The development of LibraryHub followed Agile principles, supporting flexible iteration, and delivering working parts of the software step by step. Agile also played an important role in ensuring clear communication and strong teamwork.

2.2 Scrum

Scrum was applied throughout the project, giving each sprint a clear and structured workflow. Each sprint lasted two weeks, and the development was divided into several stages. During each sprint, the team planned, developed, and reviewed new features. The Scrum Master role rotated regularly within the team. Every team member had the opportunity to create a sprint plan and sprint review, while the current Scrum Master was responsible for assigning tasks to the team members. Stand-ups were held regularly, and the team used Jira for task tracking and Discord for daily communication.

2.2 DevOps Practices

By implementing DevOps, we maintained better code quality, improved collaboration, and ensured consistent development environments.

2.2.1 Continuous integration (CI)

CI was set up using GitHub Actions. It automatically tests the code and checks coverage and quality. The pipeline pulls the newest code, runs all tests, and then reports the results, giving the team quick updates about any problems.

2.2.2 Automated Testing and Code Quality

For testing, we used Jest to write unit tests. Jest also created the coverage reports. SonarQube checked the code quality and maintainability, while ESLint handled the static code analysis.

2.2.3 Containerization

For containerization, we used Docker to create consistent environments for the application. Docker containers made setup easier in new environments and allowed the app to be tested and run smoothly on different systems.

3. Planning and Risk Analysis

Effective planning and early risk identification were essential to ensure the successful delivery of the project. The planning phase began with establishing a clear product vision and understanding the needs and expectations of potential users. From this, the team developed user stories, project goals, key functionality requirements, and the overall scope of the system. Alongside defining these elements, the team evaluated the technological landscape and identified potential risks that could affect project progress or quality. These risks were then assessed and managed through mitigation strategies described below.

3.1 Risk Identification and Planning

The project team systematically identified risks in areas such as communication, technology, and scheduling. Each risk was evaluated based on its likelihood and potential impact, followed by the development of mitigation plans to reduce the chance of negative outcomes.

3.1.1 Communication

Since the team consisted of members working together in a new environment, communication was a critical focus. The risk was that misunderstandings or a lack of coordination could slow progress or cause inconsistencies in work output. To address this, the team implemented regular communication practices including scheduled meetings and daily use of external messaging platforms. These measures ensured that responsibilities were clearly understood and that challenges could be quickly discussed and resolved.

3.1.2 Technical Risks

The project remained intentionally scoped within the team's abilities, but technical risks still emerged due to varying levels of experience with tools such as Docker, SonarQube, and other development technologies. Some team members were able to troubleshoot and solve problems more quickly than others, which could have led to uneven progress. This was solved with strategic task management and distribution to plan according to the strengths of each member.

3.1.3 Time Constraints

Although the project scope was designed to match the team's size and skill level, scheduling posed a significant challenge. Members had differing commitments outside of school, which made joint work sessions and meetings difficult to schedule. As a result, much of the work had to be completed independently. To manage this, the team planned tasks according to individual availability and divided responsibilities in a way that allowed each member to progress autonomously while still contributing to shared goals. Regular updates helped ensure alignment despite limited overlapping time.

4. Agile Implementation Approach

LibraryHub was developed in eight sprints. Sprints one to four were completed during OTP1 and five to eight during OTP2.

1. Sprints: Structured two-week development cycles, using user stories to implement features.
2. Meetings: Meetings were held regularly two times during each sprint to assign roles and do a review on the final days of the sprint via Discord.
3. Sprint planning and reviews: During the start of a sprint a sprint plan was to be made, that clearly describes the tasks to be completed during the sprint. During the end of each sprint a review was also made describing tasks done and if the sprint completed all planned tasks.

4.1 Sprint 1 - Project Planning and Initial Design

Sprint 1 focused on establishing a strong foundation for the project through careful planning and initial design activities. The team began by defining the product through a product vision document, which clarified the overall goals, scope, and direction of the application. This shared vision ensured that all team members had a clear understanding of the intended outcome.

Once the goal was established, the team created user stories on the Jira board to organize development tasks. The primary deliverables for this sprint included the finalized product vision, a comprehensive project plan, and an initial prototype design created in Figma. In parallel, the team selected the technology stack to be used throughout the project and began modeling the database structure to support the application’s core functionality. These activities collectively set the groundwork for subsequent sprints and development efforts.

4.2 Sprint 2 - Core Features and Prototype Development

During the second sprint we created a front-end for the project, that held all the core features that were also created during the sprint. The team also created the database and deployed it to the system.

4.3 Sprint 3 - Integration and UI Expansion

The third sprint was spent adding new features to the system, expanding the core functionalities. This sprint was also used to integrate a CI/CD pipeline to the project and developing unit tests for the features made in sprints up to this point. The Docker image was also developed and tested during sprint 3.

4.4 Sprint 4 - OTP1 Completion and Project Refinement

During the fourth sprint test coverage was improved to better enhance the reliability of the project. This sprint finished our backlog of undone tests for former features, so the entire project was now mostly under unit tests. The team also prepared for the first project presentation that was conducted at the end of the sprint.

4.5 Sprint 5 - Localization and UI Enhancements

During Sprint 5, we focused on removing all hard-coded text from the application and implementing a full localization system. The UI was internationalized using the **i18next** framework, with **react-i18next** providing seamless integration for React components. For routing support, we adopted **next-i18n-router**, and localization resources were managed through **i18next-resources-to-backend**.

As part of this effort, translations were added for three languages—**English, Japanese, and Russian**—along with supporting localization files to ensure consistent multilingual functionality across the platform.

4.6 Sprint 6 - Statistical code review and Documentation

In Sprint 6, we focused on improving code quality, extending localization, and preparing for acceptance testing. A default language was configured at the database level, enabling the application to automatically switch to the user's preferred language after login. Code quality was strengthened through SonarCloud/SonarScanner, with all fixable issues resolved and around one hundred problems eliminated, resulting in improved maintainability and readability. Documentation was expanded with new system design artifacts, including the Use Case, ER, and Activity diagrams. Finally, we prepared for

acceptance testing by planning the UAT alpha phase to ensure the product meets expected standards.

4.7 Sprint 7 - User Acceptance Testing and Heuristic Evaluation

In Sprint 7, our focus was on validating the system through acceptance testing, refining usability, and finalising documentation. We carried out **performance testing** using Google Lighthouse, successfully improving the application's performance score from 69 to 97 and accessibility from 90 to 96. Alongside this, a **heuristic evaluation** was conducted to assess usability, with findings documented and recommendations provided for further improvement.

We also executed the **acceptance tests** defined in the previous sprint, producing a report of the outcomes to confirm that the system meets the expected requirements. On the documentation side, we expanded the architectural design materials by creating additional UML models, including **sequence diagrams** and **deployment diagrams**, ensuring clarity and maintainability for future development.

4.8 Sprint 8 - Final Delivery and Presentation

In the final sprint, our team concentrated on completing acceptance testing, refining evaluations, and preparing the project for delivery. We conducted the **UAT alpha** as planned, ensuring that all members participated after the earlier misstep of running a beta test in Sprint 7. A new version of the **heuristic evaluation report** was produced, incorporating an Excel table to consolidate group-wide feedback and usability findings.

We also developed a **static code review report** to document code quality, updated the **README file** for clarity, and created both the **project process report** and the **final product report**. To conclude, we prepared the **final presentation**, creating slides and recording a video submission to present the completed system.

5. Scrum Team Roles and Responsibilities

LibraryHub was initially developed by 4 members that later turned to 3 during the middle of development. The initial members were Riku Kaartoaho (left), Monami Kirjavainen, Riku Toivanen and Victoria Vavulina. Though there was a difference in scheduling between the members, all members contributed equally to the project and had equally distributed responsibilities. The group used the scrum process and rotated the role of scrum master, so every member had the responsibility at least once during OTP1 and OTP2.

Links:

GitHub Link (https://github.com/vickneee/SEP-R1)

Jira Link (https://rikukaartoaho-1755679798806.atlassian.net/jira/software/projects/SCRUM/boards/1/backlog)

6. Summary

LibraryHub was a successful project that achieved its goal of becoming a web-based library application that could handle the basic services a library provides. The project uses modern technology trying to achieve current development standards. Using agile, scrum and devops methologies, the project could maintain a steady development schedule.