Elias Norta, Nikita Rybakov, Taif Jalo, Miska Voutilainen

Timetable Management System

Metropolia Ammattikorkeakoulu

Ohjelmistotuotantoprojekti 1 TX00EY27-3009

Project Plan

26/1/2026

Table of Contents

1. Project Overview

1.1. Project Title

Timetable Management System

1.2. Problem Summary

Students and teachers currently struggle to access accurate and up-to-date class schedules because timetables are scattered across multiple platforms, updated manually, and often communicated late.

This leads to confusion, missed classes, scheduling conflicts, and unnecessary administrative work.

There is no centralized, real-time system where users can easily view changes, track upcoming classes, or receive timely notifications.

1.3. Intended Audience/Users

Students: Need quick access to class information and dynamic updates.

Teachers: Require tools for schedule management, class coordination, and communication.

1.4. Main Features/Components

1.4.1 Features

- Create, view, and modify timetables.

- Alerts for upcoming classes and assignments.
- Option to view schedules for different days and weeks.

1.4.2 Components

- N/A

2. Product Objectives

- Increase user adoption rates by 30% within half a year.
- Improve user satisfaction scores to 90 or above.
- Generate 50,000 € in revenue within a year.

3. Scope and Deliverables

- Included:
  - Timetable/calendar
    - View by day, week
  - Chat system
  - Notification system
  - Creation, modification, and deletion of sessions (admin management)
  - Authentication:
    - Login
    - Signup
- Not included
  - Mobile app
  - Payment Systems
  - 2FA
  - Calendar sync

4. Project Timeline (Trello)

For project planning, we use Scrum. The sprint schedule is based on the Metropolia course. Each sprint has a duration of two weeks.

- Sprint 1 Planning & Setup (15.01-29.01):

        - Project Plan & Vision
        - Trello Workspace setup
        - Use Case Diagram
        - Initial Documentions
    - Sprint 2 Core Features & Design (29.01-05.02):
        - Unit testing
        - Maven setup
        - Generating code coverage reports
        - Clean code practices implementation
    - Sprint 3 Jenkins & Docker (12.02-26.02.):
        - Jenkins automation, testing, and deployments
        - Introduction to Docker
        - Creating & deploying Docker images
    - Sprint 4 Docker & Kubernetes (05.03-13.03.):
        - Running images using Kubernetes
        - SecDevOps
        - Creating MVP
    - Presentation:
        - Share the minimum valuable product (MVP) with classmates

Milestones:

- Calendar view
- The integration of a lesson system (adding, editing, and deleting lessons)
- MVP
- Chat

Task tracking and milestones are managed in Trello.

5. Resource Allocation

5.1. Team Members and Roles

5.1.1 Team Members

Taif Jalo, Elias Norta, Miska Voutilainen, Nikita Rybakov

5.1.2 Roles

Taif Jalo – Sprint 1 Scrum Master, Developer

Elias Norta – Sprint 2 Scrum Master, Developer

Miska Voutilainen – Sprint 3 Scrum Master, Developer

Nikita Rybakov – Sprint 4 Scrum Master, Developer

5.2. Software, Hardware, and Tools

5.2.1 Softwares: Visual Studio Code, Intellij Idea

5.2.2 Hardware: Laptop(s)

5.2.3 Tools: HeidiSQL, JavaFX, Artificial Intelligence

5.3. External Resources or Support

5.3.1 External Resources: Azure, Jenkins, Docker, Kubernetes, CalendarFX

5.3.2 Support: Amir Dirin

6. Risk Management

6.1. Description

The project may face issues related to scheduling, technology, teamwork, and unclear requirements. These risks could affect progress, functionality, or deadlines.

6.2. Likelihood

- Technical issues (database errors, setup problems): Medium
- Task delays: Medium
- Miscommunication: Medium
- Bugs in core-features: Medium-High
- Scope expanding too much: Low-Medium

6.3. Impact

- Technical problems: Can slow development.
- Delays: May affect sprint goals.
- Miscommunication: Leads to rework.
- Bugs: Can break timetable functionality.
- Scope creep: Can take time away from essential features.

6.4. Mitigation Strategies

- Keep tasks small and clear in Trello.
- Communicate regularly to avoid misunderstandings.
- Test features as they are developed.
- Document technical setups so everyone can follow them.
- Stick to agreed-upon scope and avoid adding new features mid-sprint.

7. Testing and Quality Assurance