KahoOST

SE Project – Dokumentation
Frühlingssemester FS 2026

SE PROJECT

Version: 1.0.0
Datum: 20-02-2026
Git Version:

Team: Gioele Petrillo (gioele.petrillo@ost.ch)
Ethann Baumgartner (ethan.baumgartner@ost.ch)
Jasmin Fässler (jasmin.faessler@ost.ch)
Andrin Klarer (andrin.klarer@ost.ch)
András Tarlós (andras.tarlos@ost.ch)
Betreuer: Abinas Kuganathan (abinas.kuganathan@ost.ch)

Informatik
Ostschweizer Fachhochschule

Inhaltsverzeichnis

Management Summary

Even though this is the first chapter of your document, it is typically the last one filled with content. The **Management Summary** is a **brief and high-level summary** of your project. It should give any reader unfamiliar to the project an overview of the contents included later in the document.

A common structure is:

- What is the problem we wanted to solve?
- How did we solve the problem?
- Does your solution solve the problem in a successful way?
- Will there be consecutive projects based on your work?

Diagrams and images work very well in this chapter, especially screenshots of your software.

One final remark: a well written management summary is a good starting point for your {Project Presentation}, as you will address a similar audience there.

Produktdokumentation

Requirements

Describe the functional and non-functional requirements as covered in the SEP1 module.

Functional Requirements

The functional requirements should contain at least:
- The actors of the system under development (SUD).
- The goals of each actor.
- A use case digram that provides a good overview of the actors, use cases, and the relationships between them.
- Use case descriptions in the brief, casual, or fully-dressed format (depending on the level of detail required at the current phase of the project) for the SUD.
- Tip: Use identifiers such as `UC1', UC2′`, ldots as references for your use cases.

Non-Functional Requirements

- Choose an appropriate form (bare minimum, ISO/IEC 25010:2011, FURPS, ldots) for your non-functional requirements (NFRs) as covered in SEP1.
- Try to concentrate on the NFRs that are most relevant or special to the SUD.
- Make sure that your NFRs are verifiable (acceptance criteria).
- Document how and when you plan to verify your stated NFRs.
- TIP: Use identifiers such as `NFR1', NFR2′`, ldots as references for your NFRs.

Domain Analysis

Describe the problem domain using a domain model as covered in the SEP1 module. The domain model should contain at least:
- A domain model diagram that describes the problem domain at an appropriate level of abstraction.
- A description of aspects of the problem domain that are important, non-standard, or are difficult to understand form the domain model alone.

Architecture

Describe the architecture of your software as covered in the SEP2 module. The main goal of this chapter is describing the textbf{technical implementation} in a way that a new team member can start working on the product as fast as possible.
- Use an existing template as a starting point (texttt{arc42}, texttt{C4 model}, ...)
- Focus on stable, high-level concepts rather than details
- Cover different views (static, dynamic, deployment, ...)
- Prefer diagrams over text (ideally UML)
- Explain the reasons behind your actions: textit{Why did we build it like this?}

Quality Measures

Working Environment

Describe the quality measures applied in your project as covered in the SEP1 and SEP2 modules. Things that might be included in this chapter:

- Organizational means like **Merge Requests**, **Definition of Done**, etc.
- Guidelines for code, documentation, version control, etc.
- Tools used to assess the quality of your product (linter, metrics, ...)
- Tools used to build and deploy your product (CI/CD)

Try to avoid duplication with other chapters such as the **Project Plan**. Work with cross-references when appropriate.

Testing

Describe the **Test Strategy** used for testing your product as discussed in the SEP1 module. How each functional and non-functional requirement is verified, at what level (Unit, Integration, System), etc.

Projektdokumentation

Project Proposal

Projektname:
KahoOST

Teammitglieder
Gioele Petrillo (gioele.petrillo@ost.ch)

Ethan Baumgartner (ethan.baumgartner@ost.ch)

Jasmin Fässler (jasmin.faessler@ost.ch)

Andrin Klarer (andrin.klarer@ost.ch)

András Tarlós (andras.tarlos@ost.ch)

Verfügbarkeiten

Time slot	Mon	Tue	Wed	Thu	Fri
08h00-09h00	-	-	-	-	XR
09h00-10h00	-	-	-	-	XR
10h00-11h00	-	-	-	-	XR
11h00-12h00	-	-	-	-	XR
12h00-13h00	XR	-	-	XR	XR
13h00-14h00	-	-	-	-	XR
14h00-15h00	-	-	-	-	XR
15h00-16h00	XR	-	-	-	XR
16h00-17h00	XR	-	-	-	XR
17h00-18h00	-	-	-	-	XR
18h00-19h00	-	-	-	-	XR

Vision
Aktuelle Quiz-Tools wie Kahoot, Slido oder Particify sind für den Einsatz an der OST unflexibel und unpersönlich. Wir wollen eine ähnliche Webapplikation erstellen, womit Dozenten ein Quiz mit Fragen erstellen können (diesen Bereich nennen wir Admin-Konsole). Dieses Quiz kann dann in der Vorlesung durchgeführt werden in Echtzeit mit allen Studenten und am Schluss wird eine Rangliste angezeigt.

Zusätzlich bietet Kahoost diese wichtigen Funktionalitäten:

- das exportieren als Anki-Kartenset und CSV
- eine detailiertere (z.B. schnellstantwortente Person wird hervorgehoben) und visuell ansprechendere Rangliste
- Es wird eine Zusammenfassung/Heatmap angezeigt, damit der Dozierende sieht, bei welchen Fragen Nachholbedarf besteht
- Responsive Layout (Studenten können Quiz am Handy oder Laptop lösen)

Mögliche Erweiterungen

- PDF Export
- PvP-Modus unter Studenten von existierenden Vorlesungskarten
- Karten generieren mit "KI"
- Zugriff auf Moodle
- Markdown
- Quality of Life

Proposed Realisation

Für die Umsetzung setzen wir auf Next.js als Full-Stack-Framework, da es Frontend und Backend in einer Codebase vereint und damit die Entwicklung beschleunigt. Die Echtzeit-Kommunikation wird über WebSockets realisiert, während PostgreSQL in Kombination mit dem Drizzle ORM für eine strukturierte und performante Datenhaltung sorgt. Die Zusammenfassung wird beispielsweise mit Chart.js umgesetzt, um die Quiz-Ergebnisse dynamisch und anschaulich darzustellen. Für die Authentication nutzen wir NextAuth.js, wo wir nur OST-E-Mail Adressen erlauben. Das Styling übernimmt Tailwind CSS, um ein modernes, responsives UI mit minimalem Aufwand zu gewährleisten. Durch die Verwendung dieser Technologien ist Kahoost plattformunabhängig einsetzbar und lässt sich später leicht um zusätzliche Features erweitern.

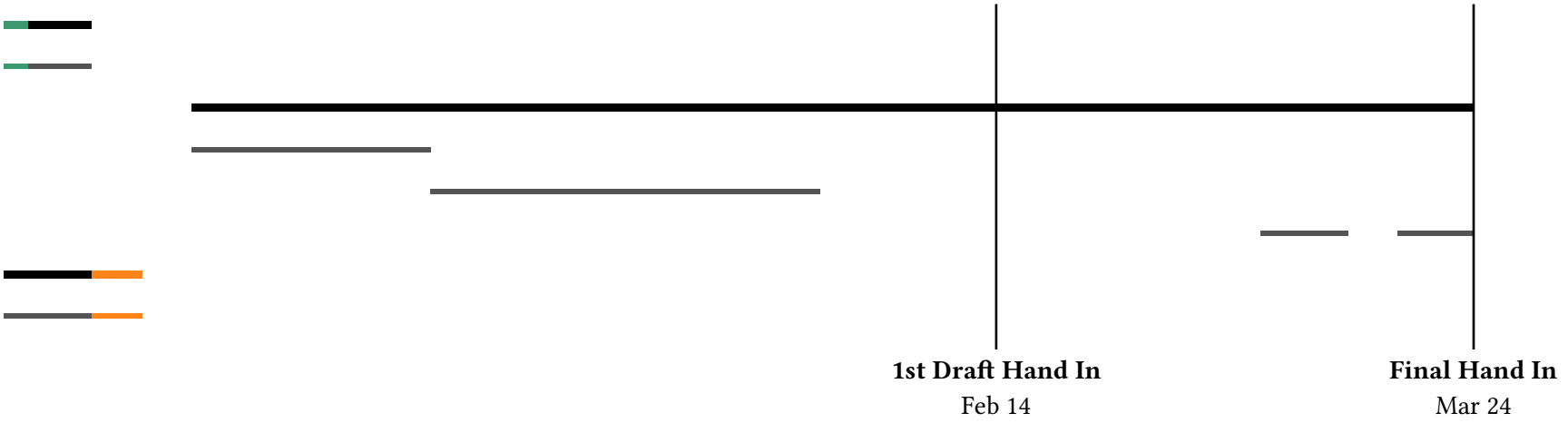
Projektplan

Scope – Estimated Features

Aktuelle Quiz-Tools wie Kahoot, Slido oder Particify sind für den Einsatz an der OST unflexibel und unpers¨onlich. Wir wollen eine ¨ahnliche Webapplikation erstellen, womit Dozenten ein Quiz mit Fragen erstellen k¨onnen (diesen Bereich nennen wir AdminKonsole). Dieses Quiz kann dann in der Vorlesung durchgef¨uhrt werden in Echtzeit mit allen Studenten und am Schluss wird eine Rangliste angezeigt. Zus¨atzlich bietet Kahoost diese wichtigen Funktionalit¨aten: • das exportieren als Anki-Kartenset und CSV • eine detailiertere (z.B. schnellstantwortente Person wird hervorgehoben) und visuell ansprechendere Rangliste • Es wird eine Zusammenfassung/Heatmap angezeigt, damit der Dozierende sieht, bei welchen Fragen Nachholbedarf besteht • Responsive Layout (Studenten k¨onnen Quiz am Handy oder Laptop l¨osen) Mögliche Erweiterungen • PDF Export • PvP-Modus unter Studenten von existierenden Vorlesungskarten • Karten generieren mit ”KI” • Zugriff auf Moodle • Markdown • Quality of Life

Ressourcen

Zeitplan



Organisation und Rollen

Risk Management

Guidelines

In diesem Abschnitt gehen wir in die Details der Guidelines ein, die es uns als Team ermöglichen, einheitlich, zielorientiert und sturkturiert an diesem Projekt zu arbeiten.

Dokumentation

Code

Versionskontrolle
Als Versionskontrollsystem verwenden wir Git; GitHub dient uns, um unser Projekt-Repository zu hosten. Im Grunde existieren zwei Basis-Branches im Projekt: `master` und `dev`.

Beim Entwickeln und Testen von neu geschriebener Software müssen neue Branches erstellt werden, wobei die folgenden Namingstandards beachtet werden sollen:

`<feature|bugfix|documentation>/<work-item-id>-<titel-des-work-items>`

Beispiele:

- feature/12-ui-der-loginpage-ueberarbeitet
- bugfix/32-der-knopf-soll-rot-anstatt-blau-sein
- documentation/titel-mit-typo-korrigiert

Nachdem ein neuer Branch erstellt wurde und das gewünschte Update erfolgreich implementiert ist, muss er in den `dev` Branch gemergt werden; in den `master`-Branch dürfen Änderungen direkt nicht gemergt oder gepusht werden. Der `master`-Branch ist dementsprechend auch geschützt, sodass Entwickler nur mit einer Pull-Request von `dev` nach `master` mergen können.

Der Workflow sieht im Endeffekt so aus: `feature -> dev -> master`.

Time Tracking

Für Time-Tracking wird die Webseite **Clockify** verwendet. Sie ermöglicht es uns, unsere Arbeitszeiten zu erfassen, sodass der Fortschritt des Projekts übersichtlich nachvollzogen werden kann. Clockify ist auch mit zahlreichen Features ausgerüstet, die mit Hilfe von Diagrammen anzeigt, wer wie viel an einem Feature gearbeitet hat. Diese Visualisierungen können auch dem Stakeholder gezeigt werden, falls nötig.

Issue & Project Tracking Software

Um Work-Items erstellen zu können wird **Jira** verwendet. Jira erleichtert uns, den Überblick darüber zu behalten, wie viele Features noch implementiert werden müssen und respektive, wie viel Zeit diese in Anspruch nehmen. Klicke den folgenden Link an, um zur Übersicht des Projekts auf Jira zu kommen: KahoOST Jira.

Time Tracking Report

The main goal of this chapter is to show your stakeholders that your project is on track, i.e. you will be able to finish it within the remaining time. The chapter should cover two topics:

- Describe **how you track time** in your project. Try to avoid duplication with the **Project Plan**.
- The **current time tracking statistics** for your project.

For the latter one, keep the information on a high-level:

- How much time did we invest in total yet? How much is remaining?
- How much time did we invest in iteration 1, 2, 3, …?
- How much time did we invest in which topic?

Diagrams work very well to communicate these information. Low-level information (e.g. **{how much time did we spend on task XYZ?**) is best kept within the chosen time tracking tool.

Personal Reports

Before the final submission, **personally reflect** your work in this project:

- What things did go well?
- Which areas could we improve?
- What were your personal highlights?

The information gathered in this chapter will be ***very*** useful for all your future projects.

Meeting-Zeiten

Überblick

Datum	Zeit	Name	Teilnehmende
16.02.2026	16:00-16:45	Informelles Meeting, Projekt Kick-off	Alle
16.02.2026	17:00-17:15	Rollenaufteilung & Organisation	EB,AK,JF,JH,AT

EB=Ethan Baumgartner, **AK**=Andrin Klarer, **JF**=Jasmin Fässler, **JH**=Joris Hänseler, **AT**=András Tarlós, **BAK**=(Betreuer) Abinas Kuganathan

Meeting-Details

- **Datum:**
- **Uhrzeit:**
- **Ort / Plattform:**
- **Leitung:**
- **Protokollführung:**
- **Teilnehmende:**

Agenda

---

1. Agenda-Punkt 1 Agenda-Punkt 2 Agenda-Punkt 3

Besprochene Punkte & Entscheidungen

Thema, **Diskussion / Kernaussagen**, **Entscheidung**, **Verantwortlich**	Thema 1, Notizen zur Diskussion, Beschluss / Ergebnis, Name	Thema 2, Notizen zur Diskussion, Beschluss / Ergebnis, Name	

Bibliografie