
**Technical University of Munich**

School of Computation, Information and Technology

-- Informatics --


Bachelor's Thesis in Computer Science

# Development of a Thesis Management System


Fabian Emilius

# Technical University of Munich

School of Computation, Information and Technology

-- Informatics --

Bachelor's Thesis in Computer Science

## Development of a Thesis Management System

Entwicklung eines Verwaltungssystems für wissenschaftliche Arbeiten


| | |
|---|---|
| **Author:** | Fabian Emilius |
| **Supervisor:** | Prof. Dr. Stephan Krusche |
| **Advisor:** | Prof. Dr. Stephan Krusche |
| **Start Date:** | 12.08.2024 |
| **Submission Date:** | 12.12.2024 |

I confirm that this Bachelor's thesis is my own work and I have documented all sources and material used.

Munich, 12.12.2024 Fabian Emilius

# Acknowledgements

I want to thank everyone who supported me during my thesis. First and foremost, I would like to express my gratitude to my advisor and supervisor, Stephan Krusche. He gave me the opportunity to write this thesis and helped me understand the various processes involved. Despite his busy schedule, and even while on vacation, he patiently answered all my questions, providing guidance and support throughout the entire journey. I would also like to thank Matthias Linhuber, whose insights into the writing process and guidance on structuring the thesis were extremely helpful. Additionally, Robert Jandow provided essential assistance with code reviews and deployment configuration, which were integral to the technical aspects of this thesis. A special thanks to Benedikt Geisberger, Yannik Schmidt, and Ramona Beinstingel for their early code reviews. Their feedback helped to lay a strong foundation for the technical quality of this thesis. Finally, I would like to thank my parents for their support during my studies. Their encouragement and understanding made a real difference throughout this journey.

**Abstract**

The management of theses at large universities is often hindered by inefficient, manual processes that are not scalable for a large number of students. This bachelor's thesis presents the development of a web-based thesis management system aimed at digitalizing and streamlining the entire thesis lifecycle, from finding a student to grading the thesis. The platform is designed to serve multiple stakeholders; advisors, supervisors, and students, allowing for a seamless exchange of information and efficient task management. By automating repetitive tasks and reducing media breaks in the process, the system intends to address the challenges of thesis supervision in large academic institutions, while improving communication and transparency at the same time.

The platform enables advisors to post topics, manage applications, and guide students through the thesis writing process. Students can view available topics, submit applications, and follow a clear workflow from proposal submission to final grading. Automated notifications keep all stakeholders informed at each stage of the thesis lifecycle, improving coordination and reducing administrative burden. The system also consolidates feedback and assessment processes, providing a historical overview that can enhance future interactions and evaluations.

This thesis also discusses the integration of an overview for supervisors, which allows for real-time tracking of thesis progress across multiple students and advisors.

**Zusammenfassung**

Die Betreuung von Abschlussarbeiten an großen Universitäten wird häufig durch ineffiziente, manuelle Prozesse erschwert, die für das hohe Volumen an Studierenden nicht skalierbar sind. Diese Bachelorarbeit stellt die Entwicklung eines webbasierten Systems zur Verwaltung von Abschlussarbeiten vor, das den gesamten Prozess der Arbeit von der Studierendensuche bis zur Benotung digitalisiert und vereinfacht. Die Plattform ist darauf ausgelegt, den Kommunikationsaustausch von Advisor, Supervisor und Studierenden zu verbessern. Durch die Automatisierung wiederkehrender Aufgaben und die Reduzierung von benutzten Anwendungen zielt das System darauf ab, die Herausforderungen der Abschlussarbeitsbetreuung zu bewältigen.

Die Plattform ermöglicht es Betreuern, Themen zu veröffentlichen, Bewerbungen zu verwalten und Studierende durch den Prozess des wissenschaftlichen Arbeitens zu begleiten. Studierende können verfügbare Themen sehen, Bewerbungen abschicken und einem klaren Workflow von der Erstellung des Proposals bis zur endgültigen Bewertung folgen. Automatisierte Benachrichtigungen halten alle Beteiligten über den Fortschritt der Arbeit informiert, was die Koordination verbessert und den administrativen Aufwand reduziert. Das System stellt zudem Feedback- und Bewertungsprozesse bereit und bietet eine Übersicht über alle abgeschlossene Arbeiten.

Diese Arbeit diskutiert auch die Integration einer Übersicht für Betreuer, die einen Echtzeit-Überblick des Fortschritts von Abschlussarbeiten über mehrere Studierende und Betreuende hinweg ermöglicht.

# Contents

# 1 Introduction

The process of finding and securing a thesis topic can be very complex, particularly in large faculties where many students often seek guidance from the same advisors. The digitalization of this process offers a logical progression, facilitating time-saving measures for both advisors and supervisors. Moreover, digitalization introduces a variety of new tools and platforms that allow for the evaluation and improvement of the overall management process [1].

In the context of thesis management, there are three primary stakeholders: the supervisor, typically a professor; the advisor, often a doctoral candidate; and the student, who is responsible for writing the thesis. Each of these stakeholders encounters different challenges throughout the process of thesis selection and finalization. The main focus of this thesis is to simplify the tasks of the advisor and supervisor, but student's perspective is also taken into account where appropriate.

The Chair of Applied Software Engineering[1] has developed a platform called ThesisTrack[2] which was originally created as part of a master's thesis focusing on project-based learning [2]. The platform currently enables students to propose topics, with chair members reviewing these proposals through an administrative interface.

The goal of this thesis is to add functionality to the existing platform with features that make it easier for advisors, supervisors and students to manage the thesis process after they accepted an application, show an overview of active theses and keep a detailed record of past theses.

## 1.1 Problem

In the absence of personal contact with a potential advisor, students typically apply for thesis supervision via email. While this approach may be feasible in smaller faculties, it is not scalable in larger universities where many students lack direct connections with advisors or supervisors [3] and are required to submit

[1] https://ase.cit.tum.de/

[2] https://github.com/ls1intum/thesis-track

applications via email. For example, in October 2023, the Technical University of Munich (TUM) had 14,942 new students[3], many of whom will need to complete a thesis around the same time. Advisors listed on faculty websites may receive multiple applications daily, leading to challenges in processing and reviewing every submission effectively. This results in a significant number of unanswered applications, creating inefficiencies in the thesis allocation process. The current version of ThesisTrack addresses part of this issue by offering an application form, but it requires students to search for available topics on external websites. This disjointed process leads to applications that are misaligned with the chair's research focus, further increasing inefficiencies.

Another challenge is the repetitive nature of onboarding new thesis students. Advisors must invite each student to the chair's services and provide a consistent explanation of the thesis process. This repetition consumes valuable time that could be allocated more effectively.

Supervisors, often responsible for multiple doctoral candidates advising numerous students, also face difficulties in maintaining an overview of all ongoing theses. Each supervisor typically develops an independent system for tracking the progress of their students' work which can lead to inconsistencies and increased administrative burdens.

Additionally, retrieving important information or documents related to completed theses is problematic. Feedback and documents such as proposals are typically exchanged via email or messaging services, making it difficult to locate specific files after the thesis has been completed.

## 1.2 Motivation

The digitalization of university processes is becoming increasingly common. While digitalization is not a universal solution, it plays a key role in preparing students for a job market that increasingly relies on digital skills and infrastructure [4]. Concerns regarding the handling and storage of data have been raised by

[3]https://www.tum.de/en/about-tum/facts-and-figures/tum-in-figures/students

students [5], but in this instance, the university already publishes thesis-related information, mitigating these concerns for the platform. Consolidating the thesis writing and application processes into a single platform enhances efficiency and consistency. This integration reduces the likelihood of data loss and streamlines administrative tasks, allowing advisors and supervisors to allocate more time to accepting additional theses or focusing on research activities.

## 1.3 Objectives

The thesis management system aims to make the advisor and supervisor tasks more efficient and save time for both actors. The goal is to improve key tasks while not trying to replace existing workflows that are already efficient and working well.

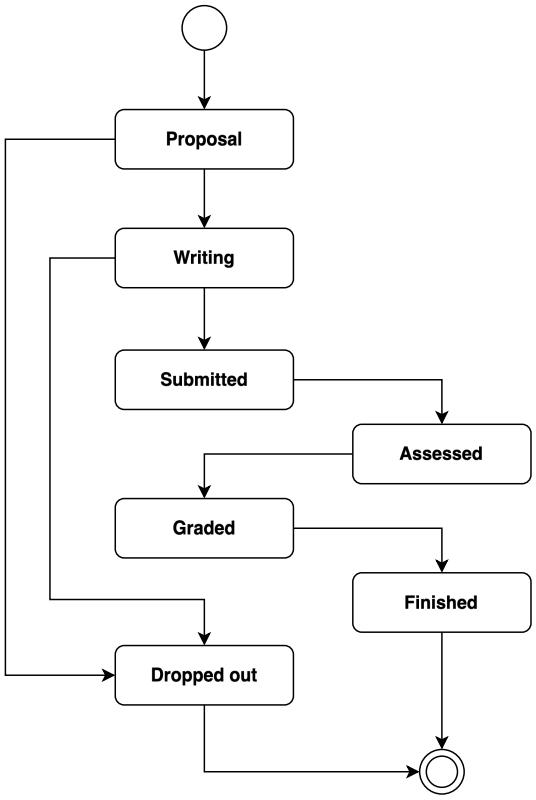


Figure 1: Thesis State Lifecycle for Writing a Thesis

A thesis progresses through various states, as outlined in Figure 1. After an advisor accepts the thesis application, the student must submit a proposal for the thesis first, which changes the state to **Writing**. Upon submission of the final

thesis by the student, the state changes to **Submitted**. At this point, the advisor provides an assessment, including a grade recommendation, shifting the state to **Assessed**. The supervisor then reviews the assessment and submits the final grade and feedback, updating the state to **Graded**. After the grade is submitted to the university, the supervisor can set the thesis as **Finished**, marking it as complete. If a student discontinues work during the proposal or writing phase, the thesis state can be changed to **Dropped out**. The aim of this thesis is to integrate and manage all of these thesis states within a web-based interface.

### 1.3.1 Improve Application Process for Students

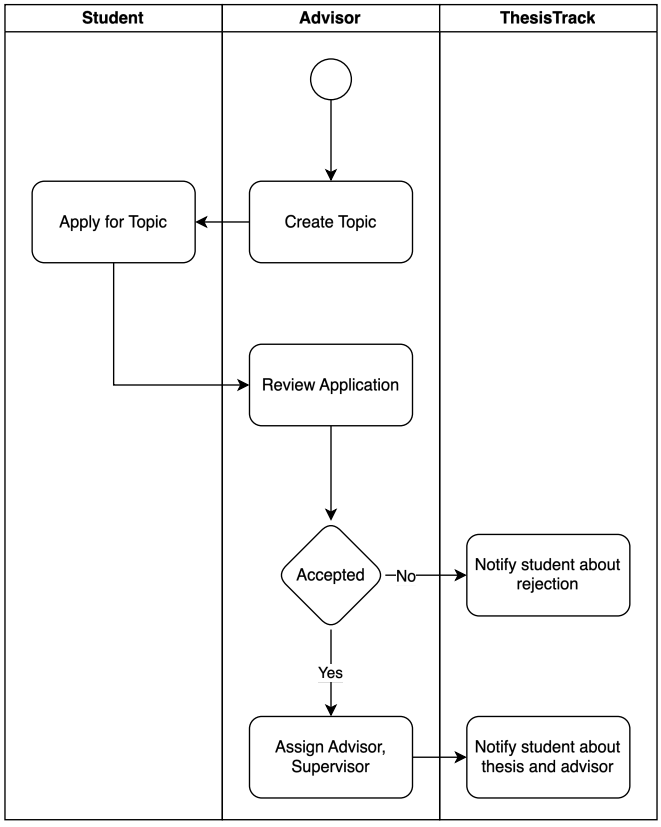


Figure 2: Thesis Application Process for Students and Advisors

Figure 2 shows the thesis application process. Advisors can post topics they are seeking students for, enabling them to quickly receive high-quality, targeted applications. Students can easily browse available topics, apply, and receive timely updates on their application status.

This structured process provides both students and advisors with clear updates, reducing stress and uncertainty for students by keeping them informed throughout the entire journey. Applying for specific topics results in more focused applications, which simplifies the review process for advisors.

### 1.3.2 Integrate Thesis Writing Process

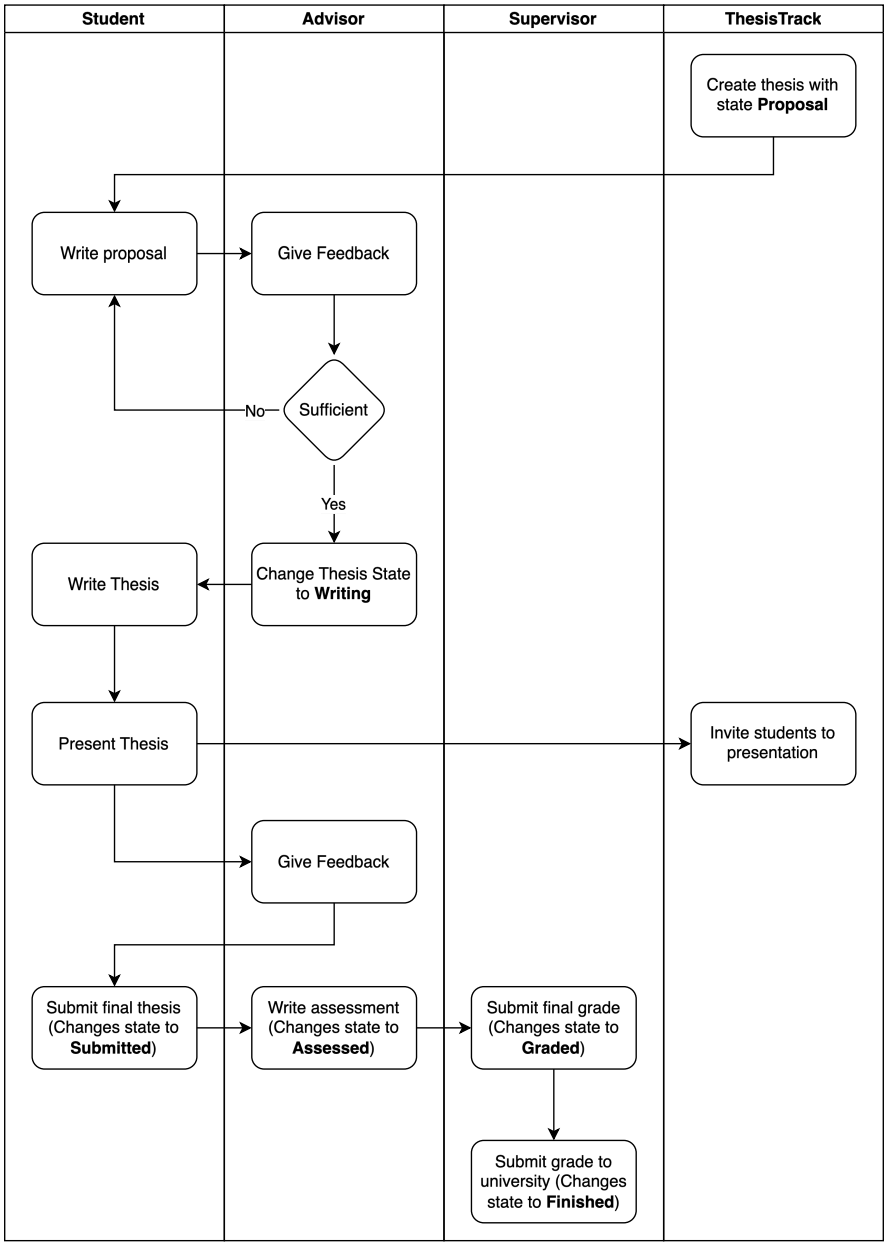


Figure 3: Thesis Writing and Grading Process for Students, Advisors and Supervisors

The thesis writing flow, as illustrated in Figure 3, enhances efficiency by eliminating the need for supervisors and advisors to manually collect and store data from students. This structured flow guides students through each stage, clearly outlining the next steps and tasks they need to complete.

Integrated feedback mechanisms play a crucial role in helping students, especially since a bachelor's thesis is often their first experience with scientific writing [6]. These built-in feedback cycles not only allow for the collection of data on common mistakes but also help improve the quality of the final submission [7].

Embedding different feedback processes within the platform makes it easier to maintain a detailed record of the feedback given and track changes throughout the lifecycle of the thesis. Consolidating all key processes related to thesis writing into a single platform ensures a seamless, user-friendly experience and transparent workflow for all parties involved [8].

### 1.3.3 Provide Overview for Supervisors

Managing advisors and students throughout the thesis process is comparable to overseeing large-scale projects. In such projects, multiple teams work on interrelated tasks, and it's crucial that at least one person has a clear understanding of each team's progress. In the context of thesis supervision, this responsibility falls to the supervisor.

Rather than tracking each thesis individually, an integrated overview that displays the timelines and states of all ongoing theses is essential. This approach allows supervisors to gain a comprehensive understanding of the overall progress, ensuring that key milestones are met and resources are allocated efficiently.

### 1.3.4 Reduce Media Breaks in the Process

Centralizing all documents on a single platform enhances consistency and efficiency. By having all current and past theses accessible in one place, advisors and supervisors no longer need to manually organize files across different services. This approach eliminates the manual overhead that typically burdens advisors, saving time and reducing the likelihood of missing important steps.

The platform is designed to offer a comprehensive view of both completed and ongoing theses, enabling advisors and supervisors to easily track the history and progress of each thesis. This consolidation not only streamlines the process but also ensures consistency and transparency throughout the entire thesis lifecycle.

## 1.4 Outline

In Chapter 2, we first introduce the existing process, followed by an overview of the gathered functional and non-functional requirements. The chapter concludes with UI mockups that demonstrate how the objectives can be realized. In Chapter 3, we explain the transition from the application domain to the solution domain, detailing the platform's development. The design goals are presented initially, followed by a thorough technical description of the implementation. In Chapter 4, we highlight several design patterns that support maintaining high code quality and a good user experience. In Chapter 5, we review the goals and requirements discussed in previous chapters, emphasizing the achievements and identifying any remaining gaps. The chapter concludes with an outlook on potential future improvements for ThesisTrack.

# 2 Requirements Analysis

Based on the identified problems and in close communication with advisors and the professor (supervisor), we collected functional and non-functional requirements. In this chapter, we first go over the existing process and then we introduce the requirements collected. At the end we will show UI Mockups how the solutions could look like.

## 2.1 Existing Process

The professor and doctoral candidates are using the platform ThesisTrack to manage thesis applications. Students can propose a thesis topic and fill in data like previous projects or their motivation. Advisors review submitted applications and can accept or reject them, which will notify the student via email.

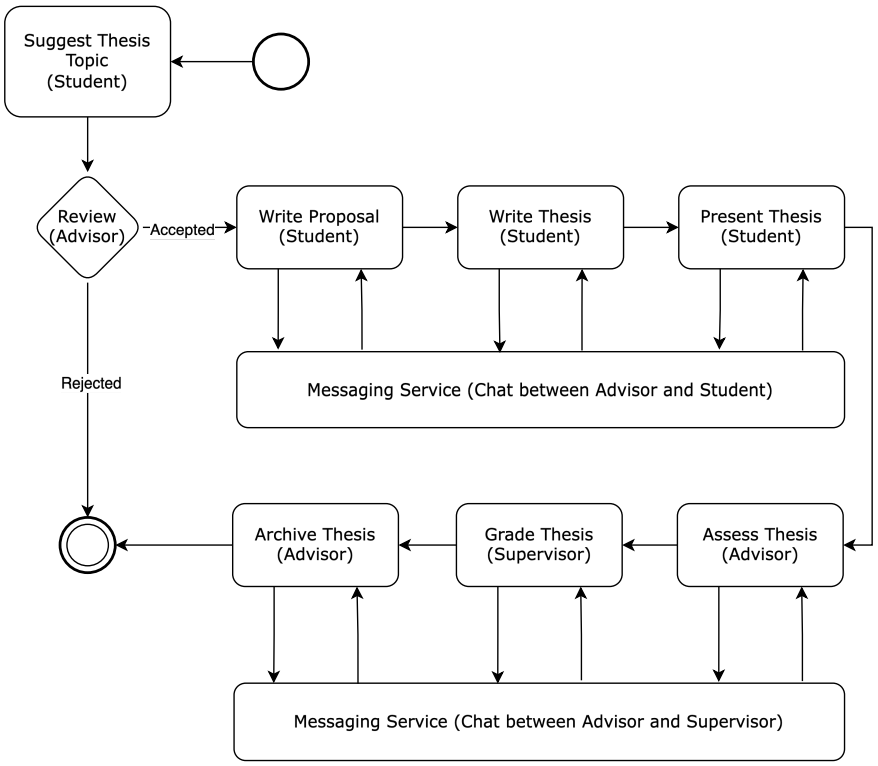


Figure 4: Existing Thesis Writing and File Exchange Process

Once the advisor accepted the application, the communication moves to a messaging service like Slack for all further communication as shown in Figure 4. The

first task of the student is to collect requirements and write a proposal how the identified problems can be solved. The advisor reviews the proposal and sends the student feedback using the messaging service.

After the advisor accepted the proposal, the student starts with the implementation and thesis writing. This consists of multiple feedback cycles with the advisor, that is mostly done via messaging services. The supervisor and advisor communicate over the same way to sync on progress and blockers. At the end of the writing phase, the advisor schedules a presentation date where the student will present their outcomes. The advisor invites other students to the presentation to improve their presentation quality once they have to present their thesis.

After the student submitted and presented the thesis, the advisor assesses it. This includes a summary, strengths, weaknesses and a grade suggestion. This process is communicated with the supervisor using again the messaging service. The supervisor reviews the assessment and decides on a final grade that they send to the student.

Once a thesis is completed, the advisor uploads the files to a cloud with all completed theses. If a thesis is relevant for a student, the advisor will search for it in the cloud and send it to the student.

## 2.2 Functional Requirements

We collected the following functional requirements based on the objectives outlined in Section 1.3. These requirements address the inefficiencies observed in the current system and aim to enhance the overall process for students, advisors, and supervisors. Every functional requirement has an assigned name which will be used as reference in later chapters.

**FR1** Show supervisors and advisors an overview that includes all ongoing theses and their current timeline.

**FR2** Allow students to apply for specific topics the professor is seeking.

**FR3** Ensure advisors can reason why a student's application was rejected, and communicate this effectively so students know what to expect.

**FR4** Assign the appropriate roles and permissions of other services to students when they begin working on their thesis.

**FR5** Capture feedback on submitted files within the platform and make it accessible for improving future feedback cycles.

**FR6** Store all critical thesis-related data on the platform to prevent data loss and reduce reliance on external services for accessing completed theses.

**FR7** Allow students to upload their thesis proposal, final thesis, and presentation files directly to the platform.

**FR8** Enable advisors to submit their assessment on the platform and allow supervisors to add the final grade based on this assessment.

**FR9** Notify all involved actors via email when important actions or updates occur.

**FR10** Allow students and advisors to set presentation dates for theses and invite other students working on their theses to attend.

**FR11** Permit multiple students to collaborate on the same thesis, with each student able to submit their own files like proposals, theses, and presentations.

## 2.3 Nonfunctional Requirements

Nonfunctional requirements specify the system's **usability**, **reliability**, **performance**, and **supportability** (URPS) characteristics [9]. These requirements are essential for long-term sustainability and adoption across different academic environments. We collected the following non functional requirements:

**NFR1** **Supportability**: Ensure that the platform is configurable, allowing other universities to customize university-specific details (e.g. university name, study degrees, study programs, thesis types, required user data), without requiring any modifications to the source code.

**NFR2** **Usability**: Integrate Single Sign-On authentication using university credentials.

**NFR3** **Reliability**: Implement a role-based access control system that restricts access to sensitive data, ensuring that unauthorized access attempts are blocked. The system should support four roles: admin, supervisor, advisor, and student.

**NFR4** **Usability**: Make the platform responsive to work on screen sizes ranging from 320px to 2560px in width.

**NFR5** **Performance**: Provide a RESTful API with an average response time of less than 500 milliseconds for standard requests under typical load, supporting at least 20 concurrent requests without performance degradation.

**NFR6** **Reliability**: Use Docker to containerize the application, ensuring that the application can be deployed on any environment that supports Docker. All configurations should be configurable via environment variables, with zero manual intervention post-deployment.

**NFR7** **Reliability**: Implement Continuous Deployment using pipelines (e.g. GitHub Actions) to automatically deploy code changes to the production or staging environment within 10 minutes after the merge.

## 2.4 Demo Scenario

The following scenario is intended to serve a common way how the new system is supposed to be used: A student logs into ThesisTrack using their university account and navigates to the thesis application page. They are presented with a list of open thesis topics. The student selects a topic of interest, fills out an application form (including a brief motivation letter and background information), and submits it for review.

On the advisor's end, the system automatically notifies the advisor about the new application. The advisor can review the application and provide feedback directly on the platform, either accepting or rejecting the application with detailed reasons. If accepted, the student is automatically granted access to the thesis workspace.

During the thesis writing process, students upload their documents (proposal, presentation, thesis) through the system, where advisors provide structured feedback that's stored and organized for easy future reference. Advisors have a comprehensive overview of all ongoing theses, allowing them to manage their time and students effectively, without needing to rely on external tools.

When the student submits the final version of their thesis, the file upload is locked. At this stage, the advisor schedules a presentation directly in the system, selecting a date and time that is suitable for all relevant parties. Once the presentation is scheduled, automated notifications are sent to the student, the supervisor, and any other participants. The presentation details are recorded, and students working on related topics can be invited to attend.

After the presentation is completed, the advisor writes a formal assessment of the thesis. This includes comments on the quality of the work, the student's performance during the process, and a suggested grade.

Upon submission of the advisor's assessment, the supervisor is notified via an automated email. The supervisor then reviews the thesis and the assessment within the platform. After considering the feedback, the supervisor adds a final grade and provides their own comments.

The supervisor can then mark the thesis as completed, which automatically archives it in the system. The thesis, along with all related files, feedback, and grades is stored in a centralized database. This ensures easy access for future reference and eliminates the risk of losing important information. If the visibility of the thesis is set to public, it is made available on the platform's public page for other students to view.

## 2.5 Analysis Object Model

Figure 5 outlines the most important entities within the problem domain. The **Thesis** is the central entity and the main addition to ThesisTrack as part of this bachelor's thesis. The system differentiates between users as **Supervisors**, **Advisors** and **Students**. A **Student** can create an **Application** for a **Topic** or

suggest their own topic. When the **Advisor** accepted the **Application**, a **Thesis** is created based on it.

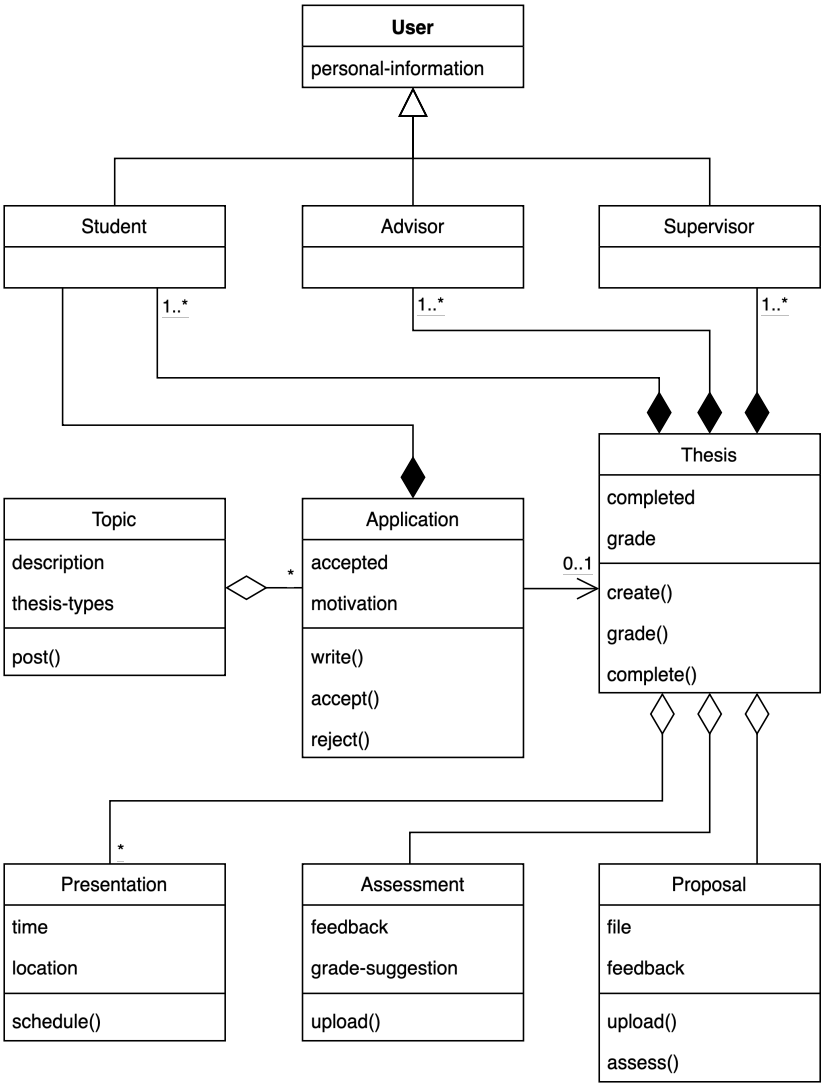


Figure 5: Analysis Object Modal of the Problem Domain

Every **Thesis** must have at least one **Supervisor**, **Advisor** and **Student**. It can have a **Proposal** when the **Student** added it, or an **Assessment** if the **Advisor** assessed the thesis. A **Thesis** can have multiple presentations. Every **Thesis** should have at least one **Presentation**, **Assessment** and **Proposal** once completed. The data is added as part of the the process described in Section 2.4.

## 2.6 User Interface

Based on the defined objectives in Section 1.3 and the functional requirements, there are three core parts of the platform we want to highlight and explain. These interfaces represent the core addition of this thesis and help to solve the objectives.

### 2.6.1 Updated Application Flow

Students should be able to apply for specific topics the professor is searching for (FR2). This is achieved with a new three-step application process that allows students to apply for multiple topics without having to re-enter data that does not change.

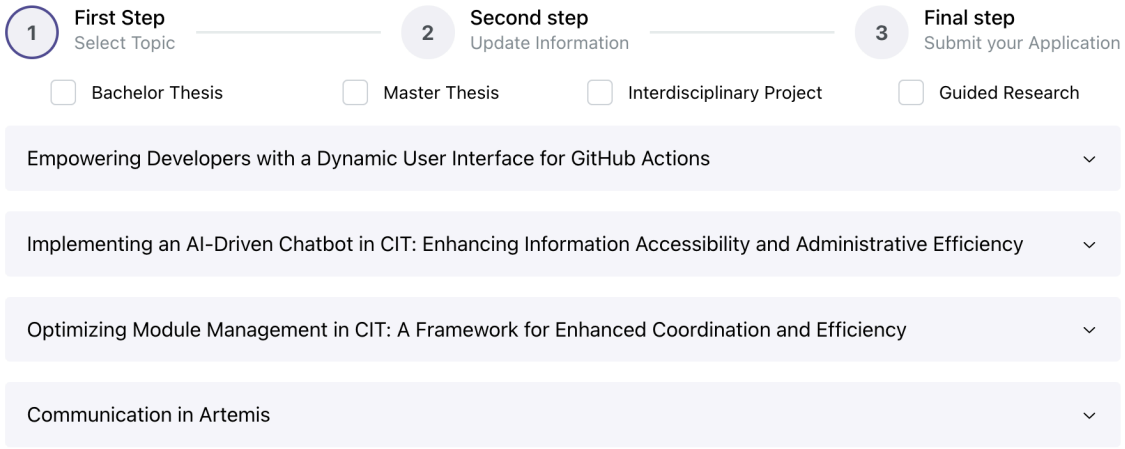


Figure 6: Submit Thesis Application Step 1 Mockup

At the first stage, shown in Figure 6, the student can select a topic to apply for. By default the topics are collapsed in an accordion but the student can expand each topic to get more information about it. The student can filter for the thesis type at the top to filter out topics that are not of interest. The student can also suggest an own topic if desired.

In the second step, the student adds personal information like study program, current semester, past projects, special skills and interests. This data is stored on user level and only needs to be entered once.

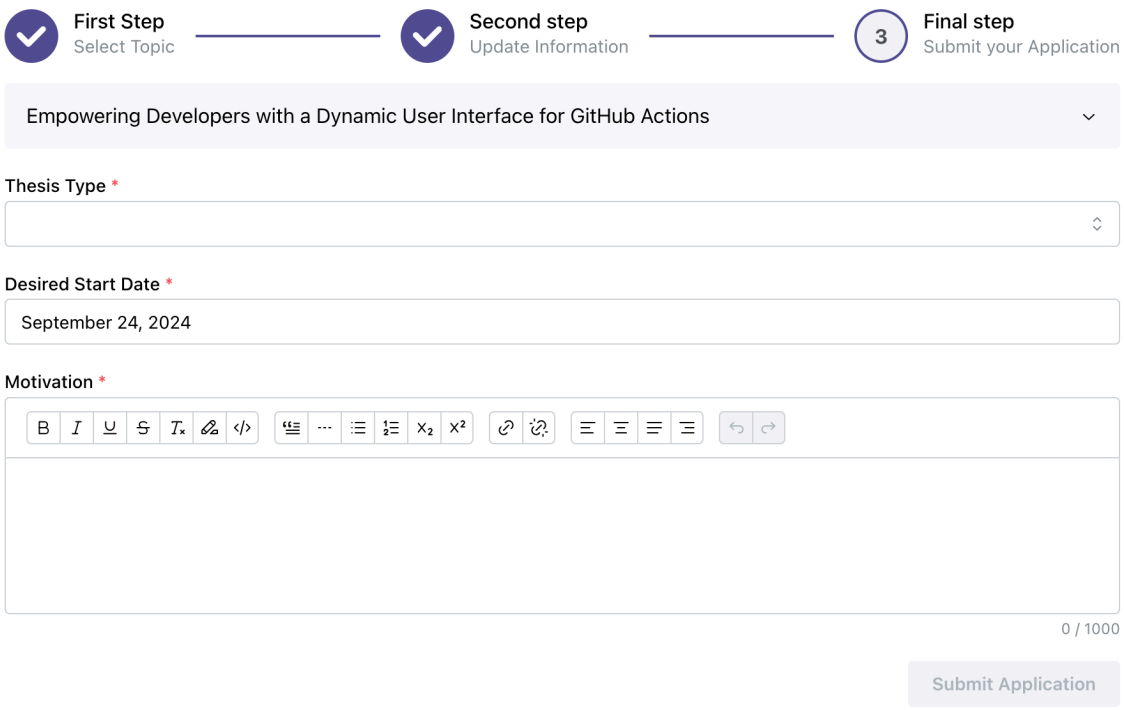


Figure 7: Submit Thesis Application Step 3 Mockup

The third and last step is writing the motivation for the selected topic as shown in Figure 7. Details of the topic can be expanded at the top to help students to write a concise text on why they would be the best choice. The student can select a thesis type and a desired start date for the thesis to set the right expectations.

### 2.6.2 Thesis Management Page

The thesis management page is a single page that provides the functionality for managing the complete thesis writing and grading process (FR5, FR6, FR7, FR8, FR10). The page has different sections that represent the various steps involved in the process. The following figures show the different steps in the same order like shown on the page. Each section can be folded out to reduce the page height and hide sections that are not relevant in the current status of the thesis. The sections are only visible as soon as the data is required.

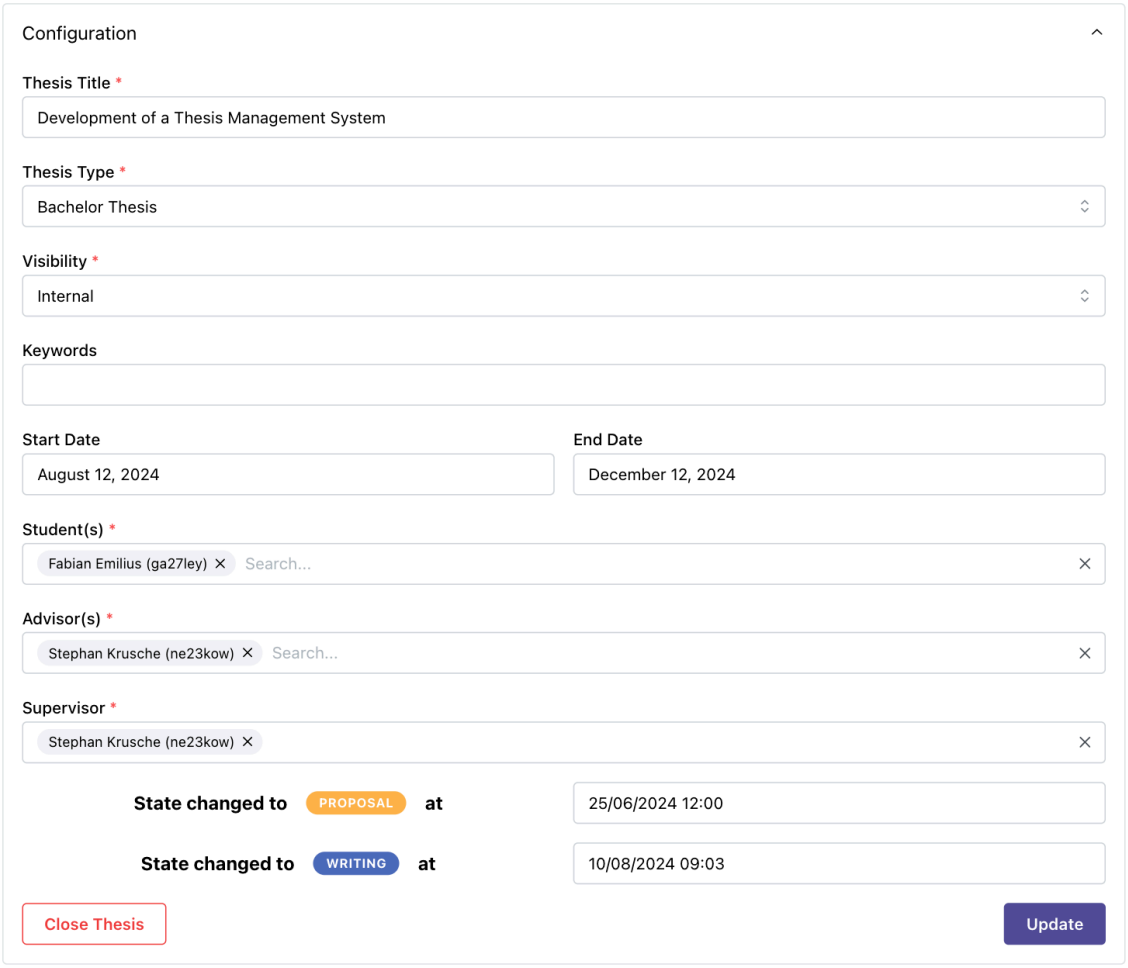


Figure 8: Thesis Configuration Section Mockup

Figure 8 shows the **Thesis Configuration Section**, which offers a structured interface for setting up thesis-related parameters, such as start date, end date, visibilities, and responsible users. Advisors can use this section to update thesis data after it was created. When a student stops working on the thesis, it can be closed with the **Close Thesis** button. This section is collapsed on initial load because these settings do not change frequently in the writing process.

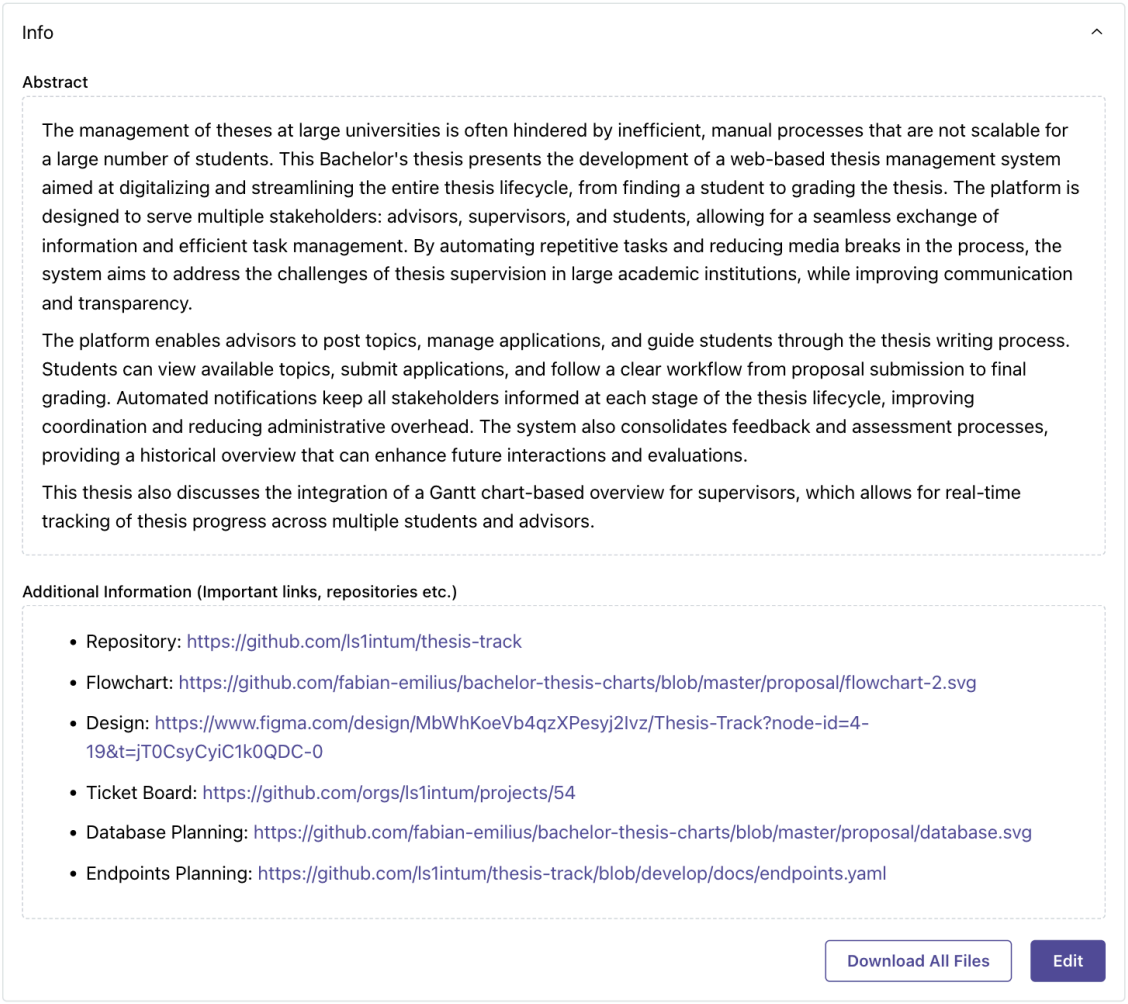


Figure 9: Thesis Info Section Mockup

Figure 9 shows the **Thesis Info Section**, which serves as a central hub for displaying relevant information about a particular thesis. The student can add the abstract of the thesis which is used in other functionality like the presentation invitations to add relevant context. The **Additional Information** field can be used to add important links to the thesis page like coding repositories or mockups. At the bottom there is a **Download All Files** button that downloads all uploaded files as a compressed archive. This section helps to quickly get an overview of the thesis.

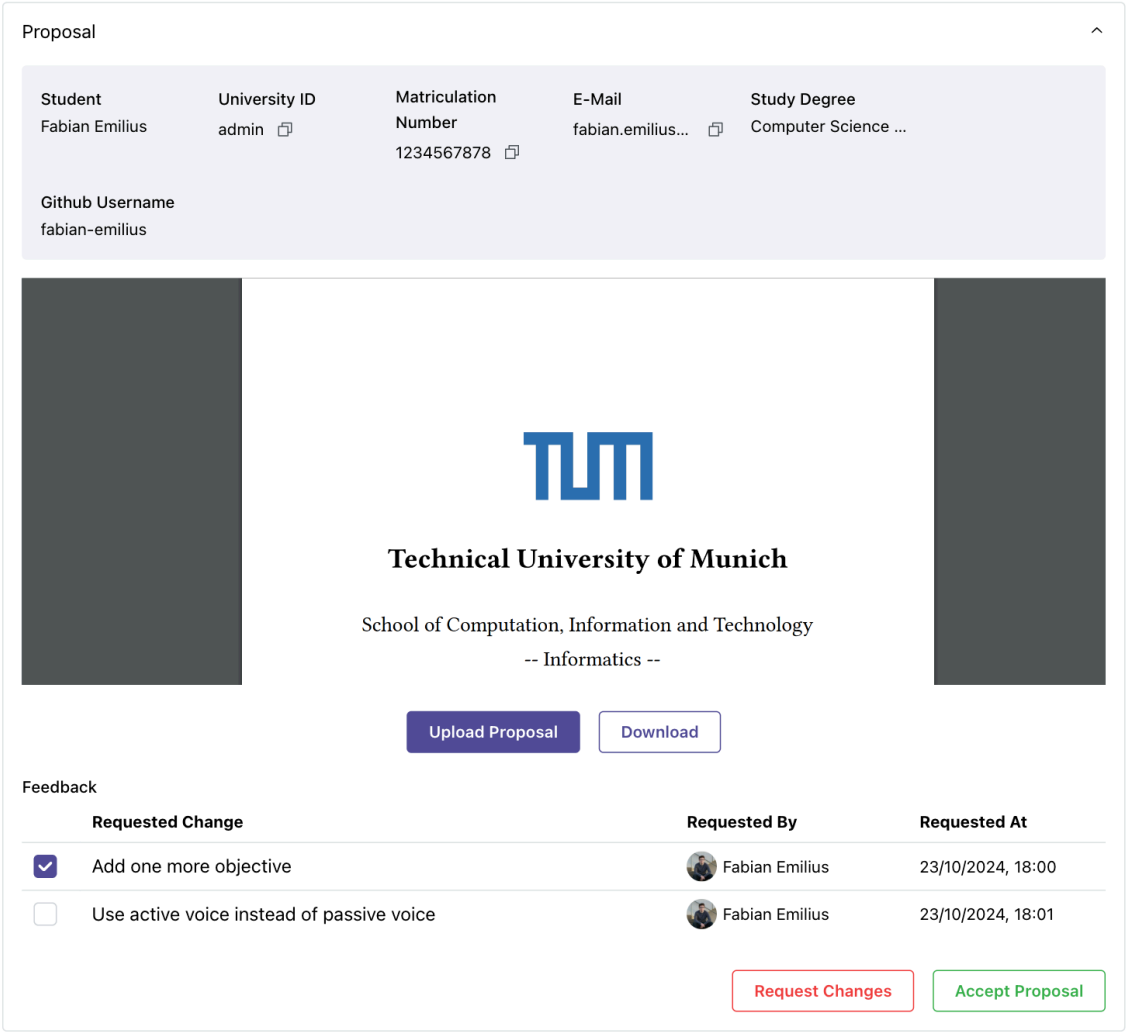


Figure 10: Thesis Proposal Section Mockup

Figure 10 shows the **Thesis Proposal Section**, which allows students to submit their proposals directly through the platform. Advisors can review the proposal and provide feedback within the system, promoting a clear feedback loop to help the student to improve their scientific writing skills [10]. The student can mark requested changes as completed and upload a new proposal. When the advisor accepts the proposal, the thesis state changes to **Writing**.

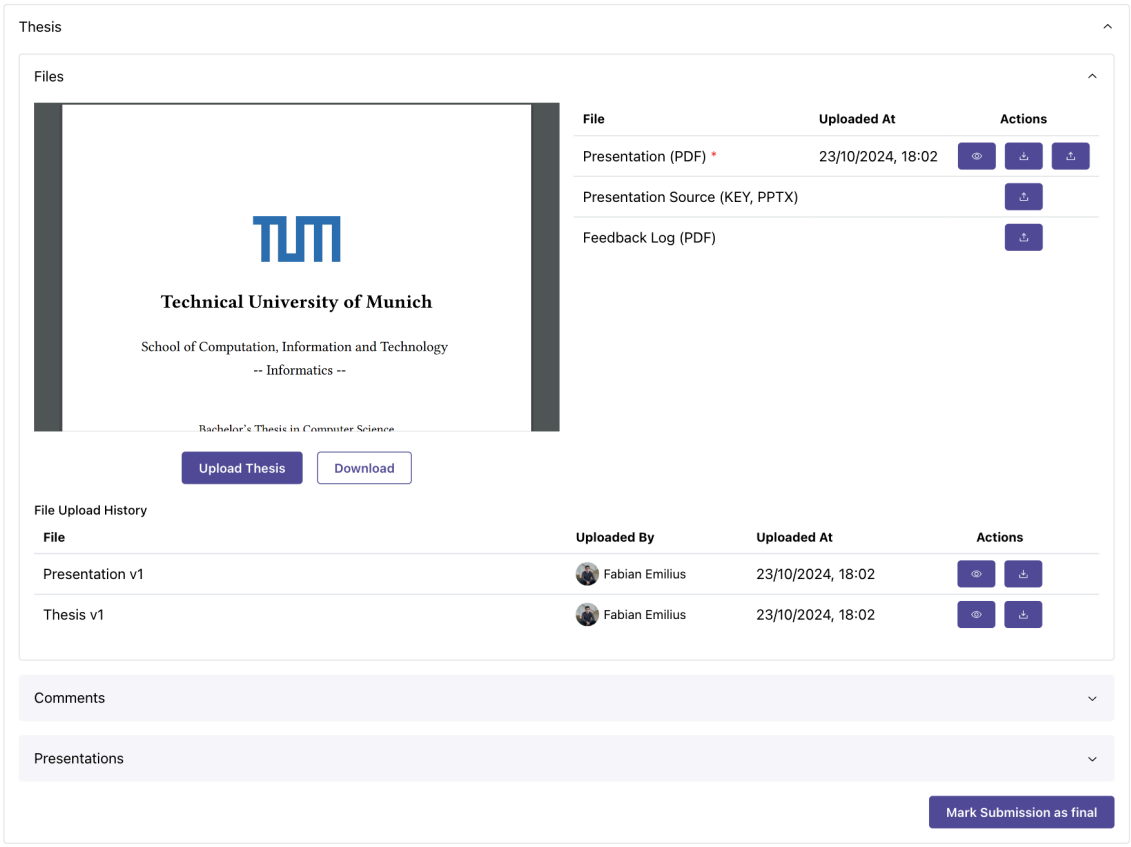


Figure 11: Thesis File Upload Section Mockup

Figure 11 is part of the **Thesis Writing Section** and allows the student to upload all thesis-related documents, such as the thesis and the presentation. The student can change uploaded files as long as the thesis is not submitted yet. The **File Upload History** shows an upload log of all files added to the thesis, making it possible to download older versions as well. The button **Mark Submission as Final** changes the thesis state to **Submitted**. Then no further changes can be made.

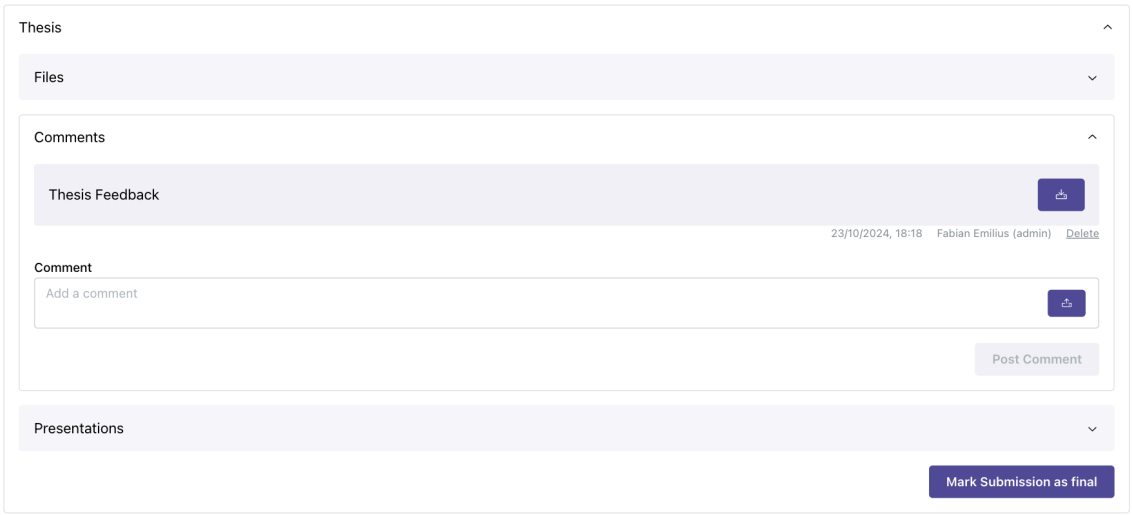


Figure 12: Thesis Comment Section Mockup

Figure 12 is part of the **Thesis Writing Section** and allows students and advisors to add important information or files to the thesis. It's not meant to be a communication channel, but rather a way to log feedback or important milestones that are relevant for grading the thesis. A comment can contain text, but also an arbitrary file that can be downloaded. This helps to persist arbitrary files that are not part of the main submission.

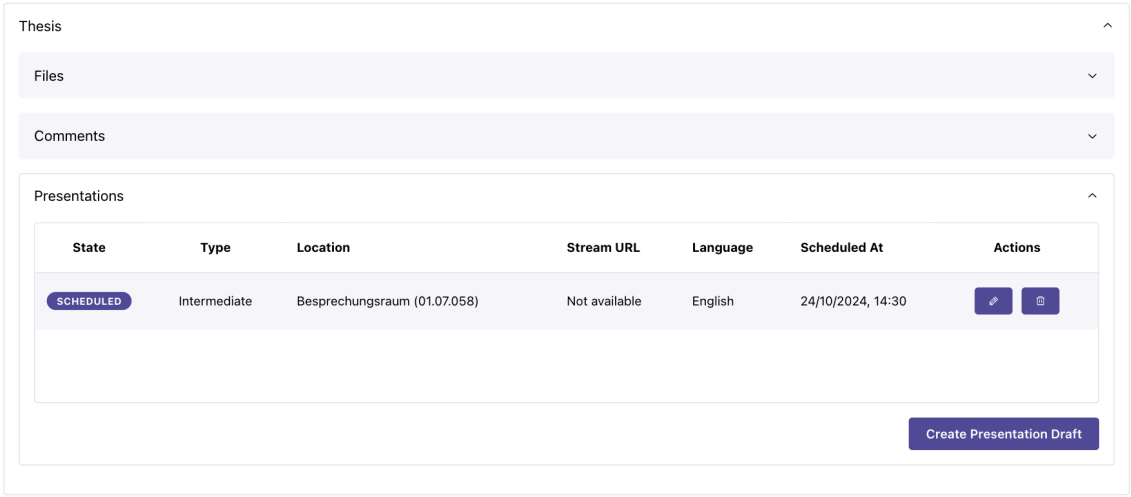


Figure 13: Thesis Presentation Section Mockup

Figure 13 is part of the **Thesis Writing Section** and provides a dedicated space for scheduling and managing thesis presentations. This section enables students and advisors to coordinate presentation dates and invite relevant participants. A student can create a presentation draft which can be reviewed and edited by the advisor. The advisor can approve the draft and configure which other users should be invited to the presentation. The options are: **All chair members** and **All students that are currently writing a thesis**. The advisor can invite custom email addresses to the presentation as well.

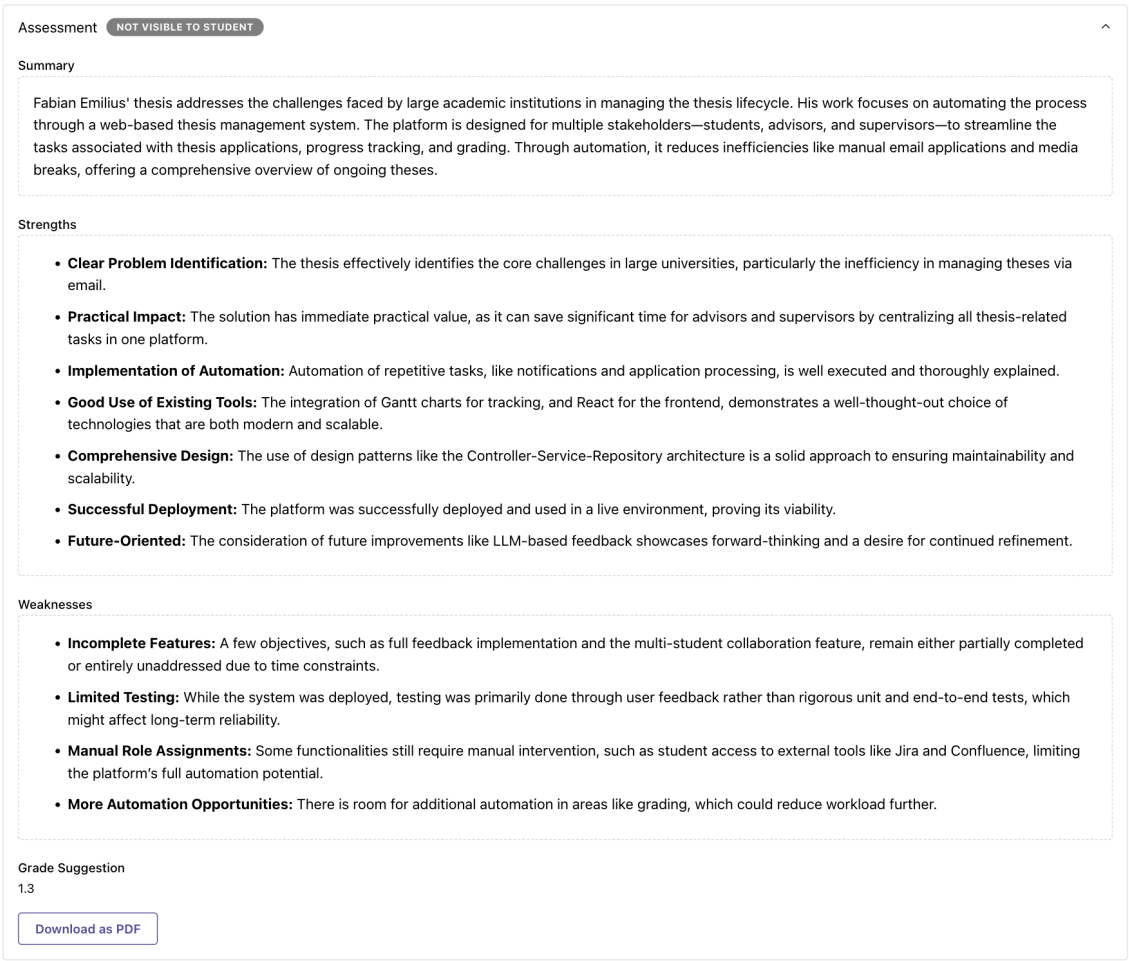


Figure 14: Thesis Assessment Section Mockup

Figure 14 shows the **Thesis Assessment Section**, where advisors can provide a formal evaluation of the thesis. This section is only visible to the advisor and supervisor but not to the student. The advisor adds qualitative feedback and a grade suggestion that describes the student's performance throughout the process after the student submitted the thesis. Once the assessment is added, the supervisor is notified and the thesis state changes to **Assessed**.

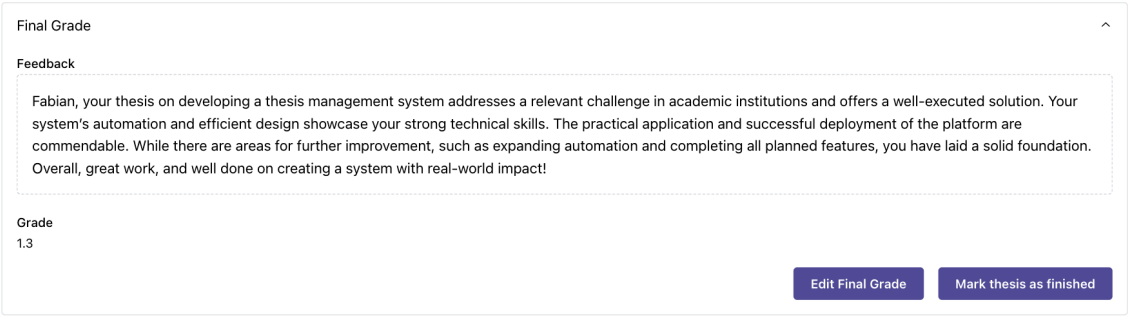


Figure 15: Thesis Grading Section Mockup

Figure 15 shows the **Thesis Grading Section**, which allows the supervisor to submit the final grade after reviewing the advisor's assessment. This section is designed to ensure that grading is done transparently and is aligned with the formal assessment provided earlier. The supervisor can add feedback which will be visible to the student. Once the grade is added, the supervisor can complete the thesis marking it as closed.

### 2.6.3 Theses Overview Gantt Chart

The theses overview should provide advisors and especially supervisors an overview of all ongoing theses (FR1). Gantt charts are widely used in project management across different industries because they are an effective and simple way to track the progress of tasks and teams [11].

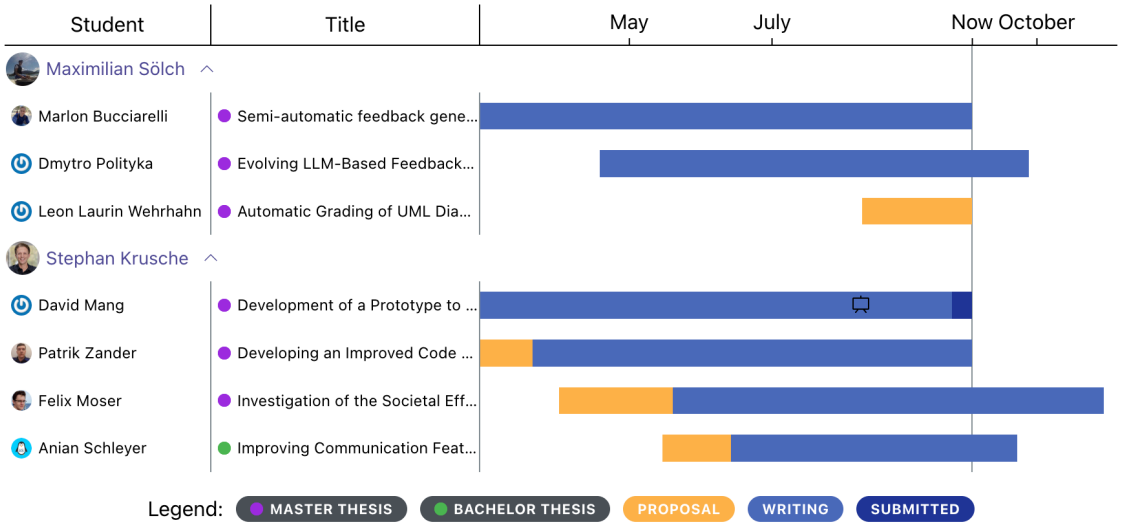


Figure 16: Thesis Overview Gantt Chart Mockup

A Gantt chart to track the progress of students and advisors could look like the one shown in Figure 16. The timelines are grouped by the advisor to see which advisor has capacity or when their theses are expected to end. Each timeline has different colors to represent the states from Figure 1. This helps to see what is still missing and how long it will take to finish the thesis. The chart includes the student's name and the thesis title to quickly see who is working on it and which topic the thesis is about. The TV icon shows the presentation date of the thesis. When hovering over the icon or another element of the chart, more information like the date is shown to the user.

# 3 System Design

In this chapter we convert the application domain from Chapter 2 into the solution domain. The structure follows the System Design chapter in "Object-Oriented Software Engineering Using UML, Patterns, and Java" [9].

The platform ThesisTrack is using a simple client-server architecture with one client and one API server. The client is using React in combination with TypeScript to provide a modern single page app because React is the most used web framework in modern software development [12]. We are using Mantine as component library for basic styling and base components. The client communicates with the server over a REST API. The server is written in Java using the Spring Boot framework. We use PostgreSQL in combination with Hibernate to store and access the user data.

## 3.1 Design Goals

The design goals are derived from the non-functional requirements introduced in Section 2.3. Defining design goals is a key step in guiding the development process of the proposed system. These goals serve as a point of reference when facing conflicting requirements [9]. The design goals are ordered according to their importance. If a feature conflicts with a goal, it needs to be reprioritized or changed so that it fits.

**DG1** **Security:** Since the system will handle sensitive student data, it must incorporate a role-based access control mechanism to ensure that only authorized users have access to the information (NFR2, NFR3).

**DG2** **Usability:** The user interface should follow Nielsen's usability principles [13], ensuring an intuitive experience that does not require additional training for students or advisors (NFR4).

**DG3** **Configuration:** The platform should be configurable for other universities to host it and adjust it to their requirements (NFR1).

**DG4** **Maintainability:** The platform should be developed with a modular, decoupled architecture to facilitate easy updates, bug fixes, and the addition of new features (NFR5, NFR6, NFT7).

## 3.2 Subsystem Decomposition

ThesisTrack is composed of 4 main subsystems that handle the thesis application and thesis writing lifecycle: the **Client**, the **Server**, a PostgreSQL **Database** and a **Postfix** instance. The design of this system follows a modular architecture, where components are decoupled into manageable services that interact via well-defined APIs, ensuring maintainability and security.

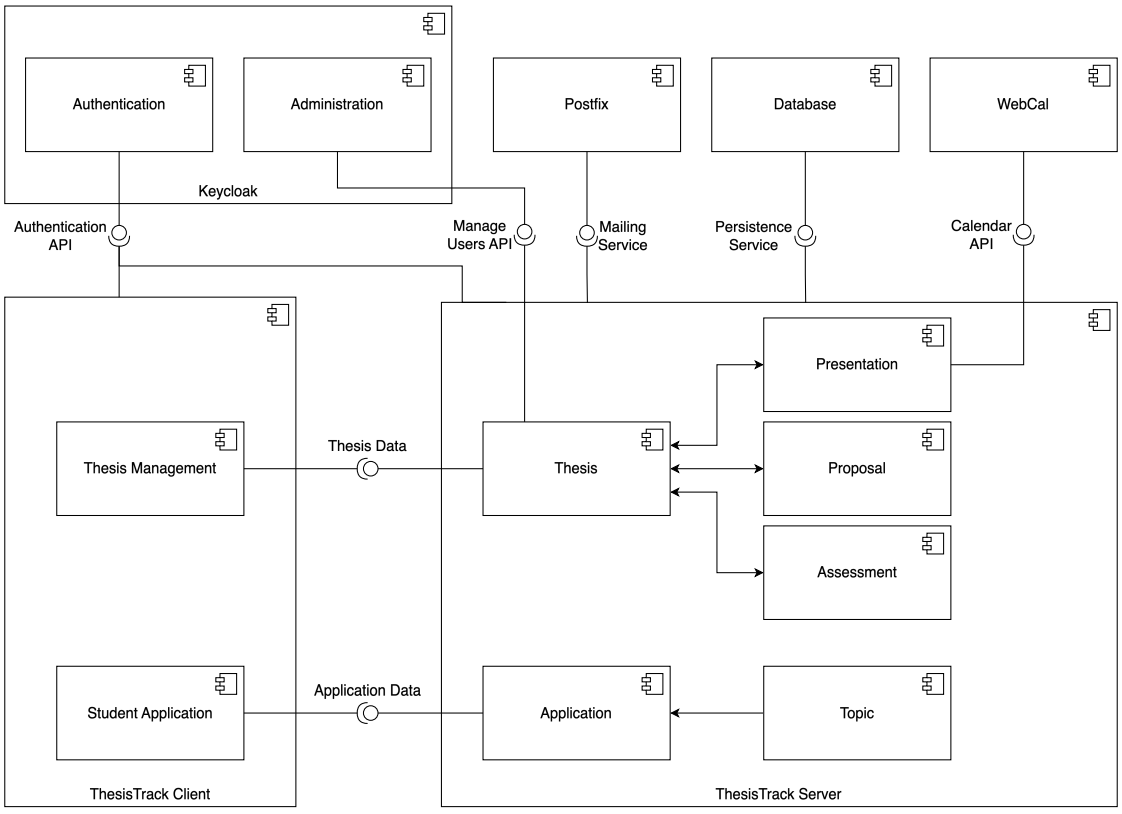


Figure 17: ThesisTrack Subsystem Decomposition

The client is divided into two independent subsystems as illustrated in Figure 17: the **Student Application** subsystem, which handles the thesis application and review process, and the **Thesis Writing** subsystem, which manages the writing, feedback, and grading stages. Each subsystem communicates with the server via

a REST API, ensuring data consistency and security. The server can be divided into similar subsystems that provide the data for the client. The server subsystems exchange data with other systems that are part of the server, namely the **Presentation**, **Proposal**, **Assessment** and **Topic** service.

**Keycloak** provides an authentication API that is used in the client and server to verify user authentication. **Keycloak** is connected to the university authentication server, and students can login with their university accounts. The **Client** gets the authentication secrets from **Keycloak**, and the server verifies them on every request. The thesis subsystem of the server connects to the **Keycloak Admin API** to assign the student group to users when they are starting to write a thesis. This role is removed once the student completes their thesis. When a **CalDav** server is configured, it is used to store the presentation dates as calendar events.

## 3.3 Persistent Data Management

There are primarily two different kinds of data stored on the platform. The first type are files (CVs, uploaded theses, uploaded presentations) that are stored in the file system. When using Docker, these files need to be mounted to the OS filesystem together with a backup strategy for that folder to avoid data loss. The second type is the user data being stored and queried in the PostgreSQL database shown in Figure 18.

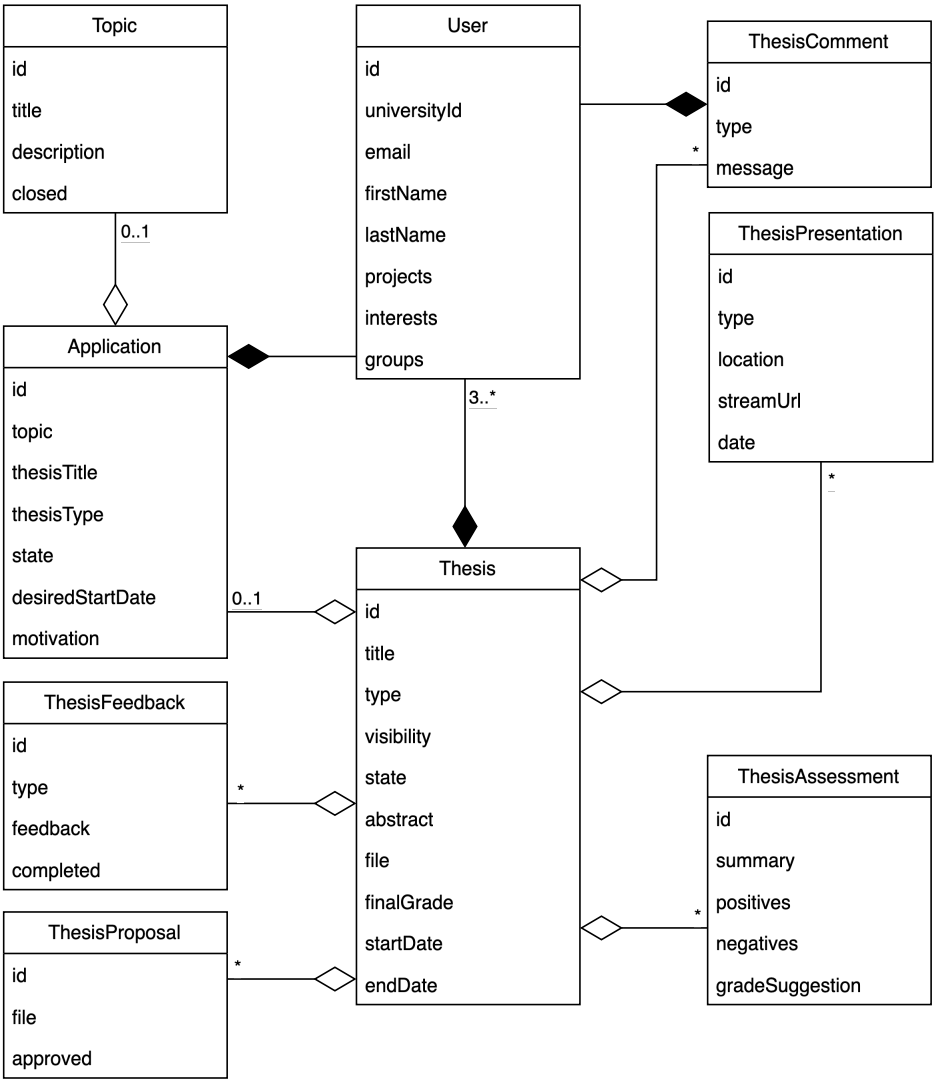


Figure 18: Simplified Relational Database Schema for ThesisTrack

We designed the relational database schema to support the complex, multi-role nature of academic thesis management. The use of a relational model is driven by the need to represent and enforce structured relationships between core entities in the thesis process. Central to this system is the **Thesis** entity, which connects to key participants to support associated processes, such as **Application**, **Proposal**, **Assessment**, and **Presentation**. These entities are tightly coupled in terms of workflow, and their relationships exhibit clear one-to-many or many-to-one mappings that are represented in the relational schema. For instance, a single thesis may involve multiple students, while each advisor or supervisor can oversee

numerous theses. By employing foreign keys and relational integrity constraints, the schema ensures that these relationships are consistently maintained across the system.

## 3.4 Access Control

ThesisTrack uses Keycloak[4] as the primary authentication provider. There are four Keycloak roles to manage user access throughout the platform. Each role is a superset of the previous role, meaning that all permissions from the lower group are inherited (e.g. an advisor has all permissions a student has).

| | |
|---|---|
| **No Role** | Users without a role can see open topics, published theses, published presentations, and can submit new applications for a topic. Users can update their user information like study degree and current semester. |
| **Student** | ThesisTrack assigns and unassigns the student role automatically when the user starts writing a thesis or completes it. Only users that are currently writing a thesis have the student role. These users can view the "browse theses" page and can see theses that have the visibility **Public** or **Students**. |
| **Advisor** | Users with the advisor role can be added as advisor to a thesis. They can review applications and create or update topics. Advisors can see all theses that have the visibility **Internal**. |
| **Supervisor** | Users with the supervisor role can be added as supervisor to a thesis. |
| **Admin** | The admin role has all permissions and can perform every action. |

The read acccess of a thesis is managed by the visibility setting in the thesis configuration section. Similar to the role permissions, every visibility is a superset of the last one (e.g all users that can access a thesis with **Internal** visibility, can also see theses with **Students** visibility)

[4]https://www.keycloak.org/

| | |
|---|---|
| **Public** | Every user who is authenticated to the platform can see the thesis. When the thesis is completed, it will be published on the landing page and unauthenticated users can see it as well. |
| **Students** | All users with the student role can see the thesis. |
| **Internal** | Users with the roles admin, supervisor or advisor can see the thesis. |
| **Private** | Only users that are specifically assigned to the thesis and admins can see the thesis. |

The write access of a thesis is managed outside of the Keycloak role system. Every thesis has specifically assigned users as student, advisor and supervisor. The permissions follow the same role names, but the Keycloak roles do not affect the thesis permissions except for the admin role. The admin role can view and manage all theses on the platform.

| **Thesis Action** | **Supervisor** | **Advisor** | **Student** |
|---|---|---|---|
| Create Thesis | ✓ | ✓ | ✗ |
| View Thesis | ✓ | ✓ | ✓ |
| Configure Thesis | ✓ | ✓ | ✗ |
| Add Abstract / Additional Information | ✓ | ✓ | ✓ |
| Upload Proposal | ✓ | ✓ | ✓ |
| Request Proposal Changes | ✓ | ✓ | ✗ |
| Accept Proposal | ✓ | ✓ | ✗ |
| Upload Thesis | ✓ | ✓ | ✓ |
| Create Presentation Draft | ✓ | ✓ | ✓ |
| Schedule Presentation | ✓ | ✓ | ✗ |
| Post Student Comment | ✓ | ✓ | ✓ |
| Post Advisor Comment | ✓ | ✓ | ✗ |
| Submit Thesis | ✓ | ✓ | ✓ |
| Add Assessment | ✓ | ✓ | ✗ |
| View Assessment | ✓ | ✓ | ✗ |
| Add final grade | ✓ | ✗ | ✗ |
| Complete Thesis | ✓ | ✗ | ✗ |

Table 1: Manage Thesis Access Rights for Student, Advisor, Supervisor Role

Table 1 shows the thesis manage permissions for the assigned thesis roles: student, advisor, supervisor. Similar to the other permissions, every role inherits the permissions of the lower role (e.g. a supervisor has all permissions of student and advisor).

# 4 Object Design

In modern web development, designing systems that prioritize reusability and scalability is critical for maintaining efficient, long-term software solutions [14]. This chapter focuses on object design principles that support creating modular components and managing state in a structured manner. By applying design patterns that emphasize consistency and flexibility, the architecture enhances overall system performance while improving both user experience and maintainability.

## 4.1 Controller-Service-Repository Pattern

For the server, we use the Controller-Service-Repository (CSR) pattern, a widely adopted architectural design in Spring Boot applications [15]. This pattern, shown in Figure 19, organizes the application into three distinct layers, each responsible for a specific part of the application's logic. The **Controller** layer manages HTTP requests and responses, ensuring proper interaction with client-side inputs and outputs. It delegates business logic to the **Service** layer, which contains the core functionality of the application. The **Service** layer handles operations, orchestrates interactions between components, and enforces business rules. Lastly, the **Repository** layer manages database interactions, encapsulating data access and providing an abstraction over the persistence mechanism.

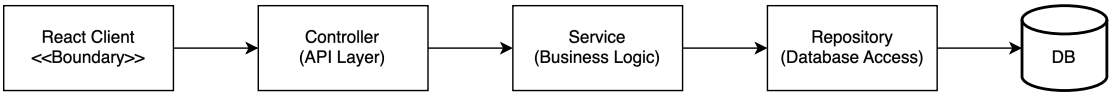


Figure 19: Spring Boot Controller-Service-Repository Pattern

The key advantage of the CSR pattern lies in its clear separation of responsibilities, which enhances maintainability, scalability, and testability. By keeping the API handling, business logic, and data access in separate layers, developers can modify one part of the system without impacting others, improving flexibility during development and system evolution. This structure also promotes reusability, as services and repositories can be shared across different controllers or external systems. Moreover, isolating business logic within the Service layer allows for easier unit testing, as the services can be tested independently from the web

interface and database, leading to more reliable and efficient testing processes. Additionally, the repository abstraction minimizes direct dependencies on specific database implementations, making the architecture adaptable to future changes in data storage solutions.

## 4.2 Reusable Gantt Chart Component

In Section 2.6.3, we discussed the implementation of a Gantt chart to provide a comprehensive overview of all ongoing theses. The **ThesesGanttChart** component utilizes the **GanttChart** component, which is designed to be highly adaptable and can be repurposed for various project management and tracking requirements. The key to its reusability lies in the customizable properties. By defining a flexible interface, the component can accept various configurations, such as custom columns, data, and event handlers, making it reusable for different use cases or tools.

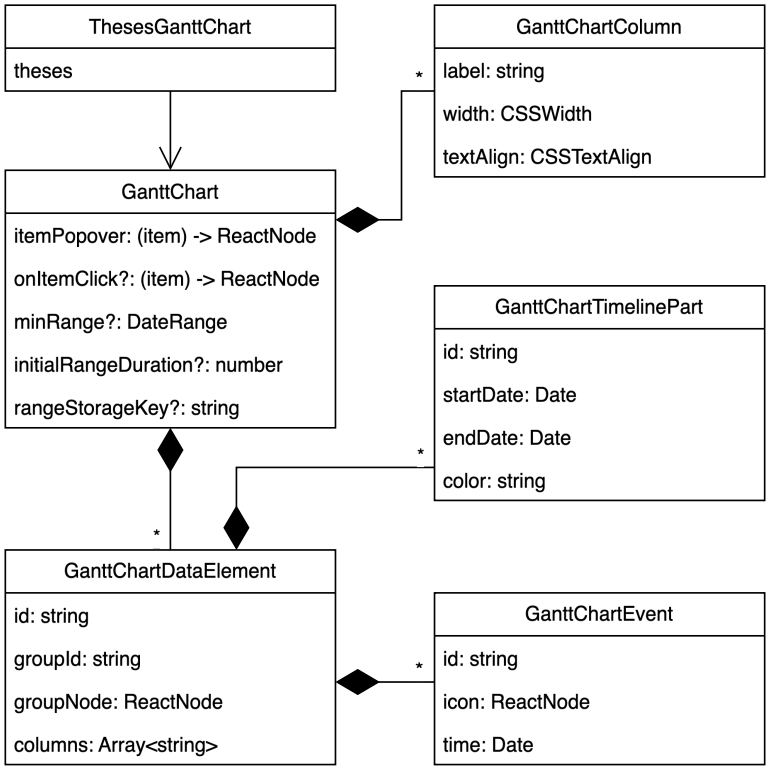


Figure 20: Reusable GanttChart Component Properties Definition

In Figure 20, the **GanttChart** class represents the React component responsible for rendering the Gantt chart. Its attributes and structure define how the component is configured. The **ThesesGanttChart** class is a React component that populates the **GanttChart** with the necessary data to display an overview of the theses, representing one potential use case for the **GanttChart** component.

The **itemPopover** attribute defines a rendering function that displays a custom popover when hovering over a timeline item or event, enhancing user interaction. Optional functionality is introduced via the **onItemClick** callback, which triggers an action when a timeline item is clicked. The **minRange** attribute defines the minimum range of the timeline filter, while **initialRangeDuration** sets a default duration for the displayed timeline. The **rangeStorageKey** allows local storage of the timeline range settings, ensuring persistence across user sessions.

The **GanttChartDataElement** class, shown in Figure 20, represents the core data structure for each row of the Gantt chart. The **id** and **groupId** fields uniquely identify each element and its associated group, respectively, while **groupNode** contains the visual representation of the group, such as the advisor's name. The **columns** array holds the data that populates the cells corresponding to the columns defined by the **GanttChartColumn** class.

Each **GanttChartDataElement** may contain multiple instances of the **GanttChartTimelinePart** class, which provides detailed information for each timeline entry, including an **id**, **startDate**, **endDate**, and a **color**, visualizing different parts of the timeline. The **GanttChartEvent** class represents events such as presentations, marked by an **id**, an **icon** for visual representation, and the specific **time** of the event. Multiple events can be associated with a single **GanttChartDataElement**. Together, these attributes offer flexibility in displaying and interacting with Gantt chart data across a variety of use cases.

## 4.3 Data Provider Pattern

We efficiently reused data fetching logic across multiple pages by leveraging React's Context API. This approach centralizes the logic for fetching and managing the state of entities like Applications, Theses, and Topics, allowing seamless access across child components within different parts of the application. This enables child component to update objects for all other components when changes occur avoiding page reloads or data refreshes. This pattern significantly reduces duplication of data-fetching logic while ensuring consistency in how data is fetched and updated.

```
<ThesesProvider
  limit={20}
  defaultStates={[
    ThesisState.PROPOSAL,
    ThesisState.WRITING,
    ThesisState.SUBMITTED,
    ThesisState.ASSESSED,
    ThesisState.GRADED,
  ]}
>
  <ThesesFilters />
  <ThesesGanttChart />
  <ThesesTable />
</ThesesProvider>
```

Figure 21: ThesesProvider Example with Filters, GanttChart and a Table

In Figure 21, components like **ThesesFilters**, **ThesesGanttChart**, and **ThesesTable** all share the same data context ensuring updates propagate instantly. This setup is flexible, allowing components to be composed differently depending on the page's requirements: For example, omitting the theses filters or the Gantt chart on pages that don't need them.

# 5 Summary

In this chapter we summarize what we have achieved and what is still missing. In the end we provide an outlook what could be implemented to furthermore improve the processes and save time for advisors and supervisors.

Table 2 shows the current status of the functional requirements introduced in Section 2.2.

| # | **Short Description** | **Status** |
|---|---|---|
| **FR1** | Show supervisors and advisors an overview | Done |
| **FR2** | Allow students to apply for specific topics | Done |
| **FR3** | Ensure advisors can reason why an application was rejected | Done |
| **FR4** | Assign the appropriate roles and permissions of other services | Partially Done |
| **FR5** | Capture feedback on submitted files within the platform | Partially Done |
| **FR6** | Store all critical thesis-related data on the platform | Done |
| **FR7** | Allow students to upload their files on the platform | Done |
| **FR8** | Enable supervisor to manage the grading process | Done |
| **FR9** | Notify all involved actors | Done |
| **FR10** | Allow students and advisors to set presentation dates | Done |
| **FR11** | Permit multiple students to collaborate on the same thesis | Open |

Table 2: Status of Functional Requirements

## 5.1 Realized Goals

We successfully implemented all high-priority requirements requested by the professor and advisors (FR1 to FR10). Using Continuous Delivery, we deployed incremental updates, with each feature released to production upon completion. As a result, the platform is already in use for students beginning or finalizing their theses in the Winter Semester 2024/2025. This approach enables continuous feedback cycles and testing during the development of this thesis, ensuring a stable and reliable system.

## 5.2 Open Goals

**FR4** is marked as partially completed due to the potential for additional automation when a student begins working on a thesis. Currently, the system adds the student to a Keycloak group, which grants the necessary permissions for other services. However, some tools, such as Confluence and Jira, do not utilize Keycloak as their primary authentication and permission provider, requiring manual invitations for students to access these services. This limitation conflicts with **DG3** which states that ThesisTrack should be reusable for other universities, as custom integrations would be necessary for the specific tools.

**FR5** is marked as partially completed because, at present, the feedback process is only implemented for the thesis proposal. While the proposal phase is critical, particularly for students with limited experience in scientific writing, extending this feedback mechanism to the full thesis and other related submissions could provide additional value.

**FR11** remains incomplete, as the requirement for it was introduced late in the development process. Implementing this feature would have required significant changes to the codebase and database structure, which we could not complete within the remaining timeframe. Additionally, the professor indicated that this feature had a lower priority.

Throughout the development of the platform, we focused on delivering a fully functional system that integrates the essential features requested by the professor and advisors. After implementing each feature, we deployed it to the production environment to enable user testing in real-world scenarios. We added basic unit tests to ensure that future students can continue to work on and extend ThesisTrack. However, due to time constraints, these tests cover only basic functionality and do not provide comprehensive test coverage.

## 5.3 Conclusion

In this bachelor's thesis, we integrated the thesis writing process into ThesisTrack. The platform now centralizes all uploaded files and data, making them easily

accessible and searchable. We enhanced the application flow, allowing students to apply directly for specific topics that align with the research focus. Additionally, we introduced a Gantt chart on the thesis overview page, providing advisors and supervisors with a clear, real-time view of thesis progress across multiple students. This feature enables a more structured approach to managing thesis projects and helps plan advisor availability more effectively. By consolidating these key features into a single platform, we reduced media breaks and minimized manual tasks for advisors and supervisors.

## 5.4 Future Work

While the outcome of this thesis has significantly streamlined the thesis management process for advisors and supervisors, a considerable time investment is still required to guide students throughout the writing process.

Bachelor students, in particular, tend to make recurring mistakes in scientific writing, necessitating similar feedback from advisors across multiple theses [16]. The integration of Large Language Models (LLMs) could present a solution to this challenge. Leveraging LLMs trained specifically on scientific writing could automate the provision of feedback to reduce the number of manual feedback cycles [17]. This system could preemptively provide suggestions to students before advisors review their work, potentially addressing common issues earlier in the process like shown in Figure 22.

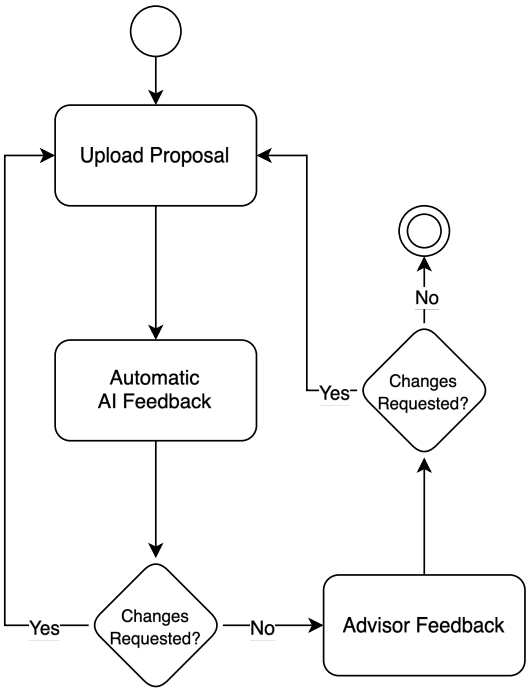


Figure 22: Feedback Cycle for Proposal and Thesis with AI Feedback

To implement this, an LLM could be integrated with the existing ThesisTrack platform as part of the feedback loop for thesis proposals and submissions. The system could analyze textual content for clarity, adherence to academic writing standards, and the correct use of citations. Furthermore, it could provide automated suggestions for improvement based on predefined scientific writing best practices. By integrating natural language processing (NLP) capabilities, the platform could detect patterns of recurring errors and offer tailored feedback, reducing the manual workload for advisors.

In terms of infrastructure, LLMs could be deployed as a microservice within the ThesisTrack ecosystem, allowing real-time feedback to be provided for the students. This microservice would communicate with the main platform via REST APIs, ensuring that feedback is accessible within the user interface. To train and fine-tune the LLMs, anonymized data from previous theses, including common mistakes and advisor feedback, could be used as input. This would create a robust model capable of identifying and addressing typical issues encountered by students during the writing process.

In addition to assisting students, LLMs could also support supervisors by aiding in the grading and fact-checking of theses [18]. The LLM could highlight inconsistencies or factual inaccuracies within the text and suggest areas that require further review. Moreover, by cross-referencing the thesis content with academic databases, it could provide insights into the relevance and accuracy of cited sources.

This work lays the foundation for further advancements in the automation of academic feedback. By incorporating LLM-based feedback mechanisms, the system has the potential to significantly reduce the time advisors spend on routine feedback tasks, allowing them to focus on higher-level academic guidance. In the long term, such developments could reshape the way universities manage theses, making the process more efficient and scalable, particularly in larger universities.

# List of Figures

# List of Tables

# AI Notice

We used Grammarly and ChatGPT in all sections to correct typos, address grammar issues, and enhance wording. The AI tools were utilized exclusively for writing assistance. All text produced with AI was carefully reviewed to ensure accuracy and correctness.

# Bibliography


[1] H. Jaakkola, H. Aramo-Immonen, J. Henno, and J. Mäkelä, "The digitalization push in universities," in *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2016, pp. 1025–1031. doi: 10.1109/MIPRO.2016.7522290.

[2] V. Andraichuk, "PROMPT: Process Model for Managing Project-based Teaching," Munich, Germany, 2023.

[3] A. S. Chukwu and K. Walker, "Towards a deeper understanding of the graduate student and faculty-advisor relationship," *Journal of Contemporary Issues in Education*, vol. 18, no. 1, pp. 61–78, 2023, doi: 10.20355/jcie29505.

[4] O. Popova, N. Gagarina, and D. Karkh, "Digitalization of Educational Processes in Universities: Achievements and Problems," in *Proceedings of the International Scientific Conference "Digitalization of Education: History, Trends and Prospects" (DETP 2020)*, Atlantis Press, 2020, pp. 738–742. doi: 10.2991/assehr.k.200509.131.

[5] H. Brink, S. Packmohr, and K. Vogelsang, "The digitalization of universities from a students' perspective," in *6th international conference on higher education advances*, 2020, pp. 967–974.

[6] N. Hizriani, K. Khairatunnisa, H. Nor, and C. Handrianto, "Online and offline students' presentation in thesis proposal: challenges and solution," *Randwick International of Social Science Journal*, vol. 3, no. 3, pp. 549–560, 2022, doi: 10.47175/rissj.v3i3.501.

[7] E. Saragih, T. T. Zein, and D. M. Sumbayak, "Contextualizing corrective feedback in scientific writing through online learning platforms," *Studies in English Language and Education*, vol. 10, no. 3, pp. 1216–1235, 2023, doi: 10.24815/siele.v10i3.25867.

[8] N. Chidi and O. A. Sylvia, "Determination of factors that contribute to postgraduate students' delay in their thesis/dissertation completion," *The*

*Universal Academic Research Journal*, vol. 2, no. 2, pp. 78–86, 2021, doi: 10.17220/tuara.2020.02.3.

[9] B. Bruegge and A. H. Dutoit, *Object Oriented Software Engineering Using UML, Patterns, and Java*. Prentice Hall, 2009.

[10] M. Almeatani, H. Alotaibi, E. Alasmari, M. Meccawy, and B. Alghamdi, "Thesis supervision mobile system for enhancing student-supervisor communication," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 13, no. 6, p. 4, 2019, doi: 10.3991/ijim.v13i06.9533.

[11] P. P. Kumar, "Effective use of Gantt chart for managing large scale projects," *Cost engineering*, vol. 47, no. 7, p. 14, 2005.

[12] V. Komperla, D. Pratiba, P. Ghuli, and R. Pattar, "React: a detailed survey," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 26, no. 3, p. 1710, 2022, doi: 10.11591/ijeecs.v26.i3.pp1710-1717.

[13] J. Nielsen and R. Molich, "Heuristic evaluation of user interfaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, in CHI '90. Seattle, Washington, USA: Association for Computing Machinery, 1990, pp. 249–256. doi: 10.1145/97243.97281.

[14] M. Raees and M. A. Ashraf, "Effects of refactoring and design patterns on the software source code quality : an empirical assessment," *VAWKUM Transactions on Computer Sciences*, vol. 15, no. 2, p. 59, 2018, doi: 10.21015/vtcs.v15i2.514.

[15] S. C. Hubli and D. R. C. Jaiswal, "Efficient backend development with spring boot: a comprehensive overview," *International Journal for Research in Applied Science and Engineering Technology*, vol. 11, no. 11, pp. 759–765, 2023, doi: 10.22214/ijraset.2023.56617.

[16] A. Prihatini, F. Pangesti, and R. Wuryaningrum, "The relationship among language mindset, corrective feedback preferences, and follow-up strategies of students in writing scientific texts," *Journal of Languages and Language Teaching*, vol. 12, no. 1, p. 104, 2024, doi: 10.33394/jollt.v12i1.9078.

[17] T. Jansen *et al.*, "Empirische arbeit: comparing generative ai and expert feedback to students' writing: insights from student teachers," *Psychologie in Erziehung Und Unterricht*, vol. 71, no. 2, pp. 80–92, 2024, doi: 10.2378/peu2024.art08d.

[18] E. Hoes, S. Altay, and J. P. Bermeo, "Leveraging chatgpt for efficient fact-checking," 2023, doi: 10.31234/osf.io/qnjkf.