



Diplomarbeit

Artificial Intelligence in the Industry and Education Environment

SUBTITLE

Eingereicht von

**Gabriel Mrkonja
Florian Prandstetter
Luna P. I. Schätzle**

Eingereicht bei

**Höhere Technische Bundeslehr- und Versuchsanstalt
Anichstraße**

Abteilung für Wirtschaftsingenieure/Betriebsinformatik

Betreuer

Greinöcker
Egger

Projektpartner

HTL Anichstraße

Innsbruck, April 2025

Abgabevermerk:

Betreuer/in:

Datum:

Gabriel Mrkonja
Florian Prandstetter
Luna P. I. Schätzle

SPERRVERMERK

Auf Wunsch der Firma

HTL Anichstraße

ist die vorliegende Diplomarbeit
für die Dauer von drei / fünf / sieben Jahren
für die öffentliche Nutzung zu sperren.
Veröffentlichung, Vervielfältigung und Einsichtnahme sind ohne
ausdrückliche Genehmigung der Firma *** und der Verfasser
bis zum TT.MM.JJJJ nicht gestattet.

Innsbruck, TT.MM.JJJJ

Verfasser:

Vor- und Zuname

Unterschrift

Vor- und Zuname

Unterschrift

Firma:

Firmenstempel

Gabriel Mrkonja
Florian Prandstetter
Luna P. I. Schätzle

Kurzfassung / Abstract

Eine Kurzfassung ist in deutscher sowie ein Abstract in englischer Sprache mit je maximal einer A4-Seite zu erstellen. Die Beschreibung sollte wesentliche Aspekte des Projektes in technischer Hinsicht beschreiben. Die Zielgruppe der Kurzbeschreibung sind auch Nicht-Techniker! Viele Leser lesen oft nur diese Seite.

Beispiel für ein Abstract (DE und EN)

Die vorliegende Diplomarbeit beschäftigt sich mit verschiedenen Fragen des Lernens Erwachsener – mit dem Ziel, Lernkulturen zu beschreiben, die die Umsetzung des Konzeptes des Lebensbegleitenden Lernens (LBL) unterstützen. Die Lernfähigkeit Erwachsener und die unterschiedlichen Motive, die Erwachsene zum Lernen veranlassen, bilden den Ausgangspunkt dieser Arbeit. Die anschließende Auseinandersetzung mit Selbstgesteuertem Lernen, sowie den daraus resultierenden neuen Rollenzuschreibungen und Aufgaben, die sich bei dieser Form des Lernens für Lernende, Lehrende und Institutionen der Erwachsenenbildung ergeben, soll eine erste Möglichkeit aufzeigen, die zur Umsetzung dieses Konzeptes des LBL beiträgt. Darüber hinaus wird im Zusammenhang mit selbstgesteuerten Lernprozessen Erwachsener die Rolle der Informations- und Kommunikationstechnologien im Rahmen des LBL näher erläutert, denn die Eröffnung neuer Wege zur orts- und zeitunabhängiger Kommunikation und Kooperation der Lernenden untereinander sowie zwischen Lernenden und Lernberatern gewinnt immer mehr an Bedeutung. Abschließend wird das Thema der Sichtbarmachung, Bewertung und Anerkennung des informellen und nicht-formalen Lernens aufgegriffen und deren Beitrag zum LBL erörtert. Diese Arbeit soll

einerseits einen Beitrag zur besseren Verbreitung der verschiedenen Lernkulturen leisten und andererseits einen Reflexionsprozess bei Erwachsenen, die sich lebensbegleitend weiterbilden, in Gang setzen und sie somit dabei unterstützen, eine für sie geeignete Lernkultur zu finden.

This thesis deals with the various questions concerning learning for adults – with the aim to describe learning cultures which support the concept of live-long learning (LLL). The learning ability of adults and the various motives which lead to adults learning are the starting point of this thesis. The following analysis on self-directed learning as well as the resulting new attribution of roles and tasks which arise for learners, trainers and institutions in adult education, shall demonstrate first possibilities to contribute to the implementation of the concept of LLL. In addition, the role of information and communication technologies in the framework of LLL will be closer described in context of self-directed learning processes of adults as the opening of new forms of communication and co-operation independent of location and time between learners as well as between learners and tutors gains more importance. Finally the topic of visualisation, validation and recognition of informal and non-formal learning and their contribution to LLL is discussed.

Gliederung des Abstract in **Thema, Ausgangspunkt, Kurzbeschreibung, Zielsetzung**.

Projektergebnis Allgemeine Beschreibung, was vom Projektziel umgesetzt wurde, in einigen kurzen Sätzen. Optional Hinweise auf Erweiterungen. Gut machen sich in diesem Kapitel auch Bilder vom Gerät (HW) bzw. Screenshots (SW). Liste aller im Pflichtenheft aufgeführten Anforderungen, die nur teilweise oder gar nicht umgesetzt wurden (mit Begründungen).

Erklärung der Eigenständigkeit der Arbeit

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe. Meine Arbeit darf öffentlich zugänglich gemacht werden, wenn kein Sperrvermerk vorliegt.

Ort, Datum

Verfasser 1

Ort, Datum

Verfasser 1

Inhaltsverzeichnis

Abstract	iii
I. Introduction	1
1. Introduction: AI in the Industry and Education Environment	3
1.1. Objectives and Scope	3
1.2. Technical and Economic Context	4
1.3. Team Composition	4
1.4. Detailed Task Description	5
1.4.1. Luna Schaetzle	5
1.4.2. Florian Prandstetter	5
1.4.3. Gabriel Mrkonja	5
1.5. Structure of the Diploma Thesis	5
1.6. The Use of Artificial Intelligence in this Diploma Thesis	6
2. Conceptual Evolution and Rationale	9
2.1. Introduction	9
2.2. Timeline and Milestones	9
2.3. Initial Concept	10
2.4. The Self-Sufficiency Project	11
2.5. Challenges and Limitations of the Self-Sufficiency Project	12
2.6. Transition to the Current Project Concept	13
2.6.1. AI for Education	13
2.6.2. AI Integration in Software Development	14
2.6.3. AI on Edge Devices	14
2.7. Overcoming Challenges in the Current Project Concept	14
2.7.1. Challenges	15
2.7.2. Solutions	15

2.8. Insights and Lessons Learned	16
2.9. Conclusion	17
II. Hardware	19
3. Raspberry PI	21
4. Server	23
5. Server Structure	25
III. Theoretical foundations	27
6. Operating Systems (Florian Prandstetter)	29
6.1. Introduction	29
6.2. What is an Operating System?	29
6.2.1. Types of Operating Systems	30
6.3. Operating Systems used on the server	31
6.3.1. Evaluating different Operating Systems	31
6.3.2. Outcome of the Evaluation	32
6.4. Raspberry Pi Operating System(Gabiriel ???)	33
6.4.1. Raspberry Pi OS	33
7. Used Programming Languages	35
7.1. Python	35
7.1.1. MediaPipe	35
7.1.2. Transformers	35
7.1.3. JSON	35
7.2. HTML, CSS, and JavaScript in Combination with Vue.js	36
7.2.1. HyperText Markup Language (HTML)	36
7.2.2. Cascading Style Sheets (CSS)	37
7.2.3. JavaScript	37
7.2.4. Vue.js	38
7.3. Type Script	39
7.3.1. Axios	39

IV. Implementation of Large Language Models	41
8. Overview and Integration of Large Language Models	43
8.1. Large Language Models (LLMs)	43
8.1.1. Key Characteristics of LLMs	44
8.1.2. Applications of LLMs	44
8.1.3. Examples of Popular LLMs	44
8.1.4. Advantages and Challenges of LLMs	45
8.2. Utilized Large Language Models	45
8.3. Ollama Application Overview	45
8.3.1. Ollama Features	46
8.3.2. Ollama Architecture	46
8.3.3. Ollama Models	47
8.3.4. Ollama API	47
8.3.5. Ollama Integration	47
8.3.6. Benefits and Challenges of Ollama	48
8.4. Evaluation of Models via the Ollama Platform	49
8.4.1. Model Selection Criteria	49
8.4.2. Challenges in Model Testing and Updates	50
8.4.3. Model Selection for the Final Application	50
8.5. Ollama Model Testing and Evaluation	51
8.5.1. Quantitative Evaluation Methods	51
8.5.2. Qualitative Evaluation Methods	51
8.5.3. Models Evaluated During Testing	52
8.5.4. Data Collection	53
8.5.5. Data Preparation	56
8.5.6. Data Processing	59
8.5.7. Test Results and Analysis	66
8.5.8. Qualitative Data Analysis	70
8.5.9. Model Comparison for Different Use Cases and Scenarios	75
8.5.10. Model Selected for the Final Application	78
8.5.11. Model Integration and Deployment	78
8.6. Integration of OpenAI's API	79
8.6.1. Overview of the OpenAI API	79
8.6.2. Data Security and Privacy in Compliance with Austrian and EU Regulations	81

8.6.3. OpenAI API Implementation in Vue.js	82
8.7. Conclusion	86
9. hosted Flask Service	89
9.1. Introduction	89
9.2. Advantages of a Self-hosted Service	89
9.3. Architecture and Service Structure	90
9.3.1. System Architecture	90
9.3.2. Expandability and Modularity	92
9.4. Flask as a Web Framework	92
9.4.1. Core Functionalities of Flask	92
9.4.2. Rationale for Selecting Flask	92
9.5. RESTful Endpoints and Functionalities	93
9.5.1. Endpoint Specifications	93
9.5.2. Code Illustrations	93
9.5.3. Utility Functions	93
9.6. Utilized Libraries	93
9.7. Deployment	94
9.8. Docker	94
9.8.1. Used Docker Images	94
9.8.2. Docker Compose	94
9.9. Scalability and Performance Concerns	95
9.10. Conclusion and Future Work	95
10. Intelligent Student AI Hub: An Integrated Learning Platform	97
10.1. Introduction	97
10.2. System Architecture and Technologies	97
10.2.1. Vue.js	98
10.2.2. Flask API	98
10.2.3. ChatGPT API	98
10.2.4. Firebase for Authentication and Data Storage	99
10.3. Core Functionalities	99
10.4. Authentication and User Profiles	100
10.4.1. Why Firebase?	100
10.4.2. Firebase Authentication Factors	100
10.4.3. Firebase Integration with Vue.js	101
10.4.4. Implementation of Firebase Authentication	102

10.4.5. User Overview and Personalization	107
10.4.6. Outlook for Account Management	108
10.4.7. TSN Integration	108
10.5. Interactive Chatbot for Day-to-Day AI Questions	109
10.6. OpenAI Integration	111
10.6.1. ChatGPT API and Its Limitations	112
10.6.2. User Access to Paid Services	113
10.6.3. Integration of OpenAI's API	113
10.7. Programming Bot for Different Programming Languages	113
10.7.1. Programming Bot Features	114
10.8. Image Recognition Tool	116
10.8.1. Implementation of the Image Recognition Tool	116
10.9. Image to Text Tool	118
10.10. Saved Chats	121
10.10.1. Implementation of the Saved Chats Feature	121
10.11. Structured and Intuitive Navigation	123
10.12. Styling and Theming	124
10.13. Features Excluded from the Final Version	125
10.14. Conclusion	125
10.15. Conclusion	125
11. Visual Studio code extension	127
11.1. Introduction	127
11.2. what is Visual Studio Code	127
11.3. Development	127
11.4. Core Functionalities	128
11.4.1. Chatbot Integration	128
11.4.2. Code completion	128
V. Implementation of Object Detection	129
12. Introduction to Object Detection	131
13. Implementation of Object Detection	133

VI. Evaluations	135
14. Artificial Intelligence in Economics	137
14.1. Introduction	137
15. Open source evaluation on Economics	139
15.1. Introduction	139
15.1.1. What is Open Source?	139
15.1.2. Advantages of Open Source	140
15.1.3. Why Do People Use Open Source?	141
15.2. What is and isn't Open Source?	141
15.2.1. Definition and Guiding Principles	141
15.2.2. Misconceptions About Open Source	142
15.3. Challenges and Disadvantages of Open Source Software	143
15.3.1. Disadvantages of Open Source Software	143
15.3.2. Technical Challenges	144
15.3.3. Economic Challenges	144
15.3.4. Social Challenges	145
15.3.5. Legal Challenges	145
15.4. Potential Risks and Security Concerns	146
15.4.1. Common Risks Associated with Open Source Software	146
15.4.2. Specific Security Concerns in Open Source Environments	147
15.5. The Role of Open Source in Economics	148
15.5.1. Driving Innovation and Shaping Market Dynamics . .	148
15.5.2. Supporting Startups and small Enterprises	149
15.5.3. Facilitating Cross-Industry Collaboration and Open Innovation	149
15.6. Open Source in Key Industries	150
15.6.1. Examples of Open Source Success Stories	151
15.7. Revenue Models in Open Source	152
15.7.1. Common Business Models	152
15.8. Open Source Support in Austria	153
15.9. Open Source in Practice: A Personal Experience	153
15.10. Licence Model of the Diploma Thesis	154
15.10.1. GNU General Public License (GPL) Version 3	154
15.11. Conclusion	155

16. Economic aspect of Operating Systems	157
16.1. Introduction	157
 VII. Conclusion	 159
17. Conclusion	161
17.1. Key Findings	161
17.2. Implications and Recommendations	161
17.3. Limitations and Challenges	161
 18. Outlook	 163
18.1. Future Trends in AI	163
18.1.1. AI in Industry	164
18.1.2. AI in Education	164
18.1.3. Challenges and Opportunities	164
18.2. Further Development of the Flask Server	165
18.3. Further Development of the Student AI Hub	165
18.4. Open Source in Future Projects	166
18.5. Challenges and Opportunities	167
18.6. Recommendations for Further Research	167
18.7. Concluding Remarks	167
 A. Time Protocol	 169
A.1. Luna P. I. Schätzle	169
A.2. Florian Prandstetter	177
 Literaturverzeichnis	 189

Teil I.

Introduction

1. Introduction: AI in the Industry and Education Environment

Author: Luna P. I. Schätzle, Florian Prandstetter, Gabriel Mrkonja

In the context of ongoing digital transformation, the use of artificial intelligence (AI) is becoming increasingly critical across various industries. AI technologies are employed to automate processes, optimize production, and enhance quality, while in the education sector, they support learning processes, personalize educational content, and provide targeted feedback to students. Nonetheless, the implementation of AI solutions in both industry and education remains challenging due to the inherent complexity of these technologies, a shortage of specialized expertise, and the high costs associated with their development and deployment.

1.1. Objectives and Scope

The primary objective of this diploma thesis is to develop AI-driven solutions that address specific challenges in both industrial and educational settings.

The scope of the project encompasses the following tasks:

- **Task 1:** Establish a server (API) to host and run AI models.
- **Task 2:** Develop a robust backend for the AI Hub and the Code Extension.
- **Task 3:** Research and identify the most suitable AI models for various tasks, and integrate them into the backend of the AI Hub and the Code Extension.

- **Task 4:** Create an AI Hub for students to access a variety of AI tools and resources, including chatbots, image recognition, and natural language processing (NLP) models.
- **Task 5:** Implement an AI solution for coding assistance, available as an extension within a code editor.
- **Task 6:** Develop an AI-driven service on an edge device (e.g., Raspberry Pi) to explore the potential of AI in industrial applications with minimal resources.
- **Task 7:** Compile comprehensive documentation of the project outcomes, covering the conceptual framework, theoretical foundations, practical implementation, primary and alternative solution approaches, results, and their interpretation.

1.2. Technical and Economic Context

The project is embedded within the current wave of digital transformation, which has significantly increased the importance of AI technologies in both industrial and educational environments. By leveraging state-of-the-art AI models and tools—such as chatbots, image recognition, and natural language processing—the project aims to develop innovative solutions tailored to specific challenges in these fields.

Furthermore, the project will investigate the feasibility of deploying AI-driven services on edge devices, like Raspberry Pi, to explore cost-effective and resource-efficient industrial applications. The successful execution of the project relies on the team's combined expertise in software development, AI technologies, and project management.

1.3. Team Composition

The project team consists of three members:

- Luna Schaetzle (Project lead)
- Florian Prandstetter

Gabriel Mrkonja

Florian Prandstetter

Luna P. I. Schätzle

- Gabriel Mrkonja

Each team member is assigned specific roles and responsibilities, which are detailed in the following sections.

1.4. Detailed Task Description

1.4.1. Luna Schaetzle

Luna Schaetzle is responsible for a wide range of tasks and duties, including:

- **Team Leadership and Project Management:** Serving as the team leader, project manager, and project lead, she oversees the planning, coordination, and monitoring of all project activities.
- **Documentation and Repository Management:** Ensuring comprehensive documentation of project outcomes and managing the GitHub repository.
- **Backend Development:** Leading the development of the backend for both the AI Hub and the Code Extension.
- **Website Development:** Directing the design and implementation of the AI Hub's website.
- **Open Source Evaluation:** Assessing the economic implications and potential of open-source technologies.

1.4.2. Florian Prandstetter

1.4.3. Gabriel Mrkonja

1.5. Structure of the Diploma Thesis

To enhance readability and clarity, the diploma thesis is organized into distinct parts that mirror the various objectives of the project:

- **Part 1: Introduction and Project Overview** — Provides an introduction to the project, outlining its objectives, scope, and technical as well as economic context.
- **Part 2: Hardware Components** — Describes the hardware utilized in the project, including the Raspberry Pi and the server.
- **Part 3: Theoretical Foundations** — Covers the theoretical background, detailing the operating systems and programming languages employed.
- **Part 4: Implementation of Large Language Models** — Details the integration of large language models, encompassing the hosted Flask service, the Student AI Hub, and the VS Code extension.
- **Part 5: Implementation of Object Detection** — Focuses on object detection techniques, presenting both an introduction and a step-by-step account of the implementation process.
- **Part 6: Economic Evaluations** — Examines the economic aspects of AI in education and industry, including evaluations of open-source approaches and their financial implications.
- **Part 7: Conclusion and Proof of Work** — Summarizes the key findings and outcomes of the project while providing an outlook on future developments.

Each part is further divided into chapters, sections, and subsections, ensuring a structured and comprehensive overview of the project.¹

1.6. The Use of Artificial Intelligence in this Diploma Thesis

Artificial intelligence was employed throughout the creation of this diploma thesis to support various aspects of the writing process. AI tools were utilized for text generation, data analysis, and to provide constructive feedback on the content. The following AI tools were integrated into the project:

- ChatGPT

¹Under every chapter, the author of the chapter is listed.

- GitHub Copilot
- Deepseek
- Ollama
- Claude

2. Conceptual Evolution and Rationale

Authors: Luna P. I. Schaetzle

This chapter provides a comprehensive analysis of the project's evolution. It delineates the progression from an initial theoretical proposal, through the Self-Sufficiency Raspberry Pi Project, to the final project concept. The discussion elucidates the key motivations, challenges, and pivotal decisions that have ultimately shaped the design.

2.1. Introduction

In the context of this Diploma Thesis, the project has undergone several conceptual transformations aimed at refining its scope, objectives, and implementation strategy. Documenting this evolution is essential for a holistic understanding of the development process, as it highlights the iterative nature of design and the interplay between feasibility, scalability, and the initial research goals. Key considerations included evaluating the project's feasibility, ensuring scalability, and aligning with the overarching research objectives. Additionally, a critical examination of technical challenges encountered during implementation informed subsequent refinements.

2.2. Timeline and Milestones

To provide a clear overview of the project's evolution, a timeline outlining the major milestones, decision points, and revisions is presented in the

following chart.

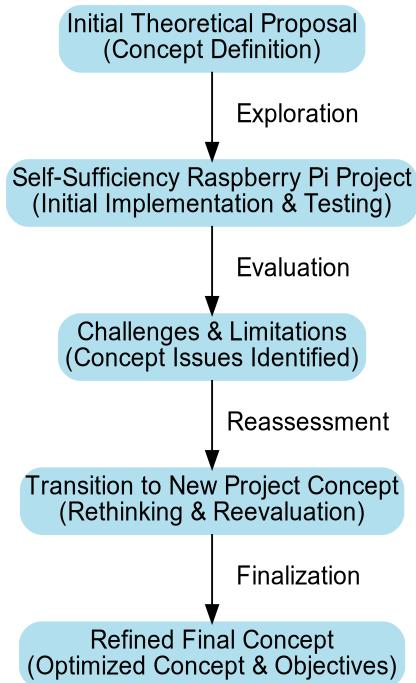


Abbildung 2.1.: Gantt Chart of the Project Evolution

This timeline visually summarizes the progression of ideas and the key changes made over time, offering a concise reference to the project's developmental history.

2.3. Initial Concept

The initial concept for this Diploma Thesis was to explore various methodologies for leveraging artificial intelligence (AI) across diverse application domains. The primary objective was to develop a comprehensive framework for evaluating the performance of AI models in a range of tasks and use cases. This framework was intended to include both quantitative metrics

and qualitative assessments, facilitated in part by the use of structured questionnaires to capture user experiences and application-specific requirements. However, early analysis revealed that the scope of this concept was overly broad. The project team quickly recognized that the lack of a focused research question, as well as the ambiguous integration of the various components, would make it challenging to implement the project within the available time and resources.

2.4. The Self-Sufficiency Project

In response to these challenges, the project team refined its approach by narrowing the scope to a more tangible and achievable idea—the Self-Sufficiency Project. This project was designed around the development of a self-contained system based on a Raspberry Pi, capable of autonomously executing a variety of tasks. The core idea was to demonstrate a practical application of AI by minimizing reliance on external APIs and services, thereby increasing system robustness and independence. Additionally, the design emphasized portability, with the entire system housed in a compact enclosure to facilitate deployment in diverse environments. Figure 2.2 illustrates the conceptual framework of the Self-Sufficiency Raspberry Pi Project.



Abbildung 2.2.: Conceptual Illustration of the Self-Sufficiency Raspberry Pi Project

?

2.5. Challenges and Limitations of the Self-Sufficiency Project

During the development of the Self-Sufficiency Project, several significant challenges and limitations emerged that prompted a critical reassessment of the project's direction. These challenges included:

- **Complexity of Implementation:** The integration of heterogeneous AI models with multiple hardware components introduced considerable technical difficulties, necessitating extensive iterative development and rigorous testing.
- **Scalability Constraints:** The self-sufficiency paradigm, while beneficial for system independence, inherently restricted scalability and interoperability with external systems, thereby limiting future expansion.

- **Resource Limitations:** The ambitious scope of the project, combined with constrained time and technical expertise, rendered the initial plan impractical.
- **Ambiguity in Objectives:** The absence of clearly defined, measurable goals complicated the assessment of progress and the determination of success criteria.
- **Integration Challenges:** The complexity of harmonizing various AI models, hardware interfaces, and supporting software systems proved to be more formidable than initially anticipated.
- **Time Constraints:** Preliminary evaluations indicated that the project's timeline was insufficient to address the technical and logistical challenges identified.

In light of these challenges, the project team concluded that a more narrowly defined and feasible approach was necessary. Consequently, the focus shifted towards a new concept that emphasizes AI applications and their specific use cases.

2.6. Transition to the Current Project Concept

After reevaluating the limitations of the initial approaches, the project team restructured the initiative into a more focused framework, subdividing it into three distinct components. This revised approach not only addresses the previously identified constraints but also aligns more closely with the overarching research objectives. The current project concept is organized into the following three components:

2.6.1. AI for Education

This component explores the integration of AI technologies within educational settings. The primary goal is to develop an AI-powered web application that facilitates interactive and personalized learning experiences, particularly for IT students. By leveraging adaptive learning algorithms, the system

aims to enhance comprehension and engagement, thereby contributing to more effective educational practices.

2.6.2. AI Integration in Software Development

Focusing on the software development domain, this component investigates how AI can assist developers in writing and debugging code. The objective is to develop an AI model that integrates seamlessly into the development workflow, for example, via a Visual Studio Code extension. This extension is designed to provide real-time code analysis, suggestions, and bug detection, ultimately improving code quality and development efficiency.

2.6.3. AI on Edge Devices

The third component examines the deployment of AI models on resource-constrained edge devices, using the Raspberry Pi as a primary test platform. The goal is to implement and optimize real-time object detection and other AI-driven tasks on the Raspberry Pi, thereby demonstrating the feasibility of running sophisticated AI applications on low-power hardware. This component is crucial for understanding the limitations and opportunities associated with edge computing in AI applications.

2.7. Overcoming Challenges in the Current Project Concept

The evolution of any project is accompanied by a series of challenges. In transitioning to the current project concept, the team identified and addressed several critical obstacles to ensure a robust and effective implementation of the new framework.

2.7.1. Challenges

The primary challenges encountered during this transition included:

- **Technical Complexity:** Integrating advanced AI models with diverse platforms—such as web applications, code editors, and edge devices—required a deep understanding of multiple technologies and frameworks.
- **Resource Limitations:** The expansive scope of the project necessitated a judicious allocation of time, technical expertise, and hardware resources.
- **Scalability Issues:** Balancing the goal of system independence with the need for scalability and interoperability posed a significant challenge, particularly when deploying on resource-constrained edge devices.
- **Temporal Constraints:** The limited project timeline imposed restrictions on the depth and breadth of research and development activities.
- **Integration Complexity:** Harmonizing the heterogeneous components of the project—including AI models, hardware interfaces, and software systems—proved more intricate than initially anticipated.
- **Ambiguity in Evaluation Criteria:** The absence of clearly defined, quantifiable success metrics complicated the objective assessment of project progress and outcomes.
- **Server Resource Limitations:** Particularly within the AI for Education component, limitations in server capacity emerged as a significant impediment to achieving desired scalability.

2.7.2. Solutions

In response to these challenges, the project team adopted a series of strategic measures:

- **Modularization:** Dividing the project into three distinct components allowed for a more focused development approach, effectively managing complexity and scalability concerns.

Gabriel Mrkonja
Florian Prandstetter
Luna P. I. Schätzle

- **Scope Refinement:** By narrowing the project's scope, the team could allocate resources more efficiently, thereby establishing a realistic timeline and achievable milestones.
- **Iterative Development:** Implementing an iterative development methodology enabled the incremental resolution of technical challenges, leading to a more robust and scalable final implementation.
- **Optimization of Server Resources:** Specific efforts were made to streamline the AI for Education component, thereby reducing server resource consumption and mitigating scalability issues.
- **Establishment of Clear Evaluation Metrics:** The development and adoption of explicit, measurable evaluation criteria for each project component facilitated systematic progress tracking and performance assessment.
- **Utilization of AI for Process Optimization:** Leveraging AI technologies to automate routine tasks and optimize overall system performance further enhanced efficiency and scalability.

2.8. Insights and Lessons Learned

Throughout the project's evolution, several key insights were gained, which have significantly informed the current project concept and its implementation strategy. These insights include:

- **Adaptive Planning:** The capacity to adapt and refine project parameters in response to emerging challenges is critical for successful project execution.
- **Incremental Development:** An iterative development approach facilitates steady progress and enables the timely resolution of technical issues, contributing to continuous improvement of the project design.
- **Strategic Resource Allocation:** Effective management of available resources—time, expertise, and hardware—is essential for mitigating complexity and ensuring the project's successful completion.
- **Balancing Independence and Scalability:** Striking the right balance between system autonomy and scalability is paramount for ensuring long-term viability and facilitating future integrations.

- **Clear Metrics for Evaluation:** Establishing definitive, quantifiable success metrics is vital for tracking progress, objectively assessing outcomes, and guiding subsequent development efforts.
- **Leveraging AI for Optimization:** The strategic use of AI to automate processes and optimize system performance can markedly enhance overall efficiency and scalability.
- **Importance of Targeted Optimization Strategies:** Focused measures—such as reducing server resource usage and refining AI model implementations—are crucial for addressing scalability concerns and boosting system performance.

2.9. Conclusion

The transition to the current project concept was driven by a deliberate process of modularization, scope refinement, and iterative development. By segmenting the initiative into three distinct components AI for Education, AI Integration in Software Development, and AI on Edge Devices the project team was able to align the research objectives with practical constraints and emerging technological opportunities. The strategic measures implemented to overcome obstacles, along with the insights and lessons learned throughout this evolution, have established a robust foundation for future development.

Overall, the iterative nature of this process underscores the importance of adaptive planning, rigorous evaluation, and targeted optimization strategies in the successful execution of complex projects. The refined concept not only addresses the initial challenges but also sets a clear pathway for the subsequent phases of the Diploma Thesis, paving the way for a detailed exploration of each component in the chapters that follow.

Teil II.

Hardware

3. Raspberry PI

4. Server

5. Server Structure

Teil III.

Theoretical foundations

6. Operating Systems (Florian Prandstetter)

6.1. Introduction

An important part of building an AI Service is to host an API endpoint for all applications to use. To improve the quality of the service, having a flawless and seamless connection with a fast response time is necessary. Having a good foundation to build upon is thus unavoidable to build a strong and secure API endpoint.

This chapter is going to briefly explain what an Operating System (OS) is and its importance. It will also evaluate different options suitable for hosting the AI Server.

6.2. What is an Operating System?

An Operating System (OS) acts as an intermediary between the user and the computer hardware, ensuring smooth and efficient operation of the system. It is responsible for managing hardware components such as the CPU, memory, storage devices, and input/output peripherals, allowing users and applications to interact with them seamlessly.

The OS provides essential functions like process management, ensuring that multiple applications can run without conflicts. It also handles memory management by allocating and deallocating memory to different processes. It also prevents unauthorized access to the storage. Additionally, the OS

handles file system management, enabling organized and efficient data storage.

Another key role of the OS is device management, where it communicates with hardware components using device drivers.

Furthermore, Operating Systems provide networking capabilities, enabling connection to local and global networks. They try to ensure a stable and secure connection between devices.

?

6.2.1. Types of Operating Systems

The type of Operating System used is determined by the needs of the user. Different types of Operating Systems use different system architectures to provide varying benefits. Factors like scalability, performance, and reliability have to be considered in choosing a suitable OS.

- **Batch Operating Systems** are designed to handle a large number of processes. They are used to process large amounts of data and to run complex calculations.
 - **Advantages** Multiple users can share the batch system, and it is easy to manage large amounts of traffic.
 - **Disadvantages** The CPU isn't used efficiently. Also, the response time can be slow due to processes being processed one by one.
- **Multi Programming/Time Sharing Operating System** is used to utilize the available resources as efficiently as possible. They allow using multiple programs at the same time, which allows multiple users to share the system. They are often used for servers and also have been used by the development team in the project.
 - **Advantages** Allows multiple users to share the resources. Also, it helps avoid duplicated software.
 - **Disadvantages** There can be problems with reliable data communication.

- **Real-Time Operating System** is used when a very short response time is needed. They are often used on microcontrollers like the ESP32.
 - **Advantages** They utilize the device to its maximum. Fast and reliable memory allocation. They usually are error-free.
 - **Disadvantages** There can only be a limited number of tasks. They also require complex algorithms that can be resource-heavy.

?

6.3. Operating Systems used on the server

To host the required AI service, choosing a suitable Operating System is a crucial part to guarantee a satisfying performance. For the project, multiple users need to access the server simultaneously, so the development team decided to use a Multi Programming OS.

6.3.1. Evaluating different Operating Systems

The preferred OS for an AI Server is Linux distributions like Ubuntu or Red Hat. Their open-source nature makes it very easy to customize and integrate AI frameworks.

The Operating Systems that were taken into consideration are Ubuntu Server, Debian, and Red Hat Enterprise.

?

Debian

Debian is a good choice for servers due to its stability and security. It is widely used since it has a large package repository and offers long-term support. It also has a large community that provides good documentation and resources. The distribution is well-suited for hosting AI services and provides a good foundation for building and deploying applications.

?

Red Hat Enterprise

Red Hat Enterprise is a commercial Linux distribution. It is the most popular distribution for servers. It comes with a lot of features that are useful for hosting AI services. It is a good choice for companies that need a stable and secure server. It also provides long-term support and has good documentation. The main downside is that it is a commercial distribution and requires a subscription.

?

Ubuntu Server

Ubuntu Server is a popular choice for hosting AI services. It is based on Debian and has a large package repository. It is easy to use and has a large community that provides good documentation and support. It also provides NVIDIA drivers and CUDA support, which is useful for running AI applications that require GPU acceleration.

?

6.3.2. Outcome of the Evaluation

After evaluating the different Operating Systems, the development team decided to use Ubuntu Server for hosting the AI Service. The main reason for the decision was the experience of the team with Ubuntu and the easy integration of AI frameworks like TensorFlow and PyTorch. Also, the large community and the good documentation were important factors in the decision. The NVIDIA drivers were also an important factor that played a role in the decision.

6.4. Raspberry Pi Operating System(Gabriel ???)

6.4.1. Raspberry Pi OS

7. Used Programming Languages

7.1. Python

To enhance readability and comprehension, the most widely used libraries are explained in the following sections.

7.1.1. MediaPipe

7.1.2. Transformers

The Transformers library, developed by Hugging Face, is an open-source Python package that provides implementations of state-of-the-art transformer models. It supports multiple deep learning frameworks, including PyTorch, TensorFlow, and JAX, facilitating seamless integration across various platforms. The library offers access to a vast array of pre-trained models tailored for tasks such as natural language processing, computer vision, and audio analysis. Utilizing these pre-trained models enables researchers and practitioners to achieve high performance in tasks like text classification, named entity recognition, and question answering without the necessity of training models from scratch, thereby conserving computational resources and time ?.

7.1.3. JSON

JavaScript Object Notation (JSON) is a lightweight, text-based data interchange format that is easy for humans to read and write, and straightfor-

ward for machines to parse and generate. In Python, the built-in `json` module provides functionalities to serialize Python objects into JSON-formatted strings and deserialize JSON strings back into Python objects. This module supports the conversion of fundamental Python data types, such as dictionaries, lists, strings, integers, and floats, into their corresponding JSON representations. The `json` module is indispensable for tasks involving data exchange between Python applications and external systems, particularly in web development contexts where JSON is a prevalent format for client-server communication ?.

Gabriels Part

7.2. HTML, CSS, and JavaScript in Combination with Vue.js

In this project, the languages HTML, CSS, and JavaScript were used in conjunction with Vue.js to create an interactive and dynamic user experience. For more information about Vue.js, see Chapter ??, Section "Vue.js."

7.2.1. HyperText Markup Language (HTML)

HyperText Markup Language (HTML) is the standard markup language for creating and structuring content on the web. It serves as the backbone of web pages by organizing content through elements represented by tags. Key features of HTML include:

- **Structure Definition:** Tags such as `<html>`, `<head>`, and `<body>` define the structural hierarchy of a web page.
- **Content Organization:** Elements like headings, paragraphs, links, images, and tables provide a clear and user-friendly layout.
- **Web Compatibility:** HTML is universally supported, ensuring seamless integration across browsers and devices.

As one of the core technologies of the World Wide Web, alongside CSS and JavaScript, HTML enables the creation of interactive and visually appealing websites. Its simplicity and adaptability make it an essential tool for web development.

?

7.2.2. Cascading Style Sheets (CSS)

Cascading Style Sheets (CSS) is a style sheet language designed to control the visual presentation of web pages. CSS enhances the user experience by allowing developers to define the look and feel of a website. Key functionalities of CSS include:

- **Design Customization:** Control over layout, colors, fonts, and spacing for a cohesive visual identity.
- **Responsive Design:** Ensures consistent and optimized appearance across different devices and screen sizes.
- **Cascading Rules:** Allows styles to be applied at element, class, or global levels, offering flexibility in design.

As a foundational technology of the web, CSS plays a vital role in creating modern, responsive, and aesthetically pleasing websites.

?

7.2.3. JavaScript

JavaScript is a high-level programming language used to add interactivity and dynamic content to web pages. It works seamlessly alongside HTML and CSS to create rich and engaging user experiences. Key features of JavaScript include:

- **Dynamic Content:** Enables animations, form validation, and real-time updates.

- **Client and Server-Side Usage:** Runs in web browsers via JavaScript engines and supports server-side applications through platforms like Node.js.
- **Extensive Ecosystem:** Offers libraries, frameworks, and tools for building feature-rich web applications.

JavaScript's flexibility and versatility have established it as a cornerstone of web development, making it essential for developing interactive and responsive applications.

?

7.2.4. Vue.js

Vue.js is a progressive, open-source JavaScript framework that provides a robust foundation for the development of sophisticated user interfaces and single-page applications. By leveraging standard web technologies—namely HTML, CSS, and JavaScript—Vue.js streamlines the development process through its intuitive and adaptable architecture. Its key attributes include:

- **Component-Based Architecture:** Facilitates modular development by enabling the creation of reusable components that encapsulate both data and functionality.
- **Reactivity System:** Automatically synchronizes the Document Object Model (DOM) with underlying data changes, ensuring a dynamic and responsive user interface.
- **Directives and Template Syntax:** Employs a declarative approach to simplify data binding and event handling.
- **Vue CLI:** Offers a comprehensive command-line interface for project scaffolding, build configuration, and plugin management.
- **Vue Router and Vuex:** Integrates official libraries for client-side routing and state management, thereby enhancing scalability and maintainability.
- **Vibrant Community Support:** Benefits from an active ecosystem that provides extensive documentation, tutorials, and a wide range of third-party plugins.

- **Performance Optimization:** Utilizes a Virtual DOM and efficient rendering algorithms to optimize performance, even in complex applications.
- **Enhanced Developer Experience:** Incorporates developer-centric tools such as Vue Devtools and an intuitive CLI to boost productivity and simplify debugging.

In summary, Vue.js equips developers with the necessary tools to efficiently construct interactive and feature-rich web applications, making it a widely adopted framework in contemporary web development ?.

7.3. Type Script

TypeScript is a superset of JavaScript that adds static type definitions to the language. It is designed for the development of large-scale applications and transcompiles to JavaScript. JavaScript is not very strict when it comes to types. This can lead to issues during development. Typescript adds a type system to JavaScript, which can help to catch errors early in the development process.

?

7.3.1. Axios

Axios is the key library used to make HTTP request from the frontend to the backend. It is a promise-based HTTP client for the browser and Node.js. It is used to make asynchronous requests to a server, and it returns a promise that resolves with the response data.

?

Flos Part

Teil IV.

Implementation of Large Language Models

8. Overview and Integration of Large Language Models

Author: Luna P. I. Schätzle

For the Diploma thesis, there are many different AI models that are in use. There are different Types of AI models, such as:

- LLMs (Large Language Models)
- Defusion Models (Models that are used to create images)
- Object Detection Models (Models that are used to detect objects in images)
- Face Recognition Models (Models that are used to recognize faces in images)

In the following chapters, the different Types and the used models will be explained in more detail.

8.1. Large Language Models (LLMs)

Large Language Models (LLMs) represent a significant advancement in artificial intelligence, enabling machines to process and generate natural language. LLMs are built on the concept of deep learning, utilizing neural networks with billions of parameters to understand and generate text in a contextually accurate and coherent manner. These models are trained on vast datasets encompassing diverse topics, allowing them to handle a wide range of tasks, such as translation, summarization, content generation, and conversational AI.

8.1.1. Key Characteristics of LLMs

- **Scale and Complexity:** LLMs are distinguished by their immense size, often containing billions of parameters, enabling them to capture intricate patterns in language.
- **Transfer Learning:** These models benefit from pretraining on large datasets, followed by fine-tuning for specific tasks, making them highly versatile.
- **Contextual Understanding:** LLMs excel at understanding context, which allows them to generate coherent and contextually appropriate responses.
- **Multilingual Capabilities:** Many LLMs are trained on datasets in multiple languages, enabling them to process and generate text in various languages.

8.1.2. Applications of LLMs

- Text summarization and paraphrasing.
- Question answering and information retrieval.
- Conversational agents and chatbots.
- Code generation and debugging assistance.
- Creative writing, including story and poetry generation.

8.1.3. Examples of Popular LLMs

- **GPT Models:** Developed by OpenAI, these models include GPT-3, GPT-4, and ChatGPT, known for their state-of-the-art performance in text generation and comprehension.
- **BERT (Bidirectional Encoder Representations from Transformers):** Developed by Google, BERT focuses on understanding context by analyzing text bidirectionally.
- **LLama Models:** Created by Meta, these models are designed for efficient natural language understanding and generation.
- **Mistral Models:** Aimed at specialized tasks with high precision and multilingual capabilities.

8.1.4. Advantages and Challenges of LLMs

Advantages:

- High accuracy in generating and understanding text.
- Adaptability to a variety of domains and languages.
- Ability to process complex and context-rich queries.

Challenges:

- High computational and memory requirements.
- Potential biases due to the training data.
- Difficulty in maintaining factual accuracy in generated content.

?

8.2. Utilized Large Language Models

In the context of this diploma thesis, various free and commercial large language models (LLMs) were evaluated to determine their suitability for integration. Leveraging the Ollama application, we were able to test and compare several LLMs. Additionally, we explored different ChatGPT models available through the OpenAI API. OpenAI offers a range of models that vary in terms of size and complexity, with more advanced models incurring higher usage costs.

??

8.3. Ollama Application Overview

The Ollama application is an advanced, locally hosted platform designed to provide a versatile environment for deploying and interacting with a wide array of artificial intelligence models. It offers a comprehensive solution for both text and image processing tasks, facilitating the integration, fine-tuning, and management of models in a secure and scalable manner.

8.3.1. Ollama Features

Ollama is distinguished by several key features that enhance its functionality and usability:

- **Multi-Model Support:** The platform supports a variety of AI models, each optimized for specific tasks such as natural language processing and image analysis.
- **Local API Hosting:** The API is hosted on a local server, ensuring rapid and secure processing of requests while maintaining full control over data.
- **Image Processing Capabilities:** In addition to textual data, certain models within Ollama are capable of processing images. These models can analyze visual content, thereby extending the application's utility.
- **Model Customization and Fine-Tuning:** Users can fine-tune existing models to suit their specific needs. Once customized, these models can be re-uploaded to the Ollama server, allowing for continuous improvement and adaptation.

8.3.2. Ollama Architecture

The architecture of Ollama is modular and designed to support high performance and scalability:

1. **Model Management Layer:** This layer is responsible for deploying, fine-tuning, and updating the various AI models. It provides a structured approach to manage model versions and customizations.
2. **API Service Layer:** Hosted locally, this layer facilitates communication between client applications and the AI models. It exposes endpoints for both text and image processing, ensuring secure and efficient data exchange.
3. **Integration Interfaces:** These interfaces enable seamless connectivity with external services and applications, promoting interoperability and flexibility in diverse operational environments.

This layered design supports efficient resource management while enabling rapid response times and scalability to handle increasing user demands.

8.3.3. Ollama Models

Ollama provides a diverse selection of models, each tailored to specific application domains:

- **Text Generation Models:** Optimized for tasks such as dialogue generation, summarization, and other natural language processing applications.
- **Image Analysis Models:** Developed for image recognition, generation, and related tasks.

Furthermore, the platform allows users to fine-tune these models based on their particular requirements. Customized models can be re-uploaded to the server, enabling a continuous cycle of refinement and performance enhancement.

8.3.4. Ollama API

The Ollama API is the primary interface through which client applications interact with the hosted models. It provides robust and secure endpoints for processing both textual and visual data:

- **Data Exchange:** The API facilitates structured data exchange between client applications and the backend, ensuring that requests and responses are handled efficiently.
- **Security and Performance:** Designed with stringent security protocols, the API ensures that all interactions are encrypted and managed in a way that maximizes performance while minimizing latency.
- **Extensibility:** The API's modular design allows for the easy addition of new endpoints and functionalities as the platform evolves.

8.3.5. Ollama Integration

Integrating the Ollama API into external applications is straightforward. For instance, a Python-based client can send HTTP requests to the API to perform tasks such as generating text or processing images. This section

is further elaborated in the chapter dedicated to the hosted Flask Service, where detailed examples and implementation guidelines are provided. In brief, the integration involves:

- Establishing a connection to the local API endpoint.
- Sending appropriately formatted requests (e.g., JSON payloads) that include user inputs.
- Handling responses from the API, which may include generated text or URLs to processed images.

8.3.6. Benefits and Challenges of Ollama

Ollama presents several benefits:

- **Ease of Use:** The platform is user-friendly, with intuitive APIs that simplify deployment and integration.
- **Versatility:** A wide array of models enables the application of Ollama to diverse tasks, from natural language processing to image analysis.
- **Multilingual Support:** The models are capable of processing multiple languages, thereby broadening the scope of potential applications.
- **Customization:** Users can fine-tune models to meet specific needs and update them on the server, ensuring tailored performance.

However, several challenges must be addressed:

- **Performance Limitations:** Larger models may experience slower response times due to higher computational demands.
- **API Request Management:** Ensuring that the API can handle a high volume of requests efficiently requires robust load balancing and error handling mechanisms.
- **Model Management Complexity:** Coordinating updates, fine-tuning, and deployment of multiple models demands an effective management strategy.
- **Concurrency:** Managing simultaneous user requests, as discussed in the chapter on the hosted Flask Service, is critical to maintaining system performance under high load.

In summary, while Ollama offers a flexible and powerful platform for AI model deployment and interaction, addressing its inherent challenges is crucial for optimizing performance and ensuring long-term scalability in practical applications.

?

8.4. Evaluation of Models via the Ollama Platform

In this project, we conducted an evaluation of various models accessible through the Ollama application, which are available for download from the Ollama server.

Given that Ollama operates locally, it was imperative to select models that align with specific criteria to ensure optimal performance. Consequently, we assessed models of diverse sizes and complexities to determine their suitability for local deployment. This evaluation encompassed both the efficacy and efficiency of the models within a local environment.

?

8.4.1. Model Selection Criteria

The selection of models was guided by the following criteria:

- **Model Size:** The model must be capable of running on the server without exceeding available memory capacity.
- **Performance Speed:** The response time of the model, i.e., how quickly it can generate output.
- **Complexity:** The model's ability to handle complex prompts and generate coherent, contextually accurate text.
- **Accuracy:** The overall precision of the model's responses, particularly in terms of factual correctness and linguistic quality.
- **Language Support:** The model's proficiency in understanding and generating text in multiple languages, particularly English and German.

- **User Experience:** The model's overall usability and user-friendliness, including ease of integration and customization.

There is often a trade-off between these criteria. Larger models tend to exhibit higher accuracy and greater contextual understanding but are generally slower and require more computational resources.

8.4.2. Challenges in Model Testing and Updates

One significant challenge lies in the rapid development and frequent release of new models, which complicates the process of continuous integration and comprehensive evaluation of recent advancements. Regular testing and updates are imperative to ensure the incorporation of state-of-the-art models while maintaining system reliability and relevance.

Another challenge involves achieving an optimal balance between performance, accuracy, and resource efficiency, ensuring that the chosen model meets the application's functional requirements without compromising the overall user experience.

During the evaluation process, we encountered several obstacles. For instance, identifying standardized questions that could be uniformly answered by all models proved challenging. Some smaller models demonstrated limitations in addressing certain questions comprehensively. Additionally, specialized models, while excelling in niche areas, often lacked the ability to provide detailed answers across a broader range of topics.

For the final evaluation phase, we employed a diverse question set consisting of self-crafted questions, publicly available questions from online sources, and queries generated by ChatGPT-4 to ensure a comprehensive assessment covering a wide spectrum of queries.

8.4.3. Model Selection for the Final Application

In the final implementation, users are provided with a curated list of recommended models from which they can select their preferred option. This list

was carefully compiled based on our comprehensive testing and reflects the models that demonstrated the best balance between performance, accuracy, and resource efficiency.

For the production version of the application, this list must be updated periodically to include newly released models and maintain optimal performance.

8.5. Ollama Model Testing and Evaluation

Based on the following criteria, we conducted an extensive evaluation of the upcomming models:

8.5.1. Quantitative Evaluation Methods

For the quantitative evaluation, we focused on key performance metrics to assess the efficiency and reliability of each model:

- **Response Time:** The time taken by the model to generate a response after receiving input.
- **CPU Usage:** The percentage of CPU resources utilized during model execution.
- **GPU Usage:** The extent to which GPU resources were leveraged to enhance performance.
- **Memory Usage:** The amount of RAM consumed while the model was running.
- **Multiple Choice Question Answering:** The accuracy of the model when answering structured multiple-choice questions.

8.5.2. Qualitative Evaluation Methods

While qualitative evaluation is inherently resource-intensive due to its reliance on human judgment, it remains essential for assessing aspects that cannot

be fully captured through quantitative metrics. Consequently, although our primary focus was on quantitative evaluation, we conducted qualitative assessments for key criteria where human input was indispensable:

- **Translation Quality:** Evaluated using the BLEU (Bilingual Evaluation Understudy) score, which measures the similarity between the model-generated translation and a human reference translation.
- **Text Generation Quality:** Assessed through the ROUGE (Recall-Oriented Understudy for Gisting Evaluation) score, which quantifies the lexical overlap between generated text and reference texts.
- **Grammatical Accuracy:** Manually reviewed by human evaluators to identify grammatical errors and assess syntactic correctness.
- **Readability:** Measured using the Flesch Reading Ease score, which indicates the complexity and accessibility of the generated text.
- **Sentiment Polarity:** Analyzed to determine whether the generated text conveys a positive, negative, or neutral sentiment.

?

By integrating both quantitative and qualitative evaluation methods, we achieved a more comprehensive understanding of each model's strengths and weaknesses, allowing for a well-rounded assessment of their performance.

8.5.3. Models Evaluated During Testing

For the evaluation process, we selected some of the most popular models available in the Ollama application and conducted extensive testing on each. The models vary in size, specialization, and intended use cases, covering general-purpose, coding, mathematical, reasoning, and image-processing tasks.

- **qwen2.5-coder:0.5b** – A compact model with 0.5 billion parameters, specifically designed for coding-related tasks.
- **qwen2.5-coder:7b** – A small-scale model with 7 billion parameters, optimized for software development and code generation.

- **qwen2.5-coder:14b** – A mid-sized model with 14 billion parameters, tailored for complex coding tasks.⁷
- **qwen2-math** – A specialized model with 1 billion parameters, fine-tuned for mathematical computations.
- **llama3.2:1b** – A lightweight model with 1 billion parameters, designed by Meta for general-purpose applications.
- **llama3.2:2b** – A medium-sized model with 2 billion parameters, developed by Meta for a broader range of general-purpose tasks.
- **mistral:7b** – A versatile 7-billion-parameter model created by Mistral AI, a European AI company, for general applications.
- **mathstral** – A specialized model optimized for mathematical problem-solving, developed by Mistral AI.
- **phi4:14b** – A mid-sized model with 14 billion parameters, designed by Microsoft for general-purpose reasoning tasks.
- **deepseek-r1:1.5b** – A small-scale model with 1.5 billion parameters, featuring enhanced reasoning capabilities.
- **deepseek-r1:7b** – A 7-billion-parameter model optimized for reasoning tasks, offering a balance between performance and hardware compatibility.
- **deepseek-r1:14b** – A more powerful variant with 14 billion parameters, fine-tuned for complex reasoning tasks.
- **gemma2** – A lightweight model with 1 billion parameters, designed by Google for general-purpose applications.
- **llava:13b** – A large-scale model with 13 billion parameters, developed for image processing tasks by a dedicated research team. ?

?

8.5.4. Data Collection

For the Data collection we used the School AI Server which was provided by the HTL Anichstraße, to run the different models in the Evaluation phase. For the later Data collection we used our own Server to run the different models.

To collect the data we used a variety of different python scripts. For the quantitative data we used the following python script:

```

1 import time
2 import json
3 import psutil # For CPU, memory usage
4 from ollama import chat
5
6 # Prompts for testing
7 prompts = [
8     "Explain the theory of relativity in simple terms.",
9     "Create a short story about a knight.",
10    "What are the advantages of open-source projects?",
11    "Write a Python function that outputs prime numbers up to 100.",
12    "# ..."
13]
14
15 # Model name
16 model_name = "qwen2-math"
17
18 # Store results
19 results = []
20
21 # Function to get GPU usage if available
22 def get_gpu_usage():
23     try:
24         import torch
25         if torch.cuda.is_available():
26             gpu_memory = torch.cuda.memory_allocated() / (1024 ** 2) # Convert to MB
27             gpu_utilization = torch.cuda.utilization(0) if hasattr(torch.cuda, 'utilization') else
28             "N/A"
29             return gpu_memory, gpu_utilization
30         else:
31             return 0, "No GPU detected"
32     except ImportError:
33         return 0, "torch not installed"
34
35 # Loop through prompts
36 for prompt in prompts:
37     try:
38         # Measure system usage before model execution
39         cpu_before = psutil.cpu_percent(interval=None)
40         memory_before = psutil.virtual_memory().used / (1024 ** 2) # Convert to MB
41
42         start_time = time.time()
43         # Ollama chat request
44         response = chat(model=model_name, messages=[{"role": "user", "content": prompt}])
45         end_time = time.time()
46
47         latency = end_time - start_time
48
49         # Measure system usage after model execution
50         cpu_after = psutil.cpu_percent(interval=None)
51         memory_after = psutil.virtual_memory().used / (1024 ** 2) # Convert to MB
52
53         cpu_usage = cpu_after - cpu_before
54         memory_usage = memory_after - memory_before
55         gpu_memory_usage, gpu_utilization = get_gpu_usage()
56
57         # Extract content from the Message object
58         if response and hasattr(response["message"], "content"):
59             response_text = response["message"].content # Accessing the attribute of the Message
60             object
61         else:
62             response_text = "No content returned or unexpected format"
63
64         print(f"Prompt: {prompt}\nResponse Time: {latency:.2f} seconds\n")
65
66         # Save the result
67         results.append({
68             "Prompt": prompt,
69             "Response Time (seconds)": latency,
70             "Response": response_text,
71             "CPU Usage (%)": cpu_usage,
72         })
73     except Exception as e:
74         print(f"An error occurred while processing prompt: {prompt}. Error: {e}")
75
76 # Print the collected results
77 print(results)

```

```

70     "Memory Usage (MB)": memory_usage,
71     "GPU Memory Usage (MB)": gpu_memory_usage,
72     "GPU Utilization (%)": gpu_utilization
73   })
74 except Exception as e:
75   print(f"Error with prompt '{prompt}': {e}")
76   results.append({
77     "Prompt": prompt,
78     "Response Time (seconds)": "Error",
79     "Response": f"Error: {str(e)}",
80     "CPU Usage (%)": "N/A",
81     "Memory Usage (MB)": "N/A",
82     "GPU Memory Usage (MB)": "N/A",
83     "GPU Utilization (%)": "N/A"
84   }))20.01.2025
85
86 # Save to JSON file
87 json_file_name = model_name + "_response_time_results_ressours_usage.json"
88 with open(json_file_name, "w") as file:
89   json.dump(results, file, indent=4)
90
91 print(f"The results have been saved in {json_file_name}.")

```

Listing 8.1: Python-quantitative-data-collection

For this Pyhton Skript we used ChatGPT to help with the psutil library to get the CPU and Memory usage. We also used ChatGPT to get more Questions for the data collection.

The results were saved as JSON files for the later analysis. One Problem we encountered were the tourch librarys, witch didn't function probally on the School AI Server, so we couldn't get the GPU usage for the models.

Used python libaries

psutil libarie

The psutil library is a Python module that provides an interface for retrieving information on system utilization, including CPU, memory, disk, network, and processes. It is commonly used for monitoring and managing system performance and is highly efficient due to its low overhead. psutil is cross-platform, supporting major operating systems like Windows, Linux, and macOS. It enables developers to create scripts for system diagnostics, process control, and resource management, making it an essential tool for performance optimization and system administration in Python-based projects.

?

Gabriel Mrkonja
 Florian Prandstetter
 Luna P. I. Schätzle

Ollama

The `ollama` library is a Python package designed to provide a seamless interface for interacting with the Ollama application. It allows users to easily access and leverage various AI models for natural language processing tasks. By simplifying the integration of AI models into Python applications, the library supports a wide range of functionalities, making it an efficient tool for developing AI-powered solutions.

?

8.5.5. Data Preparation

To get more information about the collected data, we used the following Python script to collect more data:

```

1 import json
2 import os
3 from nltk.translate.bleu_score import sentence_bleu, SmoothingFunction
4 from rouge_score import rouge_scorer
5 import language_tool_python
6 import textstat
7 from transformers import pipeline, logging
8
9 # Suppress warning messages from the Transformer library for a cleaner output.
10 logging.set_verbosity_error()
11
12 def load_json(file_path):
13     """
14         Load and parse JSON data from a file.
15
16     Parameters:
17         file_path (str): The file system path to the JSON file.
18
19     Returns:
20         dict or list: The JSON data parsed from the file.
21     """
22     with open(file_path, 'r') as file:
23         return json.load(file)
24
25 def calculate_metrics(data):
26     """
27         Compute multiple evaluation metrics for generated text responses.
28
29         For each data item, the function calculates:
30             - BLEU Score: Quantifies the similarity between the generated response and the reference text.
31             - ROUGE Scores: Evaluates the n-gram overlap between the reference and the generated text,
32                 using ROUGE-1, ROUGE-2, and ROUGE-L.
33             - Grammar Check: Determines the number of grammatical errors present in the response.
34             - Readability Score: Computes the Flesch Reading Ease score to assess the text's readability.
35             - Sentiment Analysis: Infers the sentiment polarity (e.g., positive or negative) of the
36                 response text.
37
38     Parameters:
39         data (list): A list of dictionaries, each containing 'Prompt', 'Response', and 'Reference'
40             keys.
41
42     Returns:
43         list: A list of dictionaries that include the original text elements along with the
44             computed metrics.

```

```

41      """
42      results = []
43      rouge = rouge_scorer.RougeScorer(['rouge1', 'rouge2', 'rougeL'], use_stemmer=True)
44      grammar_tool = language_tool_python.LanguageTool('en-US')
45      sentiment_analyzer = pipeline(
46          'sentiment-analysis',
47          model="distilbert-base-uncased-finetuned-sst-2-english",
48          truncation=True,
49          max_length=512
50      )
51
52      for item in data:
53          prompt = item['Prompt']
54          response = item['Response']
55          reference = item['Reference']
56
57          try:
58              # Calculate the BLEU Score with smoothing to address potential issues with short
59              # sequences.
60              bleu_score = sentence_bleu(
61                  [reference.split()], response.split(),
62                  smoothing_function=SmoothingFunction().method4
63              )
64
65              # Calculate ROUGE Scores for comprehensive n-gram overlap assessment.
66              rouge_scores = rouge.score(reference, response)
67
68              # Perform grammatical analysis by counting detected errors in the response.
69              grammar_errors = len(grammar_tool.check(response))
70
71              # Determine the readability score using the Flesch Reading Ease metric.
72              readability_score = textstat.flesch_reading_ease(response)
73
74              # Analyze the sentiment of the response text, with error handling to capture any
75              # exceptions.
76              try:
77                  sentiment_result = sentiment_analyzer(response)[0]
78                  sentiment = sentiment_result['label']
79              except Exception as e:
80                  print(f"Sentiment error for prompt '{prompt}': {e}")
81                  sentiment = "Error"
82
83              results.append({
84                  "Prompt": prompt,
85                  "Response": response,
86                  "Reference": reference,
87                  "BLEU": bleu_score,
88                  "ROUGE-1": rouge_scores['rouge1'].fmeasure,
89                  "ROUGE-2": rouge_scores['rouge2'].fmeasure,
90                  "ROUGE-L": rouge_scores['rougeL'].fmeasure,
91                  "Grammar Errors": grammar_errors,
92                  "Readability Score": readability_score,
93                  "Sentiment": sentiment
94              })
95          except Exception as e:
96              print(f"Error processing prompt '{prompt}': {e}")
97              results.append({
98                  "Prompt": prompt,
99                  "Response": response,
100                 "Reference": reference,
101                 "Error": str(e)
102             })
103
104     return results
105
106     def save_results(results, output_path):
107         """
108             Persist the computed metrics to a JSON file.
109
110             Parameters:
111                 results (list): A list of dictionaries containing evaluation metrics and corresponding
112                 texts.
113                 output_path (str): The file system path where the output JSON should be saved.
114
115             with open(output_path, 'w') as file:

```

Gabriel Mrkonja
 Florian Prandstetter
 Luna P. I. Schätzle

```

113         json.dump(results, file, indent=4)
114
115     def main():
116         """
117             Execute the main workflow of the script.
118
119             This function prompts the user to specify a directory containing preprocessed JSON files.
120             It then iterates through each file that matches the pattern 'processed_*.json',
121             computes the evaluation metrics for the contained data,
122             and saves the results in a new JSON file prefixed with 'scored_'.
123             """
124
125     directory = input("Enter the directory containing the processed JSON files: ")
126
127     try:
128         for file_name in os.listdir(directory):
129             if file_name.startswith("processed_") and file_name.endswith(".json"):
130                 input_file = os.path.join(directory, file_name)
131                 model_name = file_name.split("processed_")[1].split(".json")[0]
132                 output_file = os.path.join(directory, f"scored_{model_name}.json")
133
134                 print(f"Processing file: {input_file}")
135                 data = load_json(input_file)
136                 metrics = calculate_metrics(data)
137                 save_results(metrics, output_file)
138                 print(f"Metrics saved to: {output_file}")
139
140     except FileNotFoundError:
141         print(f"Error: The directory {directory} was not found.")
142     except Exception as e:
143         print(f"An error occurred: {e}")
144
145 if __name__ == "__main__":
146     main()

```

Listing 8.2: Python-data-preperation-for-analysis

This Python script implements a comprehensive evaluation framework for assessing the quality of generated textual responses. It systematically processes JSON files containing a prompt, a generated response, and a reference text, computing several quantitative metrics: the BLEU score for assessing n-gram overlap, ROUGE metrics for evaluating text similarity, a grammatical error count via LanguageTool, the Flesch Reading Ease score for readability, and sentiment analysis using a Transformer-based model. By integrating these diverse analytical techniques from state-of-the-art natural language processing libraries, the script facilitates a rigorous and multifaceted scientific evaluation of text generation performance.

Utilized Python Libraries

For Collecting those data we used the following Python Libraries:

Natural Language Toolkit (NLTK) The Natural Language Toolkit (NLTK) is a comprehensive library for natural language processing in Python. It

provides easy-to-use interfaces to over 50 corpora and lexical resources, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning. NLTK is widely used for building Python programs that work with human language data and is a leading platform in both research and education ?.

ROUGE Score The rouge-score library is a native Python implementation of the ROUGE metric, which is commonly used for evaluating automatic summarization and machine translation. It replicates the results of the original Perl package and supports the computation of ROUGE-N (N-gram) and ROUGE-L (Longest Common Subsequence) scores. The library also offers functionalities such as text normalization and the use of stemming to enhance evaluation accuracy ?.

LanguageTool Python language-tool-python is a wrapper for LanguageTool, an open-source grammar, style, and spell checker. This library enables the integration of LanguageTool's proofreading capabilities into Python applications, supporting the detection and correction of grammatical errors, stylistic issues, and spelling mistakes across multiple languages.

Textstat The textstat library provides simple methods for calculating readability statistics from text. It helps determine the readability, complexity, and grade level of textual content by computing various metrics such as the Flesch Reading Ease, SMOG Index, and Gunning Fog Index. These metrics are valuable for assessing and ensuring the comprehensibility of text, particularly in educational and professional settings ?.

8.5.6. Data Processing

To gain a better understanding of the collected data, we utilized Python scripts to generate visualizations, providing a clearer representation of the results. Additionally, the processed data was formatted into a LaTeX table to facilitate structured analysis and comparison.

Gabriel Mrkonja
Florian Prandstetter
Luna P. I. Schätzle

Quantitative Data Analysis

To visualize the quantitative data, we employed the following Python script:

```

1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4 import glob
5 import numpy as np
6 import os
7 import logging
8
9 # Set up logging for consistent error and information messages.
10 logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')
11
12 # Set scientific plotting style with an increased default figure height.
13 plt.style.use('default')
14 sns.set_theme(style="whitegrid", context="paper")
15 plt.rcParams.update({
16     # Here are all the params for the settings for matplotlib
17 })
18
19 def load_and_process_data() -> pd.DataFrame:
20     """
21     Loads and processes all JSON files in the current directory.
22
23     This function searches for all files matching "*.json", reads them into pandas DataFrames,
24     assigns a 'Model' column based on the file name (without extension), converts specified
25     numeric columns to numeric type, and removes rows with missing values in those columns.
26
27     Returns:
28         pd.DataFrame: A concatenated and cleaned DataFrame containing all data.
29     """
30     json_files = glob.glob("*.json")
31     if not json_files:
32         logging.warning("No JSON files found in the current directory.")
33         return pd.DataFrame()
34
35     dfs = []
36     for file in json_files:
37         try:
38             model_name = os.path.splitext(file)[0]
39             df = pd.read_json(file)
40             df['Model'] = model_name
41             dfs.append(df)
42         except Exception as e:
43             logging.error(f"Error loading {file}: {e}")
44
45     if not dfs:
46         logging.error("No data could be loaded from the JSON files.")
47         return pd.DataFrame()
48
49     combined_df = pd.concat(dfs, ignore_index=True)
50
51     # Convert selected columns to numeric and drop rows with missing values in these columns.
52     numeric_cols = ['Response Time (seconds)', 'CPU Usage (%)', 'Memory Usage (MB)']
53     combined_df[numeric_cols] = combined_df[numeric_cols].apply(pd.to_numeric, errors='coerce')
54     combined_df = combined_df.dropna(subset=numeric_cols)
55
56     return combined_df
57
58 def create_resource_plot(df: pd.DataFrame, metric: str, title: str, ylabel: str, filename: str)
59     -> None:
60     """
61     Creates a resource usage plot with violin and strip plots, annotated with statistical
62     measures.
63
64     The function generates a violin plot for the given metric across different AI models,
65     overlays a strip plot
66     to display individual data points, and annotates each model with its median and mean absolute
67     deviation (MAD).

```

```

64     The resulting plot is saved in both PDF and PNG formats.
65
66     Parameters:
67         df (pd.DataFrame): DataFrame containing the metric and 'Model' columns.
68         metric (str): The column name representing the metric to be visualized.
69         title (str): The title of the plot.
70         ylabel (str): The label for the y-axis.
71         filename (str): Base filename used for saving the plot.
72     """
73     plt.figure(figsize=(10, 10))
74
75     ax = sns.violinplot(
76         x='Model',
77         y=metric,
78         data=df,
79         inner='quartile',
80         palette='muted',
81         cut=0
82     )
83
84     sns.stripplot(
85         x='Model',
86         y=metric,
87         data=df,
88         color='#303030',
89         size=2.5,
90         alpha=0.7
91     )
92
93     # Calculate median and mean absolute deviation (MAD) for each model.
94     stats = df.groupby('Model')[metric].agg(median='median', mad=lambda x: np.mean(np.abs(x - x.median())))
95
96     # Annotate each model with the calculated median and MAD.
97     for xtick, model in enumerate(stats.index):
98         model_stats = stats.loc[model]
99         annotation = f"Med: {model_stats['median']:.1f}\nMAD: {model_stats['mad']:.1f}"
100        # Position annotation at 5% above the minimum value.
101        y_pos = df[metric].min() + (df[metric].max() - df[metric].min()) * 0.05
102        ax.text(
103            xtick,
104            y_pos,
105            annotation,
106            ha='center',
107            va='bottom',
108            fontsize=8,
109            color="#404040"
110        )
111
112     plt.title(title, pad=15)
113     plt.xlabel('AI Model', labelpad=12)
114     plt.ylabel(ylabel, labelpad=12)
115     plt.xticks(rotation=45, ha='right')
116     plt.ylim(bottom=0)
117     plt.tight_layout()
118
119     # Save the plot in both vector (PDF) and raster (PNG) formats.
120     plt.savefig(f'{filename}.pdf', bbox_inches='tight')
121     plt.savefig(f'{filename}.png', bbox_inches='tight')
122     plt.close()
123
124 def plot_cpu_memory_comparison(df: pd.DataFrame) -> None:
125     """
126         Generates comparative plots for CPU usage and memory consumption across AI models.
127
128         This function calls 'create_resource_plot' for both CPU and Memory metrics.
129
130         Parameters:
131             df (pd.DataFrame): DataFrame containing performance metrics.
132         """
133         create_resource_plot(
134             df=df,
135             metric='CPU Usage (%)',
136             title='Comparative Analysis of CPU Utilization Across AI Models',
137             ylabel='CPU Usage (%)',

```

Gabriel Mrkonja
 Florian Prandstetter
 Luna P. I. Schätzle

```

138         filename='model_cpu_usage_comparison'
139     )
140
141     create_resource_plot(
142         df=df,
143         metric='Memory Usage (MB)',
144         title='Comparative Analysis of Memory Consumption Across AI Models',
145         ylabel='Memory Usage (MB)',
146         filename='model_memory_usage_comparison'
147     )
148
149     def generate_advanced_statistics(df: pd.DataFrame) -> None:
150         """
151             Generates advanced performance statistics for AI models and outputs the results both in the
152             console and as a LaTeX table.
153
154             The statistics include mean, standard deviation, and maximum values for CPU and memory usage,
155             as well as mean and standard deviation for response times.
156
157             Parameters:
158                 df (pd.DataFrame): DataFrame containing performance metrics.
159
160             stats = df.groupby('Model').agg({
161                 'CPU Usage (%)': ['mean', 'std', 'max'],
162                 'Memory Usage (MB)': ['mean', 'std', 'max'],
163                 'Response Time (seconds)': ['mean', 'std']
164             })
165
166             print("\nAdvanced Performance Statistics:")
167             print(stats.round(2).to_string())
168
169             # Export the statistics as a formatted LaTeX table.
170             try:
171                 latex_str = stats.style.format({
172                     ('CPU Usage (%)', 'mean'): "{:.1f}",
173                     ('Memory Usage (MB)', 'mean'): "{:.1f}"
174                 }).to_latex(
175                     hrules=True,
176                     caption="Model Performance Statistics",
177                     label="tab:model_stats"
178                 )
179                 with open('resource_stats.tex', 'w') as f:
180                     f.write(latex_str)
181             except Exception as e:
182                 logging.error(f"Error generating LaTeX table: {e}")
183
184     def plot_response_times(df: pd.DataFrame) -> None:
185         """
186             Creates a comparative boxplot for model response times overlaid with a swarm plot for
187             individual data points.
188
189             The function annotates each AI model with its median response time and saves the plot in both
190             PDF and PNG formats.
191
192             Parameters:
193                 df (pd.DataFrame): DataFrame containing the 'Response Time (seconds)' and 'Model' columns.
194
195             plt.figure(figsize=(8, 10))
196
197             ax = sns.boxplot(
198                 x='Model',
199                 y='Response Time (seconds)',
200                 data=df,
201                 width=0.6,
202                 showfliers=False,
203                 palette='muted'
204             )
205
206             sns.swarmplot(
207                 x='Model',
208                 y='Response Time (seconds)',
209                 data=df,
209                 color='#404040',
210                 size=3,
211                 alpha=0.6
212             )

```

```

210
211
212     # Annotate the median response time for each model.
213     medians = df.groupby('Model')['Response Time (seconds)'].median()
214     for xtick, model in enumerate(medians.index):
215         median_val = medians.loc[model]
216         ax.text(
217             xtick,
218             median_val + 0.05,
219             f'{median_val:.2f}s',
220             ha='center',
221             va='bottom',
222             fontsize=8,
223             color='#2f2f2f'
224         )
225
226     plt.title('Comparative Analysis of Model Response Times', pad=15)
227     plt.xlabel('AI Model', labelpad=10)
228     plt.ylabel('Response Time (seconds)', labelpad=10)
229     plt.xticks(rotation=45, ha='right')
230     plt.tight_layout()
231
232     plt.savefig('model_response_times_comparison.pdf', bbox_inches='tight')
233     plt.savefig('model_response_times_comparison.png', bbox_inches='tight')
234     plt.close()
235
236 def generate_statistics(df: pd.DataFrame) -> None:
237     """
238         Generates a statistical summary of response times for each AI model and exports the results.
239
240         The summary includes the mean, standard deviation, minimum, median, and maximum values.
241         The results are printed to the console and saved as a LaTeX table.
242
243         Parameters:
244             df (pd.DataFrame): DataFrame containing the 'Response Time (seconds)' and 'Model' columns.
245         """
246     stats = df.groupby('Model')['Response Time (seconds)'].describe()
247     print("\nResponse Time Statistics:")
248     print(stats[['mean', 'std', 'min', '50%', 'max']].round(3).to_string())
249
250     try:
251         with open('response_stats.tex', 'w') as f:
252             f.write(
253                 stats[['mean', 'std', 'min', '50%', 'max']]
254                 .round(3)
255                 .style.to_latex(hrules=True)
256             )
257     except Exception as e:
258         logging.error(f"Error generating LaTeX response stats: {e}")
259
260 def main() -> None:
261     """
262         Main execution function.
263
264         Loads and processes data from JSON files, generates various comparative plots (response
265             times, CPU, and memory usage),
266             and outputs advanced performance statistics along with their LaTeX representations.
267         """
268     df = load_and_process_data()
269     if df.empty:
270         logging.error("No data available for plotting and analysis.")
271         return
272
273     plot_response_times(df)
274     plot_cpu_memory_comparison(df)
275     generate_advanced_statistics(df)
276     generate_statistics(df)
277
278 if __name__ == "__main__":
279     main()

```

Listing 8.3: Python-quantitative-data-analysis

This script performs a comprehensive performance evaluation of various AI models by loading JSON files from the working directory, extracting key metrics such as response time, CPU usage, and memory consumption, and preprocessing the data for analysis.

It generates high-resolution visualizations—including violin, strip, box, and swarm plots—to effectively illustrate the distributions and central tendencies of these metrics. Additionally, it computes descriptive statistics and presents the results both in the console and as LaTeX-formatted tables, ensuring structured and reproducible scientific reporting.

Qualitative Data Analysis

To visualize the qualitative data, we utilized the following Python script:

```

1  import pandas as pd
2  import matplotlib.pyplot as plt
3  import seaborn as sns
4  import os
5
6  # =====#
7  # Data Aggregation and Visualization for Model Performance Metrics
8  # =====#
9  # This script aggregates experimental results from JSON files, each containing
10 # performance metrics (e.g., BLEU, ROUGE, grammatical errors, readability, sentiment)
11 # for various AI models. The data are visualized using high-quality plots for scientific
12 # analysis, and descriptive statistics are exported in LaTeX format.
13 #
14 # -----
15 # Data Aggregation
16 # -----
17 directory = "./"
18 aggregated_data = []
19 for file in os.listdir(directory):
20     # Process files that follow the naming convention "scored_<model>.json"
21     if file.startswith("scored_") and file.endswith(".json"):
22         model = file.replace("scored_", "").replace(".json", "")
23         df_temp = pd.read_json(os.path.join(directory, file))
24         df_temp["Model"] = model
25         aggregated_data.append(df_temp)
26 df = pd.concat(aggregated_data, ignore_index=True)
27
28 # -----
29 # Global Plotting Style Settings
30 # -----
31 sns.set_theme(style="whitegrid", font_scale=0.9)
32 plt.rcParams['axes.titlepad'] = 15
33 plt.rcParams['axes.labelpad'] = 10
34
35 def rotate_labels(ax, rotation: int = 45, ha: str = 'right') -> None:
36     """
37     Rotate the x-axis labels for improved readability.
38
39     Parameters:
40         ax (matplotlib.axes.Axes): The axes on which to rotate the labels.
41         rotation (int): Angle in degrees to rotate the labels.
42         ha (str): Horizontal alignment of the labels.
43
44     """
45     ax.set_xticklabels(ax.get_xticklabels(), rotation=rotation, ha=ha, fontsize=9)
46     plt.tight_layout()

```

```

46 # =====
47 # Visualization of Performance Metrics
48 # =====
49
50
51 # --- 1. BLEU and ROUGE Scores ---
52 plt.figure(figsize=(14, 7))
53 # Reshape data for plotting multiple text quality metrics
54 df_melt = df.melt(id_vars=["Model"], value_vars=["BLEU", "ROUGE-1", "ROUGE-2", "ROUGE-L"],
55                   var_name="Metric", value_name="Score")
56 ax = sns.barplot(x="Model", y="Score", hue="Metric", data=df_melt, palette="viridis")
57 plt.title("Comparison of BLEU and ROUGE Scores", fontweight='bold')
58 plt.ylim(0, 0.05)
59 plt.legend(loc="upper right", frameon=True)
60 rotate_labels(ax)
61 plt.savefig("bleu_rouge.png", dpi=300, bbox_inches="tight")
62
63 ##### Other plots are similar to the first one, but with different kinds of plots
64 # The plots are for:
65
66 # --- 2. Grammatical Errors ---
67 # --- 3. Readability (Flesch Score) ---
68 # --- 4. Sentiment Analysis ---
69 # --- 5. Combined Metrics Overview with Subplots ---
70 #####
71
72 # =====
73 # Descriptive Statistics Export
74 # =====
75 # Compute summary statistics for selected metrics by model
76 summary = df.groupby("Model")[["BLEU", "ROUGE-L", "Grammar Errors"]].agg(["mean", "std",
77   "median", "min", "max"])
summary.to_latex("summary.tex", float_format=".3f")

```

Listing 8.4: Python-qualitative-data-analysis

This script aggregates performance metrics from multiple JSON files—each corresponding to an AI model evaluation—into a unified dataset. It then generates high-resolution visualizations, including bar, box, and point plots, to illustrate text quality (BLEU/ROUGE scores), grammatical accuracy, readability, and sentiment distribution. Finally, it computes descriptive statistics for these metrics and exports a summary table in LaTeX format for rigorous scientific reporting.

Utilized Python Libraries

In this project, several Python libraries were employed to facilitate data manipulation, analysis, and visualization:

Pandas Pandas is an open-source data analysis and manipulation library for Python. It provides data structures such as Series and DataFrames, which allow for efficient handling of structured data. Pandas supports operations

Gabriel Mrkonja
 Florian Prandstetter
 Luna P. I. Schätzle

like data alignment, merging, and reshaping, making it indispensable for data preprocessing and analysis tasks.

?

Matplotlib Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It offers an object-oriented API for embedding plots into applications and supports various plot types, including line, bar, scatter, and 3D plots. Matplotlib's flexibility and extensive customization options make it a fundamental tool for data visualization.

?

Seaborn Seaborn is a statistical data visualization library built on top of Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics, including functions for visualizing univariate and bivariate distributions, categorical data, and linear regression models. Seaborn integrates closely with Pandas data structures, enhancing the aesthetic appeal and interpretability of visualizations.

?

NumPy NumPy is a foundational library for numerical computing in Python. It introduces support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. NumPy serves as the backbone for many other libraries, including Pandas and Matplotlib, by providing efficient array operations and numerical computations.

?

8.5.7. Test Results and Analysis

After data collection and processing, the following visualizations were obtained:

Quantitative Data Analysis

The visualizations below offer insights into the performance metrics of various AI models, focusing on response times, CPU usage, and memory consumption.

CPU Usage The violin plot below illustrates the distribution of CPU usage across different AI models.

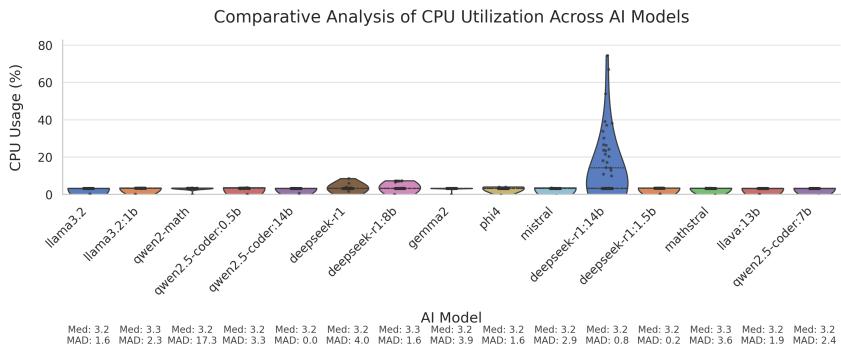


Abbildung 8.1.: Comparative Analysis of CPU Utilization Across AI Models

This plot highlights the variability and central tendencies in CPU usage among the evaluated models.

Observations:

CPU usage is relatively consistent across the models; however, the Deepseek (reasoning) models exhibit higher CPU usage, with the largest Deepseek model (14 billion parameters) showing the highest utilization.

Memory Usage The following violin plot visualizes the memory consumption of various AI models.

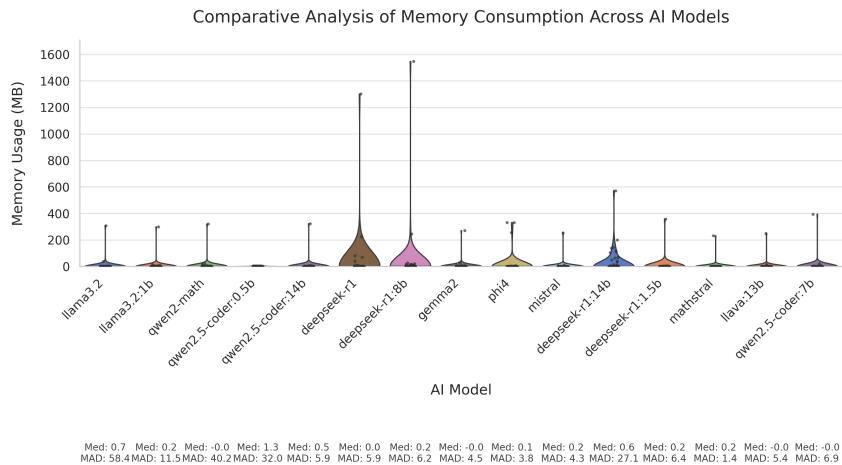


Abbildung 8.2.: Comparative Analysis of Memory Consumption Across AI Models

This visualization reveals the memory usage patterns and distributions, providing valuable insights into resource allocation.

Observations:

Memory usage is generally comparable across the models. Nevertheless, the Deepseek (reasoning) models demonstrate higher memory consumption with a wider spread, particularly in the 8 and 7 billion parameter variants.

Response Times A box plot was generated to depict the distribution of response times across the different models.

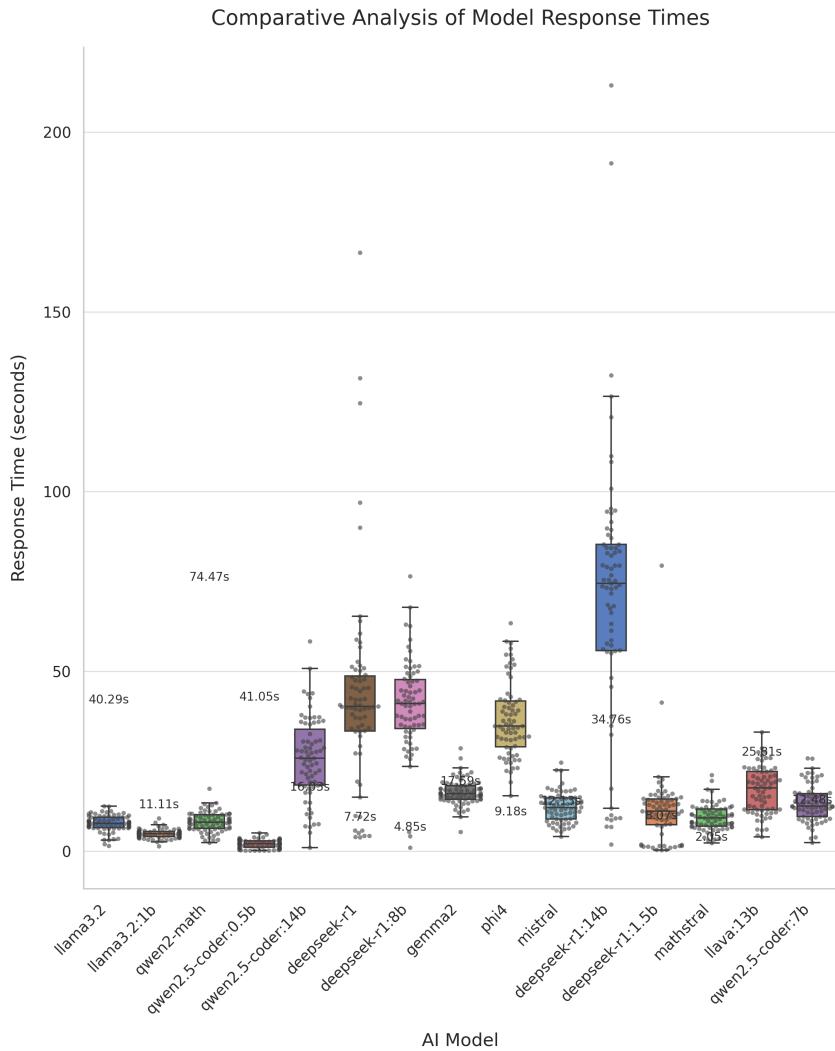


Abbildung 8.3.: Comparative Analysis of Model Response Times

This box plot provides a clear overview of the central tendencies and variability in response times.

Observations:

Response times vary significantly among the models. The Llama models display very low and consistent response times, which is advantageous

for the intended use case of the School AI Server. In contrast, the Gemma, Mistral, Mathstral, and two Qwen-coder models exhibit higher response times with a noticeable spread. Larger models, such as the Qwen-coder 14 billion parameter model and the Phi4 model, also show increased response times and variability. Notably, the Deepseek models, particularly the 14 billion parameter variant, record the highest response times and spread, likely due to the additional computational demands required for their reasoning processes.

8.5.8. Qualitative Data Analysis

The visualizations presented in this section provide insights into various text quality metrics of different AI models, including BLEU and ROUGE scores, grammatical error counts, readability, and sentiment analysis.

BLEU and ROUGE Scores The bar plot below compares the BLEU and ROUGE scores across multiple AI models.

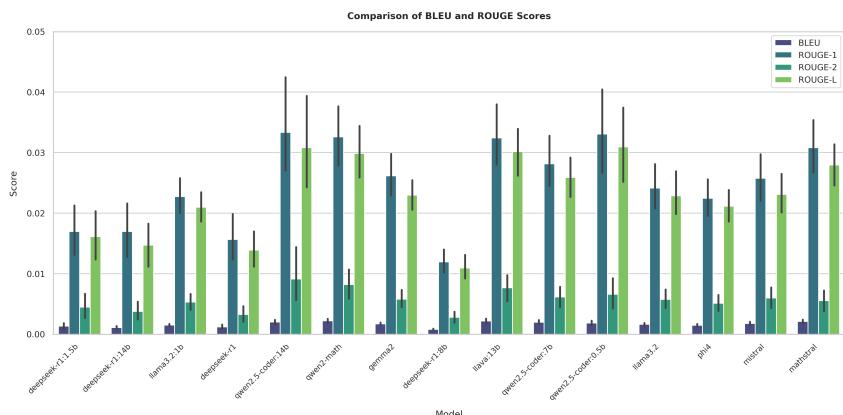


Abbildung 8.4.: Comparison of BLEU and ROUGE Scores

This visualization offers a comprehensive overview of text quality metrics. The BLEU score quantifies the similarity between generated text and a

reference text, while the ROUGE scores (including ROUGE-1, ROUGE-2, and ROUGE-L) measure the n-gram overlap between them.

Observations:

BLEU and ROUGE scores vary significantly among the models, reflecting differences in text generation quality. Specialized models for mathematics and coding (e.g., Mathstral and Qwen-coder models) achieve the highest scores due to their technical focus. In contrast, general-purpose models such as Llama and Gemma exhibit lower scores, likely as a consequence of their broader but less specialized capabilities. An exception is the Llava model, which—despite being a general-purpose model with vision capabilities—demonstrates a relatively high BLEU score, possibly due to its unique text generation approach based on images. Additionally, the Mistral and Gemma2 models display higher scores among general-purpose models, whereas reasoning models like Deepseek yield the lowest BLEU and ROUGE scores, prioritizing complex logical reasoning over text quality.

Grammatical Errors The box plot below illustrates the distribution of grammatical errors detected across different AI models. Grammatical error count serves as an important metric for assessing the linguistic accuracy of generated text. The Language Tool library was used to detect and count errors, with English specified as the target language for consistent and reliable results.¹

¹Note that some errors detected in code snippets were manually corrected to preserve the intended meaning of the visualization.

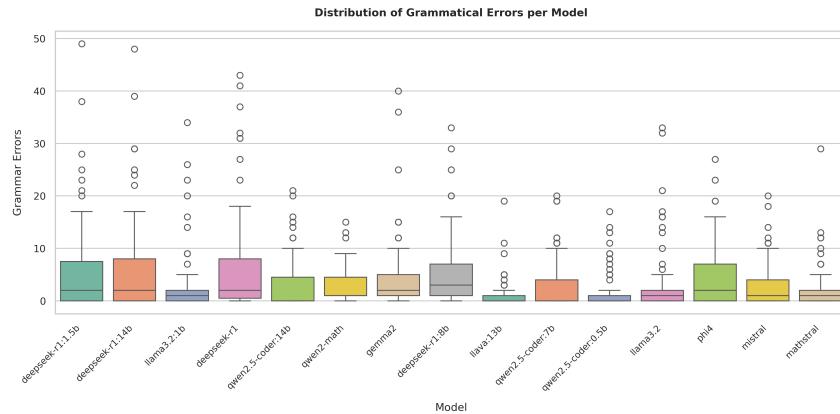


Abbildung 8.5.: Comparison of Grammatical Errors Across AI Models

Observations:

The number of grammatical errors varies considerably between models. The Mathstral, Llava, and Llama models exhibit the fewest errors, with the Qwen-coder models also performing well despite their focus on code generation. In contrast, reasoning models such as Deepseek produce a higher number of errors, which may be attributable to their emphasis on complex logical reasoning rather than linguistic precision. Additionally, some models might generate more errors due to non-native English influences. Notably, the Phi4 model, a general-purpose model developed in the United States, also shows a higher error count.

Readability (Flesch Score) The Flesch Reading Ease score quantifies text readability based on the average number of syllables per word and words per sentence. Higher scores indicate more accessible text, while lower scores suggest increased complexity.

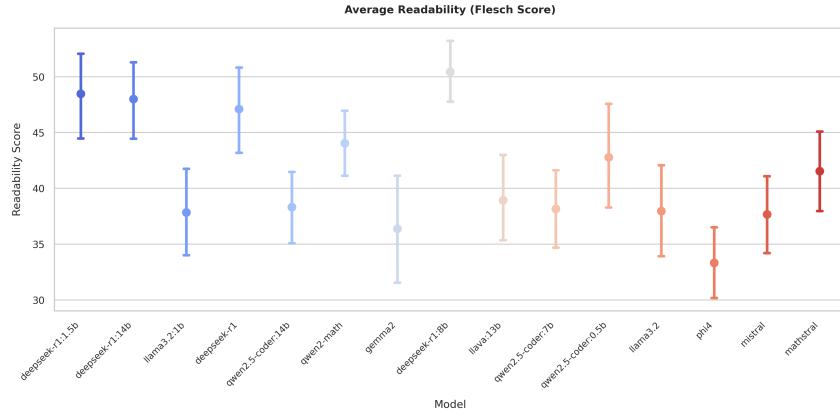


Abbildung 8.6.: Comparison of Readability Scores Across AI Models

Observations:

Readability scores differ markedly among the models. Interestingly, the Deepseek models achieve the highest readability scores despite their lower BLEU, ROUGE, and grammatical accuracy metrics, possibly due to the nature of their reasoning-focused design. On the other hand, the Gemma2, Phi4, and Llama models score lowest in readability, which is unexpected for general-purpose models that typically aim for accessible language. Specialized models generally fall within the mid-range of readability scores.

Sentiment Analysis Sentiment analysis evaluates the emotional tone and polarity of the generated text. A Transformer-based model was employed to classify text as positive or negative.

Gabriel Mrkonja
Florian Prandstetter
Luna P. I. Schätzle

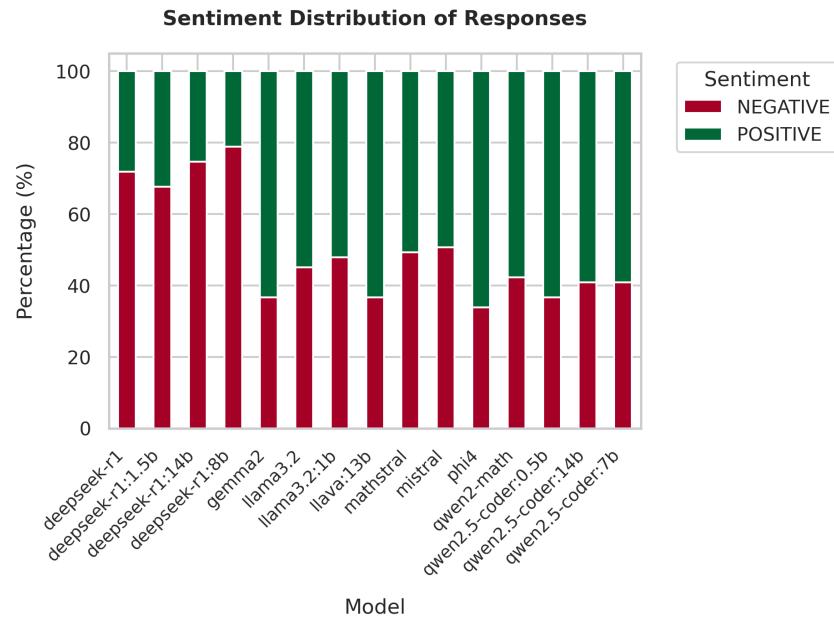


Abbildung 8.7.: Comparison of Sentiment Analysis Across AI Models

Observations:

Most models exhibit similar sentiment distributions, with approximately 50–60% of the text classified as positive and 40–50% as negative. However, the Deepseek models deviate from this pattern, showing a notably higher negative sentiment score (around 70–80%), which is intriguing given their high readability scores.

Combined Metrics Overview The subplots below provide a holistic overview of the combined text quality metrics, including BLEU and ROUGE scores, grammatical errors, readability, and sentiment analysis.

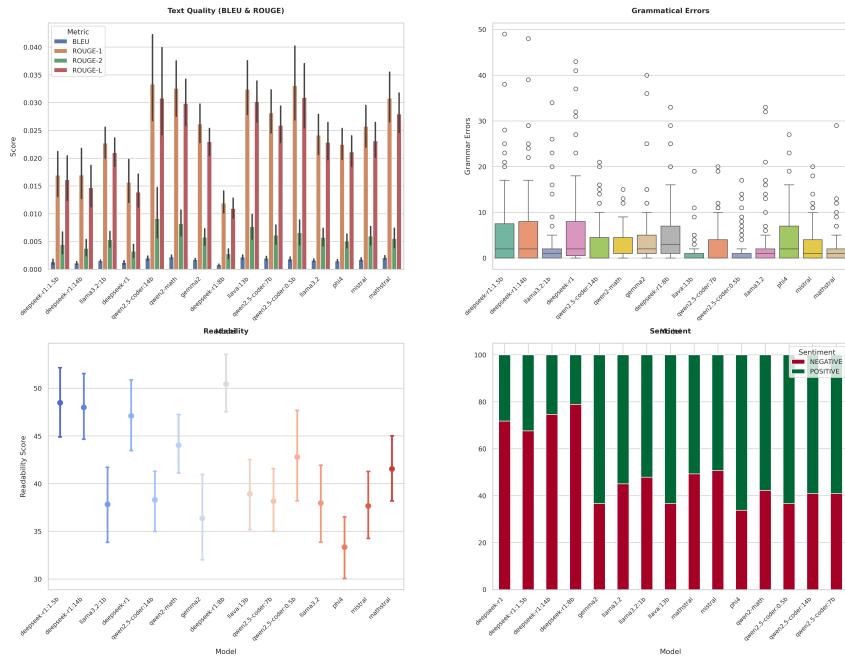


Abbildung 8.8.: Combined Metrics Overview for AI Models

These subplots facilitate a comprehensive comparative analysis of the performance of different AI models with respect to text quality.

8.5.9. Model Comparison for Different Use Cases and Scenarios

This section presents a comparative analysis of various AI models based on both quantitative and qualitative performance metrics. The evaluation focuses on aspects such as computational efficiency, text generation quality, and overall suitability for specific application scenarios.

Gabriel Mrkonja
 Florian Prandstetter
 Luna P. I. Schätzle

Performance Metrics Overview

The quantitative analysis reveals significant differences in resource utilization and response times across the models:

- **CPU and Memory Usage:** The Deepseek (reasoning) models tend to exhibit higher CPU and memory consumption compared to others. This is indicative of the increased computational demand required for complex logical processing.
- **Response Times:** Models such as Llama demonstrate very low and consistent response times, making them particularly well-suited for interactive applications. In contrast, larger models (e.g., Qwen-coder 14 billion and Phi4) and the Deepseek models show increased response times and higher variability.

Text Quality and Accuracy

The qualitative analysis, based on BLEU and ROUGE scores, grammatical error counts, readability, and sentiment analysis, provides further insights into each model's text generation capabilities:

- **BLEU and ROUGE Scores:** Specialized models (e.g., Mathstral and Qwen-coder) achieve the highest scores, reflecting superior performance in generating technical content. General-purpose models like Llama and Gemma tend to score lower due to their broader, less specialized focus.
- **Grammatical Accuracy:** Models such as Mathstral, Llava, and Llama exhibit fewer grammatical errors, while reasoning models like Deepseek show a higher error count, potentially due to their prioritization of logical reasoning over linguistic precision.
- **Readability:** Interestingly, the Deepseek models score highest on the Flesch Reading Ease metric, despite other text quality metrics indicating lower overall quality. In contrast, some general-purpose models such as Gemma2 and Phi4 show lower readability scores.
- **Sentiment Analysis:** Most models present a balanced sentiment distribution. However, the Deepseek models are characterized by a higher

proportion of negative sentiment, which may be attributed to the inherent complexity of their reasoning processes.

Use Case Recommendations

Based on the integrated analysis of both performance and text quality, the following recommendations can be made for various use cases:

Interactive and Real-Time Applications For scenarios requiring rapid response and minimal latency—such as the School AI Server—models like Llama are highly advantageous due to their consistently low response times.

Technical and Domain-Specific Content Generation Applications demanding high accuracy in technical content, particularly in mathematics and coding, are best served by specialized models like Mathstral and Qwen-coder. These models not only achieve superior BLEU and ROUGE scores but also exhibit fewer grammatical errors.

Complex Reasoning Tasks For tasks that necessitate advanced logical reasoning, the Deepseek models are appropriate despite their higher computational resource demands and slower response times. However, users should be aware of the trade-offs, including higher negative sentiment and a greater number of grammatical errors.

General-Purpose Applications For broader applications where versatility is key, general-purpose models such as Gemma, Gemma2, and Phi4 offer a balanced performance. Although they may not excel in any single metric, they provide reliable performance across a range of tasks.

Gabriel Mrkonja
Florian Prandstetter
Luna P. I. Schätzle

Overall Evaluation

The comparative analysis underscores that no single model excels universally across all metrics. Instead, the choice of an AI model should be closely aligned with the specific requirements and constraints of the intended application. Developers must weigh the trade-offs between computational efficiency, text quality, and domain-specific accuracy when selecting the most appropriate model for a given scenario.

8.5.10. Model Selected for the Final Application

For the initial version of the Student AI Hub application, the decision was made to allow users to choose between the models. This approach enables users to select the model that best meets their specific use cases and requirements, while also effectively demonstrating the distinct capabilities of each model.

Specialized models, such as Qwen-coder and LLava, are accessible through dedicated application tabs within the Student AI Hub. In contrast, general-purpose models, including Gemma 2, Phi4, and Llama, are available under the General AI tab.

While reasoning models are available, they have not been fully integrated into the application. Their unique reasoning processes require specialized formatting to ensure that the output is comprehensible to the user. As a result, this formatting was not completely finalized at the time of the initial release of the Student AI Hub.

8.5.11. Model Integration and Deployment

Following the evaluation process, the selected models were systematically integrated into the School AI Server and the Visual Studio Code extension.

Leveraging the user-friendly API provided by the Ollama application, we facilitated a seamless integration of the models into the School AI Server,

ensuring efficient accessibility and deployment. To enhance request management and optimize communication between different components, we developed a Python-based Flask server that hosts a dedicated API. This API serves as an intermediary layer, enabling structured and scalable interactions between the School AI Server and the Visual Studio Code extension.

A comprehensive discussion of the hosted Flask service, including its architecture and functionality, is presented in Chapter 9: *Hosted Flask Service*.

8.6. Integration of OpenAI's API

In this work, we integrated the OpenAI API to leverage proprietary, high-performance AI models that are hosted on dedicated servers with advanced hardware capabilities. The utilization of external computing power allows for the concurrent execution of multiple models, thereby enhancing both scalability and efficiency in our application.

The decision to adopt the OpenAI API was influenced by its widespread adoption, robust performance, and extensive documentation. Numerous examples, tutorials, and community resources are available, which greatly facilitate the integration process and ensure that best practices are followed in scientific and industrial applications.

8.6.1. Overview of the OpenAI API

The OpenAI API provides access to state-of-the-art AI models developed by OpenAI, including various iterations of the ChatGPT model. These models are capable of generating human-like text, answering queries, and engaging in complex conversations. The API supports a range of models with different sizes and capabilities, allowing users to select the model that best fits the requirements of their specific use cases.

Designed with user accessibility in mind, the API comes with comprehensive documentation and a wealth of code samples, which significantly

streamline the process of embedding advanced AI functionalities into diverse applications and platforms. Furthermore, the API utilizes a token-based pricing model, which charges users according to the number of tokens processed during interactions. This pricing structure is not only transparent but also aligns closely with the computational effort required to generate responses.

Before accessing the API's full functionality, users must pre-fund their accounts by depositing a specified amount of money. This account-based billing system enables users to manage their expenditures effectively, including the option to set monthly spending limits. In addition to text generation, the OpenAI ecosystem also includes DAL-E, an image-generation model that creates visuals based on textual input, thus broadening the spectrum of applications available through the API.

?

Tokens in Large Language Models

Tokens are the fundamental units of text that large language models (LLMs) process and generate. In this context, a token represents the smallest segment of text that a model can understand, which may correspond to an entire word, a fragment of a word, or even an individual character or punctuation mark.

The process of tokenization involves converting raw text into these discrete units. This approach enables LLMs to efficiently capture complex patterns in both syntax and semantics, even when encountering new or out-of-vocabulary terms. Techniques such as subword tokenization are particularly valuable, as they break down words into meaningful components, thereby reducing the overall vocabulary size and enhancing the model's ability to manage linguistic variability.

Moreover, tokens are closely related to the concept of a context window, which defines the span of tokens a model can consider during text generation or prediction. Typically, one token is estimated to average around four characters in English or roughly three-quarters of a word. This estimation is

crucial for determining computational requirements and understanding the limitations imposed by the model's finite context window.

In summary, tokens are indispensable for the operation of LLMs, providing a structured means to process language. Their effective management through advanced tokenization strategies is essential for optimizing both the computational efficiency and the overall performance of these models.

?

8.6.2. Data Security and Privacy in Compliance with Austrian and EU Regulations

The integration of OpenAI's API into our systems necessitates a thorough examination of data security and privacy considerations, particularly in the context of Austrian and European Union (EU) regulations. The General Data Protection Regulation (GDPR) serves as the cornerstone of data protection within the EU, imposing stringent requirements on the processing of personal data.

OpenAI has implemented several measures to safeguard user data and align with GDPR mandates. Notably, they support compliance with privacy laws such as the GDPR and the California Consumer Privacy Act (CCPA), offering a Data Processing Addendum to customers. Their API and related products have undergone evaluation by an independent third-party auditor, confirming alignment with industry standards for security and confidentiality.

?

Despite these measures, concerns have been raised regarding data handling practices. For instance, data transmitted through the OpenAI API could potentially be exposed, and compliance with GDPR remains a complex issue. Additionally, data may be accessible to third-party subprocessors, introducing further privacy considerations.

?

Gabriel Mrkonja
Florian Prandstetter
Luna P. I. Schätzle

To address these concerns, we have proactively informed our user community through a notice on the school website. This notice outlines the data handling practices associated with the OpenAI API and provides guidance on how users can manage their data when interacting with our systems. By maintaining transparency and offering clear instructions, we aim to uphold the highest standards of data security and privacy in our academic environment.

In light of the evolving regulatory landscape, it is imperative to remain vigilant and responsive to any changes in data protection laws within Austria and the broader EU. Continuous monitoring and adaptation of our data handling practices will ensure ongoing compliance and the safeguarding of user privacy.

8.6.3. OpenAI API Implementation in Vue.js

This section details the integration of the OpenAI API within a Vue.js application framework, with a focus on both text and image generation capabilities. The implementation not only illustrates the interaction between the Vue.js frontend and the OpenAI API but also demonstrates adherence to security best practices and modular code design. The following discussion is supported by annotated code examples and an explanation of the libraries used.

Overview of the Implementation

The implementation is structured as a Vue.js component that facilitates the following functionalities:

- Accepting user input via a text area.
- Initiating API calls for generating text responses (using ChatGPT models) and creating images (via the DALL-E endpoint).
- Displaying the results (generated text and images) dynamically within the user interface.

The component is designed with a clear separation between presentation and business logic, ensuring that the code remains both maintainable and scalable.

Explanation of the Used Libraries

OpenAI Library The openai library is employed as the primary interface to interact with OpenAI's API endpoints. This library abstracts the complexities of HTTP communication and provides a user-friendly API to access advanced AI functionalities such as natural language generation and image synthesis. Its integration simplifies the process of constructing API requests and handling responses, which is critical for developing robust AI-driven applications.

API Key Management To ensure secure handling of sensitive credentials, the OpenAI API key is imported from an external module (i.e., OPENAI_API_KEY from the secrets file). This approach adheres to security best practices by preventing the direct embedding of API keys within the source code, thereby mitigating the risk of unauthorized exposure.

Code Example: Vue.js Component for OpenAI API Integration

Below is an illustrative example of a Vue.js component that integrates the OpenAI API for both text and image generation. The code is presented in two parts: the HTML template and the JavaScript logic.

HTML Template

```
1 <template>
2   <div class="openai-container">
3     <h1>OpenAI API Integration in Vue.js</h1>
4     <textarea
5       v-model="userInput"
6       placeholder="Enter your prompt here..."
7       rows="4"
8       cols="50">
9   </textarea>
```

```

10   <div class="action-buttons">
11     <button @click="generateText">Generate Text</button>
12     <button @click="generateImage">Generate
13       Image</button>
14   </div>
15   <div v-if="generatedText" class="output-section">
16     <h2>Generated Text</h2>
17     <p>{{ generatedText }}</p>
18   </div>
19   <div v-if="generatedImage" class="output-section">
20     <h2>Generated Image</h2>
21     
23   </div>
</template>

```

Listing 8.5: Vue.js Template for OpenAI API Integration

JavaScript Logic

```

1 <script>
2 import OpenAI from "openai";
3 import { OPENAI_API_KEY } from "../secrets";
4
5 export default {
6   name: "OpenAIComponent",
7   data() {
8     return {
9       userInput: "",
10      generatedText: "",
11      generatedImage: ""
12    };
13  },
14  methods: {
15    async generateText() {
16      // Initialize OpenAI client with API key
17      const openai = new OpenAI({ apiKey: OPENAI_API_KEY
18        });
19      try {
20        const response = await
21          openai.chat.completions.create({
model: "gpt-3.5-turbo",
messages: [{ role: "user", content:
this.userInput }]

```

```

22      });
23      // Extract and assign the generated text
24      this.generatedText =
25          response.choices[0].message.content;
26    } catch (error) {
27      console.error("Error during text generation:",
28          error);
29    },
30    async generateImage() {
31      // Initialize OpenAI client for image generation
32      const openai = new OpenAI({ apiKey: OPENAI_API_KEY
33          });
34      try {
35        const response = await openai.images.generate({
36          prompt: this.userInput,
37          n: 1,
38          size: "512x512"
39        });
39        // Extract and assign the URL of the generated
40        // image
41        this.generatedImage = response.data[0].url;
42      } catch (error) {
43        console.error("Error during image generation:",
44            error);
45      }
46    }
46
</script>

```

Listing 8.6: Vue.js Script for OpenAI API Integration

Discussion

The presented component exemplifies how modern web applications can seamlessly integrate AI capabilities while maintaining a secure and modular architecture. Key points of consideration include:

- **Modularity:** The separation of the UI (HTML template) and the business logic (JavaScript methods) facilitates easier maintenance and potential scalability.

Gabriel Mrkonja
 Florian Prandstetter
 Luna P. I. Schätzle

- **Security:** By importing the API key from an external secrets module, the risk of credential leakage is minimized. This practice is crucial in academic and production environments where data security is paramount.
- **Extensibility:** The design allows for further expansion, such as additional error handling mechanisms or the integration of more advanced functionalities provided by the OpenAI API.

In conclusion, this integration not only demonstrates the practical application of AI APIs in modern web development but also reflects best practices in secure and maintainable code design. Such an approach is essential for building reliable applications in both academic research and industrial contexts.

8.7. Conclusion

The comprehensive evaluation of AI models for the Student AI Hub application has yielded valuable insights into their performance across various metrics, including response times, resource utilization, text quality, and overall suitability for diverse use cases. By integrating both quantitative and qualitative analyses, we have identified the strengths and weaknesses of each model, enabling us to formulate informed recommendations tailored to specific application scenarios.

Furthermore, the integration of the OpenAI API into the School AI Server and the Visual Studio Code extension has significantly enhanced the application's capabilities, providing users with seamless access to advanced AI functionalities. Adherence to best practices in data security and privacy has ensured full compliance with Austrian and EU regulations, thereby safeguarding user data.

In the following three chapters, we will discuss the technical implementation aspects of the project, including the integration of AI into the Flask server, the development of the Student AI Hub application, and the incorporation of AI functionalities into the Visual Studio Code extension. These chapters will provide a detailed exploration of the project's technical architecture,

complete with code examples, architectural diagrams, and deployment strategies.

9. hosted Flask Service

Author: Luna P. I. Schätzle

Author: Florian Prandstetter (docker section [9.8](#))

This chapter delineates the implementation, architecture, and deployment of the self-hosted Flask service, which functions as a pivotal interface between the front-end and back-end components of the system. In addition to detailing the technical design, the chapter critically examines the advantages of a self-hosted solution and the rationale behind key architectural decisions.

9.1. Introduction

This section offers a comprehensive overview of the motivations underpinning the development of the Flask service. Functioning as the backbone for both the Student AI Hub and the code extension's backend, the service was conceived to address a range of specific operational requirements. Here, we elaborate on the core functionalities of the service, detail the technical imperatives that drove its inception, and position its role within the broader system architecture, thereby laying the groundwork for subsequent technical discussions.

9.2. Advantages of a Self-hosted Service

A self-hosted service confers a multitude of benefits relative to externally managed or third-party solutions. This section examines these advantages in depth:

- **Enhanced Customization and Environmental Control:** By hosting the service internally, developers gain complete authority over the configuration and optimization of the operating environment. This control facilitates the implementation of domain-specific modifications and enables precise tuning to meet the unique needs of the project.
- **Rapid Prototyping and Agile Deployment:** The self-hosted nature of the service supports agile development practices. New features and bespoke functionalities can be rapidly prototyped, iteratively tested, and deployed, thereby significantly reducing development cycles and accelerating time-to-market.
- **Improved Data Security and Regulatory Compliance:** Hosting the service in-house allows for stringent oversight of data management practices. This approach is particularly advantageous in contexts governed by strict data protection regulations and institutional policies, as it enables the implementation of tailored security measures and enhances overall control over sensitive information.

Collectively, these factors validate the strategic decision to pursue a self-hosted approach, underscoring its technical, operational, and regulatory merits.

9.3. Architecture and Service Structure

The Flask service functions as a robust and flexible backend for the Student AI Hub and the code extension. Its modular and extensible design facilitates the seamless integration of additional functionalities and services. The figure below illustrates the key architectural components of the Flask service and their interactions.

9.3.1. System Architecture

The system is organized around a central Python script, `app.py`, which handles incoming requests, registers endpoints, initiates the server, and manages logging. Server configuration is maintained in `config.py`, where

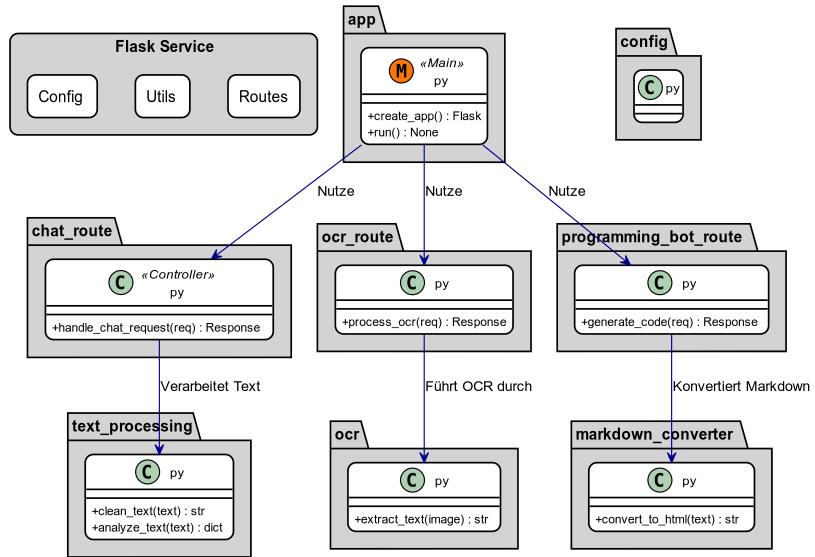


Abbildung 9.1.: Flask Service Architecture

settings such as the language model, image processing rules, upload folder, and conversion language are defined.

Endpoints are organized within the `routes` subfolder, with similar endpoints grouped together. They are divided into the following files:

- **chatbot_route.py**: Contains endpoints related to chatbot functionality.
- **Image_recognition_route.py**: Manages endpoints for image recognition tasks.
- **Programming_bot_route.py**: Houses endpoints for the programming bot.

Within these files, endpoints process user requests and return responses, and are designed for easy extension. They also handle error management and user feedback by converting data from the user and the Ollama API into the appropriate format when necessary. Connections to the Ollama API and the OCR (Optical Character Recognition) system are managed within the `Utils` folder, which contains all utility functions used by the endpoints, including those for sending requests, performing OCR, and converting Markdown to HTML.

9.3.2. Expandability and Modularity

The server is designed for easy expandability and modularity. Endpoints are stored in separate files, which allows for the straightforward addition of new functionality. Similarly, utility functions can be updated or replaced with minimal effort, and the server configuration is flexible enough to accommodate new parameters.

This modular and extensible architecture ensures that the Flask service can readily adapt to future enhancements and integrations. These design principles contribute to a flexible, maintainable server that can be effortlessly updated with new features and functionalities.

9.4. Flask as a Web Framework

9.4.1. Core Functionalities of Flask

This subsection explains the intrinsic capabilities of Flask, such as request routing, templating, and middleware support. It illustrates how these features are employed to manage web requests and responses within the service.

9.4.2. Rationale for Selecting Flask

A critical discussion is presented on why Flask was chosen for the project. Key factors include its simplicity, extensive documentation, and robust community support, which collectively make it an ideal framework for rapid development and prototyping.

9.5. RESTful Endpoints and Functionalities

9.5.1. Endpoint Specifications

This section enumerates the various RESTful endpoints implemented in the service. Each endpoint is described in detail, outlining its specific purpose and the type of data it handles.

9.5.2. Code Illustrations

To enhance understanding, code examples are provided to demonstrate the implementation of key endpoints. These examples highlight the methods used to process requests and generate responses.

9.5.3. Utility Functions

An overview of auxiliary utility functions is given, focusing on their roles in logging, data validation, and error handling. These functions contribute to the overall robustness and reliability of the service.

9.6. Utilized Libraries

A comprehensive inventory of the external libraries used in the project is presented in this section. For each library, its functionality, role within the project, and integration aspects are discussed.

Gabriel Mrkonja
Florian Prandstetter
Luna P. I. Schätzle

9.7. Deployment

9.8. Docker

Docker is a platform that allows you to run applications in containers. A container is like a small, isolated environment where software runs with everything it needs – including the operating system, libraries, and dependencies.

No matter which computer or server the container runs on, it always works the same way. This means you don't have to worry about an application suddenly throwing errors on a different system just because a different software version is installed there.

Docker is often used in software development and cloud applications because it simplifies testing, deployment, and scaling of apps. Developers can store their software as images and share them with others without requiring complicated installations.

9.8.1. Used Docker Images

A docker image is a blueprint that specifies how to run the application. The instructions for the build are stored in the Dockerfile. ?

- **flask_app** The Flask image is used to easily implement the Flask application in a Docker container.
- **ollama** The Ollama image is used to avoid running LLMs globally and use them in a secluded environment.

9.8.2. Docker Compose

Docker Compose is used for running multiple containers at the same time. It simplifies your application and makes it easier to manage. The Configuration is stored in a single YAML file. All the services can be started with a simple command. It is a very compact way to manage Docker application. ?

9.9. Scalability and Performance Concerns

One notable limitation of the Flask server is its inherent lack of scalability. Flask, being primarily designed for lightweight applications, is not optimized for handling high volumes of concurrent requests. In our implementation, the server was deployed on a modest PC with limited computational resources. Consequently, if the service were to be deployed in a production environment, it would be imperative to migrate to more robust hardware or consider a distributed, multi-server architecture to effectively manage the anticipated load. Given the constraints of the project timeline and the prototype nature of this work, scalability was not prioritized during development.

9.10. Conclusion and Future Work

This concluding section synthesizes the chapter's key points and reflects on the efficacy of the implemented service. It also outlines potential avenues for future enhancements, such as further scalability improvements, additional functionalities, and more robust deployment automation.

10. Intelligent Student AI Hub: An Integrated Learning Platform

Authors: Luna P. I. Schaetzle

This chapter presents an in-depth overview of the Intelligent Student AI Hub, a comprehensive web platform designed to empower students in exploring artificial intelligence (AI) concepts. The platform integrates state-of-the-art technologies and innovative features to facilitate both learning and practical experimentation in AI. The following sections detail the system architecture, core functionalities, and future directions for this educational tool.

10.1. Introduction

The Intelligent Student AI Hub provides a robust environment for students to learn about AI and its real-world applications. It offers a diverse range of educational resources—including articles, tutorials, and interactive tools—to foster a deep understanding of AI concepts. By combining engaging content with advanced technological integration, the platform aims to make AI accessible, dynamic, and relevant to learners at all levels.

10.2. System Architecture and Technologies

This section outlines the principal technologies that form the backbone of the Intelligent Student AI Hub. By employing a combination of modern

web frameworks and cloud-based services, the platform achieves a secure, scalable, and high-performance architecture.

10.2.1. Vue.js

The frontend of the platform is primarily developed using Vue.js—a progressive JavaScript framework renowned for its component-based structure and reactive data binding. Utilizing standard web technologies such as HTML, CSS, and JavaScript, Vue.js enables the creation of dynamic, single-page applications that are both modular and easy to maintain. For additional details on the implementation of Vue.js, please refer to Chapter 7, subsection "Vue.js."

10.2.2. Flask API

The backend infrastructure is powered by a custom-developed Flask API. This API manages client requests and facilitates communication with a suite of self-hosted AI models and tools. Through efficient data handling and secure request management, the Flask API forms a critical link between the frontend interface and the underlying AI services. More comprehensive insights into the backend architecture are available in Chapter 9.

10.2.3. ChatGPT API

To augment the platform's interactive capabilities, the Intelligent Student AI Hub integrates the ChatGPT API via the OpenAI library. This integration supports a sophisticated chatbot feature that enables students to ask questions and receive detailed, context-aware responses on a variety of AI-related topics. Further information on this integration can be found in Chapter 8, subsection "Integration of OpenAI's API."

10.2.4. Firebase for Authentication and Data Storage

For secure user management and efficient data handling, the platform employs Firebase services. Firebase Authentication provides a flexible and robust solution for verifying user identities through multiple sign-in methods—including email/password, third-party providers, and anonymous authentication. Additionally, Firebase's real-time database and Cloud Firestore facilitate scalable and responsive data storage, synchronization, and retrieval. The seamless integration of Firebase with Vue.js components ensures that user data and authentication states are managed in real time, enhancing both security and user experience.

?

10.3. Core Functionalities

To get a comprehensive understanding of the Intelligent Student AI Hub, this section delves into its core functionalities and interactive features of the end product. Some of the key features of the platform include:

- **Interactive Chatbot for AI Questions:** The platform hosts an AI-powered chatbot that can answer a wide range of AI-related queries, providing students with instant access to information and explanations.
- **OpenAI Integration:** By integrating OpenAI's cutting-edge models, such as ChatGPT and DALL-E, the platform offers advanced AI capabilities for generating text and images, enhancing the learning experience.
- **Programming Bot for Different Languages:** A specialized bot is available to assist students in learning and practicing various programming languages, offering code snippets, explanations, and interactive coding exercises.
- **Image Recognition Tool:** The platform includes an image recognition tool that leverages AI algorithms to identify objects, scenes, and patterns within uploaded images, enabling students to explore computer vision concepts.

Gabriel Mrkonja
Florian Prandstetter
Luna P. I. Schätzle

- **Image-to-Text Tool:** Students can utilize an image-to-text tool that converts text embedded within images into editable and searchable content, facilitating the extraction of information from visual data.
- **Saved Chats:** The platform allows users to save and revisit previous chat interactions with the AI chatbot, enabling seamless continuity in learning and knowledge retention.
- **User Profiles and Authentication:** Each user can create a personalized profile, manage their learning progress, and access customized content based on their preferences and history.

10.4. Authentication and User Profiles

For the Intelligent Student AI Hub, Firebase is a cornerstone technology for managing user authentication and profile creation, ensuring both secure access and a personalized user experience.

10.4.1. Why Firebase?

Although numerous alternatives exist for user authentication and data management, Firebase distinguishes itself through its comprehensive feature set, seamless integration, and robust security measures. Additionally, the abundance of tutorials and detailed documentation facilitates a swift onboarding process for the development team.

10.4.2. Firebase Authentication Factors

Implementing robust authentication and user profile management involves several critical aspects:

- **Firebase Authentication:** The platform leverages Firebase Authentication to facilitate secure user sign-in and verification. By supporting multiple authentication methods—including email/password,

social logins (e.g., Google, Facebook), and anonymous authentication—Firebase offers a versatile solution that adapts to diverse user needs while ensuring a seamless and reliable experience.

- **Real-time Data Synchronization:** Utilizing Firebase's Realtime Database and Cloud Firestore, the platform ensures that user data is consistently synchronized across all devices. This real-time updating mechanism provides immediate access to personalized content, settings, and user profiles, thus significantly enhancing user engagement.
- **Secure Data Handling:** Firebase incorporates robust security measures, including data encryption, secure authentication tokens, and finely tuned access control rules. These features work together to protect user data from unauthorized access, maintaining both data integrity and user privacy in accordance with best practices and regulatory requirements.
- **Integration with Vue.js Components:** The tight integration between Firebase and Vue.js enables dynamic data binding and responsive user interfaces. Leveraging Vue.js reactivity in combination with Firebase's real-time updates results in a fluid user experience, where UI elements automatically refresh to reflect the most current state of user data.
- **Future Enhancements:** As the platform evolves, additional features such as recommendation engines, learning analytics, and collaborative learning tools could be integrated. These enhancements would further tailor content to individual user needs and foster a more engaging and personalized educational environment.

10.4.3. Firebase Integration with Vue.js

The integration of Firebase services within Vue.js is essential to achieving a seamless, interactive user experience on the Intelligent Student AI Hub. The process involves several key steps:

- **Installing the Firebase SDK:** The Firebase JavaScript SDK is added to the Vue.js project via package managers like npm or yarn, providing access to Firebase's suite of services directly within the application.
- **Initializing Firebase:** The SDK is initialized using project-specific configuration settings, including API keys, authentication methods,

Gabriel Mrkonja
Florian Prandstetter
Luna P. I. Schätzle

and database URLs. This step establishes a secure connection between the Vue.js application and Firebase services.

- **Implementing Authentication:** Vue.js components integrate Firebase Authentication methods to handle various sign-in options. These components are responsible for managing user sessions and ensuring secure access to personalized content and features.
- **Managing User Profiles:** User-specific data—such as preferences, settings, and learning progress—is stored in Firebase databases. Vue.js components interact with these services to create, update, and retrieve profiles, with real-time synchronization ensuring that updates are reflected immediately across all user devices.
- **Handling Real-time Updates:** Vue.js reactivity is combined with Firebase's real-time data listeners. This ensures that any changes in user data trigger immediate UI updates, thereby providing a consistently accurate and current view of the user's profile and settings.
- **Implementing Security Rules:** Firebase security rules are configured to enforce strict access control policies. By restricting read and write permissions to authenticated users only, these rules help maintain data integrity and protect user privacy.

For the integration process, the VueJS Firebase library is utilized, streamlining the connection between Vue.js projects and Firebase. This library simplifies access to numerous Firebase features—including Authentication, Realtime Database, Firestore, Storage, and restricted pages for non-authenticated users—making it easier to implement a secure and efficient system.

10.4.4. Implementation of Firebase Authentication

A robust implementation of Firebase Authentication within Vue.js involves both proper configuration and thoughtful component design. The following code snippets illustrate key aspects of this integration.

Firebase Initialization and Authentication Setup:

Gabriel Mrkonja

Florian Prandstetter

Luna P. I. Schätzle

```

1 import firebase from 'firebase/app';
2 import 'firebase/auth';
3
4 // Firebase configuration object containing keys and
5 // identifiers
6 const firebaseConfig = {
7   apiKey: "YOUR_API_KEY",
8   authDomain: "YOUR_PROJECT_ID.firebaseio.com",
9   databaseURL: "https://YOUR_PROJECT_ID.firebaseio.com",
10  projectId: "YOUR_PROJECT_ID",
11  storageBucket: "YOUR_PROJECT_ID.appspot.com",
12  messagingSenderId: "YOUR_SENDER_ID",
13  appId: "YOUR_APP_ID"
14};
15
16 // Initialize Firebase with the configuration
17 firebase.initializeApp(firebaseConfig);
18
19 // Export the authentication module for use in Vue
20 // components
21 export const auth = firebase.auth();
22
23 // Monitor authentication state changes
24 auth.onAuthStateChanged(user => {
25   if (user) {
26     // User is signed in; update application state
27     // accordingly
28     console.log('User signed in:', user);
29   } else {
30     // User is signed out; update the UI to reflect
31     // sign-out state
32     console.log('No user is signed in.');
33   }
34 });

```

Listing 10.1: Initializing Firebase and setting up authentication

Explanation:

- **Firebase Import and Configuration:** The Firebase modules are imported, and the application is initialized using a configuration object that

contains the necessary API keys and identifiers. This setup establishes the connection to Firebase services.

- **Authentication Monitoring:** The `onAuthStateChanged` listener is used to monitor changes in the user's authentication state. This enables the application to dynamically update its interface in response to sign-in or sign-out events.

Vue.js Component Example with Authentication:

```

1 <template>
2   <div>
3     <!-- Display a welcome message if the user is
4       signed in -->
5     <h2 v-if="user">Welcome, {{ user.email }}</h2>
6     <!-- Otherwise, show the sign-in form -->
7     <div v-else>
8       <input v-model="email" placeholder="Email" />
9       <input v-model="password" type="password"
10          placeholder="Password" />
11      <button @click="signIn">Sign In</button>
12      <button @click="signInWithGoogle">Sign In with
13        Google</button>
14      <p v-if="errorMessage" class="error">{{
15        errorMessage }}</p>
16    </div>
17  </div>
18 </template>
19
20 <script>
21 import { auth } from '@firebase'; // Adjust the path
22   according to your project structure
23 import firebase from 'firebase/app';
24 import 'firebase/auth';
25
26 export default {
27   data() {
28     return {
29       email: '',
30       password: '',
31       user: null,
32       errorMessage: ''
33     };
34   },
35 }

```

```

30     created() {
31         // Listen for authentication state changes and
32         // update the component state
33         auth.onAuthStateChanged(user => {
34             this.user = user;
35         });
36     },
37     methods: {
38         signIn() {
39             // Attempt to sign in using the provided email
40             // and password
41             auth.signInWithEmailAndPassword(this.email,
42                 this.password)
43                 .then(credential => {
44                     this.user = credential.user;
45                     this.errorMessage = '';
46                 })
47                 .catch(error => {
48                     // Handle authentication errors by
49                     // updating the errorMessage state
50                     this.errorMessage = error.message;
51                     console.error("Authentication error:", error);
52                 });
53         },
54         signInWithGoogle() {
55             const provider = new
56                 firebase.auth.GoogleAuthProvider();
57             auth.signInWithPopup(provider)
58                 .then(result => {
59                     this.user = result.user;
60                     this.errorMessage = '';
61                 })
62                 .catch(error => {
63                     this.errorMessage = error.message;
64                     console.error("Google sign-in error:", error);
65                 });
66     }
67 };
68 </script>
69 <style scoped>

```

```
67 | .error {  
68 |     color: red;  
69 |     font-size: 0.9em;  
70 | }  
71 | </style>
```

Listing 10.2: Vue.js component for user sign-in

Explanation:

- **Conditional Rendering:** The template uses Vue.js directives (`v-if` and `v-else`) to conditionally display content based on whether a user is authenticated. A personalized welcome message is shown when the user is signed in, while a sign-in form is presented otherwise.
- **Data Binding and State Management:** The component's data properties (`email`, `password`, `user`, and `errorMessage`) are used to manage form inputs, the authenticated user state, and error messages.
- **Sign-In Method:** The `signIn` method invokes Firebase Authentication's `signInWithEmailAndPassword` function. Proper error handling is implemented to provide feedback to the user in case of sign-in failures.
- **Real-time Authentication Updates:** The `onAuthStateChanged` listener, set up in the created hook, ensures that the component's state is kept in sync with the authentication status, thereby reflecting any changes immediately in the UI.
- **Google Sign-In:** The `signInWithGoogle` method demonstrates how to enable Google sign-in using Firebase's `GoogleAuthProvider`. This method follows a similar pattern to the email/password sign-in process.
- **Styling and Error Handling:** The component includes scoped styles for error messages and provides visual feedback to users when authentication errors occur.

Best Practices and Future Considerations:

- **Error Handling and User Feedback:** Robust error handling is essential for providing clear user feedback and maintaining a secure application environment.
- **Scalability and Maintainability:** Modularizing the Firebase configuration and authentication logic allows for easier maintenance and future feature integrations, such as multi-factor authentication.

- **Security Enhancements:** Implementing advanced security measures, such as multi-factor authentication and periodic token refresh, can further enhance the platform's security posture.

Through these implementations, the Intelligent Student AI Hub not only provides secure authentication and personalized user experiences but also lays the groundwork for future enhancements in user engagement and data security.

10.4.5. User Overview and Personalization

To enhance user engagement, the Intelligent Student AI Hub provides a personalized overview of each user's profile. This dedicated account management page consolidates essential information—including profile details, learning progress, and tailored recommendations—into a central hub that facilitates the management of user settings, preferences, and overall platform interactions.

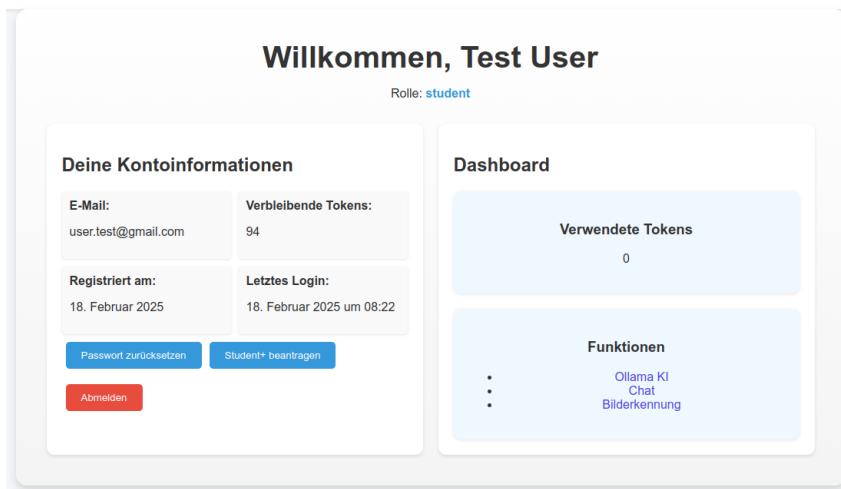


Abbildung 10.1.: User Account Management and Overview

10.4.6. Outlook for Account Management

Looking forward, there is considerable potential to expand the account management functionality with advanced features, such as:

- **Enhanced Personalization:** Integration of sophisticated personalization tools, including dynamic content recommendations, curated learning paths, and detailed progress tracking, to provide a more tailored and effective learning experience.
- **Premium Account Options:** Introduction of premium account tiers that unlock advanced features and increase the allocation of request tokens for the ChatGPT API.
- **Social Integration:** Implementation of social login options, content sharing functionalities, and collaborative learning tools to foster a more interactive and community-driven environment.
- **Teacher Functionality:** Development of specialized tools for educators, enabling them to better manage classroom interactions and support student learning.
- **Administrator Dashboard:** Creation of a comprehensive administrative interface for efficient management of user accounts, content moderation, and platform analytics, thereby streamlining oversight and enhancing operational efficiency.

10.4.7. TSN Integration

Every student at HTL is provided with a TSN email account, which serves as the primary communication channel within the institution. During the development of the Student AI Hub, integrating the TSN email account was considered as a potential feature. However, after careful evaluation, we decided against this integration for several critical reasons:

- **Security Concerns:** Incorporating the TSN email account would necessitate accessing sensitive user data. Without robust safeguards, this could significantly increase the risk of security breaches and data leakage.

- **Technical Complexity:** The integration would require the implementation of more sophisticated authentication mechanisms than those offered by Firebase. This added complexity could result in compatibility issues and pose significant challenges in terms of ongoing maintenance and support.
- **Impact on User Experience:** Requiring users to navigate additional authentication steps to access the platform could negatively affect the overall user experience. A more complicated login process may lead to reduced adoption rates and lower user satisfaction.
- **Regulatory and Compliance Challenges:** Ensuring compliance with data protection regulations and institutional policies would be more demanding with the TSN email integration. This approach would require addressing additional legal and technical considerations to maintain adherence to relevant standards.

10.5. Interactive Chatbot for Day-to-Day AI Questions

The objective of this feature is to provide students with immediate, context-aware responses to a broad spectrum of AI-related inquiries. To achieve this, the Intelligent Student AI Hub integrates multiple advanced Ollama AI models (e.g., LLaMA 3.2 and Mistral), enabling users to select the model that best fits the complexity and responsiveness required by their query. This modular approach ensures that students receive the most effective and contextually relevant answers, thereby enhancing their learning experience.

A self-hosted Flask API serves as the intermediary between the user interface and the Ollama API. This architecture allows the system to capture user input, forward the request to the selected AI model via the Flask API, and then relay the model's response back to the user in real time. The following abbreviated code listing illustrates the core implementation within a Vue.js component, demonstrating how the integration is achieved:

```
1 <template>
2   <div>
```

Gabriel Mrkonja
Florian Prandstetter
Luna P. I. Schätzle

```

3      <!-- AI Model Selection -->
4      <select v-model="selectedModel">
5          <option value="llama3.2:1b">LLaMA 3.2 - 1B
6              (Fast)</option>
7          <option value="llama3.2">LLaMA 3.2 - 2B
8              (Latest)</option>
9          <!-- more Models -->
10     </select>
11     <!-- Chat Display -->
12     <div v-for="msg in currentChat.messages" :key="msg.id">
13         <p v-if="msg.type === 'user'>{{ msg.text }}</p>
14         <p v-else>{{ msg.text }}</p>
15     </div>
16     <!-- User Input and Submission -->
17     <input v-model="userInput"
18             @keydown.enter="sendMessage" placeholder="Ask the
19             AI question..." />
20     <button @click="sendMessage">Send</button>
21 </div>
22 </template>
23
24 <script>
25 import axios from 'axios';
26 export default {
27     data() {
28         return {
29             userInput: '',
30             selectedModel: 'llama3.2:1b',
31             currentChat: { messages: [] },
32         };
33     },
34     methods: {
35         async sendMessage() {
36             if (!this.userInput.trim()) return;
37             // Append user message to chat
38             this.currentChat.messages.push({ id: Date.now(),
39                 type: 'user', text: this.userInput });
40             // Send the query to the Flask API
41             const response = await
42                 axios.post('http://server-address/ask_ollama', {
43                     prompt: this.userInput,
44                     model: this.selectedModel,
45                 });
46             // Append AI response to chat

```

```
41      this.currentChat.messages.push({ id: Date.now() ,
42          type: 'ollama', text:
43              response.data.choices[0].text });
44      this.userInput = '';
45  },
46};
```

Listing 10.3: Abbreviated Vue.js Integration Example

This example demonstrates the fundamental components of the integration:

- **Model Selection:** A dropdown menu allows users to choose from various AI models, balancing speed and sophistication.
- **Real-Time Communication:** User inputs are captured and transmitted asynchronously to the Flask API, which then retrieves responses from the selected Ollama AI model.
- **Dynamic Chat Interface:** The chat interface updates dynamically with both user queries and AI responses, ensuring an engaging, real-time interaction.

By decoupling the frontend from the backend AI processing via a RESTful API, this design not only simplifies maintenance but also facilitates future scalability. New models or enhanced features can be integrated with minimal changes to the existing codebase, ensuring the platform remains adaptable to evolving educational needs.

10.6. OpenAI Integration

Given that locally hosted Ollama AI models may not achieve the same performance level as commercially available state-of-the-art solutions, the Intelligent Student AI Hub integrates advanced OpenAI models—such as ChatGPT and DALL-E—to deliver superior capabilities in text generation and image synthesis. This integration not only augments the quality of responses but also significantly enhances platform scalability. With the OpenAI API, scalability is effectively decoupled from hardware limitations,

Gabriel Mrkonja
Florian Prandstetter
Luna P. I. Schätzle

unlike the self-hosted Ollama API, which is inherently constrained by the available computational resources.

10.6.1. ChatGPT API and Its Limitations

A primary limitation of the ChatGPT API is the cost associated with each API request. Every query incurs a fee that can rapidly accumulate with high usage volumes.

GPT-4o	GPT-4o mini
High-intelligence model for complex tasks 128k context length	Affordable small model for fast, everyday tasks 128k context length
Price	Price
Input: \$2.50 / 1M tokens	Input: \$0.150 / 1M tokens
Cached input: \$1.25 / 1M tokens	Cached input: \$0.075 / 1M tokens
Output: \$10.00 / 1M tokens	Output: \$0.600 / 1M tokens

Abbildung 10.2.: ChatGP API Pricing

?

For instance, utilizing ChatGPT-4o Mini costs \$0.30 per million input tokens, whereas the full ChatGPT-4o model incurs a cost of \$3.75 per million input tokens equating to a 12.5-fold increase in expense (see Figure 10.2). ¹

Consequently, the development team has opted to restrict the use of the ChatGPT API. This measured approach enables students to benefit from ChatGPT's advanced functionalities while effectively managing costs. Moreover, it is more economical for the institution to subsidize access to the API than to provide every student with an individual premium account.

¹Pricing data is current as of 17.02.2025 and may be subject to change.

10.6.2. User Access to Paid Services

Standard users are allocated a finite number of ChatGPT API requests per month, with these request tokens being replenished on a monthly basis. The number of tokens consumed per query is contingent upon the chosen model.

² Should a user exhaust their monthly token quota, they may alternatively direct their queries to the Ollama API. In addition to standard user access, premium, teacher, and administrator accounts are available, each benefiting from a higher monthly token allocation. Initially, all users are granted standard access; any desired upgrade to premium, teacher, or administrator status requires a formal request to the platform administration.

10.6.3. Integration of OpenAI's API

For a comprehensive guide on integrating OpenAI's API into a Vue.js project, please refer to Chapter 8, Section 8.6.3.

10.7. Programming Bot for Different Programming Languages

The Intelligent Student AI Hub incorporates a dedicated programming bot designed to facilitate the learning and practice of various programming languages. Building upon the foundational architecture of the general chatbot, this bot leverages code-centric large language models (LLMs) that have been specifically trained on source code. As with the standard chatbot, users can select from different models that are optimized for programming-related tasks.

²Token allocations and model thresholds are periodically adjusted in response to current pricing structures and monthly usage limits.

10.7.1. Programming Bot Features

Several enhancements have been integrated into the programming bot to improve its functionality and user experience:

- **Programming Language Selection:** A dropdown menu enables users to specify the programming language for which they require assistance. The selected language is incorporated into the user's query—modifying the prompt sent to the LLM—to ensure that responses are tailored appropriately. This modification is handled on the backend via a dedicated Flask API endpoint.
- **Prompt Refinement:** An integrated "refine" option allows users to modify their initial query. By clicking the refine button, users can adjust their question, prompting the LLM to generate a more precise response.
- **Code Rendering and Clipboard Functionality:** The frontend renders responses in Markdown, which facilitates automatic syntax highlighting of code blocks. Additionally, a "copy to clipboard" feature is provided, enabling users to easily extract and reuse the code samples.

Further details regarding the backend implementation are provided in Chapter 9.

Markdown for Code Formatting

Markdown is a lightweight markup language that supports plain-text formatting and can be easily converted into various output formats. Given that most LLM responses are delivered in Markdown, this feature simplifies the rendering of code with syntax highlighting. This approach is particularly useful for displaying well-formatted code snippets alongside explanatory text ?.

Illustrative Implementation Example: The following abbreviated Vue.js component demonstrates the key aspects of the programming bot integration. This example illustrates how the user can select a programming language, refine their input, and receive formatted code output:

```

1 <template>
2   <div class="programming-bot">
3     <!-- Selection Area for Model and Programming Language
4       -->
5     <div class="selection-area">
6       <select v-model="selectedModel">
7         <option value="modelA">Model A</option>
8         <option value="modelB">Model B</option>
9       </select>
10      <select v-model="selectedLanguage">
11        <option value="python">Python</option>
12        <option value="java">Java</option>
13      </select>
14    </div>
15    <!-- Chat Interface -->
16    <div class="chat-box">
17      <div v-for="msg in messages" :class="msg.type">{{ msg.text }}</div>
18    </div>
19    <!-- Input Area with Refine Option -->
20    <textarea v-model="userInput" placeholder="Enter your
21      programming question..."></textarea>
22    <button @click="sendMessage">Send</button>
23    <button v-if="isLastUserMessage"
24      @click="prepareRefine">Refine</button>
25  </div>
26</template>
27
28<script>
29 export default {
30   data() {
31     return {
32       userInput: '',
33       selectedModel: 'modelA',
34       selectedLanguage: 'python',
35       messages: [],
36     };
37   },
38   methods: {
39     async sendMessage() {
40       // Append the selected programming language to the
41       // user prompt
42       const prompt = `Language:

```

```
39         `${this.selectedLanguage}\n${this.userInput}`;  
40     // Send the prompt to the Flask API and process the  
41     // response...  
42     },  
43     prepareRefine() {  
44         // Open a modal to refine the user prompt for  
45         // improved accuracy  
46     }  
47 };  
48 </script>
```

Listing 10.4: Abbreviated Vue.js Component for the Programming Bot

This concise example encapsulates the core integration features: selecting a programming model and language, refining user input, and rendering code responses with Markdown-enhanced formatting.

10.8. Image Recognition Tool

The Intelligent Student AI Hub incorporates an image recognition feature by leveraging the Ollama API. This functionality allows users to upload images that are subsequently processed by a dedicated endpoint on the Flask API. Detailed information on the backend implementation is provided in Chapter 9. On the front end, a Vue.js component has been developed to facilitate image uploads and transmit them, along with user-provided prompts, to the Flask API for analysis.

10.8.1. Implementation of the Image Recognition Tool

The following abbreviated code listing illustrates the key elements of the Vue.js component responsible for handling image uploads and processing the API responses. This example provides an overview of how the component captures a text prompt, manages image upload (by converting the image to a Base64 string), and displays the response from the Flask backend.

```

1 <template>
2   <div class="image-recognition">
3     <h1>Upload Image and Send to Ollama</h1>
4     <!-- Text prompt input -->
5     <input v-model="userPrompt" placeholder="Enter a
       prompt..." @keydown.enter="sendRequest" />
6     <!-- File input for image upload -->
7     <input type="file" @change="handleImageUpload" />
8     <!-- Submission button -->
9     <button @click="sendRequest">Submit</button>
10    <!-- Status and response display -->
11    <div v-if="loading">Sending request...</div>
12    <div v-if="error">{{ error }}</div>
13    <div v-if="response">
14      <h3>Ollama Response:</h3>
15      <p>{{ response }}</p>
16    </div>
17  </div>
18 </template>
19
20 <script>
21 export default {
22   data() {
23     return {
24       userPrompt: "",
25       imageData: "",
26       loading: false,
27       error: "",
28       response: null
29     };
30   },
31   methods: {
32     handleImageUpload(event) {
33       const file = event.target.files[0];
34       const reader = new FileReader();
35       reader.onload = () => {
36         // Extract the Base64-encoded string from the data
37         // URL
38         this.imageData = reader.result.split(",")[1];
39       };
40       if (file) reader.readAsDataURL(file);
41     },
42     async sendRequest() {

```

```
42      if (!this.userPrompt || !this.imageData) {
43          this.error = "Both prompt and image are required!";
44          return;
45      }
46      this.loading = true;
47      // Send the prompt and image data to the Flask API
48      // (request details omitted)
49      // e.g., using axios.post(url, { prompt:
50      //     this.userPrompt, image: this.imageData })
51      // Process the response and update this.response
52      // accordingly.
53      this.loading = false;
54  }
55 }
56 };
57 </script>
```

Listing 10.5: Abbreviated Vue.js Component for Image Recognition

In this implementation, the component first captures a user-defined text prompt and an image file. The image is converted into a Base64 string to facilitate secure and efficient data transmission. Once both inputs are validated, the component sends an HTTP request to the Flask API. The response from the backend typically a textual analysis or description generated by the Ollama API is then displayed to the user. This approach ensures a seamless and interactive experience for users engaging with the image recognition functionality.

10.9. Image to Text Tool

The Intelligent Student AI Hub incorporates a dedicated image-to-text tool designed to extract textual information from images. This feature first utilizes a text-to-image model to perform optical character recognition (OCR) on uploaded images. Once the raw text is extracted, a large language model (LLM) optimizes the content to enhance readability and clarity. The refined text is then presented in a clean, easily accessible format, and users have the option to copy the text to their clipboard for further use.

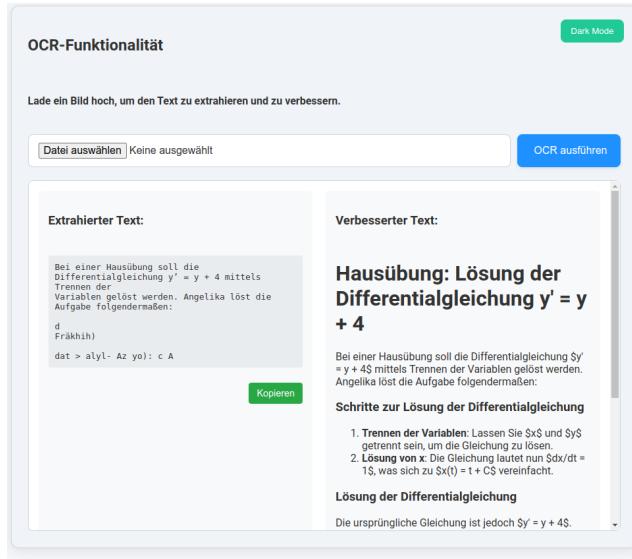


Abbildung 10.3.: Image to Text Tool

Figure 10.3³ illustrates the user interface of the image-to-text tool, highlighting both the image upload mechanism and the display of the extracted text.

The following abbreviated code listing provides a high-level overview of the Vue.js component responsible for handling image uploads, invoking the OCR process via a Flask API, and displaying the processed text:

```

1 <template>
2   <div class="ocr-component">
3     <h2>OCR Functionality</h2>
4     <!-- Prompt Input & Image Upload -->
5     <input v-model="prompt" placeholder="Enter a
       prompt..." @keydown.enter="sendRequest" />
6     <input type="file" @change="handleImageUpload"
           accept="image/*" />
7     <button @click="sendRequest" :disabled="!selectedImage
       || loading">
8       Submit
9     </button>

```

³The image-to-text tool is in German because the platform is designed for students at HTL in Austria.

```

10     <!-- Status and Result Display -->
11     <div v-if="loading">Processing image...</div>
12     <div v-if="error">{{ error }}</div>
13     <div v-if="rawText">
14         <h3>Extracted Text:</h3>
15         <pre>{{ rawText }}</pre>
16         <button @click="copyText(rawText)">Copy</button>
17     </div>
18 </div>
19 </template>
20
21 <script>
22 export default {
23     data() {
24         return {
25             prompt: "",
26             selectedImage: null,
27             rawText: "",
28             loading: false,
29             error: ""
30         };
31     },
32     methods: {
33         handleImageUpload(event) {
34             const file = event.target.files[0];
35             if (file) this.selectedImage = file;
36         },
37         async sendRequest() {
38             if (!this.prompt || !this.selectedImage) {
39                 this.error = "Both prompt and image are required.";
40                 return;
41             }
42             this.loading = true;
43             // Create FormData and send request to the Flask API
44             // Response processing updates rawText with
45             // extracted and optimized text
46             this.loading = false;
47         },
48         copyText(text) {
49             navigator.clipboard.writeText(text);
50         }
51     }
52 }</script>
```

Listing 10.6: Abbreviated Vue.js Component for Image-to-Text Conversion

This Vue.js component encapsulates the core functionality of the image-to-text tool, enabling users to upload images, extract text content, and copy the processed text for further use. By combining OCR capabilities with large language models, the platform delivers a powerful and user-friendly tool for extracting textual information from images.

For detailed backend implementation of this tool, please refer to Chapter 9, Section 9.5.

10.10. Saved Chats

To enhance user experience, the Intelligent Student AI Hub includes a feature that enables users to save their chat interactions with the AI chatbot. This functionality allows users to revisit previous conversations, review the information provided by the AI, and continue their learning journey from where they left off. To achieve this, chat data is stored in Firebase's Firestore database, ensuring scalable and secure management of user interactions.

10.10.1. Implementation of the Saved Chats Feature

The following abbreviated code snippet provides an overview of how chat messages are stored and retrieved from Firestore. This example demonstrates the core functionality, including loading the list of saved chats, displaying the current chat, and saving updates to Firestore.

```
1 <template>
2   <div class="chat-container">
3     <!-- Chat Window -->
4     <div class="chat-window" v-if="currentChat">
5       <div v-for="msg in currentChat.messages"
6           :key="msg.id" :class="msg.type">
7         <p v-if="msg.type === 'user'">{{ msg.text }}</p>
```

```

7      <div v-else
8          v-html="renderMarkdown(msg.text)"></div>
9      </div>
10     <!-- Sidebar for Saved Chats -->
11     <aside class="chat-sidebar">
12         <ul>
13             <li v-for="chat in chats" :key="chat.id"
14                 @click="loadChat(chat.id)">
15                 {{ chat.name }}
16             </li>
17             <button @click="startNewChat">+ New Chat</button>
18         </ul>
19     </aside>
20 </div>
21 </template>
22 <script>
23 import firebase from 'firebase/app';
24 import 'firebase/firestore';
25
26 export default {
27     data() {
28         return {
29             chats: [],
30             currentChat: null
31         };
32     },
33     methods: {
34         async loadChatList() {
35             const snapshot = await
36                 firebase.firestore().collection('chats').get();
37             this.chats = snapshot.docs.map(doc => ({ id: doc.id,
38                 ...doc.data() }));
39         },
40         async loadChat(chatId) {
41             const doc = await
42                 firebase.firestore().collection('chats').doc(chatId).get();
43             this.currentChat = { id: doc.id, ...doc.data() };
44         },
45         async saveChat() {
46             if (this.currentChat) {
47                 await
48                     firebase.firestore().collection('chats').doc(this.currentChat.id).set({
49                         ...this.currentChat
50                     });
51             }
52         }
53     }
54 }

```

```
45         .set(this.currentChat);
46     }
47 },
48 startNewChat() {
49     // Create a new chat session and persist it to
50     // Firestore.
51 },
52 renderMarkdown(text) {
53     // Convert Markdown text to HTML.
54 }
55 mounted() {
56     this.loadChatList();
57 }
58 };
59 </script>
```

Listing 10.7: Abbreviated Implementation of the Saved Chats Feature

This concise implementation outlines how the platform leverages Firestore to manage and persist chat sessions, providing users with a seamless and consistent learning experience.

10.11. Structured and Intuitive Navigation

The platform's navigation is designed to ensure that users can efficiently access desired content and features. To achieve this, the primary components of the platform are placed in the main navigation bar at the top of the page. These key elements include:

- Home
- Account Management
- Chat Bots
- Image to Text
- Bildgenerierung
- Logout

10. Intelligent Student AI Hub: An Integrated Learning Platform



Abbildung 10.4.: Main Navigation Bar

To enhance usability when interacting with chat bots, the platform provides a dedicated Chat Bot Section. Within this section, users can select from different chat bots via a navigation bar located on the left side of the page, ensuring an intuitive and structured browsing experience.

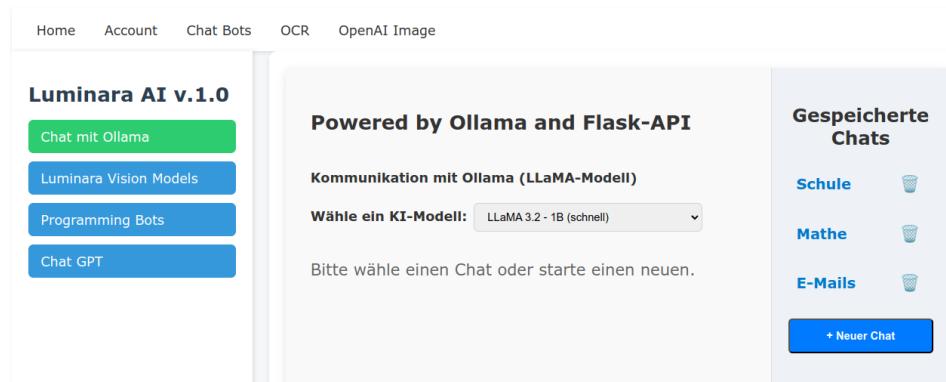


Abbildung 10.5.: Chat Bot Navigation

10.12. Styling and Theming

The platform's design follows a clean and modern aesthetic. To achieve this, a light and contemporary color scheme has been implemented. The background is predominantly white, with blue serving as the primary accent color and green used for highlights. For instance, buttons are styled in blue, while the currently selected chat bot is indicated in green.

Most of the styling was implemented using a CSS framework, with initial specifications defined by the development team. These were later refined with the assistance of AI tools, including ChatGPT (versions 3.5, 4, 40, and 01) as well as GitHub Copilot.

10.13. Features Excluded from the Final Version

Due to the limited development time and the emphasis on core functionalities, several planned features were not incorporated into the final version of the Student AI Hub. These include:

- **Multilingual Website:** While the platform currently supports multiple languages for the chatbot, the website itself is only available in German.
- **Chat Transcripts:** A feature designed to convert a user's chat history into a downloadable transcript for review and reference.
- **Test Preparation:** A module intended to generate practice tests and quizzes based on user preferences and learning progress.
- **Learning Analytics:** Tools for tracking and analyzing user learning patterns, progress, and areas for improvement.
- **Collaborative Learning:** Features enabling users to collaborate on projects, share knowledge, and engage in group learning activities.
- **Enhanced User Profiles:** Additional profile customization options, learning preferences, and progress tracking capabilities.
- **Dark Mode:** An alternative color scheme for the platform to reduce eye strain and improve readability in low-light environments.

The majority of these planned features were not implemented due to their time-intensive nature. For instance, the development of a multilingual website would have required extensive effort to translate and maintain all website content.

10.14. Conclusion

10.15. Conclusion

The Intelligent Student AI Hub represents a significant advancement in educational technology, providing students with a comprehensive and interactive platform to explore and learn about artificial intelligence. By integrating state-of-the-art technologies such as Vue.js, Flask, Firebase, and advanced AI

Gabriel Mrkonja
Florian Prandstetter
Luna P. I. Schätzle

models from OpenAI and Ollama, the platform offers a robust and scalable solution for AI education.

The core functionalities—including interactive chatbots, programming assistance, image recognition, and image-to-text conversion are designed to enhance the learning experience by making complex AI concepts accessible and engaging. The secure and personalized user management system, powered by Firebase, ensures that users can safely interact with the platform while enjoying a tailored educational journey.

While some planned features were not included in the final version due to time constraints, the platform's modular architecture allows for future enhancements and scalability. Potential future developments include multilingual support, collaborative learning tools, and advanced learning analytics, which will further enrich the educational experience.

In conclusion, the Intelligent Student AI Hub stands as a testament to the potential of integrating modern web technologies and AI to create a dynamic and effective learning environment. It not only empowers students to delve into the world of AI but also sets a foundation for continuous improvement and innovation in educational tools.

11. Visual Studio code extension

11.1. Introduction

This chapter provides an overview of the Visual Studio Code extension developed for the project. It describes its core functionalities, and explains how it integrates with the broader system architecture.

11.2. what is Visual Studio Code

Visual Studio Code is a free code editor from Microsoft. It supports many programming languages such as Python, JavaScript, and C++.

A major advantage of VS Code is its extensibility. With extensions, you can customize the editor, for example, with debugging tools, themes, or special functions for specific programming languages. It also offers features like auto-completion, integrated Git support, and a built-in terminal function.

VS Code is lightweight and runs on Windows, macOS, and Linux. Despite this, it provides many features that are also found in a full-fledged integrated development environment. This makes it perfect for both beginners and professionals.

11.3. Development

- **TypeScript:** The Visual Studio Code extension was developed using TypeScript. TypeScript is well-suited for developing VS Code exten-

sions, as it provides type checking and code completion, making it easier to work with the VS Code API.

- Axios: Axios is used to make HTTP requests from the extension to the Flask Service. It provides an easy implementation of asynchronous requests and simplifies handling responses.
- Visual Studio Code API: The extension interacts with the Visual Studio Code API. The API allows the extension to access and modify the editor's functionality, enabling it to provide a seamless development experience.

11.4. Core Functionalities

The planned core Functionalities of the extension are, an integrated chatbot and code completion.

11.4.1. Chatbot Integration

The extension integrates a chatbot into the editor, allowing developers to interact with the chatbot directly from the editor. This feature enables developers to quickly get information, ask questions, or perform tasks without leaving the editor.

11.4.2. Code completion

Teil V.

Implementation of Object Detection

12. Introduction to Object Detection

13. Implementation of Object Detection

Teil VI.

Evaluations

14. Artificial Intelligence in Economics

14.1. Introduction

15. Open source evaluation on Economics

Author: Luna P. I. Schätzle

15.1. Introduction

This chapter introduces the concept of Open Source and highlights its significance in the modern economy. Key aspects such as the advantages and disadvantages of Open Source, as well as the challenges associated with its adoption and creation, are discussed. Additionally, the chapter explores revenue models within the Open Source ecosystem and its role in economic systems. Finally, the chapter concludes by presenting the Open Source tools utilized in this project, alongside a reflection on the experiences gained through their application.

15.1.1. What is Open Source?

Open Source represents a collaborative and transparent approach to software development and distribution, where the source code is made publicly accessible. This philosophy empowers users not only to utilize the software but also to modify, improve, and redistribute it freely. By fostering an environment of openness and collaboration, Open Source drives innovation and democratizes access to technology.

Linus Torvalds, the creator of the Linux operating system, encapsulated this spirit of freedom and collaboration with his famous remark:

“Software is like sex: it’s better when it’s free.”

?

This statement highlights the fundamental ethos of Open Source—the belief that open access and shared knowledge result in better, more impactful solutions.

The development process for Open Source software is often a collective effort, with contributions from diverse communities of developers, users, and organizations. These collaborative efforts enhance the software's functionality, security, and usability, resulting in products that are robust and adaptable. Prominent examples include the Linux operating system, the Apache web server, and the Firefox web browser, all of which have significantly influenced technological innovation and market dynamics.

?

15.1.2. Advantages of Open Source

Open Source software offers a wide range of benefits, making it a cornerstone of modern technology:

- **Cost Efficiency:** Open Source software is typically free of charge, helping organizations and individuals save on licensing and maintenance costs.
- **Flexibility:** Users can access the source code, enabling them to tailor the software to their specific needs and requirements.
- **Security:** The open nature of the source code allows for peer review, ensuring vulnerabilities are identified and addressed promptly.
- **Community Support:** Open Source projects often benefit from vibrant developer communities, providing updates, patches, and user assistance.
- **Innovation:** The collaborative ecosystem of Open Source encourages creativity, leading to groundbreaking solutions and advancements.
- **Compatibility:** Many Open Source projects are designed to integrate seamlessly with existing systems, reducing technical barriers.

- **Transparency:** Open access to the source code ensures that users can understand and verify how the software operates.
- **Freedom:** Users are granted the liberty to use, modify, and share the software without restrictive licensing agreements.

??

15.1.3. Why Do People Use Open Source?

The adoption of Open Source software is motivated by several compelling factors:

- **Control:** Users gain full control over the software, enabling customization and optimization for specific use cases.
- **Cost Savings:** The absence of licensing fees significantly reduces expenses, making Open Source particularly attractive for startups and educational institutions.
- **Security:** Transparency in the source code allows for thorough auditing, enhancing trust and reliability.
- **Community:** The collaborative spirit of Open Source connects users with knowledgeable communities that share resources and support.
- **Stability:** Many Open Source projects offer long-term support and regular updates, ensuring reliability over time.
- **Skill Development:** Learning and using Open Source tools are valuable in educational and professional contexts, equipping individuals with in-demand skills.

15.2. What is and isn't Open Source?

15.2.1. Definition and Guiding Principles

Open Source, as defined by the Open Source Initiative (OSI), is a development approach that prioritizes accessibility and transparency of software

Gabriel Mrkonja
Florian Prandstetter
Luna P. I. Schätzle

source code. It allows users to view, modify, and distribute the code freely, fostering collaboration and innovation.

The OSI outlines several key principles that define Open Source software:

- **Free Redistribution:** The software can be freely shared and distributed without restrictions.
- **Source Code Access:** Users must have access to the source code to study, modify, and improve the software.
- **Modification and Sharing:** Users are allowed to create and share modified versions, as long as they follow the license terms.
- **No Discrimination:** The software must be available for everyone, regardless of individual characteristics or professional field.
- **Neutrality and Compatibility:** The license must not favor specific technologies or restrict the use of other software.

These principles ensure that Open Source remains a transparent, inclusive, and adaptable approach to software development, enabling innovation and collaboration across industries and communities.

?

15.2.2. Misconceptions About Open Source

Open Source is often misunderstood and confused with other software distribution models, which can lead to misconceptions about its nature, functionality, and benefits. It is crucial to distinguish Open Source from other types of software:

- **Open Source:** Software that is freely accessible, modifiable, and redistributable under an Open Source license, adhering to principles such as transparency and collaboration.
- **Freeware:** Software available at no cost but typically without access to the source code, meaning users cannot modify or redistribute it.
- **Proprietary Software:** Software owned and controlled by a single entity, restricting access to the source code and preventing users from making modifications or redistributions.

- **Commercial Software:** Software sold for profit, which may be either Open Source or proprietary, depending on the licensing terms.

Understanding these distinctions helps users make informed choices about software selection and ensures their expectations align with the capabilities and freedoms provided by the chosen software.

To verify whether a software is truly Open Source, it is essential to examine the license agreement and confirm the availability of the source code. Software with an OSI-approved license is a reliable indicator that it adheres to Open Source principles, providing transparency, freedom, and collaboration opportunities.

One common misconception about Open Source software arises from the phrase "free as in freedom" versus "free as in free beer." While "free as in freedom" emphasizes the liberty to access, modify, and share the software, "free as in free beer" simply denotes that the software is free of cost. Although Open Source software is often available without charge, its true value lies in the freedom it grants to users, developers, and organizations. This distinction highlights the broader significance of Open Source as a philosophy, not just a pricing model.

?

15.3. Challenges and Disadvantages of Open Source Software

Although open source software provides numerous advantages, it also presents several challenges that can affect its adoption, development, and sustainability. The following sections outline the primary disadvantages and challenges encountered in open source environments.

15.3.1. Disadvantages of Open Source Software

Key drawbacks associated with open source software include:

Gabriel Mrkonja
Florian Prandstetter
Luna P. I. Schätzle

- **Limited Support:** Many open source projects lack dedicated support teams, often resulting in slower response times for bug fixes and technical issues.
- **Reliance on Hobby Developers:** Projects maintained by volunteers or hobbyists may experience irregular updates and inconsistent maintenance.
- **Fragmentation:** The decentralized development model can lead to fragmentation, with multiple versions and distributions causing compatibility challenges.
- **Reduced Feature Set:** Certain open source applications might not offer the advanced features or functionalities that are common in commercial alternatives.

?

15.3.2. Technical Challenges

Integrating open source software into a project requires adequate technical expertise to understand, modify, and deploy the software effectively. When in-house expertise is insufficient, organizations may need to hire external developers or consultants. Although this can help prevent technical issues and ensure successful integration, it may increase overall costs. In some cases, proprietary software—despite being more expensive—offers easier integration due to dedicated support and streamlined installation processes.

15.3.3. Economic Challenges

While open source software is generally free to use, significant costs may arise from its implementation, customization, maintenance, and support. These expenses can accumulate over time, especially when frequent updates or extensive customization are required. Outsourcing technical support can help mitigate these economic challenges, but it may not be a viable solution for every organization.

Gabriel Mrkonja

Florian Prandstetter

Luna P. I. Schätzle

15.3.4. Social Challenges

The collaborative nature of open source development, which depends on contributions from a diverse community of developers and organizations, can lead to an ambiguous support structure. This lack of clarity often makes it difficult for companies to identify the appropriate contact for assistance, potentially causing delays in addressing technical issues and adversely affecting project outcomes.

15.3.5. Legal Challenges

Navigating the legal landscape of open source software can be complex, largely due to the variety of licensing models (e.g., GPL, MIT, Apache) that impose different obligations and restrictions. Ensuring compliance with these licenses demands a thorough understanding of their terms, which can be both time-consuming and legally challenging. Failure to adhere to license conditions may result in legal disputes, costly litigation, and damage to an organization's reputation. It is therefore crucial to educate team members on compliance requirements and establish robust processes for managing open source software usage.

?

Overview of License Models

A license is a legal instrument that defines the conditions under which a work may be used, modified, and distributed, thereby outlining the rights and obligations of both the licensor and the licensee.

Open-Source Licenses

Open source licenses are a specific type of software license that promotes collaborative development by allowing unrestricted use, modification, and sharing of the software. Their main features include:

Gabriel Mrkonja
Florian Prandstetter
Luna P. I. Schätzle

- **Unrestricted Use:** The software may be used for any purpose without limitations.
- **Source Code Access:** Availability of the source code enables users to inspect, modify, and enhance the software.
- **Redistribution Rights:** Users can distribute the original or modified versions of the software, thereby fostering community-driven development.

Notable examples include the GNU General Public License (GPL), which mandates that all modifications remain open source; the permissive MIT License, which imposes minimal restrictions; and the Apache License, which provides a balance between flexibility and patent protection. The choice of license is critical, as it can profoundly influence the software's development trajectory, market adoption, and the engagement of its community.

?

15.4. Potential Risks and Security Concerns

Before integrating open source software into its operations, a company must conduct a comprehensive risk assessment to identify potential security concerns and other associated liabilities. Although open source solutions can offer cost savings, flexibility, and rapid innovation, they may also expose organizations to vulnerabilities that compromise data security, expose sensitive information, or disrupt business operations.

15.4.1. Common Risks Associated with Open Source Software

Several risks are inherently linked to the use of open source software, including:

- **Security Vulnerabilities:** Open source projects may contain inherent security flaws that, if left unpatched, can be exploited by malicious actors to gain unauthorized access to systems and data.

- **Compliance and Licensing Issues:** The complex landscape of open source licenses requires strict adherence; non-compliance can lead to legal disputes, financial penalties, and reputational damage.
- **Dependency and Supply Chain Risks:** Open source applications often rely on third-party libraries and components, each introducing additional vulnerabilities and potential compatibility issues across the software supply chain.
- **Limited Support and Maintenance:** Many projects are maintained by volunteer communities rather than dedicated support teams, which can result in delayed updates and prolonged exposure to unresolved security issues.
- **Quality and Code Integrity Concerns:** Variability in coding practices, insufficient testing, and poor documentation can lead to inconsistent software quality, increasing the likelihood of bugs and security weaknesses.

?

15.4.2. Specific Security Concerns in Open Source Environments

Security risks in open source software manifest in various ways, including:

- **Malware and Backdoors:** The public availability of source code can allow malicious actors to inject harmful code or create backdoors if rigorous code reviews and continuous monitoring are not in place.
- **Supply Chain Attacks:** As organizations integrate multiple open source components, attackers may target less secure dependencies, thereby compromising the broader software ecosystem.
- **Delayed Patch Management:** Open source projects may experience delays in vulnerability identification and patch deployment, leaving systems exposed to potential exploitation.
- **Suboptimal Developer Practices:** Inadequate testing, inconsistent coding standards, and poor documentation can exacerbate security issues, as these practices increase the risk of undetected errors.

Gabriel Mrkonja
Florian Prandstetter
Luna P. I. Schätzle

- **Compliance Risks Impacting Security:** Non-compliance with licensing terms not only poses legal risks but may also force disruptive changes to the software stack, potentially introducing new vulnerabilities during transitions.

?

In summary, while open source software can serve as a powerful and cost-effective tool for innovation, its adoption demands vigilant risk management. Organizations should implement robust security protocols, perform regular audits of open source components, and ensure strict compliance with licensing requirements to mitigate these risks effectively.

15.5. The Role of Open Source in Economics

Cost efficiency, innovation, and collaboration are key factors that have positioned Open Source as a cornerstone of modern economic systems. Many industries and organizations utilize Open Source software to reduce costs, increase flexibility, and promote creativity, thereby driving economic growth and sustainability.

15.5.1. Driving Innovation and Shaping Market Dynamics

Open Source software fosters a culture of experimentation, creativity, and knowledge sharing, leading to the rapid development of new technologies and solutions. By granting users access to modify and redistribute the source code, Open Source encourages collaboration and innovation, enabling individuals and organizations to build upon existing software to create new products and services.

A distinctive strength of Open Source is its inclusivity—anyone, regardless of their affiliation with a company, can contribute to its development. This openness lowers barriers to entry for innovation and allows passionate individuals to make meaningful contributions.

Companies also play a significant role in advancing Open Source projects. With greater resources and structured teams, organizations can contribute in a more organized and impactful manner, accelerating development and enhancing software quality.

The collaborative nature of Open Source facilitates cross-industry partnerships, allowing organizations from diverse sectors to share knowledge, resources, and best practices. This cross-pollination of ideas not only enhances software development but also fosters innovation across industries, ultimately shaping market dynamics and driving economic progress.

The study [?] by Mike Hendrickson, Roger Magoulas, and Tim O'Reilly underscores that Open Source is not only a catalyst for small business growth but also a driver of future success for many startups today. By providing cost-effective and flexible solutions, Open Source enables small and medium-sized enterprises to strengthen their online presence and enhance their economic performance.

15.5.2. Supporting Startups and small Enterprises

The impact of Open Source on startups and small enterprises is both profound and transformative. For these businesses, Open Source software provides a highly cost-effective alternative to proprietary solutions, granting access to advanced tools and technologies without the financial burden of high licensing fees typically associated with commercial software. This affordability allows startups and small enterprises to allocate their limited resources more strategically, fostering innovation and growth while maintaining financial flexibility.

?

15.5.3. Facilitating Cross-Industry Collaboration and Open Innovation

Leveraging the intrinsic collaborative nature of open source platforms, organizations are empowered to forge cross-industry alliances and pursue

Gabriel Mrkonja
Florian Prandstetter
Luna P. I. Schätzle

open innovation strategies. By pooling shared resources, expertise, and technologies, these collaborations accelerate progress and address multifaceted challenges. This integrative approach transcends traditional industry boundaries, fostering cooperation among diverse sectors in the pursuit of common objectives and mutually beneficial solutions.

15.6. Open Source in Key Industries

Across numerous industries, open-source software has exerted a profound influence on organizational operations, catalyzing innovation and facilitating collaborative practices. The adoption of open-source solutions is pervasive across various sectors, including:

- **Information Technology:** Open-source software underpins a wide array of critical IT infrastructures, ranging from operating systems and databases to web servers, thereby enhancing system reliability and flexibility.
- **Artificial Intelligence:** Frameworks such as TensorFlow and PyTorch have democratized access to artificial intelligence technologies, promoting extensive research and innovative development.
- **Education:** Platforms like Moodle and Jupyter Notebooks have transformed the landscape of online education by making learning more accessible and interactive, which in turn fosters broader pedagogical engagement.
- **Healthcare:** Increasingly, open-source solutions are being integrated into healthcare systems to manage electronic health records, medical imaging, and telemedicine applications, thereby improving patient care and system interoperability.
- **Finance:** In the financial sector, open-source software is instrumental in operating trading platforms, risk management systems, and blockchain technologies, which contributes to enhanced transparency and operational efficiency.

15.6.1. Examples of Open Source Success Stories

The following examples illustrate the transformative impact of open-source software across key industries:

GNU/Linux in Information Technology: The GNU/Linux operating system, initiated by Linus Torvalds, has evolved into a cornerstone of modern IT infrastructure. Its adoption extends beyond the personal computing domain to include servers, supercomputers, and embedded systems. The system's inherent stability, robust security features, and considerable flexibility have been critical to its widespread acceptance.

Mozilla: Mozilla Firefox is a widely used web browser developed by the Mozilla Foundation—an open-source community dedicated to promoting an open and accessible internet. Its strong commitment to privacy, security, and user empowerment has established it as a favored alternative to proprietary browsers. Additionally, the Mozilla ecosystem encompasses projects such as Thunderbird, a free and open-source email client, and the Mozilla Developer Network (MDN), a comprehensive resource that supports web developers worldwide.

LibreOffice: LibreOffice is a comprehensive, free, and open-source office suite that offers a robust alternative to proprietary software such as Microsoft Office. It encompasses applications for word processing, spreadsheets, presentations, and more, thereby providing a versatile and cost-effective solution for both individuals and organizations. Its compatibility with multiple operating systems—including Windows, macOS, and Linux—ensures broad accessibility, making it suitable for a diverse range of sectors from education and non-profit organizations to small enterprises and governmental agencies.

Gabriel Mrkonja
Florian Prandstetter
Luna P. I. Schätzle

OpenEMR: OpenEMR is an open-source practice management software solution that has been widely adopted in the healthcare industry. It is estimated that OpenEMR currently manages the records of over 90 million patients in the United States. Utilized by a diverse spectrum of healthcare providers—from small practices to large hospitals—OpenEMR facilitates the management of patient records, appointment scheduling, billing, and other critical functions. This example underscores the potential of open-source software to revolutionize healthcare delivery by offering customizable and cost-effective solutions.

?

15.7. Revenue Models in Open Source

For an open-source project to develop effectively and remain sustainable, it is crucial to establish a revenue model that aligns with its goals and objectives. The open-source ecosystem offers a variety of revenue models, each with its own advantages and challenges. By selecting the most suitable model, project maintainers can secure the necessary funding, support ongoing development, and ensure long-term viability.

15.7.1. Common Business Models

Several business models have proven successful in the open-source landscape, including:

- **Open Core:** In this model, the core software is open source and freely available, while additional features and functionalities are provided under a commercial license. Examples include MongoDB and GitLab.
- **Hosting and Cloud Solutions:** Companies offer hosting services or cloud-based solutions for their open-source software, charging users for the enhanced services. Examples include WordPress and Databricks.

- **Support and Maintenance Services:** Revenue is generated by offering support, maintenance, and consulting services to users of the open-source software. Notable examples are Red Hat and Canonical (Ubuntu).

Other revenue streams—such as donations, dual licensing, and strategic partnerships—also play a role in sustaining open-source projects.

15.8. Open Source Support in Austria

In Austria, numerous organizations are dedicated to supporting and promoting open-source software. Some groups focus on networking and knowledge exchange, while others offer direct services and support for open-source initiatives. Additionally, the Wirtschaftskammer Österreich (WKO) provides assistance to companies that wish to adopt or develop open-source solutions.

To further promote open-source software in Austria, it is essential to raise awareness of its benefits and encourage collaboration among organizations, developers, and users. By nurturing a vibrant open-source community, Austria can leverage collaborative innovation to drive both economic growth and technological advancement.

?

15.9. Open Source in Practice: A Personal Experience

For the Diploma Thesis, our project team leveraged a diverse range of open-source technologies. The project made use of Python, Flask, Vue.js, Linux, Ollama, Visual Studio Code, and many other open-source tools to develop robust applications.

Our decision to adopt open-source technologies was driven by several factors, including cost efficiency, flexibility, and strong community support.

Gabriel Mrkonja
Florian Prandstetter
Luna P. I. Schätzle

Access to the source code enabled us to customize and extend the software to meet specific project requirements, while vibrant developer communities offered valuable resources and guidance throughout the development process.

Although open-source software presents numerous advantages, it also comes with challenges such as limited official support, potential security vulnerabilities, and licensing complexities. Successfully navigating these issues required careful planning, continuous monitoring, and adherence to best practices to ensure the project's success.

15.10. Licence Model of the Diploma Thesis

The source code for this Diploma Thesis is publicly available under the GNU General Public License (GPL) version 3. This license ensures that the software remains open source and freely accessible to all users, reflecting the project's commitment to transparency, collaboration, and innovation. By adopting this license, we empower others to build upon our work and contribute to its ongoing development.

The source code is hosted on GitHub, which serves as a platform for collaboration, feedback, and community engagement. The repository can be accessed at <https://github.com/Luna-Schaetze/Diploma-thesis-website>.

15.10.1. GNU General Public License (GPL) Version 3

Published by the Free Software Foundation in 2007, the GNU General Public License Version 3 (GPLv3) is a widely adopted open-source license designed to safeguard software freedom. It grants users the rights to use, study, modify, and distribute software while its *copyleft* clause ensures that any derivative works are also licensed under GPLv3, preventing proprietary exploitation. GPLv3 addresses modern challenges such as patent threats and digital rights management (DRM) restrictions by offering robust patent protection, prohibiting DRM technologies, and enhancing compatibility with other licenses. Employed by projects like GNU tools and Bash, GPLv3

remains a cornerstone of the open-source movement, ensuring that software stays free and accessible.

15.11. Conclusion

Open-source software has become an integral part of the modern economy, driving innovation, fostering collaboration, and promoting economic growth. By providing cost-effective, flexible, and transparent solutions, open source empowers individuals, organizations, and entire industries to achieve their objectives more efficiently and sustainably. Moreover, a variety of viable business models support the monetization and continued development of open-source projects.

Our experience with open-source technologies underscores the immense value of community-driven development, customization, and collaboration. By leveraging these tools, our team was able to devise innovative solutions, overcome complex challenges, and contribute meaningfully to the broader open-source ecosystem.

16. Economic aspect of Operating Systems

16.1. Introduction

Teil VII.

Conclusion

17. Conclusion

In this thesis, we have conducted a comprehensive study on *[Artificial Intelligence in the Industry and Education Environment]*. The project aimed to investigate the potential applications of AI in the context of *[Industry and Education]* and to develop innovative solutions for addressing the challenges in these domains. The research objectives were to:

- Identify key areas where AI can be effectively applied in the industry and education environment.
- Develop practical solutions for enhancing productivity, efficiency, and learning outcomes in these sectors.
- Evaluate the performance and impact of the proposed methods through empirical studies and case analyses.
- Provide recommendations for future research and implementation strategies based on the findings of this study.
- Contribute to the ongoing discourse on the role of AI in shaping the future of industry and education.
- Compare the results of the project with the initial research questions and hypotheses.

17.1. Key Findings

17.2. Implications and Recommendations

17.3. Limitations and Challenges

18. Outlook

Author: Luna P. I. Schätzle, Florian Prandstetter and Gabriel Mrkonja

In this chapter, we provide an outlook on the future of AI in the industry and education environment. We discuss potential trends, challenges, and opportunities that may arise in the coming years and offer recommendations for further research and development in this field.

18.1. Future Trends in AI

The AI landscape is constantly evolving, driven by the rapid emergence of new technologies and applications. While it remains challenging to predict the exact trajectory of AI, several discernible trends are already shaping its future. One of the most significant developments is the increasing integration of AI into everyday devices and services. As AI becomes more ubiquitous, it is poised to play an even greater role in influencing our daily lives and interactions with the world.

Another key trend is the evolution of AI models toward more human-like behavior, with advanced reasoning capabilities becoming increasingly sophisticated. This progress is further supported by enhancements in memory capacity and computational power, which contribute to higher accuracy and faster performance.

A further emerging trend is the development of autonomous AI agents capable of interacting with their environment and making decisions independently. For instance, systems like the Claude Computer Use demonstrate how AI can engage with the entire user operating system and interface autonomously?

18.1.1. AI in Industry

In industry, AI is anticipated to revolutionize manufacturing processes, supply chain management, and quality control. AI-powered robots and autonomous systems are expected to play a critical role in streamlining operations and enhancing efficiency. Additionally, predictive maintenance and asset management will benefit from AI technologies, enabling companies to minimize downtime and optimize resource allocation.

18.1.2. AI in Education

Within the education sector, AI is set to transform both teaching and learning methodologies. Personalized learning platforms, intelligent tutoring systems, and automated grading tools promise to enhance the educational experience for students and educators alike. Moreover, AI will facilitate the creation of adaptive learning environments tailored to individual learning styles and preferences.

18.1.3. Challenges and Opportunities

Despite its potential, AI also presents significant challenges. Ethical implications such as bias, privacy, and accountability must be addressed to ensure that AI technologies are developed and deployed responsibly. There is also a pressing need for greater transparency and explainability in AI systems, particularly in high-stakes fields such as healthcare and finance.

Another major concern is related to copyright and data ownership. AI systems often draw upon extensive amounts of data available on the internet to generate new content, which may not be owned by the original creators. This issue raises complex questions about intellectual property rights and the ethical use of data.

18.2. Further Development of the Flask Server

The Flask server is a cornerstone of the AI Hub, providing the essential infrastructure for hosting AI models and enabling students to access a broad spectrum of AI tools and resources. To elevate the server's functionality and performance, several key enhancements should be prioritized:

- **Optimized Architecture:** Refine the server architecture to efficiently manage high volumes of concurrent requests.
- **Robust Security Measures:** Implement comprehensive security protocols—including encryption, authentication, and access control—to safeguard user data and protect against unauthorized access.
- **Expanded Integration:** Integrate additional AI models and services to further broaden the platform's capabilities.

Furthermore, regular updates and maintenance of both the server and the Ollama API are imperative to ensure compatibility with the latest AI models and technologies. Consistent maintenance guarantees the reliability and security of the platform, while also providing students with uninterrupted access to cutting-edge AI tools and resources.

Enhanced error handling and resource management are also crucial to maintain continuous server operation and optimize resource utilization. Overall, prioritizing these improvements will ensure that the Flask server remains robust, secure, and scalable, meeting the evolving demands of the AI Hub.

18.3. Further Development of the Student AI Hub

The Student AI Hub represents a promising initiative with the potential to transform how students interact with AI technologies. By providing a dedicated platform where students can both learn about AI and utilize a variety of AI tools to enhance their educational experience, the Student AI Hub seeks to democratize access to AI education and empower learners to expand their skills and knowledge in this rapidly evolving field.

To advance the development of the Student AI Hub, several key areas should be prioritized. These include expanding the range of available AI tools and resources, fostering collaboration and knowledge-sharing among students, and establishing strategic partnerships with industry and academic institutions to enrich the platform's offerings.

Additionally, it is crucial for the Student AI Hub to promote diversity and inclusion in AI education. This can be achieved by offering tailored resources and dedicated support to underrepresented groups, thereby ensuring that a broader spectrum of students can benefit from and contribute to advancements in AI.

18.4. Open Source in Future Projects

Open-source software has become increasingly integral to the AI community, empowering developers to collaborate, share resources, and drive innovation at an accelerated pace. By embracing open-source principles in future projects, we can harness the collective expertise of the global AI community to develop state-of-the-art solutions and tackle complex challenges in both industry and education.

Adopting open-source methodologies not only facilitates the widespread dissemination of knowledge and best practices, but it also enables students and professionals to access valuable resources and actively contribute to the advancement of AI technologies. Moreover, open-source initiatives promote transparency, accountability, and inclusivity, thereby fostering a collaborative culture of knowledge exchange within the AI ecosystem.

In key industries, open source is already a fundamental component of the development process. Observing the evolution of the open-source community will be pivotal in understanding its future impact on the development and integration of AI within both industrial and educational environments.

18.5. Challenges and Opportunities

18.6. Recommendations for Further Research

18.7. Concluding Remarks

Anhang A.

Time Protocol

A.1. Luna P. I. Schätzle

Date	Start	End	Hours	Tasks	Details	Persons
08.09.2024	10:00:00	12:30:00	02:30:00	Planning; Re-search	Research and definition of the SAIPiA concept	Luna
13.09.2024	08:50:00	09:20:00	00:30:00	Planning	Discussed and assigned the new idea	All (oL)
15.09.2024	11:30:00	12:30:00	01:00:00	Planning; Re-search	Definition for the thesis database which models can be used	Luna
15.09.2024	14:00:00	15:00:00	01:00:00	Meeting		All (oL)
16.09.2024	06:40:00	07:00:00	00:20:00	Research	Elaboration of the project definition	Luna
17.09.2024	08:00:00	08:40:00	00:40:00	Research	Improving entries in the thesis databases	Luna
23.09.2024	09:35:00	10:15:00	00:40:00	Software Test	Testing the LLama model 8b	Luna
23.09.2024	14:00:00	15:30:00	01:30:00	Planning; Re-search	Addressing Ollama from Python GitHub repository created	Luna
23.09.2024	16:00:00	16:40:00	00:40:00	Software Development	Testing various scripts	Python Luna

Continued on next page

Anhang A. Time Protocol

Date	Start	End	Hours	Tasks	Details	Persons
24.09.2024	07:30:00	08:00:00	00:30:00	Planning	Git README update and Aba portal	Luna
24.09.2024	12:30:00	13:20:00	00:50:00	Software Development	Fetching news via News API and storing in a database	Luna
24.09.2024	15:30:00	16:20:00	00:50:00	Software Development	Time and news integrated into the Ollama Python script	Luna
24.09.2024	17:20:00	18:00:00	00:40:00	Software Development	Python sound test	Luna
26.09.2024	07:30:00	08:00:00	00:30:00	Planning	Improving entries in the thesis databases	Luna
26.09.2024	17:30:00	18:30:00	01:00:00	Software Development	Python TTS test various libraries Java version of AI addressing (via API)	Luna
30.09.2024	10:00:00	10:50:00	00:50:00	Planning	Resubmission of the thesis Development of the thesis red thread	Luna
30.09.2024	11:10:00	11:40:00	00:30:00	Software Development	Testing a simple graphical interface for the open day	Luna
30.09.2024	20:40:00	21:00:00	00:20:00	Software Development	Sound input and output via Python test qtts including trigger word	Luna
01.10.2024	14:30:00	15:00:00	00:30:00	Software Test	Testing various LLM models	Luna
03.10.2024	18:55:00	19:25:00	00:30:00	Research	Looked at LaTeX template and various LaTeX programs	Luna
03.10.2024	22:20:00	23:10:00	00:50:00	Research	Using LaTeX template, adding own part Looked at other theses + wrote down own ideas	Luna
04.10.2024	07:50:00	08:00:00	00:10:00	Meeting	Discussion of orders Definition of the parts list Cost allocation defined	All (oL)
04.10.2024	13:20:00	14:00:00	00:40:00	Research	Skimming through other theses Thinking about additions for the structure	Luna

Continued on next page

A.1. Luna P. I. Schätzle

Date	Start	End	Hours	Tasks	Details	Persons
08.10.2024	14:20:00	14:50:00	00:30:00	Writing	Testing the LaTeX template + fixing the library Dealing with the different parts	Luna
10.10.2024	11:30:00	13:00:00	01:30:00	Software Development	Tried to run Flux[DEV] on the PC, didn't work :(SWAP memory too low and CPU fully utilized	Luna
10.10.2024	14:30:00	18:00:00	03:30:00	Hardware Test	HAILO on Raspberry test and hardware test	All (oL)
11.10.2024	10:50:00	11:30:00	00:40:00	Software Development	Webuntis API fetch via the Webuntis Python library test of the outputs	Luna
15.10.2024	16:30:00	16:50:00	00:20:00	Research	Organization of notes, overcoming "project crisis"	Luna
17.10.2024	10:00:00	11:30:00	01:30:00	Planning; Research	Test of the AI school server	Luna
17.10.2024	12:30:00	12:50:00	00:20:00	Meeting	Further discussion with Greinöcker regarding the thesis	Flo, Luna
17.10.2024	17:20:00	18:20:00	01:00:00	Research; Software Development	Searching how to fine-tune Searching for datasets Download LLAMA3.2:1b	Luna
17.10.2024	20:20:00	22:20:00	02:00:00	Research; Software Development	Trying the llama3 weight download and operation Searching for alternatives How to fine-tune Research documents as context (art fine-tuning) Trying various applications	Luna
18.10.2024	00:30:00	01:00:00	00:30:00	Software Development	Programming the Vue.js app (just the connection to Ollama)	Luna
18.10.2024	08:00:00	08:30:00	00:30:00	Software Test	Test of downloading the llama3.2:1b weights on the school server	Luna

Continued on next page

Anhang A. Time Protocol

Date	Start	End	Hours	Tasks	Details	Persons
18.10.2024	08:50:00	09:40:00	00:50:00	Software Test	Test of datasets and training	Luna
18.10.2024	10:20:00	12:50:00	02:30:00	Software Test	Test training Also on the school server + bugfix Training a model	Luna
18.10.2024	17:50:00	18:10:00	00:20:00	Software Test	Test local training minimizing the training data -> less time	Luna
21.10.2024	07:40:00	08:00:00	00:20:00	Software Test	Start of the test training of the llama3.2:1b model	Luna
21.10.2024	14:25:00	15:05:00	00:40:00	Software Test	Test of the fine-tuned model Test image generation on the server	Luna
21.10.2024	15:45:00	16:45:00	01:00:00	Software Test	Test of image and video generation on the server	Luna
22.10.2024	07:40:00	09:00:00	01:20:00	Software Test	Further models tested on the server	Luna
22.10.2024	13:00:00	15:00:00	02:00:00	Software Test	Further testing of the server Vue.js add image port Flask img	Luna
21.10.2024	17:00:00	17:30:00	00:30:00	Software Development	Porting the image generation to Vue.Js using Flask as backend	Luna
24.10.2024	10:00:00	11:30:00	01:30:00	Software Development	Vue port of the image generation with Flask in the background	Luna
24.10.2024	12:00:00	14:40:00	02:40:00	Software Development	Vue port bugfix + Upload to the server	Luna
24.10.2024	15:40:00	16:10:00	00:30:00	Software Test	Test of Vue + Further deployment VPN research	Luna
24.10.2024	16:40:00	16:50:00	00:10:00	Planning	Working hours extraction	Luna
24.10.2024	17:30:00	18:30:00	01:00:00	Software Development	AI vision LLAVA test with API Vue adaptation	Luna
24.10.2024	19:20:00	20:10:00	00:50:00	Software Development	AI vision LLAVA test with API Vue adaptation fix	Luna

Continued on next page

Date	Start	End	Hours	Tasks	Details	Persons
28.10.2024	11:15:00	12:00:00	00:45:00	Software Development	Test of voice input and output researched things	Luna
29.10.2024	13:55:00	14:55:00	01:00:00	Writing	Economic part Open source and AI and its impact	Luna
01.11.2024	14:30:00	14:40:00	00:10:00	Research	Benchmarking Ollama models with numbers	Luna
12.11.2024	15:00:00	15:20:00	00:20:00	Planning	New division of the thesis	Luna
12.11.2024	16:30:00	18:00:00	01:30:00	Software Development	Testing the VueJS framework for the chatbot	Luna
14.11.2024	10:55:00	11:15:00	00:20:00	Software Development	Testing Supabase	Luna
15.11.2024	11:45:00	12:05:00	00:20:00	Meeting	Division and discussion	All (oL)
18.11.2024	16:40:00	16:50:00	00:10:00	Research	Open AI API key research	Luna
19.11.2024	17:40:00	17:50:00	00:10:00	Research	Searching for suitable models for our applications	Luna
21.11.2024	10:30:00	10:40:00	00:10:00	Research	How to build the backend of the server	Luna
21.11.2024	10:50:00	11:50:00	01:00:00	Software Development	Development of the VueJS website	Luna
22.11.2024	12:30:00	13:20:00	00:50:00	Software Development	Vue login with Firebase	Luna
22.11.2024	17:40:00	18:20:00	00:40:00	Software Development	Vue login and optimization of user management	Luna
23.11.2024	15:30:00	16:20:00	00:50:00	Software Development	Website development improvement of the login	Luna
25.11.2024	09:10:00	09:40:00	00:30:00	Software Development	Development of the OCR application	Luna
25.11.2024	10:00:00	11:45:00	01:45:00	Software Development	Further development of the OCR application using Flask server	Luna
26.11.2024	10:15:00	11:10:00	00:55:00	Software Development	Programming the chat with LLAMA	Luna
26.11.2024	14:10:00	16:30:00	02:20:00	Software Development	Fix OCR, implementation of programming chats	Luna

Continued on next page

Anhang A. Time Protocol

Date	Start	End	Hours	Tasks	Details	Persons
26.11.2024	18:20:00	18:35:00	00:15:00	Software Development	Tried to implement admin features	Luna
28.11.2024	09:00:00	09:30:00	00:30:00	Software Test	Test of a local RAG with Ol-lama	Luna
28.11.2024	10:10:00	10:30:00	00:20:00	Software Test	OpenAI API initialization and test	Luna
28.11.2024	10:50:00	11:40:00	00:50:00	Software Test	Test of the OpenAI API + Image generation	Luna
28.11.2024	13:00:00	13:40:00	00:40:00	Software Test	RAG test and co	Luna
29.11.2024	09:50:00	10:00:00	00:10:00	Writing	Discussion with Egger regarding the economic part (Open Source)	Luna
30.11.2024	13:40:00	16:25:00	02:45:00	Research; Software Development	Research on object recognition, program start of finger recognition for a user interface	Gabriel
01.12.2024	12:53:00	13:53:00	01:00:00	Software Development	Further programming of the object recognition	Gabriel
02.12.2024	10:50:00	11:30:00	00:40:00	Writing	Writing Open Source and impact on economy	Luna
04.12.2024	08:00:00	12:28:00	04:28:00	Writing	Writing Open Source and impact on economy	Luna
12.12.2024	13:30:00	15:15:00	01:45:00	Software Test	Extension of functionality and setting of a new base prompt	Luna
01.01.2025	19:45:00	21:15:00	01:30:00	Writing	Organizing the structure, writing some sections	Luna
07.01.2025	08:00:00	09:40:00	01:40:00	Writing	Further structuring and improving chapters; Searching for Bibtex converter	Luna
08.01.2025	11:30:00	12:32:00	01:02:00	Software Test	Website backend config and model testing	Flo, Luna
09.01.2025	10:45:00	11:30:00	00:45:00	Software Development	Fixing Firebase database issues; Extension of chat function (Saving chats)	Luna

Continued on next page

Date	Start	End	Hours	Tasks	Details	Persons
09.01.2025	13:50:00	14:40:00	00:50:00	Software Development	Extension of chat functions Imprint and privacy policy	Luna
13.01.2025	15:30:00	17:00:00	01:30:00	Writing	Used AI models Testing of the models	Luna
14.01.2025	08:10:00	08:50:00	00:40:00	Software Development	Evaluation of AI models	Luna
14.01.2025	14:45:00	15:00:00	00:15:00	Software Test	Further testing of evaluation attempts	Luna
14.01.2025	16:15:00	16:55:00	00:40:00	Writing	Explanation of the testing of evaluation models	Luna
14.01.2025	21:10:00	21:40:00	00:30:00	Writing	Model evaluation	Luna
20.01.2025	11:45:00	12:00:00	00:15:00	Software Test	Further data collection on the server	Luna
20.01.2025	13:20:00	13:50:00	00:30:00	Writing	Sage	Luna
21.01.2025	11:30:00	11:40:00	00:10:00	Software Test	Data collection for model evaluation	Luna
21.01.2025	13:00:00	13:15:00	00:15:00	Writing	OpenAI API description	Luna
21.01.2025	14:30:00	15:00:00	00:30:00	Writing	LLMs explanations Code listing improvement	Luna
23.01.2025	12:50:00	13:20:00	00:30:00	Writing	Organizing the structure	Luna
23.01.2025	21:10:00	22:00:00	00:50:00	Writing	Used programming languages	Luna
28.01.2025	14:00:00	14:50:00	00:50:00	Software Test	Evaluation of AI models	Luna
28.01.2025	15:50:00	17:10:00	01:20:00	Software Test	Further score collection	Luna
29.01.2025	08:00:00	11:00:00	03:00:00	Writing	Writing the first part	Flo
08.02.2025	14:00:00	16:00:00	02:00:00	Writing	Ollama explained Chat GPT API explanation	Luna
10.02.2025	10:00:00	12:30:00	02:30:00	Writing	Discussion, structure of the thesis, division, general writing	All (oL)

Continued on next page

Anhang A. Time Protocol

Date	Start	End	Hours	Tasks	Details	Persons
13.02.2025	14:45:00	15:25:00	00:40:00	Writing	Optimizing the model testing part	Luna
13.02.2025	09:30:00	10:30:00	01:00:00	Writing	Improvement of chapter structure (Introduction to LLM) Language improvement	Luna
13.02.2025	15:50:00	16:15:00	00:25:00	Writing	Data processing implementation of images and more Python scripts	Luna
14.02.2025	08:30:00	09:30:00	01:00:00	Writing	Hosted Flask service Used LLMs	Luna
14.02.2025	12:20:00	13:20:00	01:00:00	Writing	Build.sh for easier build via Linux only Original idea optimization	Luna
14.02.2025	14:50:00	16:20:00	01:30:00	Writing	Hello Luna / Original idea overworked / Timeline created v1 / Project timeline	Luna
14.02.2025	17:30:00	18:00:00	00:30:00	Writing	Different project evolutions	Luna
15.02.2025	22:15:00	22:40:00	00:25:00	Writing	Conceptual evolution and rationale finished	Luna
15.02.2025	22:40:00	00:00:00	01:20:00	Writing	Headlines improvement Structure Beginning and ending	Luna
16.02.2025	00:00:00	01:10:00	01:10:00	Writing	Core functionalities Student AI hub Hosted Flask service	Luna
16.02.2025	10:30:00	12:00:00	01:30:00	Writing	Firebase integration TSN integration issues	Luna
17.02.2025	08:50:00	09:30:00	00:40:00	Meeting	Discussion with Prof. Greinöcker	Luna
17.02.2025	10:40:00	12:10:00	01:30:00	Writing	Student AI hub Settings for listing Open AI integration Programming bot	Luna
17.02.2025	16:00:00	16:30:00	00:30:00	Writing	Image recognition Flask server	Luna

Continued on next page

A.2. Florian Prandstetter

Date	Start	End	Hours	Tasks	Details	Persons
18.02.2025	08:00:00	09:00:00	01:00:00	Writing	Account management Image to text Saved chats	Luna
18.02.2025	10:00:00	11:00:00	01:00:00	Writing	Open source challenges and disadvantages	Luna
18.02.2025	14:20:00	15:00:00	00:40:00	Writing	Risks and chances of open source	Luna
18.02.2025	16:00:00	16:45:00	00:45:00	Writing	Fixing code Open risks of open source	Luna
24.02.2025	18:20:00	18:50:00	00:30:00	Writing	Feature excluded from website	Luna
26.02.2025	08:00:00	12:30:00	04:30:00	Writing	Economic part finished	Luna
01.03.2025	18:00:00	20:00:00	02:00:00	Writing	Finishing website section Small improvements	Luna
02.03.2025	13:00:00	15:30:00	02:30:00	Writing	Pictures added Test results and analysis (Quantitative and qualitative) Comparison and selection	Luna
03.03.2025	09:20:00	13:00:00	03:40:00	Writing	Outlook Introduction Small improvements	Luna
03.03.2025	13:50:00	15:00:00	01:10:00	Writing	Flask Server structure	Luna

A.2. Florian Prandstetter

Date	Start	End	Hours	Tasks	Details	Persons
13.09.2024	08:50:00	09:20:00	00:30:00	Planning	Discussed and assigned the new idea	All (oL)
15.09.2024	14:00:00	15:00:00	01:00:00	Meeting		All (oL)
18.09.2024	16:09:00	16:50:00	00:41:00	Research	Searching for project components	Gabriel, Flo
19.09.2024	14:50:00	17:30:00	02:40:00	Research	OS for the Raspberry Pi 5	Flo

Continued on next page

Anhang A. Time Protocol

Date	Start	End	Hours	Tasks	Details	Persons
27.09.2024	17:30:00	20:00:00	02:30:00	Software Development	Testing DietPi on Raspberry Pi 5	Flo
04.10.2024	07:50:00	08:00:00	00:10:00	Meeting	Discussion of orders Definition of the parts list Cost allocation defined	All (oL)
10.10.2024	14:30:00	18:00:00	03:30:00	Hardware Test	HAILO on Raspberry test and hardware test	All (oL)
10.10.2024	19:00:00	21:00:00	02:00:00	Software Test	Operating system tested and first implementation attempts	Flo
17.10.2024	10:00:00	00:00:00	14:00:00	Research; Software Development	UI implementation methods comparison	Flo
17.10.2024	12:30:00	12:50:00	00:20:00	Meeting	Further discussion with Greinöcker regarding the thesis	Flo, Luna
21.10.2024	18:00:00	18:30:00	00:30:00	Research; Software Development	Selecting the server operating system	Flo
26.11.2024	18:40:00	20:13:00	01:33:00	Software Test	Visual Studio Code extension development	Flo
27.11.2024	08:34:00	11:51:00	03:17:00	Software Development	Visual Studio Code extension: First requests to the AI server	Flo
28.11.2024	08:10:00	11:35:00	03:25:00	Software Development	VS Code extension development	Flo
28.11.2024	12:30:00	13:50:00	01:20:00	Research; Software Development	VS Code extension chat with OLLAMA via AI server	Flo
28.11.2024	15:15:00	16:50:00	01:35:00	Research; Software Development	VS Code status bar updated	Flo
30.11.2024	13:40:00	16:25:00	02:45:00	Research; Software Development	Research on object recognition, program start of finger recognition for a user interface	Gabriel

Continued on next page

Date	Start	End	Hours	Tasks	Details	Persons
02.12.2024	08:00:00	08:50:00	00:50:00	Research; Software Development	Research on code completion	Flo
04.12.2024	08:00:00	12:28:00	04:28:00	Research; Software Development	VS Code extension / Inline chat /	Flo
12.12.2024	10:54:00	11:05:00	00:11:00	Software Development	Dark mode for VS Code extension	Flo
26.12.2024	13:31:00	17:00:00	03:29:00	Software Test	Server setup and installing software	Flo
27.12.2024	15:00:00	16:37:00	01:37:00	Software Test	Server VPN tunnel	Flo
30.12.2024	14:08:00	15:16:00	01:08:00	Software Test	Setting up Tailscale, network architecture	Flo
02.01.2025	14:00:00	15:52:00	01:52:00	Software Development	Extension	Flo
03.01.2025	14:22:00	15:39:00	01:17:00	Software Development	Extension	Flo
08.01.2025	11:00:00	11:30:00	00:30:00	Software Test	Setting up backend on home server	Flo
08.01.2025	11:30:00	12:32:00	01:02:00	Software Test	Website backend config and model testing	Flo, Luna
13.01.2025	13:00:00	15:20:00	02:20:00	Software Development	I hate my life (VS Code extension)	Flo
14.01.2025	08:00:00	10:59:00	02:59:00	Software Development	Visual Studio Code extension development	Flo
21.01.2025	10:00:00	11:00:00	01:00:00	Research	Model RAM setting (Fail)	Flo
29.01.2025	08:00:00	11:00:00	03:00:00	Writing	Writing the first part	Flo
09.02.2025	14:12:00	18:14:00	04:02:00	Software Development	Deepseek interface for server / Extension / Dockerizing	Flo
10.02.2025	10:00:00	12:30:00	02:30:00	Writing	Discussion, structure of the thesis, division, general writing	All (oL)

Continued on next page

Anhang A. Time Protocol

Date	Start	End	Hours	Tasks	Details	Persons
10.02.2025	16:35:00	17:54:00	01:19:00	Software Test	Docker	Flo
11.02.2025	14:45:00	15:31:00	00:46:00	Software Test	Docker	Flo
11.02.2025	16:50:00	17:54:00	01:04:00	Software Test	Docker (now it works)	Flo
11.02.2025	14:30:00	17:06:00	02:36:00	Writing	Docker + Docker Compose + Docker Images	Flo
14.02.2025	14:28:00	15:30:00	01:02:00	Writing	Hello Flo / Extension / TypeScript (+ axios)	Flo
26.02.2025	08:00:00	12:30:00	04:30:00	Writing	Operating system / Research economic aspects	Flo
27.02.2025	13:30:00	14:48:00	01:18:00	Writing	Operating system	Flo
01.03.2025	15:42:00	17:35:00	01:53:00	Writing	Operating system	Flo

Appendix

Tabellenverzeichnis

Abbildungsverzeichnis

2.1. Gantt Chart of the Project Evolution	10
2.2. Conceptual Illustration of the Self-Sufficiency Raspberry Pi Project	12
8.1. Comparative Analysis of CPU Utilization Across AI Models	67
8.2. Comparative Analysis of Memory Consumption Across AI Models	68
8.3. Comparative Analysis of Model Response Times	69
8.4. Comparison of BLEU and ROUGE Scores	70
8.5. Comparison of Grammatical Errors Across AI Models	72
8.6. Comparison of Readability Scores Across AI Models	73
8.7. Comparison of Sentiment Analysis Across AI Models	74
8.8. Combined Metrics Overview for AI Models	75
9.1. Flask Service Architecture	91
10.1. User Account Management and Overview	107
10.2. ChatGP API Pricing	112
10.3. Image to Text Tool	119
10.4. Main Navigation Bar	124
10.5. Chat Bot Navigation	124

Listings

8.1.	Python-quantitative-data-collection	54
8.2.	Python-data-preperation-for-analysis	56
8.3.	Python-quantitative-data-analysis	60
8.4.	Python-qualitative-data-analysis	64
8.5.	Vue.js Template for OpenAI API Integration	83
8.6.	Vue.js Script for OpenAI API Integration	84
10.1.	Initializing Firebase and setting up authentication	103
10.2.	Vue.js component for user sign-in	104
10.3.	Abbreviated Vue.js Integration Example	109
10.4.	Abbreviated Vue.js Component for the Programming Bot	115
10.5.	Abbreviated Vue.js Component for Image Recognition	117
10.6.	Abbreviated Vue.js Component for Image-to-Text Conversion	119
10.7.	Abbreviated Implementation of the Saved Chats Feature	121

Literaturverzeichnis