Diplomarbeit

Integration of Artificial Intelligence in Education and Software Development

Eingereicht von

Florian Prandstetter
Luna Schätzle

Eingereicht bei

Höhere Technische Bundeslehr- und Versuchsanstalt Anichstraße

Abteilung für Wirtschaftsingenieure/Betriebsinformatik

Betreuer

Mag. Dr. Dipl. -Ing. Albert Greinöcker
MMag.[a] Eva-Maria Egger, MA

Projektpartner

HTL Anichstraße
Innsbruck, April 2025

Abgabevermerk: Betreuer/in:

Datum:

Kurzfassung /Abstract

Abstract:

This diploma thesis examines the integration of Artificial Intelligence (AI) in educational and software development environments. Recognizing the increasing role of AI in automating processes, optimizing operations, and enhancing quality—as well as supporting personalized learning—the thesis addresses challenges such as complexity, limited expertise, and high development costs.

The primary goal was to develop AI-driven solutions for both fields. This involved setting up a server (API) to host AI models, developing a backend for an AI Hub and a Visual Studio Code Extension, and integrating suitable AI models. The work documents a shift from a broad evaluation of AI models to practical, focused applications.

A key aspect of the thesis is the implementation of Large Language Models (LLMs). Various free and commercial LLMs were evaluated using the Ollama application for local deployment and the OpenAI API, with quantitative and qualitative methods assessing response time, resource utilization, and text quality. This led to the selection and integration of specific models.

Additionally, a self-hosted Flask service was developed to link the frontend and backend components for both the Intelligent Student AI Hub and the Visual Studio Code Extension. The thesis also outlines the architecture of a web platform featuring an interactive chatbot, image-to-text tool, programming bot, and user account management, and includes economic evaluations of AI and open-source technologies. The project's source code is available on GitHub under the GPL-3.0 License.

Key findings highlight the transformative potential of AI in education and industry, and although some planned features were omitted due to time constraints, the modular architecture supports future enhancements.

Kurzfassung:

Diese Diplomarbeit untersucht die Integration Künstlicher Intelligenz (KI) in Bildungs- und Softwareentwicklungsumgebungen. Angesichts der zunehmenden Bedeutung von KI für Prozessautomatisierung, Optimierung und Qualitätssteigerung sowie zur Unterstützung personalisierten Lernens werden Herausforderungen wie Komplexität, Fachkräftemangel und hohe Entwicklungskosten beleuchtet.

Ziel war die Entwicklung KI-gestützter Lösungen für beide Bereiche, wozu die Einrichtung eines Servers (API) zum Hosten von KI-Modellen, die Entwicklung eines Backends für einen KI-Hub und eine Visual Studio Code Extension sowie die Integration geeigneter KI-Modelle gehören. Die Arbeit beschreibt den Übergang von einer breiten Evaluierung zu praxisnahen Anwendungen.

Ein Schwerpunkt liegt auf der Implementierung von Large Language Models (LLMs), die mittels der Ollama-Applikation und der OpenAI API bewertet wurden. Quantitative und qualitative Methoden zur Bewertung von Antwortzeit, Ressourcenauslastung und Textqualität führten zur Integration spezifischer Modelle.

Zudem wurde ein selbstgehosteter Flask-Dienst entwickelt, der als Schnittstelle zwischen Frontend und Backend für den Intelligent Student AI Hub und die Visual Studio Code Extension dient. Die Architektur der Webplattform – mit interaktivem Chatbot, Bild-zu-Text-Werkzeug, Programmierbot und Benutzerverwaltung – sowie wirtschaftliche Betrachtungen von KI- und Open-Source-Technologien runden die Arbeit ab. Der Quellcode ist auf GitHub unter der GPL-3.0 Lizenz verfügbar.

Die Ergebnisse unterstreichen das transformative Potenzial von KI in Bildung und Industrie und bilden eine Basis für zukünftige Erweiterungen.

Erklärung der Eigenständigkeit der Arbeit

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbständig 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

Danksagung / Acknowledgements

Deutsch

An dieser Stelle möchten wir unseren tief empfundenen Dank all jenen aussprechen, die uns während der Konzeption, Durchführung und Fertigstellung dieser Diplomarbeit tatkräftig unterstützt und begleitet haben. Ohne ihre unermüdliche Hilfe und die wertvollen Beiträge wäre die Realisierung dieser Arbeit in der vorliegenden Form nicht möglich gewesen.

Zunächst gilt unser besonderer Dank den betreuenden Professoren, die uns mit ihrer fachlichen Expertise, konstruktiven Kritik und kontinuierlichen Ermutigung zur Seite standen. Ihre fundierten Impulse haben maßgeblich dazu beigetragen, das Projekt methodisch zu schärfen und inhaltlich weiterzuentwickeln. Ebenso möchten wir unseren Familien und Freunden danken, die uns in allen Phasen – sowohl in akademischen als auch in persönlichen Herausforderungen – stets unterstützend begleitet haben.

Unser Dank richtet sich ferner an die HTL Anichstraße, die uns nicht nur den notwendigen Rahmen zur Durchführung dieses Projekts zur Verfügung gestellt, sondern uns auch mit den erforderlichen Ressourcen und einem inspirierenden Lernumfeld ausgestattet hat. Die hier gewonnenen Erfahrungen und Einblicke werden uns auf unserem weiteren beruflichen Weg nachhaltig begleiten und prägen.

Besonders hervorheben möchten wir folgende Personen, deren engagierte Unterstützung und wertvolle Hinweise einen entscheidenden Beitrag zum Erfolg dieser Arbeit geleistet haben:

- **Mag. Dr. Dipl. -Ing. Albert Greinöcker:** Für die fachliche Betreuung, die inhaltlichen Anregungen und die stetige Motivation, die maßgeblich zur kontinuierlichen Verbesserung dieser Arbeit beigetragen haben.

- **MMag.ª Eva-Maria Egger, MA:** For her economic supervision and comprehensive support during the development of the thesis, which propelled us forward during critical moments.MA Egger Eva-Maria: Für die wirtschaftliche Betreuung und die umfassende Unterstützung bei der Erstellung der Arbeit, die uns in entscheidenden Momenten vorangebracht hat.
- **Dr. Manuela Schätzle:** Für die konstruktiven Rückmeldungen, die sorgfältige Korrektur und die methodische Unterstützung, die wesentlich zur inhaltlichen und sprachlichen Qualität der Arbeit beigetragen haben.
- **Mag. Elke Peuschler:** Für die wertvollen Hinweise zur Verbesserung der Arbeit und die konstruktive Kritik, die uns in entscheidenden Phasen weitergebracht hat.
- **Mag. Michael Prandstetter:** Für wervolle Anregungen und kritisches Feedback, die zur Verbesserung der Arbeit beigetragen haben.
- **Niclas Sachse:** Für die kontinuierliche Unterstützung, die wertvollen Hinweise zur Verbesserung des Inhalts sowie die emotionale Rückendeckung in herausfordernden Phasen.

Unser Dank gilt all jenen, die auf unterschiedliche Weise zum Gelingen dieser Arbeit beigetragen haben. Wir schätzen die Unterstützung sehr und sind dankbar für das entgegengebrachte Vertrauen und die wertvollen Erfahrungen, die uns auf unserem weiteren Lebens- und Berufsweg begleiten werden.

English

We would like to express our heartfelt gratitude to everyone who supported and accompanied us throughout the conception, execution, and completion of this diploma thesis. Without their tireless assistance and invaluable contributions, the successful realization of this work would not have been possible.

First and foremost, our sincere thanks go to our supervising professors, whose expert guidance, constructive criticism, and constant encouragement played a pivotal role in refining the methodology and enhancing the overall content of the project. We are equally grateful to our families and friends, who have supported us through all academic and personal challenges, providing unwavering encouragement and assistance.

Our appreciation also extends to HTL Anichstraße, which not only provided us with the essential framework and resources to carry out this project but also fostered

an inspiring learning environment that significantly contributed to the success of this work. The experiences and insights gained here will undoubtedly influence and support our future professional endeavors.

We would like to especially acknowledge the following individuals, whose dedicated support and valuable advice have been instrumental to the success of this thesis:

- **Mag. Dr. Dipl. -Ing. Albert Greinöcker:** For his academic supervision, insightful suggestions, and continuous motivation, which have been crucial in the ongoing improvement of this work.
- **MMag.ª Eva-Maria Egger, MA:** For her economic supervision and comprehensive support during the development of the thesis, which propelled us forward during critical moments.
- **Dr. Manuela Schätzle:** For her constructive feedback, meticulous review, and methodological support, which significantly enhanced the content quality and linguistic precision of the work.
- **Mag. Elke Peuschler:** For her valuable suggestions for improving the work and constructive criticism that guided us through critical phases.
- **Mag. Michael Prandstetter:** For his valuable insights and critical feedback that contributed to the enhancement of the work.
- **Niclas Sachse:** For his consistent support, valuable input for content enhancement, and emotional backing during challenging phases.

We extend our heartfelt thanks to all those who contributed in various ways to the success of this work. We deeply appreciate their support, the trust placed in us, and the invaluable experiences that will continue to guide us on our future academic and professional paths.

Information zur Verwendung von Künstlicher Intelligenz in dieser Arbeit / Information on the Use of Artificial Intelligence in this Work

Deutsch

Die vorliegende Diplomarbeit nutzt Künstliche Intelligenz (KI) als integralen Bestandteil des Forschungs- und Schreibprozesses. Ziel war es, die wissenschaftliche Arbeit durch den Einsatz moderner Technologien zu unterstützen, ohne die intellektuelle Eigenleistung und kritische Bewertung der Autorin bzw. des Autors zu ersetzen. Dabei wurde großer Wert darauf gelegt, sämtliche rechtlichen, ethischen und methodischen Aspekte vollumfänglich zu berücksichtigen.

Anwendungsbereiche und Einsatz der KI

Im Rahmen dieser Arbeit kamen verschiedene KI-basierte Tools zum Einsatz:

- **Textgenerierung und Sprachliche Optimierung:** Unterstützung bei der Formulierung, Strukturierung und stilistischen Verfeinerung wissenschaftlicher Inhalte.
- **Datenanalyse:** Verarbeitung und Analyse großer Datenmengen zur Identifikation relevanter Muster und Erkenntnisse.
- **Feedback und Verbesserungsvorschläge:** Generierung konstruktiver Hinweise zur inhaltlichen und methodischen Optimierung.

Zu den in diesem Projekt verwendeten KI-Tools zählen unter anderem:

- ChatGPT
- GitHub Copilot
- Deepseek
- Ollama Modells
- Mistrals le Chat

Wissenschaftliche Methodik und Validierung

Alle durch KI generierten Inhalte und Vorschläge wurden einer sorgfältigen wissenschaftlichen Überprüfung unterzogen. Die methodische Vorgehensweise umfasste:

- Eine manuelle Prüfung aller KI-Ergebnisse hinsichtlich inhaltlicher Richtigkeit und Relevanz.
- Die Integration der KI-bezogenen Ergebnisse in den Gesamtzusammenhang der Arbeit unter strenger Beachtung der etablierten wissenschaftlichen Qualitätsstandards.
- Eine kritische Reflexion der Limitationen der verwendeten KI-Methoden, um Verzerrungen und fehlerhafte Interpretationen zu vermeiden.

So wurde sichergestellt, dass die KI-gestützten Beiträge ausschließlich als ergänzende Hilfsmittel zur Steigerung der Effizienz und Qualität der Arbeit verwendet wurden.

Rechtliche Absicherung und Haftungsausschluss

Die Nutzung der genannten KI-Tools erfolgte unter strikter Einhaltung aller geltenden gesetzlichen Bestimmungen. Hierzu zählen insbesondere:

- Die Einhaltung der Datenschutzbestimmungen, insbesondere im Hinblick auf die Verarbeitung personenbezogener Daten.
- Die Berücksichtigung der spezifischen Nutzungsbedingungen und vertraglichen Vereinbarungen der jeweiligen Anbieter.

Es wird ausdrücklich darauf hingewiesen, dass die alleinige Verantwortung für die wissenschaftliche Integrität und die inhaltliche Richtigkeit dieser Arbeit bei der Autorin bzw. dem Autor liegt. Die von den KI-Tools generierten Ergebnisse sind als unterstützende Instrumente zu verstehen und stellen keine eigenständigen

wissenschaftlichen Erkenntnisse dar. Alle automatisiert erzeugten Inhalte wurden manuell überprüft und bei Bedarf korrigiert. Mit der Einreichung dieser Arbeit bestätigt die Autorin bzw. der Autor, dass sämtliche rechtlichen, ethischen und methodischen Rahmenbedingungen eingehalten wurden.

Ethische Überlegungen und Transparenz

Neben der Einhaltung rechtlicher Vorgaben wurde auch auf die ethische Dimension des KI-Einsatzes geachtet. Transparenz im Umgang mit den eingesetzten Technologien war ein zentraler Aspekt dieser Arbeit:

- Es wurde offen gelegt, welche KI-Tools zum Einsatz kamen und in welchen Bereichen sie genutzt wurden.
- Die Limitationen der KI-gestützten Verfahren wurden kritisch reflektiert, um eine einseitige Darstellung der wissenschaftlichen Ergebnisse zu vermeiden.
- Der Einsatz der KI diente ausschließlich der Unterstützung und nicht der Substitution der eigenständigen wissenschaftlichen Arbeit.

English

This diploma thesis integrates Artificial Intelligence (AI) as a fundamental component of the research and writing process. The primary objective was to enhance the quality and efficiency of the academic work through modern technological support, while ensuring that the intellectual contribution and critical oversight of the author remain paramount. Special care was taken to fully comply with all relevant legal, ethical, and methodological requirements.

Application Areas and Use of AI

In this work, various AI-based tools were employed:

- **Text Generation and Linguistic Optimization:** Assist in formulating, structuring, and refining the stylistic aspects of academic content.
- **Data Analysis:** Process and analyze large datasets to identify significant patterns and insights.

Florian Prandstetter
Luna Schätzle

- **Feedback and Improvement Suggestions:** Generate constructive recommendations for content and methodological enhancements.

The AI tools used in this project include, but are not limited to:

- ChatGPT
- GitHub Copilot
- Deepseek
- Ollama Modells
- Mistrals le Chat

Scientific Methodology and Validation

All outputs and suggestions generated by AI were subjected to rigorous scientific review. The methodological approach included:

- A thorough manual review of all AI-generated content to ensure its accuracy and relevance.
- Integrating the AI-assisted results within the broader context of the thesis while strictly adhering to established scientific quality standards.
- Critically reflecting on the limitations of the AI methods employed to avoid biases and erroneous interpretations.

This process ensured that AI-based contributions were used solely as auxiliary tools to enhance the efficiency and quality of the work.

Legal Safeguards and Disclaimer

The use of the aforementioned AI tools was carried out in strict accordance with all applicable legal requirements. In particular, the following measures were observed:

- Adherence to data protection laws, especially regarding the processing of personal data.
- Consideration of the specific terms of service and contractual agreements of the respective providers.

It is expressly stated that the sole responsibility for the scientific integrity and accuracy of this work rests with the author. The outputs generated by the AI tools are to be regarded as supplementary aids and do not constitute independent scientific findings. All automated outputs were manually verified and corrected if necessary. By submitting this work, the author confirms that all legal, ethical, and methodological frameworks have been fully observed.

Ethical Considerations and Transparency

In addition to legal compliance, significant emphasis was placed on the ethical aspects of using Artificial Intelligence. Transparency regarding the employed technologies was a key element of this work:

- Full disclosure of the AI tools used and the areas in which they were applied.
- A critical reflection on the limitations of the AI-assisted methods to prevent biased or incomplete representation of the scientific results.
- Ensuring that the use of AI served solely as a supportive tool and did not replace the author's independent scholarly work.

Contents

Part I.

Introduction

1. Introduction: AI in the Industry and Education Environment

Author: Luna Schätzle

Author: Florian Prandstetter 1.3.2

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:** 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.

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 and Detailed Task Description

The project team consists of three members:

- Luna Schaetzle (Project lead)
- Florian Prandstetter

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

1.3.1. 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.

1.3.2. Florian Prandstetter

Florian Prandstetter is in charge of some key tasks to ensure the projects success. These tasks are:

- **Setting up a server:** Building and maintaning a server to host the AI models.
- **Managing the Servers Operating System:** Installing and managing the operating system of the server.
- **Managing a Docker Service:** Configuring and managing a Docker service on the server.
- **Developing an VS Code extension:** Developing an extension for the Visual Studio Code editor.
- **Analyzing the OS Market:** Analyzing the Operating System market and its different competitors.

1.4. 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: Server** - Details the server component, including the hardware, infrastructure, and operating system.

- **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: Economic Evaluations** — Examines the economic aspects of AI in education and industry, including evaluations of open-source approaches and their financial implications.
- **Part 6: 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]

1.5. Source Code Repository

The project's source code is hosted on GitHub and is accessible via the following repositories:

- **General experimentation Repository**: https://github.com/Luna-Schaetzle/Diploma-thesis
- **Repository for the Backend and Frontend of the AI Hub**: https://github.com/Luna-Schaetzle/Diploma-thesis-website
- **Repository for the VS Code Extension**: https://github.com/jamati123/luminara-coworker
- **The Repository for the Documentation**: https://github.com/Luna-Schaetzle/Diploma-thesis-document

All team members have access to these repositories and can actively contribute by committing and pushing changes to the designated branches. The repositories are licensed under the GPL-3.0 License, permitting the free use, modification, and distribution of the source code while ensuring compliance with open-source principles.

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

2. Conceptual Evolution and Rationale

Authors: Luna Schaetzle

This chapter presents a comprehensive analysis of the conceptual and technical evolution of the project within the context of this Diploma Thesis. The investigation delineates the progression from an initial theoretical proposal, through intermediate developmental phases such as the Self-Sufficiency Raspberry Pi Project, to the formulation of the final design concept. Documenting this iterative transformation is critical to achieving a holistic understanding of the design process, as it underscores the dynamic interplay between feasibility constraints, scalability requirements, and alignment with overarching research objectives.

The discussion elucidates the key motivations, challenges, and pivotal decisions that shaped the project's trajectory, with particular emphasis on evaluating technical and operational feasibility, ensuring scalable implementation frameworks, and maintaining coherence with core research goals. Furthermore, a critical examination of the technical obstacles encountered during prototyping and implementation phases is provided, highlighting how empirical insights informed subsequent refinements. By synthesizing these elements, this chapter not only chronicles the project's developmental arc but also contextualizes the methodological rigor inherent in balancing theoretical aspirations with practical execution.

2.1. 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.

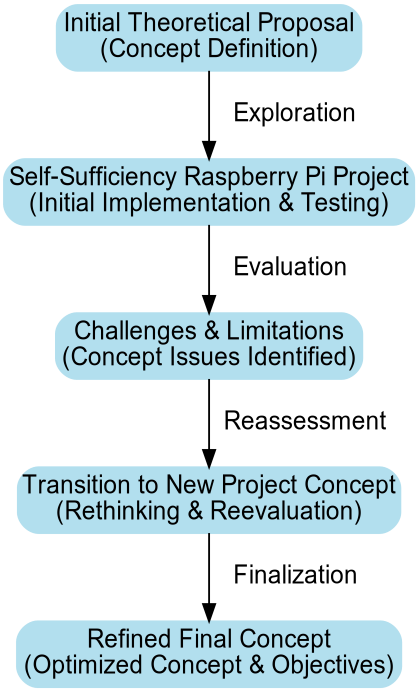

Figure 2.1.: Gantt Chart of the Project Evolution

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.

2.2. 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.

2.3. The Self-Sufficiency Raspberry-PI 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.

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

DTeK (2023)

2.4. Challenges and Limitations of the Self-Sufficiency Raspberry-PI 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.

2.5. 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:

2.5.1. 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.

2.5.2. 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. 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.

2.6.1. Challenges

The primary challenges encountered during this transition included:

- **Technical Complexity:** Integrating advanced AI models with diverse platforms—such as web applications and code editors 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.
- **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 definable, 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.

2.6.2. 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.
- **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.

2.7. 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.

2.8. 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 and AI Integration in Software Development 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, igorous 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.

Part II.

Server

3. Server Hardware

Author: Florian Prandstetter

3.1. Introduction

The foundation of every server is its hardware. The hardware consists of the physical components that make up the server. It is responsible for the performance, reliability, and scalability of the server.

In this chapter, the different components of a server will be discussed, and the importance of each component will be explained.

3.2. Server Components

The main components of a server are the Processor, the Graphics Processing Unit, the Random Access Memory, the Power Supply, the Motherboard and the Data Storage. Without these components the server cannot work. Thus choosing suitable parts is a very important task.

3.2.1. Processor

The Central Processing Unit is the brain of the server. It is responsible for executing the instructions of the software.

If the CPU is too slow, the server will not be able to handle the workload, creating a significant bottleneck. However, high-performance CPUs are expensive, so it is important to find the right balance between performance and cost.

In this project, an Intel Core i5-8600K was used. This CPU has 6 cores and a base clock speed of 3.6 GHz, which is sufficient to handle the server's workload.

Intel (n.d.)

3.2.2. Graphics Processing Unit

The Graphics Processing Unit is responsible for rendering images and videos. It is usually used to process images, videos, and other graphical tasks, but it can also be used for AI tasks.

The GPU is much faster than the CPU when it comes to parallel processing. This makes it ideal for AI tasks, which typically involve parallel computations.

In this project, an NVIDIA GeForce RTX 2060 was used. This GPU has 1920 CUDA cores and 6GB of GDDR6 memory, which is sufficient for handling the AI tasks of the server.

Contributors (n.d.*a*) NVIDIA (n.d.)

3.2.3. Random Access Memory

Random Access Memory is a type of memory that stores data currently in use. It is much faster than storage memory, but it is also more expensive. The more RAM a server has, the more data it can store and access quickly.

For servers, it is important to have enough RAM to handle the workload. If the server runs out of RAM, it will start using storage memory, which is significantly slower.

For the AI server in this project, 16GB of DDR4 RAM was used. This is sufficient to handle the workload while keeping costs low.

Contributors (n.d.*c*)

3.2.4. Motherboard

The Motherboard is the main circuit board of the server. It connects all the components and allows them to communicate with each other.

An important factor when choosing a Motherboard is its compatibility with other components. If the Motherboard is not compatible with the CPU, GPU, or RAM, the server will not function properly.

To avoid compatibility issues, an H370 Motherboard with an Intel socket was used. This Motherboard is compatible with the CPU, GPU, and RAM selected for this project.

Kirvan (n.d.)

3.2.5. Power Supply

The Power Supply is responsible for providing power to the server. It converts electricity into the appropriate voltage for the server components.

It is crucial to select a Power Supply that can handle the server's power requirements. If the Power Supply is too weak, the server will not function. If it is too strong, it will waste energy.

In this project, a 500W Power Supply was used. This is sufficient to power the CPU, GPU, and RAM of the server.

Contributors (n.d.*b*)

3.2.6. Storage

To store AI models and data, storage memory is required. Storage memory is much slower than RAM, but it is also much more affordable.

There are different types of storage memory, such as Hard Drive Disk and Solid State Drives. SSDs are much faster than HDDs, but they are also more expensive.

In this project, a 512GB NVMe SSD was used. This provides enough storage capacity for AI models.

IBM (2024*c*)

3.3. Conclusion

The hardware of a server is the foundation of its performance, reliability, and scalability. Selecting the right components is essential to building a server that can handle its workload while keeping costs manageable.

The components used in this project are mostly consumer-grade. This keeps costs low while providing adequate performance for the server.

For a production server, it is recommended to use server-grade components. These components are more expensive but offer greater reliability and better performance.

4. Server Infrastructure

Author: Florian Prandstetter

4.1. Introduction

The server infrastructure is the backbone of every IT system. It is responsible for hosting the applications and services that are used by the clients. A well-designed server infrastructure can significantly improve the server performance. It also ensures that the server is reliable and scalable.

In this chapter, the different components of a server infrastructure will be discussed. The importance of each component will be explained.

4.2. Server Infrastructure Components

The main components of the projects server infrastructure are the following:

4.2.1. Server Hardware

The server hardware consists of the physical components that make up the server. It is explained in more detail in Chapter 3.

4.2.2. Networking

The networking components of a server infrastructure are responsible for connecting the server to the internet and other devices. This includes routers, switches, and firewalls.

While networking is a broad topic, it is essential to have a basic understanding of how it works to build a server infrastructure. Security is also an important aspect of networking. Firewalls and other security measures are used to protect the server from cyber attacks.

To ensure security and reliability, the server in this project is connected to a secure network. The network is protected by a firewall and other security measures to prevent unauthorized access.

IBM (2024*a*)

4.2.3. Remote Management

Remote management tools are used to monitor and manage the server from a remote location. This is essential for maintaining the server and ensuring that it is running smoothly.

There are different remote management tools available, such as SSH, RDP, and VNC.

The server in this project is managed using SSH. This allows the server administrator to access the server remotely and perform maintenance tasks. To access the server from a different network, a VPN connection is used. This ensures that the connection is secure and encrypted. The tool used for the VPN connection is Tailscale.

Joos (2023) **?**

4.2.4. Backup and Recovery

Regular backups are essential to ensure that data is not lost in case of a hardware failure or other disaster. There are different backup methods, such as full backups,

incremental backups, and differential backups. It is important to have a backup strategy in place to ensure that data can be recovered quickly and efficiently.

The backups for the server in this project are stored on an external hard drive. This hard drive is connected to the server and automatically backs up the data at regular intervals.

4.2.5. Containerization

Containerization is a method of packaging and deploying applications in a lightweight, isolated environment called a container. Containers are portable and can run on any system that supports containerization. This makes it easy to deploy applications and services across different environments.

The server in this project uses Docker for containerization. Docker allows the server administrator to package applications and services in containers and deploy them on the server. This makes it easy to manage and scale the server infrastructure. For a more detailed explanation of Docker, refer to Chapter 8.7.

IBM (2024*b*)

4.3. Conclusion

The server infrastructure is a critical component of any IT system. It is responsible for hosting the applications and services that are used by the clients.

By understanding the different components of a server infrastructure, server administrators can design and implement a robust and reliable server infrastructure that meets the needs of the organization.

In this project, the server infrastructure is designed to be secure and reliable. By using the right hardware, networking components, and other components, the server infrastructure can provide the necessary resources and services to support the AI Hub and other applications.

Part III.

Theoretical foundations

5. Operating System

Author: Florian Prandstetter

5.1. 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.

5.2. 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.

manjeetks007 (2024)

5.2.1. 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.

akash1295 (2025)

5.3. Kernel

The Kernel is the core of the Operating System. It is responsible for managing the system's resources and providing a bridge between the hardware and software components. The Kernel is responsible for managing the CPU, memory, and input/output devices. It also provides essential services like process management, memory management, and device management. When building an AI Service, the Kernel plays a crucial role in ensuring the system's stability and performance.

Kernels can classified into different categories. The categories depend on the architecture of the OS. There are two main types of kernel architectures, monolithic kernels and microkernels.

5.3.1. Monolithic Kernels

In this architecture, the entire kernel operates as a single large block of code that directly controls and manages hardware resources. While this can provide greater performance because all operations run in one space, it can also lead to higher complexity and harder maintenance.

5.3.2. Microkernels

A microkernel design is more minimalistic, with only essential services like process management and communication being handled by the kernel. Other functions, like device management or file systems, are run as separate processes. This separation can improve stability and security but may come at the cost of performance since communication between the kernel and other components can introduce overhead.

5.4. Key Differnceses of Microsoft Windows and Linux Kernels

While both Linux and Microsoft Windows serve as the core of their respective operating systems, their kernels differ significantly in terms of design and functionality.

5.4.1. Linux Kernel

- **Open Source**: The Linux kernel is open-source, anyone can view, modify, and distribute its code. This gives developers a high level of flexibility and customization.
- **Monolithic Architecture**: The Linux kernel is considered monolithic, meaning it includes all the key components necessary for running the system in one large block of code. This results in faster communication between components and potentially better performance, but at the cost of complexity in its implementation.
- **Modular**: While Linux is monolithic, it also allows modularity. You can add or remove specific kernel modules dynamically without rebooting the system.
- **Stability and Performance**: The Linux kernel is renowned for its stability and performance, especially in server environments. It is highly optimized for resource management and efficient multitasking, making it a popular choice for high-performance AI servers.

5.4.2. Microsoft Windows Kernel

- **Closed Source**: The Windows kernel is proprietary and closed-source, which limits the ability for users to modify or extend the kernel's capabilities. However, it provides a highly integrated user experience designed for ease of use and compatibility.
- **Hybrid Architecture**: While the Windows kernel is primarily based on a hybrid architecture, it blends aspects of both monolithic and microkernel designs. The core kernel handles basic tasks like process management and system calls, while additional services are handled by user-space applications.
- **Compatibility**: The Windows kernel is designed to provide excellent compatibility with a wide range of hardware, especially consumer devices like laptops and desktops. While it offers strong support for graphical interfaces

and is known for its user-friendly experience, it is not as well-suited for high-performance AI workloads when compared to Linux.
- **System Overhead**: Due to its larger footprint and the inclusion of more background services the Windows kernel can sometimes introduce more system overhead, which may not be ideal for resource-intensive applications like AI modeling or training.

pp$_p$*ankaj* (2025)

5.5. 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.

5.5.1. 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 OS needs to be sutiable to performe many different tasks on a high levle to be suitable for an AI server. Tasks like containerization, netwrok monitoring, process monitoring and scripting tasks are essential for ths server to operate

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

Cao et al. (2024)

5.5.2. Windows for Server

Microsoft provides a stable and secure OS to use on servers. There is a wide span of easy to implement native services that are can be implement easily. But due to many compatibility, performance and flexibility issues with popular AI development tools, it is not widely used for AI servers. Linux based systems offer

a more resource efficient and flexible enviorment. Thus Windows does not make the cut for the projects needs.

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. Trough the apt packet manager it provides a consistent tool to install and update any relevant packages.

Pontikis (2013)

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. But due to the frequent updates, good support and large amount of available packages the cost does not proove to be a big hinderance.

Hat (2024)

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. The OS can flawlessly run Docker multiple Docker containers even on lower end hardware due the the lightweight structure. Also it Debian foundation makes it easy to install any needed packages.

LaCroix (2022)

5.5.3. Outcome of the Evaluation

There are many factors that need to be considered when choosing a suitable OS for a specific application. Depending on the requirements the ideal OS for a job can needs to be evaluated very carefully. Often times it also comes down to the experience and skills of the development team working on a project.

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. Used Programming Languages

Author: Luna Schätzle

Author: Florian Prandstetter 6.3

6.1. Python

Python is a high-level, interpreted programming language renowned for its simplicity, readability, and versatility. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python's design philosophy emphasizes code readability, notably using significant indentation, which enhances the clarity and maintainability of its codebase. Its comprehensive standard library and supportive community have contributed to its widespread adoption across various domains, such as web development, data science, machine learning, and automation.

Key features of Python include:

- **Easy to Code:** Python's syntax is clear and concise, making it accessible to both beginners and experienced programmers.
- **Free and Open Source:** Python is freely available for use and distribution, including for commercial purposes, under an open-source license.
- **Object-Oriented Language:** Python supports object-oriented programming paradigms, facilitating code reuse and modularity.
- **GUI Programming Support:** Python offers libraries like Tkinter, PyQt, and wxPython for developing graphical user interfaces.
- **High-Level Language:** As a high-level language, Python abstracts complex details of the computer's hardware, allowing developers to focus on coding solutions without worrying about memory management.

For a comprehensive understanding of Python's features and capabilities, refer to the official Python documentation *Introduction to Python* (n.d.).

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

6.1.1. 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.

Gul (2024)

6.1.2. JSON

JavaScript Object Notation (JSON) is a lightweight, text-based data interchange format that is easy for humans to read and write, and straightforward 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 Foundation (2025).

6.2. 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.

6.2.1. 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.

HTML - Wikipedia (2001)

6.2.2. 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.

CSS Introduction (n.d.)

6.2.3. 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. *JavaScript Tutorial* (n.d.)

6.2.4. 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 *Introduction | Vue.js* (n.d.)

6.3. 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.

6.3.1. 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. *Axios Docs* (2025)

Part IV.

Implementation of Large Language Models

7. Overview and Integration of Large Language Models

Author: Luna Schätzle

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

- LLMs (Large Language Models)
- Diffusion 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 and identify faces in images or videos)
- Speech Recognition Models (Models that are used to recognize and transcribe speech)
- Speech Synthesis Models (Models that are used to generate human-like speech)
- Translation Models (Models that are used to translate text from one language to another)

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

7.1. 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.

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.

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.

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.

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.

IBM (n.d.*a*)

7.2. 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. *Overview - OpenAI API* (n.d.*a*)

7.3. 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. Ankush (2024)

7.3.1. 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.

7.3.2. 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.

7.3.3. 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.

7.3.4. 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.

7.3.5. 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.

7.3.6. 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.

A Comprehensive Guide to Ollama - Cohorte Projects (n.d.)

7.4. 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.

Ollama (n.d.*a*)

7.4.1. 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.

7.4.2. 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.

7.4.3. 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.

7.5. Ollama Model Testing and Evaluation

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

7.5.1. 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.

7.5.2. 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. Brownlee (2017)
- **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.

GeeksforGeeks (2024)

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.

7.5.3. 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. Liu et al. (2023)

Ollama (n.d.*b*)

7.5.4. Data Collection

For the Data collection we used the School AI Server wich 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 different python scripts. For the quantitative data we used the following python script:

```python
import time
import json
import psutil  # For CPU, memory usage
from ollama import chat

# Prompts for testing
prompts = [
    "Explain the theory of relativity in simple terms.",
    "Create a short story about a knight.",
    "What are the advantages of open-source projects?",
    "Write a Python function that outputs prime numbers up to 100.",
    # ...
]

# Model name
model_name = "qwen2-math"

# Store results
results = []

# Function to get GPU usage if available
def get_gpu_usage():
    try:
        import torch
        if torch.cuda.is_available():
            gpu_memory = torch.cuda.memory_allocated() / (1024 ** 2)  # Convert to MB
            gpu_utilization = torch.cuda.utilization(0) if hasattr(torch.cuda, 'utilization') else
                "N/A"
            return gpu_memory, gpu_utilization
        else:
            return 0, "No GPU detected"
    except ImportError:
        return 0, "torch not installed"

# Loop through prompts
for prompt in prompts:
    try:
        # Measure system usage before model execution
        cpu_before = psutil.cpu_percent(interval=None)
        memory_before = psutil.virtual_memory().used / (1024 ** 2)  # Convert to MB

        start_time = time.time()
        # Ollama chat request
        response = chat(model=model_name, messages=[{'role': 'user', 'content': prompt}])
        end_time = time.time()

        latency = end_time - start_time

        # Measure system usage after model execution
        cpu_after = psutil.cpu_percent(interval=None)
        memory_after = psutil.virtual_memory().used / (1024 ** 2)  # Convert to MB

        cpu_usage = cpu_after - cpu_before
        memory_usage = memory_after - memory_before
        gpu_memory_usage, gpu_utilization = get_gpu_usage()

        # Extract content from the Message object
        if response and hasattr(response["message"], "content"):
            response_text = response["message"].content  # Accessing the attribute of the Message
                object
        else:
            response_text = "No content returned or unexpected format"

        print(f"Prompt: {prompt}\nResponse Time: {latency:.2f} seconds\n")

        # Save the result
        results.append({
            "Prompt": prompt,
            "Response Time (seconds)": latency,
            "Response": response_text,
            "CPU Usage (%)": cpu_usage,
```

```
            "Memory Usage (MB)": memory_usage,
            "GPU Memory Usage (MB)": gpu_memory_usage,
            "GPU Utilization (%)": gpu_utilization
        })
    except Exception as e:
        print(f"Error with prompt '{prompt}': {e}")
        results.append({
            "Prompt": prompt,
            "Response Time (seconds)": "Error",
            "Response": f"Error: {str(e)}",
            "CPU Usage (%)": "N/A",
            "Memory Usage (MB)": "N/A",
            "GPU Memory Usage (MB)": "N/A",
            "GPU Utilization (%)": "N/A"
        })20.01.2025

# Save to JSON file
json_file_name = model_name + "_response_time_results_ressours_usage.json"
with open(json_file_name, "w") as file:
    json.dump(results, file, indent=4)

print(f"The results have been saved in {json_file_name}.")
```

Listing 7.1: Python-quantitative-data-collection

In this Python script, ChatGPT was employed as an auxiliary tool to facilitate the integration of the psutil library for monitoring CPU and memory usage. Additionally, ChatGPT assisted in generating supplementary questions to enhance the data collection process. The resulting data were subsequently stored in JSON format for later analysis. However, a significant challenge was encountered: the Torch libraries did not function properly on the school AI server, which prevented the collection of GPU usage data 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.

psutil · PyPI (2024)

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.

ollama/ollama-python: Ollama Python library (n.d.)

7.5.5. Data Preperation

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

```python
import json
import os
from nltk.translate.bleu_score import sentence_bleu, SmoothingFunction
from rouge_score import rouge_scorer
import language_tool_python
import textstat
from transformers import pipeline, logging

# Suppress warning messages from the Transformer library for a cleaner output.
logging.set_verbosity_error()

def load_json(file_path):
    """
    Load and parse JSON data from a file.

    Parameters:
        file_path (str): The file system path to the JSON file.

    Returns:
        dict or list: The JSON data parsed from the file.
    """
    with open(file_path, 'r') as file:
        return json.load(file)

def calculate_metrics(data):
    """
    Compute multiple evaluation metrics for generated text responses.

    For each data item, the function calculates:
    - BLEU Score: Quantifies the similarity between the generated response and the reference text.
    - ROUGE Scores: Evaluates the n-gram overlap between the reference and the generated text,
          using ROUGE-1, ROUGE-2, and ROUGE-L.
    - Grammar Check: Determines the number of grammatical errors present in the response.
    - Readability Score: Computes the Flesch Reading Ease score to assess the text's readability.
    - Sentiment Analysis: Infers the sentiment polarity (e.g., positive or negative) of the
          response text.

    Parameters:
        data (list): A list of dictionaries, each containing 'Prompt', 'Response', and 'Reference'
              keys.

    Returns:
        list: A list of dictionaries that include the original text elements along with the
              computed metrics.
    """
    results = []
    rouge = rouge_scorer.RougeScorer(['rouge1', 'rouge2', 'rougeL'], use_stemmer=True)
    grammar_tool = language_tool_python.LanguageTool('en-US')
```

```
    sentiment_analyzer = pipeline(
        'sentiment-analysis',
        model="distilbert-base-uncased-finetuned-sst-2-english",
        truncation=True,
        max_length=512
    )

    for item in data:
        prompt = item['Prompt']
        response = item['Response']
        reference = item['Reference']

        try:
            # Calculate the BLEU Score with smoothing to address potential issues with short sequences.
            bleu_score = sentence_bleu(
                [reference.split()], response.split(),
                smoothing_function=SmoothingFunction().method4
            )

            # Calculate ROUGE Scores for comprehensive n-gram overlap assessment.
            rouge_scores = rouge.score(reference, response)

            # Perform grammatical analysis by counting detected errors in the response.
            grammar_errors = len(grammar_tool.check(response))

            # Determine the readability score using the Flesch Reading Ease metric.
            readability_score = textstat.flesch_reading_ease(response)

            # Analyze the sentiment of the response text, with error handling to capture any exceptions.
            try:
                sentiment_result = sentiment_analyzer(response)[0]
                sentiment = sentiment_result['label']
            except Exception as e:
                print(f"Sentiment error for prompt '{prompt}': {e}")
                sentiment = "Error"

            results.append({
                "Prompt": prompt,
                "Response": response,
                "Reference": reference,
                "BLEU": bleu_score,
                "ROUGE-1": rouge_scores['rouge1'].fmeasure,
                "ROUGE-2": rouge_scores['rouge2'].fmeasure,
                "ROUGE-L": rouge_scores['rougeL'].fmeasure,
                "Grammar Errors": grammar_errors,
                "Readability Score": readability_score,
                "Sentiment": sentiment
            })
        except Exception as e:
            print(f"Error processing prompt '{prompt}': {e}")
            results.append({
                "Prompt": prompt,
                "Response": response,
                "Reference": reference,
                "Error": str(e)
            })

    return results

def save_results(results, output_path):
    """
    Persist the computed metrics to a JSON file.

    Parameters:
        results (list): A list of dictionaries containing evaluation metrics and corresponding texts.
        output_path (str): The file system path where the output JSON should be saved.
    """
    with open(output_path, 'w') as file:
        json.dump(results, file, indent=4)

def main():
    """
```

```
    Execute the main workflow of the script.

    This function prompts the user to specify a directory containing preprocessed JSON files.
    It then iterates through each file that matches the pattern 'processed_*.json',
    computes the evaluation metrics for the contained data,
    and saves the results in a new JSON file prefixed with 'scored_'.
    """
    directory = input("Enter the directory containing the processed JSON files: ")

    try:
        for file_name in os.listdir(directory):
            if file_name.startswith("processed_") and file_name.endswith(".json"):
                input_file = os.path.join(directory, file_name)
                model_name = file_name.split("processed_")[1].split(".json")[0]
                output_file = os.path.join(directory, f"scored_{model_name}.json")

                print(f"Processing file: {input_file}")
                data = load_json(input_file)
                metrics = calculate_metrics(data)
                save_results(metrics, output_file)
                print(f"Metrics saved to: {output_file}")

    except FileNotFoundError:
        print(f"Error: The directory {directory} was not found.")
    except Exception as e:
        print(f"An error occurred: {e}")

if __name__ == "__main__":
    main()
```

Listing 7.2: 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 Bird et al. (2025).

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. Otten (2024)

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 Bansal & Aggarwal (2025).

7.5.6. 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.

Quantitative Data Analysis

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

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import glob
import numpy as np
import os
import logging

# Set up logging for consistent error and information messages.
logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

# Set scientific plotting style with an increased default figure height.
plt.style.use('default')
sns.set_theme(style="whitegrid", context="paper")
plt.rcParams.update({
      # Here are all the params for the settings for matplotlib
})

def load_and_process_data() -> pd.DataFrame:
    """
    Loads and processes all JSON files in the current directory.

    This function searches for all files matching "*.json", reads them into pandas DataFrames,
    assigns a 'Model' column based on the file name (without extension), converts specified
    numeric columns to numeric type, and removes rows with missing values in those columns.

    Returns:
        pd.DataFrame: A concatenated and cleaned DataFrame containing all data.
    """
    json_files = glob.glob("*.json")
    if not json_files:
        logging.warning("No JSON files found in the current directory.")
        return pd.DataFrame()

    dfs = []
    for file in json_files:
        try:
            model_name = os.path.splitext(file)[0]
            df = pd.read_json(file)
            df['Model'] = model_name
            dfs.append(df)
        except Exception as e:
            logging.error(f"Error loading {file}: {e}")

    if not dfs:
        logging.error("No data could be loaded from the JSON files.")
        return pd.DataFrame()

    combined_df = pd.concat(dfs, ignore_index=True)

    # Convert selected columns to numeric and drop rows with missing values in these columns.
    numeric_cols = ['Response Time (seconds)', 'CPU Usage (%)', 'Memory Usage (MB)']
    combined_df[numeric_cols] = combined_df[numeric_cols].apply(pd.to_numeric, errors='coerce')
    combined_df = combined_df.dropna(subset=numeric_cols)

    return combined_df

def create_resource_plot(df: pd.DataFrame, metric: str, title: str, ylabel: str, filename: str)
     -> None:
    """
    Creates a resource usage plot with violin and strip plots, annotated with statistical
         measures.

    The function generates a violin plot for the given metric across different AI models,
         overlays a strip plot
    to display individual data points, and annotates each model with its median and mean absolute
         deviation (MAD).
    The resulting plot is saved in both PDF and PNG formats.

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

```
                xtick,
                median_val + 0.05,
                f'{median_val:.2f}s',
                ha='center',
                va='bottom',
                fontsize=8,
                color='#2f2f2f'
            )

        plt.title('Comparative Analysis of Model Response Times', pad=15)
        plt.xlabel('AI Model', labelpad=10)
        plt.ylabel('Response Time (seconds)', labelpad=10)
        plt.xticks(rotation=45, ha='right')
        plt.tight_layout()

        plt.savefig('model_response_times_comparison.pdf', bbox_inches='tight')
        plt.savefig('model_response_times_comparison.png', bbox_inches='tight')
        plt.close()

    def generate_statistics(df: pd.DataFrame) -> None:
        """
        Generates a statistical summary of response times for each AI model and exports the results.

        The summary includes the mean, standard deviation, minimum, median, and maximum values.
        The results are printed to the console and saved as a LaTeX table.

        Parameters:
            df (pd.DataFrame): DataFrame containing the 'Response Time (seconds)' and 'Model' columns.
        """
        stats = df.groupby('Model')['Response Time (seconds)'].describe()
        print("\nResponse Time Statistics:")
        print(stats[['mean', 'std', 'min', '50%', 'max']].round(3).to_string())

        try:
            with open('response_stats.tex', 'w') as f:
                f.write(
                    stats[['mean', 'std', 'min', '50%', 'max']]
                    .round(3)
                    .style.to_latex(hrules=True)
                )
        except Exception as e:
            logging.error(f"Error generating LaTeX response stats: {e}")

    def main() -> None:
        """
        Main execution function.

        Loads and processes data from JSON files, generates various comparative plots (response
            times, CPU, and memory usage),
        and outputs advanced performance statistics along with their LaTeX representations.
        """
        df = load_and_process_data()
        if df.empty:
            logging.error("No data available for plotting and analysis.")
            return

        plot_response_times(df)
        plot_cpu_memory_comparison(df)
        generate_advanced_statistics(df)
        generate_statistics(df)

    if __name__ == "__main__":
        main()
```

Listing 7.3: 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:

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os

# ================================================================================
# Data Aggregation and Visualization for Model Performance Metrics
# ================================================================================
# This script aggregates experimental results from JSON files, each containing
# performance metrics (e.g., BLEU, ROUGE, grammatical errors, readability, sentiment)
# for various AI models. The data are visualized using high-quality plots for scientific
# analysis, and descriptive statistics are exported in LaTeX format.

# ------------------------------
# Data Aggregation
# ------------------------------
directory = "./"
aggregated_data = []
for file in os.listdir(directory):
    # Process files that follow the naming convention "scored_<model>.json"
    if file.startswith("scored_") and file.endswith(".json"):
        model = file.replace("scored_", "").replace(".json", "")
        df_temp = pd.read_json(os.path.join(directory, file))
        df_temp["Model"] = model
        aggregated_data.append(df_temp)
df = pd.concat(aggregated_data, ignore_index=True)

# ------------------------------
# Global Plotting Style Settings
# ------------------------------
sns.set_theme(style="whitegrid", font_scale=0.9)
plt.rcParams['axes.titlepad'] = 15
plt.rcParams['axes.labelpad'] = 10

def rotate_labels(ax, rotation: int = 45, ha: str = 'right') -> None:
    """
    Rotate the x-axis labels for improved readability.

    Parameters:
        ax (matplotlib.axes.Axes): The axes on which to rotate the labels.
        rotation (int): Angle in degrees to rotate the labels.
        ha (str): Horizontal alignment of the labels.
    """
    ax.set_xticklabels(ax.get_xticklabels(), rotation=rotation, ha=ha, fontsize=9)
    plt.tight_layout()

# ================================================================================
# Visualization of Performance Metrics
# ================================================================================

# --- 1. BLEU and ROUGE Scores ---
plt.figure(figsize=(14, 7))
# Reshape data for plotting multiple text quality metrics
df_melt = df.melt(id_vars=["Model"], value_vars=["BLEU", "ROUGE-1", "ROUGE-2", "ROUGE-L"],
                  var_name="Metric", value_name="Score")
ax = sns.barplot(x="Model", y="Score", hue="Metric", data=df_melt, palette="viridis")
```

```
plt.title("Comparison of BLEU and ROUGE Scores", fontweight='bold')
plt.ylim(0, 0.05)
plt.legend(loc="upper right", frameon=True)
rotate_labels(ax)
plt.savefig("bleu_rouge.png", dpi=300, bbox_inches="tight")

# ##############################################################################
# Other plots are simular to the first one, but with different kinds of plots
# The plots are for:
# --- 2. Grammatical Errors ---
# --- 3. Readability (Flesch Score) ---
# --- 4. Sentiment Analysis ---
# --- 5. Combined Metrics Overview with Subplots ---
# ##############################################################################

# ==============================================================================
# Descriptive Statistics Export
# ==============================================================================
# Compute summary statistics for selected metrics by model
summary = df.groupby("Model")[["BLEU", "ROUGE-L", "Grammar Errors"]].agg(["mean", "std",
    "median", "min", "max"])
summary.to_latex("summary.tex", float_format="%.3f")
```

Listing 7.4: 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 like data alignment, merging, and reshaping, making it indispensable for data preprocessing and analysis tasks.

McKinney & the Pandas Development Team (2025)

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.

Hunter & the Matplotlib Development Team (2025)

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. GeeksforGeeks (2020)

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.

Oliphant & the NumPy Development Team (2025)

7.5.7. 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.

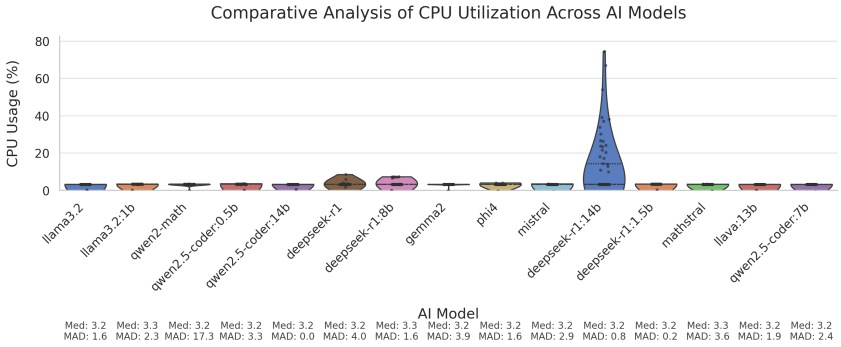

Figure 7.1.: Comparative Analysis of CPU Utilization Across AI Models

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.

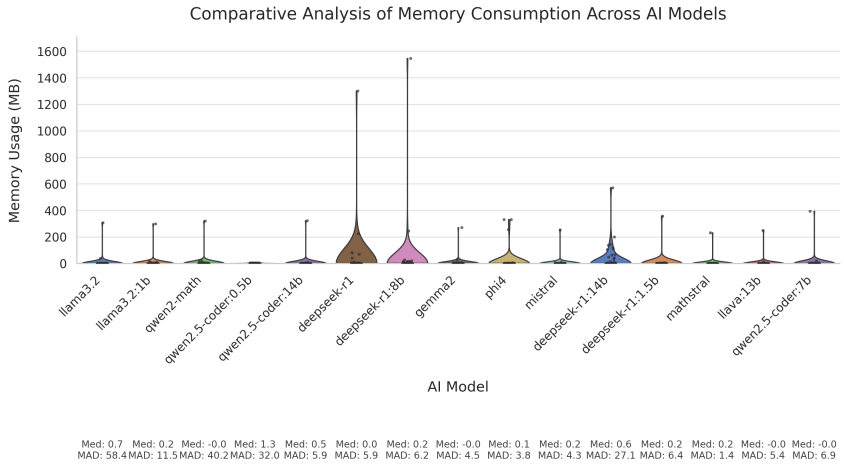

Figure 7.2.: Comparative Analysis of Memory Consumption Across AI Models

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.

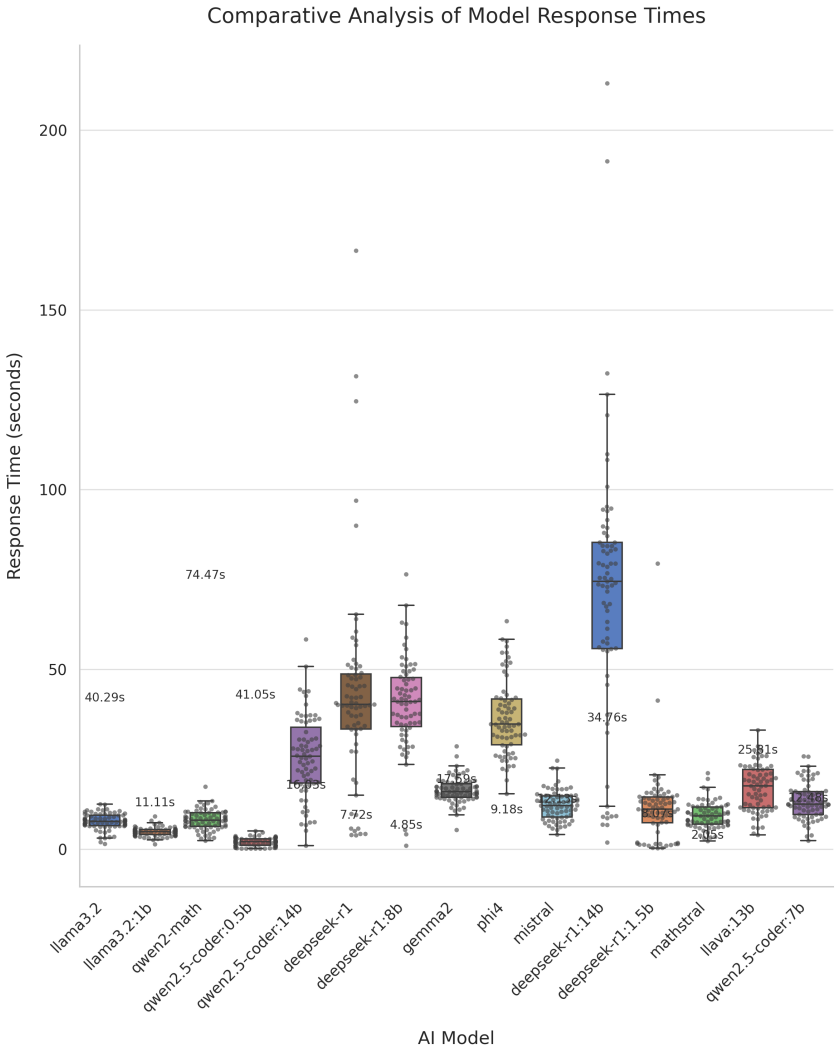

Figure 7.3.: Comparative Analysis of Model Response Times

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.

7.5.8. 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.

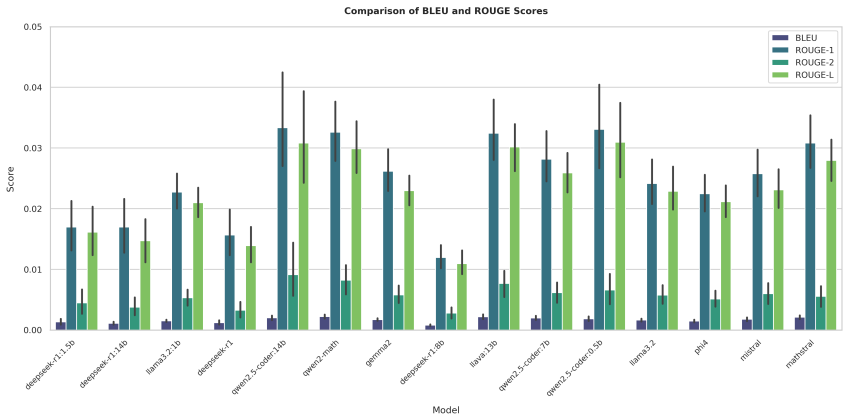

Figure 7.4.: Comparison of BLEU and ROUGE Scores

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]

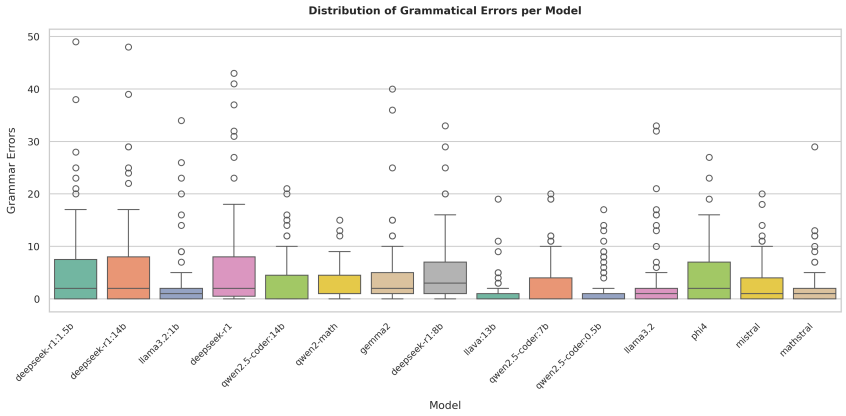

Figure 7.5.: Comparison of Grammatical Errors Across AI Models

Observations:
The number of grammatical errors varies considerably between models. The Math-

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

stral, 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.

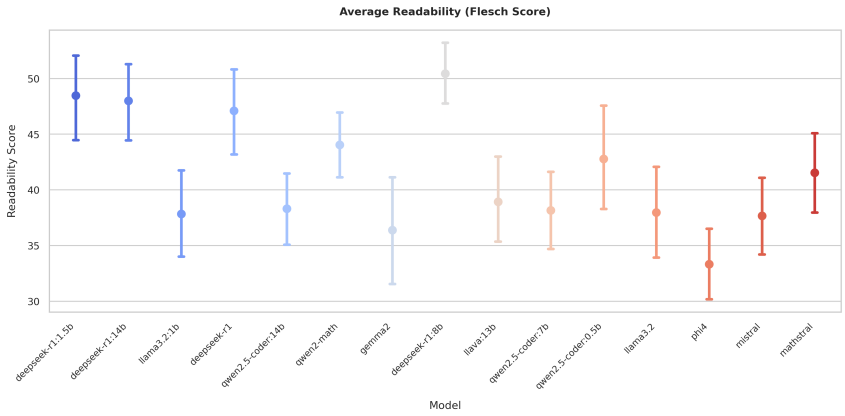

Figure 7.6.: Comparison of Readability Scores Across AI Models

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.

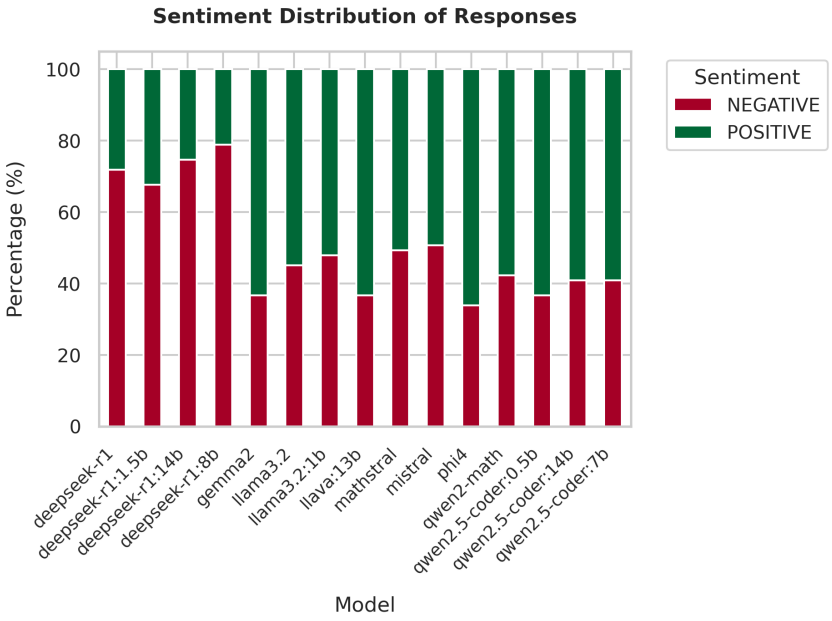

Figure 7.7.: Comparison of Sentiment Analysis Across AI Models

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.

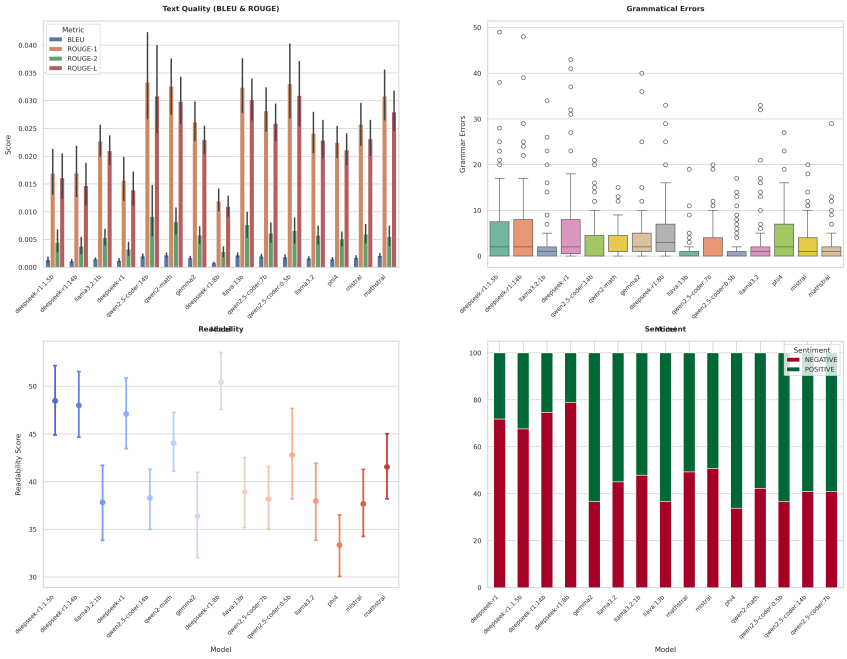

Figure 7.8.: Combined Metrics Overview for AI Models

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

7.5.9. 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.

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.

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.

7.5.10. 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.

7.5.11. 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*.

7.6. 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.

7.6.1. 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.

Overview - OpenAI API (n.d.*b*)

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.

Foy (2024)

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

The integration of OpenAI's API into our systems necessitates a comprehensive examination of data security and privacy, particularly with regard to compliance with both Austrian and European Union regulatory frameworks. In this context, our approach adheres to the stringent requirements set forth in Regulation (EU) 2016/679 (General Data Protection Regulation, GDPR) *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data* (2016), which forms the cornerstone of data protection within the EU. Additionally, the relevant provisions of the Austrian Data Protection Act (Datenschutzgesetz DSG) *Datenschutzgesetz (DSG)* (1999) have been carefully to ensure that personal data is handled in accordance with national standards.

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.

Introducing data residency in Europe | OpenAI (n.d.)

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.

Fortis (2024)

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 school environment.

In light of the evolving regulatory landscape, it is important 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.

7.6.3. 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 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
              Image</button>
13      </div>
14      <div v-if="generatedText" class="output-section">
15        <h2>Generated Text</h2>
16        <p>{{ generatedText }}</p>
17      </div>
18      <div v-if="generatedImage" class="output-section">
19        <h2>Generated Image</h2>
20        <img :src="generatedImage" alt="Image generated by
              OpenAI API" />
21      </div>
22    </div>
23  </template>
```

Listing 7.5: 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        try {
19          const response = await
                openai.chat.completions.create({
20            model: "gpt-3.5-turbo",
```

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

Listing 7.6: 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.
- **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.

7.7. 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.

8. Self-hosted Flask Service

Author: Luna Schätzle

Author: Florian Prandstetter (docker section 8.7)

This chapter details the implementation, architecture, and deployment of a self-hosted Flask service that serves as a critical interface linking the front-end and back-end components of the system. It outlines the technical design and deployment strategies while critically evaluating the benefits of a self-hosted solution and the rationale behind key architectural decisions. The service, which underpins both the Student AI Hub and the code extensions backend, was developed to address specific operational requirements. This section provides a comprehensive overview of the motivations for its development, elaborates on its core functionalities, and situates its role within the broader system architecture—thus laying the groundwork for the ensuing technical discussion.

8.1. 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.

8.2. Flask as a Web Framework

Flask is a lightweight, yet versatile web Python framework that plays a central role in the development of our Student AI Hub. Its minimalistic design and powerful capabilities make it an excellent choice for both rapid prototyping and scalable application development.

8.2.1. Core Functionalities of Flask

Flask's architecture is built around several core functionalities that streamline web application development:

- **Request Routing:** Flask's routing system allows for the efficient mapping of URLs to Python functions. This ensures that incoming web requests are correctly handled and directed to the appropriate endpoints, which is crucial for managing user interactions.
- **Middleware Support:** Flask supports middleware, facilitating the insertion of additional processing layers before or after a request is handled. This capability is essential for tasks such as authentication, logging, and error handling.
- **Extensibility:** Its modular design allows developers to integrate third-party extensions or develop custom modules, thereby expanding Flask's functionalities to meet the evolving demands of the project.
- **Simplicity and Clarity:** Flask's clear and concise API encourages clean, maintainable code. This simplicity does not come at the expense of power; rather, it provides a robust foundation upon which complex systems can be built.

These features collectively contribute to an efficient workflow, ensuring that web requests and responses are managed in a structured and scalable manner.

8.2.2. Rationale for Selecting Flask

The decision to adopt Flask as the backbone of our web service was influenced by several key considerations:

- **Simplicity and Flexibility:** Flask's minimalistic core allows for rapid development and iterative prototyping without the overhead of a full-stack framework. This flexibility is invaluable during the early stages of project development and testing.
- **Extensive Documentation and Community Support:** With comprehensive documentation and a vibrant community, Flask offers abundant resources for troubleshooting and extending functionality. This support network accelerates development and helps resolve challenges efficiently.
- **Scalability:** Despite its simplicity, Flask is designed to scale. Its modular nature means that as the project grows, new features can be seamlessly integrated, ensuring that the framework remains robust even under increased demand.
- **Integration Capabilities:** Flask can be easily combined with various libraries and APIs. This makes it particularly well-suited for integrating AI models, data processing tools, and other external services—core requirements of the Student AI Hub.

In summary, Flask's combination of ease-of-use, extensibility, and strong community support provides a solid and adaptable framework for developing a dynamic web service. Its ability to handle everything from simple routing to complex middleware interactions makes it the ideal choice for building a robust backend that meets the diverse needs of our project.

8.3. 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.

8.3.1. System Architecture

The figure below illustrates the key architectural components of the Flask service and their interactions.

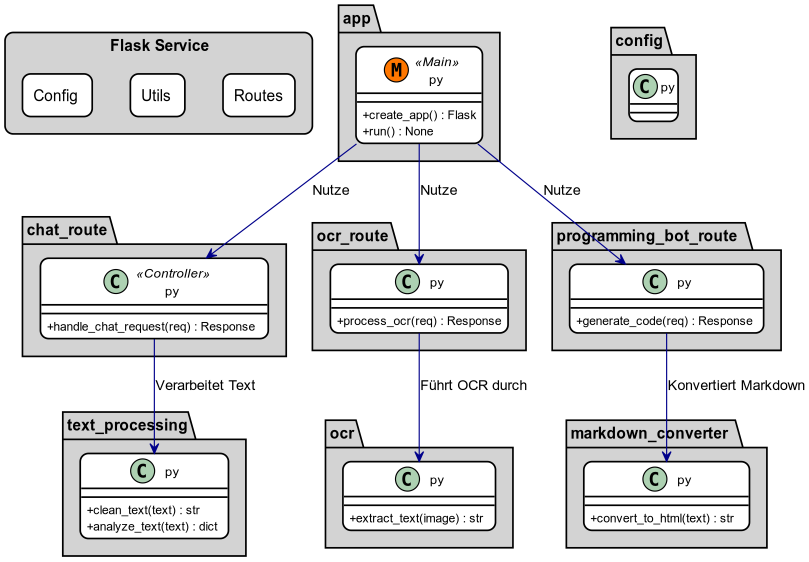

Figure 8.1.: Flask Service Architecture

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 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.
- **OCR_route.py:** Manages endpoints for image to Text conversion.
- **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.

8.3.2. 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.

8.4. RESTful Endpoints and Functionalities

This section provides a comprehensive overview of the RESTful endpoints implemented in the Flask service. Each endpoint is detailed in terms of its input parameters, expected output, and the underlying logic that processes incoming requests.

8.4.1. Chatbot Endpoints

The Chatbot endpoints are responsible for handling all server requests associated with both the general chatbot and the image recognition chatbot.

Chatbot Endpoint The following code snippet illustrates the endpoint for the general chatbot:

```python
@chat_bp.route('/ask_ollama', methods=['POST'])  # Defines the endpoint
def ask_ollama_endpoint():
    """
    Chat endpoint: Receives a user query and a model, communicates with Ollama, and returns the response.
    """
    data = request.get_json()

    if not data:
        logger.warning('No JSON data provided')
        return jsonify({'error': 'No JSON data provided'}), 400

    prompt = data.get('prompt')
    model = data.get('model', Config.OLLAMA_MODEL_DEFAULT)  # Default model

    if not prompt:
        logger.warning('Prompt missing')
        return jsonify({'error': 'Prompt missing'}), 400

    try:
        response = ask_ollama(prompt, model=model)

        if 'choices' in response:
            return jsonify(response), 200
        else:
            return jsonify(response), 500

    except Exception as e:
        logger.error(f'Error communicating with Ollama: {str(e)}')
        return jsonify({'error': 'Error communicating with Ollama'}), 500
```

Listing 8.1: Chatbot Endpoint

This endpoint extracts the necessary data from the API request and passes it to the `ask_ollama` utility function, which communicates with the Ollama API and returns the corresponding response. If any required data is missing or if the Ollama API is unreachable, the endpoint returns an appropriate error message.

Image Recognition Endpoint The following code snippet demonstrates the endpoint for the Image Recognition Chatbot:

```
@chat_bp.route('/ask_ollama_vision', methods=['POST'])  # Defines the endpoint using POST method
def ask_ollama_vision_endpoint():
    """
    Vision Chat endpoint: Receives a user query with images and a model, communicates with Ollama Vision, and returns the response.
    """
    data = request.get_json()

    if not data:
        logger.warning('No JSON data provided')
        return jsonify({'error': 'No JSON data provided'}), 400

    prompt = data.get('prompt')
    model = data.get('model', 'llama3.2-vision')  # Default model for vision tasks
    images = data.get('images', [])

    if not prompt and not images:
        logger.warning('Neither prompt nor images provided')
        return jsonify({'error': 'Neither prompt nor images provided'}), 400

    # Process the images
    image_paths = []
    try:
        for idx, img_str in enumerate(images):
            # Extract image data from base64 string
            if ',' in img_str:
                header, encoded = img_str.split(',', 1)
            else:
                header, encoded = '', img_str
            img_data = base64.b64decode(encoded)
            img = Image.open(BytesIO(img_data))

            # Validate the image format
            if img.format.lower() not in Config.ALLOWED_EXTENSIONS:
```

```
34                 logger.warning(f'Invalid image format:
                       {img.format}')
35                 return jsonify({'error': f'Invalid image
                       format: {img.format}'}), 400
36
37             # Optionally reduce or compress image size
38             img.thumbnail((1024, 1024))  # Example:
                   maximum size 1024x1024
39
40             # Save the image temporarily
41             img_filename =
                   f"temp_{idx}.{img.format.lower()}"
42             img_path = os.path.join(Config.UPLOAD_FOLDER,
                   img_filename)
43             img.save(img_path)
44             image_paths.append(img_path)
45
46     except Exception as e:
47         logger.error(f'Error processing images: {str(e)}')
48         return jsonify({'error': f'Error processing
               images: {str(e)}'}), 400
49
50     try:
51         response = ask_ollama_vision(prompt, model=model,
               image_paths=image_paths)
52
53         if 'choices' in response:
54             return jsonify(response), 200
55         else:
56             return jsonify(response), 500
57
58     except Exception as e:
59         logger.error(f'Error communicating with Ollama
               Vision: {str(e)}')
60         return jsonify({'error': 'Error communicating with
               Ollama Vision'}), 500
61
62     finally:
63         # Delete temporary images
64         for img_path in image_paths:
65             if os.path.exists(img_path):
66                 os.remove(img_path)
```

Listing 8.2: Image Recognition Endpoint

The Image Recognition endpoint functions similarly to the general chatbot endpoint, with the added capability of processing image data. It decodes the base64-encoded images, saves them temporarily, validates their format, and resizes them if necessary to optimize processing speed and reduce server load. Once processed, the images are passed to the `ask_ollama_vision` utility function. After the request is handled, the endpoint ensures that all temporary images are deleted.

Overall, these endpoints are designed for scalability and robustness, efficiently managing both text and image inputs while providing meaningful feedback to the user in case of errors.

8.4.2. OCR Endpoints

This file contains all the OCR endpoints. Currently, there is a single endpoint that handles input for optical character recognition.

The following code snippet illustrates the OCR endpoint:

```
1  def allowed_file(filename):
2      """
3      Checks if the file has an allowed extension.
4      """
5      return '.' in filename and filename.rsplit('.',
           1)[1].lower() in Config.ALLOWED_EXTENSIONS
6
7  @ocr_bp.route('/ocr', methods=['POST'])
8  def process_ocr():
9      """
10     OCR endpoint: Accepts an image (either as a Base64
           string or as a file), extracts text using OCR,
11     enhances the extracted text via the Ollama API, and
           returns both the raw and improved text.
12     Supports both JSON and multipart/form-data requests.
13     """
14     img_path = None  # Initialize img_path for cleanup
15     try:
16         # Check if the request contains JSON data
17         if request.is_json:
18             data = request.get_json()
19             image_str = data.get('image')
20             if not image_str:
```

```
21                  logger.warning('Image data missing in JSON
                        request')
22                  return jsonify({'error': 'Image data
                        missing'}), 400
23
24              # Decode the Base64 image
25              if ',' in image_str:
26                  header, encoded = image_str.split(',', 1)
27              else:
28                  header, encoded = '', image_str
29              try:
30                  img_data = base64.b64decode(encoded)
31                  img = Image.open(BytesIO(img_data))
32              except Exception as e:
33                  logger.error(f'Error decoding Base64
                        image: {str(e)}')
34                  return jsonify({'error': 'Invalid Base64
                        image data'}), 400
35
36          else:
37              # Handle multipart/form-data request
38              if 'image' not in request.files:
39                  logger.warning('No image file provided in
                        multipart/form-data request')
40                  return jsonify({'error': 'No image file
                        provided'}), 400
41
42              file = request.files['image']
43              if file.filename == '':
44                  logger.warning('Empty filename in
                        multipart/form-data request')
45                  return jsonify({'error': 'Empty
                        filename'}), 400
46
47              if file and allowed_file(file.filename):
48                  try:
49                      img = Image.open(file.stream)
50                  except Exception as e:
51                      logger.error(f'Error opening uploaded
                            image file: {str(e)}')
52                      return jsonify({'error': 'Invalid
                            image file'}), 400
53              else:
54                  logger.warning(f'Invalid image format:
```

```
                {file.filename}')
55              return jsonify({'error': f'Invalid image
                    format. Allowed: {",
                    ".join(Config.ALLOWED_EXTENSIONS)}'}),
                    400
56
57          # Validate the image format
58          if img.format.lower() not in
                Config.ALLOWED_EXTENSIONS:
59              logger.warning(f'Invalid image format:
                    {img.format}')
60              return jsonify({'error': f'Invalid image
                    format: {img.format}'}), 400
61
62          # Optionally resize the image to reduce processing
                time
63          img.thumbnail((1024, 1024))  # Example: maximum
                size 1024x1024
64
65          # Save the image temporarily if required by the
                OCR tool
66          img_filename = "temp_ocr.png"
67          img_path = os.path.join(Config.UPLOAD_FOLDER,
                img_filename)
68          try:
69              img.save(img_path)
70          except Exception as e:
71              logger.error(f'Error saving temporary image:
                    {str(e)}')
72              return jsonify({'error': 'Error saving
                    image'}), 500
73
74          # Perform OCR using Tesseract
75          try:
76              extracted_text =
                    pytesseract.image_to_string(img,
                    lang=Config.TESSERACT_LANG)
77              if not extracted_text.strip():
78                  logger.warning('OCR could not extract any
                        text')
79                  return jsonify({'error': 'OCR could not
                        extract any text'}), 400
80          except Exception as e:
81              logger.error(f'OCR processing error: {str(e)}')
```

```
82              return jsonify({'error': 'OCR processing
                    error'}), 500
83
84          # Enhance the extracted text using the Ollama API
85          try:
86              improved_text =
                    improve_text_with_ollama(extracted_text,
                    model='llama3.2')
87          except Exception as e:
88              logger.error(f'Error improving text with
                    Ollama: {str(e)}')
89              return jsonify({'error': 'Error improving
                    text'}), 500
90
91          return jsonify({
92              'raw_text': extracted_text,
93              'improved_text': improved_text
94          }), 200
95
96      except Exception as e:
97          logger.error(f'General error in OCR endpoint:
                {str(e)}')
98          return jsonify({'error': f'General error:
                {str(e)}'}), 500
99
100     finally:
101         # Clean up the temporary image
102         if img_path and os.path.exists(img_path):
103             try:
104                 os.remove(img_path)
105             except Exception as e:
106                 logger.error(f'Error removing temporary
                        image: {str(e)}')
```

Listing 8.3: OCR Endpoint

This endpoint accepts image data in both JSON and multipart/form-data formats. It decodes the image, validates its format, and processes it using Optical Character Recognition (OCR). The raw text extracted from the image is then enhanced through the Ollama API, and both the original and improved texts are returned as part of the response. Additionally, the endpoint manages errors effectively and ensures that temporary image files are cleaned up after processing.

The following utility functions play a crucial role in the OCR process:

- `allowed_file`: Verifies whether the uploaded file has an allowed extension.
- `improve_text_with_ollama`: Sends the extracted text to the Ollama API for enhancement.
- `pytesseract.image_to_string`: Extracts text from images using the Tesseract OCR engine.

These functions are essential for processing image data, interfacing with external APIs, and ensuring the accuracy and reliability of the OCR workflow.

8.4.3. Programming Bot Endpoints

The Programming Bot endpoints manage all requests related to the Programming Bot functionality within the Student AI Hub and the Code Extension. These endpoints route user queries to the Ollama API and return code-based responses, tailored specifically for programming inquiries.

The following code snippet illustrates the endpoint for the Programming Bot:

```
1  @chat_bp.route('/ask_programming_bot', methods=['POST'])
2  def ask_programming_bot_endpoint():
3      """
4      Programming Bot Endpoint: Receives a user query,
           communicates with Ollama, and returns the response.
5      """
6      data = request.get_json()
7
8      if not data:
9          logger.warning('No JSON data provided')
10         return jsonify({'error': 'No JSON data
               provided'}), 400
11
12     prompt = data.get('prompt')
13     model = data.get('model', 'qwen2.5-coder:0.5b')
14
15     if not prompt:
16         logger.warning('Prompt missing')
17         return jsonify({'error': 'Prompt missing'}), 400
18
19     try:
20         # Define a specialized system message for the
               programming bot
21         messages = [
```

```
22              {
23                  'role': 'system',
24                  'content': (
25                      "You are an experienced programmer and technical advisor named Luminara. "
26                      "Answer programming questions precisely and provide clear code examples in the requested programming language."
27                  )
28              },
29              {
30                  'role': 'user',
31                  'content': prompt
32              }
33          ]
34
35          # Send the request to Ollama
36          response = chat(
37              model=model,
38              messages=messages,
39              stream=False
40          )
41
42          # Extract and clean the response
43          bot_response = response.message.content.strip()
44
45          if not bot_response:
46              logger.error('Ollama did not return a response.')
47              raise Exception('Ollama could not generate a response.')
48
49          return {'choices': [{'text': bot_response}]}
50
51      except ResponseError as e:
52          logger.error(f'Ollama ResponseError: {e.error}')
53          return jsonify({'error': f'Ollama API error: {e.error}'}), 500
54      except Exception as e:
55          logger.error(f'Ollama error: {str(e)}')
56          return jsonify({'error': str(e)}), 500
```

Listing 8.4: Programming Bot Endpoint

This endpoint follows a structure similar to other endpoints but includes modifications to the prompt to enhance the quality of the AI's response. A system message informs the AI that it is interacting with an experienced programmer and technical advisor named Luminara, which guides it to provide more accurate and context-specific answers. The endpoint handles the connection to the Ollama API, processes the response, and ensures that errors are logged and appropriately returned to the user.

8.5. Utility Functions

The utility functions presented in this section are fundamental to the operation of the Flask service. They execute critical tasks—such as interfacing with external APIs, processing data, and managing server resources—in a modular, reusable, and efficient manner. This design ensures that the service operates reliably and seamlessly.

8.5.1. Text Processing Utilities

The text processing utilities are used to enhance the quality and readability of text extracted from images or user queries. They leverage the Ollama API to refine the text, correct errors, and improve coherence, thereby enhancing the overall user experience.

ask_ollama The `ask_ollama` function facilitates communication with the Ollama API to generate responses for user queries. It accepts a prompt along with an optional model parameter, dispatches the request to the Ollama API, and returns the resulting response.

The code snippet below demonstrates the implementation of the `ask_ollama` utility function:

```
1 def ask_ollama(prompt, model='llama3.2:1b'):
2     """
3     Sends the user prompt to the selected Ollama model and
          returns the response.
4     """
5     try:
```

```
6           # Define the message structure
7           messages = [
8               {
9                   'role': 'system',
10                  'content': 'You are a helpful assistant.
                        Please answer the following user query:'
11              },
12              {
13                  'role': 'user',
14                  'content': prompt
15              }
16          ]
17
18          # Send the request to Ollama
19          response = chat(
20              model=model,
21              messages=messages,
22              stream=False
23          )
24
25          # Extract the response content
26          bot_response = response.message.content.strip()
27
28          if not bot_response:
29              logger.error('Ollama did not return any
                    response.')
30              raise Exception('Ollama was unable to generate
                    a response.')
31
32          return {'choices': [{'text': bot_response}]}
33
34      except ResponseError as e:
35          logger.error(f'Ollama ResponseError: {e.error}')
36          return {'error': f'Ollama API error: {e.error}'}
37      except Exception as e:
38          logger.error(f'Ollama error: {str(e)}')
39          return {'error': str(e)}
```

Listing 8.5: ask_ollama Utility Function

This function encapsulates the logic necessary for transmitting user prompts to the Ollama API and processing the responses. It is engineered to handle diverse user queries, thereby generating accurate and contextually relevant answers. Consequently, it serves as a critical component underpinning the chatbot functionality

of the Flask service.

ask_ollama_vision The `ask_ollama_vision` function augments the capabilities of the standard `ask_ollama` utility by incorporating support for image-based queries. It processes image data, integrates it with textual prompts, and dispatches the combined input to the Ollama Vision API for response generation. This extension enables the Flask service to handle multimodal inputs, thereby broadening its applicability and enhancing user interaction.

The following code snippet demonstrates the implementation of the `ask_ollama_vision` function:

```
1  def ask_ollama_vision(prompt, model='llama3.2-vision',
       image_paths=[]):
2      """
3      Dispatches the user's prompt along with associated
           images to the selected Ollama Vision model,
4      and returns the generated response.
5      """
6      try:
7          # Define the message structure with associated
               image paths
8          messages = [
9              {
10                 'role': 'user',
11                 'content': prompt,
12                 'images': image_paths  # Attach image
                       paths to the user message
13             }
14         ]
15
16         # Send the request to Ollama Vision
17         response = chat(
18             model=model,
19             messages=messages,
20             stream=False
21         )
22
23         # Extract the response content
24         bot_response = response.message.content.strip()
25
26         if not bot_response:
```

```
27              logger.error('Ollama did not return any
                    response.')
28              raise Exception('Ollama was unable to generate
                    a response.')
29
30          return {'choices': [{'text': bot_response}]}
31
32      except ResponseError as e:
33          logger.error(f'Ollama ResponseError: {e.error}')
34          return {'error': f'Ollama API error: {e.error}'}
35      except Exception as e:
36          logger.error(f'Ollama error: {str(e)}')
37          return {'error': str(e)}
```

Listing 8.6: ask_ollama_vision Utility Function

This function is instrumental in enabling the Flask service to process and interpret image data in conjunction with textual prompts. By integrating image recognition capabilities, it enhances the service's versatility and responsiveness, thereby significantly enriching the overall user experience.

improve_text_with_ollama The `improve_text_with_ollama` function is designed to refine text extracted through the OCR process by leveraging the Ollama API. It improves readability, grammar, and coherence, converting raw text into a well-structured Markdown format that is both clear and professionally formatted.

The code snippet below illustrates the implementation of the `improve_text_with_ollama` function:

```
1  def improve_text_with_ollama(text, model='llama3.2'):
2      """
3      Enhances the provided text using the Ollama API and
           returns the refined Markdown text.
4      """
5      try:
6          # Define the message structure with system and
               user roles to improve the text
7          messages = [
8              {
9                  'role': 'system',
10                 'content': (
11                     "You are a highly proficient text
                           processor and Markdown expert. "
```

```
12                      "Your task is to enhance the following
                            text, which has been extracted via
                            OCR from an image, "
13                      "and convert it into a well-structured
                            Markdown format. "
14                      "Ensure correct spelling, grammar, and
                            syntax, and format the text using
                            appropriate Markdown elements "
15                      "such as headings, lists, code blocks,
                            and emphasis. "
16                      "Provide only the enhanced Markdown
                            text without any additional
                            commentary or explanations."
17                  )
18              },
19              {
20                  'role': 'user',
21                  'content': (
22                      "Here is the text to be processed:\n\n"
23                      f"```\n{text}\n```"
24                  )
25              }
26          ]
27
28          # Send the request to Ollama
29          response = chat(
30              model=model,
31              messages=messages,
32              stream=False
33          )
34
35          # Retrieve the improved text from the response
36          markdown_text = response.message.content.strip()
37
38          if not markdown_text:
39              logger.error('Ollama did not return any text.')
40              raise Exception('Ollama was unable to improve
                    the text.')
41
42          return markdown_text
43
44      except ResponseError as e:
45          logger.error(f'Ollama ResponseError: {e.error}')
46          raise e
```

```
47      except Exception as e:
48          logger.error(f'Ollama error: {str(e)}')
49          raise e
```

Listing 8.7: improve_text_with_ollama Utility Function

This utility function plays a pivotal role in enhancing the quality and coherence of OCR-extracted content. By refining the raw text into a structured Markdown format, it ensures that the final output is both accurate and user-friendly, thereby improving the overall efficacy of the OCR workflow.

8.5.2. OCR (Optical Character Recognition) Utilities

This section details the OCR utilities integrated within the Flask service, which are responsible for processing images and extracting textual content. Currently, the primary function implemented is `perform_ocr`, which utilizes the `pytesseract` library to perform OCR operations.

perform_ocr The `perform_ocr` function harnesses the Tesseract OCR engine to extract text from an image file. It accepts an image file path and an optional language parameter, processes the image, and returns the extracted text. The function is designed to handle a variety of image formats and ensures accurate text retrieval, thus streamlining the OCR workflow within the Flask service.

The code snippet below demonstrates the implementation of the `perform_ocr` function:

```
1  import pytesseract
2  from PIL import Image
3
4  def perform_ocr(image_path, lang='deu'):
5      """
6      Performs OCR on the specified image file and returns
           the extracted text.
7      """
8      try:
9          # Load the image and convert it to grayscale
10         image = Image.open(image_path).convert('L')  #
               Convert to grayscale
11
```

```
12          # Optional: Apply image preprocessing techniques
                (e.g., denoising, thresholding, etc.)
13          custom_config = r'--oem 3 --psm 6'
14          text = pytesseract.image_to_string(image,
                lang=lang, config=custom_config)
15          return text
16      except Exception as e:
17          logger.error(f'OCR error: {str(e)}')
18          raise e
```

Listing 8.8: perform_ocr Utility Function

This function encapsulates the essential logic for image processing and text extraction using the Tesseract OCR engine. By leveraging robust libraries, it ensures that the Flask service can efficiently convert image-based data into readable text.

The key libraries utilized are:

pytesseract: A Python wrapper for Google's Tesseract-OCR Engine, used for extracting text from images via optical character recognition (OCR).

PIL (Python Imaging Library): An image processing library that enables the opening, manipulation, and saving of various image file formats, thereby enhancing Python's capabilities in handling visual data.

8.6. Deployment

The Flask service was deployed across multiple platforms to support comprehensive testing, evaluation, and integration with other system components. The deployment process involved configuring the server environment, establishing required dependencies, and ensuring seamless operation across various hosting infrastructures.

For development and testing, the Flask service is deployed locally on a laptop or desktop machine. This setup enables rapid iteration, feature testing, and effective troubleshooting in a controlled environment. Local deployment requires installing Python, setting up a virtual environment, and running the Flask server on the local system.

In production environments, the Flask service is hosted on dedicated servers or cloud platforms to ensure high availability, scalability, and reliability. This approach

enables the service to handle a high volume of requests and users efficiently. Production deployment typically involves meticulous server configuration, dependency management, and the implementation of robust security measures to mitigate potential threats.

To further streamline production deployment, a Docker container was developed. More information on the Docker-based deployment approach is provided in Section 8.7.

8.7. 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 do not 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.

8.7.1. 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. *Docker*$_{F}$*lask* (n.d.)

- **Flask Application** 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.

8.7.2. 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 applications. *Doocker*$_{C}$*ompose* (n.d.)

8.8. 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.

8.9. Conclusion and Future Work

In summary, this chapter has detailed the architecture, functionality, and deployment strategy of the self-hosted Flask service. The design emphasizes modularity and expandability, allowing for rapid prototyping, tailored configurations, and seamless integration with external APIs. The clear separation of concerns across endpoints and utility functions ensures both maintainability and flexibility, which are essential for evolving project requirements.

However, practical limitations regarding scalability and performance on modest hardware were also noted. Addressing these challenges through distributed architectures and enhanced concurrency mechanisms will be crucial for future deployments. As a result, future work should focus on optimizing resource management, scaling the service to meet higher demand, and further refining API integrations for improved robustness and security.

The next two chapters will delve into the front-end components of the Student AI Hub and the code extension, providing a comprehensive overview of their design, functionalities, and integration with the Flask service.

9. 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.

9.1. 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.

9.2. 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.

9.2.1. 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 HTML, JavaScript and CSS, please refer to Chapter 6, subsection "Vue.js."

9.2.2. 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 8.

9.2.3. 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 7, subsection "Integration of OpenAI's API."

9.2.4. 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.

Firebase Features Explained in Detail (2025 Update) (n.d.)

9.3. 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.
- **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.

9.4. 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.

9.4.1. 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.

9.4.2. 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.

9.4.3. 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, 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.

9.4.4. 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:

```
1  import firebase from 'firebase/app';
2  import 'firebase/auth';
3
4  // Firebase configuration object containing keys and
       identifiers
5  const firebaseConfig = {
6    apiKey: "YOUR_API_KEY",
7    authDomain: "YOUR_PROJECT_ID.firebaseapp.com",
8    databaseURL: "https://YOUR_PROJECT_ID.firebaseio.com",
9    projectId: "YOUR_PROJECT_ID",
10   storageBucket: "YOUR_PROJECT_ID.appspot.com",
11   messagingSenderId: "YOUR_SENDER_ID",
12   appId: "YOUR_APP_ID"
13 };
14
15 // Initialize Firebase with the configuration
16 firebase.initializeApp(firebaseConfig);
17
18 // Export the authentication module for use in Vue
       components
19 export const auth = firebase.auth();
20
```

```
// Monitor authentication state changes
auth.onAuthStateChanged(user => {
  if (user) {
    // User is signed in; update application state accordingly
    console.log('User signed in:', user);
  } else {
    // User is signed out; update the UI to reflect sign-out state
    console.log('No user is signed in.');
  }
});
```

Listing 9.1: 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:

```
<template>
    <div>
        <!-- Display a welcome message if the user is signed in -->
        <h2 v-if="user">Welcome, {{ user.email }}</h2>
        <!-- Otherwise, show the sign-in form -->
        <div v-else>
            <input v-model="email" placeholder="Email" />
            <input v-model="password" type="password" placeholder="Password" />
            <button @click="signIn">Sign In</button>
            <button @click="signInWithGoogle">Sign In with Google</button>
            <p v-if="errorMessage" class="error">{{ errorMessage }}</p>
```

```
12          </div>
13      </div>
14  </template>
15
16  <script>
17  import { auth } from '@/firebase'; // Adjust the path
        according to your project structure
18  import firebase from 'firebase/app';
19  import 'firebase/auth';
20
21  export default {
22      data() {
23          return {
24              email: '',
25              password: '',
26              user: null,
27              errorMessage: ''
28          };
29      },
30      created() {
31          // Listen for authentication state changes and
                update the component state
32          auth.onAuthStateChanged(user => {
33              this.user = user;
34          });
35      },
36      methods: {
37          signIn() {
38              // Attempt to sign in using the provided email
                    and password
39              auth.signInWithEmailAndPassword(this.email,
                    this.password)
40                  .then(credential => {
41                      this.user = credential.user;
42                      this.errorMessage = '';
43                  })
44                  .catch(error => {
45                      // Handle authentication errors by
                            updating the errorMessage state
46                      this.errorMessage = error.message;
47                      console.error("Authentication error:",
                            error);
48                  });
49          },
```

```
50          signInWithGoogle() {
51              const provider = new
                    firebase.auth.GoogleAuthProvider();
52              auth.signInWithPopup(provider)
53                  .then(result => {
54                      this.user = result.user;
55                      this.errorMessage = '';
56                  })
57                  .catch(error => {
58                      this.errorMessage = error.message;
59                      console.error("Google sign-in error:",
                            error);
60                  });
61          }
62      },
63  };
64  </script>
65
66  <style scoped>
67  .error {
68      color: red;
69      font-size: 0.9em;
70  }
71  </style>
```

Listing 9.2: 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 `error messages`) are used to manage form inputs, the authenticated user state, and error messages.
- **Sign-In Method:** The `Sign-in` 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.

9.4.5. 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.

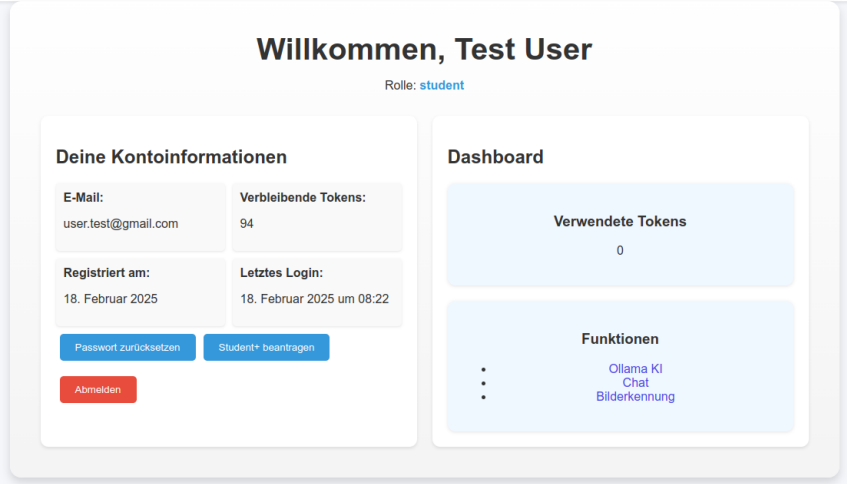

Figure 9.1.: User Account Management and Overview

9.4.6. 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.

9.4.7. 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.

9.5. 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:

```
<template>
  <div>
    <!-- AI Model Selection -->
    <select v-model="selectedModel">
      <option value="llama3.2:1b">LLaMA 3.2 - 1B (Fast)</option>
      <option value="llama3.2">LLaMA 3.2 - 2B (Latest)</option>
      <!-- more Models -->
    </select>
    <!-- Chat Display -->
    <div v-for="msg in currentChat.messages" :key="msg.id">
      <p v-if="msg.type==='user'">{{ msg.text }}</p>
      <p v-else>{{ msg.text }}</p>
    </div>
    <!-- User Input and Submission -->
    <input v-model="userInput"
       @keydown.enter="sendMessage" placeholder="Ask the AI question..." />
    <button @click="sendMessage">Send</button>
  </div>
</template>

<script>
import axios from 'axios';
export default {
  data() {
    return {
      userInput: '',
      selectedModel: 'llama3.2:1b',
      currentChat: { messages: [] },
    };
  },
  methods: {
    async sendMessage() {
      if (!this.userInput.trim()) return;
      // Append user message to chat
```

```
34          this.currentChat.messages.push({ id: Date.now(),
                type: 'user', text: this.userInput });
35          // Send the query to the Flask API
36          const response = await
                axios.post('http://server-address/ask_ollama', {
37            prompt: this.userInput,
38            model: this.selectedModel,
39          });
40          // Append AI response to chat
41          this.currentChat.messages.push({ id: Date.now(),
                type: 'ollama', text:
                response.data.choices[0].text });
42          this.userInput = '';
43        },
44      },
45    };
46    </script>
```

Listing 9.3: 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.

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

9.6.1. 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.

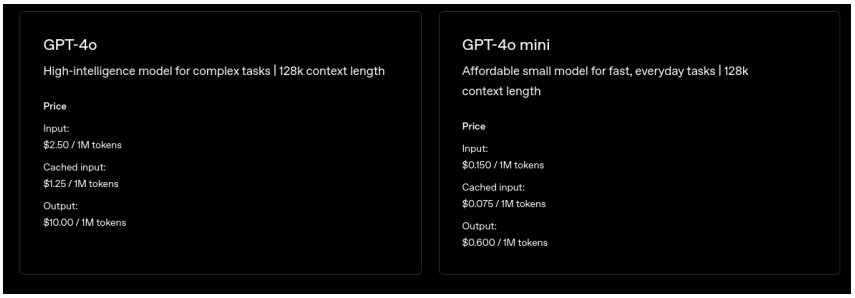

Figure 9.2.: ChatGP API Pricing

Pricing | OpenAI (n.d.)

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 9.2). [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.

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

9.6.2. 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.

9.6.3. Integration of OpenAI's API

For a comprehensive guide on integrating OpenAI's API into a Vue.js project, please refer to Chapter 7, Section 7.6.3.

9.7. 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.

Programming Bot Features

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

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

- **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 8.

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 *Introduction to Markdown — Write the Docs* (n.d.).

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     <div class="selection-area">
5       <select v-model="selectedModel">
6         <option value="modelA">Model A</option>
7         <option value="modelB">Model B</option>
```

```
8         </select>
9         <select v-model="selectedLanguage">
10          <option value="python">Python</option>
11          <option value="java">Java</option>
12        </select>
13      </div>
14      <!-- Chat Interface -->
15      <div class="chat-box">
16        <div v-for="msg in messages" :class="msg.type">{{
              msg.text }}</div>
17      </div>
18      <!-- Input Area with Refine Option -->
19      <textarea v-model="userInput" placeholder="Enter your
            programming question..."></textarea>
20      <button @click="sendMessage">Send</button>
21      <button v-if="isLastUserMessage"
            @click="prepareRefine">Refine</button>
22    </div>
23  </template>
24
25  <script>
26  export default {
27    data() {
28      return {
29        userInput: '',
30        selectedModel: 'modelA',
31        selectedLanguage: 'python',
32        messages: [],
33      };
34    },
35    methods: {
36      async sendMessage() {
37        // Append the selected programming language to the
              user prompt
38        const prompt = `Language:
              ${this.selectedLanguage}\n${this.userInput}`;
39        // Send the prompt to the Flask API and process the
              response...
40      },
41      prepareRefine() {
42        // Open a modal to refine the user prompt for
              improved accuracy
43      }
44    }
```

```
};
</script>
```

Listing 9.4: 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.

9.8. 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 8. 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.

9.8.1. 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.

```
<template>
  <div class="image-recognition">
    <h1>Upload Image and Send to Ollama</h1>
    <!-- Text prompt input -->
    <input v-model="userPrompt" placeholder="Enter a
       prompt..." @keydown.enter="sendRequest" />
    <!-- File input for image upload -->
    <input type="file" @change="handleImageUpload" />
    <!-- Submission button -->
    <button @click="sendRequest">Submit</button>
    <!-- Status and response display -->
    <div v-if="loading">Sending request...</div>
    <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
                URL
37          this.imageData = reader.result.split(",")[1];
38        };
39        if (file) reader.readAsDataURL(file);
40      },
41      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
              (request details omitted)
48        // e.g., using axios.post(url, { prompt:
              this.userPrompt, image: this.imageData })
49        // Process the response and update this.response
              accordingly.
50        this.loading = false;
51      }
52    }
```

```
};
</script>
```

Listing 9.5: 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.

9.9. 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.

Figure 9.3 [3] 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:

```
<template>
  <div class="ocr-component">
    <h2>OCR Functionality</h2>
    <!-- Prompt Input & Image Upload -->
    <input v-model="prompt" placeholder="Enter a
       prompt..." @keydown.enter="sendRequest" />
    <input type="file" @change="handleImageUpload"
       accept="image/*" />
```

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

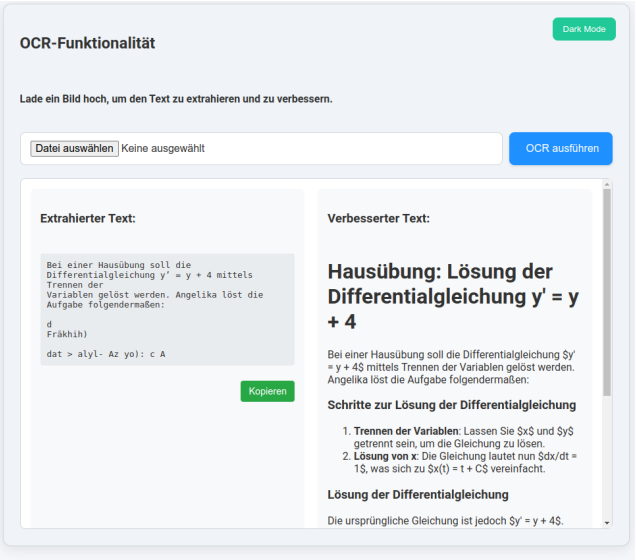

Figure 9.3.: Image to Text Tool

```
7       <button @click="sendRequest" :disabled="!selectedImage
            || loading">
8         Submit
9       </button>
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
              extracted and optimized text
45        this.loading = false;
46      },
47      copyText(text) {
48        navigator.clipboard.writeText(text);
49      }
50    }
51  };
52  </script>
```

Listing 9.6: 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 8, Section 8.4.

9.10. 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.

9.10.1. 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.

```
<template>
  <div class="chat-container">
    <!-- Chat Window -->
    <div class="chat-window" v-if="currentChat">
      <div v-for="msg in currentChat.messages"
          :key="msg.id" :class="msg.type">
        <p v-if="msg.type==='user'">{{ msg.text }}</p>
        <div v-else
            v-html="renderMarkdown(msg.text)"></div>
      </div>
    </div>
    <!-- Sidebar for Saved Chats -->
    <aside class="chat-sidebar">
      <ul>
        <li v-for="chat in chats" :key="chat.id"
            @click="loadChat(chat.id)">
          {{ chat.name }}
        </li>
      </ul>
      <button @click="startNewChat">+ New Chat</button>
    </aside>
  </div>
</template>

<script>
import firebase from 'firebase/app';
import 'firebase/firestore';

export default {
  data() {
```

```
28      return {
29        chats: [],
30        currentChat: null
31      };
32    },
33    methods: {
34      async loadChatList() {
35        const snapshot = await
              firebase.firestore().collection('chats').get();
36        this.chats = snapshot.docs.map(doc => ({ id: doc.id,
              ...doc.data() }));
37      },
38      async loadChat(chatId) {
39        const doc = await
              firebase.firestore().collection('chats').doc(chatId).get();
40        this.currentChat = { id: doc.id, ...doc.data() };
41      },
42      async saveChat() {
43        if (this.currentChat) {
44          await
                firebase.firestore().collection('chats').doc(this.currentChat.id)
45              .set(this.currentChat);
46        }
47      },
48      startNewChat() {
49        // Create a new chat session and persist it to
              Firestore.
50      },
51      renderMarkdown(text) {
52        // Convert Markdown text to HTML.
53      }
54    },
55    mounted() {
56      this.loadChatList();
57    }
58  };
59  </script>
```

Listing 9.7: 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.

9.11. 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: Overview of the platform
- Account: User Profile
- Chat Bots: Selection of available chat bots
- OCR: Image-to-Text Conversion
- OpenAI Image: Image Genderation with OpenAI
- Logout

Figure 9.4.: 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.

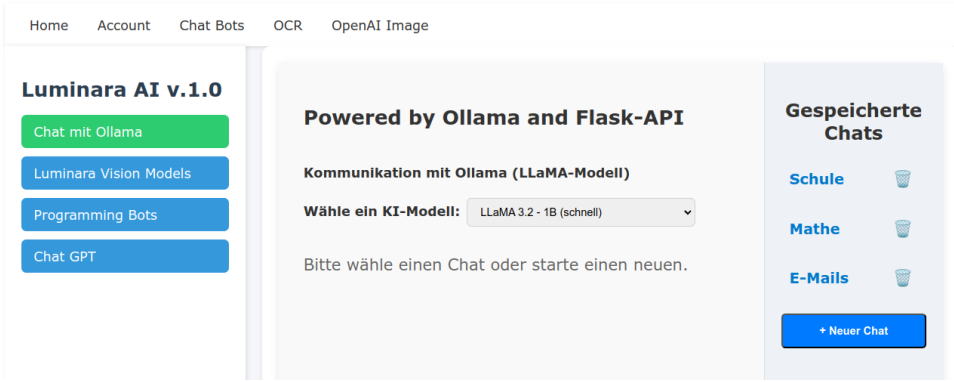

Figure 9.5.: Chat Bot Navigation

9.12. 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. As seen in Figure 9.4 and Figure 9.5.

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, 4o, and o1) as well as GitHub Copilot.

9.13. 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.

9.14. 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 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 summary, the Intelligent Student AI Hub exemplifies the transformative potential of integrating contemporary web technologies with Artificial Intelligence to cultivate a dynamic and effective learning environment. This platform not only enables students to explore the realm of AI but also establishes a foundation for ongoing enhancement and innovation in educational tools.

10. Visual Studio Code extension

Author: Florian Prandstetter

10.1. Introduction

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

10.2. What is VS Code

VS 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.

10.3. Development

10.3.1. Technologies used

- TypeScript: The VS Code extension was developed using TypeScript. TypeScript is well-suited for developing VS Code extensions, 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.
- VS Code API: The extension interacts with the VS Code API. The API allows the extension to access and modify the editor's functionality, enabling it to provide a seamless development experience.

10.4. Core Functionality

The planned core functionality of the extension is an integrated chatbot, that can answer questions without leaving the IDE. The chatbot should send the request to the server where the prompt is executed. Then the response is directly sent to the chat in the IDE. This chat should be accessible with an item that can be found in the VS Code Status Bar.

The final extension can be seen in the screenshot presented. 10.1

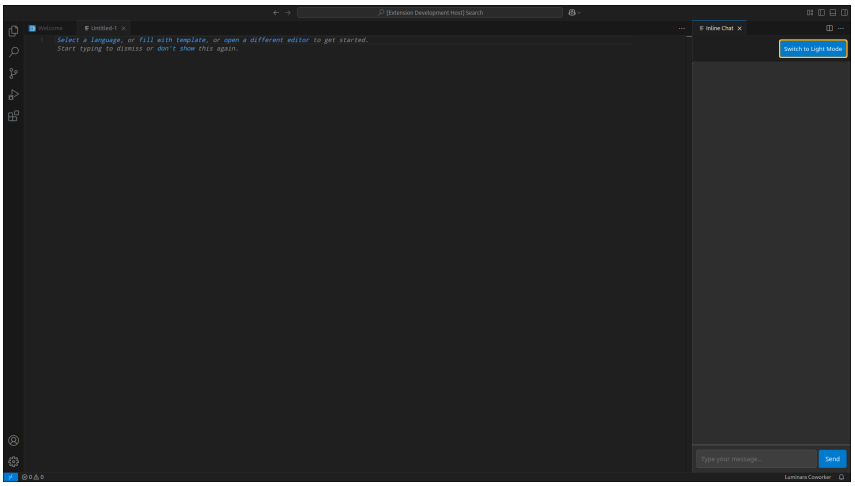

Figure 10.1.: Extension in VS Code

10.4.1. Chat

The chat is integrated into the IDE with a webview. Webviews are a tool provided by the VS Code API. They can be used to implement various GUIs into the IDE.

The following code, defines the properties of the Webview an initiates the GUI.

```
1
2     // defining the properties of the webview
3     const panel = vscode.window.createWebviewPanel(
4         'inlineChat', // Identifier
5         'Inline Chat', // Title
6         vscode.ViewColumn.Beside, // Position
7         {
8         enableScripts: true, // Allow JavaScript
9         retainContextWhenHidden: true, // Keep the webview
              state
10        }
11    );
12
13    //executing the getWebviewContent function to
          implement the GUI
14    panel.webview.html = getWebviewContent();
```

Listing 10.1: Create webview

The getWebviewContent function defines the HTML and CSS elements of the Webview. It also defines the interactions for the GUI with the TypeScript code.

```
1
2     function getWebviewContent(): string {
3         return `
4           <!DOCTYPE html>
5           <html lang="en">
6           <head>
7             <meta charset="UTF-8">
8             <meta name="viewport"
                  content="width=device-width,
                  initial-scale=1.0">
9             <title>Inline Chat</title>
10
11
12            <style>
13            <!--Style for the Chatbot -->
14            </style>
15
16
17          </head>
18          <body>
19            <!--Button to switch between dark and light
                  mode -->
20            <button class="mode-switch"
                  id="modeSwitch">Switch to Dark Mode</button>
21
22            <!--Container for the message -->
23            <div class="chat-container">
24              <div class="messages" id="messages"></div>
25              <div class="input-box">
26                <input type="text" id="input"
                      placeholder="Type your message..." />
27                <button id="send">Send</button>
28              </div>
29            </div>
30
31            <!--Backend logic for the chat -->
32            <script>
33              const vscode = acquireVsCodeApi();
34
35              <!--Sends the message and updates the GUI -->
36              document.getElementById('send').addEventListener('click',
```

```
                () => {
37              const input =
                    document.getElementById('input');
38              const message = input.value.trim();
39              if (message) {
40                vscode.postMessage({ command:
                      'sendMessage', text: message });
41
42                <!--Add user message to chat display -->
43
44                const messagesDiv =
                      document.getElementById('messages');
45                const newMessage =
                      document.createElement('div');
46                newMessage.className = 'message';
47                newMessage.textContent = message;
48                messagesDiv.appendChild(newMessage);
49
50                <!--Scroll to the latest message -->
51                messagesDiv.scrollTop =
                      messagesDiv.scrollHeight;
52
53                input.value = '';
54              }
55            });
56            <!--Logic for dark and lightmode switch -->
57            document.getElementById('modeSwitch').addEventListener('click',
                  () => {
58              const body = document.body;
59              const modeSwitch =
                    document.getElementById('modeSwitch');
60
61              if (body.classList.contains('dark-mode')) {
62                body.classList.remove('dark-mode');
63                modeSwitch.textContent = 'Switch to Dark
                      Mode';
64              } else {
65                body.classList.add('dark-mode');
66                modeSwitch.textContent = 'Switch to
                      Light Mode';
67              }
68            });
69
70            <!--Handles the display of the reply message
```

```
                -->
71              window.addEventListener('message', (event)
                    => {
72                const message = event.data;
73                if (message.command === 'displayReply') {
74                  const messagesDiv =
                        document.getElementById('messages');
75                  const newReply =
                        document.createElement('div');
76                  newReply.className = 'bot-reply';
77                  newReply.textContent = message.text;
78                  messagesDiv.appendChild(newReply);
79
80                  <!--Scroll to the latest message -->
81                  messagesDiv.scrollTop =
                        messagesDiv.scrollHeight;
82                }
83              });
84            </script>
85          </body>
86          </html>
87        `;
```

Listing 10.2: IDE Chat GUI

10.4.2. Server Request

After the user clicks send in the chats GUI, the onDidReceiveMessage command is executed.This command initiates the server request. It also defines the payload that the server receives. The payload contains the question asked, aswell as the baseprompt defined by the developers.

The request to the server is handled with Axios. Axios sends the respond to one of the Flask Service endpoints that are hosted on the server. 8

After the response reaches the client, the responses text is displayed in the chat GUI. On a failed request an error message is displayed.

```
1   // executes when the "send" button is clicked in the
        GUI
2   panel.webview.onDidReceiveMessage(
3       async (message) => {
```

```
4           if (message.command === 'sendMessage') {
5             //defines the payload that is sent
6             const payload = {
7               model: 'qwen2.5-coder:0.5b',
8               prompt: message.text
9             };
10
11      try {
12          const response = await
                axios.post('http://10.10.11.11:5001/ask_programming_bot',
                payload ); //Selecting the endpoint for the
                request and sending the payload.
13          const reply = await response.data.choices[0].text;
                //the response is then stored in the "reply"
                variable
14
15          //formating the reply and showing it in the editor
16          editor.edit((editBuilder) => {
17            reply.split('```').forEach((reply: string) => {
18              editBuilder.insert(position, `\n  # ${reply}
                    \n`);
19            });
20          });
21
22          // Send the reply to the webview to display it
23          panel.webview.postMessage({
24            command: 'displayReply',
25            text: reply.split('```').map((reply: string) =>
                  reply)
26          });
27
28      //Display and error message on failed request
29        } catch (error) {
30          vscode.window.showErrorMessage('Failed to get
                response from the server.');
31        }
```

Listing 10.3: Axios request

10.4.3. Status Bar Item

The extension also provides a status bar button that can be used to open the chat.

In the following code the Status Bar Item is created and the commands are registered.

```
1  //Creates a status bar item
2  const commandId = 'luminara-coworker.statusBarClicked';
3
4  //Registers the command
5  context.subscriptions.push(vscode.commands.registerCommand(commandId,
       async () => {
6    // Defines the options of the Quick Pick menu
7    const pageType = await vscode.window.showQuickPick(
8      ['Message', 'Chat GPT-04', 'Chat Ollama', 'Inline
         chat'],
9      { placeHolder: 'select a function' });
```

Listing 10.4: Status Bar

To assign a command it first has to be registered into the extension. This is done at the start of every TypeScript file that defines an command.

```
1
2  export async function createInlineChat(context:
       vscode.ExtensionContext ) {
3
4  const disposable =
       vscode.commands.registerCommand('luminara_coworker.startInlineChat
       () => {
5    // Code for the Command
6  }
```

Listing 10.5: registering Command

After registering a command it can be assigned to the status bar. After selecting the command the defined code will be executed.

```
1  if (pageType === 'Inline chat') {
2
3    vscode.commands.executeCommand("luminara_coworker.startInlineChat")
4  }
```

Listing 10.6: assigning Command

10.4.4. Deploying the Extension

After programming all the necessary components for the extension, they have to be registered into the extension.ts file. There they can be assigned to a specific extension context. Only files that are registered in the activate function are deployed when the extension is started.

```
1
2   export function activate(context:
        vscode.ExtensionContext) {
3
4   console.log('Congratulations, your extension
        "luminara-coworker" is now active!');
5
6   const disposable =
        vscode.commands.registerCommand('luminara-coworker.helloWorld',
        () => {
7     vscode.window.showInformationMessage('Hello World from
          luminara_coworker!');
8   });
9
10  createStatusbarItem(context);
11  luminaraChat(context);
12  luminaraChatOllama(context);
13  createInlineChat(context);
14
15
16
17  context.subscriptions.push(disposable);
18 }
```

Listing 10.7: Deploying the Extension

10.5. Conclusion

The VS Code extension provides a seamless integration of the chatbot functionality into the IDE. By leveraging the VS Code API and Axios, the extension enables users to interact with the chatbot directly within the editor, enhancing the development experience. The extension's core features, such as the chat interface and server request handling, are designed to streamline the user's workflow and provide quick access to AI-powered assistance.

Part V.

Economic aspects of AI, Open Source Projects and Operating Systems

11. Artificial Intelligence in Economics

Author: Florian Prandstetter: 11.1, 11.2, 11.3, 11.3.1, 11.3.2, 11.3.3, 11.4.4, 11.4.5,

Author: Luna Schätzle: 11.3.4, 11.4, 11.4.1, 11.4.2, 11.4.3, 11.5, 11.6

11.1. Introduction

In this chapter,the transformative role of Artificial Intelligence (AI) within economic systems and its broad impact on various sectors has been examined. The potential benefits and challenges of integrating AI into economic processes are being assesed, considering the implications for businesses, consumers, and policymakers. Additionally, current AI applications in economics and highlight emerging trends that are poised to shape the future of the field are reviewed.

11.2. The Role of AI in Economics

Artificial Intelligence has fundamentally reshaped the landscape of economics by introducing advanced analytical tools and automation capabilities. AI-driven solutions are employed to optimize production processes, enhance supply chain management, and improve customer service operations. By processing large volumes of data, AI can identify complex patterns and trends that might otherwise go unnoticed, thereby supporting more informed decision-making. Moreover, the automation of repetitive tasks allows employees to focus on strategic and creative activities, further driving innovation and efficiency within organizations.

11.3. Risks of AI

While the various benefits of AI seem promising, there are also risks associated with its implementation in economics. It is essential for businesses and policymakers to address these risks and develop strategies to mitigate them effectively.

Dizikes (2024)

11.3.1. Job displacement

One of the main concerns is the potential for job displacement, as AI technologies have the potential to automate many tasks that are currently performed by humans. This could lead to widespread unemployment and economic instability if not managed properly.

Since the benefits of AI are vast, there will be a change in the labor market comparable to the industrial revolution. But in the changing market there also will be openings for new jobs that need to be done.

Farrell (2025)

11.3.2. Bias in algorithms and data

But there also is a risk of bias. Since for many AI models it is unclear how they have been trained. Poorly trained AI models can lead to potentially devastating errors that negatively affect the company.

One of the main reasons for bias is a flawed dataset. If the training data is missing some crucial data, the results can become very unpleasing. Therefore, it is essential to evaluate the training data for potential biases.

Another factor is a poorly develpoed algorithm. A bad algorithm can amplify the data bias, and create new biases on accident.

When going to the root of the problem, it often is a cognitve bias that the model inherits from its creators. These biases can be implemented intentional and unintentional. Since all humans have some kind of bias it is nearly impossible to eliminate them.

Mitigation AI bias can be a very resource heavy task. After identifying a bias, the model needs to be retrained. This can cost a lot of both money and time.

Data & Team (2023)

11.3.3. Transparency

Some of the biggest tech companies that offer AI services tend to be very non-transparent with their data. The latest example of this is the Deepseek r1 model developed by a chinese startup. But other players like OpenAi's ChatGPT also have similar issues.

Due to the non-transparent handling of private information, many concerns arise. Most private users and companys do not want their data to be on unknown servers.

Additionally it is hard to identify the source of biases these models may have. Since both algorithm and training data are not public knowledge, it becomes nearly impossible to mitigate them.

Larsson & Heintz (2020)

11.3.4. Training data and ethical concerns

Training data forms the foundation of AI systems, fundamentally influencing their performance, biases, and decision-making processes. However, the methods employed in assembling and curating these datasets have sparked significant ethical debates, primarily due to concerns surrounding copyrighted material, transparency, and data privacy.

A primary ethical issue is the incorporation of copyrighted content into training datasets. Many AI models are trained on vast amounts of internet-sourced data, much of which is subject to copyright restrictions. The unauthorized use of such material not only raises legal questions regarding intellectual property rights and fair use but also challenges the ethical legitimacy of reproducing or generating content that may infringe on these rights. This dilemma underscores the need for clear guidelines and licensing frameworks that respect the rights of original content creators.

Equally concerning is the lack of transparency in the disclosure of training data sources. Many state-of-the-art models do not provide comprehensive information about the origin, composition, or curation process of the data they use. This opacity hampers independent verification of data quality and bias, making it difficult for researchers, regulators, and the public to evaluate the fairness, representativeness, and potential ethical implications of the underlying datasets. In the absence of detailed documentation—such as datasheets for datasets—the risks associated with hidden biases and privacy violations remain unaddressed.

Furthermore, the aggregation of large datasets often includes personal information that may have been collected without explicit consent. The use of such data can lead to privacy breaches and raises questions about the ethical treatment of individuals whose information is repurposed for AI training. This issue not only compromises personal privacy but also potentially exposes organizations to legal repercussions under data protection regulations.

In conclusion, while extensive training data is essential for developing robust AI systems, the ethical concerns regarding copyrighted material, lack of transparency, and privacy breaches highlight the urgent need for standardized ethical guidelines and regulatory oversight. Addressing these challenges is crucial to ensure that the development and deployment of AI technologies are both legally compliant and ethically responsible.

11.4. Applications of AI

There are numerous applications of AI in economics. In the following sections, some of the most common use cases that are currently being implemented in various sectors of the economy will be discussed.

11.4.1. Customer Service

AI-powered chatbots are increasingly deployed in customer service roles, offering capabilities such as answering queries, providing information, and even completing transactions without human intervention. This integration has the potential to significantly enhance service quality while simultaneously reducing operational costs.

Compared to traditional human-driven customer support, AI-supported chatbots provide several key advantages for companies. Their round-the-clock availability ensures that customer inquiries can be addressed at any time, eliminating the constraints of business hours. Additionally, they contribute to reduced operational costs by automating routine interactions, minimizing the need for large customer service teams. Their ability to scale effortlessly allows businesses to handle a high volume of requests without compromising efficiency. As a result, overall customer satisfaction is often improved through faster response times and consistent service delivery.

From the customers perspective, AI-powered chatbots offer immediate responses, eliminating the frustration of long waiting times. They ensure consistency and accuracy in the information provided, reducing the likelihood of errors or miscommunication. The increased convenience of instant access to assistance, coupled with personalized interactions tailored to individual preferences, enhances the overall user experience. Furthermore, the multilingual capabilities of AI chatbots facilitate seamless communication across diverse customer bases, making them a valuable tool for global businesses.

Despite these benefits, challenges remain. Complex inquiries often require human intervention, as AI systems may struggle with nuanced or highly specialized requests. Moreover, achieving widespread customer acceptance of AI-driven support solutions remains a critical consideration, as some users may still prefer human interaction Global (2025).

Looking ahead, advancements in AI are expected to further expand the role of chatbots beyond customer service into areas such as sales and marketing. By analyzing customer data, AI could enable the delivery of highly personalized product recommendations and service offers, enhancing customer engagement and driving business growth.

11.4.2. Supply Chain Optimization

AI-driven solutions are increasingly being integrated into supply chain management, significantly enhancing operational efficiency and resilience. By leveraging machine learning algorithms and advanced analytics, these systems enable organizations to forecast demand with greater accuracy, optimize inventory levels, and streamline logistics. This proactive approach allows businesses to mitigate potential disruptions before they escalate, ensuring smoother supply chain operations.

For companies, AI integration offers several advantages. Improved forecasting accuracy enhances demand planning, reducing instances of overstocking or stock shortages. Optimized inventory management leads to cost reductions by minimizing waste and maximizing resource allocation. Additionally, AI-powered logistics and transportation solutions improve efficiency, ensuring timely deliveries and reducing operational bottlenecks. Overall, these advancements contribute to greater agility, allowing companies to adapt swiftly to market fluctuations.

From a broader supply chain perspective, AI facilitates greater visibility across the entire network, providing stakeholders with real-time insights into inventory movement and demand patterns. Enhanced communication and collaboration among supply chain partners streamline workflows and reduce inefficiencies. Furthermore, AI-driven systems enable faster responses to market changes and potential disruptions, ensuring business continuity. Risk management is also strengthened through real-time data analysis, helping organizations anticipate and mitigate vulnerabilities.

Despite these benefits, several challenges persist. The integration of diverse data sources remains complex, requiring sophisticated infrastructure and interoperability between different systems. Implementing advanced AI solutions can also be resource-intensive, necessitating significant investment in technology and expertise. Additionally, ensuring robust cybersecurity measures is crucial, as AI-driven supply chains handle vast amounts of sensitive data.

Nonetheless, ongoing advancements in AI, coupled with emerging technologies such as IoT and blockchain, are expected to further refine supply chain processes. These innovations will support the development of more resilient, agile networks, enabling businesses to navigate an increasingly dynamic and interconnected global market IBM (n.d.*b*).

11.4.3. Predictive Analysis

Artificial Intelligence has become a cornerstone in modern economic forecasting, with predictive analysis playing a pivotal role in enhancing decision-making processes. By leveraging sophisticated machine learning algorithms and big data analytics, AI-driven predictive models can process complex historical and real-time datasets to uncover hidden trends and non-linear relationships, thereby increasing the accuracy of economic predictions **?**.

The integration of AI in economic forecasting brings several key benefits. It improves forecasting accuracy through advanced pattern recognition, which enables the identification of subtle trends that might otherwise go unnoticed. Additionally, AI allows for the early detection of market shifts and potential economic downturns, providing a proactive approach to economic management. Real-time analysis further enhances these capabilities by allowing for dynamic adjustments to forecasting models, ensuring they remain accurate and relevant as new data becomes available. Furthermore, AI aids in enhanced risk assessment, enabling more proactive and informed decision-making by pinpointing potential threats and opportunities in advance.

These predictive capabilities empower policymakers and business leaders to mitigate risks and capitalize on emerging opportunities by providing a clearer understanding of economic trajectories. However, challenges remain, such as the need for high-quality data, addressing model biases, and ensuring transparency in AI methodologies. Continued advancements in AI and data analytics are expected to refine these predictive techniques further, paving the way for even more robust economic forecasting frameworks **?**.

11.4.4. Data Analysis

One of the most significant benefits of AI in economics is its ability to analyze large amounts of data quickly and accurately. This can help businesses identify trends and patterns that would be difficult for humans to detect, allowing them to make more informed decisions.

AI often has a more objective view on the provided data. Thus an AI analysis can help to get a new perspective on the data.

Another benefit is the visualization of data. Generative AI can give a clear graphical overview of given data, making it easier to find patterns or anomalies.

But there are still a lot of Risks that are tied to AI. If trained poorly an AI model can easily make devastating mistakes that negatively affect the company. So it is essential to still have a human check the results to avoid AI hallucination.

Houbrechts (2024)

11.4.5. Automation

AI can also be used to automate repetitive tasks, thus freeing up employees to focus on more important activities. This can help businesses increase their productivity and reduce costs.

Due to the learning capabilities of AI, it can not only repeat a give task, but also adapt quickly to changes without a lot of intervention. With the help of AI there are also many ways to automat a detailed documentation of the tasks. This can help to find errors and provide valuable data for process optimization. If utilized correctly this will help reduce labor and production costs in many sectors.

IBM (2021)

11.5. Regulatory Challenges

The increasing deployment of Artificial Intelligence in various economic sectors has introduced a wide range of regulatory challenges, particularly regarding data security, privacy, and compliance with both established legal frameworks and emerging regulatory measures. A landmark initiative addressing these concerns is the EU AI Act (Regulation (EU) 2024/1689 *Regulation - EU - 2024/1689 - EN - EUR-Lex* (2024)), which represents the first comprehensive legal framework for Artificial Intelligence on a global scale. This Act is designed to harmonize AI regulations across EU member states by adopting a risk-based approach that categorizes AI systems according to their potential impact on fundamental rights and public safety.

Under the EU AI Act, AI systems are classified into several risk tiers—from minimal to high risk—with the most critical applications subject to stringent requirements, including mandatory conformity assessments, robust data governance practices, and enforced human oversight. These measures aim to enhance transparency, accountability, and public trust in AI technologies. Moreover, the framework promotes continuous post-market monitoring and adaptive regulatory responses to keep pace with rapid technological advancements.

By establishing clear guidelines for AI deployment, the framework seeks to balance the dual objectives of fostering innovation while safeguarding societal values. This comprehensive approach not only addresses the complex legal and ethical issues

inherent in AI but also sets a precedent for future regulatory initiatives at the international level. *AI Act | Shaping Europe's digital future* (2025)

As discussed in Chapter 15 and Section 15.1.3, numerous experts advocate for the implementation of more extensive global regulations to effectively mitigate the risks associated with Artificial Intelligence.

As discussed in Chapter 7, particularly in Section 7.6.2 on Data Security and Privacy in Compliance with Austrian and EU Regulations, several regulatory challenges affect the data security and privacy of AI models.

A key issue is that user data is frequently stored on cloud servers, which increases the risk of security breaches and data leaks. Moreover, many leading AI companies are headquartered outside the European Union, potentially complicating compliance with EU and Austrian data protection laws.

11.6. Conclusion

The rapid evolution of Artificial Intelligence (AI) is poised to significantly transform economic landscapes worldwide. As AI technologies mature, they are expected to drive profound changes in labor markets, productivity, and the overall distribution of economic gains. This transformation is multifaceted, presenting both substantial opportunities for growth and innovation as well as considerable challenges related to workforce displacement, income inequality, and the need for robust regulatory frameworks.

Moreover, the shift towards an AI-centric economy underscores the critical importance of workforce adaptation. As routine tasks become increasingly automated, there is a growing imperative for reskilling and continuous learning initiatives that can equip workers with the skills necessary for emerging roles in the digital era. Policymakers are therefore called upon to design inclusive strategies that not only harness the potential of AI but also mitigate the risks of technological disruption.

In summary, although AI presents unprecedented opportunities for economic advancement, it also necessitates careful consideration of its broader societal impacts. As this transformative era is develping furhter, proactive measures ranging from investment in human capital to the formulation of forward-looking regulatory policies will be crucial to ensuring that the benefits of an AI-powered economy are both sustainable and equitably shared.

12. Open source evaluation on Economics

Author: Luna Schätzle

12.1. 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.

12.1.1. What is open source?

Open safeguardource 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. – Linus Torvalds

Linus Torvalds - Software is like sex: it's better when... (1999)

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.

Opensource.com (n.d.)

12.1.2. Advantages of open source

Open source software offers a wide range of benefits that render it a cornerstone of modern technology. Notably, its cost efficiency, being typically available free of charge, allows organizations and individuals to significantly reduce licensing and maintenance expenditures. Moreover, the flexibility inherent in open source solutions permits users to access the source code and tailor the software to meet their specific needs and requirements. This openness further enhances security by facilitating extensive peer review, which enables the prompt identification and remediation of vulnerabilities. In addition, the vibrant community support characteristic of open source projects provides continuous updates, patches, and assistance, thereby fostering an environment where collaborative innovation thrives. This collaborative ecosystem encourages creativity and often leads to groundbreaking advancements and solutions. Furthermore, many open source projects are designed with compatibility in mind, ensuring seamless integration with existing systems and reducing technical barriers. Finally, the transparency afforded by open access to the source code not only allows users to thoroughly understand and verify the operational mechanics of the software but also guarantees the freedom to use, modify, and share the software without restrictive licensing agreements.

10 biggest advantages of open-source software (2022)

12.1.3. Why Do People Use open source?

The adoption of open source software is driven by a variety of compelling factors. One significant aspect is the control it affords users, who can fully customize and optimize the software for specific use cases. This control is closely linked to cost savings, as the absence of licensing fees considerably reduces expenses, a factor that is particularly advantageous for startups and educational institutions. Additionally, the transparency of the source code not only facilitates thorough auditing and bolsters security, but it also enhances overall trust and reliability. The collaborative spirit intrinsic to open source initiatives further connects users with knowledgeable communities that actively share resources and provide support. Moreover, many open source projects are characterized by long-term stability, offering regular updates and ongoing support that ensure the software remains reliable over time. Finally, the use and development of open source tools present valuable opportunities for skill development in both educational and professional contexts, equipping individuals with skills that are increasingly in demand.

12.2. What is and isn't open source?

Open source, as defined by the Open Source Initiative (OSI), represents a development paradigm that emphasizes both accessibility and transparency in software creation. This approach grants users the freedom to inspect, modify, and distribute the source code, thereby fostering an environment ripe for collaboration and innovation.

The OSI delineates several fundamental principles that underpin open source software. First, free redistribution ensures that the software can be shared and disseminated without any restrictions, thereby promoting widespread use. Second, guaranteed access to the source code allows users not only to study the inner workings of the software but also to modify and enhance it according to their needs. Third, the principle of modification and sharing permits users to develop and disseminate derivative works, provided that they adhere to the stipulated license terms. Additionally, the commitment to non-discrimination ensures that the software is accessible to all individuals, regardless of their background or professional affiliation. Finally, neutrality and compatibility are maintained by ensuring that the license does not favor any specific technology or impede the integration of other software solutions.

Collectively, these principles secure open source as a transparent, inclusive, and adaptable model for software development, thus driving innovation and facilitating collaboration across diverse industries and communities.

Initiative (2007)

12.2.1. Misconceptions About open source

Open source is frequently misunderstood and often conflated with other software distribution models, which can lead to misconceptions regarding its nature, functionality, and benefits. It is essential to differentiate open source from other categories of software, as each has distinct characteristics and implications for users.

Open source software is defined by its free accessibility, modifiability, and redistributability under an open source license, all of which promote transparency and collaboration. In contrast, freeware refers to software that is available at no cost but typically does not provide access to its source code, thereby preventing users from modifying or redistributing it. Proprietary software, on the other hand, is owned and controlled by a single entity, restricting access to the source code and limiting user modifications or redistribution. Additionally, commercial software is sold for profit and may be either open source or proprietary, depending on the licensing terms.

Understanding these distinctions enables users to make informed decisions regarding software selection and ensures that their expectations align with the capabilities and freedoms offered by the chosen software. To verify whether a piece of software is genuinely open source, one should examine its license agreement and confirm that the source code is readily available. An OSI-approved license serves as a reliable indicator that the software adheres to open source principles, thus providing transparency, freedom, and opportunities for collaboration.

A common misconception about open source arises from the phrase “free as in freedom” versus “free as in free beer.” The former underscores the liberty to access, modify, and share the software, whereas the latter merely denotes that the software is available at no cost. Although many open source projects are free of charge, their true value lies in the freedoms they confer upon users, developers, and organizations. This distinction underscores the broader significance of open source as a philosophy rather than just a pricing model.

Forbes Technology Council (2024)

12.3. 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.

Disadvantages of open source Software

Key drawbacks associated with open source software include limited support, reliance on hobby developers, fragmentation, and a potentially reduced feature set. In many cases, open source projects do not have dedicated support teams, which can result in slower response times for addressing bugs and technical issues. Additionally, projects maintained by volunteers or hobbyists may experience irregular updates and inconsistent maintenance, thereby affecting their overall reliability. The decentralized nature of open source development sometimes leads to fragmentation, with multiple versions and distributions emerging and causing compatibility challenges. Furthermore, certain open source applications may lack some of the advanced features and functionalities that are commonly found in commercial alternatives.

Mathpati (2023)

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.

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.

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.

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.

Hermansen (2024)

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 are specialized software licenses that foster collaborative development by permitting unrestricted use, modification, and sharing of software. These licenses are characterized by several key features. First, unrestricted use means that the software can be employed for any purpose without limitations. Second, the availability of the source code allows users to inspect, modify, and enhance the software, thus encouraging continuous improvement. Third, redistribution rights enable users to share both the original and modified versions of the software, further promoting community-driven development.

Notable examples of open source licenses include the GNU General Public License (GPL), which requires that all derivative works remain open source; the permissive MIT License, which imposes minimal restrictions on usage and redistribution;

and the Apache License, which strikes a balance between flexibility and patent protection. The selection of an open source license is a critical decision, as it can significantly influence the software's development trajectory, market adoption, and the level of community engagement.

Rahmatallah (n.d.)

12.4. 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.

12.4.1. Common Risks Associated with Open Source Software

Several risks are inherently linked to the utilization of open source software. One prominent concern is the presence of security vulnerabilities; open source projects can harbor inherent flaws that, if not promptly patched, may be exploited by malicious actors to gain unauthorized access to systems and sensitive data. Furthermore, the intricate landscape of open source licenses necessitates strict compliance, as non-adherence can precipitate legal disputes, financial penalties, and reputational damage. Additionally, dependency and supply chain risks emerge from the reliance on third-party libraries and components, each potentially introducing vulnerabilities and compatibility challenges across the software ecosystem. Moreover, many open source projects are maintained by volunteer communities rather than dedicated support teams, which may result in delayed updates and prolonged exposure to unresolved security issues. Finally, concerns regarding quality and code integrity arise due to variability in coding practices, insufficient testing, and poor documentation, factors that can contribute to inconsistent software quality and elevate the likelihood of bugs and security weaknesses.

Mathpati (2023)

12.4.2. Specific Security Concerns in open source Environments

Security risks in open source software can manifest in various ways, posing significant challenges if not properly managed. One major concern is the presence of **malware and backdoors**; since the source code is publicly accessible, malicious actors may attempt to inject harmful code or create covert backdoors if rigorous code reviews and continuous monitoring are not enforced. Additionally, **supply chain attacks** are a growing threat, as organizations integrating multiple open source components become vulnerable when attackers exploit less secure dependencies, potentially compromising the broader software ecosystem.

Another risk involves **delayed patch management**—open source projects may face delays in identifying vulnerabilities and deploying patches, leaving systems exposed to potential exploitation. Furthermore, **suboptimal developer practices**, including inadequate testing, inconsistent coding standards, and poor documentation, can exacerbate security concerns by increasing the risk of undetected errors and weaknesses in the code. Lastly, **compliance risks impacting security** arise when organizations fail to adhere to licensing terms, which can not only lead to legal consequences but also force disruptive changes to the software stack. Such transitions may introduce new vulnerabilities if not handled carefully.

Helms (2023)

In summary, while open source software offers powerful and cost-effective solutions for innovation, its adoption necessitates vigilant risk management. Organizations must implement robust security protocols, conduct regular audits of open source components, and maintain strict compliance with licensing requirements to effectively mitigate these risks.

12.5. 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.

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 Hendrickson et al. (2012) 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.

12.5.1. 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. *How does open-source benefit startups?* (n.d.)

12.5.2. 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 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.

12.6. Open source in Key Industries

Across numerous industries, open source software has exerted a profound influence on organizational operations, catalyzing innovation and fostering collaborative development. In the field of information technology, open source solutions form the backbone of critical infrastructures, including operating systems, databases, and web servers, thereby enhancing system reliability, scalability, and flexibility. Similarly, Artificial Intelligence has witnessed significant advancements due to open source frameworks such as TensorFlow and PyTorch, which have democratized access to AI technologies and accelerated research and innovation.

Education has also benefited from open source platforms like Moodle and Jupyter Notebooks, which have transformed online learning by making educational resources more accessible and interactive. This shift has fostered broader pedagogical engagement and enabled institutions to develop more dynamic learning environments. In the healthcare sector, open source software plays an increasingly vital role in managing electronic health records, medical imaging, and telemedicine applications, thereby improving patient care, data security, and system interoperability.

The financial industry, too, has embraced open source solutions, integrating them into trading platforms, risk management systems, and blockchain technologies. By doing so, financial institutions enhance transparency, operational efficiency, and innovation in a sector that demands both security and adaptability. Across these diverse domains, open source software continues to drive technological progress, providing cost-effective, scalable, and collaborative solutions that reshape industry practices and standards.

12.6.1. 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. Galope (2024)

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. *Home | LibreOffice - Free and private office suite - Based on OpenOffice - Compatible with Microsoft* (n.d.)

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.

Openemr Software Review - Ambula Healthcare (n.d.)

12.7. 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.

12.7.1. Common Business Models

Several business models have proven successful in the open source landscape, each leveraging the principles of openness while generating sustainable revenue. One prevalent approach is the open core model, in which the core software remains open source and freely accessible, while advanced features, enterprise functionalities, or premium support are offered under a commercial license. Companies such as MongoDB and GitLab have successfully adopted this model, balancing community-driven development with monetizable enhancements.

Another widely utilized strategy involves hosting and cloud-based solutions, where companies provide managed services, infrastructure, or cloud-hosted versions of their open source software. By charging for reliability, scalability, and additional features, businesses like WordPress and Databricks have capitalized on this model, ensuring both accessibility and financial viability. Similarly, revenue can be derived from support and maintenance services, where organizations offer professional assistance, security updates, and consulting to enterprises relying on open source technologies. Companies like Red Hat and Canonical (Ubuntu) exemplify this approach, demonstrating how expertise and long-term support can serve as a profitable foundation for open source businesses.

Beyond these primary models, alternative revenue streams, such as donations, dual licensing, and strategic partnerships, also contribute to the sustainability of open source projects. These diverse monetization strategies highlight the adaptability of open source ecosystems, allowing businesses to thrive while maintaining the collaborative and transparent ethos that defines the movement.

12.7.2. Open source regarding AI Models

In the field of Artificial Intelligence, categorizing models as open source or proprietary presents significant challenges due to the ambiguity surrounding their accessibility and licensing. While some models are advertised as open source, they often fail to meet the fundamental principles that define true openness.

For an AI model to be genuinely open source, it must satisfy several key criteria. Firstly, both the model weights and its underlying architecture must be publicly available, ensuring that users can fully understand and utilize the model. Additionally, it must be distributed under a recognized open source license, such as the MIT License or Apache License, which guarantees the rights to use, modify, and distribute the software without restrictive limitations. Beyond licensing, true openness also requires that users have the ability to modify and redistribute the model, fostering a collaborative development environment. Furthermore, comprehensive documentation and usage guidelines must accompany the model, ensuring accessibility and ease of implementation for a broad user base. Finally, transparency in training data is essential, either by providing direct access to the datasets used in model development or by clearly specifying the sources and methodologies involved in data collection.

The distinction between genuinely open source AI models and those that merely claim openness is critical, as it impacts the broader AI ecosystem, influencing collaboration, research, and ethical considerations in AI development.

12.8. 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.

Open Source Guide für Österreich | netidee (n.d.)

12.9. 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. 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.

12.10. 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-Schaetzle/Diploma-thesis-website`.

12.10.1. 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.

12.11. 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.

13. Economic Aspect of Operating Systems

Author: Florian Prandstetter

13.1. Introduction

In this section, the relevance of Operating Systems in our economy will be discussed. The importance of OS in our daily life is not always obvious. The OS is the most important software on a computer. It manages the computer's memory, processes, and all of its software and hardware. With the rising amount of digitalization, the OS is also becoming more important than ever.

To get an overview of what an OS is, please refer to Chapter 5.2.

13.2. Operating System Market Share

The OS market is dominated by four major players: Android, Microsoft Windows, Linux, and macOS. The market share of the Operating Systems is shown in Figure 13.1.

With the rise of smartphones, Android has become the most popular OS in the world. It is used on more than 70% of all smartphones. It is followed by iOS, which is used on Apple's iPhones.

On desktop computers, Microsoft Windows is the most popular OS. It is used on more than 20% devices. It is followed by macOS, which is used on Apple's MacBooks. Linux also is a popular OS, though it is not as widely used as Windows or macOS.

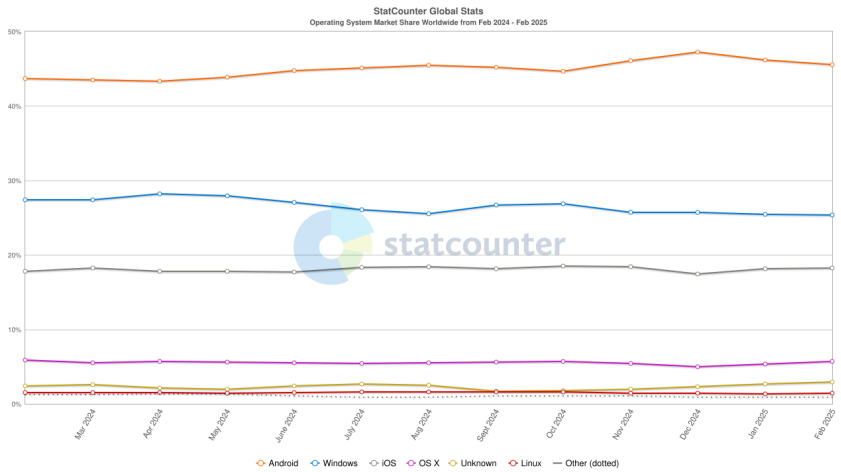

Figure 13.1.: Operating Systems Market Share

Operating System Market Share Worldwide | Statcounter Global Stats (n.d.)

The global market of Operating Systems is expected to reach $48.18 billion at a CAGR of 1.9% in 2026. businesswire (2022)

13.2.1. Operating Systems for Servers

When looking at the Operating Systems used for web servers, the market share looks very different. The two main competitors are Microsoft and Red Hat.

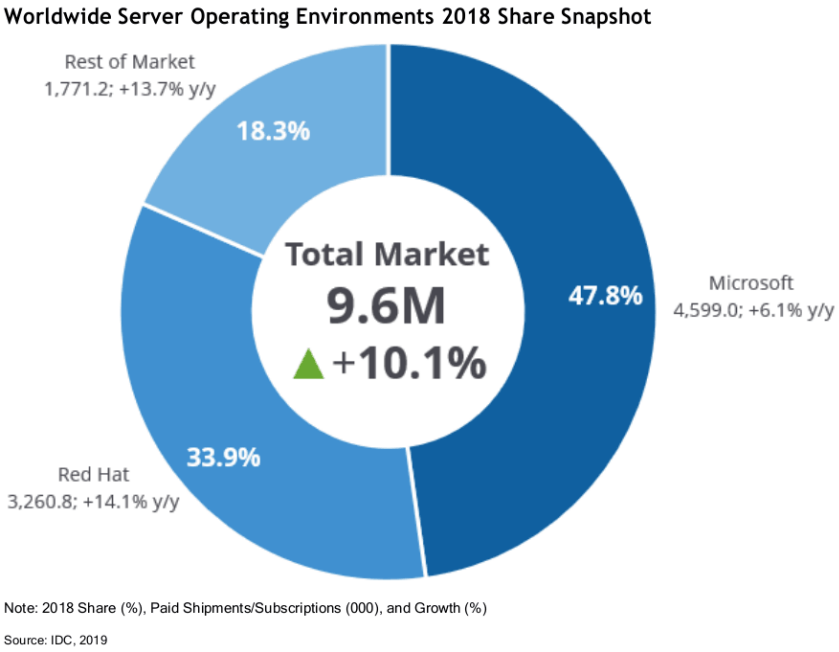

Figure 13.2.: Operating Systems for Servers Market Share

As of 2018, Microsoft held 47.8% of the market, while Red Hat held 33.9%. The remaining 18.3% of the market is shared by different competitors.

Red Hat is known for its Linux-based Operating Systems.Due to the reliability and security of Linux, it is a popular choice for servers.

Microsoft on the other hand is known for its Windows Server OS. The company has a long history in the server market and is a trusted provider of server OS. though the trend over the last years has been a decrease in market share for Microsoft.

(2022)

Market Volume

Due to the rapid growth of the internet and related services, the market for servers and Operating Systems grew alongside it. As shown in Figure **??**, the market volume for Operating Systems for Servers has been steadily growing over the last

years. It's expected to grow to double its current volume by 2032. This is due to the increasing demand for cloud services and the digitalization of many industries.

The increasing demand for AI and Big Data services is also driving the growth of the server market. These services require powerful servers with specialized Operating Systems to handle the workload. This trend is expected to continue as more industries adopt digital technologies and cloud services. Since these services mostly run on Linux-based servers, the market share of Linux is also expected to grow in the future.

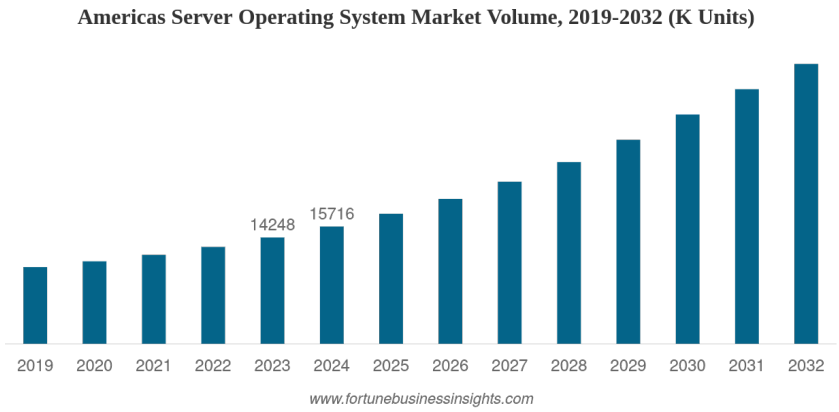

Figure 13.3.: Operating Systems Market Volume

fortune business insights (2025)

13.2.2. Operating Systems for Embedded Systems

Embedded Systems are also a big market for Operating Systems. These are compact systems that are used in many different devices. The CAGR for the Embedded Systems Market is expected to be 6.65% from 2023 to 2030. Thus, the market volume is expected to grow to $9,519.09 million by 2028. One of the largest markets for Embedded Systems is the automotive industry.

Dharmadhikari (2025)

13.3. Important Players

There are many different companies that develop Operating Systems. In this section, some of the most important ones will be discussed.

13.3.1. Microsoft

Microsoft is one of the biggest players in the OS market. They are most widely known for their Windows OS. The company was founded by Bill Gates and Paul Allen in 1975. The company is headquartered in Redmond, Washington.

The company has a market capitalization of 2.5 trillion dollars.

They currently hold 47.8% of the Server OS Market and 70.62% of the Desktop OS Market. They are also active in the Server OS Market with Windows Server and the Embedded Systems Market with their Windows Embedded OS.

Contributors (2025*b*) *Desktop Operating System Market Share Worldwide | Statcounter Global Stats* (n.d.)

13.3.2. Apple

Apple is another big player in the OS market. They are known for their macOS for desktop computers and iOS for their phones and tablets. The company was founded by Steve Jobs, Steve Wozniak, and Ronald Wayne in 1976. The company is headquartered in Cupertino, California.

The company has a market capitalization of 2.5 trillion dollars. They currently hold 15.74% of the Desktop OS Market and 27.78% of the Mobile OS Market.

Contributors (2025*a*)

13.3.3. Red Hat

Red Hat specializes in Linux-based Operating Systems. The company was founded in 1993 and is headquartered in Raleigh, North Carolina. The company was acquired by IBM in 2019 for 34 billion dollars.

They are the second biggest player in the Server OS Market with a market share of 33.9%. Due to its open-source nature, Red Hat has a large community of developers contributing to the OS growth.

Red Hat – Wikipedia (n.d.)

13.3.4. Wind River Systems

Wind River Systems is a company that specializes in Embedded Systems. The company is known for its VxWorks OS. The system is used in many different industries like automotive, aerospace, defense, and industrial. The company was founded in 1981 and is headquartered in Alameda, California.

They used to be a subsidiary of TPG Capital but were sold to Aptiv in 2022 for $3.5 billion dollars.

Aptiv (2022)

13.4. Conclusion

The Market for Operating Systems in all relevant branches of the Industry has been steadily growing over the last decade. Due to the high amount of competition there has been constant development of the technology. This trend is expected to persist as digitalization becomes more integrated into everyday life and industries continue to demand more sophisticated and specialized Operating Systems. With emerging technologies like cloud computing, Artificial Intelligence, and the Internet of Things (IoT), new opportunities for OS development and market expansion are constantly arising.

Moreover, the competition between major OS developers ensures continuous innovation and improvement in performance, security, and user experience. Companies

are investing heavily in research and development to maintain their market positions and adapt to evolving consumer and enterprise needs.

In conclusion, the Operating System market remains a vital and dynamic sector in the global economy. Its influence extends beyond traditional computing devices, shaping the future of technological progress across various industries.

Part VI.

Conclusion

14. Conclusion

Author: Luna Schätzle

Author: Florian Prandstetter 14.1

In this thesis, a comprehensive investigation into *[Artificial Intelligence in the Industry and Education Environment]* was undertaken. The primary aim was to explore the potential applications of AI within both industrial and educational contexts and to develop innovative solutions to address the challenges inherent in these domains. The research objectives were defined as follows:

- **Task 1:** Establish an API server for hosting and executing AI models.
- **Task 2:** Develop a robust backend architecture for both the AI Hub and the Code Extension.
- **Task 3:** Identify and integrate optimal AI models for a range of tasks into the backend systems of the AI Hub and the Code Extension.
- **Task 4:** Develop an AI Hub that provides students with access to a diverse array of AI tools and resources, including chatbots, image recognition, and natural language processing (NLP) models.
- **Task 5:** Implement an AI-driven coding assistance solution, available as an integrated extension within a code editor.
- **Task 6:** Compile comprehensive documentation detailing the conceptual framework, theoretical foundations, practical implementation, primary and alternative solution approaches, as well as the results and their interpretation.

14.1. Key Findings

The research indicates that AI technologies possess the potential to fundamentally transform both industrial and educational settings by providing innovative solutions to complex challenges. The following section summarizes the principal findings of this study:

AI in Education In academic institutions such as schools and universities, AI can significantly enhance the learning experience by offering personalized educational resources, automating grading processes, and supporting intelligent tutoring systems. However, the implementation of AI in educational settings is often challenged by a shortage of specialized expertise, high implementation costs, and ethical concerns regarding data privacy and security. Nonetheless, the substantial benefits offered by AI necessitate that educational institutions actively explore and adopt these technologies to improve learning outcomes and better prepare students for future challenges. Moreover, given the wide range of available AI applications—ranging from chatbots and image recognition to natural language processing (NLP) models—it is imperative to select and integrate the most appropriate models for specific educational tasks.

AI in Development AI tools and resources can significantly enhance the development process by providing intelligent coding assistance, automating repetitive tasks, and improving code quality. The integration of AI-driven solutions into code editors, such as Visual Studio Code, enables developers to write code more efficiently, identify errors, and optimize their workflow. These tools are likely going to improve over time, as the AI models are trained with more data and the technology advances. The future for AI in development looks promising, as more and more companies are investing in AI-driven solutions to improve their development processes. This trend can already be seen in tools like GitHub Copilot, which uses AI to provide code suggestions and completions to developers.

Server and operating systems Building and maintaining the Server Environment turned out to be more challenging than originally anticipated. The team had to deal with various issues, such as compatibility problems and performance bottlenecks. After hard work the Team was able to establish a stable and efficient server environment, which was crucial for the successful deployment of the AI models and the overall functionality of the AI Hub and the Code Extension.

Different LLM Models for Different Purposes For the AI Hub and the Code Extension, the research team conducted an extensive evaluation of available AI models to identify the most appropriate options for various tasks, including chatbots, image recognition, and natural language processing (NLP). The challenges associated with implementing the optimal AI models for specific tasks are substantial, due to the inherent complexity of these technologies, the limited availability

of specialized expertise, and the considerable costs related to their development and deployment. Consequently, identifying suitable AI models necessitates thorough research and systematic evaluation to select those that best meet the project's requirements. Although this process is both time- and resource-intensive, it is essential for ensuring the project's overall success.

Flask Server for API The research team established an API server to host and execute the AI models, leveraging the Flask framework a lightweight and efficient web framework for Python. This Flask server provides a RESTful API that facilitates client interaction with the hosted AI models by handling model loading, input data processing, and the delivery of output predictions. As a critical component of both the AI Hub and the Code Extension, the server enables the seamless integration of AI functionalities into these applications. By centralizing the hosting of the models, the team ensures that they remain easily accessible, as well as readily updatable and maintainable. Flask's versatility and extensive feature set further underscore its suitability for developing robust APIs, offering straightforward route definition, request handling, and response generation.

Web Application for User Interface The frontend of the AI Hub and the Code Extension was developed using the Vue framework, a progressive JavaScript framework recognized for its ease of use and flexibility. The development team's prior experience with Vue enabled the rapid creation of frontend components, which are essential for building interactive and responsive user interfaces. Vue's comprehensive suite of features and tools facilitated the integration of various components—including chatbots, image recognition tools, and NLP models—into the applications. This robust and versatile framework is particularly well-suited for modern web application development, as it supports the creation of engaging, intuitive, and dynamic user interfaces.

Working with Visual Studio Code The main IDE used by the team to develope the application was Visual Studio Code. This IDE offers a wide range of features and extensions that enhance the development process, including code completion, debugging tools, and version control integration. Working with VS Code can sometimes be challenging due to its complexity and the need to configure various settings and extensions. However, the benefits of using this powerful IDE—such as its extensive customization options, rich ecosystem of extensions, and seamless integration with other tools—far outweigh the challenges.

14.2. Working in a Team

The project team comprised three members, each assigned distinct roles and responsibilities to ensure the successful execution of the project. Collaborative work in a team setting is both challenging and rewarding, demanding effective communication, coordination, and mutual support. The following sections delineate the critical aspects of teamwork that significantly contributed to the project's success:

- **Constant and Open Communication** Due to the direct and clear communication between team members, the project was able to progress smoothly and efficiently. Regular meetings, updates, and feedback sessions were instrumental in fostering a collaborative work environment and ensuring that all team members were aligned with the project's objectives and timelines.
- **Distinction of Tasks and Responsibilities** With a clear division of the tasks, the team was able to work efficiently in their areas of expertise. Their were rarely any overlaps or conflicts.
- **Documentation and Repository Management** The team maintained a structured and detailed documentation of the current status. This allowed a quick overview of the project and the progress made.
- **Flexibility and Adaptability** Due to the constant communication, the team was able to adapt to any changes and challenges that arose during the project.
- **Accessibility** The team members were always available for questions and feedback. This allowed for quick problem solving and a smooth workflow.

Team Leadership and Project Management Luna Schätzle assumed the roles of team leader, project manager, and project lead, overseeing the planning, coordination, and monitoring of all project activities. Her leadership acumen, organizational skills, and strategic vision were pivotal in guiding the team through each project phase, ensuring that tasks were completed within the stipulated timeframes and budgets. Her proactive approach to problem-solving, clear communication, and commitment to fostering team cohesion were essential in cultivating a positive and collaborative work environment.

Documentation and Repository Management In addition to her leadership responsibilities, Luna was charged with maintaining comprehensive project documentation and managing the GitHub repository. Her meticulous attention to

detail and structured approach to documentation, coupled with her proficiency in version control systems, were instrumental in preserving accurate records of the project's progress, results, and deliverables. This rigorous documentation process not only facilitated effective knowledge sharing but also promoted transparency and ensured that all project-related information was readily accessible to team members.

Backend Development Florian Prandstetter led the development of the server and backend infrastructure for both the AI Hub and the Code Extension. His technical expertise, problem-solving skills, and extensive software development experience were crucial in designing and implementing a robust backend architecture. This included the integration of AI models and the optimization of server performance. Florian's ability to work independently, his commitment to quality assurance, and his dedication to continuous improvement were key to delivering a backend system that met the rigorous requirements of the project.

Website Development Luna also directed the design and implementation of the AI Hub's website. Her creative vision, user-centric design approach, and proficiency in web development technologies were vital in creating an engaging and intuitive user interface. Her emphasis on user experience, coupled with a strong commitment to accessibility and usability standards, ensured that the website effectively addressed the needs of its target audience.

Working with GitHub The team utilized GitHub as the version control system to manage the project's source code, documentation, and related artifacts. GitHub's collaborative features—such as branching, merging, and pull requests enabled effective code management, review, and integration among team members. By leveraging GitHub's distributed version control capabilities, the team ensured that all modifications to the codebase were systematically tracked, documented, and synchronized, thereby facilitating seamless collaboration and efficient project development.

14.3. Lessons Learned

Throughout the course of the project, the research team encountered various challenges and obstacles that provided valuable learning opportunities. The following section outlines the key lessons learned from the project:

- **Effective Communication** Clear and open communication is essential for successful teamwork. Regular updates, feedback sessions, and discussions help align team members, foster collaboration, and ensure that everyone is on the same page.
- **Time Management** Efficient time management is crucial for meeting project deadlines and milestones. Prioritizing tasks, setting realistic timelines, and monitoring progress are essential for effective project planning and execution.
- **Adaptability and Flexibility** Flexibility and adaptability are essential qualities for navigating unforeseen challenges and changes. Being open to new ideas, approaches, and solutions enables teams to respond effectively to evolving project requirements and constraints.
- **Continuous Learning** Embracing a growth mindset and a commitment to continuous learning are key to personal and professional development. Seeking feedback, acquiring new skills, and expanding one's knowledge base are essential for overcoming obstacles and achieving success in complex projects.

15. Outlook

Author: Luna Schätzle

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.

15.1. 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.*Computer use (beta) - Anthropic* (n.d.)

15.1.1. 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.

15.1.2. AI in Software Development

In the realm of software development, AI technologies are revolutionizing the way code is written, tested, and deployed. AI-powered tools such as code completion engines, bug detection systems, and automated testing frameworks are streamlining the development process and improving code quality. Additionally, AI is enabling the creation of self-healing software systems that can detect and resolve issues autonomously, reducing the need for manual intervention.

15.1.3. Challenges and Opportunities

Despite its considerable potential, Artificial Intelligence presents a range of formidable challenges. Ethical considerations—including bias, privacy, and accountability—must be rigorously addressed to ensure that AI technologies are developed and deployed responsibly. Furthermore, there is an urgent demand for enhanced transparency and explainability in AI systems, particularly in high-stakes domains such as healthcare and finance.

Another critical issue pertains to copyright and data ownership. AI systems frequently leverage extensive online datasets to generate new content, which may not be subject to the ownership rights of the original creators. This practice raises intricate questions regarding intellectual property rights and the ethical utilization of data.

Numerous experts and investors in the AI field have expressed substantial concern over the current absence of comprehensive legislative regulation. This regulatory gap could lead to the deployment of AI in ways that are not aligned with societal benefits. Some advocates propose the establishment of a global regulatory framework, comparable to international treaties governing nuclear weapons, to

ensure that AI is employed in a manner that serves the public interest and is strictly confined to approved domains.

?

15.2. 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.

15.3. 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.

15.4. 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.

15.5. Further development of the VS Code extension

The VS Code Extension still has room for improvement in terms of functionality and performance. Also the user interface could be more user-friendly and intuitive. To further develop the extension, the following aspects should be considered:

- **Enhanced AI Capabilities:** Integrate additional AI models and services to provide users with a wider range of tools and resources.
- **Code completion:** Implement code completion functionality to assist users in writing code more efficiently.
- **Improved User Experience:** Enhance the user interface and user experience to make the extension more intuitive and user-friendly.
- **Optimized Performance:** Optimize the extension's performance to ensure fast response times and seamless integration with VS Code.
- **Enhanced Error Handling:** Implement robust error handling mechanisms to provide users with clear feedback and guidance in case of errors.

15.6. Further Optimization of the Server

To further optimize the server, there are a few things that need to be addressed.

- **Scalability:** The server should be able to handle a large number of concurrent requests without compromising performance.
- **Security:** Implement robust security measures to protect user data and prevent unauthorized access.
- **Reliability:** Ensure that the server is reliable and stable, with minimal downtime and disruptions.
- **Hardware:** The server is currently hosted on dated hardware, which is a potential bottleneck. Upgrading the hardware could significantly improve performance.

These changes would drasitcally improve the performance and security of the server, so it could be easily implemented in larger scale projects.

15.7. Conclusion

The future of AI in Software development and education is promising, with numerous opportunities for innovation and growth. By leveraging cutting-edge technologies and embracing open-source principles, we can drive advancements in AI and empower students and professionals to excel in this dynamic field. The ongoing development of the Flask server, Student AI Hub, and VS Code extension represents a significant step toward realizing this vision, with the potential to revolutionize how AI is integrated into industry and education. By continuing to refine and expand these initiatives, we can create a more inclusive, collaborative, and accessible AI ecosystem that benefits learners, developers, and organizations worldwide.

Appendix

Appendix A.

Glossary

This section provides definitions and explanations for all the terms used throughout this document.

API (Application Programming Interface) A set of protocols, tools, and definitions that allow different software applications to communicate with each other.

AI (Artificial Intelligence) A branch of computer science focused on building systems that can perform tasks requiring human intelligence, such as learning, reasoning, and problem-solving.

IDE (Integrated Development Environment) A software application that provides comprehensive facilities for software development, including a code editor, compiler, debugger, and automation tools.

OS (Operating System) A system software that manages computer hardware, software resources, and provides services for computer programs.

VS Code (Visual Studio Code) A lightweight yet powerful source code editor developed by Microsoft, supporting multiple programming languages and extensions.

REST (Representational State Transfer) An architectural style for designing networked applications, which relies on stateless communication and standard HTTP methods.

HTTP (Hypertext Transfer Protocol) A protocol used for communication between web browsers and servers, enabling the retrieval of hypertext documents on the internet.

JSON (JavaScript Object Notation) A lightweight data interchange format that is easy for humans to read and write and easy for machines to parse and generate.

XML (Extensible Markup Language) A flexible text format used to store and transport data, often used in web services and configuration files.

HTML (Hypertext Markup Language) The standard markup language used for creating web pages and web applications.

CSS (Cascading Style Sheets) A stylesheet language used to control the presentation and layout of web documents.

JS (JavaScript) A programming language commonly used in web development to create interactive and dynamic web pages.

TS (TypeScript) A strongly typed programming language that builds on JavaScript, adding static typing for improved maintainability.

SQL (Structured Query Language) A domain-specific language used for managing and manipulating relational databases.

EU (European Union) A political and economic union of 27 European countries that are located primarily in Europe.

Appendix B.

Time Protocol

B.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; Research	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
15.09.2024	11:30:00	12:30:00	01:00:00	Planning; Research	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
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; Research	Addressing Ollama from Python | GitHub repository created	Luna
23.09.2024	16:00:00	16:40:00	00:40:00	Software Development	Testing various Python scripts	Luna

Continued on next page

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
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

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

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

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 Ollama	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

Continued on next page

Date	Start	End	Hours	Tasks	Details	Persons
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
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

Continued on next page

Date	Start	End	Hours	Tasks	Details	Persons
10.02.2025	10:00:00	12:30:00	02:30:00	Writing	Discussion, structure of the thesis, division, general writing	All (oL)
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

Continued on next page

Date	Start	End	Hours	Tasks	Details	Persons
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
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
04.03.2025	7:30:00	8:30:00	01:00:00	Writing	Flask Service	Luna
04.03.2025	15:15:00	16:15:00	01:00:00	Writing	Flask Service	Luna
06.03.2025	11:30:00	13:30:00	02:00:00	Writing	Flask Service Utility functions	Luna
06.03.2025	10:00:00	11:30:00	01:30:00	Writing	KI use and Aknowlagement	Luna
10.03.2025	09:00:00	10:00:00	01:00:00	Writing	Conclusion	Luna
16.03.2025	08:00:00	09:00:00	01:00:00	Writing	Smaller fixes and improvements	Luna

Continued on next page

Date	Start	End	Hours	Tasks	Details	Persons
18.03.2025	14:00:00	15:00:00	01:00:00	Organization	Group meeting: about printing and binding	All
18.03.2025	17:30:00	21:00:00	03:30:00	Writing	Corrections and improvements	Luna

B.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
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

Continued on next page

Date	Start	End	Hours	Tasks	Details	Persons
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
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

Continued on next page

Date	Start	End	Hours	Tasks	Details	Persons
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)
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
10.03.2025	09:40:00	11:35:00	01:55:00	Writing	Operating Systems in Economics	Flo

Continued on next page

Date	Start	End	Hours	Tasks	Details	Persons
11.03.2025	12:30:00	13:40:00	01:10:00	Writing	Operating Systems in Economics	Flo
12.03.2025	11:20:00	12:30:00	01:10:00	Writing	Operating Systems in Economics	Flo
12.03.2025	17:00:00	19:00:00	02:00:00	Writing	Operating Systems in Economics	Flo
13.03.2025	08:00:00	08:40:00	00:40:00	Writing	Corrections	Flo
14.03.2025	22:30:00	22:43:00	00:13:00	Writing	Corrections	Flo
15.03.2025	15:42:00	15:50:00	02:50:00	Software Development	Extension	Flo
15.03.2025	15:50:00	17:50:00	02:00:00	Writing	VS Code Extension	Flo
16.03.2025	14:50:00	16:50:00	02:00:00	Writing	Introduction / Conclusion	Flo
17.03.2025	08:00:00	08:27:00	00:27:00	Writing	Structure	Flo
18.03.2025	19:30:00	23:32:00	04:02:00	Writing	AI Economics / Corrections	Flo
19.03.2025	13:00:00	15:00:00	02:30:00	Writing	AI Economics	Flo
23.03.2025	13:00:00	16:10:00	03:20:00	Writing	Corrections	Flo
24.03.2025	13:00:00	16:00:00	03:00:00	Writing	Finalization	Flo

List of Tables

List of Figures

Listings

Bibliography

(2022). T4.
URL: *https://www.t4.ai/industry/server-operating-system-market-share*

10 biggest advantages of open-source software (2022). [Online; accessed 2025-01-28].
URL: *https://www.rocket.chat/blog/open-source-software-advantages*

A Comprehensive Guide to Ollama - Cohorte Projects (n.d.). [Online; accessed 2025-02-14].
URL: *https://www.cohorte.co/blog/a-comprehensive-guide-to-ollama*

AI Act | Shaping Europe's digital future (2025). [Online; accessed 2025-03-24].
URL: *https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai*

akash1295 (2025), 'What is an operating system'. [Online; accessed 2025-02-26].
URL: *https://www.geeksforgeeks.org/types-of-operating-systems/*

Ankush (2024), 'What is ollama? everything important you should know'. [Online; accessed 2025-01-13].
URL: *https://itsfoss.com/ollama/*

Aptiv (2022), 'Aptiv completes the acquisition of wind river from tpg'. [Online; accessed 2025-03-12].
URL: *https://www.aptiv.com/en/newsroom/article/aptiv-completes-the-acquisition-of-wind-river-from-tpg*

Artificial Intelligence in Society (2021), https://www.oecd.org/going-digital/ai-in-society.htm. Accessed: 2025-03-19.

Axios Docs (2025). Accessed: 2025-02-14.
URL: *https://axios-http.com/docs/intro*

Bansal, S. & Aggarwal, C. (2025), 'Textstat: Python library for text statistics'. Zugriff am 13. Februar 2025.
URL: *https://pypi.org/project/textstat/*

Bird, S., Klein, E. & Loper, E. (2025), 'Natural language toolkit (nltk)'. Zugriff am 13. Februar 2025.
URL: *https://www.nltk.org/*

Brownlee, J. (2017), 'A gentle introduction to calculating the bleu score for text in python - machinelearningmastery.com'. [Online; accessed 2025-03-24].
URL: *https://machinelearningmastery.com/calculate-bleu-score-for-text-python/*

businesswire (2022), 'Operating systems global market to reach $48.18 billion by 2026'. [*Online*; *accessed* $2025 - 03 - 11$].
URL:*https://www.businesswire.com/news/home/20220825005370/en/Operating-Systems-Global-Market-to-Rea*

Cao, C., Wang, F., Lindley, L. & Wang, Z. (2024), 'Managing linux servers with llm-based ai agents: An empirical evaluation with gpt4', *Machine Learning with Applications* **17**, 100570.

Computer use (beta) - Anthropic (n.d.). [Online; accessed 2025-03-03].
URL: *https://docs.anthropic.com/en/docs/agents-and-tools/computer-use*

Contributors, W. (2025*a*), 'Apple'. [Online; accessed 2025-03-12].
URL: *https://de.wikipedia.org/wiki/Apple*

Contributors, W. (2025*b*), 'Microsoft'. [Online; accessed 2025-03-12].
URL: *https://de.wikipedia.org/wiki/Microsoft*

Contributors, W. (n.d.*a*), 'Graphics processing unit'. [Online; accessed 2025-03-18].
URL: *https://en.wikipedia.org/wiki/Graphics*$_p$*rocessing*$_u$*nit*

Contributors, W. (n.d.*b*), 'Power supply'. [Online; accessed 2025-03-18].
URL: *https://en.wikipedia.org/wiki/Power*$_s$*upply*

Contributors, W. (n.d.*c*), 'Random-access memory'. [Online; accessed 2025-03-18].
URL: *https://de.wikipedia.org/wiki/Random-Access*$_M$*emory*

CSS Introduction (n.d.). [Online; accessed 2025-03-24].
URL: *https://www.w3schools.com/css/css*$_i$*ntro.asp*

Data, I. & Team, A. (2023), 'Shedding light on ai bias with real world examples'. [Online; accessed 2025-03-18].
URL: *https://www.ibm.com/think/topics/shedding-light-on-ai-bias-with-real-world-examples*

Datenschutzgesetz (DSG) (1999), https://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=10001597. Accessed: 2025-03-18.

Desktop Operating System Market Share Worldwide | Statcounter Global Stats (n.d.).

Dharmadhikari, S. (2025), 'embedded operating systems market report'. [Online; accessed 2025-03-12].
URL: *https://www.cognitivemarketresearch.com/embedded-operating-systems-market-report*

DigitalDefynd, T. (2024), '10 ways ai is being used in economics', https://digitaldefynd.com/IQ/ai-use-in-economics/. Accessed: 2025-03-18.

Dizikes, P. (2024), 'What do we know about the economics of ai?', https://news.mit.edu/2024/what-do-we-know-about-economics-ai-1206. Accessed: 2025-03-18.

Docker_Flask(n.d.).[Online; accessed2025 − 02 − 13].
URL:*https://www.freecodecamp.org/news/how-to-dockerize-a-flask-app/*

Doocker_Compose(n.d.).[Online; accessed2025 − 02 − 13].
URL:*https://docs.docker.com/compose/*

DTeK (2023), '(51) diy: Cyberdeck multi-function backup computer - youtube'. [Online; accessed 2025-03-03].
URL: *https://www.youtube.com/watch?v=88tgZ6Xq3Jolist=PLoyOhsqi57GmrLomvDFnLXHn17OD3onLsindex=1t=1199s*

Farrell, R. (2025), 'The impact of ai on job roles, workforce, and employment: What you need to know'. [Online; accessed 2025-03-18].
URL: *https://www.innopharmaeducation.com/blog/the-impact-of-ai-on-job-roles-workforce-and-employment-what-you-need-to-know*

Firebase Features Explained in Detail (2025 Update) (n.d.). [Online; accessed 2025-02-16].
URL: *https://www.lido.app/firebase/firebase-features*

Forbes Technology Council (2024), 'Misconceptions about open source solutions clarified by tech experts'. Accessed: 2024-12-04.
URL: *https://www.forbes.com/councils/forbestechcouncil/2024/10/09/misconceptions-about-open-source-solutions-clarified-by-tech-experts/*

Fortis, S. (2024), 'Openai hit with privacy complaint in austria, potential eu law breach'. [Online; accessed 2025-02-07].
URL: *https://cointelegraph.com/news/openai-privacy-complaint-austria-potential-eu-law-breach*

fortune business insights (2025), 'Server operating system market'. [Online; accessed 2025-03-12].
URL: *https://www.fortunebusinessinsights.com/server-operating-system-market-106601*

Foundation, P. S. (2025), 'json — json encoder and decoder'. Accessed: 2025-02-14.
URL: *https://docs.python.org/3/library/json.html*

Foy, P. (2024), 'Understanding tokens & context windows'. [Online; accessed 2025-02-07].
URL: *https://blog.mlq.ai/tokens-context-window-llms/*

Galope, S. (2024), 'Open source case studies: Success stories of impactful projects'. [Online; accessed 2025-03-21].
URL: *https://www.samgalope.dev/2024/11/08/open-source-case-studies-success-stories-of-impactful-projects/*

GeeksforGeeks (2020), 'Introduction to seaborn - python - geeksforgeeks'. [Online; accessed 2025-03-24].
URL: *https://www.geeksforgeeks.org/introduction-to-seaborn-python/*

GeeksforGeeks (2024), 'Understanding bleu and rouge score for nlp evaluation - geeksforgeeks'. [Online; accessed 2025-03-24].
URL: *https://www.geeksforgeeks.org/understanding-bleu-and-rouge-score-for-nlp-evaluation/*

Global, A. (2025), 'The future of customer service with ai chatbots | ada'. [Online; accessed 2025-03-19].
URL: *https://www.adaglobal.com/resources/insights/the-future-of-customer-service-with-ai-chatbots*

Gul, S. (2024), 'Transformers library for generative ai — the basics | by sercan gul | data scientist | datascientisttx | medium'. [Online; accessed 2025-03-24].
URL: *https://sercangl.medium.com/transformers-library-for-generative-ai-the-basics-7d5dc3a0aafe*

Hat, R. (2024), 'Red hat enterprise linux ai'. [Online; accessed 2025-03-01].
URL: *https://www.redhat.com/de/technologies/linux-platforms/enterprise-linux/ai*

Helms, J. (2023), '10 risks of open-source software | connectwise'. [Online; accessed 2025-02-18].
URL: *https://www.connectwise.com/blog/cybersecurity/open-source-software-risks*

Hendrickson, M., Magoulas, R. & O'Reilly, T. (2012), *Economic Impact of Open Source on Small Business: A Case Study*, O'Reilly Media.
URL: *https://www.oreilly.com/library/view/economic-impact-of/9781449343408/*

Hermansen, A. (2024), 'How to navigate the complexity of open source license compliance'. [Online; accessed 2025-03-24].
URL: *https://www.linuxfoundation.org/blog/how-to-navigate-the-complexity-of-open-source-license-compliance*

Home | LibreOffice - Free and private office suite - Based on OpenOffice - Compatible with Microsoft (n.d.). [Online; accessed 2025-03-21].
URL: *https://www.libreoffice.org/*

Houbrechts, M. (2024), 'Using ai for data analysis: The ultimate guide', https://www-luzmo-com.translate.goog/blog/ai-data-analysis?_x_tr_sl=en&_x_tr_tl=de&_x_tr_hl=de&_x_tr_pto=sc. Accessed: 2025-03-18.

How does open-source benefit startups? (n.d.). [Online; accessed 2025-03-24].
URL: *https://milvus.io/ai-quick-reference/how-does-opensource-benefit-startups*

HTML - Wikipedia (2001). [Online; accessed 2025-01-23].
URL: *https://en.wikipedia.org/wiki/HTML*

Hunter, J. D. & the Matplotlib Development Team (2025), 'Matplotlib: Python plotting library'. Accessed: 2025-02-14.
URL: *https://matplotlib.org/*

IBM (2021), 'What is intelligent automation?', https://www.ibm.com/think/topics/intelligent-automation. Accessed: 2025-03-18.

IBM (2024*a*), 'Was ist computer-networking?'. [Online; accessed 2025-03-18].
URL: *https://www.ibm.com/de-de/topics/networking*

IBM (2024*b*), 'What is containerization?'. [Online; accessed 2025-03-18].
URL: *https://www.ibm.com/think/topics/containerization*

IBM (2024*c*), 'What is data storage?'. [Online; accessed 2025-03-18].
URL: *https://www.ibm.com/think/topics/data-storage*

IBM (n.d.*a*), 'What are large language models (llms)? | ibm'. [Online; accessed 2025-01-21].
URL: *https://www.ibm.com/think/topics/large-language-models*

IBM (n.d.*b*), 'What is ai in supply chain? | ibm'. [Online; accessed 2025-03-19].
URL: *https://www.ibm.com/think/topics/ai-supply-chain*

Initiative, O. S. (2007), 'The open source definition', https://opensource.org/osd. Accessed: 2024-12-02.

Intel (n.d.), 'Intel® core™ i5-8600k processor'. [Online; accessed 2025-03-18].
URL: *https://www.intel.com/content/www/us/en/products/sku/126685/intel-core-i58600k-processor-9m-cache-up-to-4-30-ghz/specifications.html*

Introducing data residency in Europe | OpenAI (n.d.). [Online; accessed 2025-02-07].
URL: *https://openai.com/index/introducing-data-residency-in-europe/?utm*$_s$*ource = chatgpt.com*

Introduction to Markdown — Write the Docs (n.d.). [Online; accessed 2025-02-17].
URL: *https://www.writethedocs.org/guide/writing/markdown/*

Introduction to Python (n.d.). [Online; accessed 2025-03-24].
URL: *https://www.w3schools.com/python/python*$_i$*ntro.asp*

Introduction | Vue.js (n.d.). [Online; accessed 2025-03-24].
URL: *https://vuejs.org/guide/introduction*

JavaScript Tutorial (n.d.). [Online; accessed 2025-03-24].
URL: *https://www.w3schools.com/js/*

Joos, T. (2023), 'Die 12 besten remote administration tools'. [Online; accessed 2025-03-18].
URL: *https://www.ip-insider.de/die-12-besten-remote-administration-tools-a-a5b8f65d154f5a5c3953dde613eb3e16/*

Kirvan, P. (n.d.), 'Motherboard'. [Online; accessed 2025-03-18].
URL: *https://www.computerweekly.com/de/definition/Motherboard*

LaCroix, J. (2022), *Mastering Ubuntu Server: Explore the versatile, powerful Linux Server distribution Ubuntu 22.04 with this comprehensive guide*, Packt Publishing Ltd.

Larsson, S. & Heintz, F. (2020), 'Transparency in artificial intelligence', *Internet policy review* **9**(2), 1–16.

Linus Torvalds - Software is like sex: it's better when... (1999). [Online; accessed 2025-03-23].
URL: *https://www.brainyquote.com/quotes/linus*$_t$*orvalds*$_1$35583

Liu, H., Li, C., Li, Y. & Lee, Y. J. (2023), 'Improved baselines with visual instruction tuning'.

manjeetks007 (2024), 'What is an operating system'. [Online; accessed 2025-02-26].
URL: *https://www.geeksforgeeks.org/what-is-an-operating-system/*

Mathpati, N. (2023), 'Open-source software overview: Benefits, risks, & best practices'. [Online; accessed 2025-02-18].
URL: *https://www.cobalt.io/blog/risks-of-open-source-software*

McKinney, W. & the Pandas Development Team (2025), 'Pandas: Python data analysis library'. Accessed: 2025-02-14.
URL: *https://pandas.pydata.org/*

Munteanu, A. (2024), 'Top 5 reasons to use ubuntu for your ai/ml projects'. [Online; accessed 2025-02-27].
URL: *https://ubuntu.com/blog/ubuntu-ai-ml-projects*

Naber, D. & andere (2025), 'Languagetool: A multilingual grammar and style checker'. Zugriff am 13. Februar 2025.
URL: *https://pypi.org/project/language-tool-python/*

NVIDIA, T. P. . (n.d.), 'Nvidia geforce rtx 2060'. [Online; accessed 2025-03-18].
URL: *https://www.techpowerup.com/gpu-specs/geforce-rtx-2060.c3310*

Office, C. B. (2024), 'Artificial intelligence and its potential effects on the economy and the federal budget', https://www.cbo.gov/publication/61147. Accessed: 2025-03-18.

Oliphant, T. & the NumPy Development Team (2025), 'Numpy: The fundamental package for numerical computation'. Accessed: 2025-02-14.
URL: *https://numpy.org/*

Ollama (n.d.*a*). [Online; accessed 2025-02-07].
URL: *https://ollama.com/search*

Ollama (n.d.*b*). [Online; accessed 2025-01-20].
URL: *https://ollama.com/search*

ollama/ollama-python: Ollama Python library (n.d.). [Online; accessed 2025-01-21].
URL: *https://github.com/ollama/ollama-python*

Openemr Software Review - Ambula Healthcare (n.d.). [Online; accessed 2025-03-24].
URL: *https://www.ambula.io/openemr-software-review/*

Opensource.com (n.d.), 'What is open source? | opensource.com'. [Online; accessed 2025-03-23].
URL: *https://opensource.com/resources/what-open-source*

Open Source Guide für Österreich | netidee (n.d.). [Online; accessed 2025-02-26].
URL: *https://www.netidee.at/opensource-guide-oesterreich*

Operating System Market Share Worldwide | Statcounter Global Stats (n.d.).

Otten, N. V. (2024), 'Rouge metric in nlp: Complete guide & how to tutorial'. [Online; accessed 2025-03-24].
URL: *https://spotintelligence.com/2024/08/12/rouge-metric-in-nlp/*

Overview - OpenAI API (n.d.*a*). [Online; accessed 2025-01-13].
URL: *https://platform.openai.com/docs/overview*

Overview - OpenAI API (n.d.*b*). [Online; accessed 2025-02-07].
URL: *https://platform.openai.com/docs/overview*

Pontikis, C. (2013), 'Five reasons to use debian as a server'. [Online; accessed 2025-03-01].
URL: *https://www.pontikis.net/blog/five-reasons-to-use-debian-as-a-server*

$pp_pankaj(2025), 'Kernelinoperatingsystem', .$ Accessed: 2025-03-18.

Pricing | OpenAI (n.d.). [Online; accessed 2025-02-17].
URL: *https://openai.com/api/pricing/*

psutil · PyPI (2024). [Online; accessed 2025-01-21].
URL: *https://pypi.org/project/psutil/*

Rahmatallah, D. (n.d.), 'Absolute guide to software licensing types | licensing models'. [Online; accessed 2025-02-18].
URL: *https://cpl.thalesgroup.com/software-monetization/software-licensing-models-guide*

Red Hat – Wikipedia (n.d.).

Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (2016), https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32016R0679. Accessed: 2025-03-18.

Regulation - EU - 2024/1689 - EN - EUR-Lex (2024). [Online; accessed 2025-03-19].