



FACULTY OF APPLIED SCIENCES  
UNIVERSITY  
OF WEST BOHEMIA

DEPARTMENT OF  
COMPUTER SCIENCE  
AND ENGINEERING



## Bachelor's Thesis

# Artificial intelligence for facilitating software development

Andrei Akhramchuk





**FACULTY OF APPLIED SCIENCES  
UNIVERSITY  
OF WEST BOHEMIA**

**DEPARTMENT OF  
COMPUTER SCIENCE  
AND ENGINEERING**

## **Bachelor's Thesis**

# **Artificial intelligence for facilitating software development**

Andrei Akhramchuk

### **Thesis advisor**

Doc. Ing. Roman Mouček, Ph.D.

© 2024 Andrei Akhramchuk.

All rights reserved. No part of this document may be reproduced or transmitted in any form by any means, electronic or mechanical including photocopying, recording or by any information storage and retrieval system, without permission from the copyright holder(s) in writing.

**Citation in the bibliography/reference list:**

AKHRAMCHUK, Andrei. *Artificial intelligence for facilitating software development*. Pilsen, Czech Republic, 2024. Bachelor's Thesis. University of West Bohemia, Faculty of Applied Sciences, Department of Computer Science and Engineering. Thesis advisor Doc. Ing. Roman Mouček, Ph.D.

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd  
Akademický rok: 2023/2024

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: **Andrei AKHRAMCHUK**  
Osobní číslo: **A21B0074P**  
Studijní program: **B0613A140015 Informatika a výpočetní technika**  
Specializace: **Informatika**  
Téma práce: **Umělá inteligence pro usnadnění softwarového vývoje**  
Zadávající katedra: **Katedra informatiky a výpočetní techniky**

## Zásady pro vypracování

1. Seznamte se s technologiemi a programovacími jazyky použitými v produktech firmy Eurosoftware s.r.o. (Java, XML, domain specific languages).
2. Seznamte se s přístupy, jak se v zákaznických projektech rozšiřují produkty firmy Eurosoftware s.r.o.
3. Prostudujte přístupy a nástroje umělé inteligence, které by usnadnily vývoj ve firmě Eurosoftware.
4. Navrhněte a realizujte řešení, které bude podporovat vývojové vzory použité v Eurosoftware.
5. Efektivnost výsledného řešení otestujte na praktických případech použití a zhodnoťte.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**  
Rozsah grafických prací: **dle potřeby**  
Forma zpracování bakalářské práce: **tištěná/elektronická**  
Jazyk zpracování: **Angličtina**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Ing. Radek Hoštička**  
Eurosoftware s.r.o.  
Konzultant bakalářské práce: **Doc. Ing. Roman Mouček, Ph.D.**  
Katedra informatiky a výpočetní techniky  
Datum zadání bakalářské práce: **2. října 2023**  
Termín odevzdání bakalářské práce: **2. května 2024**

L.S.

---

**Doc. Ing. Miloš Železný, Ph.D.**  
děkan

---

**Doc. Ing. Přemysl Brada, MSc., Ph.D.**  
vedoucí katedry

V Plzni dne 25. října 2023

# Declaration

I hereby declare that this Bachelor's Thesis is completely my own work and that I used only the cited sources, literature, and other resources. This thesis has not been used to obtain another or the same academic degree.

I acknowledge that my thesis is subject to the rights and obligations arising from Act No. 121/2000 Coll., the Copyright Act as amended, in particular the fact that the University of West Bohemia has the right to conclude a licence agreement for the use of this thesis as a school work pursuant to Section 60(1) of the Copyright Act.

In Pilsen, on 14 February 2024

.....  
Andrei Akhramchuk

The names of products, technologies, services, applications, companies, etc. used in the text may be trademarks or registered trademarks of their respective owners.

## Abstract

This bachelor thesis examines the technologies and programming languages used by Eurosoftware. It provides a comprehensive analysis and practical evaluation of publicly available Artificial Intelligence tools that can facilitate software development processes, focusing on those that offer potential benefits to Eurosoftware and similar organisations. The study includes an evaluation of various AI tools to determine their effectiveness. In addition, the thesis details the training of a Eurosoftware-oriented neural network model, which is oriented to generalise architectural data from the company's enterprise applications. It includes the creation of a software tool to assist in the preparation of the data necessary to train this model, as well as the training and subsequent evaluation of the model itself.

## Abstrakt

Tato bakalářská práce se zabývá zkoumáním technologií a programovacích jazyků používaných společností Eurosoftware. Poskytuje komplexní analýzu a praktické zhodnocení veřejně dostupných nástrojů umělé inteligence, které mohou usnadnit procesy vývoje softwaru, se zaměřením na ty, které nabízejí potenciální výhody pro společnost Eurosoftware a podobné organizace. Součástí výzkumu je hodnocení různých nástrojů umělé inteligence s cílem určit jejich účinnost. Kromě toho práce podrobně popisuje trénink modelu neuronové sítě orientovaného na společnost Eurosoftware, který je zaměřen na zobecnění architektonických dat z podnikových aplikací společnosti. Zahrnuje vytvoření softwarového nástroje, který pomáhá při přípravě dat potřebných k trénování tohoto modelu, a také samotné trénování a následné vyhodnocení modelu.

## Keywords

Eurosoftware • artificial intelligence • AI • Chat GPT • TabNine • GitHub Copilot • company-oriented NLP model

## Acknowledgement

I would like to express my sincere gratitude to Ing. Radek Hoštička, my bachelor thesis supervisor, for his valuable advice and guidance, and for helping me to access a server to conduct all my research. I would also like to acknowledge Doc. Ing. Roman Mouček, Ph.D., for his guidance on the proper structure of this thesis, his quick replies to my emails, and his overall support. I would also like to thank Eurosoftware for their kind support. Special thanks to Mr Pascal Singer for his expert advice on data validation and his invaluable guidance at the start.



# Contents

<b>1</b>	<b>Introduction</b>	<b>6</b>
<b>2</b>	<b>Eurosoftware s.r.o.</b>	<b>7</b>
2.1	Company introduction . . . . .	7
2.2	Basic technologies . . . . .	8
2.2.1	Confluence . . . . .	8
2.2.2	Jira . . . . .	9
2.2.3	Anna . . . . .	9
2.3	Development Technologies . . . . .	10
2.3.1	Git Lab . . . . .	10
2.3.2	Eclipse . . . . .	10
2.3.3	JetBrains Products . . . . .	11
2.3.4	SoapUI and Postman . . . . .	11
2.3.5	Dbeaver . . . . .	12
2.4	Programming Languages . . . . .	12
2.4.1	Java . . . . .	13
2.4.2	XML . . . . .	14
2.4.3	Web Development Languages at Eurosoftware . . . . .	15
2.4.4	ProMo . . . . .	16
2.4.5	ProMoJ . . . . .	16
2.4.6	Domain Specific Language (DO DSL) . . . . .	17
2.5	Architecture . . . . .	17
2.5.1	Core Architectural Solutions . . . . .	18
2.5.2	OmniPoint Of Sale (POS) Architecture . . . . .	19
<b>3</b>	<b>Possible development facilitations</b>	<b>21</b>
3.1	Universal Improvement Opportunities . . . . .	21
3.1.1	Structured Query Language (SQL) Operations . . . . .	21
3.1.2	Lambda Expressions . . . . .	21
3.1.3	Advanced Code Refactoring . . . . .	22
3.1.4	Cross-Language Code Translation . . . . .	22

3.1.5	Code Explanation and Interpretation: . . . . .	22
3.2	Facilitation Opportunities for Eurosoftware . . . . .	23
3.2.1	Confluence . . . . .	23
3.2.2	Domain Specific Languages . . . . .	23
3.2.3	Custom Extensible Markup Language (XML) and Cascading Style Sheets (CSS) . . . . .	24
3.2.4	Integrated Coding Advisor Plugin . . . . .	24
<b>4</b>	<b>Leveraging Existing Tools for Enhanced Development</b>	<b>25</b>
4.1	Exploring ChatGPT: A Comparative Analysis . . . . .	25
4.1.1	ChatGPT 3.5 . . . . .	26
4.1.2	ChatGPT 4 . . . . .	26
4.1.3	Conclusion . . . . .	27
4.2	Exploring Tabnine: Artificial Intelligence (AI)-Powered Code Assis- tance . . . . .	27
4.2.1	Cloud-Based Model . . . . .	27
4.2.2	Local Processing Model . . . . .	28
4.2.3	Hybrid Model Adaptability . . . . .	28
4.2.4	Conclusion . . . . .	28
4.3	Exploring GitHub Copilot: AI-Assisted Coding . . . . .	28
4.3.1	Adaptive Learning . . . . .	28
4.3.2	Security Considerations . . . . .	29
4.3.3	Personal Evaluation . . . . .	29
4.3.4	Conclusion . . . . .	29
4.4	Custom Natural Language Processing (NLP) Model: Eurosoftware’s Approach . . . . .	29
4.4.1	Potential of NVIDIA NeMo in Custom AI Development . .	30
4.4.2	Llama 2 7B Model . . . . .	30
4.4.3	Megatron-GPT 5B Model . . . . .	31
<b>5</b>	<b>Practical Application of General Tools</b>	<b>32</b>
5.1	ChatGPT 3.5: Free-to-use conversational AI . . . . .	32
5.1.1	Explaining SQL Scripts . . . . .	32
5.1.2	Optimizing SQL Queries . . . . .	33
5.1.3	Explaining Lambda expression . . . . .	33
5.1.4	Optimizing Lambda expression . . . . .	34
5.1.5	Lambda expression generation . . . . .	35
5.1.6	Conclusion . . . . .	35
5.2	ChatGPT 4: Paid conversational AI . . . . .	35
5.2.1	Code Interpretation with ChatGPT Version 4.0 (GPT-4) . .	36

5.2.2	Advanced Code Refactoring with GPT-4 . . . . .	36
5.2.3	Groovy to JShell Translation . . . . .	37
5.2.4	Java to Python Translation . . . . .	37
5.2.5	Extensible Hypertext Markup Language (XHTML) to Vue.js Translation . . . . .	38
5.2.6	Conclusion . . . . .	39
5.3	TabNine: Enhancing Development with AI . . . . .	39
5.3.1	Evaluating TabNine with TypeScript . . . . .	39
5.3.2	Evaluating TabNine with Vue.js Projects . . . . .	40
5.3.3	Conclusion . . . . .	41
5.4	GitHub Copilot: A Revolution in AI-Assisted Coding . . . . .	41
5.4.1	GitHub Copilot with TypeScript . . . . .	42
5.4.2	GitHub Copilot in Java Development . . . . .	42
5.4.3	Vue.js code generation with GitHub Copilot Chat . . . . .	43
5.4.4	GitHub Copilot Chat for TypeScript Development . . . . .	44
5.4.5	Conclusion . . . . .	44
5.5	Common Conclusion . . . . .	45
5.5.1	ChatGPT Version 3.5 (GPT-3.5) . . . . .	45
5.5.2	ChatGPT 4 . . . . .	46
5.5.3	TabNine . . . . .	46
5.5.4	GitHub Copilot . . . . .	47
<b>6</b>	<b>Custom AI Model for Enhanced Knowledge Management</b>	<b>48</b>
6.1	Enhancing Training Data Management with a Graphical User In- terface (GUI) Application . . . . .	48
6.1.1	User Interface Components . . . . .	49
6.1.2	Hotkeys in the Main Application . . . . .	51
6.1.3	Hotkeys in Modal Windows . . . . .	51
6.1.4	Data Validation and Management . . . . .	52
6.1.5	Handling Unsaved Changes . . . . .	53
6.1.6	Technical Details . . . . .	53
6.2	Training the Model: An Iterative Approach . . . . .	54
6.2.1	First Iteration: Exploratory Phase . . . . .	54
6.2.2	Second Iteration: Preparatory Steps for Future Training . . . . .	55
6.2.3	Third Iteration: Technical Implementation . . . . .	57
6.2.4	Fourth Iteration: Initial Model Training . . . . .	60
6.2.5	Fifth Iteration: Data Expansion and Model Refinement . . . . .	62
<b>7</b>	<b>Conclusion</b>	<b>64</b>

<b>A</b>	<b>List of Abbreviations</b>	<b>65</b>
<b>B</b>	<b>Attachments Structure</b>	<b>67</b>
B.1	Application and Libraries . . . . .	67
B.2	Input Data . . . . .	68
B.3	Results . . . . .	69
B.3.1	GPT-3.5 . . . . .	69
B.3.2	GPT-4 Results . . . . .	70
B.3.3	TabNine . . . . .	70
B.3.4	GitHub Copilot . . . . .	71
<b>C</b>	<b>Supplementary Materials</b>	<b>72</b>
C.1	Code Examples . . . . .	72
C.1.1	Example 1: XML Layout . . . . .	72
C.1.2	Example 2: ProMo Source Code . . . . .	73
C.1.3	Example 3: ProMoJ Source Code . . . . .	74
C.1.4	Example 4: Usage of DO DSL . . . . .	75
C.2	Detailed Model Outputs . . . . .	76
C.2.1	Fourth Iteration: Initial Model Training Outputs . . . . .	76
C.2.2	Fifth Iteration: Expanded Data Model Outputs . . . . .	77
C.3	User Manual: JSON Editor Application . . . . .	78
C.3.1	Setting Up Application on Windows . . . . .	78
C.3.2	Setting Up Application on Linux . . . . .	78
C.3.3	File Management . . . . .	79
C.3.4	Editing Entries . . . . .	79
C.3.5	Opening a File . . . . .	79
C.3.6	Adding and Editing Entries . . . . .	79
C.3.7	Saving Files . . . . .	79
C.4	User Manual: AI Training and Validation Scripts . . . . .	79
C.4.1	System Requirements . . . . .	79
C.4.2	Setting up the Environment . . . . .	80
C.4.3	Installing PyTorch . . . . .	80
C.4.4	Apex Installation . . . . .	80
C.4.5	Transformer Engine and NeMo Toolkit . . . . .	81
C.4.6	Docker Container Setup . . . . .	81
C.4.7	Setting Environment Variables . . . . .	81
C.4.8	Executing Training and Validation Scripts . . . . .	81
C.4.9	Logging and Monitoring . . . . .	83
	<b>Bibliography</b>	<b>84</b>

<b>List of Figures</b>	<b>86</b>
<b>List of Tables</b>	<b>87</b>
<b>List of Listings</b>	<b>88</b>

# Introduction

# 1

AI is at the forefront of today's rapidly advancing technology landscape, bringing a new era of innovation and efficiency. Staying abreast of these developments is not only beneficial for businesses, it is essential to survive and succeed. While it may be easier for individual developers to adopt and experiment with these cutting-edge technologies, larger organizations face a maze of challenges. The integration of AI into the established and often rigid infrastructures of these companies is a complex endeavour, fraught with hurdles but teeming with potential for transformative change.

The drive to stay current, improve operational efficiency and gain competitive advantage is driving the need to explore the potential of AI. However, the path to successful integration is not standardized; it requires a tailored approach that aligns with an organization's unique needs and strategic goals. The quest for tailored, impactful solutions stands out among the universal benefits of AI as a tool for simplifying and improving business processes.

The main objective of this bachelor thesis is to delve into the workings of Eurosoftware, exploring its operational aspects and the tools used for software development. The focus is on identifying how AI can simplify and enhance these processes, making the developers' work more efficient and effective.

Artificial intelligence-based solutions that meet Eurosoftware's unique needs involved in this process must be carefully selected and tested. This involves a detailed testing process within the company's working environment to see how the AI performs in real-life scenarios. It's about seeing if these AI tools can speed up the development process and simplify developers' tasks. The end goal is to find and recommend the best AI practices that will not only work for Eurosoftware but will advance the company by providing lessons on the practical application of AI in a corporate environment.

## 2.1 Company introduction

Eurosoftware is a subsidiary of GK Software SE, a technology company specializing in software and services for the operation of large retail stores. Headquartered in Schöneck/Vogtland, Germany, GK Software operates worldwide with approximately 1,230 employees in sixteen locations. Eurosoftware is located in Pilsen, Czech Republic, Katowice, Poland and Lviv, Ukraine.

Eurosoftware maintains an internal student lab where trainees, typically students, complete paid internships. In this lab, they contribute to the development of internal products that are used within the company. These projects provide practical experience as they are integral to the company's operations. However, they are not marketed externally. This hands-on approach allows interns to gain valuable skills and knowledge through meaningful projects. Alongside this initiative, Eurosoftware's primary focus remains on the development of POS software, which is a key aspect of their business.

Eurosoftware's approach to POS development is strategically divided into two core departments: 'Product Development' and 'Customer Solutions'. The 'Product Development' department focuses on basic, non-customized features of the software. This includes primary design and standard functionality. The 'Customer Solution' department, on the other hand, focuses on customizing the software for individual customers. This involves dedicated teams managed by team managers who allocate resources and personnel to specific projects, each representing a unique customer.

Eurosoftware distinguishes itself in the POS software industry by offering highly customizable solutions tailored to meet specific customer needs. Their software facilitates easy integration with various external services and allows for custom design options that enhance both functionality and convenience. Key features include the

ability for customers to seamlessly add new items or discounts directly into the system without additional programming. Overall, Eurosoftware provides a versatile and reliable POS system that demonstrates a strong commitment to delivering efficient and customer-focused solutions.

## 2.2 Basic technologies

In the rapidly evolving technology landscape, Eurosoftware uses a range of sophisticated tools to streamline operations, enhance project management and promote employee well-being. With a focus on the collaborative platforms and internal management systems that are critical to the company's success, this section provides an overview of the foundational technologies that facilitate these functions.

### 2.2.1 Confluence

The primary mechanism for storing and structuring vast quantities of information at Eurosoftware is the Confluence wiki system developed by Atlassian. This tool enables the company to manage its technological documentation within a secure repository. Access to this information is strictly confined to company employees, ensuring privacy and security.

Within Confluence, access rights are meticulously assigned to different levels of employees. Certain pages are designated as read-only, while others are editable. This stratification is critical for maintaining a proper hierarchy and preventing the loss of information due to errors by less experienced users. [Iwu24]

In Eurosoftware's 'Customer Solution' department, each project has its page on Confluence, which serves several purposes: to provide developers with critical information to quickly get started on the project, to archive documentation of existing code, and to present project concepts written by technical consultants. Each concept, once approved by the client, serves as a definitive guide for the developers, describing in detail the intended program logic.

Conversely, the approach to 'Product development' is different. Concepts are not client-specific but are designed with a broader scope to contribute to the fundamental aspects of the program that are critical to the overall software and will be used in all future projects. Through the systematic organization of information within Confluence, Eurosoftware ensures precise placement and secure accessibility of data, whether for specific customer projects or general product development,



thus protecting intellectual property and increasing collaboration efficiency.

### 2.2.2 Jira

Similar to Confluence, Jira, another tool developed by Atlassian, plays a key role at Eurosoftware in maintaining clarity and order. The main function of this tool is to create and manage tasks.

Jira makes possible not only the simple task creation but also the allocation of specific timeframes, the detailing of task descriptions and prioritisation of them. Furthermore, Jira includes a chat feature on each task page. This feature is necessary for communication between different departments of the company within the context of a particular task.

By tightly integrating with Confluence, Jira increases the efficiency of workflow management. It simplifies the process of linking a task in Jira to its more comprehensive description on a Confluence page. Beyond this connection, Jira allows automatic links to be set up between the task and Git. This feature ensures that any commit or merge request related to a task is automatically recorded in the comment section of the task. [Bru19]

### 2.2.3 Anna

Developed by Eurosoftware's student lab, Anna serves as a strategic addition to the company's toolkit, diverging from Jira's project management focus to focus on human resource planning and employee welfare. This tool is uniquely designed to handle the administrative and welfare aspects of employee management, offering a distinct set of functionalities.

Unlike Jira, which tracks tasks and projects, Anna streamlines employee-related administrative tasks. It manages work schedules, tracks attendance, facilitates leave requests, and provides a comprehensive view of human resource allocation. Anna also allows employees to register for company events, review work logs, and communicate health concerns, fostering a supportive work environment.

By delineating the roles of Anna and Jira, Eurosoftware achieves an effective balance between operational efficiency and employee satisfaction. Anna's focus on improving the internal operational aspects related to human resources ensures that

Eurosoftware not only excels in project management through Jira, but also prioritizes the well-being and effective management of its employees.

## 2.3 Development Technologies

Within Eurosoftware development, the range of technical tools goes beyond the working tools mentioned above. The company has prepared a set of tools that are essential for the daily work of every developer. These tools not only make the software development process much easier but also improve version control and facilitate the use of company-specific programming languages. These carefully selected tools enable greater efficiency and productivity, resulting in a higher standard of software products.

### 2.3.1 Git Lab

Within Eurosoftware, GitLab plays a central role as a robust platform for version control and collaborative software development. Hosted on Eurosoftware's own servers for enhanced security, this web-based tool allows developers to efficiently manage code changes, collaborate on projects, and maintain a detailed history of development progress. The functionality of GitLab is integral to ensuring that software development processes are both organized and traceable.

GitLab features a sophisticated role-based access control system, similar to the setup of other Eurosoftware tools such as Confluence and Jira. This system allows for customized permissions, ensuring that team members have appropriate levels of access to different projects, thereby promoting a unified and streamlined workflow. This setup not only protects sensitive project details but also improves overall operational efficiency.

For the Product Development department, Eurosoftware uses GitLab to develop the core software platform, focusing on maintenance, general enhancements and innovations. These efforts are supported by multiple non-customer-specific repositories. Conversely, in the Customer Solution department, each project is given its dedicated repository - or often multiple repositories - to reflect the specific needs and requirements of each customer. This client-centric approach ensures customized access rights, which increases the security and integrity of custom development.

### 2.3.2 Eclipse

Eclipse plays a crucial role in Eurosoftware's development landscape, primarily due to its versatility and adaptability. Eclipse is an open source Integrated Development

Environment (IDE). It offers significant advantages, particularly for Java-based development projects.

The most important aspect of Eclipse is its ability to seamlessly integrate custom plugins. These plugins are essential for the correct functioning of custom programming languages and for ensuring their compatibility with Java. They play an important role in the validation of these custom languages and contribute to the accurate compilation of projects into final software products.[Son23]

Eclipse is not only a preferred IDE at Eurosoftware, it is a mandatory tool for all developers. This requirement comes from its ability to support specific programming languages that were developed internally at Eurosoftware. By standardizing Eclipse, the company leverages its comprehensive set of features and capabilities, making it the ideal environment for developing high-quality software solutions.

### **2.3.3 JetBrains Products**

At Eurosoftware, JetBrains IDEs such as WebStorm and IntelliJ IDEA are utilized specifically for projects where Eclipse's capabilities may be insufficient, typically within web development. These IDEs are not mandatory for all developers; their use is contingent upon necessity, with the company funding subscriptions only when required.

The selective adoption of JetBrains products is influenced by several factors. First, the cost - unlike the free open source Eclipse, JetBrains IDEs require a paid subscription. Second, Eclipse supports the custom programming languages developed by Eurosoftware, a feature that JetBrains IDEs do not offer. Over time, as JetBrains products have become recognized for their sleek design and user-friendly interface, they have begun to complement the traditional use of Eclipse.

### **2.3.4 SoapUI and Postman**

At Eurosoftware, both SoapUI and Postman play critical roles in Application Programming Interface (API) development and testing, although neither is mandatory for all developers. SoapUI is highly valued for its capability in testing both SOAP and REST APIs. Beyond its primary testing functions, it offers a mock service feature, which is essential for simulating web services before their full development. This allows testers and developers to configure SoapUI to return specific responses to API requests, a crucial feature for testing scenarios where the actual service is not

yet available or is still under development.

Complementing the capabilities of SoapUI, Postman provides an intuitive, easy-to-use interface that is widely used for building, testing, and modifying APIs. It allows developers and testers to send requests to web servers and examine the responses. Postman supports the creation and storage of both simple and complex HyperText Transfer Protocol (HTTP) requests and their responses, which are essential for debugging and developing APIs. It is particularly useful for retrieving data from servers or sending data during various stages of development. Although not mandatory, Postman makes interacting with services much easier, and without it, some of Eurosoftware's developments would be much more difficult.

### 2.3.5 Dbeaver

DBeaver emerges as a highly effective tool for managing the complex databases at Eurosoftware. This database management software is renowned for its intuitive user interface, which facilitates seamless interaction with a variety of database types. DBeaver's versatility is particularly beneficial for Eurosoftware's diverse database needs, allowing connections to multiple databases simultaneously. This tool is particularly helpful for viewing and editing database structures, running SQL queries, and analyzing data. Users can easily visualize relationships between tables, which is crucial for understanding complex database schemas.

While DBeaver is a popular choice among the developers at Eurosoftware, its use is not mandatory. Developers have the flexibility to select from other available tools that may better suit their specific needs. However, due to its comprehensive features and intuitive design, DBeaver remains a preferred option for many at Eurosoftware, contributing significantly to the efficient management of their extensive and complex databases.

## 2.4 Programming Languages

Programming languages are more than just a means of instructing a computer to perform tasks. They are the foundation upon which software's logic, design, and functionality are built. At Eurosoftware, the chosen languages reflect a blend of industry standards, practical efficiency, and the capacity to meet the complex demands of custom software development. Each language, with its distinct syntax and capabilities, plays a critical role in the diverse projects undertaken by the company, from web development to complex system architectures.

## 2.4.1 Java

Java is the foundation of Eurosoftware's development, representing the most fundamental and essential programming language. Such is the importance of Java that when developers apply for a development position at the company, they are primarily evaluated as "Java developers" and their knowledge of Java is a key criterion during the interview process. This underscores the language's central role in the company's technology ecosystem.

Java is known as an Object Oriented Programming (OOP) language. OOP is a paradigm that organises software design around data, or objects, rather than functions and logic. This approach models real-world entities as objects and emphasises the principles of encapsulation, inheritance and polymorphism. Encapsulation allows data to be bundled with methods that operate on that data; inheritance facilitates the creation of new classes based on existing ones; and polymorphism allows objects to be treated as instances of their parent class, increasing flexibility and reusability.[Koç09]

**Java Swing (Swing).** Swing at Eurosoftware has evolved significantly from its standard form to a highly customized tool tailored to the company's unique development needs and customer requirements. While retaining the fundamental aspects of Swing, such as component rendering and core framework logic, Eurosoftware has extensively modified the technology to simplify development and better serve its customers.

In Eurosoftware's implementation, the visual elements of POS systems are defined using specialized formats. The layout of the POS is described in a unique XML format that specifies the layout and properties of the GUI components. In addition, a special format known as 'tableau.json' is used for more detailed design aspects such as button placement, actions and configuration.

These XML and 'tableau.json' visual definitions are seamlessly integrated with the Java and custom programming languages used at Eurosoftware. This integration allows for a highly flexible and efficient development process, enabling the creation of customized and user-friendly interfaces.

**Java Spring Framework (Spring).** Spring's strength lies in its comprehensive approach to application development. It makes managing objects and classes efficient, ensuring that every component of a software project is instantiated, used, and disposed of correctly. This is critical to ensuring the long-term sustainability of a

project by maintaining a clean and easily managed codebase.

In addition, Spring offers a wide range of features for various aspects of application development, such as security, transaction management and data access. Its robust security framework helps protect applications against common vulnerabilities, while its transaction management ensures data consistency and integrity. Spring also simplifies interactions with databases, making it easier to manage large data sets and perform complex database operations, thanks to its extensive support for data access technologies. [Beh23]

At Eurosoftware, Java Spring is more than a framework. It's a catalyst for creating robust, efficient and future-proof software solutions that embody the company's commitment to delivering high-quality and versatile POS systems.

**Maven.** Maven plays a crucial role at Eurosoftware by optimizing the coupling of different application modules and managing the build lifecycle of Java applications, especially those that include custom components. Its ability to effectively handle multiple interconnected components is critical in Eurosoftware's complex development environments. Maven automates the inclusion and management of external libraries, streamlining the integration and functioning of all parts of an application. It also manages dependencies, ensuring that each project is equipped with the most appropriate tools for efficient development.[Col23]

In particular, Maven's structured approach is essential for compiling the custom-generated properties files and Java classes that are an integral part of Eurosoftware's proprietary programming languages. This ensures seamless integration with the main Java application, maintaining a cohesive and functional software architecture. By managing the build process and resolving dependencies, Maven significantly enhances the robustness and scalability of Eurosoftware's POS systems and is a key component of their development strategy.

## 2.4.2 XML

XML is a flexible, structured language for storing and transporting data. It plays a crucial role in Eurosoftware's development environment, where it is one of the most widely used formats.

Within the Java ecosystem at Eurosoftware, XML is used in the Spring framework to declare beans - these XML configurations dictate how beans are created, managed and wired in the context of the Spring application. Similarly, security con-

figurations use XML to define security rules and filters to ensure a secure and robust application.

In the Maven ecosystem, the pom file, written in XML format, is the cornerstone of project configuration. It defines exactly how a project is built, its dependencies and other build parameters. This file is critical because it dictates the entire build process, ensuring consistency and accuracy in project compilation and dependency management.

In addition, the flexibility of XML extends to the creation of custom layouts for Eurosoftware's POS systems. These XML-defined layouts allow for a high degree of customisation, enabling developers to specify intricate user interface designs and interactions tailored to specific business requirements. An example of this can be found in the XML layout code example detailed in the Appendix, Section C.1.1.

Through these diverse applications, XML proves to be more than just a data format at Eurosoftware. It is a crucial element in defining application structures, security configurations, build processes, and user interface designs.

## 2.4.3 Web Development Languages at Eurosoftware

JavaScript is a cornerstone of web development at Eurosoftware, enabling the creation of dynamic and responsive web applications. Its integration with back-end systems via Representational State Transfer (REST) APIs supports a design architecture that offloads significant application logic to the Java back-end, allowing the JavaScript front-end to efficiently manage user interactions. In addition, TypeScript, a superset of JavaScript that adds static types, is used to improve code quality and readability, ensuring that applications are not only interactive but also robust and easy to maintain.

Vue.js is another key player in Eurosoftware's technology stack, known for its efficiency and flexibility in developing user interfaces and single-page applications. While not as widely used as JavaScript, Vue.js is increasingly being used in key projects such as UTPOS, demonstrating its ability to deliver dynamic and efficient user experiences. The core library's focus on the view layer allows for easy integration with other libraries or existing projects, promoting a modular and scalable front-end architecture.

This diverse suite of web development technologies - JavaScript, TypeScript, and Vue.js - underscores Eurosoftware's strategic approach to using advanced tools

to meet complex customer needs. By leveraging these technologies, Eurosoftware ensures that its applications are at the forefront of innovation, delivering powerful, user-friendly, cutting-edge solutions that increase overall user engagement and satisfaction.

## 2.4.4 ProMo

ProMo, developed by Eurosoftware, is a programming language designed to simplify the development of implementation-independent models, specifically Process Model Domain Specific Language (ProMoDSL) models, into actionable Java and other technology implementations. It serves as an essential bridge between high-level process designs and their practical applications, integrating seamlessly with Java Swing, Java Spring, and custom XML layouts. The language is specialized for Eurosoftware's internal use and features a unique syntax that is not commonly taught outside the company.

The language facilitates the creation of complex workflows and process management, enabling Eurosoftware developers to create sophisticated software applications. ProMo's syntax allows the definition of actions, conditions, and state management, which streamlines process flow and increases modularity within software projects. This structured approach enhances process customization and flexibility, supporting dynamic adaptations and the integration of sub-processes for comprehensive application logic. An example of ProMo source code can be found in the Appendix, section C.1.2.

## 2.4.5 ProMoJ

ProMoJ extends the functionality of ProMo by binding the high-level process definitions to specific implementations in Java or other underlying technologies. This stage is critical because it translates abstract ProMo processes into executable actions within the software's operational environment. ProMoJ focuses on the implementation of the actions, conditions and transitions defined in ProMo, ensuring that the designed workflows are effectively implemented and integrated into the application's technical stack.

In addition to implementing actions and conditions, ProMoJ also manages the integration of user interface elements defined in XML layouts, linking User Interface (UI) components directly to process steps. This capability enables a seamless user experience where high-level process flows are not only executed in the background but also visually presented and interacted with through the application's front end. By combining complex processes with intuitive interfaces, ProMoJ plays a key role



in ensuring that Eurosoftware's software solutions are both functional and user-friendly. See the Appendix, section C.1.3 for an example of the ProMoJ code.

## 2.4.6 DO DSL

In the landscape of Eurosoftware's proprietary technologies, DO DSL plays a critical role by providing a unique and customized solution specifically designed to enhance the configuration flexibility of its POS systems. This specialized framework allows the creation of configurations that adaptively modify system behavior to meet specific business requirements. DO DSL is used to generate a variety of configuration components, including .properties files that provide a textual representation of settings, as well as specialized Java classes and factory classes that integrate seamlessly with Java code. Detailed examples of configuration files and Java class generation using DO DSL can be found in Appendix C.1.4.

DO DSL's capabilities extend to defining how configuration objects should be structured without dictating their actual content, and it includes mechanisms for setting default values that are embedded in the Java code during generation. These defaults can be overridden in specific instances, providing flexibility and control over system behavior. This framework supports the generation of specific instances with predefined attribute parameters, facilitating the integration of these configurations into the runtime environment or into other Eurosoftware proprietary frameworks such as ProMo/J.

The ability to dynamically configure and customize system behavior through DO DSL increases not only the adaptability but also the efficiency of Eurosoftware's POS systems. It simplifies the management and deployment of complex configurations, reduces operational complexity and ensures that systems are both robust and flexible. This makes DO DSL a cornerstone of Eurosoftware's strategy to use advanced technology to meet the evolving needs of its customers.

## 2.5 Architecture

Eurosoftware stands as a considerable entity in the technology landscape, boasting an array of POS systems designed to meet a variety of customer needs. These systems are not only optimized for performance, including speed and reliability but also tailored to fit various price points, making them accessible to a broad spectrum of businesses.

The architecture of Eurosoftware's offerings is built on a foundation of flexibility and scalability, ensuring that each POS solution can be customized to the specific requirements of its customers. From small retail outlets to large, multinational chains, Eurosoftware can provide a POS system that aligns with the customer's operational needs, aesthetic preferences, and budget constraints.

## 2.5.1 Core Architectural Solutions

At the core of Eurosoftware's architecture lies a fundamental information flow consisting of MasterData and Transactions. MasterData represents all non-transaction and journal information, such as items, merchandise hierarchy groups, taxes, employees, store characteristics, etc. Transactions encapsulate all accounting-relevant operations, serving as the virtual representation of money movement. These are primarily created on POS systems based on MasterData and subsequently processed on the POS-Server during accounting procedures.

**Universal Connector (UCON/ECON):** The UCON/ECON is the main interface for external communication within the GK/Retail Business Suite. It facilitates the import and export of data, transforming external data formats, such as Intermediary Document (IDOC) or XML from Systems Applications and Products in Data Processing (SAP), into internal XML format and the other way around. This conversion process is supported by the use of Java Message Service (JMS) messages and Camel routes, ensuring efficient data processing and exchange.

**Cluster Infrastructure Server (CIS):** As a prerequisite for the operation of central applications, the CIS initializes the system infrastructure. It hosts the ActiveMQ Broker, responsible for the processing and distribution of JMS messages throughout the system, thus serving as the backbone for message communication.

**Store Device Control (SDC):** The SDC functions as the Masterdata manager, importing and storing essential data in an internal XML format. It also handles the replication of this data to other systems and manages the transaction pool, acting as a repository for transactional data. This capability is crucial for ensuring data consistency across the POS ecosystem.

**POS-Server:** Serving as the central processing unit, the POS-Server is tasked with transaction management, report generation, and the execution of export functions. It also facilitates the monitoring and control of financial movements, translating transaction data into actionable insights through predefined data sources. For

a detailed view of how these components are organized within the system, refer to the architecture diagram in Figure 2.1.

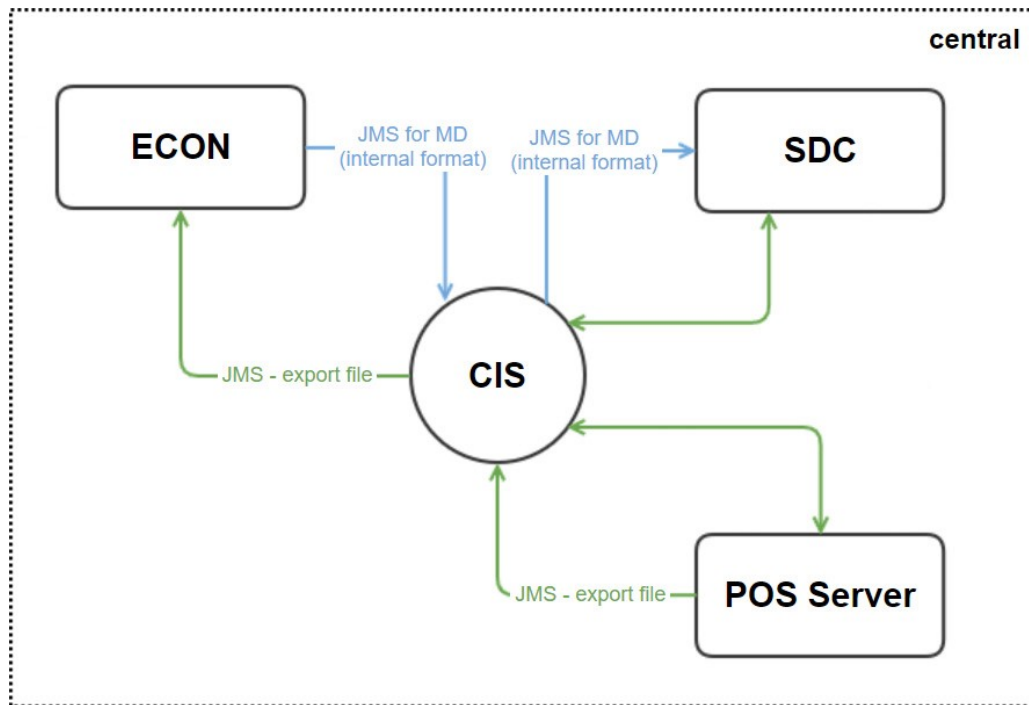


Figure 2.1: Central Architecture Scheme

While the previously described components form the backbone of Eurosoftware's POS solutions, offering a robust and scalable infrastructure, it is often the OmniPOS system where developers find themselves most engaged. OmniPOS stands as a front-runner in the suite of Eurosoftware's applications, embodying the convergence of user interface and transactional functionality, essential for the seamless selling of goods and processing of payments.

## 2.5.2 OmniPOS Architecture

OmniPOS features a layered architecture, pivotal for its omni-channel capabilities. This structured approach ensures modularity, wherein each layer furnishes a distinct set of APIs, and interaction across layers adheres to a strict top-down methodology. This design philosophy underpins two primary deployment models within OmniPOS: Thin-POS and Fat-POS, delineating the separation between client and server sides. Notably, the server side of OmniPOS is designed to be stateless, optimizing for both scalability and stability.

### 2.5.2.1 Service Layer

At the heart of OmniPOS's architecture is the Service Layer, which orchestrates a variety of services:

- **Domain Object (DO) Services:** These services encapsulate access to other APIs and database interactions, functioning as the foundational layer for data management within OmniPOS.
- **Business Logic Services (BLServices):** Dedicated to internal operations, these services encapsulate the core business logic of OmniPOS, ensuring the business rules and processes are adhered to and efficiently executed.
- **Exposed Services:** As the name suggests, these services are designed to provide external access to OmniPOS functionalities, incorporating error handling mechanisms and Java API for RESTful Web Services (JAX-RS) annotations to facilitate communication.

**Service Exporters** play a crucial role by making Exposed Services available for external use, predominantly through REST web services with JavaScript Object Notation (JSON). This mechanism is vital in Thin-POS deployments or for other clients like the mobile POS or SAP Open User Interface 5 (UI5) Client, allowing for a pluggable architecture that can be activated via configuration.

**Service Locator** emerges as a central component in navigating the services on both the Client and Back-End sides. It enables the inclusion of external or custom services into the overall infrastructure, masking deployment-specific characteristics and facilitating the registration of services through API or Spring configurations.

**Process Layer** introduces a higher level of abstraction, utilizing the company-wide standard ProMoDSL for modelling business processes. This layer separates product and project processes to minimize migration efforts, supported by an Eclipse-Tooling environment for editing, validating, visualizing, and debugging process models.

**Hardware Abstraction Layer (HAL)** a critical component of OmniPOS, which encapsulates access to POS-related hardware, offering an abstraction layer for various device categories like printers, scanners, cash drawers, and more. This layer ensures event-based interaction with the rest of the system, providing a seamless integration of hardware functionalities into the POS operations.

# Possible development facilitations

## 3

AI is revolutionizing the software industry, driving a transformative shift where efficiency, scalability, and predictive capabilities are critical. As AI accelerates development cycles and improves code quality, it enables products to reach the market faster and with greater sophistication, giving companies a competitive edge. This technology not only speeds production, but also acts as a collaborative tool that refines and optimizes the work of developers, making it an indispensable ally for staying agile and innovative in a rapidly evolving digital landscape.

### 3.1 Universal Improvement Opportunities

There are general ways in which AI can make things easier and improve the development process. These enhancements could be used by any company

#### 3.1.1 SQL Operations

In data-driven applications, efficient data retrieval and manipulation is crucial. AI can meticulously analyze database structures and existing SQL queries to offer optimized versions, ensuring faster data retrieval and more effective database interactions. Machine learning algorithms can predict query performance and suggest indexes or changes to query structures. When provided with table content, AI can generate insert scripts, streamlining the data population process and significantly reducing the risk of human error.

#### 3.1.2 Lambda Expressions

Lambda expressions, while powerful, can sometimes be complex and difficult to decipher. AI can step in to simplify these expressions without losing their essence, ensuring that the code remains elegant and is easier for other developers to understand. By analyzing the functional context and the intended purpose of the lambda expres-

sion, AI can recommend more readable and maintainable code patterns, thereby enhancing the overall development workflow.

### **3.1.3 Advanced Code Refactoring**

Artificial intelligence could significantly improve software development by optimizing code refactoring and preventing potential failures. AI could analyze code structures to identify opportunities for improvement, suggesting the modularization of bloated features and the integration of design patterns to improve scalability and readability. It could also free developers to focus on more strategic improvements by automating the boring aspects of refactoring, such as identifying redundant code and recommending code consolidation. In addition, AI could improve software durability by preemptively identifying code sections that may be error-prone, suggesting appropriate error handling. This proactive approach by AI could not only speed up the development process but also significantly improve the overall quality and reliability of software.

### **3.1.4 Cross-Language Code Translation**

In today's technological landscape, developers often find themselves working across multiple programming languages. One of the challenges they sometimes encounter is the need to translate code from one language to another — either for integration purposes or to migrate a project. Traditional manual code translation can be error-prone and time-consuming. Artificial Intelligence, with its pattern recognition and syntactic analysis capabilities, offers a promising solution to this dilemma. Through sophisticated algorithms, AI can not only map equivalent constructs between different languages but can also consider idiomatic usage, ensuring that the translated code retains its original intent and functionality.

### **3.1.5 Code Explanation and Interpretation:**

Legacy code can present a labyrinthine challenge, filled with sections that may seem cryptic even to seasoned developers. Historically, the approach was to scour documentation, seek advice from colleagues, or spend extended periods dissecting code to comprehend its essence. However, the dawn of AI in software development heralds a new era. By harnessing its analytical prowess to dissect the code's structure, review related comments, and understand inter-module interactions, AI delivers precise explanations of code objectives. This facilitates a clearer understanding of its primary functions and potential interdependencies, streamlining the code navigation process.

## 3.2 Facilitation Opportunities for Eurosoftware

The proposed enhancements incorporate the basic principles and architectural paradigms that are already in place in Eurosoftware s.r.o. These enhancements are intended to be integrated into the existing architecture of the company and, due to their specificity, should assist in more enterprise-specific issues.

### 3.2.1 Confluence

Traditionally, Eurosoftware stores extensive documentation and resources on Confluence, which serves as a critical repository for training materials, codebase information, and company-specific programming guidelines. This platform is essential for onboarding new developers, who often need to navigate through a wealth of information with the help of experienced mentors. This process can be time-consuming and takes experienced developers away from their primary tasks to provide the necessary guidance and support.

However, implementing an AI trained on the detailed datasets available in Confluence could revolutionize this training approach. An AI chat assistant, armed with insights from Eurosoftware's proprietary architectures and historical code examples, could act as an on-demand resource for new developers. This AI could answer questions about software architecture or specific programming nuances based on the rich information it has learned from Confluence. By providing instant clarification and suggestions, the AI assistant would not only streamline the onboarding process but also free up valuable development time, allowing experienced developers to focus on their core tasks while ensuring that newcomers integrate efficiently into the team.

### 3.2.2 Domain Specific Languages

Domain Specific Languages at Eurosoftware, although powerful, can present a steep learning curve for new developers unfamiliar with their unique syntax and operational frameworks. Mastering these languages requires poring over extensive Confluence documentation or consulting colleagues

Integrating AI trained in these specialized DO DSLs could streamline this learning process. AI could act as a highly effective support tool by accessing large datasets containing the specifics of these languages and examples of previously written code. It could assist new developers by automatically completing or suggesting code snippets, offering clarification on complex syntax, and performing code refactoring.

Such an AI system would not only act as a code assistant but also as an educational resource, simplifying the understanding of domain-specific languages and accelerating the development process at Eurosoftware. This AI-driven approach would ensure that new developers become proficient quickly, increasing the overall productivity and efficiency of the development team.

### 3.2.3 Custom XML and CSS

Eurosoftware's development approach involves highly customized XML layouts and a unique adaptation of CSS that differs significantly from standard usage. XML layouts are tailored to specific business needs and deviate from conventional structures. Similarly, CSS has evolved to work in unique harmony with these XML layouts, resulting in certain styles and XML combinations that are not typically supported. However, this unique relationship between custom CSS and XML layouts can also inadvertently disrupt the standard functionality expected with typical HyperText Markup Language (HTML) and CSS, resulting in unexpected behavior in otherwise standard scenarios. An AI system could be the key to managing this complexity. By analyzing the interaction between custom XML and modified CSS, AI can identify incompatible elements and suggest optimal configurations. This ensures that layouts and styles work together, maintaining both aesthetic quality and operational efficiency.

### 3.2.4 Integrated Coding Advisor Plugin

Beyond just answering queries, there's potential for AI to take a more proactive role in the developer's workflow. By integrating an AI-driven plugin within the development environment, it's feasible to have a system that anticipates the developer's needs. As a developer crafts code, this AI advisor, familiar with the nuances of domain-specific languages, can suggest possible next steps, code completions, or even highlight potential issues before they manifest. It acts as a co-pilot, understanding the intent and direction of the developer, and offering guidance every step of the way. Such integration not only speeds up the coding process but also ensures that the code adheres to the best practices and standards set by the Eurosoftware company.



# Leveraging Existing Tools for Enhanced Development

## 4

The integration of AI into software development tools is rapidly changing the way we approach coding, problem-solving, and project management. There is a wide variety of AI-enabled tools available, each designed to streamline development processes, improve code quality, and automate repetitive tasks. These tools range from those that help complete and optimize code to those that can identify bugs and predict coding errors.

This chapter serves as an introduction to the diverse world of AI-powered development tools. Our goal is to provide an overview of these tools, highlighting their key benefits and potential drawbacks. By doing so, we hope to provide a glimpse into how AI is revolutionizing software development and set the stage for deeper dives into specific tools and their applications in later discussions.

## 4.1 Exploring ChatGPT: A Comparative Analysis

OpenAI's development and deployment of ChatGPT represents a significant milestone in the field of NLP, harnessing the power of Generative Pre-trained Transformer (GPT) models. These models are designed to understand and generate text that closely mimics human conversation, enabling a wide range of applications from automating customer service to assisting in creative writing and technical problem-solving. Among the various versions released, GPT-3.5 and GPT-4 have attracted considerable attention for their advanced capabilities, which will be examined in detail, including their memory capacity for context and subscription-based limitations [Kel24].

## 4.1.1 ChatGPT 3.5

Introduced as an enhancement over its predecessor, GPT-3.5 was launched to bridge the capabilities gap between GPT-3.5 and GPT-4. As noted by Kelly (2024), "GPT-3.5, the refined version of ChatGPT Version 3.0 (GPT-3) rolled out in November 2022, is currently offered both in the free web app version of ChatGPT and via the paid Turbo API" [Kel24]. This model is designed to improve language processing and generation, tailored specifically to function efficiently within OpenAI's structured API offerings.

- **Language Understanding and Generation:** According to Kelly (2024), GPT-3.5 improves upon its predecessor by enhancing text translation, summarization, and content creation, making it a versatile tool for various digital communication tasks [Kel24].
- **Token Limit and Context Memory:** GPT-3.5 supports a context window of up to 16,385 tokens, allowing for more extended interactions and deeper conversational contexts compared to earlier models [Kel24].
- **Access Without Subscription:** Users without subscriptions may face access constraints during peak times, which can affect service availability.

## 4.1.2 ChatGPT 4

GPT-4 represents a significant leap forward in the model's evolution, introducing advanced features that cater to a wider array of complex applications. According to Kelly (2024), "GPT-4, released in March 2023, offers another GPT choice for workplace tasks. It powers ChatGPT Team and ChatGPT Enterprise, OpenAI's first formal commercial enterprise offerings" [Kel24]. The introduction of multimodal capabilities and expanded context windows are pivotal enhancements that define this iteration.

- **Advanced Context Understanding:** GPT-4 expands the token limit to 32,768, allowing for richer and more extended interactions. This capability is particularly beneficial for applications involving long-form content creation and detailed dialogue scenarios [Kel24].
- **Multimodal Functionality:** As Conroy (2024) highlights, GPT-4 introduces the ability to process image inputs, enabling it to analyze visual data and generate relevant textual responses, thus broadening the scope of its applications [Con24].

- **Comprehensive access:** GPT-4 subscribers benefit from comprehensive access to all new features, including internet browsing capabilities and advanced image processing, ensuring they leverage the full potential of the model [Kel24].

### 4.1.3 Conclusion

As the field of artificial intelligence continues to evolve, the continued development and refinement of models such as ChatGPT-3.5 and GPT-4 are critical to pushing the boundaries of what is possible with AI. The exploration of these models, through theoretical analysis and practical application, will illuminate the path forward for researchers, developers, and users alike, fostering a deeper integration of AI into our daily lives and work.

## 4.2 Exploring Tabnine: AI-Powered Code Assistance

Tabnine is an AI-assisted coding tool that provides an advanced code completion service, designed to enhance developer productivity through deep learning algorithms. As an alternative to traditional code completion tools, Tabnine offers both cloud-based and local processing options, making it a versatile choice for developers operating in different environments. Tabnine can significantly boost coding efficiency, which is crucial for developers working on both new and existing projects [Ros23].

### 4.2.1 Cloud-Based Model

Tabnine's cloud-based model provides powerful code completion capabilities, utilizing cloud computing to handle complex computations and data storage. This model supports high levels of security and compliance, aligning with the stringent requirements of many organizations. According to David Ramel, "Tabnine Enterprise allows isolated environment deployment using Virtual Private Cloud or on-premises deployment, ensuring total code privacy and security." Additionally, Tabnine Chat, a feature within the cloud model, is designed to interact with developers directly within their IDE, providing suggestions and corrections based on context. Importantly, Tabnine ensures that all models are trained only on open-source code with permissive licenses, which guarantees that the models are not trained on General Public License (GPL) or other copyleft code, thus avoiding potential legal and security issues [Ram23].

## 4.2.2 Local Processing Model

In contrast, the local model of Tabnine ensures all data processing occurs on the developer's own machine, thereby not transmitting any code beyond the firewall. This model addresses significant security concerns, making it ideal for projects with strict data confidentiality requirements. Weiss highlights that this setup not only enhances privacy but also allows developers to train the AI model on their specific codebase directly within their local environment, thus tailoring the AI's learning and suggestions more closely to their specific needs [Law22].

## 4.2.3 Hybrid Model Adaptability

Tabnine also supports a hybrid model, which combines the cloud's computational power with the local model's privacy benefits. This adaptability is crucial for developers who need the scalability of cloud resources but are constrained by strict data privacy and security protocols. According to Loraine Lawson, Tabnine can operate "on a developer's computer without sending code beyond the firewall, and it can also run in the cloud or a hybrid environment" [Law22]. This flexibility ensures that developers can choose the most suitable processing method based on their specific needs and constraints, offering a tailored coding assistance experience.

## 4.2.4 Conclusion

Tabnine adapts to different development environments and security requirements and has established itself as a critical tool for increasing productivity and maintaining high standards of code quality and security in software development with its comprehensive range of deployment options—cloud, local, and hybrid.

# 4.3 Exploring GitHub Copilot: AI-Assisted Coding

GitHub Copilot, developed by GitHub in collaboration with OpenAI, is an innovative tool that assists developers with intelligent code completion. This AI-powered assistant significantly improves coding efficiency by suggesting complete lines of code or entire functions based on the context of the existing code [Rar23].

## 4.3.1 Adaptive Learning

GitHub Copilot personalizes its assistance by learning from the user's coding style and preferences. This adaptive learning capability is crucial for providing more

relevant and contextually appropriate code suggestions, thereby improving coding efficiency over time [Rar23].

### 4.3.2 Security Considerations

Security and intellectual property are paramount when integrating AI-assisted tools like GitHub Copilot into organizational workflows. Tuomas Leppilampi highlights a prevalent concern: "But, there are some concerns about intellectual property leaks—especially if organizations aren't using business or enterprise licensing with AI-assisted tools" [Lep]. This underscores the importance of using appropriate licensing to safeguard sensitive data and intellectual property when deploying such technologies.

### 4.3.3 Personal Evaluation

Given these concerns, any exploration or testing of GitHub Copilot related to Eurosoftware's specific needs will be done on a personal computer. This controlled approach allows for a thorough evaluation of Copilot's utility and effectiveness without compromising the security and integrity of Eurosoftware's codebases. Evaluating Copilot in this way will provide valuable insights into how AI-assisted coding tools can be leveraged within Eurosoftware's development practices, while ensuring compatibility with existing platforms and adherence to strict security protocols.

### 4.3.4 Conclusion

While GitHub Copilot offers significant benefits in streamlining coding processes and increasing developer productivity, its integration requires careful consideration of security, intellectual property, and organizational workflows. By evaluating its potential and implementing it sensibly, organizations can maximize the benefits while minimizing the risks associated with it.

## 4.4 Custom NLP Model: Eurosoftware's Approach

At Eurosoftware, the Confluence database is an expansive and continually evolving repository that forms the backbone of the company's knowledge management. A model knowledgeable in the architectural intricacies of Eurosoftware's applications could significantly enhance training processes for new employees and assist seasoned staff when they encounter challenges.

## 4.4.1 Potential of NVIDIA NeMo in Custom AI Development

A promising tool for this purpose is NVIDIA NeMo, an advanced framework designed for the creation and deployment of generative AI models. As described by NVIDIA:

"NVIDIA NeMo™ is an end-to-end platform for developing custom generative AI—including Large Language Model (LLM)s, multimodal, vision, and speechAI—anywhere. Deliver enterprise-ready models with precise data curation, cutting-edge customization, Retrieval Augmented Generation (RAG), and accelerated performance." [NVI23c]

The NVIDIA NeMo framework is equipped with advanced features that can be instrumental in developing and scaling AI models for complex applications. As detailed by NVIDIA:

"Transformer-based LLM and multimodal models can leverage NVIDIA Megatron Core for scaling training for models with billions of parameters across thousands of Graphics Processing Unit (GPU)s. Megatron Core includes state-of-the-art parallelization techniques such as tensor, pipeline and sequence parallelism, and selective activation recomputation for optimal performance." [NVI23b]

Further enhancing its utility, NeMo provides several customization techniques to refine pretrained models:

"The NeMo framework offers customization techniques to refine pretrained LLMs including p-tuning, LoRA, and Supervised Fine Tuning (SFT). NeMo LLMs can be aligned with state-of-the-art methods such as SteerLM, Direct Preference Optimization (DPO) and Reinforcement Learning From Human Feedback (RLHF) through NVIDIA NeMo Aligner." [NVI23b]

These capabilities make NVIDIA NeMo an appealing option for organizations like Eurosoftware, looking to leverage AI for a wide range of applications, from language understanding to speech recognition.

## 4.4.2 Llama 2 7B Model

Llama 2, developed by Meta, is a versatile collection of pretrained and fine-tuned generative text models, which vary in scale from 7 billion to 70 billion parameters

[Met23]. Given Eurosoftware’s server capabilities, the 7 billion parameter model is of particular interest as it balances computational demands with advanced AI capabilities.

Llama 2 operates as an auto-regressive language model using an optimized transformer architecture. Its tuned versions employ SFT and RLHF to align closely with human preferences for safety and helpfulness [Met23].

The 7 billion parameter version is intended primarily for commercial and research applications in English, optimized for assistant-like chat functionalities. Pre-trained models provide flexibility, adapting to various natural language generation tasks, which makes Llama 2 highly suitable for enhancing Eurosoftware’s customer interactions and internal operations [Met23].

### 4.4.3 Megatron-GPT 5B Model

Megatron-GPT 5B, developed by NVIDIA, is a transformative advancement in the domain of artificial intelligence, particularly in the construction of language models. This model falls within the transformer-based GPT class, which are decoder-only models akin to GPT-2 and GPT-3, celebrated for their proficiency in producing human-like text. The designation "5B" reflects the model’s capacity, indicating it is equipped with 5 billion parameters, thereby underscoring its significant computational power and its ability to manage complex language tasks [Kuc23].

Model Parallelism is a key feature that enhances the Megatron-GPT 5B’s capabilities. According to NVIDIA’s documentation, Megatron-LM is a highly optimized library for training large language models using model parallelism. This allows the training of models with billions of parameters. Once trained, these models can be utilized in NeMo for various downstream tasks. NeMo seamlessly handles pretrained model parallel checkpoints from Megatron-LM, ensuring that model parallel models in NeMo retain all the features typical of NeMo models [NVI23a].

This integration exemplifies how Megatron-GPT 5B leverages cutting-edge technology to facilitate the development of AI models that are not only powerful but also versatile, capable of being applied across different platforms and tasks.

# Practical Application of General Tools

## 5

With a rich landscape of AI tools offering diverse capabilities, the practical application of these resources becomes a key focus for developers looking to improve their workflows. This chapter is dedicated to understanding how popular AI tools such as ChatGPT, TabNine, and GitHub Copilot can be pragmatically applied to the software development lifecycle.

## 5.1 ChatGPT 3.5: Free-to-use conversational AI

Having previously introduced the capabilities and features of GPT-3.5, we now turn our attention to its practical applications within the software development sphere. This section delves into the real-world utility of ChatGPT 3.5 as an integral part of the developer's toolkit, examining its role in streamlining the coding process, facilitating debugging, and providing on-the-fly solutions to complex programming problems.

### 5.1.1 Explaining SQL Scripts

In the realm of database maintenance, the ability to elucidate SQL scripts stands as a critical skill. Presumably, ChatGPT 3.5, with its advanced language processing capabilities, should proficiently interpret SQL scripts from various sources. ChatGPT 3.5 will be tested on scripts actively used by Eurosoftware, not on common scripts available on the Internet. This will provide us with a spectrum of complexity and contextual usage to assess the model's effectiveness in delivering clear, concise explanations.

**Details:** Detailed explanations provided by ChatGPT 3.5, the corresponding scripts, and the evaluation of how ChatGPT managed each example can be found in the `Results\gpt3_5\Explaining SQL` directory, specifically in the `SQL_exp1.txt`



file.

**Outcome:** Across a range of tests, ChatGPT 3.5 demonstrated a robust ability to explain SQL scripts, effectively explaining complex operations and providing insightful interpretations that facilitate a deeper understanding of database scripts. The model showed particular strength in breaking down complicated SQL commands into understandable explanations, highlighting its potential as a valuable tool for database management and maintenance.

## 5.1.2 Optimizing SQL Queries

Given ChatGPT 3.5's demonstrated ability to explain SQL scripts, we now focus on its potential for query improvement. This aspect tests the AI's ability not only to understand SQL scripts but also to optimize them for improved performance. Query optimization is a key part of database management, often requiring a deep understanding of both the data structure and the end goals of database queries. ChatGPT 3.5's performance in this area will be indicative of its utility in refining the speed and efficiency of data manipulation and retrieval.

**Details:** Detailed optimizations performed by ChatGPT 3.5, original and optimized scripts, as well as evaluations of the model's optimization attempts for each example, can be found in the `Results\gpt3_5\Optimizing SQL` directory, specifically in the `SQL_opt.txt` file.

**Outcome:** The evaluation of ChatGPT 3.5 in optimizing SQL queries showed mixed results: it significantly improved the execution time of one script, but slowed down another, suggesting that it has potential in improving simpler queries. However, improvements in complex scripts were minimal, primarily improving readability rather than performance. This suggests that while ChatGPT 3.5 can make SQL scripts easier to understand, it can't provide substantial performance improvements in complex or specific scenarios such as those at Eurosoftware, suggesting the need for a more customized model that understands the unique aspects of the database architecture it is applied to.

## 5.1.3 Explaining Lambda expression

By their nature, lambda expressions provide a concise way to implement functions without the need for traditional function declarations. However, their compact syntax can sometimes hide the logic they encapsulate. This makes them less accessible to developers unfamiliar with their use or the specific patterns within a codebase. GPT-3.5's advanced understanding of code semantics provides an opportunity to

demystify these expressions, offering explanations that can make the codebase more accessible and maintainable.

**Details:** Interpretations of lambda expressions by ChatGPT 3.5, along with the original expressions, are documented. These interpretations are stored in the `Results\gpt3_5\Interpreting Lambda` directory within `Lambda_interpr.txt`.

**Outcome:** When analyzing various lambda expressions, ChatGPT 3.5 generally performed well, providing clear and concise explanations in several examples. While it was adept at dealing with simple cases and those involving nested operations or comparisons against enumerations, the model was also able to provide reasonable interpretations of more complex lambdas that lacked explicit contextual clues. Despite some challenges with complex and low-context expressions, the AI demonstrated a strong ability to maintain accuracy and provide insightful explanations.

## 5.1.4 **Optimizing Lambda expression**

Beyond explanation, there is significant scope for GPT-3.5 to assist in the optimization of lambda expressions. This optimization can lead to more efficient execution, improved readability, or both, without compromising the logic the expressions are meant to implement. With GPT-3.5, developers can identify and implement optimizations that may not be immediately obvious, thereby streamlining code performance and maintenance.

**Details:** The original lambda expressions and their optimized counterparts, processed by ChatGPT 3.5 could be found in the `.txt` file within the provided directory `Results\gpt3_5\Optimizing Lambda\Lambda_opt.txt`.

**Outcome:** ChatGPT 3.5's lambda expression optimization showed generally positive results, demonstrating its ability to improve code readability and maintain functionality across different examples. Although the AI was successful in introducing several optimizations, its changes were sometimes superficial and limited to minor formatting changes. In cases where it suggested more substantial improvements, such as method abstractions or streamlined conditions, the AI occasionally failed to provide fully executable solutions. Overall, while ChatGPT 3.5 offered useful improvements in several areas, it still requires further refinement and developer oversight for comprehensive code optimization.

## 5.1.5 Lambda expression generation

The potential of GPT-3.5 extends to the generation of lambda expressions where appropriate. This includes not only the generation of new expressions but also the transformation of existing code patterns into lambda expressions where such a change offers clear advantages. This capability could further refine the efficiency and readability of the codebase, in line with modern development practices.

**Details:** Detailed information can be found in the file within the specified directory: `Results\gpt3_5\Generating Lambda\Lambda_gen.txt`.

**Outcome:** ChatGPT 3.5's efforts to generate lambda expressions showed mixed results, demonstrating its ability to improve code readability and embrace functional programming concepts such as the Stream API. However, the process was not without flaws, as evidenced by repeated failures to include critical parameters that are essential to the correctness of the operation. These oversights highlight significant limitations in the model's generation capabilities, requiring careful review and manual intervention.

## 5.1.6 Conclusion

Throughout this evaluation, ChatGPT 3.5 successfully handled a variety of tasks from explaining and enhancing SQL scripts to optimizing lambda expressions, demonstrating its capabilities as an effective software development tool. Its utility shines in simpler tasks where minimal contextual interpretation is required, making it a valuable asset for developers seeking quick solutions and routine automation. This functionality underscores its role in increasing productivity and simplifying the coding process.

However, as tasks become more complex, the model's limitations become apparent. This is especially the case when deep contextual understanding or nuanced decision-making is required. While ChatGPT 3.5 is adept at addressing a wide range of programming challenges, its performance diminishes when faced with complex, context-rich tasks. This highlights the need for more advanced models designed to handle the intricacies of sophisticated software development scenarios.

## 5.2 ChatGPT 4: Paid conversational AI

Building upon the established capabilities of GPT-3.5, the advent of ChatGPT 4 marks a significant advancement in the utility of conversational AI for software development. This section explores how the enhanced features of GPT-4 can be

leveraged to further streamline development workflows and solve complex programming challenges with greater finesse.

### 5.2.1 Code Interpretation with GPT-4

GPT-4 brings an advanced level of understanding to the interpretation of complex software constructs, aiming to demystify code segments that range from straightforward functions to intricate algorithms. This capability is crucial for making the codebase more accessible and easier to maintain by providing clear and comprehensive explanations.

**Details:** The interpretations made by GPT-4, alongside the original code segments, are recorded in detail. Documentation of these interpretations is available in the directory `Results\gpt4_0\Code Interpretation` in the `Java_interpr.txt` and `DODSL_interpr.txt` files.

**Outcome:** The evaluation of GPT-4 highlighted its robust adaptability and deep understanding of both familiar and unfamiliar programming languages. Initially challenged with DO DSL, GPT-4 provided a basic overview that improved significantly as additional context files were introduced, demonstrating its ability to quickly learn and adapt its explanations. In Java, the AI excelled at accurately describing the relationships within a complex architecture and effectively integrating a new method into the existing codebase while adhering to best practices. Although it occasionally missed some details, such as class inheritance in a Java Spring context, GPT-4's overall performance demonstrated its strong potential as a powerful software development tool capable of handling and extending complex software architectures.

### 5.2.2 Advanced Code Refactoring with GPT-4

GPT-4 is expected to redefine the standards for code refactoring, providing strategies for not only optimizing code performance but also improving readability and maintainability. This includes reorganizing code structures and introducing efficient coding practices without losing sight of the original logic and functionality.

**Details:** Examples of code refactored using GPT-4 along with a detailed evaluation can be found in the `Results\gpt4_0\Advanced Code Refactoring` directory in the `Java_refact.txt` and `DODSL_refact.txt` files.

**Outcome:** GPT-4 was evaluated on its ability to refactor code in Java and the less familiar DO DSL, displaying mixed outcomes that underscored its capabilities

and limitations. In Java, GPT-4 showcased strong proficiency, enhancing code clarity and robustness through well-implemented JavaDoc comments, error logging, and null pointer checks, alongside targeted improvements for performance and maintainability. However, its performance drastically differed when dealing with DO DSL, where it consistently failed to understand or refactor the code effectively, demonstrating significant errors and a fundamental gap in handling languages outside its training spectrum. This contrast highlights GPT-4's potential and current boundaries in AI-assisted programming, excelling in familiar contexts while struggling with untrained languages.

### 5.2.3 Groovy to JShell Translation

As the majority of Eurosoftware's projects are developed in Java, integrating and using modern programming tools and languages represents a strategic approach to fostering innovation and efficiency in its development practices. The ability of GPT-4 to translate Groovy code into JShell code is proving to be a valuable advantage in this context, as it promises to streamline the development processes for Java-centric projects. This conversion task highlights GPT-4's potential to facilitate seamless language transitions, which is particularly beneficial for Eurosoftware's Java-focused ecosystem.

**Details:** This initiative aimed to convert Groovy scripts into JShell commands, balancing Groovy's flexibility with Java's structured approach for exploratory coding. Details and outcomes are documented in the `Groovy_to_JShell.txt`, located in the `Results\gpt4_0\Groovy to JShell Translation` directory.

**Outcome:** The task of translating Groovy scripts into JShell with ChatGPT 4 showed mixed results, reflecting the complex nature of code conversion across different programming environments. The AI performed a commendable initial translation without full context but struggled with more complex scripts, often leaving significant portions of the code inadequately adapted. Success varied between examples; some translations showed partial progress, but none achieved complete accuracy, highlighting ChatGPT 4's challenges with certain Groovy constructs and the complex requirement for semantic as well as syntactic understanding when translating between these languages.

### 5.2.4 Java to Python Translation

Following the challenges encountered in translating Groovy scripts to JShell, an intriguing question arises: how would GPT-4 handle the task of translating code from Java, a statically typed language with a rich ecosystem, to Python, known for

its dynamic typing and readability? This task extends beyond the specific needs of Eurosoftware, venturing into a broader exploration of GPT-4's capabilities in code translation across significantly different programming paradigms.

**Details:** The process and outcomes of this translation exercise are detailed in the `Results\gpt4_0\Java To Python Translation` directory, within the next `Java_to_Python.txt` file.

**Outcome:** GPT-4's proficiency in translating Java code into Python was impressively demonstrated in a series of examples, demonstrating its ability to accurately convert and explain code between these popular programming languages. From simple tasks to more complex multi-class scenarios, GPT-4 consistently delivered accurate Python equivalents of Java code, complete with insightful explanations. This capability not only confirms GPT-4's adeptness at handling diverse coding challenges, but also underscores its potential as a valuable tool for developers, especially those transitioning between Java and Python. The AI's fluent navigation through programming constructs and its effectiveness in maintaining the logical and functional integrity of the original code underscores its advanced translation capabilities.

## 5.2.5 XHTML to Vue.js Translation

It is important to examine the transition from XHTML to Vue.js, building upon the previous examples of Groovy and Java. This transition not only tests GPT-4's versatility across programming languages but also its effectiveness in modern web development frameworks such as Vue.js. Such a transition underscores the importance of evaluating how well GPT-4 can handle the dynamic and responsive elements that are essential in today's web technologies.

**Details:** The translation process from XHTML to Vue.js and the resulting outputs are documented in the `Results\gpt4_0\XHTML to Vue Translation` directory, specifically within the `XHTML_to_Vue.txt` file.

**Outcome:** Attempts to translate Eurosoftware's custom XHTML into Vue.js using GPT-4 consistently failed, revealing the AI's limitations in handling highly customized template conversions. Despite various efforts, GPT-4 was unable to produce functional Vue.js templates from the provided XHTML structures. This highlighted the need for a neural network specifically trained on Eurosoftware's unique XHTML and Vue.js configurations to achieve accurate translation and effective adaptation to modern web technologies.

## 5.2.6 Conclusion

ChatGPT 4.o demonstrated significant improvements in code interpretation and refactoring, excelling at improving clarity and suggesting optimizations for complex Java and other popular languages. This underscores its potential as a useful tool for developers seeking to improve code quality and maintainability. However, its limitations became apparent in tasks such as translating specialized XHTML to Vue.js and optimizing less common languages such as DO DSL, indicating challenges with specialized or uncommon frameworks.

On the other hand, ChatGPT 4.o proved adept at translating between popular programming languages such as Java and Python, demonstrating its ability to effectively capture and translate basic programming constructs and semantics. However, its mixed results in translating between different programming paradigms - such as Groovy to JShell and XHTML to Vue.js - underscore the complexity involved in preserving the intricacies and specifics of the original code during the translation process. These findings suggest that while ChatGPT 4.o is a step forward in AI-assisted software development, further enhancements and specific training are essential to fully realize the potential of AI in different development environments.

## 5.3 TabNine: Enhancing Development with AI

While ChatGPT 4.o has introduced significant advances in AI-powered coding, it's important to also recognize tools like TabNine. This section explores TabNine's practical application in development workflows, highlighting its effectiveness in improving coding speed and simplicity. By leveraging AI to understand coding patterns and make suggestions across multiple programming languages and IDEs, TabNine provides a versatile and immediate solution to everyday coding challenges, building on the fundamental advances of AI in the coding space.

### 5.3.1 Evaluating TabNine with TypeScript

Given the prominence of TypeScript in modern web development for its ability to provide static typing for JavaScript projects, it is particularly relevant to evaluate the impact of AI-enabled coding tools in a TypeScript environment. Such tools have the potential to significantly reduce development time and improve code quality by providing accurate coding suggestions and automating routine tasks.

**Details:** The evaluation of TabNine's integration into the TypeScript development workflow has been captured in three MP4 video files and is available in the



Results\TabNine\TypeScript directory. These videos, named `carwashTable.mp4`, `getFunctionContextById.mp4`, and `carwashTable.mp4`, provide a dynamic visual representation of TabNine's code completion capabilities in various TypeScript scenarios. Accompanying these videos, detailed descriptions are also available in consequently named files in the same directory, providing further insight into each example presented.

**Outcome:** The evaluation of TabNine in TypeScript projects demonstrates its effectiveness in improving software development through intelligent code completion. During various coding tasks, TabNine skillfully suggests relevant code snippets, facilitating faster and more efficient coding processes. This capability is particularly useful in complex coding scenarios, where TabNine adapts to developer styles and project-specific requirements with minimal guidance, significantly accelerating development while ensuring code accuracy and contextual relevance.

### 5.3.2 Evaluating TabNine with Vue.js Projects

As Vue.js gains traction within Eurosoftware's technology stack due to its reactivity and component-based architecture, there is a growing interest in evaluating how AI-enabled tools like TabNine can improve coding practices within Vue.js projects. The intuitive and adaptive nature of Vue.js provides a fertile ground for such AI-powered code completion tools that could help accelerate development workflows and improve code quality.

**Details:** The evaluation of TabNine's functionality in Vue.js coding sessions is thoroughly captured in a series of videos that demonstrate the tool's real-time code completion capabilities. These videos, located in the Results\TabNine\Vue directory and named as `giftCertificateButton.mp4`, `priceParameter.mp4`, and `giftCertificateInputField.mp4`, provide detailed examples of TabNine's effectiveness in integrating with Vue.js components. To analyze each scenario more deeply, an accompanying text files named accordingly are available in the same directory that provides detailed analysis alongside the respective videos.

**Outcome:** Throughout the evaluation of TabNine in a Vue.js environment, its performance showed mixed results. TabNine excelled at simple coding tasks such as adding new variables and parameters, demonstrating that it can effectively support basic programming needs. However, it encountered significant challenges with more complex visual layout tasks involving Vue.js, such as accurately predicting unique element IDs generated by specific Eurosoftware protocols and providing useful layout suggestions. These struggles underscored a gap in TabNine's ability to handle



tasks that require deep integration with custom Vue.js environments. Overall, while TabNine proved helpful in some aspects, the evaluation highlighted its limitations in areas requiring a detailed understanding of complex and visually oriented coding structures, suggesting a need for further refinement to meet higher expectations.

### 5.3.3 Conclusion

Exploring TabNine's integration into software development workflows, particularly in TypeScript and Vue.js environments, highlights its potential to improve coding efficiency and accuracy. In TypeScript projects, TabNine excelled at providing sophisticated code completions that were well aligned with developer intent, thanks to its AI-powered engine trained on large datasets. This capability allowed it to provide logical code snippets that significantly increased productivity, as evidenced by demonstration videos showing the tool deftly handling API and additional function implementations.

In Vue.js projects, however, TabNine struggled with the framework's visual and layout-specific tasks, struggling with Vue.js' reactive system and Eurosoftware's unique conventions. These limitations illustrate the difficulties AI-powered tools face with complex web development frameworks and customizations. While TabNine excels in statically typed languages such as TypeScript, improving the speed and quality of development, its performance in Vue.js suggests the need for ongoing enhancements to the AI model to better support the diverse and dynamic aspects of modern web development. These findings underscore TabNine's role as a transformative tool in AI-powered coding, promising significant benefits for companies like Eurosoftware by enabling faster development cycles and improved code quality as the technology matures.

## 5.4 GitHub Copilot: A Revolution in AI-Assisted Coding

This section explores how GitHub Copilot integrates into developers' workflows, highlighting its role in transforming code production from a manual and sometimes tedious process into a more efficient and creative endeavour. Harnessing the power of AI, GitHub Copilot does more than just complete code; it provides contextual recommendations and understands comments to generate code snippets, making it an indispensable tool for developers across programming languages and environments.

## 5.4.1 GitHub Copilot with TypeScript

As mentioned previously, TypeScript is becoming increasingly important in Eurosoftware, reflecting its growing popularity across the technology industry. The increased use of TypeScript highlights the importance of AI-powered coding wizards such as GitHub Copilot. GitHub Copilot is a valuable tool for developers working with TypeScript, helping to streamline workflows and increase efficiency by providing accurate coding suggestions and automating routine tasks.

**Details:** The investigation into the effectiveness of GitHub Copilot for TypeScript development was meticulously documented, following the methodology previously used in the TabNine evaluations. The results are presented in a series of video recordings available in the `Results\GitHub Copilot\TypeScript Completion` directory. These recordings include `carwashTable.mp4`, `clientSecurityApi.mp4`, and `const.mp4`, which illustrate GitHub Copilot's ability to provide relevant code suggestions for various TypeScript coding scenarios. Additionally, each video is accompanied by a final text overview files named respectively that provides a detailed analysis of GitHub Copilot's performance in each scenario, further enriching the evaluation context.

**Outcome:** The evaluation of GitHub Copilot in a TypeScript environment demonstrated its exceptional ability to streamline the software development process by effectively interpreting comments to generate complex code structures. The tool consistently demonstrated advanced understanding and implementation capabilities, successfully automating repetitive coding tasks and significantly improving coding efficiency. Despite some initial challenges, Copilot quickly adapted, proving its usefulness in a variety of programming scenarios and solidifying its role as a powerful tool for developers.

## 5.4.2 GitHub Copilot in Java Development

With Java serving as a cornerstone of Eurosoftware's technology stack, the introduction of GitHub Copilot into the development process represents a leap forward in using AI to improve coding efficiency. Eurosoftware approved the use of specific Java code snippets for testing, carefully selected to minimize any potential risk to the company. These snippets were uploaded to a private GitHub repository, allowing for a safe and comprehensive evaluation of GitHub Copilot's capabilities within the Java development context.

**Details:** Videos demonstrating how GitHub Copilot improves Java development can be found in the `Results\GitHub Copilot\Java Completion` directory. These

videos demonstrate its usefulness in various coding scenarios, with corresponding detailed descriptions available in the same directory to ensure a full understanding of each demonstration.

**Outcome:** Throughout the evaluation series, GitHub Copilot demonstrated a robust capability in the Java development environment, handling a variety of programming tasks with finesse - from generating logical methods and refining user interfaces to advanced code refactoring. Whether adding methods to simplify processes, creating logical opposites for existing functions, or integrating user interface elements, Copilot not only adapted to the task at hand but also excelled at providing precise, contextually appropriate code suggestions. This performance underscores its potential as a transformative tool for accelerating development workflows and improving code quality, solidifying its status as a valuable asset for Eurosoftware's developers.

### 5.4.3 **Vue.js code generation with GitHub Copilot Chat**

Given the important role Vue.js plays in Eurosoftware's front-end development, it is crucial to explore the capabilities of GitHub Copilot Chat for Vue.js code generation. In particular, this exploration focused on evaluating how well GitHub Copilot Chat could automate and improve the creation of Vue.js components.

**Details:** The assessment process and outcomes were documented, reflecting the interaction with GitHub Copilot Chat for generating Vue.js code. This documentation, stored under the path `Results\GitHub Copilot\Vue Chat` in the file named `chatVue.txt`.

**Outcome:** It became clear that the AI tool struggled to accurately interpret and implement Vue.js development conventions during various tasks to evaluate Copilot Chat's ability to handle Vue.js components derived from XHTML templates. Copilot Chat often missed critical details and requirements, whether it was simplifying component modifications, adding elements, or making universal changes across multiple variations. These challenges, particularly within the Vue.js framework, highlight the limitations of current AI tools in handling tasks that require a deep understanding of the nuances of UI development. Despite the potential to increase productivity, the evaluation showed that developers must maintain a hands-on approach, thoroughly reviewing and refining AI-generated code to ensure alignment with project specifications and overall quality standards.

## 5.4.4 GitHub Copilot Chat for TypeScript Development

This evaluation took place within the context of an existing setup, using a personal, private GitHub repository previously used for similar explorations. This strategy allowed for a direct comparison of the effectiveness of GitHub Copilot Chat in the context of TypeScript, without changing the original test environment, and allowed for a smooth evaluation process. The goal of this methodical approach was to determine how GitHub Copilot Chat could benefit Eurosoftware's TypeScript projects.

**Details:** The evaluation process and its results have been meticulously written up, highlighting the use of GitHub Copilot Chat in the context of TypeScript development. These reports are stored in the `Results\GitHub Copilot\TS Chat` directory, specifically in a file called `chatTypeScript.txt`.

**Outcome:** GitHub Copilot demonstrated a remarkable ability to manage TypeScript logic and backend tasks, excelling at generating complex class structures, accurately calling and implementing API functionality, and analyzing class structures to fill in necessary code. These tasks, which ranged from creating new classes that mirrored existing ones to implementing missing API calls, highlighted Copilot's ability to interpret instructions, navigate codebases, and efficiently deliver precise, functional code. This performance underscores the potential of AI tools like GitHub Copilot to increase productivity and streamline the development process by providing robust code solutions that meet specific development criteria.

## 5.4.5 Conclusion

Exploring GitHub Copilot's usefulness in software development, specifically in Eurosoftware's selected Java, TypeScript, and Vue.js contexts, revealed its capabilities and limitations. In the Java and TypeScript environments, GitHub Copilot excelled, demonstrating a robust ability to understand complex code structures and generate logical solutions that closely match developer intent. This performance underscores its potential as a critical coding assistant, capable of significantly reducing manual coding effort and allowing developers to focus on more strategic aspects of software development. It has proven particularly adept at navigating the back-end logic and algorithmic complexity of Java and TypeScript projects, assisting with tasks such as method generation, code refactoring, and API integration.

Conversely, its use in Vue.js development revealed challenges, particularly in UI development and handling Vue.js' specific reactive patterns. Although GitHub Copilot was effective at generating code snippets and assisting with JavaScript logic, it was less adept at handling the declarative nature and component-based architecture

of Vue.js. These results highlight the challenges AI-powered tools face in adapting to the diverse and evolving landscape of web development frameworks. Despite these hurdles, the overall impact of GitHub Copilot at Eurosoftware is overwhelmingly positive, indicating significant progress toward intelligent coding assistance. Nonetheless, the limitations observed in Vue.js underscore the complexity of fully automating the coding process, especially in areas that require deep domain knowledge and a nuanced understanding of modern front-end development.

## 5.5 Common Conclusion

The exploration of AI-enabled development tools reveals a landscape where capabilities and growth potential coexist, providing a nuanced view of how artificial intelligence is shaping the field of software development. This section is intended to provide a concise overview of the research findings, with an emphasis on the practical applications and efficiencies that various AI tools offer for software development tasks.

### 5.5.1 GPT-3.5

The analysis begins with ChatGPT 3.5. Table 5.1 summarizes its performance in software development tasks, offering a clear perspective on its strengths and areas for enhancement.

Study Area	Rating	Description
Explaining SQL	Good	Demonstrated strong capability in explaining complex SQL scripts with insightful interpretations.
Optimizing SQL	Moderate	Showed potential in SQL query optimization with varied performance; excelled in simpler scripts.
Explaining Lambda Expression	Good	Excelled in interpreting lambda expressions, providing clear explanations. Mixed results in complexity handling.
Optimizing Lambda Expression	Moderate	Demonstrated beneficial optimizations, especially in readability; faced challenges in generating fully buildable code.
Lambda Expression generation	Good	Showed capability in generating lambda expressions, with successful outcomes in straightforward scenarios.

Table 5.1: Evaluation of ChatGPT 3.5 in Software Development Tasks

## 5.5.2 ChatGPT 4

The examination progresses to GPT-4, exploring its advancements and distinguishing features. The subsequent Table 5.2 outlines the assessment of GPT-4, underscoring its improved proficiency in handling complex coding tasks.

Study Area	Rating	Description
Code Interpretation - DO DSL	Moderate	Offered basic yet insightful overview for unfamiliar DO DSL, struggling with detailed interconnectivity and full comprehension.
Code Interpretation - Java	Excellent	Demonstrated commendable precision and insight in Java, articulating complex relationships and best practices seamlessly. Was able to generate the pieces of code that were requested.
Advanced Code Refactoring - Java	Excellent	Showed adeptness in navigating and improving complex Java code, with targeted and meaningful optimizations.
Advanced Code Refactoring - DO DSL	Poor	Encountered significant difficulty with DO DSL syntax and failed to apply useful refactoring, highlighting limitations with less common languages.
Groovy to JShell Translation	Moderate	Managed some level of Groovy to JShell translations, but struggled with complex or context-heavy scripts.
Java to Python Translation	Excellent	Excellently handled Java to Python conversions, maintaining logic and functionality with detailed explanations.
XHTML to Vue.js Translation	Poor	Faced significant challenges in translating customized XHTML to Vue.js.

Table 5.2: Evaluation of GPT-4 in Software Development Tasks

## 5.5.3 TabNine

The capabilities of TabNine are then examined, focusing on its code completion and suggestion features. The ensuing Table 5.3 summarises the findings, illustrating TabNine's potential as a valuable asset for developers seeking enhanced efficiency and precision.

Study Area	Rating	Description
TypeScript - API Integration	Excellent	Demonstrated strong code completion capability. Shown in <code>clientSecurityApi.mp4</code> .
TypeScript - Additional Functions	Good	Provided accurate coding suggestions after minor adjustments, showcasing adaptability. Efficiency demonstrated in <code>getFunctionContextById.mp4</code> .
TypeScript - Complex Implementation	Excellent	Proved to be a robust tool in handling complex TypeScript coding tasks. Demonstrated in <code>carwashTable.mp4</code> .
Vue.js - Simple Coding Tasks	Good	Handled simple Vue.js coding requirements adequately. Shown in <code>priceParameter.mp4</code> .
Vue.js - Layout and Visual Tasks	Poor	Struggled significantly with visual layout tasks in Vue.js. Issues were prominent in <code>giftCertificateInputField.mp4</code> and in <code>giftCertificateButton.mp4</code> .

Table 5.3: Evaluation of TabNine in Software Development Tasks

## 5.5.4 GitHub Copilot

Table 5.4 summarises GitHub Copilot's performance in software development tasks, identifying its strengths and limitations.

Study Area	Rating	Description
GitHub Copilot with TypeScript	Excellent	Excelled in providing accurate TypeScript code suggestions, significantly streamlining workflows.
GitHub Copilot in Java Development	Excellent	Demonstrated robust capabilities in Java, handling a range of programming tasks with finesse.
Vue.js code generation with GitHub Copilot Chat	Poor	Struggled with the nuances of Vue.js UI development, often missing critical details.
GitHub Copilot Chat for TypeScript Development	Excellent	Managed TypeScript backend tasks proficiently, showing strong performance in code generation and API integration.

Table 5.4: Evaluation of GitHub Copilot in Software Development Tasks

# Custom AI Model for Enhanced Knowledge Management

## 6

As we delve into the practical applications of general AI tools in software development, it becomes apparent that while tools like ChatGPT, TabNine, and GitHub Copilot offer significant advantages, they might not fully cater to the unique needs of every organization. This realization brings us to Eurosoftware's innovative approach to harnessing AI for internal knowledge sharing, especially in light of their recent expansion and the challenges it posed.

Eurosoftware's rapid expansion resulted in a large number of new developers, each faced with the difficult task of navigating the company's complex application architecture. This complexity was made worse by the fact that documentation was spread across Confluence, making it a real challenge for newcomers to quickly get up to speed and become productive. This situation highlighted a critical deficiency: the lack of an AI model that understood the nuances of Eurosoftware's architectural intricacies, a gap that general AI tools on the market could not fill.

## 6.1 Enhancing Training Data Management with a GUI Application

The development of AI models at Eurosoftware demands sophisticated management of training data due to the dynamically updating Confluence base that supports Eurosoftware's knowledge management. To maintain the neural network's effectiveness, the frequently updated training data must adhere to a JavaScript Object Notation Lines (JSONL) format. This format, structured as individual JSON objects per line, enhances the efficiency of data processing and is crucial for network training:



```

1 {
2 "input": "User: Question: <question>?\n\nAssistant:",
3 "output": "<answer>"
4 }

```

This structure supports scalability by handling large datasets effectively, ensuring compatibility with diverse machine learning tools and frameworks, crucial for integration with technologies like NVIDIA's Lamma and Megatron.

## 6.1.1 User Interface Components

The GUI of the JSONL Editor application is thoughtfully designed to enhance user experience through its comprehensive and interactive interface components. The following figure 6.1 shows the key components:



Figure 6.1: General Overview of the JSONL Editor Interface

The application interface is divided into several functional areas, each contributing to the streamlined management and editing of jsonl files.

**File Management Buttons.** File Management buttons, shown in figure 6.2, provide the basic functionality for opening existing jsonl files, saving current work, and saving the work as a new file, thus facilitating the easy management of jsonl files.

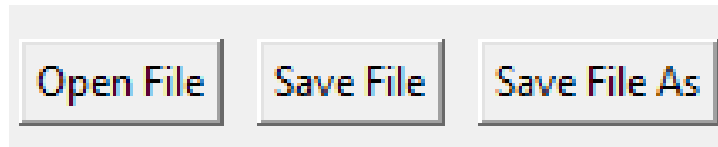


Figure 6.2: File Management Buttons: Open, Save, and Save As

**Data Management Buttons.** Dedicated buttons illustrated in figure 6.3 for adding new entries or deleting existing ones enhance dynamic data management within the application, enabling users to efficiently update the training data.

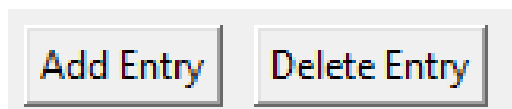


Figure 6.3: Data Management Buttons: Add and Delete Entries

**Adding a New Entry.** To add a new entry, users can click on the "Add Entry" button located in the main interface. This action opens a modal window, shown in figure 6.4, designed specifically for the input of new questions and answers.

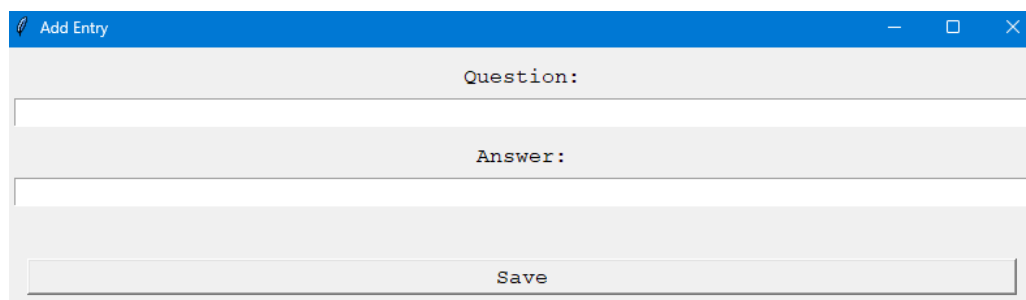
A modal window titled 'Add Entry' with a blue header bar. The window contains three input fields: 'Question:', 'Answer:', and 'Save'. The 'Question:' and 'Answer:' fields are text boxes, and the 'Save' field is a button. The window has standard window controls (minimize, maximize, close) in the top right corner.

Figure 6.4: Modal Window for Adding a New Entry

**Editing an Existing Entry.** Editing an existing entry is facilitated by a double-click on the desired entry within the interactive treeview. This action triggers the opening

of a modal window similar in design to the "Add Entry" window but with the title indicating that it is for editing rather than adding a new entry.

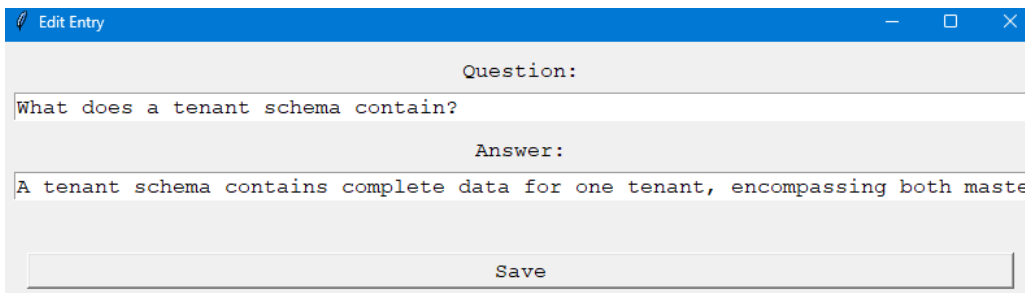


Figure 6.5: Modal Window for Editing an Existing Entry

By providing distinct modal windows for adding and editing entries, the application ensures a seamless and intuitive user experience, facilitating the accurate and efficient management of training data.

## 6.1.2 Hotkeys in the Main Application

The main application interface incorporates several hotkeys to facilitate efficient navigation and data management. The following Table 6.1 outlines these hotkeys and their respective actions:

Hotkey	Action
Enter	Edit the selected entry
Double Click	Edit the selected entry
Del	Delete the selected entry
Ctrl + S	Save the current file

Table 6.1: Hotkeys in the Main Application

## 6.1.3 Hotkeys in Modal Windows

Modal windows, used for adding or editing entries, support hotkeys to streamline the data entry and editing process. The following Table 6.2 describes the hotkeys available within these modal windows:

Hotkey	Action
Esc	Close the modal window without saving changes
Enter	Save the changes and close the modal window

Table 6.2: Hotkeys in Modal Windows

These hotkeys are designed to enhance user experience by allowing for quick and easy navigation and manipulation of data within both the main application interface and the modal windows. Implementing these hotkeys reflects the application's commitment to efficiency and ease of use.

## 6.1.4 Data Validation and Management

The integrity of the jsonl file is paramount for the effective training of AI models. To this end, the application incorporates a comprehensive validation mechanism to ensure that each entry not only adheres to the required format, with specific placeholders for questions and answers, but also verifies the overall file structure upon opening.

**Initial File Format Validation.** Upon attempting to open a file, the application first checks that the selected file has the .jsonl extension, indicating its potential compatibility with the required data format.

This process includes verifying the presence of essential keys ('input' and 'output') within each JSON object and ensuring that the structure of these objects aligns with the format expectations for AI model training.

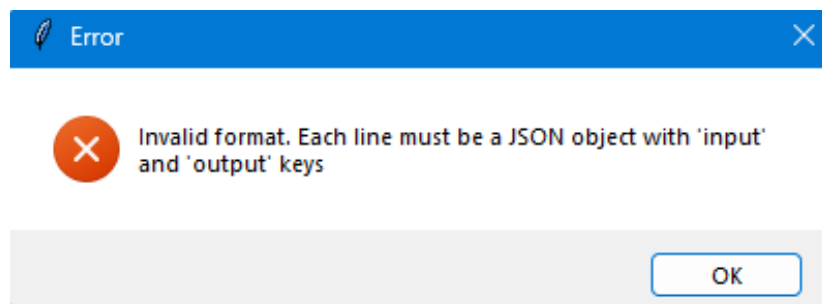


Figure 6.6: Example of an Error Message During File Validation

**Validation During Entry Creation and Editing.** The application also enforces validation checks when creating a new entry or editing an existing one. This ensures that the 'input' field is not left blank and that, upon substitution of the question and answer into the placeholders, the resulting JSON object conforms to the expected jsonl format. This measure prevents accidental or deliberate disruptions to the data structure, safeguarding the integrity of the training dataset. This form of validation is crucial for maintaining a consistent and error-free data format, which in turn, supports the efficient training of AI models.

## 6.1.5 Handling Unsaved Changes

To prevent data loss, the application continuously monitors for unsaved changes, providing alerts when attempting to open a new file or exit the application. This feature is essential for maintaining data integrity and minimizing the risk of accidental data loss.

In addition to monitoring for unsaved changes within the application, a safeguard is implemented to address scenarios in which the file being edited has been modified by another program or by a user outside of the JSONL editor. If an attempt is made to save changes to a file that has been modified since it was last opened or saved within the application, the user is immediately notified of the conflict. This notification highlights the risk of overwriting changes made by others since the file was last saved in the editor.

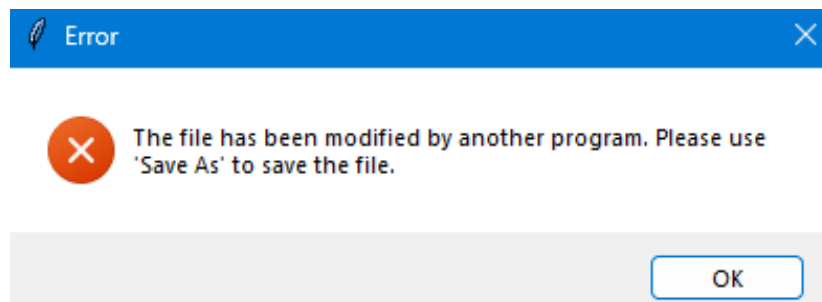


Figure 6.7: Warning Message Indicating External Modifications

Under these circumstances, the application prevents direct saving to the existing file to avoid unintentional data loss. Instead, users are directed to use the "Save File As" option to consciously decide whether to overwrite the externally modified file or save their work as a new document. This step requires explicit user action, ensuring that any decision to overwrite external changes is made deliberately, thereby safeguarding against accidental loss of important data modifications.

## 6.1.6 Technical Details

The application is engineered using Python, a versatile programming language renowned for its simplicity and the vast ecosystem of libraries. Tkinter, Python's standard GUI library, is chosen for its ease of use and the ability to rapidly develop desktop applications. The core structure of the application extends `tk.Tk`, initiating a window with a set of features tailored for editing jsonl files, including file manipulation, data entry, and visualization.

## 6.2 Training the Model: An Iterative Approach

The development and refinement of AI models is an integral part of advancing technological capabilities and solving complex problems in various industries. In this context, the management and preparation of training data play a critical role in determining the effectiveness and efficiency of AI models. This paper provides a comprehensive overview of the iterative process of training an AI model at Eurosoftware, highlighting the importance of data preparation and optimization.

### 6.2.1 First Iteration: Exploratory Phase

The first iteration of AI model training at Eurosoftware was not intended for direct model training but as an exploratory phase. This phase was crucial for evaluating the functionality of the JSONL Editor application, which was designed to simplify the preparation of training data for neural network training. In addition, this iteration aimed to establish a basic set of training examples that would provide fundamental insights into the architecture of the Eurosoftware applications.

**Objectives.** The primary objectives for this exploratory phase included:

- Verify the operational efficiency of the JSONL Editor to facilitate preparing training data.
- Prepare a basic set of training samples encapsulating basic data points to explain the architecture of the Eurosoftware applications.
- Evaluate the complexity involved in the preparation of the training data.

**Data Preparation.** This iteration selected six small Confluence pages describing the company's basic architecture and core coding standards for data preparation. The process involved creating approximately 2-3 training examples per sentence in the text, following the principle that each example should be a question and answer in the correct JSONL format.

**Results and Observations.** The outcomes of the first iteration are summarized in the following Table 6.3, indicating the page name and the number of prepared training samples:

A total of 370 training examples have been created and can be found in the `Input_data\Iteration 1` directory.

Page Name	Training Samples
Big Pictures	32
Storemanager	36
Cluster Infrastructure Service (CIS)	38
Enterprise Connector (ECON)	76
First Steps With GK Retail	77
Development Rules	111

Table 6.3: Summary of Training Samples Prepared (Iteration 1)

**Conclusion.** This iteration was considered successful because the JSONL editor significantly streamlined the data preparation process. It effectively ensured format correctness and allowed the focus to be on content preparation. Although the creation of 370 training samples was a significant achievement, it underscored the relativity of "enough data" in neural network training and highlighted the complexity of preparing a high-quality dataset. The successful establishment of a foundational base through these samples confirmed the utility of the JSONL editor and provided valuable insights into the methods and tools used, setting a clear path for dataset expansion and refinement in future iterations.

## 6.2.2 Second Iteration: Preparatory Steps for Future Training

The second iteration was a foundational phase in Eurosoftware's AI model development trajectory. Unlike the preliminary phase, this phase focused on the necessary preparations for the upcoming training phase. Eurosoftware dedicated a dedicated server to this project and received guidance from Pascal Singer, a GK expert in neural network training. Singer's expertise was crucial in configuring the training environment and securing the necessary access rights for the developer, ensuring a smooth transition to the server infrastructure.

**Objectives.** The objectives for the second iteration were multi-faceted and aimed at laying a more solid foundation for the AI model's training:

- To acquaint with the server provided by Eurosoftware and understand the steps required to commence training.
- To prepare a new set of training data and accompanying validation data to prevent model overfitting.

**Data Preparation.** Following Pascal's advice, the focus shifted towards preparing both the training and validation datasets. The first iteration's data was put aside

for future review, and effort was directed towards creating new training samples that would provide a deeper insight into Eurosoftware's architecture. The validation dataset was specifically designed to prevent overfitting by slightly modifying training samples to include synonyms or combinations of different samples. The Table 6.4 below summarizes the prepared data:

Architecture	Training Samples	Validation Samples
Label & Poster Printing	47	9
Offline Mobile Store	53	11
Self Scanning	64	15
Store Management	85	17
Multitenancy	114	22
Store Device Control	151	43
Omnichannel Point-of-Sale	188	33

Table 6.4: Summary of Training and Validation Samples Prepared (Iteration 2)

A total of 702 training samples and 150 validation samples were prepared, all of which can be found in the attachments under the: `Input_data\Iteration 2` path.

**Validation Data Preparation.** The creation of the validation dataset was a critical step designed to prevent model overfitting. The validation samples were constructed by slightly altering the training samples to include synonyms or by combining elements from different training samples, thereby ensuring that the model can generalize well beyond the training data. This approach ensures that the validation data closely mirrors the training data but differs enough to effectively test the model's ability to apply learned knowledge to new situations.

For instance, a training sample might be presented as follows in source code 6.1 example:

---

Source code 6.1: Training Sample Example

---

```

1 {"input": "User: Question: What type of communication is used
2 by central applications?\n\nAssistant:",
3 "output": "Central applications communicate over JMS
4 provided by CIS."}
```

---

A corresponding validation sample would be slightly altered to include synonyms, as shown below in source code 6.2 example:

---

Source code 6.2: Validation Sample Example

---

```

1 {"input": "User: Question: What is the primary method of
2 communication for central applications?\n\nAssistant:",
```



```
3 "output": "The primary method of communication for  
4 central applications is through JMS provided by CIS."}
```

---

**Conclusion.** The completion of the second iteration significantly advanced the AI model's training at Eurosoftware by expanding the dataset with 702 training and 150 validation samples. This phase also marked the initiation of using a Eurosoftware-provided server under Pascal Singer's guidance, setting a solid foundation for the actual training process. The focus on the critical importance of validation data during this iteration established a clear path forward for the development and refinement of the model.

### 6.2.3 Third Iteration: Technical Implementation

This iteration focuses on deeply understanding basic neural network training by working with specific libraries and tools to build a solid training and evaluation system.

**Objectives.** The objectives for this phase were clearly defined to ensure a structured and efficient approach to model training:

- Familiarize with the libraries and tools necessary for neural network training.
- Prepare scripts for training and evaluating the model.
- Select an appropriate metric for model evaluation.

**Tools and Libraries.** In addition to the server provided by Eurosoftware, many Python libraries and tools were used to facilitate the development and execution of the project. The selection of tools was tailored to effectively support large-scale neural network training:

- **PyTorch Lightning:** Used for organizing the training process, making it more structured and easier to manage. Its compatibility with distributed training environments was crucial for training large models efficiently.
- **Nvidia's CUDA 12.2 and CUDNN 8:** Enabled GPU acceleration, crucial for handling computationally intensive tasks involved in training and evaluating neural networks.
- **Nvidia Apex:** Provided functionalities for mixed precision training, allowing for faster computation and reduced memory usage while maintaining model accuracy.

- **Nvidia Transformer Engine and NeMo Toolkit:** Offered advanced capabilities for building and deploying transformer models, including model parallelism and efficient training techniques.
- **GCC12-C++:** Ensured compatibility and optimal performance for compiling and executing C++ extensions used by the PyTorch and Nvidia libraries.

These tools and libraries constituted the foundational elements of the project's technical framework, enabling the creation, training, and evaluation of complex neural network models. Inspired by the open-source resources provided by Nvidia, custom scripts were developed to integrate these advanced technologies seamlessly.

**Model Evaluation Metrics.** The initial model evaluation strategy was based on using exact string matching as the primary metric. While straightforward, this approach quickly proved unsuitable for capturing the nuanced nature of natural language processing. For example, consider the question "What does CIS stand for?" and its possible answers:

- 1 Answer 1: CIS stands for Cluster Infrastructure Server.
- 2 Answer 2: Cluster Infrastructure Server.

Both responses are correct, but exact string matching would recognize only one as valid, ignoring the semantic similarity of the responses. This realization prompted a shift to a more flexible and comprehensive evaluation metric.

The F1 score, which harmonizes precision and recall into a single metric, was chosen as an appropriate metric for evaluating model-generated responses, capturing both false positives and false negatives for a comprehensive assessment of model performance. In addition to this quantitative measure, a manual review process was in place as a qualitative assessment, providing insight into the model's interpretive capabilities and the quality of its responses. This dual approach of using the F1 score and manual reviews provided a robust system for accurately determining the results of the neural network models.

**Configuration.** The project used a standard configuration that was customized to optimize the neural network training process. This configuration was critical to effectively manage hardware resources and ensure efficient execution of training sessions. Inspired by Nvidia's open-source examples, some configurations were adapted and customized to meet the specific needs of our models.

The complete configuration, with detailed comments explaining each setting, is available in the `Application_and_libraries\Eursoftware's Custom AI\conf` directory.

**Training and Evaluation Scripts.** Two primary Python scripts, *builder.py* and *interface.py*, were developed to manage the complex processes of training and evaluating the neural network models. These scripts, inspired by and adapted from Nvidia's open-source repositories, form the operational core of the project, facilitating a streamlined workflow for both model training and performance evaluation.

**builder.py.** This script encapsulates the functionality needed to configure and initiate the model training and evaluation processes. It uses PyTorch Lightning to set up the training environment, utilizing plug-ins and strategies for distributed training and mixed accuracy. Key components of the script include

- Configuration modification functions to adjust model settings based on training needs, including aspects such as model parallelism, batch sizes, and checkpointing strategies.
- The *prepare\_trainer* function, which initializes the PyTorch Lightning Trainer with specific plugins and strategies for efficient training execution.
- Model recovery and adaptation functions for loading pre-trained models and applying prompt tuning, adapter, and other fine-tuning techniques, taking advantage of the NeMo collection's extensive support for language model training.

**interface.py.** This script serves as a user interface for performing training and evaluation tasks, parsing command line arguments to configure the training session (e.g., selecting between the LLaMA and Megatron models, specifying the training mode). It provides a straightforward mechanism for users to interact with the training framework and initiate the training or evaluation process based on the specified configurations.

Scripts are available in `Application_and_libraries\Eursoftware's custom AI` directory. This directory contains not only the scripts themselves but also extensive readme file on how to use them.

**Conclusion.** This phase marked significant progress toward the initial goals by integrating specialized tools and libraries that facilitate a structured approach to neural network training and evaluation. The creation of custom scripts inspired by Nvidia's open-source contributions significantly streamlined the operational workflow, improving the precision of model training and nuanced performance evaluation. A

dual evaluation strategy, combining F1 scoring with manual review, was adopted to effectively address the complexity of evaluating neural network output and ensure a thorough understanding of model capabilities. Details for replicating this process and advancing research are documented in the Custom AI\Iteration 3 directory.

## 6.2.4 Fourth Iteration: Initial Model Training

With the technical foundation established in the previous iterations, the focus of the fourth iteration shifted to the actual training and evaluation of the neural network models. This phase was critical for evaluating the effectiveness of the prepared training and validation data sets, as well as the selected models and configurations.

**Objectives.** The objectives for this phase were designed to advance the project's progress through practical application and assessment:

- Train and evaluate the Megatron and LLaMA models using the data prepared in the second iteration.
- Combine the data from the first and second iterations and assess the performance of both Megatron and LLaMA models on this aggregated dataset.

**Training Process.** Leveraging the developments of the third iteration, training sessions for both models began, with careful monitoring to optimize the use of computational resources. Two models were trained over six sessions, averaging eight hours per session.

The training involved two models across multiple configurations, with varying results:

- The first LLaMA model was trained on second iteration data with a validation step of 1000, utilizing the Lora PEFT scheme. It achieved an F1 score of 0.4427. However, manual evaluation yielded negative feedback, indicating the expected results were not met.
- The second session involved the Megatron model under identical configurations but showed a decrease in performance with an F1 score of 0.4285. Manual evaluations confirmed the unsatisfactory outcomes.
- A third attempt with the LLaMA model, reducing the validation step to 200, noted an improvement in the F1 score to 0.4619. Manual checks showed better, yet still not satisfactory, results.

- The fourth training session applied the same reduced validation step to the Megatron model, resulting in a slightly improved F1 score of 0.4485, which still did not meet the desired benchmarks.
- Switching strategies, a Megatron model trained using PTuning with a validation step of 200 yielded an F1 score of 0.4103. Manual evaluation was negatively received.
- A similar PTuning approach on the LLaMA model produced a marginally better F1 score of 0.4495, but manual assessments remained negative.

Given these results, it was concluded that the LLaMA model, utilizing the Lora PEFT scheme with a validation step of 200, performed the best among all configurations. This setup achieved the highest F1 score of 0.4619. Manual evaluation, supported by a selection of responses generated by the model, provided a nuanced view of its performance. These model's outputs can be found in Appendix in Listing C.7.

The first three responses, while closely reflecting the wording of the questions, can be considered correct, demonstrating the model's potential for accurate, contextually appropriate responses. The fourth response, however, was incorrect, illustrating the limitations of the model and the challenges inherent in training neural networks. These examples represent a fraction of the questions posed, highlighting both successes and areas for improvement.

**Improving the training data.** The attempt to improve the training data was made by combining the first and second iteration data sets without changing the number of validation questions. Due to the temporary availability of the Eurosoftware server, and in order not to disturb the availability of the server for others, it was decided to proceed with the training of a single model. This model, the LLaMA with the Lora PEFT scheme and a validation step of 200, was chosen because of its superior performance in previous tests. This approach resulted in a slight improvement in the F1 score to 0.4693. The model's responses also showed a minor improvement, although they still did not fully meet expectations.

Examples of the model's responses in this phase can be found in the Appendix in Listing C.8.

While the first and last responses demonstrate an improved understanding and ability to provide direct answers, the second response highlights a persistent issue with the model's ability to generate informative and contextually enriched answers, as it merely rephrases the query.

**Conclusion.** This iteration successfully advanced the project by training and evaluating neural network models, where the LLaMA model with the Lora PEFT scheme and a 200-validation step emerged as the optimal configuration, achieving the highest F1 score. Although the model showed potential for contextually appropriate responses, it also revealed limitations in generating informative content, identifying areas for future enhancement. Efforts to merge datasets from the first and second iterations led to a marginal performance boost, underscoring the significance of dataset quality in model training. This ongoing process of evaluation and refinement strengthens the groundwork for further development, promising more sophisticated neural network capabilities in upcoming phases.

### 6.2.5 Fifth Iteration: Data Expansion and Model Refinement

After setting a robust technical foundation and achieving initial model training in previous iterations, the focus of the fifth iteration shifted towards expanding the training dataset and further testing the model's performance.

**Objectives.** The main goals of this iteration were to:

- Expand the training dataset to improve the model's learning capacity.
- Test the model with the newly expanded training data to assess performance improvements.

**Data Preparation.** This phase was characterized by a significant expansion of the training dataset. Following extensive efforts, a total of 1697 new training samples were prepared, enhancing the model's exposure to varied data points. This expansion included extended versions of previously used architectures, enriching the dataset's diversity and depth. Preparing this enhanced dataset was a labour-intensive task that spanned approximately two weeks. The summary of the expanded training samples is presented below in Table 6.5:

Architecture	Training Samples
Label & Poster Printing extended	120
Offline Mobile Store extended	136
Self Scanning extended	163
Store Management extended	203
Multitenancy extended	289
Store Device Control extended	334
Omnichannel Point-of-Sale extended	452

Table 6.5: Summary of Expanded Training Samples

All these training samples can be found in `Input_data\Iteration 5` directory.

**Model Training and Evaluation.** The same model configuration identified as optimal in the fourth iteration was used to train on this significantly larger dataset. Surprisingly, the F1 score post-training was 0.52387943827323, only marginally higher than previous iterations despite the substantial increase in training data. This outcome was unexpected, considering the scale of data expansion. However, manual evaluations of the model's outputs that can be found in the Appendix in Listing C.9 revealed significant improvements.

**Conclusion.** The fifth iteration demonstrated that expanding the training dataset could incrementally improve the model's performance, as evidenced by a slight increase in the F1 score and notably more accurate manual evaluation responses. These results, while promising, suggest that further dataset enhancement and validation data refinement could yield even better outcomes. However, at this stage, Eurosoftware deemed the progress and results satisfactory, marking a potential pause in the development cycle with an open door for future enhancements. This iteration's successes reflect the cumulative efforts of the project, showcasing the iterative nature of AI model training and the ongoing quest for perfection.

# Conclusion

## 7

This bachelor's thesis has undertaken an in-depth investigation of artificial intelligence tools and their role in the facilitation of software development within Eurosoftware. The first step in this comprehensive analysis was an evaluation of publicly available AI tools to assess their potential for streamlining software development processes, particularly for Eurosoftware and similar organizations. This evaluation showed that while many tools are effective in various aspects of software development, they often fall short in tasks related to User Interface development, identifying a critical area for future improvements.

The thesis then transitioned into a more focused investigation, specifically tailored to the needs of Eurosoftware. The development and implementation of a custom JSONL editor was a major milestone. This tool proved critical in preparing and managing the large amounts of data required to train a specialized neural network model. By processing extensive data from Eurosoftware's enterprise applications, a solid training and validation database was successfully created.

After several iterations of testing and refinement, the results were promising, demonstrating its potential to significantly assist in the process of retrieving information within the company. These positive results validate the effectiveness of current AI tools and strategies and highlight the model's ability to meet and potentially exceed its intended goals. However, despite these successes, there remains a significant opportunity to further enhance the AI model to improve its accuracy and efficiency.

The journey of integrating AI into software development at Eurosoftware has laid the foundation for significant operational improvements. The lessons learned and technologies developed through this work not only enhance Eurosoftware's capabilities but also contribute to the broader field of AI in software development. This opens the way for future innovations and continuous advancements in AI technology, ensuring that Eurosoftware remains at the forefront of software development.



# List of Abbreviations



**AI** Artificial Intelligence

**POS** Point Of Sale

**IDE** Integrated Development Environment

**OOP** Object Oriented Programming

**API** Application Programming Interface

**XML** Extensible Markup Language

**Swing** Java Swing

**Spring** Java Spring Framework

**UCON/ECON** Universal Connector

**CIS** Cluster Infrastructure Server

**SDC** Store Device Control

**HAL** Hardware Abstraction Layer

**BLServices** Business Logic Services

**ProMoDSL** Process Model Domain Specific Language

**ML** Machine Learning

**GPT** Generative Pre-trained Transformer

**NLP** Natural Language Processing

**HTTP** HyperText Transfer Protocol

**SQL** Structured Query Language

**GUI** Graphical User Interface

**REST** Representational State Transfer

**UI** User Interface

**DO** Domain Object

**DO DSL** Domain Specific Language

**SAP** Systems Applications and Products in Data Processing

**JMS** Java Message Service

**IDOC** Intermediary DOCument

**JAX-RS** Java API for RESTful Web Services

**JSON** JavaScript Object Notation

**UI5** SAP Open User Interface 5

**CSS** Cascading Style Sheets

**HTML** HyperText Markup Language

**GPT-3** ChatGPT Version 3.0

**GPT-3.5** ChatGPT Version 3.5

**GPT-4** ChatGPT Version 4.0

**GPL** General Public License

**LLM** Large Language Model

**RAG** Retrieval Augmented Generation

**GPU** Graphics Processing Unit

**SFT** Supervised Fine Tuning

**DPO** Direct Preference Optimization

**RLHF** Reinforcement Learning From Human Feedback

**XHTML** Extensible Hypertext Markup Language

**JSONL** JavaScript Object Notation Lines

# Attachments Structure



## B.1 Application and Libraries

This section outlines the organization and content of the 'Application\_and\_libraries' directory, which consists of various subdirectories and files.

### JSONL Editor.

- **application** – Contains core application scripts that define the main functionality of the JSONL Editor. These scripts are written in Python ('.py' files), handling data processing and editing operations.
- **application\_ui** – Includes Python scripts responsible for creating and managing the graphical user interface of the JSONL Editor. These files define windows, dialogs, and other UI elements that interact with the user.
- **main** – This is the main entry script for the JSONL Editor application. It initializes the application, setting up the environment and starting the user interface.
- **Readme** – A markdown file ('Readme.md') providing documentation, setup instructions, and usage guidelines for the JSONL Editor.
- **utility\_functions** – Consists of Python scripts that provide additional functionality such as file handling, data validation, and other utility operations which support the main application processes.

### Eurosoftware's Custom AI.

- **conf** – Contains YAML configuration files that specify settings for different operational modes of the AI model, including:

- \* *llama\_7B.yaml* and *llama\_7B\_inference.yaml* – Configuration files for the Llama 7B model for both training and inference phases.
- \* *megatron\_gpt5B.yaml* and *megatron\_gpt5B\_inference.yaml* – Configuration files for the Megatron GPT-5B model tailored for training and inference setups.
- **builder** – Python scripts that construct and manage the AI model, including its setup, execution, and training procedures.
- **interface** – Scripts that define the interfaces between the AI models and other application components, facilitating data exchanges and model interaction.
- **Readme** – A markdown document, providing usage guidelines for builder and interface scripts.

## B.2 Input Data

The input data is organized into several iterations.

- **Iteration 1**

- Contains files such as *Cluster Infrastructure Service (CIS)*, *Development Rules*, *Enterprise Connector (ECON)*, *FirstStepsWithGKRetail*, and *Storemanager*.
- These JSONL formatted files serve as configuration or operational data, used as training data for developing AI NLP models.

- **Iteration 2**

- Includes files detailing specific architectural components such as *Architecture Label & Poster Printing*, *Architecture Multitenancy*, *Architecture Offline Mobile Store*, *Architecture Omnichannel Point-of-Sale*, *Architecture Self Scanning*, *Architecture Store Device Control*, and *Architecture Store Management*.
- Additionally, each architectural component file is paired with a corresponding validation file (*- validation*), used to validate the AI NLP model.

- **Iteration 5**

- Extends the architectural descriptions from previous iterations with files named *Architecture Label & Poster Printing extended*, *Architecture*

*Multitenancy extended, Architecture Offline Mobile Store extended, Architecture Omnichannel PointofSale extended, Architecture Self Scanning extended, Architecture Store Device Control extended, and Architecture Store Management extended.*

- These files include additional, extended information about each system component, serving as supplementary training data to enhance the AI model’s capabilities.

## B.3 Results

This section details the outcomes of several AI-assisted programming tasks. These tasks involve generating, optimizing, interpreting, and translating code using models such as ChatGPT 3.5 and GPT-4, as well as tools like TabNine and GitHub Copilot. Each sub-section below corresponds to a specific directory within the ‘Results’ folder, where the detailed results are stored.

### B.3.1 GPT-3.5

- **Explaining SQL:**
  - SQL\_exp1 file, contains detailed explanations of SQL queries by ChatGPT 3.5 with review.
- **Optimizing SQL:**
  - Optimizing SQL file, includes optimizations of SQL queries performed by ChatGPT 3.5 with review.
- **Interpreting Lambda:**
  - Lambda\_interpr file provides interpretations of lambda expressions, demonstrating ChatGPT 3.5’s understanding of functional programming constructs with review.
- **Optimizing Lambda:**
  - Lambda\_opt file contains optimized versions of lambda expressions by ChatGPT 3.5 with review.
- **Generating Lambda:**
  - Lambda\_gen file showcases lambda expressions generated by ChatGPT 3.5 with review.

## B.3.2 GPT-4 Results

- **Code Interpretation:**
  - `Java_interpr.txt` and `DODSL_interpr.txt` files contain detailed interpretations of Java and DODSL code segments by GPT-4, with reviews of the AI's understanding and analysis.
- **Advanced Code Refactoring:**
  - `Java_refact.txt` and `DODSL_refact.txt` files include examples of code refactoring performed by GPT-4 on Java and DODSL scripts, along with detailed evaluations of these refactorings.
- **Groovy to JShell Translation:**
  - `Groovy_to_JShell.txt` file documents the process and outcomes of converting Groovy scripts into JShell commands by GPT-4 with review.
- **Java To Python Translation:**
  - `Java_to_Python.txt` file details the translation of Java code to Python by GPT-4, showing the AI's capability to adapt code across different programming languages.
- **XHTML to Vue Translation:**
  - `XHTML_to_Vue.txt` file captures the translation of XHTML to Vue.js by GPT-4, documenting the process and the results.

## B.3.3 TabNine

- **TypeScript:**
  - Files `carwashTable`, `clientSecurityApi`, `getFunctionContextById` in `.txt` and `.mp4` formats contain the outputs from TabNine's code completions in TypeScript, illustrating its efficiency in various development contexts.
- **Vue:**
  - Files like `giftCertificateButton`, `giftCertificateInputField`, and `priceParameter` in `.txt` and `.mp4` formats demonstrates TabNine's real-time code completion capabilities in Vue.js development scenarios.

## B.3.4 GitHub Copilot

- **Java Completion**
  - Files such as `AccountingUtils`, `ClientFactory`, `DiscountSearch` as well as `InvoicePrintingUtils` and `TenderUtils` in `.txt` and `.mp4` formats demonstrates GitHub Copilot's assistance in Java code generation, showing its potential to enhance productivity and code quality.
- **TypeScript Completion**
  - Evaluation results from files like `carwashTable` and `clientSecurityApi` indicate GitHub Copilot's capabilities in TypeScript, providing contextually relevant code suggestions that improve the development workflow.
- **TypeScript Chat**
  - The `chatTypeScript` file contains a detailed analysis of conversations and interactions with GitHub Copilot, focusing on its responsiveness and utility in TypeScript.
- **Vue Chat**
  - The `chatVue` file reviews GitHub Copilot's performance in generating Vue.js code through interactive chat sessions, highlighting its ability to understand and translate developer intents into functional code snippets.

# Supplementary Materials



## C.1 Code Examples

### C.1.1 Example 1: XML Layout

Source code C.1: Example of an XML layout

```
1 <?xml version='1.0' encoding='UTF-8'?>
2 <html>
3   <head>
4     <link rel="stylesheet" href="css/common/common.css" />
5     <link rel="stylesheet" href="css/common/inputs.css" />
6     <link rel="stylesheet" href="css/custom/custom.css" />
7   </head>
8   <body id="rootPanel">
9
10    <!-- PAGE TITLE -->
11    <div id="leftContentTopArea" slot="true" class="topArea"
12    layout="wrapLayout">
13      <label id="accountInfoTitle" class="pageTitle" valueKey
14      ="registration.accountInfo.input.title"/>
15    </div>
16
17    <form name="promptAccountInfo">
18      <!-- INPUTS -->
19      <div id="leftContentCenterArea" slot="true" class="
20      centerArea_inputArea" layout="wrapLayout">
21        <div id="fieldLabel1" class="fieldLabel">
22          <input id="mainTextInput" type="text" required="
23          true" />
24        </div>
25      </div>
26
27      <!-- BUTTONS -->
28      <div id="rightContentCenterArea" slot="true" class="
29      centerArea_buttonsPanel_main" layout="wrapLayout">
```



```

25     <input id="cancelButton" class="large_ca" type="
button" valueKey="button.cancel" />
26     <input id="okButton" class="large_cno" type="submit"
valueKey="button.ok" />
27     </div>
28
29 </form>
30 <!-- KEYBOARD AREA -->
31 <iframe id="keyboardArea" slot="true" class="keyboardArea
" constraints="Last" src="/layouts/tpos/simple/keyboard/
alphaNumericKeyboardComponent.xml" bean="
alphaNumericKeyboardComponent" />
32 </body>
33 </html>

```

## C.1.2 Example 2: ProMo Source Code

Source code C.2: AccountInfo\_CST.promo example

```

1 dslLevel 2.13
2 Package com.gk_software.cst.pos.account_info
3 Import processModels.tpos.common.dialog.ShowMessageWithArg
4 Import processModels.tpos.common.dialog.ShowSimpleMessage
5
6 Process AccountInfo_CST
7   -> StartProcess
8
9 State
10  accountName : string
11  accountBalance : number
12  formattedBalance : string
13
14 Label StartProcess -> sub PromptUserAccount_CST(empty) /
15 take(accountName as accountName) ->
16 {
17   sub.SUCCESS / getAccountBalance -> IsZeroBalance
18   sub.CANCEL -> ReturnCancel
19 }
20
21 Switch IsZeroBalance on isZeroBalance
22   true -> ShowZeroBalanceDialog
23   false -> ShowAccountBalanceDialog
24
25 Label ShowAccountBalanceDialog ->
26 sub ShowMessageWithArg(Info,
27   "registration.accountInfo.infoDialog",

```

```
28         process.formattedBalance) ->
    ReturnSuccess
29 Label ShowZeroBalanceDialog ->
30 sub ShowSimpleMessage(Info,
31     "registration.accountInfo.zeroBalance",
32     empty) -> ReturnSuccess
33
34 Label ReturnSuccess -> return as SUCCESS
35 Label ReturnCancel -> return as CANCEL
36
37 Action getAccountBalance
38     reads process.accountName
39     writes process.accountBalance
40     writes process.formattedBalance
41
42 Condition isZeroBalance
43     reads process.accountBalance
```

---

### C.1.3 Example 3: ProMoj Source Code

Source code C.3: AccountInfo\_CST.promoj example

---

```
1 dslLevel 2.13
2 Package com.gk_software.cst.pos.account_info
3
4 //import of promo file
5 Import com.gk_software.cst.pos.client.function.registration.
    account_info.AccountInfo_CST
6
7 Process AccountInfo_CST
8     references com.gk_software.pos.api.model.config.
    DialogConfig_DO
9
10 Action getAccountBalance = java:com.gk_software.cst.pos.
    client.registration.action.account_info.
    AccountInfoActionLib_CST::getAccountBalance
11
12 Condition isZeroBalance = pojo:com.gk_software.cst.pos.client
    .registration.action.account_info.
    AccountInfoActionLib_CST::isZeroBalance
```

---

## C.1.4 Example 4: Usage of DO DSL

Source code C.4: AccountInfoRequest\_CST\_DO.do - java class definition

```

1 dslLevel 2
2
3 Package com.gk_software.cst.api.service.registration.model.
   account_info
4
5 domain object AccountInfoRequest_CST_DO {
6
7     version "1.0"
8
9     master entity AccountInfoRequest_CST {
10         attribute accountName : String
11         attribute accountBalance : double
12     }
13 }

```

Source code C.5: AccountInfoRequestConfigs.doin - .properties file definition

```

1 dslLevel 2
2 Package com.gk_software.cst.api.service.registration.model.
   account_info
3
4 Import com.gk_software.cst.api.service.registration.model.
   account_info.AccountInfoRequest_CST_DO.
   AccountInfoRequest_CST
5
6 collection AccountInfoRequestConfigs [
7
8     instance Default {
9         accountName = "testName"
10        accountInfoEnabled = 12.0
11    }
12
13 ]

```

Source code C.6: Example of .doj file - generation definition

```

1 dslLevel 2
2
3 Package com.gk_software.cst.api.service.registration.model
4
5 Import com.gk_software.cst.api.service.registration.model.
   account_info.AccountInfoRequest_CST_DO
6
7 target m5 extends mach5j_10 {

```

```
8     dbschemaPackage = "com.gk_software.pos.api.model.config.
9     dbschema"
10    mappingsPackage = "com.gk_software.pos.api.model.config.
11    utils"
12    utilsPackage = "com.gk_software.pos.api.model.config.
13    utils"
14    noReferences = "true"
15    noTuplizers = "true"
16    publicConstructor = "true"
17 }
18 target m5, dojbin
19
20 domain object AccountInfoRequest_CST_DO {
21     package "com.gk_software.cst.gen.api.service.registration.
22     model.account_info"
23     factory AccountInfoRequestFactory_CST
24
25     entity AccountInfoRequest_CST {}
26 }
```

## C.2 Detailed Model Outputs

### C.2.1 Fourth Iteration: Initial Model Training Outputs

Listing C.7: Outputs of LLaMA model (Lora PEFT scheme, validation step of 200, data from second iteration)

```
1 User: Question: Could POS Mobile Client be connected via the
2 Flow Service interface?
3 Assistant: Yes, POS Mobile Client could be connected via the
4 Flow Service interface.
5
6 User: Question: Are POS transactions retained in the Central
7 Transaction Pool?
8 Assistant: Yes, POS transactions are retained in the Central
9 Transaction Pool.
10
11 User: Question: The Self Checkout POS client variant sends
12 something to the SCO Assistant?
13 Assistant: The Self Checkout POS client variant sends
14 something to the SCO Assistant.
15
16 User: Question: What does CIS stand for?
17 Assistant: CIS stands for Central Inventory Service.
```

Listing C.8: Outputs of LLaMA model (Lora PEFT scheme, validation step of 200, data from first and second iteration)

```
1 User: Question: Is automatic print function in Label Print
2 Service done on master data change?
3 Assistant: Yes, automatic print function in Label Print
4 Service is done on master data change.
5
6 User: Question: The Self Checkout POS client variant
7 sends something to the SCO Assistant?
8 Assistant: The Self Checkout POS client variant sends
9 something to the SCO Assistant.
10
11 User: Question: How to access the Mobile Workbench web
12 client from mobile devices?
13 Assistant: Mobile devices can access the Mobile Workbench
14 web client via a mobile browser.
15
16 User: Question: What does CIS stand for?
17 Assistant: CIS stands for Cluster Infrastructure Server.
```

## C.2.2 Fifth Iteration: Expanded Data Model Outputs

Listing C.9: Outputs of LLaMA model (Lora PEFT scheme, validation step of 200, expanded data from all iterations)

```
1 User: Question: What role does SDC play in master data
  management?
2 Assistant: SDC plays a crucial role in master data
  management by providing a central repository for master
  data, facilitating the creation, maintenance, and sharing
  of master data across the system.
3
4 User: Question: How to access the Mobile Workbench web
  client from mobile devices?
5 Assistant: Access the Mobile Workbench web client from
  mobile devices via a web browser.
6
7 User: Question: What does CIS stand for?
8 Assistant: CIS stands for Cluster Infrastructure Server.
9
10 User: Question: The Self Checkout POS client variant sends
  something to the SCO Assistant?
11 Assistant: Yes, the Self Checkout POS client variant sends a
  request to the SCO Assistant for processing.
12
13 User: Question: What does ECON stand for?
14 Assistant: ECON stands for Enterprise Connector.
```

```
15
16 User: Question: Are POS transactions retained in the Central
    Transaction Pool?
17 Assistant: Yes, POS transactions are retained in the Central
    Transaction Pool.
```

## C.3 User Manual: JSON Editor Application

This manual provides guidance on using the JSON Editor, a tool designed to create, edit, and manage JSONL files effectively.

### C.3.1 Setting Up Application on Windows

1. Check if Python is installed by typing in Command Prompt:

```
python --version
```

If Python is not installed, download and install it from <https://www.python.org/downloads/> ensuring that you check “Add Python to PATH” during the installation.

2. Download and extract the JSON Editor’s source code.
3. Navigate to the application’s directory in Command Prompt and run:

```
python main.py
```

### C.3.2 Setting Up Application on Linux

1. Check if Python is installed by typing in Terminal: `python3 --version` If not, install Python and pip using your distribution’s package manager, for example:

```
sudo apt-get install python3 python3-pip
```

2. Download and extract the JSON Editor’s source code.
3. Navigate to the application’s directory and start it by running:

```
python3 main.py
```

### C.3.3 File Management

- **Open File:** Load JSONL files for editing.
- **Save File:** Save current changes to the open file.
- **Save File As:** Save the current file under a new name.

### C.3.4 Editing Entries

- **Add Entry:** Add new entries to the JSONL file.
- **Edit Entry:** Modify existing entries. Double-click an entry or select it and press Enter to edit.
- **Delete Entry:** Remove selected entries from the file.

### C.3.5 Opening a File

To open a file, use the **Open File** button on the main interface. Navigate to the file's location, select it, and click Open.

### C.3.6 Adding and Editing Entries

To add a new entry, click the **Add Entry** button. To edit an existing entry, either double-click on the entry or select it and press Enter. This will open a modal window where you can edit the Question and Answer fields.

### C.3.7 Saving Files

To save changes to the currently open file, click the **Save File** button. If you wish to save the file with a new name, click the **Save File As** button.

## C.4 User Manual: AI Training and Validation Scripts

This manual provides guidance on using the Training and Validation Scripts on Linux

### C.4.1 System Requirements

To ensure proper functionality of Training and Validation Scripts, the following dependencies must be installed:

- PyTorch (compatible with CUDA 12.2)
- Nvidia GPU with CUDA 12.2 support
- CUDA 12.2
- CUDNN 8
- GCC12-C++
- Nvidia Transformer Engine
- Nvidia NeMo Toolkit Version 1.21
- Nvidia Apex

## C.4.2 Setting up the Environment

Begin by creating and activating a new Conda environment:

```
conda create --name confluence python==3.11
conda activate confluence
```

## C.4.3 Installing PyTorch

Install PyTorch by using the PyTorch configurator:

```
conda install pytorch torchvision torchaudio pytorch-cuda=12.1 -c py-
torch -c nvidia
```

Please refer to the PyTorch configurator to select the correct installation command for your system.

## C.4.4 Apex Installation

Install Nvidia Apex by first setting up your CUDA environment:

```
conda install -c nvidia cuda-nvprof=12.1
pip install packaging
git clone https://github.com/NVIDIA/apex
cd apex
vim setup.py
pip install -v --disable-pip-version-check --no-cache-dir --
no-build-isolation --config-settings "--build-option=--cpp_ext" -
-config-settings "--build-option=--cuda_ext" ./
```



## C.4.5 Transformer Engine and NeMo Toolkit

For the Transformer Engine, ensure C++17 and compatible compiler are in use:

```
export CXX=/usr/bin/g++-12
export CC=/usr/bin/gcc-12
export NVTE_FRAMEWORK=pytorch
git clone --branch stable --recursive
https://github.com/NVIDIA/TransformerEngine.git
cd TransformerEngine
pip install .
```

Finally, install the NeMo Toolkit:

```
pip install nemo-toolkit==1.21.0
```

## C.4.6 Docker Container Setup

```
docker login nvcr.io
docker pull nvcr.io/nvidia/nemo:23.08
docker image ls
docker run --env CUDA_VISIBLE_DEVICES=0,1 --gpus all --ipc=host -
-ulimit memlock=-1 --ulimit stack=67108864 -it -P -
v ./confluence:/workspace/confluence -v ./pretrained-
models:/workspace/pretrained-models <YOUR_IMAGE_ID> /bin/bash
```

## C.4.7 Setting Environment Variables

```
export NVTE_FLASH_ATTN=1
export NVTE_FUSED_ATTN=0
export NVTE_FUSED_ATTN_BACKEND=0
```

## C.4.8 Executing Training and Validation Scripts

This section describes how to use the `torchrun` command to operate training and validation scripts. The command provided is configured for either training or inference processes based on the model and configuration file specified.

```
torchrun --nproc_per_node=4 interface.py -c llama -tr true
```

### Parameters Description.

- **-c (Model Configuration):** This parameter can accept either `llama` or `megatron` as values. It determines which model to initiate and which configuration file to use. For example:
  - `llama` will utilize `llama_7B.yaml`
  - `megatron` will utilize `megatron_gpt5B.yaml`

If the `-tr` parameter is set to `false`, the corresponding inference configuration files (`llama_7B_inference.yaml` or `megatron_gpt5B_inference.yaml`) will be used instead.

- **-tr (Training or Inference):** This boolean parameter defines whether to run the process as a training session (`true`) or as inference (`false`).
- **-a (Accuracy Evaluation):** This optional parameter indicates whether to evaluate the model's accuracy. It is set to `false` by default, but can be set to `true` to perform accuracy assessment.

**Configuration Adjustments.** Once you have decided on the operation mode and understood the parameters, you may need to adjust the configuration accordingly:

If you are setting up for a training configuration file will require adjustments such as:

- **name:** This affects the naming of your model instance, e.g., `llama_7B`.
- **model.tokenizer.model:** Enter the path to the tokenizer `../pretrained-models/language-models/llama2-7b-hf/tokenizer.model` for example.
- **model.peft.peft\_scheme:** Select a PEFT scheme, e.g., `lora`.
- **model.data.train\_ds.file\_names:** This is an array of strings containing the file names for training, e.g., `["./data/iteration5_1.jsonl"]`.
- **model.data.validation\_ds.file\_names:** For training, you also need to specify validation data files, e.g., `["./data/validation5.jsonl"]` - only in training configuration files.

For inference operations, additional adjustments include:

- **model.peft.restore\_from\_path**: Path to restore the model from, e.g.,  
./training/llama-7B-it5/checkpoints/llama-7B-it5.nemo.
- **inference.outfile\_path**: Output path for model responses, e.g., /workspace/-confluence/output.txt.

Adjustments should reflect your specific data paths and model requirements. Additional comments on what a particular position in the .yaml file does can be found in the configuration file itself

## C.4.9 Logging and Monitoring

To monitor training progress, use TensorBoard. This utility logs all events and checkpoints:

```
tensorboard --log-dir ./training_info --port 6006 --bind_all
```

# Bibliography

- [Beh23] BEHLER, Marco. *What is Spring Framework? An Unorthodox Guide*. marcobehler, 2023-06-23. Available also from: <https://www.marcobehler.com/guides/spring-framework>.
- [Bru19] BRUSH, Kate. *Definition Jira*. TechTarget, 2019-12. Available also from: <https://www.techtarget.com/searchsoftwarequality/definition/Jira>.
- [Col23] COLLINS, Tom. *What is Maven in Java? (Framework and Uses)*. BrowserStack, 2023-07-26. Available also from: <https://www.browserstack.com/guide/what-is-maven-in-java>.
- [Con24] CONROY, Shaun. *GPT 4 Image Input: How to use ChatGPT Image Input feature*. TechTarget, 2024-04-09. Available also from: <https://silentpcreview.com/gpt-4-image-input-chatgpt/>.
- [Iwu24] IWUOZOR, John. *Confluence Review 2024: Features, Pros And Cons*. Forbes, 2024-01-04. Available also from: <https://www.forbes.com/advisor/business/software/confluence-review/>.
- [Kel24] KELLY, Will. *GPT-3.5 vs. GPT-4: Biggest differences to consider*. TechTarget, 2024-02-27. Available also from: <https://www.techtarget.com/searchenterpriseai/tip/GPT-35-vs-GPT-4-Biggest-differences-to-consider>.
- [Koç09] KOÇYİĞİT, Altan. *Object Oriented Programming with Java I*. 2009. Available also from: <https://ocw.metu.edu.tr/course/view.php?id=23>.
- [Kuc23] KUCHAIEV, Oleksii. *NeMo Megatron-GPT 5B*. 2023. Available also from: <https://huggingface.co/nvidia/nemo-megatron-gpt-5B>. Accessed: 2024-04-16.
- [Law22] LAWSON, Loraine. *Code Stays Behind Firewall with Copilot Alternative Tabnine*. The New Stack, 2022-06-29. Available also from: <https://thenewstack.io/code-stays-behind-firewall-with-copilot-alternative-tabnine/>.

- [Lep] LEPPILAMPI, Tuomas. *THE BENEFITS (AND PITFALLS) OF GITHUB COPILOT*. Eficode. Available also from: <https://www.eficode.com/blog/the-benefits-and-pitfalls-of-github-copilot>.
- [Met23] META. *Llama-2-7b*. 2023. Available also from: <https://huggingface.co/meta-llama/Llama-2-7b>. Accessed: 2024-04-16.
- [NVI23a] NVIDIA. *NeMo Megatron*. 2023. Available also from: <https://docs.nvidia.com/nemo-framework/user-guide/latest/nemotoolkit/nlp/megatron.html>. Accessed: 2024-04-16.
- [NVI23b] NVIDIA. *NVIDIA NeMo Framework*. 2023. Available also from: <https://nvidia.github.io/NeMo/>. Accessed: 2024-04-16.
- [NVI23c] NVIDIA. *What Is NVIDIA NeMo?* 2023. Available also from: <https://www.nvidia.com/en-us/ai-data-science/products/nemo/>. Accessed: 2024-04-16.
- [Ram23] RAMEL, David. *Tabnine Unveils AI Assistant for Visual Studio 2022, Chat Beta*. Visual Studio Magazine, 2023-08-18. Available also from: <https://visualstudiomagazine.com/articles/2023/08/18/tabnine.aspx>.
- [Rar23] RAROQUE, Chris. *How To Use GitHub Copilot*. Aloa, 2023. Available also from: <https://aloha.co/blog/how-to-use-github-copilot>.
- [Ros23] ROSHELOVA, Albena. *Revolutionizing Development: Exploring AI-Powered Code Tools — Copilot and Tabnine*. Medium, 2023-03-27. Available also from: <https://medium.com/@aroshelova.tech/revolutionizing-development-exploring-ai-powered-code-tools-copilot-and-tabnine-6e1a88f1a2d7>.
- [Son23] SONI, Mitesh. *Eclipse IDE: An Overview*. Open Source ForU, 2023-05-08. Available also from: <https://www.opensourceforu.com/2023/05/eclipse-ide-an-overview/>.

# List of Figures

2.1	Central Architecture Scheme . . . . .	19
6.1	General Overview of the JSONL Editor Interface . . . . .	49
6.2	File Management Buttons: Open, Save, and Save As . . . . .	50
6.3	Data Management Buttons: Add and Delete Entries . . . . .	50
6.4	Modal Window for Adding a New Entry . . . . .	50
6.5	Modal Window for Editing an Existing Entry . . . . .	51
6.6	Example of an Error Message During File Validation . . . . .	52
6.7	Warning Message Indicating External Modifications . . . . .	53

# List of Tables

5.1	Evaluation of ChatGPT 3.5 in Software Development Tasks . . . . .	45
5.2	Evaluation of GPT-4 in Software Development Tasks . . . . .	46
5.3	Evaluation of TabNine in Software Development Tasks . . . . .	47
5.4	Evaluation of GitHub Copilot in Software Development Tasks . . . . .	47
6.1	Hotkeys in the Main Application . . . . .	51
6.2	Hotkeys in Modal Windows . . . . .	51
6.3	Summary of Training Samples Prepared (Iteration 1) . . . . .	55
6.4	Summary of Training and Validation Samples Prepared (Iteration 2) . .	56
6.5	Summary of Expanded Training Samples . . . . .	63

# List of Listings

6.1	Training Sample Example . . . . .	56
6.2	Validation Sample Example . . . . .	56
C.1	Example of an XML layout . . . . .	72
C.2	AccountInfo_CST.promo example . . . . .	73
C.3	AccountInfo_CST.promoj example . . . . .	74
C.4	AccountInfoRequest_CST_DO.do - java class definition . . . . .	75
C.5	AccountInfoRequestConfigs.doin - .properties file definition . . . . .	75
C.6	Example of .doj file - generation definition . . . . .	75
C.7	Outputs of LLaMA model (Lora PEFT scheme, validation step of 200, data from second iteration) . . . . .	76
C.8	Outputs of LLaMA model (Lora PEFT scheme, validation step of 200, data from first and second iteration) . . . . .	77
C.9	Outputs of LLaMA model (Lora PEFT scheme, validation step of 200, expanded data from all iterations) . . . . .	77



1101001 1100001  
10101100001110010 1100001  
101011010101 1100001



11010011101101001  
011000011010101  
11100010101110101