

Thema:

Praktische Anwendung von Large Language Models im Bereich Software Engineering am Beispiel von ChatGPT

Ausarbeitung

im Rahmen des Moduls E541 Softwareprojekt

Themensteller/: Prof. Dr. Wolfgang Albrecht

Betreuer

Vorgelegt von: Jan Walldorf – Matrikelnummer: 538992

Projektzeitraum: 01.10.2023 - 31.03.2023

Inhaltsverzeichnis

[1 Einleitung 1](#_Toc162776945)

[2 Aufgabenstellung und Ziele 1](#_Toc162776946)

[3 Technologien 2](#_Toc162776947)

[3.1 künstliche Intelligenz in der Praxis 2](#_Toc162776948)

[3.1.1 ChatGPT von OpenAI 2](#_Toc162776949)

[3.1.2 Github Copilot 2](#_Toc162776950)

[3.1.3 Alternative Produkte wie Amazon Codewhisperer 2](#_Toc162776951)

[3.1.4 Vergleich ChatGPT und Github Copilot 3](#_Toc162776952)

[3.2 Dokumentationstool 3](#_Toc162776953)

[3.2.1 Jupyter Notebook 3](#_Toc162776954)

[3.2.2 Google Colabs 3](#_Toc162776955)

[3.2.3 Anaconda 3](#_Toc162776956)

[3.2.4 Vergleich Jupyter Notebook Programme 4](#_Toc162776957)

[4 Use Cases 5](#_Toc162776958)

[5 Artefakte 6](#_Toc162776959)

[5.1 Code Refactoring mit KI 6](#_Toc162776960)

[5.2 Codeanalyse und Bewertung durch KI 6](#_Toc162776961)

[5.3 Tipps und Tricks für Prompt Engineering 7](#_Toc162776962)

[5.4 Anleitungen zu Github Copilot mit Erfahrungswerten 7](#_Toc162776963)

[6 Fazit 8](#_Toc162776964)

[7 Anhang 9](#_Toc162776965)

[7.1 Tipps\_und\_Tricks\_fuer\_ChatGPT.ipynb 9](#_Toc162776966)

[7.2 code\_Refactoring\_und\_Optimierung\_mit\_KI.ipynb 9](#_Toc162776967)

[7.3 Bewertung\_und\_konstruktive\_Kritik\_von\_Code\_durch\_KI.ipynb 9](#_Toc162776968)

[7.4 Anleitung Github Copilot Registrierung fuer Studenten.txt 9](#_Toc162776969)

[7.5 Anleitung Github Copilot Integration in QT.txt 9](#_Toc162776970)

[7.6 Anleitung Github Copilot Integration in VS Code.txt 9](#_Toc162776971)

[8 Literaturverzeichnis 10](#_Toc162776972)

# Einleitung

Die Entwicklung von Large Language Models (LLMs) hat im Bereich des Softwareengineerings einen Umbruch bewirkt, insbesondere durch die Entwicklung von leistungsstarken Tools wie ChatGPT von OpenAI. Diese Tools bieten innovative Ansätze zur Verbesserung von Entwicklungsprozessen und zur Steigerung der Produktivität. Jedoch kann die KI nicht nur zur Steigerung der Produktivität beitragen, sondern auch dazu dienen die Ausbildung künftiger Softwareentwickler zu verbessern. Die folgende Projektarbeit befasst sich mit der Anwendung künstlicher Intelligenz für die Softwareentwicklung am Beispiel von ChatGPT und Github Copilot. Dabei wird zunächst ein kurzer Überblick über einige angeschaute Technologien gegeben und deren Vor- und Nacheile diskutiert. Anschließend wird an praktischen Beispielen erläutert, wie KI bei der Softwareentwicklung und insbesondere wie im Studium den Studierenden dieses Thema näher gebracht werden kann um damit einen Mehrwert in der Ausbildung zu schaffen.

**Anmerkung**: Die in diesem Dokument dargestellten Arbeitsergebnisse stellen nur einen Teil der Projektarbeit dar und sind als Zusammenfassung zu sehen. Für den weiterführenden Inhalt wird auf die angehängten Textdokumente und Jupyter Notebooks verwiesen.

# Aufgabenstellung und Ziele

Aufgabenstellung:

* Technologieüberblick über den Einsatz von KI in der Softwareentwicklung
* Sammlung von Erfahrung im Umgang mit KI
* Testen verschiedener Tools zu Dokumentationszwecken

Ziele:

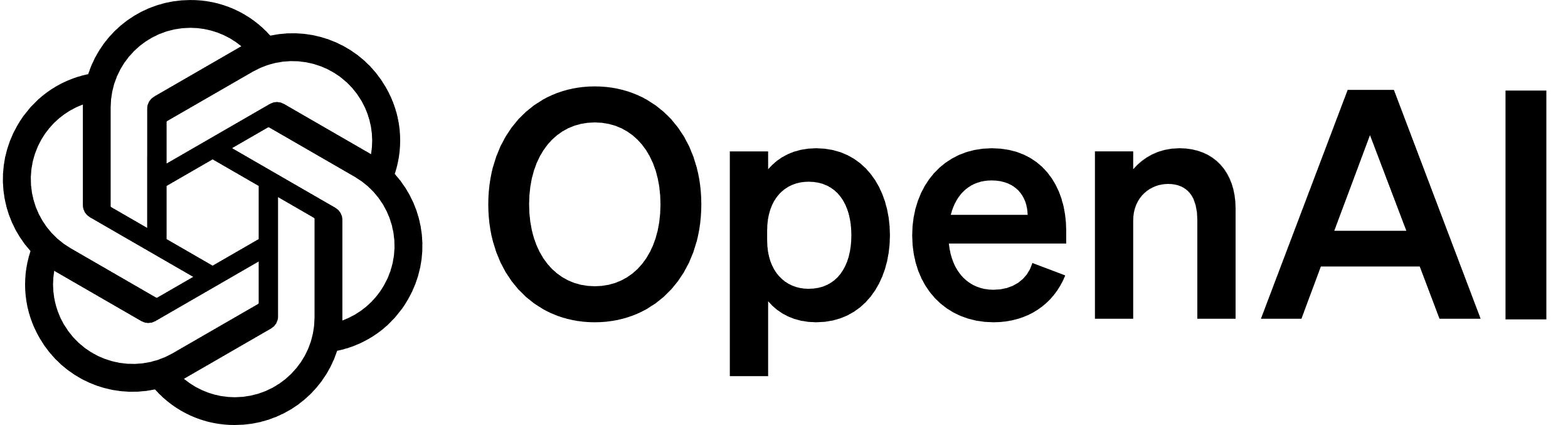
* Empfehlung für den Einsatz von KI im Studium
* Erstellung von praktischen Anwendungsbeispielen anhand von Praktikumsaufgaben für das Modul C++ Programmierung

# Technologien

## künstliche Intelligenz in der Praxis

Als Einleitung wird ein kurzer Überblick über drei weit verbreitete Large Language Models gegeben und erläutert, warum sich in dem Projekt auf ChatGPT fokussiert wurde.

### ChatGPT von OpenAI

ChatGPT ist eine moderne Softwareplattform, die entwickelt wurde, um natürliche Sprache zu verstehen und menschenähnliche Unterhaltungen zu führen. Sie wurde von OpenAI entwickelt und basiert auf einem fortschrittlichen Modell namens GPT (Generative Pre-trained Transformer). Mit ChatGPT können komplexe Ideen verstanden und entsprechende Antworten generiert werden, die auf den gegebenen Informationen beruhen. Diese Technologie findet Anwendung in einer Vielzahl von Bereichen von Kundensupport bis zur automatisierten Texterstellung. Neben einem klassischen Chatfenster um Dialoge mit der KI zu führen, bietet ChatGPT auch Programmierschnittstellen und Tools für Entwickler, um die Integration in verschiedene Anwendungen zu erleichtern. Mit diesen APIs können Entwickler ChatGPT in ihre eigenen Softwarelösungen einbinden und benutzerdefinierte Anwendungen erstellen. Darüber hinaus ermöglicht ChatGPT die Generierung von Programmcode aus einem vorgegebenen Text oder analysiert Code. (OpenAI, 2024)

### Github Copilot

GitHub Copilot ist ein Tool, das Entwicklern dabei hilft, Programmcode zu schreiben. Es basiert auf der Technologie von OpenAI's GPT-3. GitHub Copilot analysiert dabei den Kontext des Codes mithilfe von maschinellem Lernen und schlägt automatisch Vervollständigungen sowie ganze Codezeilen vor. Mit seiner Fähigkeit, verschiedene Programmiersprachen und Bibliotheken zu erkennen, ist es in der Lage, entsprechende Codeausschnitte zu generieren. Diese Funktionalitäten tragen dazu bei, die Produktivität zu steigern und den Entwicklungsprozess zu beschleunigen, indem die KI repetitive Aufgaben automatisiert und nützliche Codefragmente bereitstellt. Github Copilot hat sich dabei ausschließlich auf den Bereich Generierung sowie Analyse von Programmcode spezialisiert und ist daher nur in IDEs integriert. Daher ist Copilot nur eingeschränkt in einem Chatfenster ansprechbar. (Copilot, 2024)

### Alternative Produkte wie Amazon Codewhisperer

Desweiteren bestehen noch weitere Entwicklungen von LLMs anderer Firmen, dazu gehört unter anderem Codewhisperer von Amazon. Ähnlich zu Github Copilot erfolgt die Kommunikation direkt in einer IDE und die KI ist speziell darauf trainiert. Außerdem gibt es noch diverse andere Produkte wie Tabnine oder Google Gemini. Im Projekt wurde sich jedoch auf das derzeit bekannteste und ebenfalls für Studierende kostenfrei nutzbare Modell von OpenAI bezogen, weshalb eine detailliertere Betrachtung der anderen Produkte ausgelassen wurde. (Chornaya, 2024)

### Vergleich ChatGPT und Github Copilot

Im folgenden Abschnitt wird kurz der Unterschied zwischen ChatGPT und Github Copilot dargestellt. Da Github Copilot auf dem GPT-3 Modell von OpenAI basiert, unterscheiden sich die zwei Modelle nur geringfügig, jedoch sind einige Unterschiede feststellbar. Zunächst ist bei der Nutzung aufgefallen, dass die Antworten der KI von Github Copilot subjektiv betrachtet bessere Ergebnisse lieferte. Dies hat vor allem den Hintergrund, dass die KI speziell mit Codefragmenten trainiert wurde und daher für die Aufgabenstellung im Projekt – die Unterstützung bei der Erstellung von Programmcode – gut geeignet ist. Im Folgenden werden Vorteile von Github Copilot gegenüber dem klassischen ChatGPT aufgezählt:

1. Ausgelegt auf Codegenerierung
2. Integrationen in Entwicklungsumgebungen wie Visual Studio Code oder QT
3. Anpassung auf verschiedene Programmiersprachen und Bibliotheken

## Dokumentationstool

Zur Erstellung des unter Punkt 2 formulierten Ziels, die Erstellung von Anleitungen für Studierende, ist ein Dokumentationstool notwendig. Daher behandelt der folgende Abschnitt die Auswahl des Dokumentationstools, wobei die Wahl auf Jupyter Notebooks gefallen ist, da dieses Tool ein gutes Hands On Training erlaubt.

### Jupyter Notebook

Jupyter Notebooks sind interaktive Dokumente, die Code, Text und visuelle Elemente kombinieren und es Benutzern ermöglichen, Programmcode in Abschnitten auszuführen. Ihre Vorteile umfassen eine intuitive Benutzeroberfläche, die Zusammenarbeit erleichtert, und die Möglichkeit, Ergebnisse in Echtzeit zu visualisieren. Ein Nachteil ist unter anderem, dass als Standardprogrammiersprache Python festgelegt ist und andere Programmiersprachen nur in bestimmten Laufzeitumgebungen funktionieren. Jupyter Notebooks können mit verschiedenen Programmen erstellt werden, darunter VS Code (lokal), Google Colab (Cloud) und Anaconda (lokal und Cloud). Im Projekt wurden Jupyter Notebooks als Dokumentationstool verwendet, da das Feature Programmcode direkt auszuführen besonders für Studierende im Lernprozess vorteilhaft ist.

### Google Colabs

Google Colab ist eine Online-Plattform für interaktives erstellen von Programmcode und Dokumentationen, die es Benutzern ermöglicht, Jupyter Notebooks in der Cloud auszuführen. Zusätzlich bietet Google Colab die Möglichkeit, C++-Code in den Notebooks auszuführen, was die Flexibilität deutlich erhöht und das Tool daher die Anwendung im C++ Praktikum ermöglicht.

### Anaconda

Anaconda bietet eine umfassende Data-Science-Plattform, die eine Vielzahl von Tools und Bibliotheken für die Analyse und Visualisierung von Daten bereitstellt. Innerhalb der Anaconda-Umgebung sind Jupyter Notebooks eine zentrale Komponente, wobei Anaconda sowohl lokal auf dem eigenen Computer wie auch in der Cloud betrieben werden kann.

### Vergleich Jupyter Notebook Programme

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Programm | Cloud | Lokale Ausführung | C++ Code ausführbar | ChatGPT Integration |
| Google Colab | Ja | Nein | Ja | Ja (mit Chrome Plugin) |
| Anaconda | Ja | Ja | Nein | Ja – nativ |
| VS Code | Nein | Nein | Nein | Ja – nativ über Plugin Store |

Grundsätzlich unterscheiden sich die einzelnen Programme nur in Details und für allgemeine Anwendungen ist es irrelevant, für welches Tool man sich entscheidet. Da jedoch speziell die Erstellung von Notebooks in C++ relevant waren, wurde sich für die Nutzung von Google Colab entschieden.

# Use Cases

In der Softwareentwicklung gibt es eine Vielzahl von Anwendungsfällen für künstliche Intelligenz. Einige wichtige Use Cases sind:

* **Code-Generierung und Auto-Completion**: KI-gestützte Tools können dabei helfen, schneller und präziser Code zu schreiben, indem sie automatisch Codevorschläge generieren.
* **Fehlererkennung und Debugging:** KI-Algorithmen können eingesetzt werden, um Fehler im Code zu erkennen und zu beheben, indem sie Muster in den Daten analysieren und potenzielle Fehlerquellen identifizieren.
* **Automatisierte Tests:** KI kann verwendet werden, um automatisierte Tests zu erstellen und auszuführen, um die Qualität und Stabilität von Softwareprodukten zu verbessern.
* **Natürliche Sprachverarbeitung (NLP)**: NLP-Algorithmen können in Entwicklertools integriert werden, um natürliche Sprache zu verstehen und Entwicklern eine chatbasierte Kommunikation mit der KI bereitzustellen und Dokumentationen automatisiert zu erstellen.
* **Predictive Analytics:** Zur Analyse von Daten können KI-Modelle verwendet werden, um Vorhersagen über zukünftige Trends und Ereignisse in der Softwareentwicklung zu treffen, z. B. zur Schätzung von Entwicklungsaufwänden oder zur Identifizierung von potenziellen Engpässen im Projektmanagement.
* **Automatische Code-Optimierung (Refactoring):** KI-Algorithmen können verwendet werden, um vorhandenen Code zu analysieren und zu optimieren, um die Leistung zu verbessern, Speicherplatz zu sparen und die Wartbarkeit zu erhöhen.
* **Sicherheit und Datenschutz:** KI kann eingesetzt werden, um Sicherheitslücken und Datenschutzverletzungen in Softwareanwendungen zu erkennen und zu verhindern, indem sie verdächtiges Verhalten und Anomalien in den Daten identifiziert.

Diese Use Cases zeigen, wie KI-Technologien die Softwareentwicklung auf verschiedene Weise unterstützen können, indem sie Prozesse automatisieren, die Produktivität steigern und die Qualität von Softwareprodukten verbessern. Im vorliegenden Projekt wurde sich auf die Punkte Code Optimierung und Code Analyse beschränkt.

# Artefakte

## Code Refactoring mit KI

Dieser Abschnitt behandelt das Code Refactoring mit künstlicher Intelligenz, das im Jupyter Notebook „code\_Refactoring\_und\_Optimierung\_mit\_KI.ipynb“ dokumentiert ist. Darin wird zunächst allgemein Refactoring erläutert, was den Prozess der Umstrukturierung von Code ohne Änderung seines Verhaltens beschreibt. Danach werden Beispiele für die praktische Anwendung des Refactorings mit KI angegeben. Dabei wird ein Beispielprogramm für einen Multiple-Choice-Test in C++ schrittweise optimiert, wobei KI-basierte Anpassungen und Verbesserungen durchgeführt werden. Dabei werden die Punkte Strukturierung des Codes, die Lesbarkeit und eine bessere Erweiterbarkeit behandelt. Danach werden unabhängig zum Refactoring noch einige Tipps zur Optimierung des Codes gegeben, die über das reine Refactoring hinaus gehen. Zum Schluss kann zusammengefasst werden, dass KI ein sehr nützliches Tool zur Unterstützung beim Refactoring ist und gut bis sehr gut das Ziel erfüllt. Jedoch ist stehts zu beachten, dass die Notwendigkeit gegeben ist, den generierten Code zu testen und sicherzustellen, dass er danach noch wie erwartet funktioniert.

## Codeanalyse und Bewertung durch KI

Das Jupyter Notebook „Bewertung\_und\_konstruktive\_Kritik\_von\_Code\_durch\_KI“ bietet eine Anleitung zur Bewertung und Verbesserung von Code durch KI in einem sehr offen ausgelegten Beispiel. Dabei werden zunächst klare Kriterien für die Codequalität definiert. Die Kriterien umfassen Lesbarkeit, Korrektheit, Modularität, Dokumentation, Best Practices und Skalierbarkeit, die im Verlauf des Dokuments detailliert behandelt werden. Die Anleitung stellt dabei nach vorheriger Einleitung und Erläuterung einen Prompt zur Verfügung, der in ein KI Chatfenster von ChatGPT kopiert werden kann, das anschließend den eigenen Code analysiert. Danach wird Feedback von der KI generiert welches sowohl positive als auch negative Aspekte zu jedem der definierten Kriterien enthalten soll. Zudem wird auf einer Skala von 1 bis 10 eine Bewertung vorgenommen, wobei 10 Punkte die Höchstpunktzahl ist. Zum Schluss wird darauf hingewiesen, dass das KI-Ergebnis als Hilfestellung zu verstehen ist und Fehler nicht auszuschließen sind. Wichtig ist, dass das Ergebnis nur als ein grober Anhaltspunkt dient und es auch immer kritisch hinterfragt werden muss. Einige hier nicht näher erläuterte Tests haben ergeben, dass die Bewertung mit sehr offenen und breit gestreuten Kriterien zwar funktioniert, jedoch besser noch um einige spezifischere Kriterien zur Aufgabenstellung ergänzt werden sollten. Außerdem ist zu beachten, dass die KI nicht immer korrekt liegen muss. Der Wert für Korrektheit der Antwort kann dabei stark schwanken. Einen Anhaltspunkt bietet jedoch beispielsweise ein Test, wie von der George Mason University (USA) durchgeführt. Bei diesem hat die KI nur 44% korrekte Antworten für einen anerkannten Test für Studierende im Bereich Softwareentwicklung geliefert. (Jalil, Rafi, D. LaToza, Moran, & Lam, 2023)

## Tipps und Tricks für Prompt Engineering

Zuletzt wurden die im Projekt und sonstigen gewonnen Erfahrungen zu guten und schlechten Prompts zur Kommunikation mit künstlicher Intelligenz in dem Dokument „Tipps\_und\_Tricks\_fuer\_ChatGPT“ zusammengefasst und mit Beispielen erläutert. Dabei wird zunächst darauf eingegangen, welche allgemeinen Tipps es bei der Kommunikation mit KI zu beachten gibt. Dazu zählen unter anderem Aspekte wie Ausdrucksweise, präzise Formulierungen oder auch Hinweise auf die zu verwendende Sprache. Anschließend wird auf verschiedene Integrationen von ChatGPT aus Abschnitt 3.1 eingegangen, wo deren Vorteile liegen und was für Möglichkeiten dort bestehen. Im Hauptteil werden danach Prompts anhand von fünf Beispielen erläutert. Die Beispiele sind dabei immer so aufgebaut, dass zunächst ein schlechter Prompt gezeigt wird und danach eine verbesserte Version davon dargestellt ist. Die Antwort der KI lieferte dabei stets Programmcode in C++, der direkt in das Jupyter Notebook kopiert wurde und nun direkt ausgeführt werden kann. Die Beispiele sind recht simpel gehalten, verdeutlichen jedoch gut wie groß der Einfluss der Wortwahl des Prompts auf das Ergebnis ist. Abgeschlossen wird das Jupyter Notebook von einer Zusammenstellung von guten Command Prompts in Form eines „Cheet Sheets“.

## Anleitungen zu Github Copilot mit Erfahrungswerten

Im Verlauf des Projekts wurde wie bereits beschrieben auf die Nutzung von Github Copilot zurückgegriffen. Die Registrierung für einen Account und die Installation für entsprechende IDEs wurde von den Herstellern vorgegeben. In den folgenden Anleitungen sind Verlinkungen zu diesen Anleitungen hinterlegt. Falls die Anleitungen nicht ausreichend oder lückenhaft waren, wurde eine eigene Anleitung mit Hinweisen geschrieben. Außerdem wurden in den Dokumenten Hinweise für möglicherweise auftretende Fehler und persönliche Erfahrungen ergänzt. Dies umfasst folgende Dokumente

* Anleitung Github Copilot Registrierung für Studierende
* Anleitung Github Copilot Integration in QT
* Anleitung Github Copilot Integration in VS Code

# Fazit

In dem Softwareprojekt wurde das LLM von OpenAI ausgiebig getestet und Erfahrungen gesammelt, woraus praktische Beispiele erarbeitet und in Jupyter Notebooks dokumentiert wurden. Ziel war es, Github Copilot als Werkzeug für Refactoring, Codebewertung und generelle Entwicklungstipps zu nutzen, um den Lernprozess im Studium zu verbessern.

Dabei hat sich Github Copilot als ein zuverlässiges und sinnvolles Tool dargestellt, um Programmcode in verschiedensten Varianten zu generieren oder zu optimieren. Die Integration von GitHub Copilot in verschiedene IDEs ermöglicht eine nahtlose Verwendung dieser KI-gestützten Funktionen direkt im Lernprozess. Studierende können nun von automatisierten Refactoring-Vorschlägen profitieren, die dazu beitragen, Codequalität und -Lesbarkeit zu verbessern. Darüber hinaus bietet Github Copilot wertvolle Hinweise und Tipps während des Schreibens von Code, was zu einer schnelleren und effektiveren Lösungsentwicklung führt. Durch die automatisierten Vorschläge und Feedbackmechanismen von Github Copilot können Studierende den Lernfortschritt besser verfolgen und so gezielt ein 1 zu 1 Coaching umsetzen, was im normalen Lernalltag sehr schwierig umzusetzen ist.

Abschließend lässt sich sagen, dass Github Copilot eine Bereicherung für die Studierenden ist und die Integrationen in QT sowie VS Code eine nutzerfreundliche Interaktion ermöglichen. Sofern den Studierenden ein reflektierter Umgang mit der Nutzung nahegelegt wird, ist ein Einsatz in der Lehre definitiv zu empfehlen.

# Anhang

## Tipps\_und\_Tricks\_fuer\_ChatGPT.ipynb

Siehe Datei im Anhang und unter (Walldorf, 2024)

## code\_Refactoring\_und\_Optimierung\_mit\_KI.ipynb

Siehe Datei im Anhang und unter (Walldorf, 2024)

## Bewertung\_und\_konstruktive\_Kritik\_von\_Code\_durch\_KI.ipynb

Siehe Datei im Anhang und unter (Walldorf, 2024)

## Anleitung Github Copilot Registrierung fuer Studenten.txt

Siehe Datei im Anhang und unter (Walldorf, 2024)

## Anleitung Github Copilot Integration in QT.txt

Siehe Datei im Anhang und unter (Walldorf, 2024)

## Anleitung Github Copilot Integration in VS Code.txt

Siehe Datei im Anhang und unter (Walldorf, 2024)

# Literaturverzeichnis

Chornaya, J. (09. 02 2024). *hubspot.de*. Von hubspot.de: https://blog.hubspot.de/website/ki-code abgerufen

Copilot, G. (3 2024). *Github Copilot*. Von Github Copilot: https://github.com/features/copilot abgerufen

Jalil, S., Rafi, S., D. LaToza, T., Moran, K., & Lam, W. (08. 02 2023). ChatGPT and Software Testing Education:. George-Mason-University Fairfax, USA.

OpenAI. (3 2024). *OpenAI*. Von OpenAI: https://openai.com/ abgerufen

Walldorf, J. (03 2024). *Github - Softwareprojekt*. Von Github - Softwareprojekt: https://github.com/jwalldorf96/Softwareprojekt abgerufen