**📘 Projektdokumentation – Einsatz von KI beim „Ultimate File Renamer“**

**1️⃣ Verwendetes Modell**

Für die Entwicklung und Refaktorierung des Projekts wurde **OpenAI GPT-5 (ChatGPT)** eingesetzt.  
Das Modell diente als interaktiver Entwicklungsassistent für:

* die Erstellung und Verbesserung von **C#-Code**,
* die **Strukturierung** der Projektarchitektur,
* die **Erstellung technischer Dokumentation**,
* sowie für **Übersetzungen und Formulierungen** in mehreren Sprachen.

**2️⃣ Eingesetzte Prompts**

| **Nr.** | **Ursprüngliche Intention** | **Korrigierte Promptformulierung** |
| --- | --- | --- |
| 1 | Bestätigungsfrage nur bei Rename anzeigen | „Passe das Programm so an, dass die Bestätigungsabfrage nur angezeigt wird, wenn der Benutzer den Rename-Vorgang explizit auswählt.“ |
| 2 | Vorherigen Pfad beim erneuten Rename verwenden | „Implementiere eine Logik, die beim erneuten Starten des Umbenennungsvorgangs (R) den zuletzt genutzten Ordner beibehält.“ |
| 3 | Code in kleinere Teile aufteilen | „Refaktoriere das Projekt in mehrere klar getrennte Klassen und Module, um Wartbarkeit und Übersichtlichkeit zu verbessern.“ |
| 4 | Erklärung der Klassenstruktur | „Erstelle eine kurze Beschreibung der Aufgabe und Verantwortung jeder Klasse im refaktorierten Code.“ |
| 5 | Dokumentation auf Deutsch | „Übersetze die Klassenerklärungen und Projektdokumentation vollständig ins Deutsche.“ |
| 6 | Vollständige README | „Erstelle eine vollständige README-Datei mit Projektbeschreibung, Funktionsübersicht und Anwendungsanleitung.“ |
| 7 | Refactorspezifikation | „Erstelle ein professionell formuliertes Dokument, das alle Entwicklungsschritte und Änderungen zusammenfasst.“ |

**3️⃣ Weitere Interaktionen mit der KI**

Während des Entwicklungsprozesses waren mehrere Rückfragen notwendig — insbesondere bei der Anpassung der Benutzerführung, beim Umgang mit Regex-Mustern und bei der Umstrukturierung der Matcher-Klasse.  
Die Kommunikation mit der KI erfolgte schrittweise, um den Code testbar zu halten.

**4️⃣ Anteil der KI-Erstellung**

Etwa **70 % des Codes und der Dokumentation** wurden direkt durch die KI generiert.  
**30 %** wurden anschließend manuell überarbeitet, getestet und erweitert, insbesondere für Dateihandling, Regex-Anpassungen und Konsoleninteraktionen.

**5️⃣ Qualität der Ergebnisse**

Die KI lieferte technisch einwandfreie und sofort ausführbare Codebeispiele.  
Besonders hilfreich war die Fähigkeit, Code **strukturiert und kommentiert** zu erzeugen.  
Kleinere manuelle Anpassungen waren erforderlich, um:

* Eingabefehler abzufangen,
* Benutzerinteraktionen zu verbessern,
* und Dateipfade korrekt zu behandeln.

**6️⃣ Anpassungen am generierten Code**

Der KI-generierte Code wurde größtenteils **direkt übernommen**, jedoch an einigen Stellen überarbeitet:

* Menülogik erweitert (Option „R“ = Rename erneut / „C“ = Change folder)
* Regex-Matching optimiert
* Temporäre Dateien umbenannt, um Konflikte zu vermeiden
* Eingabeaufforderungen an das Benutzerverhalten angepasst

**🧭 Erfahrungsbogen – Umgang mit KI während der Entwicklung**

Die Zusammenarbeit mit der KI verlief insgesamt produktiv, erforderte aber auch menschliche Aufmerksamkeit und Nacharbeit.  
Im Folgenden sind die wichtigsten Erfahrungen und Herausforderungen zusammengefasst.

**✅ Positive Erfahrungen**

* Die KI konnte **komplexe C#-Strukturen** präzise umsetzen.
* Sie half beim **Refactoring** durch logische Code-Aufteilung und klare Funktionsbenennung.
* Der **Zeitaufwand** für Dokumentation und Architekturplanung wurde stark reduziert.
* Die zweisprachige Unterstützung erleichterte die **Teamkommunikation**.
* Durch die Schritt-für-Schritt-Erklärung konnten **Teammitglieder mit weniger C#-Erfahrung** schnell folgen.

**⚠️ Schwierigkeiten & Lösungsansätze (von uns aus)**

| **Herausforderung** | **Beschreibung** | **Umgang / Lösung** |
| --- | --- | --- |
| **Session-Verlust** | Bei neuen KI-Sitzungen ging der Gesprächskontext verloren. | Wichtige Prompts wurden manuell gesichert und erneut geladen. |
| **Unverständliche Antworten** | Manche Ausgaben waren zu technisch oder mehrdeutig. | Prompts wurden präziser formuliert und in Alltagssprache überführt. |
| **Zu viel Code auf einmal** | Lange Codeblöcke erschwerten Tests und Übersicht. | Code in kleinere logische Module aufgeteilt. |
| **Git-Probleme mit .vs-Dateien** | Visual Studio generierte gesperrte Dateien. | .gitignore erstellt und Index bereinigt. |
| **Sprachmischung** | KI antwortete teils auf Englisch. | Einheitliche Kommunikation auf Deutsch festgelegt. |
| **Kein permanenter Kontext** | Nach jedem Reset musste Kontext neu erklärt werden. | Protokoll aller Prompts in separater Datei geführt. |
| **Fehlinterpretationen durch das Team** | Einige KI-Vorschläge wurden missverstanden. | Gemeinsame Code-Reviews halfen bei der Klärung. |
| **Unterschiedliche Arbeitsgeschwindigkeit** | KI arbeitete schneller als manuelles Testen. | Regelmäßige Pausen für Validierung eingeplant. |

**💬 Fazit**

Der Einsatz von GPT-5 als Entwicklungshelfer im Projekt **„Ultimate File Renamer“** erwies sich als sehr wirkungsvoll.  
Die KI konnte wiederkehrende Aufgaben automatisieren, komplexe Strukturen verständlich darstellen und durch klare Vorschläge den Entwicklungsprozess beschleunigen.

Trotz einzelner Schwierigkeiten — insbesondere mit Sitzungsunterbrechungen, technischen Erklärungen und menschlichen Missverständnissen — überwiegt der Nutzen deutlich.  
Die Kombination aus **menschlicher Kontrolle und KI-Unterstützung** führte zu einem hochwertigen, wartbaren und gut dokumentierten Ergebnis.

**🔖 Zusammenfassung**

| **Kategorie** | **Bewertung** |
| --- | --- |
| Technische Qualität | ⭐⭐⭐⭐☆ (4.5 / 5) |
| Verständlichkeit der KI-Antworten | ⭐⭐⭐⭐☆ |
| Notwendige Nacharbeit | Mittel |
| Dokumentationshilfe | Sehr hoch |
| Gesamtergebnis | **Erfolgreiche Integration von KI in die Softwareentwicklung** |