hochzuladen. Das Backend verarbeitet die Datei und fügt die Informationen in die Antwort ein, die dann an das Frontend gesendet wird. Nach erfolgreicher Verarbeitung erhält der Benutzer einen Link zum Herunterladen des Berichts. Die Backend-Entwicklung basiert auf wichtigen Bibliotheken wie ADM-ZIP für ZIP-Dateien, Bottleneck für Anforderungsratekontrolle, und Express für Backend-Entwicklung. Weitere Bibliotheken werden für Dateiverwaltung, Word-Dokumenterstellung und Dateiuploads verwendet. Die LaTeX-Zip-Verarbeitung beinhaltet die Suche nach Hauptdateien, die Extraktion von Teiltexten und die Bereinigung für die AI-Zusammenfassung. Die AI-Generierung erfolgt über OpenAI, mit Stopwörtern zur Spracherkennung. Zudem wird DeepL für Kapitelübersetzungen genutzt. In der Frontend-Entwicklung werden Bibliotheken für React-Tests, Rendering und Routing verwendet. Die Benutzeroberfläche ist klar strukturiert, mit verschiedenen Seiten für Upload und Download, und bietet eine einfache, benutzerfreundliche Nutzung. Der Dateidownload-Prozess wird durch Datenübergabe und React-Hooks realisiert, um die Datei herunterzuladen.

Evaluation und Ergebnisse

Das Kapitel "Evaluation und Ergebnisse" beschreibt die Bewertung des Konfigurationsmanagements, die Kostenanalyse der API-Nutzung, die Qualität der Berichte, den Vergleich mit anderen Tools und wichtige Erkenntnisse aus dem Entwicklungsprozess. Es werden Anleitungen zur Schlüsselgenerierung für OpenAI und DeepL gegeben. Die Anwendung kann lokal betrieben werden und erfordert die Konfiguration von Umgebungsvariablen. Es wird auch ein Vergleich mit anderen Tools wie Chat, Copilot, Gemini und QuillBot durchgeführt, wobei Reforge als benutzerfreundlicher und konsistenter hervorgeht. Verschiedene Kostenmodelle und Funktionalitäten der genutzten APIs werden aufgezeigt, sowie Lessons Learned aus dem Projekt präsentiert für zukünftige Projekte.

Projektausblick von Reforge

In dem Kapitel "Projektausblick von Reforge" werden mögliche Erweiterungen und Verbesserungen für LaTeX-Dokumente vorgestellt. Es wird diskutiert, wie die Textbereinigung optimiert werden kann, um wichtige Codeabschnitte nicht zu filtern. Eine bessere Textbereinigung könnte auch Kommentare einbeziehen und das Literaturverzeichnis automatisch in die Ausgabe integrieren. Die Einführung einer Datenbank zur langfristigen Speicherung von Daten wird vorgeschlagen. Weitere Verbesserungen könnten die Berücksichtigung von Bildern, mehr Flexibilität im Frontend und die Möglichkeit des Berichtsversands per E-Mail sein. Es wird auch darauf hingewiesen, dass bei einer öffentlichen Nutzung der Anwendung Beschränkungen implementiert werden müssen, um Betriebskosten zu kontrollieren.

Fazit

Im Fazit der Bachelorarbeit wird die Entwicklung einer Webanwendung zur automatischen Textzusammenfassung und -generierung in verschiedenen Formaten wie IEEEtran oder dem OTH-Forschungsbericht zusammengefasst. Die Anwendung unterstützt mehrsprachige Ausgaben in Deutsch und Englisch und bietet eine Plattform für Studierende und Lehrende zur schnellen Erstellung technischer Berichte. Sie integriert moderne Sprachmodelle wie OpenAI und DeepL zur Textzusammenfassung und Übersetzung. Die Ergebnisse zeigen, dass die Nutzung des Generators den Zeit- und Arbeitsaufwand verringern kann, jedoch auch menschliche Validierung erfordert. Alle definierten Ziele und Anforderungen des Projekts wurden erfüllt, wodurch die Anwendung effizient und funktionsfähig ist.

Bilder

In dem Abschnitt "Bilder" wird über das Thema Bilder gesprochen.

Codeauszüge

Der Abschnitt "Codeauszüge" in einem Text oder Dokument wird thematisiert.

Literaturverzeichnis

Dieser Bericht wurde aus der hochgeladenen Datei mit OpenAI generiert.
Alle Quellen müssen vor einer Veröffentlichung noch manuell ergänzt werden.
Name der Datei: BT2024StrickerNatalieThesis.zip