**Phase 2: Erarbeitungs- und Reflexionsphase – Architekturdokument**

1. **Technologieübersicht**

**1.1 Programmiersprache**

Die für dieses Projekt ausgewählte Programmiersprache ist C++, folgend werden die wichtigsten Gründe für die Auswahl erläutert.

* Motivation: Vertiefung der eigenen Kenntnisse, wichtig für Ingenieurs- und Software-Engineering Berufe
* Fördert allgemeines Verständnis für Speicherverwaltung, effiziente Algorithmen und systemnahe Entwicklung
* Große Ökosystem, viele Libraries und plattformübergreifend einsetzbar
* Auch wenn Entwicklung schwerer als bspw. mit Python ist, ist der Lerneffekt für Studium/Beruf höher
* GUI-Anbindung mit Qt

**1.2 Frameworks**

In diesem Projekt werden primär die Frameworks Qt (Qt6) und googletest verwendet. In diesem Abschnitt werden die Gründe dafür erklärt.

* Qt (Qt6)
  + Wird für grafische Oberfläche (GUI) verwendet
  + Ermöglicht plattformübergreifende Desktop-Entwicklung
  + Gute Unterstützung für UI-Design, Events und eigene Widgets
  + Integration in C++ Projekt und Build-System (CMake) ist Standard
* googletest
  + Framework für automatisierte Unit-Tests
  + Weit verbreitet im C++ Umfeld, auch im industriellen Einsatz
  + Moderne C++ Features für Testfälle und Test Suites
  + Bringt professionelle Entwicklungsmethodik (Test Driven Development, Absicherung von Code…)

**1.3 Bibliotheken**

Bisher werden im Projekt folgende Bibliotheken verwendet, diesen können sich jedoch je nach Projektablauf noch verändern.

* C++ Standardbibliothek (STL): Für Vektoren, Strings, Maps usw.
* QtCore, QtGui, QtWidgets

**Entwicklungswerkzeuge**

* VS Code mit C++ und Qt Extension
* Qt Creator 17.0.0
* CMake als Buildsystem
* GitHub zur Versionskontrolle
* windeployqt zum Erstellen von Release-Paketen (Deployment für Windows)

1. **Architekturübersicht**

In diesem Projekt habe ich mich auf eine Schichtenarchitektur entschieden (3-Layer). Diese ist übersichtlich, leicht erweiterbar und testbar. Für das Projekt ist diese Architektur meiner Meinung nach ausreichend.

* **UI-Schicht:** Besteht aus den Klassen MainWindow und den Dialogklassen
  + Zuständig für grafische Anzeige, Benutzerinteraktion und Aufruf der Logik
* **Logikschicht:** Hauptkomponente ist Klasse TaskManager
  + Aufgaben hinzufügen, löschen, bearbeiten, speichern…
  + Schnittstellt zwischen UI und Datenhaltung
* **Datenhaltung:** durch Klasse Task und Datei-Verwaltung innerhalb TaskManager realisiert
  + Aufgaben werden lokal als .csv gespeichert

1. **Struktur**

**3.1 Hauptkomponenten**

* MainWindow
  + Verantwortlich für das Hauptfenster und die Steuerung der Oberfläche
  + Zeigt die Liste der offenen Aufgaben und bietet Buttons für Aktionen
  + Öffnet bei Bedarf die verschiedenen Dialogfenster
* DialogTaskHinzufuegen / DialogTaskBearbeiten
  + Separate Dialogfenster für das Hinzufügen bzw. Bearbeiten einer Aufgabe
  + Erfassen und Bearbeiten der Aufgabendaten durch den Benutzer
* erledigteAufgabenliste / ueberfaelligeAufgaben
  + Dialogfenster zur Anzeige aller erledigten bzw. überfälligen Aufgaben
* TaskManager
  + Verwalten der Sammlung aller Aufgaben (Hinzufügen, Löschen, Suchen, Bearbeiten)
  + Schnittstelle zwischen UI und Dateispeicherung
  + Verantwortlich für das Speichern und Laden der Aufgaben
* Task
  + Datenklasse für einzelne Aufgaben (Attribute: Titel, Beschreibung, Fälligkeitsdatum, Status, Index)
  + Methoden zum Ändern und Auslesen der Aufgabendaten

**3.2 Abhängigkeiten**

* MainWindow nutzt TaskManager für Logik und Aufgabeliste
* TaskManager verwaltet eine Sammlung von Task-Objekten
* MainWindow öffnet die Dialogfenster zur Interaktion mit dem Benutzer
* Dialogfenster übergeben ihre Eingaben an MainWindow/TaskManager

Abbildung : UML-Klassendiagramm

**Ein Bild, das Text, Screenshot, Software, Design enthält.

KI-generierte Inhalte können fehlerhaft sein.**

Quelle: Eigene Darstellung mit Hilfe von Visio

1. **Verhalten**
2. Der Benutzer gibt die Aufgabendaten im Dialog ein
3. MainWindow ruft addAufgabe im TaskManager auf
4. TaskManager legt ein neues Task-Objekt an und speichert alle Aufgaben in der Datei
5. MainWindow aktualisiert die Aufgabenliste in der Benutzeroberfläche

Abbildung 2: UML-Sequenzdiagramm

Ein Bild, das Text, Screenshot, Reihe, parallel enthält.

KI-generierte Inhalte können fehlerhaft sein.

Quelle: Eigene Darstellung