



INTERNATIONALE
HOCHSCHULE

Projekt: Software Engineering

DLMCSPSE01_D

Erstellt von:	Sebastian Ahlburg
Matrikelnummer:	9228566
Studiengang:	Master of Science Informatik
Tutor:	Prof. Markus Kleffmann
Datum:	21.04.2024

Inhaltsverzeichnis

Inhaltsverzeichnis	I
Abbildungsverzeichnis	III
Tabellenverzeichnis.....	IV
1. Projektplan	1
1.1 Ziele, Umfang und angestrebtes Ergebnis des Projektes	1
1.2 Zielgruppe.....	1
1.3 Projektrisiken und Gegenmaßnahmen.....	2
1.4 Zeitplan und Meilensteine.....	3
2. Anforderungsdokument	7
2.1 Management Summary	7
2.2 Systemumfang und Kontext	8
2.3 Funktionale Anforderungen	9
2.4 Nicht-funktionale Anforderungen	13
2.5 Glossar	14
3. Projektdokumentation	15
3.1 Vorgehensmodell für die Entwicklungsphase	15
3.2 Tech-Stack.....	15
3.2.1 Programmiersprache	15
3.2.2 Framework und Bibliotheken	16
3.3 Entwurfsmuster (Design Pattern)	17
3.4 Struktur der Anwendung.....	17
3.5 Interaktion Nutzer und Anwendung.....	20
3.6 Revisionskontrolle und Nutzung von GitHub.....	21
3.7 Benutzerhandbuch	22

4. Testprotokoll	27
4.1 Unittests	28
4.2 Systemtest	40
4.3 Anwendungstest Windows.....	46
5. Abstract	48
5.1 Making-of	48
5.2 Reflexion	48
Literaturverzeichnis	51
Verzeichnis der Anhänge	53
A.1 – Automatisierte Testprotokolle.....	53
A.1.1 Testprotokoll ToDoItem	53
A.1.2 Testprotokoll DatabaseManager	53
A.1.3 Testprotokoll UserInterface.....	54
A.1.4 Testprotokoll TaskManager	54

Abbildungsverzeichnis

ABBILDUNG 1: GANTT-CHART (ZEITPLANUNG DER MEILENSTEINE)	6
ABBILDUNG 2: UML-ANWENDUNGSFALLDIAGRAMM	9
ABBILDUNG 3: UML-KLASSENDIAGRAMM	18
ABBILDUNG 4: UML-AKTIVITÄTSDIAGRAMM	21
ABBILDUNG 5: BENUTZEROBERFLÄCHE	23
ABBILDUNG 6: BENUTZEREINGABE	24
ABBILDUNG 7: ÄNDERN TODO EIGENSCHAFTEN	25
ABBILDUNG 8: TODO ITEM LÖSCHEN	25
ABBILDUNG 9: SORTIERFUNKTION	25
ABBILDUNG 10: AUSFÜHRUNG MIT WINE	46
ABBILDUNG 11: INTERFACE DARSTELLUNG UNTER WINDOWS	47

Tabellenverzeichnis

TABELLE 1: MEILENSTEINE INKLUSIVE GESCHÄTZTER ZEITAUFWAND	5
TABELLE 2: ERSTE USER STORY – TO-DO ERSTELLEN	10
TABELLE 3: ZWEITE USER STORY – TO-DO BEARBEITEN	10
TABELLE 4: DRITTE USER STORY – TO-DO LÖSCHEN.....	11
TABELLE 5: VIERTE USER STORY – TO-DO ANZEIGEN	11
TABELLE 6: FÜNFTE USER STORY – KATEGORISIERUNG NACH STATUS.....	12
TABELLE 7: SECHSTE USER STORY – PRIORISIERUNG VON AUFGABEN	12
TABELLE 8: SIEBTE USER STORY – FÄLLIGKEITSDATUM FESTLEGEN	13
TABELLE 9: ATTRIBUTE UND METHODEN VON „ToDoItem“.....	19
TABELLE 10: ATTRIBUTE UND METHODEN VON „DatabaseManager“.....	19
TABELLE 11: ATTRIBUTE UND METHODEN VON „TaskManager“	20
TABELLE 12: ATTRIBUTE UND METHODEN VON „UserInterface“	20
TABELLE 13: TESTBESCHREIBUNG „UNITTESTS“	39
TABELLE 14: TESTBESCHREIBUNG „SYSTEMTEST“	45

1. Projektplan

1.1 Ziele, Umfang und angestrebtes Ergebnis des Projektes

Ziel dieses Projekts ist die Entwicklung einer To-Do-Listen-Anwendung, die sich durch ein ästhetisch ansprechendes Design sowie eine hohe Benutzerfreundlichkeit auszeichnet. Im Mittelpunkt steht die Fokussierung auf einige wenige Kernfunktionen, die durch ihr reduziertes Design zu einer intuitiven Anwendbarkeit führen und damit das Nutzererlebnis als Ganzes verbessern.

Der **Projektumfang** wird durch die grundlegenden Funktionen einer To-Do-Liste (Erstellen, Bearbeiten, Löschen und Anzeigen von Aufgaben) und deren Implementierung in einem Kanban-Stil definiert, wodurch eine klare und fokussierte Nutzererfahrung gewährleistet werden soll. Ich möchte so sicherstellen, dass die Anwendung einfach und frei von Ablenkungen ist und beabsichtige dabei Design-Prinzipien einzuhalten, die sowohl Funktionalität als auch visuelle Anziehungskraft in Einklang bringen. Zusätzlich gilt es im den Vorgaben einer ausführliche Dokumentation des Projektes gerecht zu werden.

Das **Ergebnis** des Projektes soll eine leistungsfähige Desktop-Anwendung für Windows darstellen, die eine effiziente Aufgabenverwaltung ermöglicht und dabei sowohl praktisch als auch visuell ansprechend ist. Die Anwendung soll sich durch ein individuelles Design und eine intuitive Benutzung von anderen To-Do-Listen-Anwendungen abheben. Das Produkt wird gemäß den vorgegebenen Software-Engineering-Methoden entwickelt, wobei ich Python als eine der zugelassenen objektorientierten Programmiersprachen nutzen werde. Die Anwendung wird eine selbsterklärende, grafische Benutzeroberfläche haben und unter Windows 10/11 lauffähig sein. Der gesamte Quellcode und die Projektdokumentation werden auf GitHub verfügbar sein, um die Nachvollziehbarkeit und Transparenz des Projekts zu gewährleisten.

1.2 Zielgruppe

Die identifizierte Zielgruppe für die To-Do-Listen-Anwendung zeichnet sich durch eine Affinität zu Design und einem minimalistischen Lebensstil aus. Diese Nutzer sind bestrebt, ihren Alltag effizient und fokussiert zu gestalten, indem sie sich auf das Wesentliche konzentrieren. Sie

bevorzugen dabei Produkte, die nicht nur funktional, sondern auch ästhetisch ansprechend sind. Ihr Ansatz ist es, durch ein klares, reduziertes Design Ablenkungen zu minimieren und so mehr Zeit für die wirklich wichtigen Dinge im Leben zu haben. Diese Gruppe umfasst Personen, die sowohl im beruflichen als auch im privaten Bereich Wert auf Ordnung, Struktur und eine harmonische Umgebung legen. Die Anwendung soll diesen Bedürfnissen durch eine Kombination aus Funktionalität, Einfachheit und visueller Anziehungskraft gerecht werden.

1.3 Projektrisiken und Gegenmaßnahmen

Die Risiken im Projekt setzen sich aus zwei wesentlichen Säulen zusammen. Einerseits treten wiederkehrende Risiken auf, die bei vielen Softwareentwicklungsprojekt auftreten. Andererseits treten Risiken auf, die mit der geringen persönlichen Erfahrung im Bereich der Softwareentwicklung im Zusammenhang stehen. Zu Beginn des Projektes wurde daher eine intensive Internetrecherche betrieben, um die häufigsten Probleme im Entwicklungsprozess zu identifizieren. Ergänzt wurde diese durch ein Brainstorming zu potenziellen Schwierigkeiten und Herausforderungen mit persönlichem Bezug. Nachfolgend werden die Ergebnisse sowie die dazugehörigen Gegenmaßnahmen dargestellt:

Mangelhafter Product-Market-Fit (Allgemein)

<u>Risiko:</u>	Die Anwendung könnte nicht den Erwartungen der Zielgruppe entsprechen oder nicht intuitiv genug sein. Eintrittswahrscheinlichkeit: Mittel Auswirkungen: Hoch
<u>Gegenmaßnahme:</u>	Umfangreiche Zielgruppen- und Marktanalyse sowie entwickeln eines MVP, um vor Abgabe der Projektarbeit ein erstes Nutzerfeedback zu erhalten. Strategie: Verminderung

Technologische Herausforderungen (Allgemein / Persönlich)

<u>Risiko:</u>	Eine falsche Wahl der Technologie nimmt einen negativen Einfluss auf die Umsetzbarkeit und Leistung der Anwendung. Ggf. werden aufgrund
----------------	---

der fehlenden Erfahrung zusätzlich die Auswirkungen falsch eingeschätzt und zu lange an der Entscheidung festgehalten.

Eintrittswahrscheinlichkeit: Mittel

Auswirkungen: Mittel

Gegenmaßnahme: Intensive Nutzung des Betreuungs- und Feedbackangebotes durch das Lehrpersonal.

Strategie: Vermeidung

Zeitmanagement (Persönlich)

Risiko: Unterschätzen der benötigten Zeit für die Entwicklung und das Testen der Anwendung sowie zeitliche Konflikte mit dem regulären Vollzeitjob.

Eintrittswahrscheinlichkeit: Sehr hoch

Auswirkungen: Gering

Gegenmaßnahme: Aufstellen eines Zeitplanes mit ausreichend Pufferzeiten, um bei Auftreten von Herausforderungen reagieren zu können. Zusätzlich einteilen von möglichst kleinen Arbeitspaketen, die aufgrund ihres Umfangs gut in die reguläre Arbeitswoche integrierbar sind. Verlängerung bis zur Projektabgabe einplanen.

Strategie: Akzeptieren

1.4 Zeitplan und Meilensteine

Basierend auf den drei vorgegebenen Phasen für das Projekt sowie den entwickelten Anforderungen an die To-Do-Listen-Anwendung wurde nachfolgender Zeitplan entwickelt. Insgesamt wurde ein Zeiteinsatz von 220 Stunden geschätzt (siehe Tabelle 1). Dabei entfallen 66 Stunden auf die Konzeptionsphase (ca. 30%), 122 Stunden auf die Erarbeitungs- und Reflexionsphase (ca. 55 %) sowie 32 Stunden auf die Finalisierungsphase (ca. 15 %). Die höhere zeitliche Gewichtung in den ersten beiden Phasen wird mit der Notwendigkeit begründet, zu Beginn eine möglichst solide konzeptionelle Grundlage zu schaffen, die im daran

anschließenden Entwicklungsprozess wiederum Zeit einspart. Zusätzlich wurde sich hier bewusst für eine vergleichsweise granulare Festlegung von Meilensteinen entschieden, damit sie als kleine Zwischenziele einfacherer im Alltag zu integrieren sind und schneller zu Erfolgserlebnissen führen. Sie stellen somit bereits die erste Maßnahme gegen eines der identifizierten Kernrisiken (Zeitmanagement) dar. Die Entscheidung mit Stunden anstatt Tagen als Zeiteinheit zu arbeiten wird damit begründet, dass diese besser in den persönlichen Kalender sowie den Ablauf eines berufsbegleitendes Studium integriert werden können. Ein erneutes Aufsplitten von Arbeitspaketen von Tagen in Stunden, um sie auf die laufende Arbeitswoche zu verteilen wird so vermieden.

Phase / Meilenstein		Zeitansatz in h
1. <u>Konzeptionsphase:</u>		
Meilenstein 1	Projektkonzept basierend auf einer vorausgegangenen Ideengenerierung mit intensiver Internetrecherche.	6
Meilenstein 2	Projektplan mit Projektziel, Zielgruppe, Risiken und Gegenmaßnahmen sowie einem Zeitplan mit Meilensteinen.	16
Meilenstein 3	Anforderungsdokument inklusive Management Summary und Festlegungen zu Systemumfang, Kontext, funktionalen und technischen Anforderungen.	16
Meilenstein 4	Projektdokumentation, die das Vorgehensmodell, die zugrundeliegenden Technologien und Tools sowie die Struktur der Anwendung darstellen.	28
2. <u>Erarbeitungs- und Reflexionsphase:</u>		
Meilenstein 5	Aufnahme des Feedbacks und Einarbeitung.	8
Meilenstein 6	Entwicklung der Grundstruktur der Anwendung und Implementierung der Kernfunktionen.	40
Meilenstein 7	Design und Implementierung der Benutzeroberfläche im Einklang mit den definierten Anforderungen.	38
Meilenstein 8	Erste Tests der Anwendung, Sammlung von Feedback und Darstellung der Testprotokollierung.	28

Meilenstein 9	Zusammenfassung der wichtigsten Design- und Implementierungsentscheidung sowie aller bisher erstellten Dokumente.	8
3. <u>Finalisierungsphase:</u>		
Meilenstein 10	Überarbeitung und Optimierung von Anwendung und Dokumentation basierend auf dem Feedback.	12
Meilenstein 11	Abschließende Tests und Finalisierung der Projektdokumentation.	10
Meilenstein 12	Erstellen einer Benutzeranleitung.	2
Meilenstein 13	Durchführen und Festhalten einer Lessons Learned.	4
Meilenstein 14	Abgabe des Projektes in der geforderten Form.	4

Tabelle 1: Meilensteine inklusive geschätzter Zeitaufwand

(Quelle: Eigene Darstellung)

Nichtsdestotrotz soll das Projekt in einen vorab festgelegten zeitlichen Rahmen von insgesamt 10 Wochen integriert werden. Hintergrund ist, dass auf diese Weise eine persönliche Deadline erzeugt werden soll, die für zusätzliche Motivation und ein vorgegebenes Ziel zweckdienlich erscheint. Nachfolgend werden die Meilensteine sowie ihre erwarteten Arbeitszeiten mit dem Gesamtzeitplan von 10 Wochen in einem Gant-Chart zusammengefasst.

Gantt Chart

Zeitplanung der Meilensteine

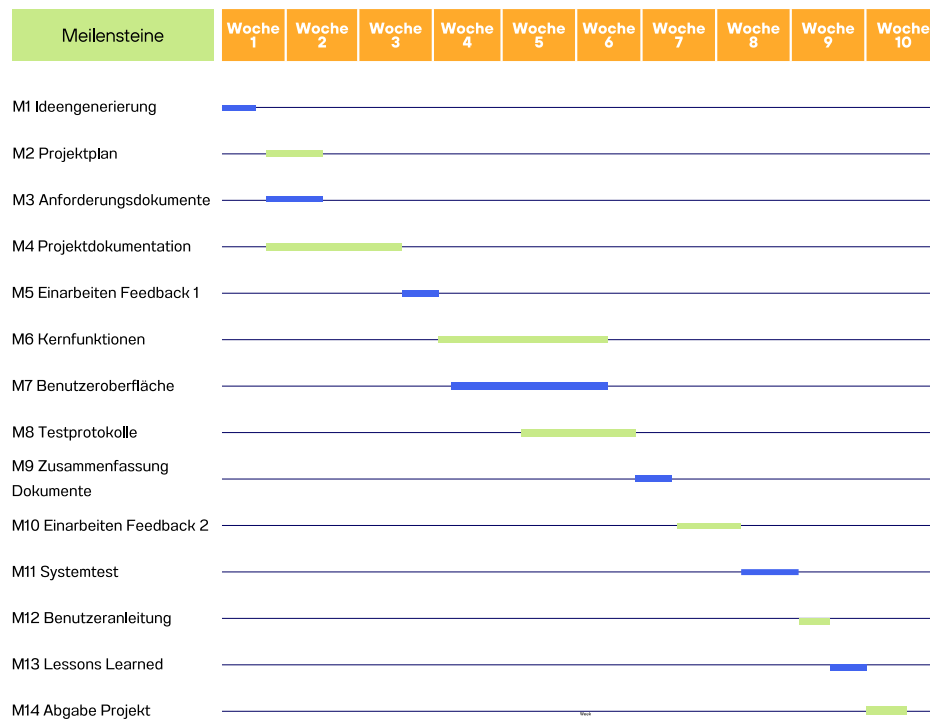


Abbildung 1: Gantt-Chart (Zeitplanung der Meilensteine)

(Quelle: Eigene Darstellung)

2. Anforderungsdokument

2.1 Management Summary

Dieses Projekt zielt auf die Entwicklung einer ästhetisch ansprechenden und minimalistischen To-Do-Listen-Anwendung für Windows ab. Der Fokus liegt auf einem benutzerfreundlichen und intuitiven Design, das sowohl die Effizienz als auch die Freude der Nutzer bei der Aufgabenverwaltung steigert. Kernfunktionalitäten (Erstellen, Bearbeiten, Löschen und Anzeigen von To-Dos) werden im Kanban-Stil dargestellt, um eine klare und strukturierte Übersicht zu bieten. Nutzer werden dadurch angehalten noch produktiver und fokussierter an ihren persönlichen Zielen zu arbeiten. Weiterhin soll sich die Anwendung nicht nur durch ihre einzigartigen Designmerkmale von bestehenden Lösungen abheben, sondern auch für heutige Nutzer besonders wichtige nicht-funktionale Anforderungen wie Leistung, Zuverlässigkeit, Kompatibilität und Sicherheit einhalten.

Die Entwicklung der Anwendung erfolgt in drei Phasen: Konzeption, Erarbeitung und Reflexion sowie Finalisierung. Dabei wird besonderer Wert auf die Einhaltung von Software-Engineering-Standards und eine effektive Projektmanagementstrategie gelegt. Durch ein iteratives Vorgehen soll zudem konstant Feedback von potentiellen Nutzern eingeholt werden, um nicht am Markt vorbei zu entwickeln. Für die Projektumsetzung ergibt sich daraus zudem die Möglichkeit zur stetigen Verbesserung eigener Standards, da so Raum für kontinuierliches Lernen entsteht. Dies spiegelt sich auch in einer vergleichsweise detaillierten Zeit- und Ressourcenplanung von insgesamt ca. 110 veranschlagten Arbeitsstunden wider, die durch einen granularen Aufbau mit klar definierten, kurzen Etappen und einem dazugehörigen Ergebnis geprägt ist. Ziel ist es auf diese Weise fortlaufende Erfolgserlebnisse zu produzieren, aber gleichzeitig eine Verschwendung zeitlicher Ressourcen durch zu lange und vom Arbeitsaufwand und -ergebnis schlechter abschätzbare Zyklen zu vermeiden.

Am Ende soll ein funktionales Produkt entstehen, welches gleichzeitig die Bedürfnisse einer design-affinen Zielgruppe anspricht, die einen minimalistischen Lebensstil bevorzugt. Ziel ist es so auf ganzheitliche Weise die Lebensqualität der Nutzer zu verbessern.

2.2 Systemumfang und Kontext

Systemumfang und Kontext der Anwendung werden in drei Schritten dargestellt. Zunächst wird der gewünschte Umfang sowie die Grenzen der Anwendung beschrieben. Im Anschluss sollen die Ergebnisse mit einem UML-Anwendungsfalldiagramm visualisiert werden.

Der **Umfang** der Anwendung wird durch die vier Aspekte (Kernfunktionen, Kanban-Style Darstellung, Minimalistisches Design und Benutzerinteraktion) geprägt.

Die *Kernfunktionen* beinhalten das Erstellen, Bearbeiten, Löschen und Anzeigen von Aufgaben.

Die *Kanban-Darstellung* ermöglicht das Verschieben von Aufgaben zwischen verschiedenen Statuskategorien (z.B. „To Do“, „In Progress“, „Done“).

Das *minimalistische Design* legt den Schwerpunkt auf eine klare, ästhetische Benutzeroberfläche, die den Nutzer die Anwendung gerne benutzen lässt.

Die *Benutzerinteraktion* ist auf intuitive Interaktionsmöglichkeiten, wie z.B. eine einfache Drag-and-Drop Funktion für Aufgaben, ausgerichtet.

Die **Grenzen** der Anwendung werden durch drei Aspekte (funktionale Beschränkungen, technische Beschränkungen und Designbeschränkungen) geprägt.

Die *funktionalen Beschränkungen* konzentrieren sich auf eine Vermeidung komplexer Funktionen, wie bspw. eine automatische Priorisierung, um die Einfachheit zu bewahren.

Die *technischen Beschränkungen* umfassen insbesondere die Interaktionsmöglichkeiten mit anderen Systemen. D.h. die Anwendung wird zunächst als lokale, standalone Desktop-Anwendung für Windows 10/11 ohne Cloud-Synchronisation oder Team-Funktionen entwickelt.

Die *Designbeschränkungen* zielen auf ein einheitliches Design und Benutzererfahrung ab, weshalb es, wenn überhaupt, nur sehr begrenzte Anpassungsoptionen für das Interface gibt.

UML-Anwendungsfalldiagramm

Zur Visualisierung wurde sich, wie in Abbildung 1 ersichtlich, für ein UML-Anwendungsfalldiagramm entschieden, da es sich besonders eignet, um die verschiedenen Funktionen der Anwendung und die Art und Weise, wie Benutzer mit diesen Funktionen interagieren, zu visualisieren (Ionos, 2020). Im Vergleich dazu stellt ein Kontextdiagramm eher die Wechselbeziehungen eines Systems mit externen Entitäten dar (Green, 2021), was für die

detaillierte Darstellung spezifischer Funktionalitäten der hier zu entwickelnden Anwendung weniger geeignet ist.

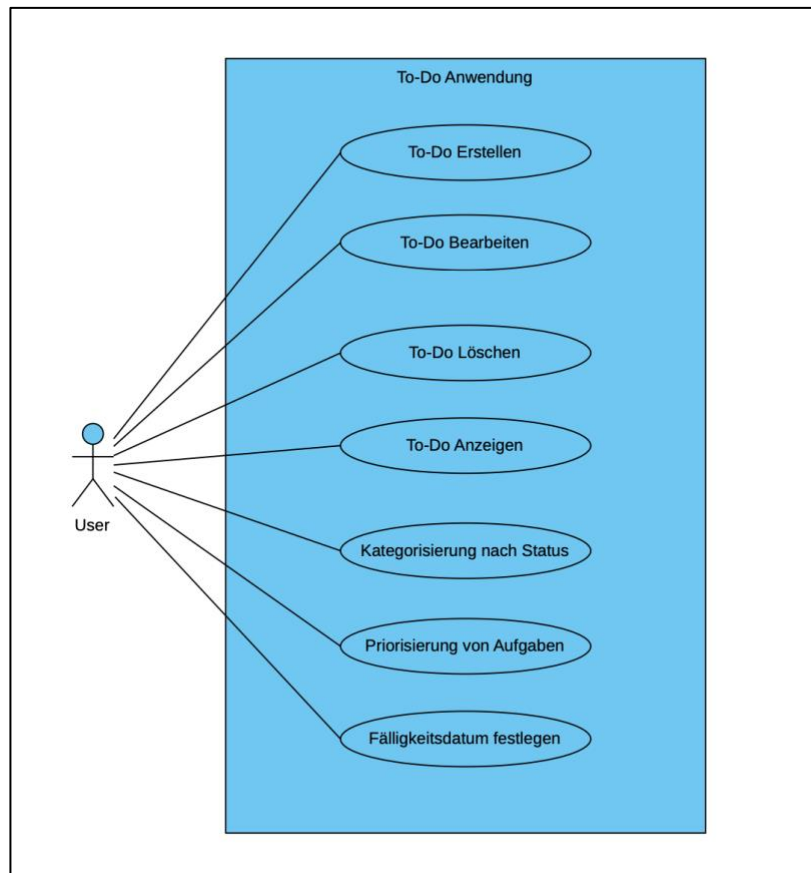


Abbildung 2: UML-Anwendungsfalldiagramm

(Quelle: Eigene Darstellung)

2.3 Funktionale Anforderungen

Die in der Anwendung bereitzustellenden Funktionalitäten werden im Folgenden in Form von User Stories dargestellt (siehe Tabelle 2 bis 8). Die Gründe für diese Entscheidung sind vielseitig. Erstens wurde sich in der Entwicklungsphase für ein agiles Vorgehensmodell in Anlehnung an Scrum entschieden, innerhalb dessen User Stories ein bekanntes und gern verwendetes Mittel sind (Koojiman, 2024). Zweitens konzentrieren sich User Stories auf den Endbenutzer und dessen Bedürfnissen (Rehkopf, 2024), wohingegen Use Cases einen stärkeren Fokus auf eine detaillierte und systematische Darstellung aller Funktionalitäten sowie deren Reaktion auf spezifische Benutzereingaben legen. Da im vorliegenden Fall der Schwerpunkt jedoch auf Benutzerfreundlichkeit und Ästhetik gelegt wird, eignen sich User Stories besser, da sich mit ihnen die Bedürfnisse und Wünsche der Zielgruppe direkt in den

Entwicklungsprozess einbeziehen lassen. Nachfolgend werden die einzelnen User Stories dargestellt.

Titel	To-Do Erstellen
Inhalt	Als ein Benutzer der To-Do-Listen-Anwendung, möchte ich die Möglichkeit haben, neue To-Dos schnell und einfach hinzuzufügen, damit ich meine anstehenden Aufgaben effektiv in die Anwendung eintragen und organisieren kann.
Akzeptanzkriterien	<ol style="list-style-type: none"> 1. Der Benutzer kann über eine Schaltfläche oder ein Eingabefeld ein neues To-Do erstellen. 2. Der Benutzer kann für das To-Do einen Titel und eine optionale Beschreibung eingeben. 3. Das neue To-Do wird sofort in der entsprechenden Kategorie (z.B. "Zu tun") angezeigt. 4. Die Benutzeroberfläche ist intuitiv und ermöglicht eine einfache Eingabe der To-Do-Informationen.

Tabelle 2: Erste User Story – To-Do Erstellen

(Quelle: Eigene Darstellung)

Titel	To-Do Bearbeiten
Inhalt	Als ein Benutzer der To-Do-Listen-Anwendung, möchte ich die Möglichkeit haben, bestehende To-Dos zu bearbeiten, damit ich Änderungen an meinen Aufgaben vornehmen kann, wie das Aktualisieren von Details oder das Anpassen von Fälligkeitsterminen.
Akzeptanzkriterien	<ol style="list-style-type: none"> 1. Der Benutzer kann ein bestehendes To-Do auswählen und in den Bearbeitungsmodus wechseln. 2. Der Benutzer kann Änderungen am Titel, dem Status und dem Fälligkeitsdatum des To-Dos vornehmen. 3. Die Änderungen können gespeichert und die aktualisierten Informationen werden sofort angezeigt. 4. Die Benutzeroberfläche für das Bearbeiten ist klar strukturiert und benutzerfreundlich.

Tabelle 3: Zweite User Story – To-Do Bearbeiten

(Quelle: Eigene Darstellung)

Titel	To-Do Löschen
Inhalt	Als ein Benutzer der To-Do-Listen-Anwendung, möchte ich die Möglichkeit haben, ein To-Do aus meiner Liste zu entfernen, damit ich Aufgaben, die erledigt sind oder nicht mehr relevant sind, aus meiner Übersicht löschen kann.
Akzeptanzkriterien	1. Der Benutzer kann ein bestimmtes To-Do auswählen und eine Option zum Löschen aufrufen. 2. Nach Bestätigung wird das To-Do dauerhaft aus der Liste entfernt. 3. Die Benutzeroberfläche bietet eine einfache und klare Möglichkeit, To-Dos zu löschen.

Tabelle 4: Dritte User Story – To-Do Löschen

(Quelle: Eigene Darstellung)

Titel	To-Do Anzeigen
Inhalt	Als ein Benutzer der To-Do-Listen-Anwendung, möchte ich eine übersichtliche Darstellung aller meiner To-Dos sehen, damit ich schnell einen Überblick über meine anstehenden und aktuellen Aufgaben gewinnen kann.
Akzeptanzkriterien	1. Der Benutzer kann alle erstellten To-Dos in einer übersichtlichen Liste oder Ansicht sehen. 2. To-Dos sind gegebenenfalls nach Kategorien oder Fälligkeitsdaten sortiert, um die Organisation zu erleichtern. 3. Wichtige Informationen wie Titel, Fälligkeitsdatum und Status sind auf einen Blick erkennbar. 4. Die Benutzeroberfläche ist klar und intuitiv gestaltet, um eine einfache Navigation und Ansicht zu ermöglichen.

Tabelle 5: Vierte User Story – To-Do Anzeigen

(Quelle: Eigene Darstellung)

Titel	Kategorisierung nach Status
Inhalt	Als ein Benutzer der To-Do-Listen-Anwendung, möchte ich die Möglichkeit haben, meine To-Dos nach ihrem Bearbeitungsstatus zu kategorisieren, damit ich meine Aufgaben effizient verwalten und ihren Fortschritt leicht verfolgen kann.

Akzeptanzkriterien	<ol style="list-style-type: none"> 1. Der Benutzer kann jedem To-Do einen Status wie "Zu tun", "In Bearbeitung" oder "Erledigt" zuweisen. 2. To-Dos können in der Benutzeroberfläche leicht zwischen diesen Statuskategorien verschoben werden, beispielsweise per Drag-and-Drop. 3. Die Kategorien sind klar in der Benutzeroberfläche abgegrenzt und leicht zu unterscheiden. 4. Änderungen des Status werden sofort in der Anwendung reflektiert, um eine aktuelle Übersicht zu gewährleisten.
--------------------	---

Tabelle 6: Fünfte User Story – Kategorisierung nach Status

(Quelle: Eigene Darstellung)

Titel	Priorisierung von Aufgaben
Inhalt	<p>Als ein Benutzer der To-Do-Listen-Anwendung, möchte ich die Möglichkeit haben, meine To-Dos nach ihrer Wichtigkeit zu priorisieren, damit ich sicherstellen kann, dass ich mich auf die wichtigsten Aufgaben konzentriere und diese rechtzeitig erledige.</p>
Akzeptanzkriterien	<ol style="list-style-type: none"> 1. Der Benutzer kann jedem To-Do eine Prioritätsstufe zuweisen, beispielsweise hoch, mittel oder niedrig. 2. To-Dos mit höherer Priorität werden in der Liste oder Ansicht prominent dargestellt. 3. Die Priorität eines To-Dos kann jederzeit geändert werden. 4. Die Benutzeroberfläche ermöglicht eine einfache und intuitive Einstellung der Prioritäten für die Aufgaben.

Tabelle 7: Sechste User Story – Priorisierung von Aufgaben

(Quelle: Eigene Darstellung)

Titel	Fälligkeitsdatum festlegen
Inhalt	<p>Als ein Benutzer der To-Do-Listen-Anwendung, möchte ich die Möglichkeit haben, für jedes To-Do ein Fälligkeitsdatum festzulegen, damit ich meine Aufgaben fristgerecht planen und erledigen kann und wichtige Termine nicht verpasse.</p>
Akzeptanzkriterien	<ol style="list-style-type: none"> 1. Der Benutzer kann bei der Erstellung oder Bearbeitung eines To-Dos ein Fälligkeitsdatum hinzufügen oder ändern.

	<p>2. Das Fälligkeitsdatum wird deutlich in der Ansicht jedes To-Dos angezeigt.</p> <p>3. To-Dos, deren Fälligkeitsdatum nahe liegt oder überschritten wurde, werden in der Benutzeroberfläche hervorgehoben.</p> <p>4. Die Benutzeroberfläche bietet eine benutzerfreundliche und intuitive Möglichkeit, Fälligkeitsdaten festzulegen und zu ändern.</p>
--	---

Tabelle 8: Siebte User Story – Fälligkeitsdatum festlegen

(Quelle: Eigene Darstellung)

2.4 Nicht-funktionale Anforderungen

Nachfolgend werden die Qualitätsattribute des Systems definiert. Im Gegensatz zu funktionalen Anforderungen, die beschreiben, was ein System tun soll, legen nicht-funktionale Anforderung fest, wie ein System seine Funktionen ausführen soll (Scand, 2021).

Die Anwendung sollte im Sinne der **Zuverlässigkeit und Leistung** stabil laufen und eine hohe Verfügbarkeit aufweisen. Fehler und Abstürze sollten minimal sein. Weiterhin sollte die Anwendung schnell auf Benutzereingaben reagieren. Zum Beispiel sollte das Hinzufügen, Bearbeiten und Löschen eines To-Dos nahezu in Echtzeit erfolgen, ohne spürbare Verzögerungen für den Benutzer.

Die Anwendung sollte im Hinblick auf die **Kompatibilität** mit verschiedenen Versionen des Windows-Betriebssystems (10/11) nutzbar sein. Weiterhin sollte der Code klar strukturiert und dokumentiert sein, um eine einfache **Wartbarkeit** sowie zukünftige Erweiterungen zu ermöglichen.

Die Anwendung sollte eine hohe **Sicherheit** aufweisen, d.h. sie sollte Daten sicher verarbeiten und speichern können. Dies beinhaltet den Schutz vor unbefugtem Zugriff, aber auch den Datenverlust.

Die Anwendung sollte im Sinne von **Benutzerfreundlichkeit, Design und Ästhetik** intuitiv und leicht zu bedienen sein. Dies umfasst eine klare und verständliche Benutzeroberfläche, einfache Navigation und eine logische Anordnung der Funktionen. Das Design sollte konsistent sein und die Ästhetik der Anwendung auf hohem Niveau halten, entsprechend den Anforderungen der Zielgruppe.

Diese nicht-funktionalen Anforderungen sollen insgesamt dazu beitragen, die Qualität der Anwendung zu verbessern und die Benutzererfahrung zu steigern.

2.5 Glossar

MVC	-	Model View Controller
UML	-	Unified Modeling Language

3. Projektdokumentation

3.1 Vorgehensmodell für die Entwicklungsphase

Als Informatikstudent mit begrenzter Programmiererfahrung fiel die Wahl für das vorliegende Projekt auf ein agiles Vorgehensmodell, bei dem Flexibilität, kontinuierliches Lernen und Anpassungsfähigkeit im Vordergrund stehen.

Konkret wurde sich für Scrum als agiles Rahmenwerk entschieden, das sich auf die iterative Entwicklung von Software konzentriert. Es basiert auf Sprints, was kurze, festgelegte Zeiträume sind (üblicherweise zwei bis vier Wochen), in denen zu Beginn definierte Aufgaben erledigt werden sollen. Die Arbeit in iterativen Zyklen mit regelmäßigem Feedback ermöglicht es, in überschaubaren Schritten voranzukommen, sich kontinuierlich zu verbessern und gleichzeitig den Überblick über den Gesamtfortschritt zu behalten (Scrum.org, 2024).

Auch wenn Scrum typischerweise in Teamumgebungen verwendet wird, soll es hier in Einzelarbeit angewendet werden. Dafür wird wie folgt vorgegangen:

- **Sprint Planning:** Am Anfang eines Sprints wird festgelegt, welche Aufgaben erledigt werden sollen. Hier dienen die Meilensteine als Hilfe.
- **Daily Scrum:** Der tägliche kollegiale Austausch soll durch eine tägliche Selbstreflexion zum eigenen Fortschritt und den aufgetretenen Herausforderungen ersetzt werden.
- **Sprint Review:** Am Ende eines Sprints soll geprüft werden, was erreicht wurde, was bei Bedarf in einer Anpassung des Plans mündet.
- **Sprint Retrospective:** Am Ende eines Sprints soll reflektiert werden, wie der letzte Sprint lief und wie die eigene Arbeitsweise verbessert werden kann.

Auf diese Weise soll eine Fokussierung auf die Zielsetzung erfolgen, regelmäßiger Fortschritt gemessen und Anpassungspotentiale gehoben werden.

3.2 Tech-Stack

3.2.1 Programmiersprache

Für die Entwicklung der Anwendung wurde sich für die Programmiersprache Python entschieden. Die Gründe dafür sind vielseitig und werden nachfolgend erläutert.

Python als Programmiersprache ist für eine vergleichsweise einfache Syntax und schnelle Entwicklungszeit bekannt. Zudem konnten bereits in einem vorherigen Studienkurs erste

Erfahrungen mit Python gesammelt werden. Darüber hinaus verfügt Python über leistungsstarke Frameworks, die für die Entwicklung von Desktop-Anwendungen geeignet sind. Zu guter Letzt bietet Python eine große Auswahl an Bibliotheken für UI-Design, was durch den gewählten Schwerpunkt auf einem ästhetisch ansprechenden Interface hilfreich erscheint.

Daneben wurden die anderen möglichen Programmiersprachen im direkten Vergleich als weniger vorteilhaft bewertet (BMU Verlag, 2024). C++ ist komplexer und weniger fokussiert auf schnelle UI-Entwicklung. C# ist stark in Microsoft-Technologien integriert, was für Windows-Anwendungen gut ist, aber möglicherweise weniger Flexibilität im Design bietet. Java ist eine gute Wahl für plattformübergreifende Anwendungen, wird aber bei der UI-Gestaltung als weniger intuitiv bewertet als Python.

3.2.2 Framework und Bibliotheken

Nach einer ersten Recherche zu einem nützlichen Framework und hilfreichen Bibliotheken für die geplante Anwendung, wurde sich für PySide (PyPi.org, 2024) als Framework entschieden. Es soll umfangreiche Möglichkeiten für das Design moderner Benutzeroberflächen bieten, welche zudem gut zu dokumentieren sind. Ergänzt werden soll es um folgende Bibliotheken:

- a) SQLAlchemy: Eine Bibliothek für Python, die als ORM (Object-Relational Mapping) Tool dient. Es ermöglicht die Interaktion mit Datenbanken durch Python-Objekte statt direkten SQL-Abfragen und unterstützt verschiedene Datenbank-Systeme (SQLAlchemy.org, 2024).
- b) Arrow: Eine Bibliothek zur Handhabung von Datums- und Zeitfunktionen in Python. Sie bietet Funktionen für das Parsen, Formatieren, Manipulieren und Konvertieren von Datum- und Zeitangaben und ist benutzerfreundlicher als das eingebaute „datetime“-Modul (Arrow, 2024).
- c) Pytest: Ein Framework für das Schreiben und Ausführen von Tests in Python. Es ist einfach zu verwenden, unterstützt fortgeschrittene Testfälle und kann zur Entwicklung automatisierter Tests genutzt werden (Krekel, 2015).
- d) PyInstaller: Ein Tool, das Python-Anwendungen in eigenständige ausführbare Dateien umwandelt, sodass sie ohne eine vorinstallierte Python-Umgebung auf dem Zielsystem ausgeführt werden können (PyInstaller, 2024).

Da im Bereich der Softwareentwicklung keine umfangreichen Erfahrungen vorhanden sind, wird die Wahl geeigneter Bibliotheken als ein iterativer Prozess verstanden, innerhalb dessen je nach Bedarf weitere Bibliotheken ergänzt oder entfernt werden können.

3.3 Entwurfsmuster (Design Pattern)

Im vorliegenden Projekt wurde sich für das Entwurfsmuster "Model-View-Controller (MVC)" entschieden (BMU Verlag, 2020). Hierzu wird die Anwendung in drei Hauptkomponenten unterteilt:

- a) **Model:** Enthält die Daten und die Logik der Anwendung, d.h. die Datenstrukturen für die To-Do-Elemente und die Logik für deren Verwaltung.
- b) **View:** Ist für die Darstellung der Informationen zuständig, d.h. für die grafische Benutzeroberfläche, die die To-Dos in einem ästhetischen, minimalistischen Stil präsentiert.
- c) **Controller:** Vermittelt zwischen Model und View, d.h. er reagiert auf Eingaben des Benutzers, manipuliert die Daten und aktualisiert die Ansicht.

Ziel ist es auf diese Weise die Anwendung modularer zu halten und dadurch eine leichtere Wartbarkeit und potenzielle Erweiterung zu ermöglichen. Es soll auch dabei helfen, die Benutzeroberfläche von der Datenlogik zu trennen, was besonders wichtig ist, um die Ästhetik und Benutzerfreundlichkeit der Anwendung besser gewährleisten zu können. Weitere Entwurfsmuster wie Singleton, Observer und Factory wurden betrachtet (MVPS.net, 2019), jedoch aufgrund ihrer Ausrichtung MVC nicht vorgezogen.

3.4 Struktur der Anwendung

Aufbau und Struktur der Anwendung wird nachfolgend in Abbildung 2 mit einem UML-Klassendiagramm dargestellt. Darin abgebildet werden die Klassen, Attribute und Methoden sowie deren Beziehung und Abhängigkeit zueinander. Der zuvor getroffenen Entscheidung MVC als Design Pattern zu nutzen, wird insofern Rechnung getragen, als dass die einzelnen Klassen farblich bereits den entsprechenden Komponenten zugeordnet sind. Model-Klassen werden blau, View-Klassen orange und Controller-Klassen grün dargestellt.

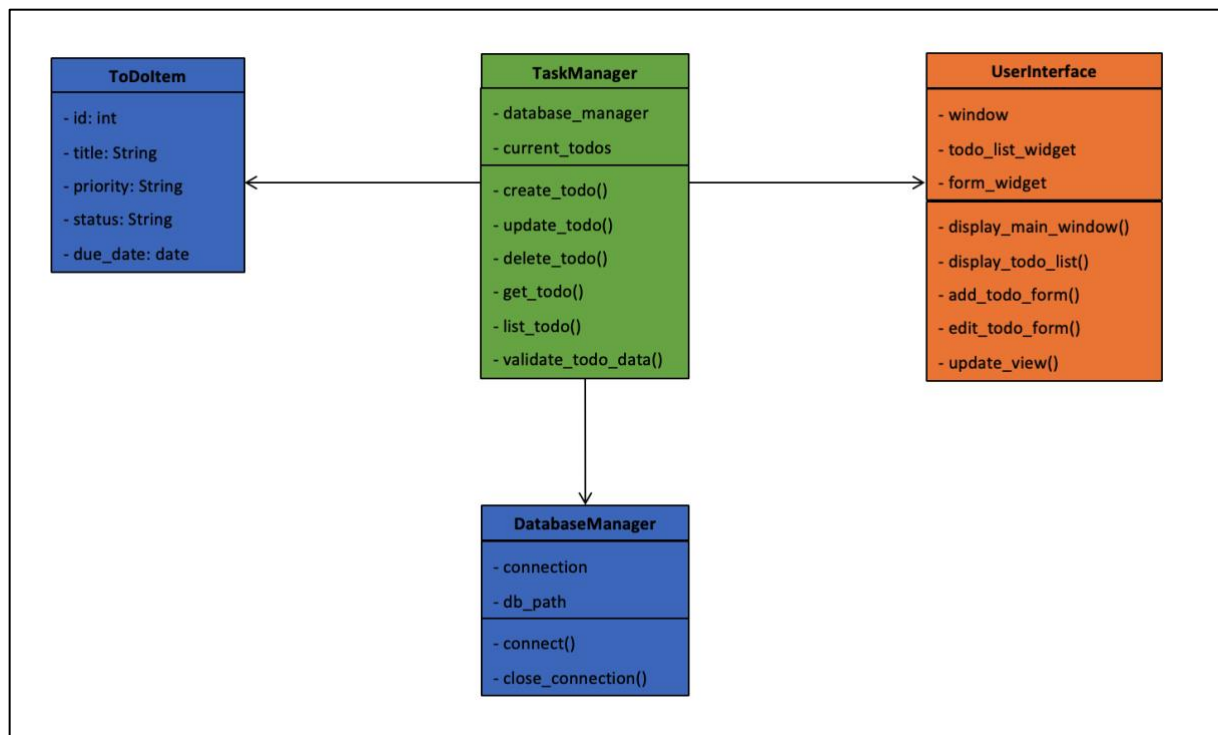


Abbildung 3: UML-Klassendiagramm

(Quelle: Eigene Darstellung)

Wie aus der Übersicht bereits ersichtlich werden nicht alle Funktionen der jeweiligen Klassen dargestellt. Hintergrund dessen ist, dass eine zu detaillierte Ansicht aller Methoden das UML-Diagramm überladen würde und dem Verständnis nicht zuträglich wäre. Gleiches gilt für eine detaillierte Darstellung aller Klassen, Attribute und Methoden. Wer eine Detailansicht aller Funktionen wünscht, wird auf das GitHub Repository der Anwendung verwiesen. Dort ist eine umfangreiche Dokumentation des gesamten Codes hinterlegt. In den nachfolgenden Tabellen 9 bis 12 werden hingegen nur die Klassen, Attribute und Methoden aus obigem UML-Diagramm nähergehend erklärt.

ToDoItem		
Eigenschaft	Bezeichnung	Beschreibung
Attribut	id	Eindeutige Kennung für jedes To-Do-Item.
Attribut	title	Der Titel des To-Dos.
Attribut	priority	Die Priorität des To-Dos, z.B. hoch, mittel, niedrig.
Attribute	status	Der aktuelle Status des To-Dos, z.B. "Zu tun", "In Bearbeitung", "Erledigt".

Attribute	due_date	Das Fälligkeitsdatum des To-Dos.
-----------	----------	----------------------------------

Tabelle 9: Attribute und Methoden von „ToDoItem“

(Quelle: Eigene Darstellung)

DatabaseManager		
Eigenschaft	Bezeichnung	Beschreibung
Attribute	connection	Eine Instanzvariable, die die Verbindung zur Datenbank hält.
Attribute	db_path	Der Pfad oder die Konfiguration der Datenbank, mit der der Manager interagiert.
Methode	connect()	Stellt eine Verbindung zur Datenbank her. Diese Methode ist wichtig, um Operationen auf der Datenbank durchzuführen.
Methode	close_connection()	Schließt die Verbindung zur Datenbank, wenn sie nicht mehr benötigt wird.

Tabelle 10: Attribute und Methoden von „DatabaseManager“

(Quelle: Eigene Darstellung)

TaskManager		
Eigenschaft	Bezeichnung	Beschreibung
Attribute	db_manager	Eine Instanz des DatabaseManager, um Datenbankoperationen durchzuführen.
Attribute	current_todos	Eine Liste oder ein anderes Datenstruktur-Element, das die aktuell geladenen To-Dos hält (optional, abhängig von der Anwendungslogik).
Methode	create_todo():	Nimmt Daten für ein neues To-Do entgegen und leitet diese an DatabaseManager zur Speicherung weiter.
Methode	close_connection()	Schließt die Verbindung zur Datenbank, wenn sie nicht mehr benötigt wird.
Methode	update_todo()	Aktualisiert die Daten eines bestehenden To-Dos in der Datenbank.
Methode	delete_todo()	Löscht ein To-Do aus der Datenbank.
Methode	get_todo()	Ruft die Details eines spezifischen To-Dos ab.
Methode	list_todos()	Gibt eine Liste aller To-Dos zurück, eventuell mit Filter- und Sortierfunktionen.

Methode	validate_todo_data()	Validiert die Daten eines To-Dos vor der Speicherung oder Aktualisierung.
---------	----------------------	---

Tabelle 11: Attribute und Methoden von „TaskManager“

(Quelle: Eigene Darstellung)

UserInterface		
Eigenschaft	Bezeichnung	Beschreibung
Attribute	window	Das Hauptfenster der GUI.
Attribute	todo_list_widget	Ein Widget, das die Liste der To-Dos darstellt.
Attribute	form_widget	Ein Widget für die Darstellung von Formularen zum Hinzufügen oder Bearbeiten von To-Dos.
Methode	display_main_window()	Zeigt das Hauptfenster der Anwendung an.
Methode	display_todo_list()	Stellt die Liste der To-Dos dar.
Methode	add_todo_form()	Zeigt ein Formular zum Hinzufügen eines neuen To-Dos.
Methode	edit_todo_form()	Zeigt ein Formular zum Bearbeiten eines bestehenden To-Dos.
Methode	update_view()	Aktualisiert die Ansicht, z.B. nachdem ein To-Do hinzugefügt, bearbeitet oder gelöscht wurde.

Tabelle 12: Attribute und Methoden von „UserInterface“

(Quelle: Eigene Darstellung)

3.5 Interaktion Nutzer und Anwendung

Nachdem die Struktur der Anwendung dargestellt wurde, soll die Interaktion der darin beschriebenen Bestandteile untereinander in Bezug auf die Interaktion mit dem Nutzer dargestellt werden. Dazu soll nachfolgend auf ein UML-Aktivitätsdiagramm zurückgegriffen werden. Grundsätzlich sind diese als ein grafisches Modellierungswerkzeug zu verstehen, das verwendet wird, um den Ablauf von Aktivitäten oder Prozessen in einem System abzubilden. Sie zeigen somit eine visuelle Darstellung der Reihenfolge von Aktionen, Entscheidungen und Verzweigungen innerhalb eines Systems auf (Ionos, 2023). Der Fokus kann hierbei bspw. auf der Benutzerinteraktion, den Systemprozessen, Zustandsübergängen oder der Benutzerführung liegen. Aufgrund des im Projekt gewählten Schwerpunktes auf der Benutzerfreundlichkeit der Anwendung, wird an dieser Stelle ein besonderes Augenmerk auf die Benutzerinteraktion gelegt. Das heißt insbesondere die Kernprozesse (Erstellen, Bearbeiten, Löschen und Anzeigen / Sortieren von Aufgaben) sind hier von Interesse. Aufgrund

des Modulumfangs wird sich hier auf ein UML-Aktivitätsdiagramm beschränkt. Im Folgenden soll daher die Änderung des Fälligkeitsdatums grafisch dargestellt werden.

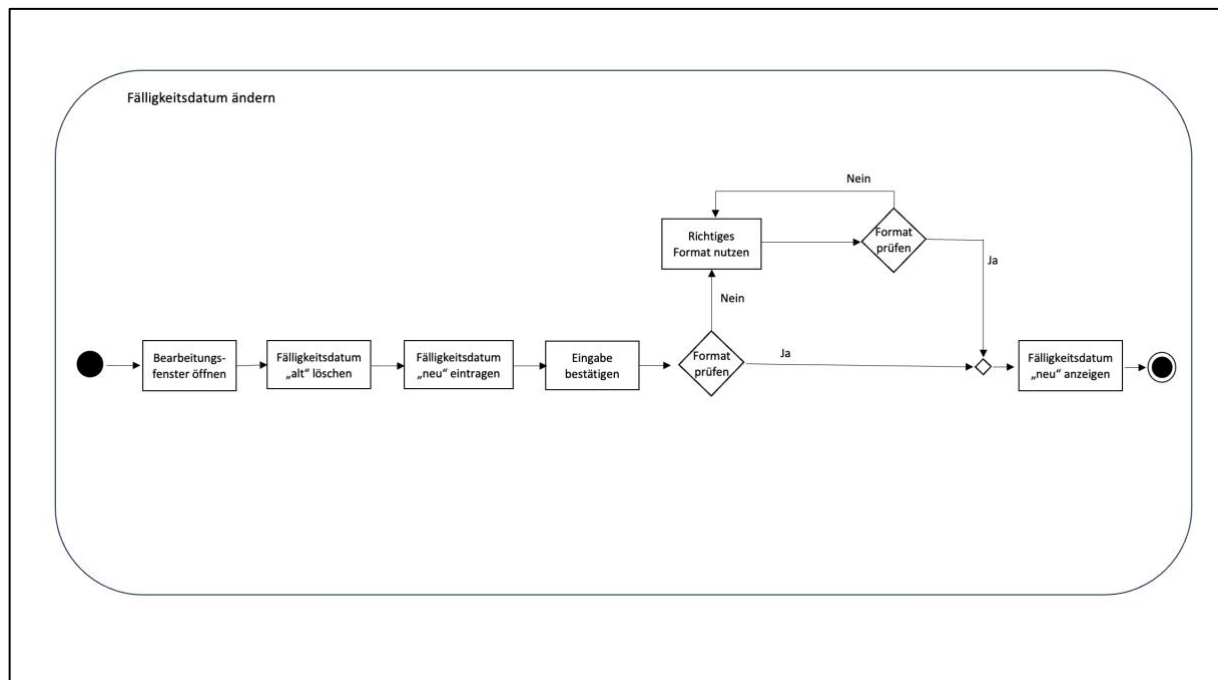


Abbildung 4: UML-Aktivitätsdiagramm

(Quelle: Eigene Darstellung)

3.6 Revisionskontrolle und Nutzung von GitHub

Für die Verwaltung und Nachverfolgung von Änderungen am Quellcode wird Git in Verbindung mit der Plattform GitHub verwendet. GitHub bietet eine benutzerfreundliche Oberfläche für das Hosting von Git-Repositories sowie Tools zur Zusammenarbeit und Versionsverwaltung. Durch die Nutzung von GitHub wird eine transparente und effiziente Entwicklung ermöglicht, bei der Änderungen protokolliert und Rückverfolgbarkeit gewährleistet werden. Außerdem erleichtert GitHub die Zusammenarbeit zwischen den Entwicklern und ermöglicht es, Probleme und Verbesserungsvorschläge systematisch zu verfolgen und zu adressieren. Auch wenn es hierzu mehrere Anbieter gibt, war es eine Auflage Github als zu präferierendes System zu wählen.

Das eigene Projekt kann über nachfolgenden Link jederzeit abgerufen und eingesehen werden:
https://github.com/meetSeb/ToDo_Application.git

3.7 Benutzerhandbuch

Dieses Handbuch bietet Ihnen eine umfassende Anleitung zur Installation, Einrichtung und Nutzung der Anwendung auf Ihrem Windows-System. Egal, ob Sie Ihre Aufgaben effizient organisieren oder Ihr Produktivitätsniveau steigern möchten, dieses Handbuch bietet Ihnen alle Informationen, die Sie benötigen, um das Beste aus unserer Anwendung herauszuholen.

1. Installation und Einrichtung:

Um die Anwendung auf Ihrem Windows-System zu installieren und einzurichten, folgen Sie bitte den nachstehenden Schritten:

- a) Herunterladen der Installationsdatei: Laden Sie die Installationsdatei der Anwendung über [Github](https://github.com/meetSeb/ToDo_Application/releases/tag/v1.0.1) herunter (https://github.com/meetSeb/ToDo_Application/releases/tag/v1.0.1). Die Datei liegt als ZIP-Archiv oder ausführbare EXE-Datei vorliegen. Letzteres (EXE-Datei) ist für die Installation zu bevorzugen.
- b) Extrahieren der ZIP-Datei (falls erforderlich): Wenn die Datei als ZIP-Archiv vorliegt, extrahieren Sie den Inhalt in einen Ordner auf Ihrem Computer.
- c) Ausführen der Installationsdatei: Doppelklicken Sie auf die heruntergeladene EXE-Datei (main.exe), um den Installationsprozess zu starten. Folgen Sie den Anweisungen im Installationsassistenten, um die Anwendung auf Ihrem System zu installieren.
- d) Abschluss der Installation: Nach Abschluss der Installation können Sie die Anwendung über das Startmenü oder eine Verknüpfung auf Ihrem Desktop öffnen.

Nach Abschluss dieser Schritte ist die Anwendung erfolgreich auf Ihrem Windows-System installiert und einsatzbereit.

2. Anmeldung und Benutzerkonto:

Diese Anwendung bietet eine nahtlose Benutzererfahrung, indem sie sich einfach öffnen lässt, sobald Sie sich am Computer mit ihrem Windows-Benutzernamen anmelden. Dies bedeutet, dass Sie sich nicht zusätzlich zur Anwendung anmelden müssen. Die damit einhergehenden Vorteile sind insbesondere eine bequeme Nutzung und Zeitersparnis, welche für die tagtägliche Nutzung einer To Do-Applikation von extremer Bedeutung sind.

3. Navigationsanleitung:

Die Benutzeroberfläche dieser Anwendung ist übersichtlich und intuitiv gestaltet, um Ihnen eine einfache Navigation bzw. Handhabung zu ermöglichen. Hier finden Sie eine kurze Übersicht über die wichtigsten Elemente der Benutzeroberfläche und deren Verwendung:

- a) **Hauptansicht:** Beim Starten der Anwendung gelangen Sie zur Hauptansicht, die eine Übersicht über Ihre aktuellen Aufgaben und Listen bietet. Hier können Sie neue Aufgaben erstellen, bestehende Aufgaben bearbeiten oder löschen. Die Kanban-Ansicht bietet dazu eine visuelle Darstellung Ihrer Aufgaben in Spalten, die den verschiedenen Status oder Kategorien entsprechen. Sie können die Aufgaben zwischen den Spalten hin und her ziehen, um ihren Bearbeitungsstatus zu aktualisieren. Neue Aufgaben können über das Eingabefeld am Kopf der Anwendung eingegeben werden.

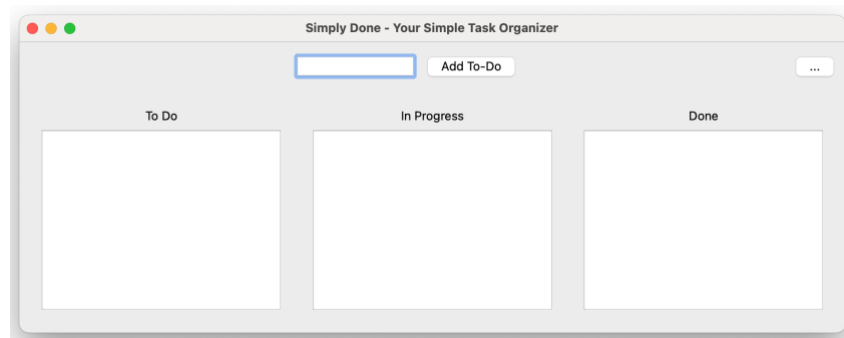


Abbildung 5: Benutzeroberfläche

(Quelle: Eigene Darstellung)

- b) **Aufgabenliste:** In der Aufgabenliste werden Ihre Aufgaben in Form einer Liste dargestellt, die immer den jeweiligen Titel des To-Dos anzeigt. Sie können die Aufgaben nach verschiedenen Kriterien sortieren, um Ihre Arbeit zu organisieren.
- c) **Navigationselemente:** Am rechten seitlichen Rand der Benutzeroberfläche befinden sich ein Navigationselemente mit dem Sie zwischen verschiedenen Sortier-Modi (Priorität und Fälligkeitsdatum) wechseln können.

4. Aufgabenverwaltung:

Diese Anwendung bietet eine Vielzahl von Funktionen zur effektiven Verwaltung Ihrer Aufgaben. Hier finden Sie eine detaillierte Anleitung zur Verwendung der wichtigsten Funktionen:

- a) Erstellen von Aufgaben: Um eine neue Aufgabe zu erstellen, klicken Sie in das Eingabefeld, geben ihrer Aufgabe einen Titel und drücken anschließend auf den „Submit“ Button oder drücken Sie „ENTER“. Im Anschluss erscheint Ihre Aufgabe im „To Do“ Feld der Anwendung.

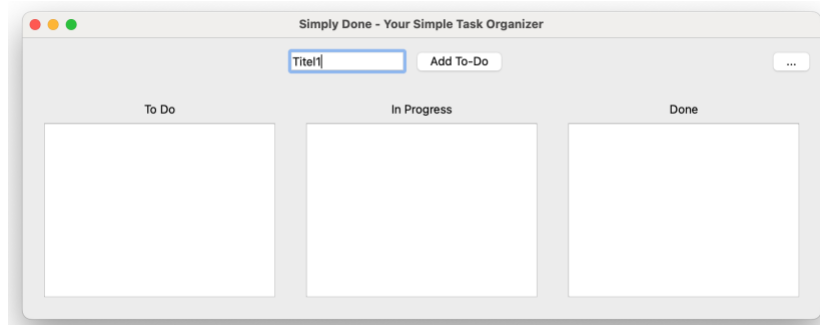


Abbildung 6: Benutzereingabe

(Quelle: Eigene Darstellung)

- b) Bearbeiten von Aufgaben: Um eine vorhandene Aufgabe zu bearbeiten, klicken Sie mit einem Doppelklick der linken Maustaste auf die entsprechende Aufgabe in Ihrer Liste. Es öffnet sich ein Feld, indem Sie den Titel, die Priorität, den Status und das Fälligkeitsdatum ändern können. Der Titel ist frei wählbar. Bei Status und Priorität wird ein Dropdown-Menü mit Auswahlmöglichkeiten geöffnet. Beim eintragen des Fälligkeitsdatums ist auf das richtige Format (YYYY/MM/DD) zu achten. Sollte nicht das richtige Format gewählt worden sein, wird eine Fehlermeldung mit einem entsprechenden Hinweis angezeigt. Es müssen nicht alle Felder befüllt werden, sondern es können auch lediglich einzelne Bereiche befüllt werden. Nur der Titel der Aufgabe muss ausgefüllt sein. Abgeschlossen wird die Bearbeitung mit einfachem Klick auf den „Submit“ Button. Wollen Sie die Bearbeitung abbrechen, können Sie dies über den regulären Schließbutton des Fensters tun.

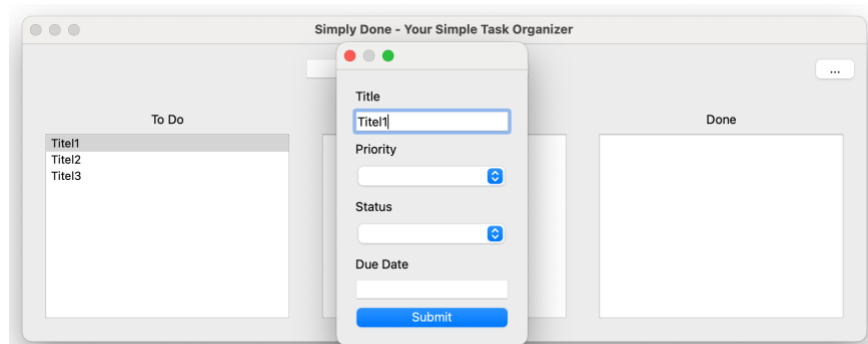


Abbildung 7: Ändern ToDo Eigenschaften

(Quelle: Eigene Darstellung)

- c) Löschen von Aufgaben: Um eine Aufgabe zu löschen, führen Sie einen Rechtsklick auf die entsprechende Aufgabe aus. Im Anschluss öffnet sich ein Fenster, bei dem Sie bestätigen müssen, dass Sie die Aufgabe wirklich löschen wollen. Bestätigen Sie die Löschung, wenn Sie die Aufgabe endgültig löschen wollen.

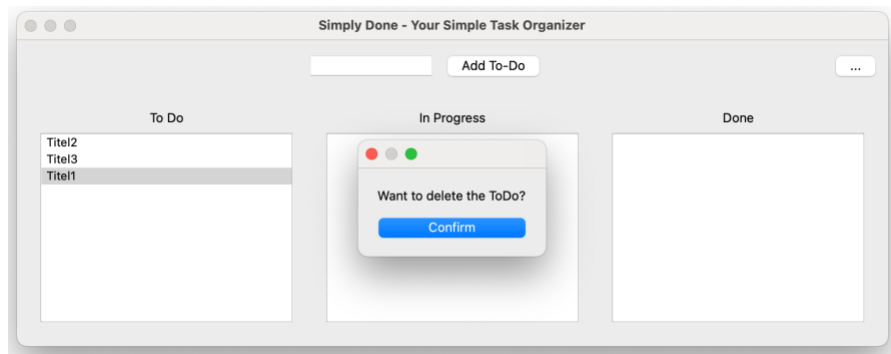


Abbildung 8: ToDo Item löschen

(Quelle: Eigene Darstellung)

- d) Sortieren von Aufgaben: Um Aufgaben zu sortieren, wählen Sie die gewünschte Sortieroption über den Button mit den drei Punkten am rechten Rand der Anwendung aus. Im Anschluss öffnet sich ein Fenster bei dem Sie zwischen einer Sortierung nach Fälligkeitsdatum und Priorität auswählen können.

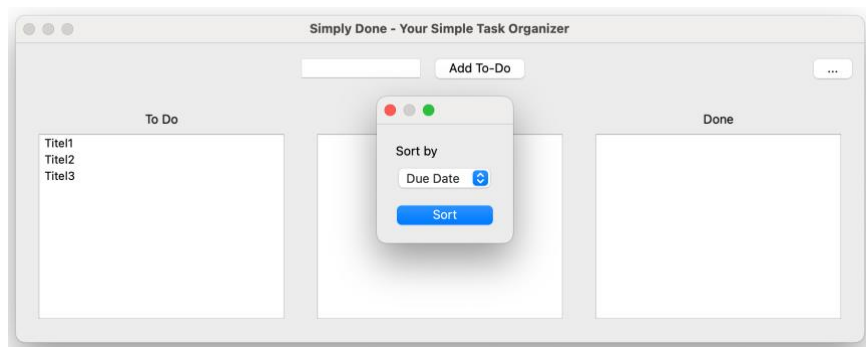


Abbildung 9: Sortierfunktion

(Quelle: Eigene Darstellung)

Alle Funktionen der Tätigkeitsverwaltung sind darauf ausgelegt den Workflow des Nutzers möglichst intuitiv und einfach zu halten.

5. Hilfe und Support:

Es wurden verschiedene Maßnahmen zur Unterstützung ergriffen. Dazu zählen eine ausführliche Dokumentation des Codes, die aktive Community-Unterstützung sowie eine Liste häufig gestellter Fragen (FAQs), die fortlaufend aktualisiert werden soll. Die Dokumentation steht online zur Verfügung und umfasst die Anleitungen zur Installation, zur Verwendung der Anwendung und zur Fehlerbehebung. Darüber hinaus ist es jederzeit möglich unserer Entwicklercommunity auf GitHub beizutreten, um Fragen zu stellen, Probleme zu melden und mit anderen Benutzern und Entwicklern in Kontakt zu treten.

6. Datenschutz und Sicherheit:

Diese Anwendung ist als eigenständige Anwendung konzipiert und hat keine direkte Verbindung zum Internet. Alle Benutzerdaten werden lokal auf dem Computer des Benutzers gespeichert und sind nicht für Dritte zugänglich. Weiterhin wurde die Anwendung gründlich getestet, um sicherzustellen, dass sie den höchsten Sicherheitsstandards entspricht. Dies umfasst sowohl manuelle als auch automatisierte Tests, um potenzielle Schwachstellen zu identifizieren und zu beheben. Regelmäßig Updates, die Fehlerkorrekturen, Leistungsverbesserungen und Sicherheitsupdates enthalten können, werden über GitHub veröffentlicht. Als Open-Source Projekt besteht darüber hinaus die Möglichkeit für die gesamte GitHub Community auf Schwachstellen hinzuweisen, sodass sie schnell behoben werden können.

7. Updates und Wartung:

Da diese Anwendung als Open-Source-Projekt konzipiert ist, haben Benutzer die Möglichkeit, den Quellcode über die Plattform GitHub herunterzuladen und zu verwenden. Updates und die Wartung der Anwendung liegen in der Verantwortung der Benutzer. Es wird trotzdem empfohlen, regelmäßig nach Aktualisierungen im Repository zu suchen und die Anwendung, falls vorhanden, zu aktualisieren, um von neuen Funktionen, Fehlerkorrekturen und Verbesserungen zu profitieren. Für Fragen oder Unterstützung bei der Wartung der Anwendung steht nicht nur der Entwickler, sondern die gesamte Entwicklercommunity auf GitHub zur Verfügung.

8. FAQs:

Eine Liste häufig gestellter Fragen und deren Antworten wird nach der Beta-Phase zur Verfügung gestellt und fortlaufend aktualisiert.

4. Testprotokoll

Es gibt in der Softwareentwicklung unterschiedliche Testverfahren, die je nach Umfang und Komplexität der Anwendung angewendet werden (Pittet, 2024). Die drei bekanntesten Kategorien sind:

- Unittests: Diese Tests überprüfen die Funktionalität einzelner Methoden oder Funktionen.
- Integrationstests: Diese Tests überprüfen, wie verschiedene Teile der Anwendung zusammenarbeiten. Zum Beispiel könnten man einen Test schreiben, der überprüft, ob ein ToDo-Element korrekt in der Datenbank gespeichert wird, wenn eine Methode aus dem Controller dies anweist.
- Systemtests: Diese Tests überprüfen die Funktionalität der gesamten Anwendung. Man könnte bspw. einen Test schreiben, der die gesamte Benutzererfahrung beim Erstellen eines neuen ToDo-Elements simuliert.

Im vorliegenden Projekt wurde sich für eine Kombination aus Unittests und Systemtest entschieden. Während die Unittests automatisiert mit Pytest durchgeführt wurden, wurde der Systemtest analog durch den Entwickler ausgewertet. Unittests werden idealerweise bereits während der Entwicklung durchgeführt, da dadurch unmittelbares Feedback zur Funktion einzelner Module entsteht. Aus diesem Grund wurde für jede Funktion ein separater Test erstellt. Die Durchführung des abschließenden Systemtests wird als finaler Test gesehen, der viele weitere, separate Integrationstests ersetzt. Sollten Fehler auftreten, sind diese aufgrund der als moderat komplex zu bewertenden Anwendung vergleichsweise einfach zu identifizieren, weshalb auf zusätzliche Integrationstest verzichtet wurde. Gleichzeitig bietet ein Systemtest die Möglichkeit das Interface in Aktion zu sehen, was gerade im Hinblick auf die design- bzw. funktionssensible Zielgruppe wichtig ist.

4.1 Unittests

Die automatisierten Testprotokolle mit PyTest wurden als Html-Datei abgespeichert und sind einerseits über das GitHub Repository abrufbar, andererseits als Bestätigung der Durchführung im Anhang zu finden. Nachfolgende Tabelle gibt eine ganzheitliche Übersicht über alle durchgeführten Unittests sowie eine Beschreibung derer.

Klasse	Test_ID	Beschreibung	Vorbedingungen	Eingabedaten / Handlung des Benutzers	Erwartetes Ergebnis	Tatsächliches Ergebnis
ToDoItem (Model)	UT1	Dieser Testfall überprüft, ob die ToDoItem Klasse korrekt initialisiert wird. Insbesondere wird überprüft, ob die Attribute id, title, priority, status und due_date korrekt gesetzt werden.	Es gibt keine spezifischen Vorbedingungen für diesen Testfall, da er eine unabhängige Instanz der ToDoItem Klasse erstellt und keine externen Abhängigkeiten hat.	Die Eingabedaten für diesen Test sind die Werte, die beim Erstellen der ToDoItem Instanz verwendet werden: 1, 'Buy milk', 'High', 'To Do', '2022-01-01'.	Die Attribute der ToDoItem Instanz stimmen mit den Eingabedaten überein. Das heißt: todo_item.id = 1 todo_item.title = 'Buy milk' todo_item.priority = 'High' todo_item.status = 'To Do' todo_item.due_date = '2022-01-01'	todo_item.id = 1 todo_item.title = 'Buy milk' todo_item.priority = 'High' todo_item.status = 'To Do' todo_item.due_date = '2022-01-01'
Database Manager (Model)	UT2	Dieser Testfall überprüft, ob ein ToDoItem korrekt in die Datenbank eingefügt wird. Insbesondere wird überprüft, ob das	Vor der Ausführung des Tests muss eine Verbindung zur Datenbank hergestellt werden. Außerdem muss	Die Eingabedaten für diesen Test sind die Attribute des ToDoItem, das in die Datenbank eingefügt wird. Die Handlung des Benutzers besteht darin, die	Das ToDoItem wird korrekt in die Datenbank eingefügt und seine Attribute title, priority und status werden korrekt in der Datenbank gespeichert.	todo_item.id = 1 todo_item.title = 'Buy milk' todo_item.priority = 'High'

		ToDoItem die Attribute title, priority und status korrekt in der Datenbank speichert.	ein ToDoItem mit der ID 1, dem Titel 'Test title', der Priorität 'High', dem Status 'In Progress' und dem Fälligkeitsdatum '2022-01-01' erstellt werden.	insert_todo_item Methode aufzurufen, um das ToDoItem in die Datenbank einzufügen.	todo_item.id = 1 todo_item.title = 'Buy milk' todo_item.priority = 'High' todo_item.status = 'To Do' todo_item.due_date = '2022-01-01'	todo_item.status = 'To Do' todo_item.due_date = '2022-01-01'
Datase Manage r (Model)	UT3	Dieser Testfall überprüft, ob ein ToDoItem korrekt aus der Datenbank abgerufen wird. Insbesondere wird überprüft, ob das abgerufene ToDoItem die gleichen Attribute wie das ursprüngliche ToDoItem hat.	Vor der Ausführung des Tests muss eine Verbindung zur Datenbank hergestellt werden. Außerdem muss ein ToDoItem mit der ID 1 in die Datenbank eingefügt werden.	Die Eingabedaten für diesen Test sind die Attribute des ToDoItem, das in die Datenbank eingefügt wird. Die Handlung des Benutzers besteht darin, die get_todo_item Methode aufzurufen, um das ToDoItem aus der Datenbank abzurufen.	Das erwartete Ergebnis ist, dass das abgerufene ToDoItem die gleiche ID wie das ursprüngliche ToDoItem hat, also 1. todo_item.id = 1	todo_item.id = 1
Datase Manage r (Model)	UT4	Dieser Testfall überprüft, ob ein ToDoItem korrekt in der Datenbank aktualisiert wird. Insbesondere wird überprüft, ob die Attribute	Dieser Testfall überprüft, ob ein ToDoItem korrekt in der Datenbank aktualisiert wird. Insbesondere wird überprüft, ob die	Dieser Testfall überprüft, ob ein ToDoItem korrekt in der Datenbank aktualisiert wird. Insbesondere wird überprüft, ob die Attribute des ToDoItem nach dem Aufruf der	Das erwartete Ergebnis ist, dass die Attribute des ToDoItem in der Datenbank auf die neuen Werte aktualisiert werden. todo_item.id = 1	todo_item.id = 1 todo_item.title = 'Updated title' todo_item.priority = 'Low'

		des ToDoItem nach dem Aufruf der update_todo_item Methode korrekt in der Datenbank aktualisiert werden.	Attribute des ToDoItem nach dem Aufruf der update_todo_item Methode korrekt in der Datenbank aktualisiert werden.	update_todo_item Methode korrekt in der Datenbank aktualisiert werden.	todo_item.title = 'Updated title' todo_item.priority = 'Low' todo_item.status = 'Done' todo_item.due_date = '2022-02-02'	todo_item.status = 'Done' todo_item.due_date = '2022-02-02'
Datase Manager (Model)	UT5	Dieser Testfall überprüft, ob ein ToDoItem korrekt aus der Datenbank gelöscht wird. Insbesondere wird überprüft, ob das ToDoItem nach dem Löschen nicht mehr in der Datenbank gefunden werden kann.	Vor der Ausführung des Tests muss eine Verbindung zur Datenbank hergestellt werden. Außerdem muss ein ToDoItem mit der ID 1 in die Datenbank eingefügt werden.	Die Eingabedaten für diesen Test sind die Attribute des ToDoItem, das in die Datenbank eingefügt wird. Die Handlung des Benutzers besteht darin, die delete_todo_item Methode aufzurufen, um das ToDoItem aus der Datenbank zu löschen.	Das erwartete Ergebnis ist, dass das ToDoItem nicht mehr in der Datenbank gefunden werden kann, nachdem es gelöscht wurde. retrieved_item = None	retrieved_item = None
Datase Manager (Model)	UT6	Dieser Testfall überprüft, ob alle ToDoItems korrekt aus der Datenbank gelöscht werden. Insbesondere wird überprüft, ob die Liste der ToDoItems leer ist,	Vor der Ausführung des Tests muss eine Verbindung zur Datenbank hergestellt werden. Außerdem muss mindestens ein ToDoItem	Die Eingabedaten für diesen Test sind die Attribute des ToDoItem, das in die Datenbank eingefügt wird. Die Handlung des Benutzers besteht darin, die delete_all_todo_items Methode aufzurufen, um alle	Das erwartete Ergebnis ist, dass die Liste der ToDoItems leer ist, nachdem alle ToDoItems gelöscht wurden. len(todo_items) == 0	len(todo_items) == 0

		nachdem die delete_all_todo_items Methode aufgerufen wurde.	in die Datenbank eingefügt werden.	ToDoltems aus der Datenbank zu löschen.		
Datase Manager (Model)	UT7	Dieser Testfall überprüft, ob alle ToDoltems korrekt aus der Datenbank abgerufen werden. Insbesondere wird überprüft, ob die Liste der ToDoltems das ursprüngliche ToDoltem enthält und ob das abgerufene ToDoltem die gleichen Attribute wie das ursprüngliche ToDoltem hat.	Vor der Ausführung des Tests muss eine Verbindung zur Datenbank hergestellt werden. Außerdem muss ein ToDoltem in die Datenbank eingefügt werden.	Die Eingabedaten für diesen Test sind die Attribute des ToDoltem, das in die Datenbank eingefügt wird. Die Handlung des Benutzers besteht darin, die list_todo_items Methode aufzurufen, um alle ToDoltems aus der Datenbank abzurufen.	Das erwartete Ergebnis ist, dass die Liste der ToDoltems das ursprüngliche ToDoltem enthält und dass das abgerufene ToDoltem die gleichen Attribute wie das ursprüngliche ToDoltem hat. len(todo_items) == 1 todo_items[0].title == 'Test title'	len(todo_items) == 1 todo_items[0].title == 'Test title'
Datase Manager (Model)	UT8	Dieser Testfall überprüft, ob ToDoltems korrekt aus der Datenbank abgerufen und nach dem angegebenen Feld sortiert werden. Insbesondere	Vor der Ausführung des Tests muss eine Verbindung zur Datenbank hergestellt werden. Außerdem müssen mehrere	Die Eingabedaten für diesen Test sind die Attribute der ToDoltems, die in die Datenbank eingefügt werden. Die Handlung des Benutzers besteht darin, die	Das erwartete Ergebnis ist, dass die ToDoltems in aufsteigender Reihenfolge nach Priorität sortiert sind. Das erste ToDoltem in der sortierten Liste sollte eine	sorted_items[0].priority == 'High'

		wird überprüft, ob die ToDoltems in aufsteigender Reihenfolge nach Priorität sortiert sind.	ToDoltems mit unterschiedlichen Prioritäten in die Datenbank eingefügt werden.	get_todo_items_sorted_by Methode aufzurufen, um die ToDoltems aus der Datenbank abzurufen und nach Priorität zu sortieren.	Priorität von 'High' haben und das letzte ToDoltem sollte eine Priorität von 'Low' haben. sorted_items[0].priority == 'High'	
Database Manager (Model)	UT9	Dieser Testfall überprüft, ob der Status eines ToDoltem korrekt in der Datenbank aktualisiert wird. Insbesondere wird überprüft, ob der Status des ToDoltem auf 'Done' gesetzt wird.	Vor der Ausführung des Tests muss eine Verbindung zur Datenbank hergestellt werden. Außerdem muss ein ToDoltem in die Datenbank eingefügt werden.	Die Eingabedaten für diesen Test sind die Attribute des ToDoltem, das in die Datenbank eingefügt wird. Die Handlung des Benutzers besteht darin, die update_todo_status Methode aufzurufen, um den Status des ToDoltem in der Datenbank zu aktualisieren.	Das erwartete Ergebnis ist, dass der Status des ToDoltem auf 'Done' gesetzt wird. retrieved_item.status == 'Done'	retrieved_item.status == 'Done'
UserInterface (View)	UT10	Dieser Testfall überprüft, ob die add_todo Methode die create_todo_item Methode der TaskManager Klasse aufruft und das todo_input Feld leert.	Vor der Ausführung des Tests muss eine Instanz der UserInterface Klasse erstellt werden. Außerdem muss ein Mock-Objekt für die TaskManager Klasse und das todo_input Feld erstellt werden.	Die Eingabedaten für diesen Test sind der Text "Test ToDo Item", der in das todo_input Feld eingegeben wird. Die Handlung des Benutzers besteht darin, die add_todo Methode aufzurufen.	Das erwartete Ergebnis ist, dass die create_todo_item Methode der TaskManager Klasse aufgerufen wird und das todo_input Feld geleert wird. user_interface.todo_input.text.return_value = "Test ToDo Item"	user_interface.todo_input.text.return_value = "Test ToDo Item"

UserInterface (View)	UT11	Dieser Testfall überprüft, ob die create_labeled_widget Methode ein Widget mit einem QLabel und einem anderen Widget erstellt. Der QLabel sollte den Text 'Label' haben und das andere Widget sollte das Widget sein, das als Argument übergeben wird.	Vor der Ausführung des Tests muss eine Instanz der UserInterface Klasse erstellt werden. Außerdem muss ein QWidget erstellt werden.	Die Eingabedaten für diesen Test sind der Text 'Label' und das QWidget, das als Argumente an die create_labeled_widget Methode übergeben werden. Die Handlung des Benutzers besteht darin, die create_labeled_widget Methode aufzurufen.	Das erwartete Ergebnis ist, dass ein Widget erstellt wird, das einen QLabel mit dem Text 'Label' und das übergebene QWidget hat. result.layout().count() == 2 result.layout().itemAt(0).widget().text() == "Label" result.layout().itemAt(1).widget() == widget	result.layout().count() == 2 result.layout().itemAt(0).widget().text() == "Label" result.layout().itemAt(1).widget() == widget
UserInterface (View)	UT12	Dieser Testfall überprüft, ob die startDrag Methode der CustomListWidget Klasse die korrekten MIME-Daten setzt. Die MIME-Daten sollten die ID des QTableWidgetItem enthalten, das mit dem ausgewählten Element verknüpft ist.	Vor der Ausführung des Tests muss eine Instanz der UserInterface Klasse und der CustomListWidget Klasse erstellt werden. Außerdem muss ein QTableWidgetItem erstellt und zur CustomListWidget hinzugefügt werden.	Die Eingabedaten für diesen Test sind die QListWidgetItem und die ID 123, die als Daten an das QListWidgetItem übergeben werden. Die Handlung des Benutzers besteht darin, die startDrag Methode aufzurufen.	Das erwartete Ergebnis ist, dass die MIME-Daten die ID 123 enthalten. mime_data.text() == "123"	mime_data.text() == "123"

UserInterface (View)	UT13	Dieser Testfall überprüft, ob die dropEvent Methode der CustomListWidget Klasse die update_todo_status Methode des Controllers mit den korrekten Argumenten aufruft. Die Methode sollte die update_kanban_board Methode der UserInterface Klasse aufrufen und den Status des ToDoItem aktualisieren, das mit dem abgelegten Element verknüpft ist.	Vor der Ausführung des Tests muss eine Instanz der UserInterface Klasse und der CustomListWidget Klasse erstellt werden. Außerdem muss ein QListWidgetItem erstellt und zur CustomListWidget hinzugefügt werden.	Die Eingabedaten für diesen Test sind die QListWidgetItem und der Status "In Progress", die als Argumente an die CustomListWidget übergeben werden. Die Handlung des Benutzers besteht darin, die dropEvent Methode aufzurufen.	Das erwartete Ergebnis ist, dass die update_todo_status Methode des Controllers genau einmal aufgerufen wird und der Status des ToDoItem auf "In Progress" gesetzt wird. True	True
UserInterface (View)	UT14	Dieser Testfall überprüft, ob die mousePressEvent Methode der CustomListWidget Klasse das itemRightClicked Signal auslöst,	Vor der Ausführung des Tests muss eine Instanz der UserInterface Klasse und der CustomListWidget Klasse erstellt werden.	Die Eingabedaten für diesen Test sind das QListWidgetItem und ein QMouseEvent, das das Drücken der rechten Maustaste simuliert. Die Handlung des Benutzers besteht darin, die	Das erwartete Ergebnis ist, dass das Signal itemRightClicked einmal ausgelöst wird. True	True

		wenn die rechte Maustaste gedrückt wird. Die Methode sollte das Signal einmal auslösen.	Außerdem muss ein QListWidgetItem erstellt werden.	mousePressEvent Methode aufzurufen.		
TaskManager (Controller)	UT15	Dieser Testfall überprüft, ob die create_todo_item Methode des task_manager ein ToDoItem korrekt erstellt. Das erstellte ToDoItem sollte die folgenden Attribute haben: Titel, Priorität, Status und Fälligkeitsdatum.	Vor der Ausführung des Tests muss eine Instanz der TestTaskManager Klasse erstellt werden. Außerdem muss die delete_all_todo_items Methode des task_manager aufgerufen werden, um sicherzustellen, dass die Datenbank leer ist.	Die Eingabedaten für diesen Test sind der Titel 'Test title', die Priorität 'High', der Status 'In Progress' und das Fälligkeitsdatum '2022-01-01', die als Argumente an die create_todo_item Methode übergeben werden. Die Handlung des Benutzers besteht darin, die create_todo_item Methode aufzurufen.	Das erwartete Ergebnis ist, dass ein ToDoItem erstellt wird, das die angegebenen Attribute hat. todo_item.title == 'Test title' todo_item.priority == 'High' todo_item.status == 'In Progress' todo_item.due_date == '2022-01-01'	todo_item.title == 'Test title' todo_item.priority == 'High' todo_item.status == 'In Progress' todo_item.due_date == '2022-01-01'
TaskManager (Controller)	UT16	Dieser Testfall überprüft, ob die get_todo_item Methode des task_manager ein ToDoItem korrekt abrufen. Das abgerufene ToDoItem sollte die folgenden	Vor der Ausführung des Tests muss eine Instanz der TestTaskManager Klasse erstellt werden. Außerdem muss ein ToDoItem mit dem Titel 'Test title', der Priorität	Die Eingabedaten für diesen Test sind die ID des ToDoItem, die als Argument an die get_todo_item Methode übergeben wird. Die Handlung des Benutzers besteht darin, die	Das erwartete Ergebnis ist, dass ein ToDoItem abgerufen wird, das die angegebenen Attribute hat. retrieved_item.due_date.format('YYYY-MM-DD') == todo_item.due_date	retrieved_item.due_date.format('YYYY-MM-DD') == todo_item.due_date

		Attribute haben: Titel, Priorität, Status und Fälligkeitsdatum.	'High', dem Status 'In Progress' und dem Fälligkeitsdatum '2022-01-01' in der Datenbank vorhanden sein.	get_todo_item Methode aufzurufen.		
TaskManager (Controller)	UT17	Dieser Testfall überprüft, ob die update_todo_item Methode des task_manager ein ToDoItem korrekt aktualisiert. Das aktualisierte ToDoItem sollte die folgenden aktualisierten Attribute haben: Titel, Priorität, Status und Fälligkeitsdatum.	Vor der Ausführung des Tests muss eine Instanz der TestTaskManager Klasse erstellt werden. Außerdem muss ein ToDoItem mit dem Titel 'Test title', der Priorität 'High', dem Status 'In Progress' und dem Fälligkeitsdatum '2022-01-01' in der Datenbank vorhanden sein.	Die Eingabedaten für diesen Test sind die ID des ToDoItem und die aktualisierten Attribute 'Updated title', 'Low', 'Done', '2022-02-02', die als Argumente an die update_todo_item Methode übergeben werden. Die Handlung des Benutzers besteht darin, die update_todo_item Methode aufzurufen.	Das erwartete Ergebnis ist, dass das ToDoItem aktualisiert wird und die angegebenen aktualisierten Attribute hat. updated_item.title == 'Updated title' updated_item.priority == 'Low' updated_item.status == 'Done' updated_item.due_date.format('YYYY-MM-DD') == '2022-02-02'	updated_item.title == 'Up-dated title' updated_item.priority == 'Low' updated_item.status == 'Done' updated_item.due_date.format('YYYY-MM-DD') == '2022-02-02'
TaskManager (Controller)	UT18	Dieser Testfall überprüft, ob die delete_todo_item Methode des task_manager ein	Vor der Ausführung des Tests muss eine Instanz der TestTaskManager Klasse erstellt werden.	Die Eingabedaten für diesen Test sind die ID des ToDoItem, die als Argument an die delete_todo_item Methode	Das erwartete Ergebnis ist, dass das ToDoItem aus der Datenbank gelöscht wird und	True

		ToDoItem korrekt löscht. Nach dem Löschen sollte das ToDoItem nicht mehr in der Datenbank vorhanden sein.	Außerdem muss ein ToDoItem mit dem Titel 'Test title', der Priorität 'High', dem Status 'In Progress' und dem Fälligkeitsdatum '2022-01-01' in der Datenbank vorhanden sein.	übergeben wird. Die Handlung des Benutzers besteht darin, die delete_todo_item Methode aufzurufen.	nicht mehr abgerufen werden kann. 'Test title 2' not in [item.title for item in todo_items]	
TaskManager (Controller)	UT19	Dieser Testfall überprüft, ob die delete_all_todo_items Methode des task_manager alle ToDoItem-Einträge korrekt aus der Datenbank löscht. Nach dem Löschen sollten keine ToDoItem-Einträge mehr in der Datenbank vorhanden sein.	Vor der Ausführung des Tests muss eine Instanz der TestTaskManager Klasse erstellt werden. Außerdem müssen mehrere ToDoItem-Einträge in der Datenbank vorhanden sein.	Es gibt keine spezifischen Eingabedaten für diesen Test. Die Handlung des Benutzers besteht darin, die delete_all_todo_items Methode aufzurufen.	Das erwartete Ergebnis ist, dass alle ToDoItem-Einträge aus der Datenbank gelöscht werden und keine Einträge mehr abgerufen werden können. len(todo_items) == 0	len(todo_items) == 0
TaskManager	UT20	Dieser Testfall überprüft, ob die list_todo_items Methode des	Vor der Ausführung des Tests muss eine Instanz der TestTaskManager	Es gibt keine spezifischen Eingabedaten für diesen Test. Die Handlung des Benutzers	Das erwartete Ergebnis ist, dass alle ToDoItem-Einträge aus der Datenbank abgerufen	len(todo_items) == len(self.todo_items)

(Control ler)		task_manager alle ToDoItem-Einträge korrekt aus der Datenbank abrufen. Nach dem Abrufen sollten alle ToDoItem- Einträge in der Liste todo_items vorhanden sein.	Klasse erstellt werden. Außerdem müssen mehrere ToDoItem- Einträge in der Datenbank vorhanden sein.	besteht darin, die list_todo_items Methode aufzurufen.	und in der Liste todo_items gespeichert werden. len(todo_items) == len(self.todo_items)	
TaskMa nager (Control ler)	UT21	Dieser Testfall überprüft, ob die create_todo_item Methode des task_manager einen Fehler auslöst, wenn versucht wird, ein ToDoItem mit einem leeren Titel zu erstellen.	Vor der Ausführung des Tests muss eine Instanz der TestTaskManager Klasse erstellt werden.	Die Eingabedaten für diesen Test sind ein leerer String für den Titel und die Attribute 'High', 'In Progress', '2022-01- 01', die als Argumente an die create_todo_item Methode übergeben werden. Die Handlung des Benutzers besteht darin, die create_todo_item Methode aufzurufen.	Das erwartete Ergebnis ist, dass ein ValueError ausgelöst wird, da der Titel des ToDoItem nicht leer sein darf. True	True

TaskManager (Controller)	UT22	Dieser Testfall überprüft, ob die <code>get_todo_item</code> Methode des <code>task_manager</code> einen Fehler auslöst, wenn versucht wird, ein nicht existierendes <code>ToDoItem</code> abzurufen.	Vor der Ausführung des Tests muss eine Instanz der <code>TestTaskManager</code> Klasse erstellt werden.	Die Eingabedaten für diesen Test ist die ID 9999, die als Argument an die <code>get_todo_item</code> Methode übergeben wird. Die Handlung des Benutzers besteht darin, die <code>get_todo_item</code> Methode aufzurufen.	Das erwartete Ergebnis ist, dass ein <code>DatabaseError</code> ausgelöst wird, da das <code>ToDoItem</code> mit der ID 9999 nicht in der Datenbank existiert.	True
--------------------------	------	---	---	---	--	------

Tabelle 13: Testbeschreibung „Unittests“

(Quelle: Eigene Darstellung)

4.2 Systemtest

Der Systemtest wurde hier als End-to-End-Test durchgeführt (Pittet, 2024), bei dem alle Funktionen der Anwendung einmal real getestet wurden. Ziel war es die Anwendung so friktionslos zu verwenden, wie ein Benutzer es tun würde. Die einzeln durchgeführten Aktionen werden in der nachfolgenden Tabelle als separate Tests ausgewiesen und beschrieben.

Test_ID	Beschreibung	Vorbedingungen	Eingabedaten / Handlung des Benutzers	Erwartetes Ergebnis	Tatsächliches Ergebnis
ST1	Dieser Testfall überprüft, ob das Erstellen eines ToDo Items funktioniert.	Es gibt keine spezifischen Vorbedingungen für diesen Testfall, da er keine externen Abhängigkeiten hat.	Die Eingabedaten für diesen Test ist ein beliebiger ToDo Item Titel. Die Handlung des Benutzers besteht darin, den Titel in das dafür vorgesehene Inputfeld einzutragen und mit „Enter“ oder dem “Submit“ Button zu bestätigen.	Das erwartete Ergebnis ist, dass ein ToDo Item mit dem eingegebenen Titel im Feld linken Kanban Feld „To Do“ erscheint.	Das tatsächliche Ergebnis ist, dass ein ToDo Item mit dem eingegebenen Titel im Feld linken Kanban Feld „To Do“ erscheint.
ST2	Dieser Testfall überprüft, ob das Öffnen des Bearbeitungsfensters für die Eigenschaften eines ToDo Items funktioniert.	Vor der Ausführung des Tests muss bereits ein ToDo Item erstellt worden sein.	Eingabedaten sind für diesen Test nicht erforderlich. Die Handlung des Benutzers besteht darin, mit der Maus einen linken Doppelklick auf ein ToDo Item auszuführen.	Das erwartete Ergebnis ist, dass sich das Bearbeitungsfenster automatisch öffnet.	Das tatsächliche Ergebnis ist, dass sich das Bearbeitungsfenster automatisch öffnet.
ST3	Dieser Testfall überprüft, ob das Ändern des ToDo	Vor der Ausführung des Tests muss bereits ein	Die Eingabedaten für diesen Test sind ein beliebiger Titel.	Das erwartete Ergebnis ist, dass sich der Titel des ToDo Items in der	Das tatsächliche Ergebnis ist, dass sich der Titel des ToDo Items in der

	Item funktioniert.	Titels	ToDo Item erstellt worden sein.	Die Handlung des Benutzers besteht darin, das Eingabefeld zu öffnen, die Änderung durchzuführen, die Eingabe mit dem dafür vorgesehenen „Submit“ Button zu bestätigen und im Anschluss die Änderung durch erneutes Öffnen des Eingabefensters zu überprüfen.	Anwendungsansicht ändert.	Anwendungsansicht ändert.
ST4	Dieser Testfall überprüft, ob das Ändern der ToDo Item Priorität funktioniert.		Vor der Ausführung des Tests muss bereits ein ToDo Item erstellt worden sein.	Die Eingabedaten für diesen Test sind eine beliebige Priorität. Die Handlung des Benutzers besteht darin, das Eingabefeld zu öffnen, die Änderung durchzuführen, die Eingabe mit dem dafür vorgesehenen „Submit“ Button zu bestätigen und im Anschluss die Änderung durch erneutes Öffnen des Eingabefensters zu überprüfen.	Das erwartete Ergebnis ist, dass sich die Eigenschaft „Priorität“ geändert hat.	Das tatsächliche Ergebnis ist, dass sich die Eigenschaft „Priorität“ geändert hat.
ST5	Dieser Testfall überprüft, ob das Ändern des ToDo Item Status funktioniert.		Vor der Ausführung des Tests muss bereits ein ToDo Item erstellt worden sein.	Die Eingabedaten für diesen Test sind ein beliebiger Status. Die Handlung des Benutzers besteht darin, das Eingabefeld zu öffnen, die Änderung durchzuführen, die Eingabe mit dem dafür vorgesehenen „Submit“ Button zu bestätigen und im Anschluss die Änderung durch erneutes Öffnen des Eingabefensters zu überprüfen.	Das erwartete Ergebnis ist, dass sich die Eigenschaft „Status“ geändert hat.	Das tatsächliche Ergebnis ist, dass sich die Eigenschaft „Status“ geändert hat.

ST6	Dieser Testfall überprüft, ob das Ändern des ToDo Item Fälligkeitsdatums im richtigen Format funktioniert.	Vor der Ausführung des Tests muss bereits ein ToDo Item erstellt worden sein.	Die Eingabedaten für diesen Test sind ein Fälligkeitsdatum im richtigen Format (YYYY/MM/DD). Die Handlung des Benutzers besteht darin, das Eingabefeld zu öffnen, die Änderung durchzuführen, die Eingabe mit dem dafür vorgesehenen „Submit“ Button zu bestätigen und im Anschluss die Änderung durch erneutes Öffnen des Eingabefensters zu überprüfen.	Das erwartete Ergebnis ist, dass sich die Eigenschaft „Fälligkeitsdatum“ geändert hat.	Das tatsächliche Ergebnis ist, dass sich die Eigenschaft „Fälligkeitsdatum“ geändert hat.
ST7	Dieser Testfall überprüft, ob das Ändern des ToDo Item Fälligkeitsdatums im falschen Format sowie Auslösen einer Fehlermeldung funktioniert.	Vor der Ausführung des Tests muss bereits ein ToDo Item erstellt worden sein.	Die Eingabedaten für diesen Test sind ein Fälligkeitsdatum in Abweichung zum Format (YYYY/MM/DD). Die Handlung des Benutzers besteht darin, das Eingabefeld zu öffnen, die Änderung durchzuführen, die Eingabe mit dem dafür vorgesehenen „Submit“ Button zu bestätigen.	Das erwartete Ergebnis ist, dass sich ein Fenster mit einem Hinweis öffnet, welcher das zu nutzende Format beschreibt.	Das tatsächliche Ergebnis ist, dass sich ein Fenster mit einem Hinweis öffnet, welcher das zu nutzende Format beschreibt.
ST8	Dieser Testfall überprüft, ob das Löschen eines ToDo Items funktioniert.	Vor der Ausführung des Tests muss bereits ein ToDo Item erstellt worden sein.	Eingabedaten sind für diesen Test nicht erforderlich. Die Handlung des Benutzers besteht darin, mit einem Rechtsklick der Maus auf dem ausgewählten ToDo Item den Löschvorgang zu initiieren und ihn mit Drücken des „Confirm“ Buttons nach Öffnen des Bestätigungsfenster abzuschließen.	Das erwartete Ergebnis ist, dass das angewählte ToDo aus der Anwendungsansicht verschwindet.	Das tatsächliche Ergebnis ist, dass das angewählte ToDo aus der Anwendungsansicht verschwindet.

ST9	Dieser Testfall überprüft, ob das Durchführen der Sortierfunktion nach Priorität funktioniert.	Vor der Ausführung des Tests müssen bereits mindestens zwei ToDo Items erstellt worden sein, die erstens in der gleichen Spalte des Kanban Boards liegen und zweitens zwei unterschiedliche Prioritäten aufweisen.	Eingabedaten sind für diesen Test nicht erforderlich. Die Handlung des Benutzers besteht darin, mit der linken Maustaste auf den Button mit den drei Punkten zu klicken und die gewünschte Sortierfunktion aus dem Drop-Down-Menu auszuwählen. Anschließend muss er die Sortierung über den dafür vorgesehenen Button bestätigen.	Das erwartete Ergebnis ist, dass sich die ToDo Items nach ihrer Priorität in allen drei Spalten des Kanban Boardes sortieren. Die höchste Priorität ist dabei ganz oben, die geringste ganz unten.	Das tatsächliche Ergebnis ist, dass sich die ToDo Items nach ihrer Priorität in allen drei Spalten des Kanban Boardes sortieren. Die höchste Priorität ist dabei ganz oben, die geringste ganz unten.
ST10	Dieser Testfall überprüft, ob das Durchführen der Sortierfunktion nach Fälligkeitsdatum funktioniert.	Vor der Ausführung des Tests müssen bereits mindestens zwei ToDo Items erstellt worden sein, die erstens in der gleichen Spalte des Kanban Boards liegen und zweitens zwei unterschiedliche Fälligkeitsdaten aufweisen.	Eingabedaten sind für diesen Test nicht erforderlich. Die Handlung des Benutzers besteht darin, mit der linken Maustaste auf den Button mit den drei Punkten zu klicken und die gewünschte Sortierfunktion aus dem Drop-Down-Menu auszuwählen. Anschließend muss er die Sortierung über den dafür vorgesehenen Button bestätigen.	Das erwartete Ergebnis ist, dass sich die ToDo Items nach ihrem Fälligkeitsdatum in allen drei Spalten des Kanban Boardes sortieren. Das nach Datum als nächstes zu erledigende ToDo Item ist dabei ganz oben, das Item mit der längsten Dauer bis zum Fälligkeitsdatum ganz unten.	Das tatsächliche Ergebnis ist, dass sich die ToDo Items nach ihrem Fälligkeitsdatum in allen drei Spalten des Kanban Boardes sortieren. Das nach Datum als nächstes zu erledigende ToDo Item ist dabei ganz oben, das Item mit der längsten Dauer bis zum Fälligkeitsdatum ganz unten.

ST11	Dieser Testfall überprüft, ob das Nutzen der Drag and Drop Funktion funktioniert.	Vor der Ausführung des Tests muss mindestens ein ToDo Item erstellt worden sein.	Eingabedaten sind für diesen Test nicht erforderlich. Die Handlung des Benutzers besteht darin, ein ToDo Item mit der linken Maustaste anwählen und die Taste dabei gedrückt halten. So geschehen muss er das ToDo Item von einer Kanban Spalte in die andere ziehen. Im Anschluss öffnet er das Bearbeitungsfenster der Eigenschaften mit einem doppelten Linksklick auf der Maus, um die Änderung der Statuseigenschaften zu überprüfen.	Das erwartete Ergebnis ist, dass sich das ToDo Item in der neu zugeordneten Spalte des Kanban Boardes befindet und die Eigenschaft „Status“ aktualisiert wurde.	Das tatsächliche Ergebnis ist, dass sich das ToDo Item in der neu zugeordneten Spalte des Kanban Boardes befindet und die Eigenschaft „Status“ aktualisiert wurde.
ST12	Dieser Testfall überprüft, ob sich die Farbe des ToDo Items bei Nutzung der Drag and Drop Funktion grün färbt.	Vor der Ausführung des Tests muss mindestens ein ToDo Item erstellt worden sein.	Eingabedaten sind für diesen Test nicht erforderlich. Die Handlung des Benutzers besteht darin, ein ToDo Item via Drag and Drop zu verschieben.	Das erwartete Ergebnis ist, dass sich die Farbe des ToDo Items beim Verschieben grün färbt.	Das tatsächliche Ergebnis ist, dass sich die Farbe des ToDo Items beim Verschieben grün färbt.
ST13	Dieser Testfall überprüft, ob die ToDo Items dauerhaft gespeichert sind und nach dem Schließen und Öffnen der	Vor der Ausführung des Tests muss mindestens ein ToDo Item erstellt worden sein.	Eingabedaten sind für diesen Test nicht erforderlich. Die Handlung des Benutzers besteht darin, die Anwendung zu schließen und erneut zu öffnen.	Das erwartete Ergebnis ist, dass sich nach Schließen und Öffnen der Anwendung alle zuvor erstellten ToDo Items wiederfinden.	Das tatsächliche Ergebnis ist, dass sich nach Schließen und Öffnen der Anwendung alle zuvor erstellten ToDo Items wiederfinden.

	Anwendung noch vorhanden sind.				
--	-----------------------------------	--	--	--	--

Tabelle 14: Testbeschreibung „Systemtest“

(Quelle: Eigene Darstellung)

4.3 Anwendungstest Windows

Zusätzlich zu den durchgeführten Unit- und Systemtest wurde die Anwendung einem weiteren Test unter „Windows-ähnlichen“ Bedingungen unterzogen. Hintergrund ist, dass auf MacOS programmiert wurde und kein eigener Computer mit Windows zur Verfügung stand. Hierzu wurde Wine verwendet. Dabei handelt es sich um eine Kompatibilitätsschicht, die es ermöglicht, Windows-Anwendungen auf anderen Betriebssystemen wie Linux, macOS und BSD auszuführen. Es übersetzt Windows-API-Aufrufe in POSIX-Aufrufe in Echtzeit, was die Leistung und den Speicherbedarf im Vergleich zu anderen Methoden reduziert und eine saubere Integration von Windows-Anwendungen ermöglicht. Der Vorteil von Wine ist das einfache Testen ohne eine tatsächliche Windows-Maschine oder eine virtuelle Maschine zu benötigen. Kompatibilitätsprobleme können auf diese Weise aber trotzdem frühzeitig erkannt und behoben werden.

Die nachfolgenden Abbildungen zeigen die Ausführung des Programmes im Terminal sowie das daraus resultierende Ergebnis unter „Windows-ähnlichen“ Bedingungen.



```
sebastianahlburg — wine-preloader ~/Downloads/main.exe __CFBundleIdentifier=com.a...
Last login: Sun Apr 21 10:02:14 on ttys011
[sebastianahlburg@MacBook-Pro-2 ~ % wine /Users/sebastianahlburg/Downloads/main.exe
007c:fixme:hid:handle_IRP_MN_QUERY_ID Unhandled type 00000005
007c:fixme:hid:handle_IRP_MN_QUERY_ID Unhandled type 00000005
007c:fixme:hid:handle_IRP_MN_QUERY_ID Unhandled type 00000005
007c:fixme:hid:handle_IRP_MN_QUERY_ID Unhandled type 00000005
0024:fixme:ntdll:NtQueryInformationToken QueryInformationToken( ..., TokenAppContainerSid, ...
) semi-stub
010c:fixme:file:NtLockFile I/O completion on lock not implemented yet
qt.qpa.window: QtWindows::DpiAwareness::PerMonitorVersion2 is not supported by c
urrent system.
010c:fixme:win:RegisterPowerSettingNotification (0000000000010068,{02731015-4510-4526-99e6-e5a
17ebd1aea},0): stub
010c:fixme:system:NtUserQueryDisplayConfig flags 0x2, paths_count 0x208d2c, paths 0x1b8dc70, m
odes_count 0x208d28, modes 0x1b89fd0, topology_id 0x0 semi-stub
010c:fixme:setupapi:SetupDiOpenDeviceInterfaceW 0000000001B87690 L"\\\\?\\DISPLAY#Default_Moni
tor#0000&0000#{E6F07B5F-EE97-4A90-B076-33F57BF4EAA7}" 00000001 0000000002089E0
qt.qpa.screen: "Unable to open monitor interface to \\.\\DISPLAY1:" "Erfolg."
010c:fixme:combase:RoGetActivationFactory (L"Windows.UI.ViewManagement.UISettings", {00000035-
0000-0000-c000-000000000046}, 0000000000209088): semi-stub
010c:fixme:ui:factory_QueryInterface {94ea2b94-e9cc-49e0-c0ff-ee64ca8f5b90} not implemented, r
eturning E_NOINTERFACE.
010c:fixme:d3d:wined3d_guess_card_vendor Received unrecognized GL_VENDOR "Apple". Returning HW
_VENDOR_NVIDIA.
010c:fixme:ntdll:NtQuerySystemInformation info_class SYSTEM_PERFORMANCE_INFORMATION
010c:fixme:msg:ChangeWindowMessageFilter c048 00000001
010c:fixme:system:EnableNonClientDpiScaling (0000000000030074): stub
EnableNonClientDpiScaling() failed for HWND 0x30074 (120) (Aufruf nicht implemen
tiert.)
010c:fixme:win:IsTouchWindow hwnd 0000000000030074, flags 00000000020A3A0 stub!
```

Abbildung 10: Ausführung mit Wine

(Quelle: Eigene Darstellung)

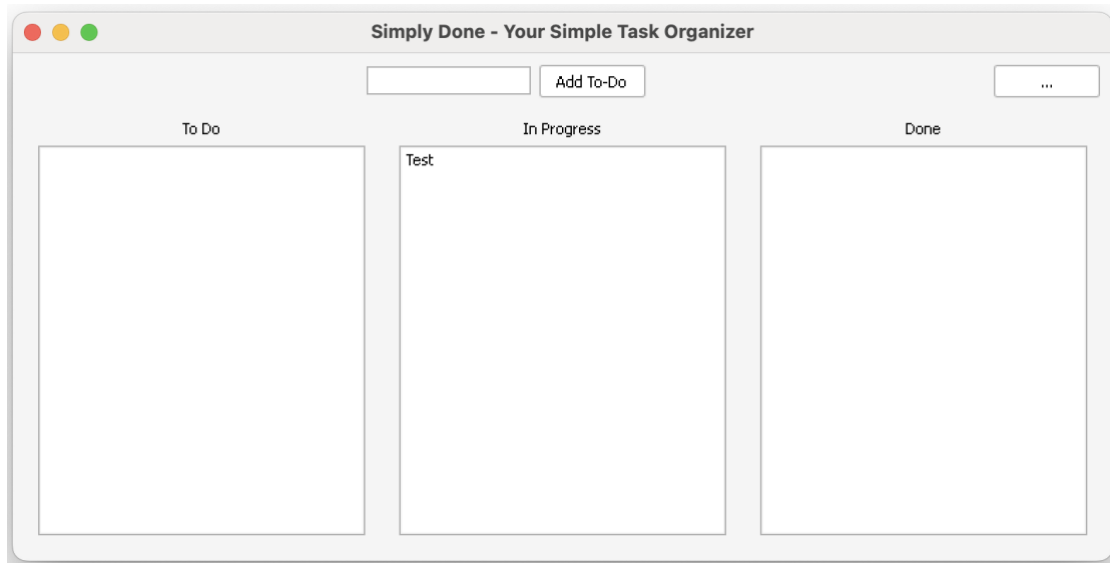


Abbildung 11: Interface Darstellung unter Windows

(Quelle: Eigene Darstellung)

Wie an den Abbildungen ersichtlich unterscheidet sich das Interface unter „Windows-ähnlichen“ Bedingungen leicht von der Darstellung unter MacOS. Während die Darstellung unter den originären Bedingungen als leicht „schöner“ wahrgenommen werden, sind die Funktionen im vollen Umfang nutzbar und produzierten keine Fehlermeldungen.

5. Abstract

Ziel des Abstracts ist es die Erfahrungen und Erkenntnisse, die während der Durchführung gesammelt wurden, zu reflektieren und als wertvolle „Lessons Learned“ festzuhalten. Sie ermöglicht es, die Herausforderungen, Erfolge und Verbesserungsmöglichkeiten des Projekts zu analysieren und wertvolle Erkenntnisse für zukünftige Projekte zu gewinnen.

5.1 Making-of

Zunächst soll die Durchführung des Projektes anhand seiner drei Phasen inklusive der darin getroffenen, zentralen Entscheidungen kurz zusammengefasst werden, um darauf aufbauend Schlüsse für zukünftig Projekte zu ziehen:

- a) *Konzeptionsphase*: Start bildete die Planung des Projekts, die Definition von Zielen, Anforderungen und Zeitplan. Nach der Planung wurde sich auf das Design der Anwendung und ihre Architektur konzentriert. Die Entscheidung fiel auf Python als Programmiersprache, PySide als Framework und das MVC-Designmuster. UML-Diagramme sollten hier helfen, die Struktur der Anwendung zu skizzieren.
- b) *Umsetzungs- und Reflexionsphase*: Nach erfolgter Planung begann die eigentliche Entwicklung der Anwendung. Die verschiedenen Funktionen wurden gemäß den Anforderungen in einem ansprechenden UI-Design umgesetzt. Bereits während der Implementierung wurden umfangreiche Tests durchgeführt, um sicherzustellen, dass die Anwendung korrekt funktionierte und allen Anforderungen entsprach.
- c) *Finalisierungsphase*: Zum Ende wurden letzte Anpassung vorgenommen, ein Benutzerhandbuch erstellt, die Anwendung zur Verteilung vorbereitet und Lessons Learned durchgeführt.

5.2 Reflexion

Die berufsbegleitende Entwicklung der Anwendung gestaltete sich als anspruchsvoll. Die zeitlichen Einschränkungen erforderten eine effiziente Planung und Nutzung jeder verfügbarer Entwicklungszeit. Dabei stellte sich heraus, dass es nicht einfach war eine ausgewogene Balance zwischen den beruflichen Verpflichtungen und dem Projekt zu finden, um sowohl die Arbeitsanforderungen zu erfüllen als auch das Projekt voranzutreiben.

In Bezug auf das Designkonzept erwies sich die Entscheidung für ein ästhetisch ansprechendes und benutzerfreundliches Interface als erfolgreich. Durch die klare Strukturierung und intuitive Benutzerführung konnte eine angenehme und effiziente Nutzung der Anwendung gewährleistet werden. Die Auswahl der Bibliotheken, darunter PySide für das Framework und SQLAlchemy für die Datenbankinteraktion, erwies sich aus der Perspektive eines noch recht jungen, unerfahrenen Programmierers als strategisch klug. Denn die Bibliotheken boten nicht nur eine solide technische Basis, sondern ermöglichten auch eine schnellere Entwicklung durch ihre benutzerfreundlichen Schnittstellen und umfangreichen Funktionen. Gleiches gilt auch für Python als Programmiersprache, welche eine einfache Syntax aufweist.

Als technische Herausforderung stellte sich die Integration der einzelnen Klassen, Methoden, etc. dar. Erstmals wurde auch ein MVC-Entwurfsmuster verwendet im Rahmen dessen ein friktionsloses Interagieren der verschiedenen Komponenten sichergestellt werden musste. Durch eine iterative Vorgehensweise und kontinuierliches Debugging konnten jedoch die meisten Probleme erfolgreich gelöst werden. Zusätzlich gestaltete sich die Verteilung und Nutzung unter Windows 10/11 als Herausforderung, da kein eigener Computer mit einem entsprechenden Betriebssystem zur Verfügung stand. Zwar wurde durch einen Test unter „Windows-ähnlichen“ Bedingungen Abhilfe geschaffen, jedoch sollte insbesondere bei kommerziell nutzbaren Programmen ein Test im echten Zielsystem durchgeführt werden.

In Bezug auf die Zeitplanung und Ressourcenallokation wäre es ratsam gewesen, realistischere Zeitvorgaben zu setzen und Pufferzeiten für unvorhergesehene Komplikationen einzuplanen. Die begrenzte Verfügbarkeit von Entwicklungszeit führte zu einem zeitlichen Verzug von vier Wochen gegenüber dem ursprünglichen Zeitplan. Diese Verzögerung hätte durch eine bessere Priorisierung und effizientere Nutzung der verfügbaren Zeit ggf. minimiert werden können.

Eine weitere wichtige Erkenntnis aus dem Projekt war die Bedeutung des gegenseitigen Austauschs. Der regelmäßige Dialog mit Kommilitonen und die Teilnahme an Veranstaltungen der Hochschule ermöglichten einen kontinuierlichen Wissensaustausch und gegenseitige Unterstützung. Insbesondere der Austausch mit einem anderen Studenten, der zur gleichen Zeit ein Projekt durchführte, erwies sich als äußerst hilfreich. Durch den Vergleich von Erfahrungen und die gemeinsame Problemlösung konnten wir Herausforderungen schneller bewältigen und voneinander lernen.

Insgesamt kann das Projekt als Erfolg gesehen werden, da es zu einem erfolgreichen Abschluss kam und auf dem Weg dahin nicht nur Lerneffekte erzeugte, sondern auch Spaß brachte.

Literaturverzeichnis

- Arrow. (2024): *Better dates & times for Python*. <https://arrow.readthedocs.io/en/latest/>
- Rehkopf, Max. (2024): *User Story smit Beispielen und Vorlagen*. Atlassian. <https://www.atlassian.com/de/agile/project-management/user-stories>
- BMU Verlag. (2020): *Model View Controller*. <https://bmu-verlag.de/model-view-controller-mvc-softwareentwicklungsmuster/>
- BMU Verlag. (2024): *Überblick über verschiedene Programmiersprachen*. <https://bmu-verlag.de/uberblick-uber-verschiedene-programmiersprachen/>
- Green, Nicko. (2021): *Einführung in System Kontext Diagramme mit editierbaren Beispielen*. Gitmind. <https://gitmind.com/de/kontextdiagramm.html>
- Ionos. (2023): *Aktivitätsdiagramme: Chronologische Abläufe von Aktivitäten mit UML übersichtlich darstellen*. <https://www.ionos.de/digitalguide/websites/web-entwicklung/uml-aktivitaetsdiagramme/>
- Ionos. (2020): *Das Use-Case-Diagramm (Anwendungsfalldiagramm) in UML*. <https://www.ionos.de/digitalguide/websites/web-entwicklung/anwendungsfalldiagramm/>
- MVPS. (2019): *Design Patterns – Singleton, Factory, Observer*. <https://www.mvps.net/docs/design-patterns-singleton-factory-observer/>
- PyInstaller. (2024): *PyInstaller Manual*. <https://www.pyinstaller.org>
- PyPi. (2024): *Python bindings for the Qt cross-platform application and UI framework*. <https://pypi.org/project/PySide/>
- Krekel, Holger. (2015): *Get Started*. Pytest. <https://docs.pytest.org/en/7.4.x/getting-started.html>
- Sten, Pittet (2024): *Unterschiedliche Arten von Softwaretests*. <https://www.atlassian.com/de/continuous-delivery/software-testing/types-of-software-testing>
- Scand. (2021): *Funktionale vs. Nicht-funktionale Anforderungen: Ein Leitfaden mit Definitionen*. <https://scand.de/unternehmen/blog/funktionale-vs-nicht-funktionale-anforderungen/>

Koojiman, Sebastian. (2024): *User Story*. Agile Scrum Group. <https://scrumguide.de/user-story/>

Scrum.org. (2024): *What is Scrum?* <https://www.scrum.org/learning-series/what-is-scrum/>

SQLAlchemy. (2024): *The Python SQL Toolkit and Object Relational Mapper*.
<https://www.sqlalchemy.org>

Verzeichnis der Anhänge

A.1 – Automatisierte Testprotokolle

A.1.1 Testprotokoll ToDoItem

test_report_todo_item.html

Report generated on 09-Apr-2024 at 17:47:33 by [pytest-html](#) v4.1.1

Environment


Python	3.11.3
Platform	macOS-14.0-arm64-arm-64bit
Packages	<ul style="list-style-type: none">• pytest: 8.0.2• pluggy: 1.4.0
Plugins	<ul style="list-style-type: none">• html: 4.1.1• metadata: 3.1.1• mock: 3.14.0

Summary

1 test took 0 ms.

(Un)check the boxes to filter the results.

✓ 0 Failed, ☒ 1 Passed, ✓ 0 Skipped, ✓ 0 Expected failures, ✓ 0 Unexpected passes, ✓ 0 Errors, ✓ 0 Reruns [Show all details](#) / [Hide all details](#)

Result 	Test	Duration	Links
Passed	tests/model/test_todo_item.py::TestToDoItem::test_todo_item_initialization	0 ms	

A.1.2 Testprotokoll DatabaseManager

report.html

Report generated on 09-Apr-2024 at 17:41:25 by [pytest-html](#) v4.1.1

Environment


Python	3.11.3
Platform	macOS-14.0-arm64-arm-64bit
Packages	<ul style="list-style-type: none">• pytest: 8.0.2• pluggy: 1.4.0
Plugins	<ul style="list-style-type: none">• html: 4.1.1• metadata: 3.1.1• mock: 3.14.0

Summary

8 tests took 5 ms.

(Un)check the boxes to filter the results.

✓ 0 Failed, ☒ 8 Passed, ✓ 0 Skipped, ✓ 0 Expected failures, ✓ 0 Unexpected passes, ✓ 0 Errors, ✓ 0 Reruns [Show all details](#) / [Hide all details](#)

Result 	Test	Duration	Links
Passed	tests/model/test_database_manager.py::TestDatabaseManager::test_insert_todo_item	3 ms	
Passed	tests/model/test_database_manager.py::TestDatabaseManager::test_get_todo_item	0 ms	
Passed	tests/model/test_database_manager.py::TestDatabaseManager::test_update_todo_item	0 ms	
Passed	tests/model/test_database_manager.py::TestDatabaseManager::test_delete_todo_item	0 ms	
Passed	tests/model/test_database_manager.py::TestDatabaseManager::test_delete_all_todo_items	0 ms	
Passed	tests/model/test_database_manager.py::TestDatabaseManager::test_list_todo_items	0 ms	
Passed	tests/model/test_database_manager.py::TestDatabaseManager::test_get_todo_items_sorted_by	0 ms	
Passed	tests/model/test_database_manager.py::TestDatabaseManager::test_update_todo_status	0 ms	

A.1.3 Testprotokoll UserInterface

test_report_UserInterface.html

Report generated on 09-Apr-2024 at 17:48:29 by [pytest-html](#) v4.1.1


Environment

Python	3.11.3
Platform	macOS-14.0-arm64-arm-64bit
Packages	<ul style="list-style-type: none">• pytest: 8.0.2• pluggy: 1.4.0
Plugins	<ul style="list-style-type: none">• html: 4.1.1• metadata: 3.1.1• mock: 3.14.0

Summary

5 tests took 222 ms.

(Un)check the boxes to filter the results.

✓ 0 Failed, <input checked="" type="checkbox"/> 5 Passed, <input checked="" type="checkbox"/> 0 Skipped, ✓ 0 Expected failures, ✓ 0 Unexpected passes, ✓ 0 Errors, ✓ 0 Reruns				Show all details / Hide all details
Result 	Test	Duration	Links	
Passed	tests/view/test_user_interface.py::TestUserInterface::test_add_todo	202 ms		
Passed	tests/view/test_user_interface.py::TestUserInterface::test_create_labeled_widget	2 ms		
Passed	tests/view/test_user_interface.py::TestCustomListWidget::test_custom_list_widget_drag	11 ms		
Passed	tests/view/test_user_interface.py::TestCustomListWidget::test_custom_list_widget_drop	5 ms		
Passed	tests/view/test_user_interface.py::TestCustomListWidget::test_custom_list_widget_right_click	3 ms		

A.1.4 Testprotokoll TaskManager

test_report_TaskManager.html

Report generated on 09-Apr-2024 at 17:49:11 by [pytest-html](#) v4.1.1


Environment

Python	3.11.3
Platform	macOS-14.0-arm64-arm-64bit
Packages	<ul style="list-style-type: none">• pytest: 8.0.2• pluggy: 1.4.0
Plugins	<ul style="list-style-type: none">• html: 4.1.1• metadata: 3.1.1• mock: 3.14.0

Summary

8 tests took 14 ms.

(Un)check the boxes to filter the results.

✓ 0 Failed, <input checked="" type="checkbox"/> 8 Passed, <input checked="" type="checkbox"/> 0 Skipped, ✓ 0 Expected failures, ✓ 0 Unexpected passes, ✓ 0 Errors, ✓ 0 Reruns				Show all details / Hide all details
Result 	Test	Duration	Links	
Passed	tests/controller/test_task_manager.py::TestTaskManager::test_create_todo_item	2 ms		
Passed	tests/controller/test_task_manager.py::TestTaskManager::test_get_todo_item	1 ms		
Passed	tests/controller/test_task_manager.py::TestTaskManager::test_update_todo_item	2 ms		
Passed	tests/controller/test_task_manager.py::TestTaskManager::test_delete_todo_item	2 ms		
Passed	tests/controller/test_task_manager.py::TestTaskManager::test_delete_all_todo_items	2 ms		
Passed	tests/controller/test_task_manager.py::TestTaskManager::test_list_todo_items	4 ms		
Passed	tests/controller/test_task_manager.py::TestTaskManager::test_create_todo_item_with_empty_title	0 ms		
Passed	tests/controller/test_task_manager.py::TestTaskManager::test_get_nonexistent_todo_item	1 ms		