IU Internationale Hochschule

FS Master of Science Informatik

Projekt: Software Engineering

Portfolio zum Thema:

Time Tracker

Vorgelegt von:

Patrick Walser

Matrikelnummer: 9224643

Abgabedatum: 15.03.2025

Prüfer/Prüferin: Prof. Dr. Markus Kleffmann

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Projektdokumentation

1.1 Projektübersicht

Im vorliegenden Projekt *Time Tracker* soll eine Desktopanwendung zur Zeiterfassung erstellt werden. Die Anwendung muss es ermöglichen, Zeiten aufzuzeichnen und nachträglich zu bearbeiten bzw. zu erstellen. Es sollen unterschiedliche Möglichkeiten zur Analyse, anhand von Visualisierungen mit unterschiedlichen Detaillierungsgraden, zur Verfügung gestellt werden. Es ist kein Ziel Möglichkeiten zur Dokumentation von Aufwänden für kollaborative Projekte oder Gruppenarbeiten zur Verfügung zu stellen. Die Versionskontrolle erfolgt mit einem GitHub-Repository das unter folgendem Link erreichbar ist (GitHub Repository).

1.2 Risikomanagement

In Tabelle 1 sind mögliche Projektrisiken und Maßnahmen zur Reduktion dieser Risiken aufgelistet. Es sind der Einfluss und die Eintrittswahrscheinlichkeit vor dem Umsetzen der Maßnahmen dargestellt. Die Risiken werden während der Projektlaufzeit und zu jedem Meilenstein neu bewertet.

Tabelle 1: Projektrisiken und Maßnahmen

#	Risiko Beschreibung	Einfluss auf das Projekt (Kosten, Zeit, Funktion, Qualität, Profitabilität)	Wahrscheinlichkeit 0…100%	Einfluss 1...10	Gefährdungspotenzial Produkt der Wahrscheinlichkeit und des Einflusses	Entschärfungsmaßnahmen Diese Aktionen werden im Projektplan berücksichtigt.
1	Anforderungsänderungen	Zeit	50%	4	2,00	Feedback nach der Konzeptionsphase einholen und einpflegen
2	Ressourcenknappheit	Zeit, Qualität	80%	9	7,20	Ressourcenplanung im eigenen Kalender
3	Zeitplanüberschreitung	Zeit	60%	5	3,00	Pufferzeiten einplanen, detaillierter Zeitplan
4	Qualitätsmängel	Qualität, Funktion	20%	5	1,00	Unit Tests und Manuelle Tests ausführen
5	Unzureichende Leistung	Qualität, Funktion	50%	8	4,00	Performance-Optimierungen, skalierbare Architektur
6	Kompatibilitätsprobleme mit verwendeten Bibliotheken	Funktion	10%	8	0,80	Definierte Versionsstände über virtuelle Umgebung verwenden und publizieren

Die dazugehörende Risikomatrix ist in Abbildung 1 dargestellt. Es ist erkennbar, dass die Ressourcenknappheit das größte Risiko bildet und dass auf sie besonders geachtet werden muss.

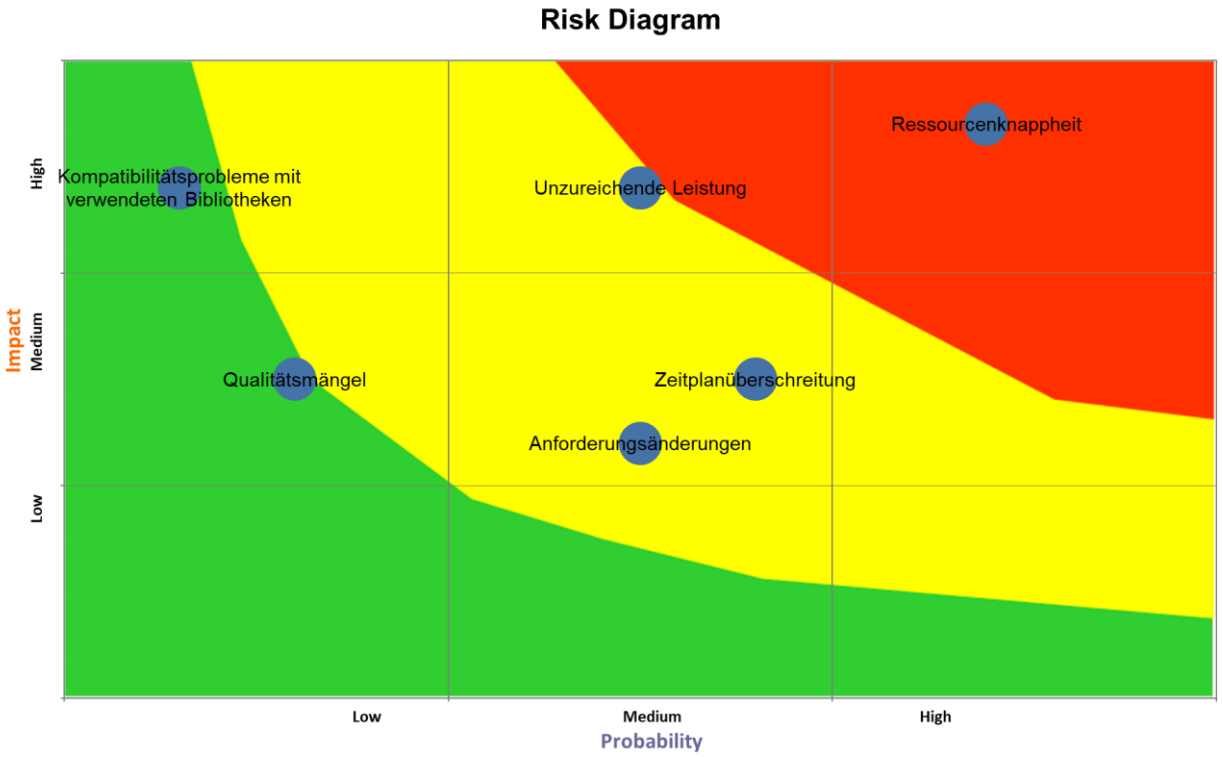

Abbildung 1: Risikomatrix

1.3 Zeitplanung

Als Meilensteine werden die durch den Projektplan vorgegebenen Phasen verwendet. Nach jeder Phase erfolgt eine Abgabe und ein Feedback durch den Betreuer, was einer Abnahme des Meilensteins entspricht. Der in Abbildung 2 und Abbildung 3 dargestellte Zeitplan in Form eines Gantt-Diagramms wird während der Projektlaufzeit gewartet und der Fortschritt der Aufgaben wird eingetragen. Aufgrund der Abbildungsgröße ist der Zeitplan in zwei Teilen dargestellt. Im ersten Teil ist die Konzeptionsphase dargestellt (Abbildung 2). Die Erarbeitungs- und Reflexionsphase und die Finalisierungsphase sind im zweiten Teil dargestellt (Abbildung 3).

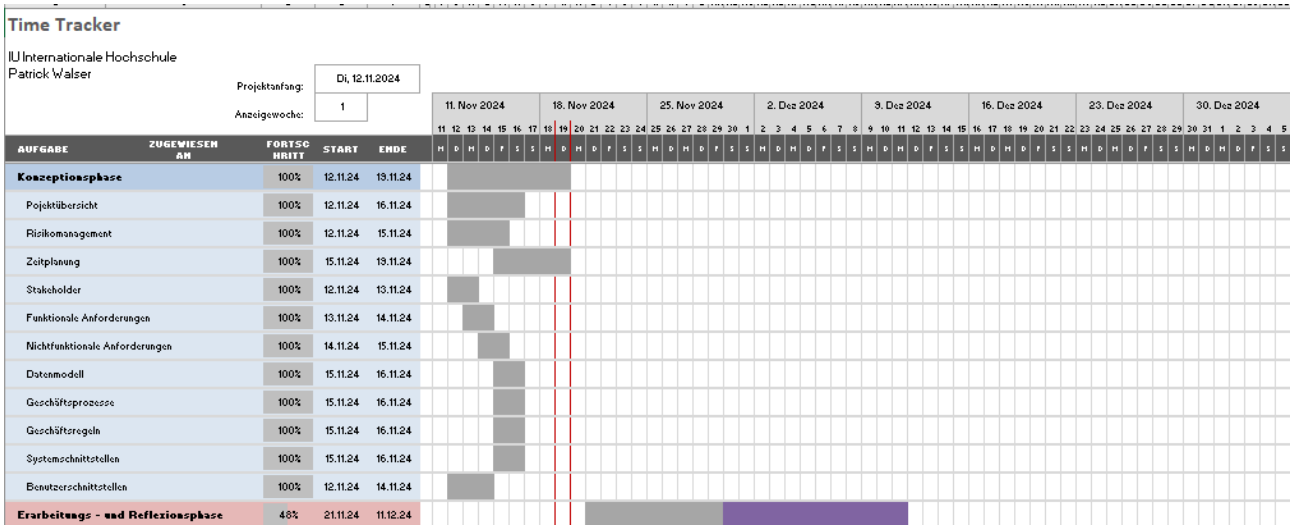

Abbildung 2: Zeitplan 1

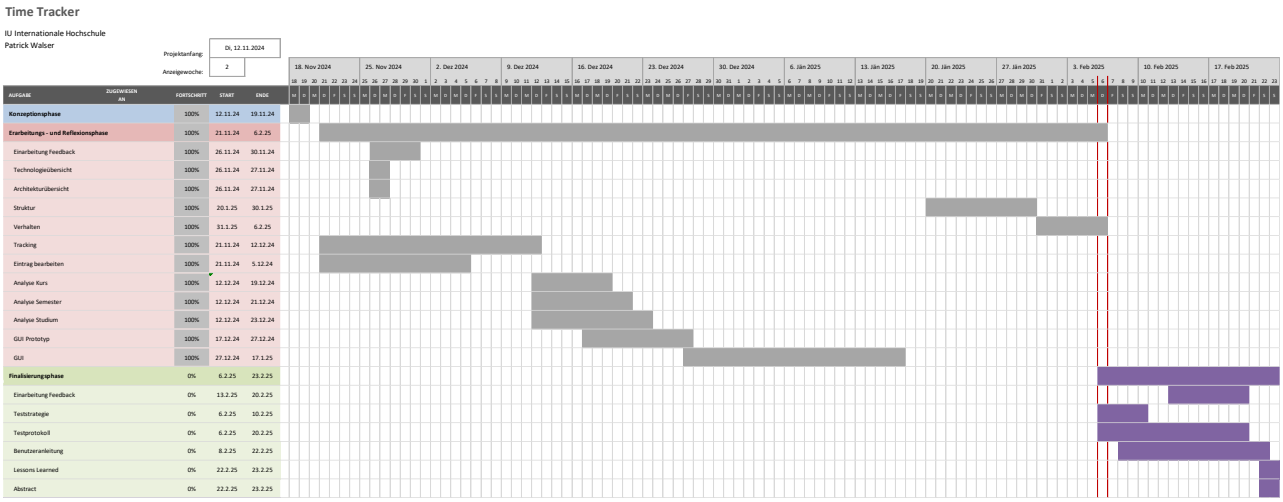

Abbildung 3: Zeitplan 2

1.4 Benutzeranleitung

1.4.1 Installation

Um das Projekt zu installieren muss der Inhalt aus dem GitHub-Repository ausgecheckt werden. Für eine einfache Installation liegt ein .zip Archiv im Repository, welches entpackt werden kann. Im Entpackten Ordner kann die ausführbare Datei direkt gestartet werden. Es ist zu beachten, dass der erste Start der Anwendung etwas länger dauern kann und die Performance im Gegensatz zu einer Ausführung in Python geringer sein kann.

Alternativ kann die Anwendung in Python ausgeführt werden. Dazu muss Python sowie ein Paketmanager installiert sein. Die benötigten Pakete sind in der Datei program_requirements.yml aufgelistet. Mit Hilfe dieser Datei kann eine virtuelle Umgebung erstellt werden, in der die notwendigen Pakete mit der korrekten Version installiert sind. Diese virtuelle Umgebung kann z.B. mit Anaconda automatisiert erstellt werden.

1.4.2 Ausführung

Beim ersten Start der Anwendung muss das Studium konfiguriert werden. Dazu wird ein zusätzlicher Dialog geöffnet, in dem die Konfigurationsdaten eingegeben werden müssen. Anschließend wird ein Speicherort für die Datendatei ausgewählt.

Menü

Über das Menü File kann ein neues Studium erstellt, ein bereits erstelltes Studium geöffnet oder das aktuelle Studium unter einem anderen Namen gespeichert werden. Zusätzlich können die Standardeinstellungen für Module geändert werden.

Über das Menü Edit kann die Konfiguration des Studiums oder der im Studium bereits in Aufzeichnungen vorhandenen Semester bearbeitet werden.

Ansichten

Die Anwendung besteht aus zwei unterschiedlichen Ansichten zwischen denen mit Buttons gewechselt werden kann (siehe Abbildung 4 und Abbildung 5).

Die Ansicht Tracker wird verwendet um Tätigkeiten aufzeichnen und zu bearbeiten und ist in Abbildung 4 dargestellt. Sobald Einträge aufgezeichnet wurden, werden sie in einer Liste dargestellt. Durch einen Doppelklick auf einen Eintrag kann dieser bearbeitet oder gelöscht werden. Es kann aber auch der geöffnete Eintrag als Grundlage zur Erstellung eines neuen Eintrags verwendet werden.

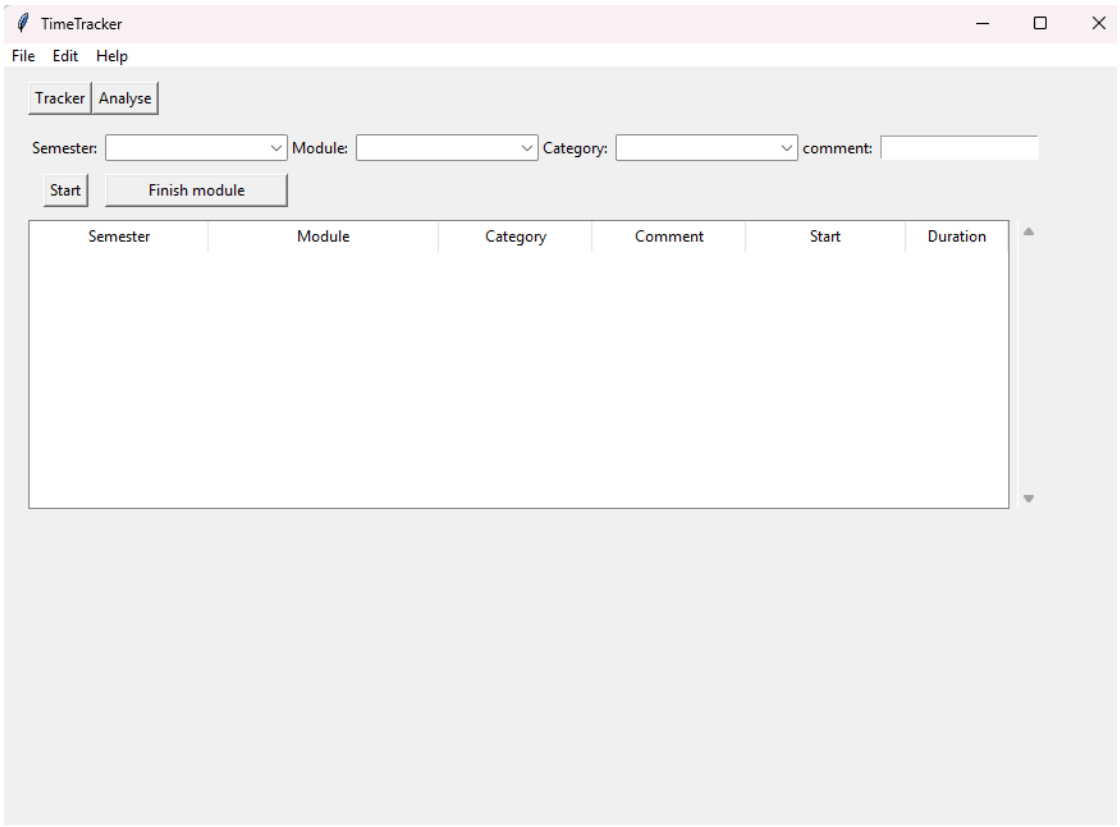

Abbildung 4: TimeTracker in der Tracking Ansicht

Die Ansicht Analyse wird verwendet um bereits aufgezeichnete Tätigkeiten zu analysieren und ist in Abbildung 5 dargestellt. Im linken Bereich ist ein Akkordeon das zur Navigation verwendet wird. Im rechten Bereich kann zwischen den beiden Diagrammtypen Burndown-Chart und Pie-Chart selektiert werden. Beim Öffnen der Ansicht werden immer die Daten des gesamten Studiums visualisiert (das kann auch durch erneutes klicken auf den Analyse Button erreicht werden). Für den Diagrammtyp wird der zuletzt ausgewählte und als Standard das Burndown-Chart verwendet.

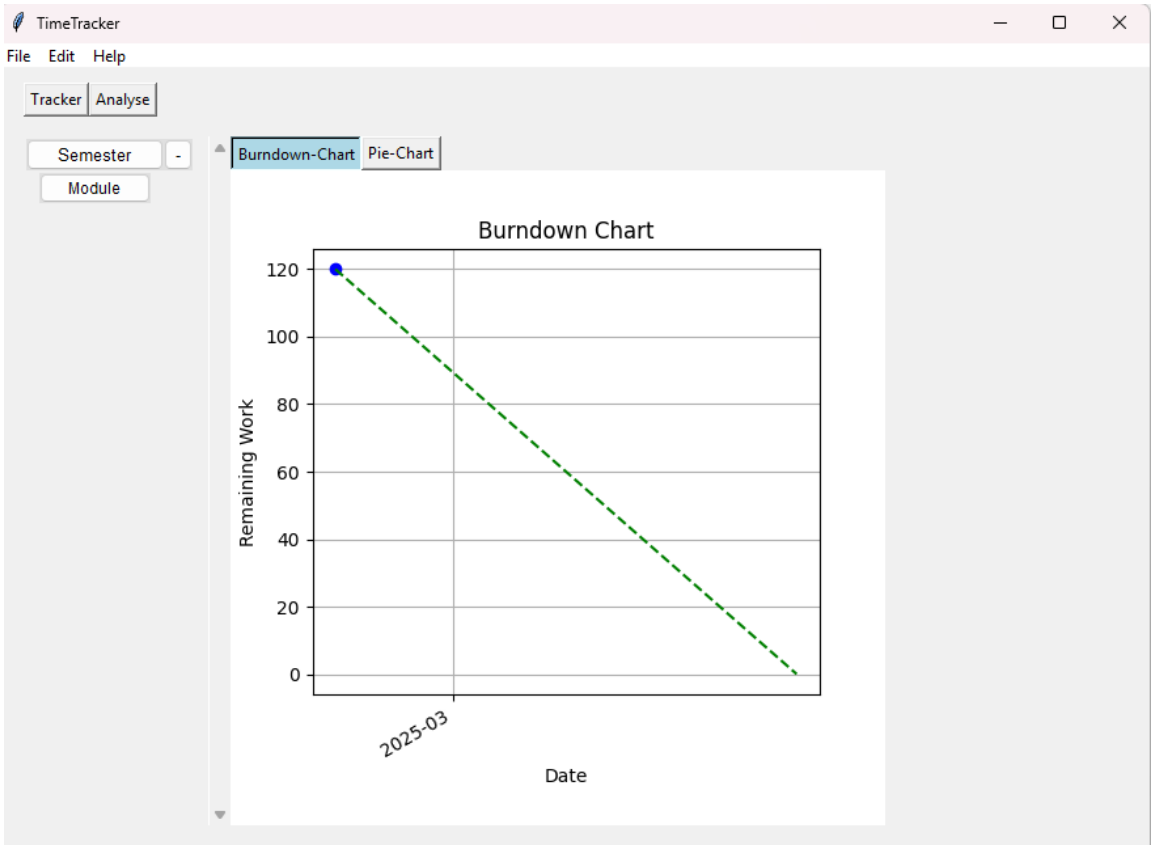

Abbildung 5: TimeTracker in der Analyse Ansicht

2 Anforderungsdokument

2.1 Stakeholder

Die Anwendung soll Studenten und Schüler dabei unterstützen ihre zeitlichen Aufwände zu dokumentieren und zu analysieren. Vor allem für das selbstorganisierte Lernen soll dadurch ein Mehrwert geschaffen werden. Personen, die alleine an einem Projekt arbeiten und ihre investierten Aufwände überwachen möchten, könnten ebenfalls den *Time Tracker* verwenden.

2.2 Anforderungen

2.2.1 Systemumfang und Kontext

Die Anwendung soll für einzelne Personen und nicht für Gruppen verwendbar sein. Dabei steht die Funktion auf einem Endgerät (Desktop) im Vordergrund. Die Daten können lokal gespeichert werden. Es soll aber ein Format verwendet werden, das in Datenbanken verwendet werden kann bzw. leicht exportierbar ist, damit eine (nicht parallele) Nutzung mit mehreren Endgeräten ermöglicht werden könnte. Der Systemumfang ist im Use-Case-Diagramm in Abbildung 6 visualisiert.

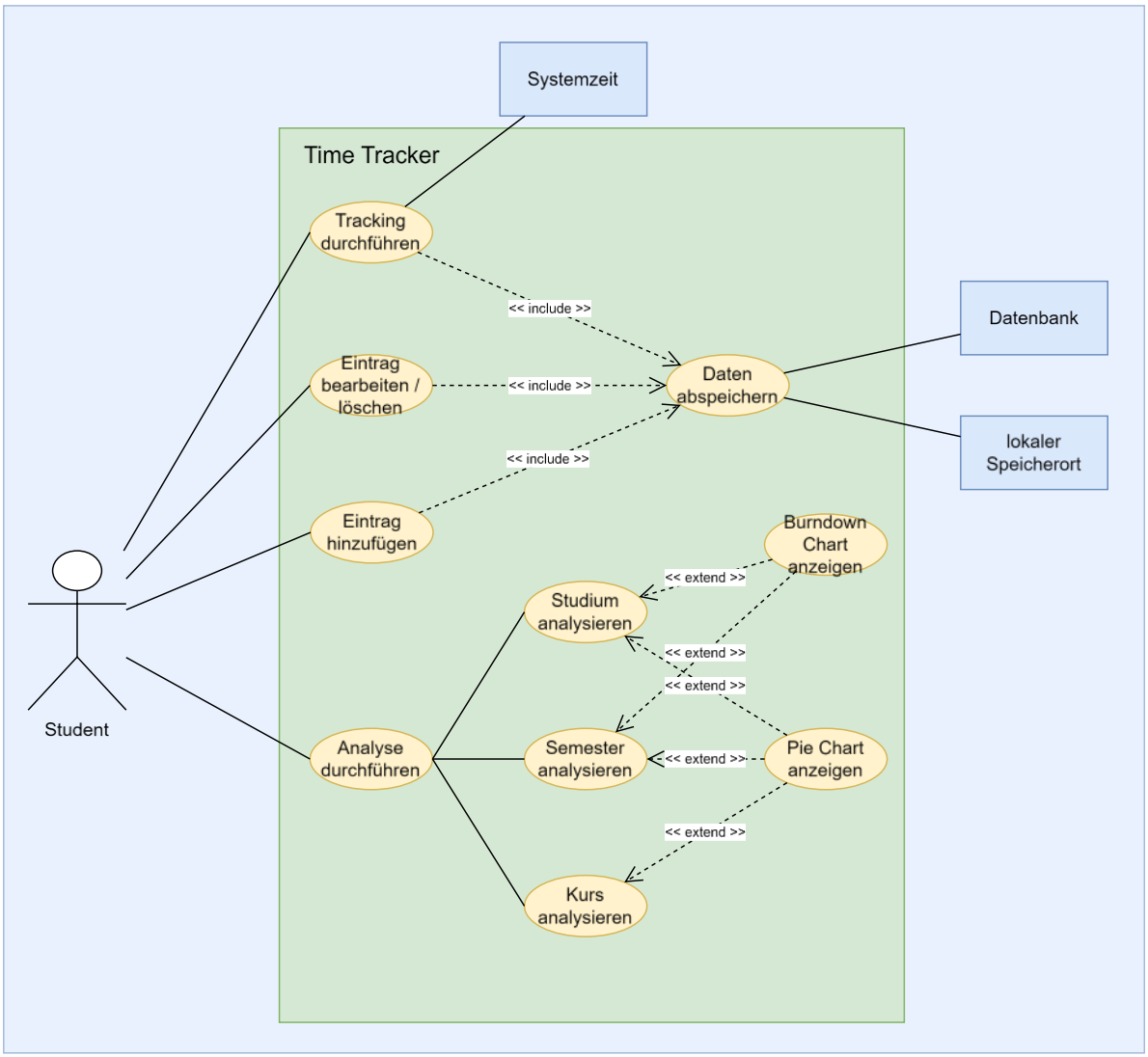

Abbildung 6: UML-Use-Case-Diagramm

2.2.2 Funktionale Anforderungen

1 Tracking (must)

Als Student möchte ich meine Aufwände live auf eine Tätigkeit buchen. Die Tätigkeit ist Teil eines Kurses, welcher wiederum einem Semester zugeordnet werden kann. Das Tracking erfolgt über die Erfassung von Zeitstempeln (Systemzeit) zum Start und Stopp des Trackings.

1.1 Tracking starten (must)

Als Student möchte ich das Tracking starten. Dazu wird mir die zuletzt gebuchte Aktivität bereits vorgeschlagen. Ich kann jedoch das Semester, den Kurs und die Tätigkeit wechseln. Es darf nicht möglich sein, mehrere Tätigkeiten parallel aufzuzeichnen.

1.2 Tracking stoppen (must)

Als Student möchte ich ein zuvor gestartetes Tracking stoppen.

2 Eintrag bearbeiten (must)

Als Student möchte ich nachträglich Aufzeichnungen bearbeiten, um mögliche Fehler zu korrigieren.

2.1 Bearbeiten (must)

Als Student möchte ich sämtliche Parameter einer beendeten Aufzeichnung bearbeiten.

2.2 Löschen (must)

Als Student möchte ich fehlerhafte abgeschlossene Aufzeichnungen löschen.

2.3 Hinzufügen (must)

Als Student möchte ich eine Aufzeichnung nachträglich hinzufügen, falls ich vergessen habe die Aufzeichnung zu starten oder an einem anderen Ort tätig war, an dem ich keinen Zugriff auf den TimeTracker hatte.

3 Analyse (must)

Als Student möchte ich meine aufgezeichneten Tätigkeiten in unterschiedlichen Detaillierungsgraden analysieren. Mithilfe der Analyse möchte ich sehen, wieviel Zeit ich für die Abarbeitung der einzelnen Tätigkeiten investiert habe. Die Analyse soll die Möglichkeit bieten die tatsächlich benötigte Zeit und die geschätzte Vorgabe (30h / ECTS) in einem zeitlichen Verlauf gegenüberzustellen.

3.1 Studium (must)

Als Student benötige ich Visualisierungen, die mir den Fortschritt im Studium darstellen.

3.1.1 Burndown Chart (must)

Im Burndown Chart kann der Abarbeitungsgrad in Prozent dargestellt werden. Als Metrik soll die Anzahl abgeschlossener ECTS verwendet werden. Die Ideallinie wird durch den Start und das Ende des Studiums definiert.

3.1.2 Pie Chart (must)

Im Pie Chart können die Aufwände der einzelnen Semester gegenübergestellt werden.

3.2 Semester (must)

Als Student benötige ich Visualisierungen, die mir den Fortschritt im aktuellen Semester darstellen.

3.2.1 Burndown Chart (must)

Im Burndown Chart kann der Abarbeitungsgrad in Prozent dargestellt werden. Als Metrik soll die Anzahl abgeschlossener ECTS verwendet werden. Die Ideallinie wird durch den Start und das Ende des Semesters, sowie die Anzahl der geplanten ECTS definiert.

3.2.2 Pie Chart (must)

Im Pie Chart können die Aufwände der einzelnen Module innerhalb eines Semesters gegenübergestellt werden.

3.3 Kurs (must)

Als Student benötige ich Visualisierungen, die mir den Fortschritt im aktuellen Kurs darstellen.

3.3.1 Pie Chart (must)

Im Pie Chart können die Aufwände für die einzelnen Tätigkeiten in einem Kurs gegenübergestellt werden.

2.2.3 Nicht-funktionale Anforderungen

1 Verhalten bei Systemausfall

Um einem Verlust der Daten vorzubeugen muss die Anwendung sämtliche Daten unmittelbar nach der Eingabe sichern.

2 Benutzerfreundlichkeit

2.1 Als Student möchte ich, dass die Anwendung eine intuitive Oberfläche hat, damit ich meine Tätigkeiten ohne Schulung nachverfolgen kann.

2.2 Als Student möchte ich, dass die Anwendung innerhalb von 2 Sekunden startet, damit ich nicht durch das Tool aufgehalten werden.

3 Skalierbarkeit

Als Softwareentwickler möchte ich neue Funktionen, z.B. neue Diagramme für die Analyse, mit minimalen Änderungen hinzufügen, damit die Anwendung auch für zukünftige Anforderungen flexibel bleibt.

4 Wartbarkeit

4.1 Als Softwareentwickler möchte ich eine klar strukturierte Codebasis mit einer durchgängigen Dokumentation, damit ich die Anwendung leicht erweitern kann.

4.2 Als Softwareentwickler möchte ich automatisierte Tests für die Geschäftslogik haben, damit ich sicherstellen kann, dass Änderungen keine bestehenden Funktionen beeinträchtigen.

5 Portabilität

5.1 Als Student möchte ich die Anwendung auf Windows, macOS oder Linux verwenden können, damit ich plattformunabhängig arbeiten kann.

5.2 Als Student möchte ich die Anwendung ohne komplizierte Installation nutzen können, damit ich sofort starten kann.

3 Spezifikationsdokument

3.1 Datenmodell

Die Daten werden mithilfe von Objekten hierarchisch verwaltet (siehe Abbildung 7). Auf der untersten Ebene existiert ein Entry-Objekt, das einen Start- und Stoppzeitpunkt, eine Kategorie und einen optionalen Kommentar hat. Die nächsthöhere Ebene wird durch ein Modul abgebildet. Dieses Modul hält eine Liste von Entry-Objekten und hat selbst ebenfalls einen Start- und Stoppzeitpunkt sowie ein geplantes Ende, die Anzahl an ECTS und einen Namen. Dasselbe wiederholt sich auf der Ebene der Semester. Jedes Semester hält wiederum eine Liste an Modul-Objekten. Auf der obersten Ebene befindet sich das Study-Objekt, das eine Liste an Semester-Objekten enthält und somit die Gesamte Hierarchie des Datenmodells abbildet. Das Study Objekt existiert nur einmal pro Projekt und dient als Wurzel für den Zugriff auf die Daten. Über die gesamte Hierarchie hinweg stellen die Objekte unterschiedliche Funktionen zur Verfügung, um auf die Daten und Informationen über die Hierarchie zuzugreifen.

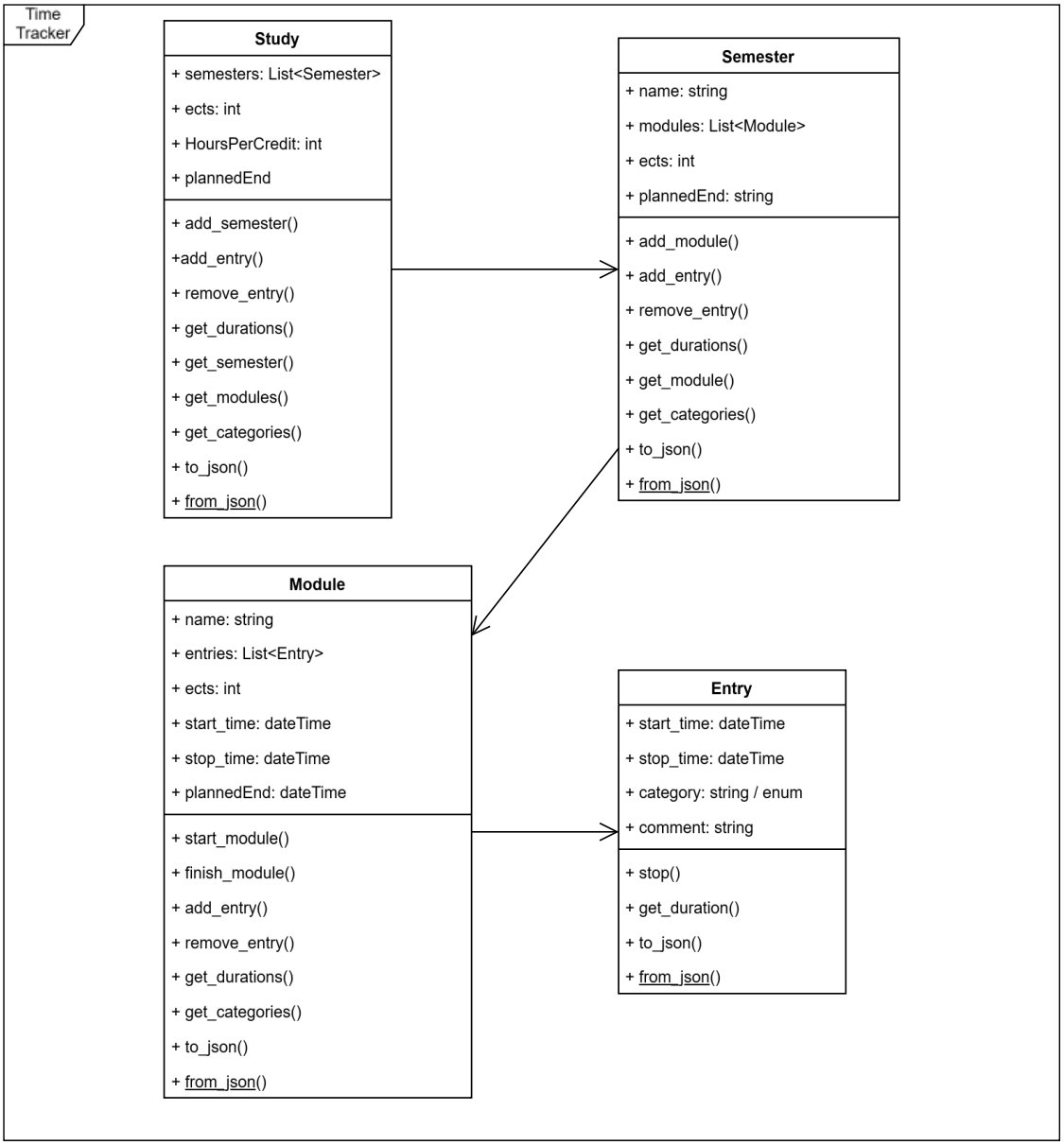

Abbildung 7: UML-Klassendiagramm Datenmodell

3.2 Geschäftsprozesse

Im Folgenden werden die Geschäftsprozesse der Applikation beschrieben. Dies sind das Aufzeichnen von Tätigkeiten, das Bearbeiten von bereits aufgezeichneten Tätigkeiten, das Hinzufügen von neuen Tätigkeiten sowie die Visualisierung der Daten.

3.2.1 Aufzeichnen von Tätigkeiten

Der wichtigste dieser Prozesse, das Aufzeichnen von Tätigkeiten, ist in der Abbildung 8 in Form eines UML-Aktivitätsdiagramms Visualisiert. Es wird auf Basis eines Semester- und Modulnamens, einer Kategorie und eines Kommentars eine Aufzeichnung gestartet.

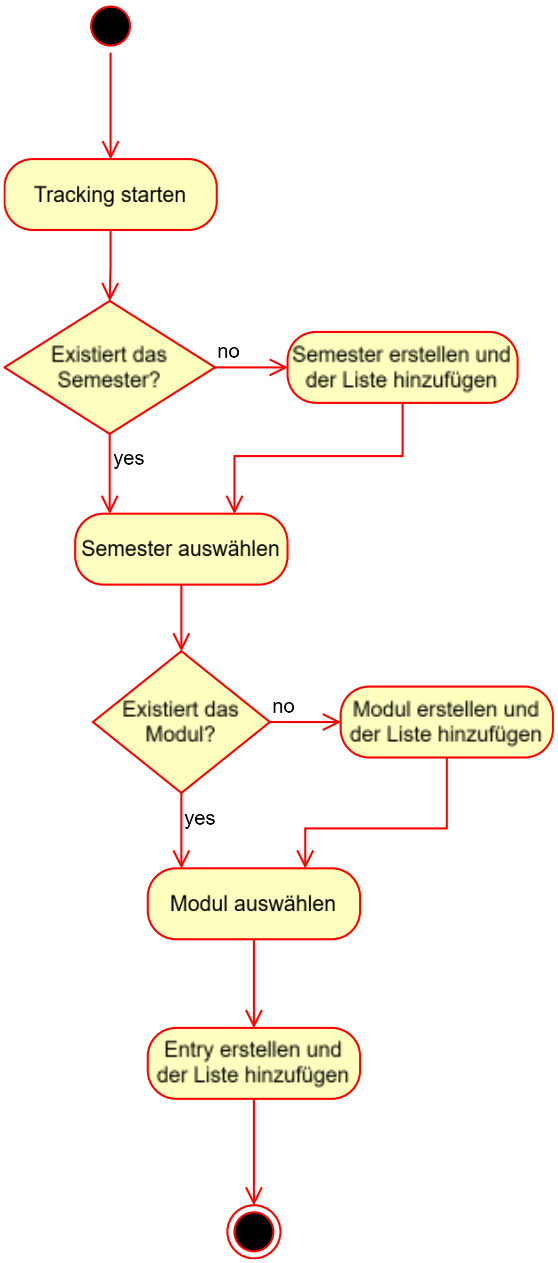

Abbildung 8: UML-Aktivitätsdiagramm des Geschäftsprozesses Tracking starten

3.2.2 Bearbeiten von bereits Aufgezeichneten Tätigkeiten

Durch selektieren einer bereits abgeschlossenen Aufzeichnung wird ein Dialog geöffnet, der es ermöglicht, sämtliche Daten der Aufzeichnung zu bearbeiten. Dazu wird ein neuer Eintrag erstellt und der alte Eintrag entfernt.

3.2.3 Hinzufügen neuer Tätigkeiten

Durch selektieren einer bereits abgeschlossenen Aufzeichnung wird ein Dialog geöffnet, der es ermöglicht, eine bearbeitete Version des selektierten Eintrags abzuspeichern.

3.2.4 Visualisieren der Daten

Aufgrund der Auswahl in der Navigation werden die Daten für das jeweilige Objekt (Studium, Semester, Modul) so aufbereitet, dass sie im ausgewählten Diagrammtyp dargestellt werden können. Dabei kann es vorkommen, dass der gewählte Diagrammtyp für die selektierten Ebene nicht verfügbar ist.

3.3 Geschäftsregeln

1. Auf allen Ebenen muss der Stoppzeitpunkt stets zeitlich nach dem Startzeitpunkt liegen.
2. Es darf nicht möglich sein zwei Aktivitäten gleichzeitig aufzuzeichnen.
3. Es dürfen nur Diagramme angezeigt werden, für die die notwendigen Daten zur Verfügung stehen.
4. Automatisiertes Speichern stellt sicher, dass keine Daten verloren gehen.

3.4 Systemschnittstellen

1. Systemzeit:
   Für das Aufzeichnen der Tätigkeiten wird die Systemzeit zur Generierung der Zeitstempel verwendet. Die erfolgt mithilfe von DateTime-Objekten.
2. Speichern der Daten:
   Es wird das Modul json verwendet um das gesamte Datenmodell serialisiert zu exportieren. Jedes Objekt im Datenmodell muss dazu eine JSON-Repräsentation von sich bereitstellen. Dabei entstehen JSON-Dateien, welche auch mit einem anderen Editor bearbeitet werden könnten.
3. Speichern der Einstellungen:
   Die Projekteinstellungen werden ebenfalls im JSON-Format mit dem Modul json gespeichert.

3.5 Benutzerschnittstellen

Das Grafische User Interface (GUI) soll zwei Ansichten zur Verfügung stellen. Eine zum Aufzeichnen der Aktivitäten und eine zur Analyse bereits aufgezeichneter Daten. Im oberen Bereich soll auf beiden Ansichten ein Menü für Dateioperationen und Einstellungen angeordnet sein. Zwischen den beiden Ansichten kann mittels Buttons gewechselt werden.

In Abbildung 9 ist eine Skizze für die Tracker Ansicht dargestellt. Die Ansicht bietet Comboboxen zur Auswahl des Semesters, Moduls und der Tätigkeit. Über eine Textbox kann ein Kommentar eingetragen werden. Mit einem Start/Stopp Button kann eine Aufzeichnung gestartet und beendet werden. Bei einem aktiven Tracking wird die aktuelle Dauer der Aufzeichnung in einem Label dargestellt. Unter den Bedienelementen wird eine Tabelle mit den Einträgen angezeigt. Die

Informationen in der Tabelle werden mithilfe der Bedienelemente für Semester, Modul, Tätigkeit und Kommentar gefiltert.

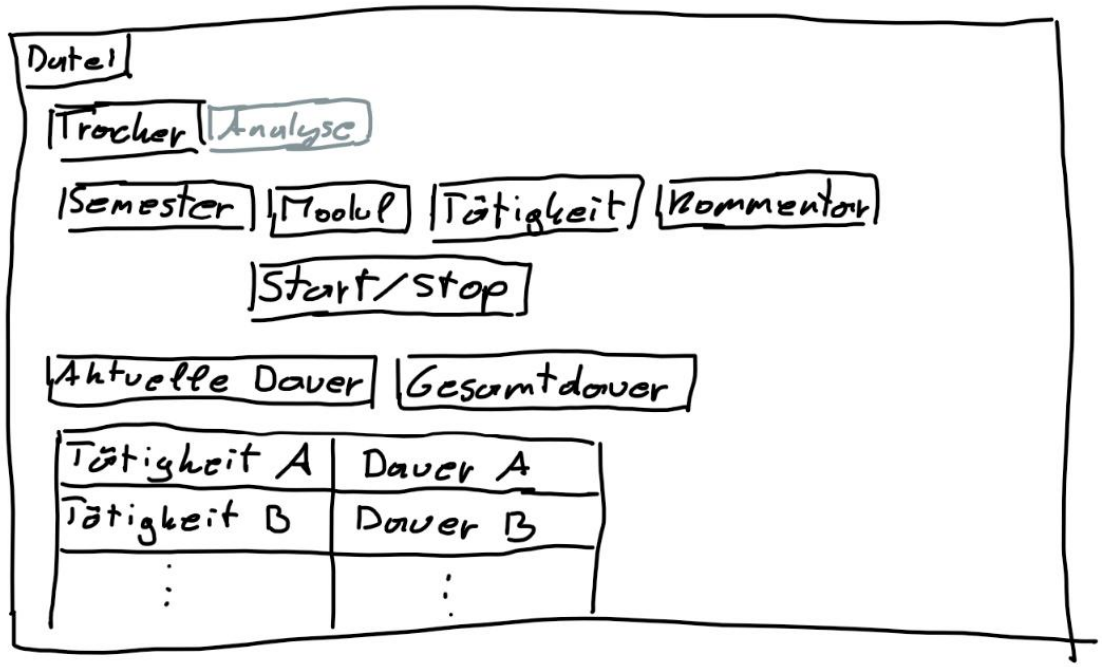

Abbildung 9: User Interface Tracking

Die zweite Ansicht zur Analyse der Daten ist in den folgenden beiden Abbildungen skizziert. Es soll eine Auswahl in der Hierarchie mit einem Akkordeon ermöglicht werden, bei dem einzelne Hierarchieebenen ein- bzw. ausgeblendet werden können. Die im Akkordeon selektierte Ebene wird in einem daneben liegenden Fenster mit dem ausgewählten Diagrammtyp visualisiert. In Abbildung 10 ist die Darstellung eines Burndown-Charts skizziert und in Abbildung 11 ist die zweite Visualisierungsart in Form eines Pie-Charts dargestellt.

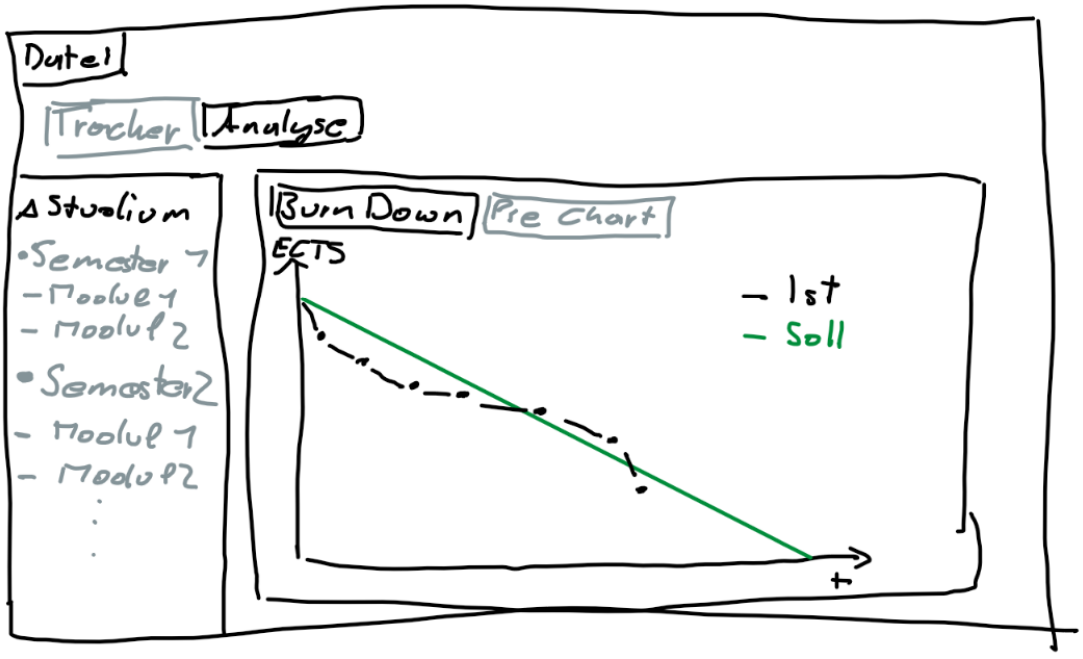

Abbildung 10: User Interface Analyse mit einem Burndown-Chart

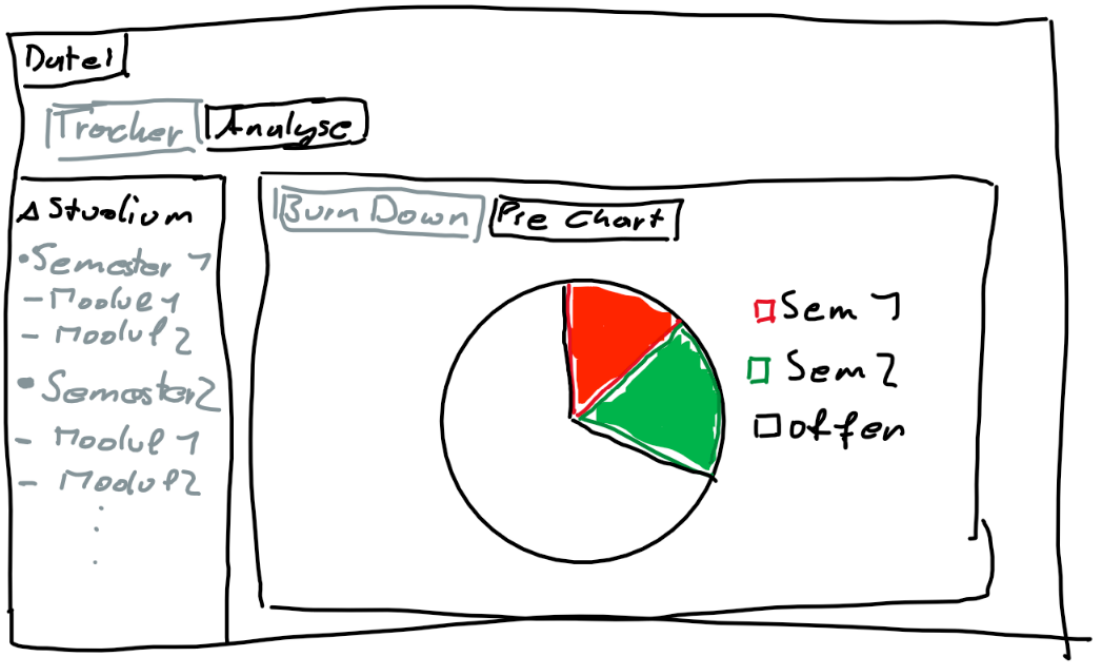

Abbildung 11: User Interface Analyse mit einem Pie-Chart

4 Architekturdokument

4.1 Technologieübersicht

Als Programmiersprache wird Python und als Entwicklungsumgebung Visual Studio Code verwendet. Die Versionskontrolle erfolgt mit GitHub. Zur Verwaltung der Pakete wird eine virtuelle Umgebung mittels Anaconda erstellt. Dadurch wird es ermöglicht eine Liste der benötigten Pakete in das Repository zu exportieren um die Installation auf einem anderen System zu vereinfachen. Für die Visualisierung werden TKinter und die MatplotLib verwendet. Das Importieren und Exportieren von Daten erfolgt im JSON-Format. Dazu wird das json Modul verwendet. Das Unittest-Framework wird für automatisierte Tests benötigt.

4.2 Architekturübersicht

Es wird eine lineare MVC-Architektur verwendet. Dabei gibt es drei Schichten: View, Controller und Model, die nachfolgend erläutert werden.

4.2.1 Model

Das Model ist für das Datenmanagement, Berechnungen und die Geschäftsregeln verantwortlich. Es repräsentiert die vollständigen Daten und durch exportieren und importieren des Models können alle Daten gespeichert und geladen werden.

4.2.2 Controller

Die Controller Schicht vermittelt zwischen dem Model und der View. Sie verarbeitet die Benutzeraufgaben und delegiert Aufgaben an das Model. Zusätzlich stellt der Controller aggregierte Daten für die Visualisierung bereit.

4.2.3 View

Die View präsentiert die Daten und reagiert auf Benutzerinteraktionen.

Die Architektur ermöglicht eine klare Trennung der Verantwortlichkeiten. Die Visualisierung ist somit losgelöst von den Daten und der Geschäftslogik. Das führt dazu, dass das User Interface unkompliziert angepasst werden kann. Wenn mehrere Entwickler an einem Projekt arbeiten, könnte einer das User Interface entwickeln, ohne sich um die dahinterliegenden Prozesse kümmern zu müssen. Umgekehrt kann sich ein anderer Entwickler auf die Prozesse konzentrieren, ohne sich Gedanken über die Visualisierung der Applikation machen zu müssen.

4.3 Struktur

Das Model beinhaltet die Klassen Study, Semester, Module, Entry, welche bereits in Abbildung 7 dargestellt sind. Zusätzlich erhalten die Klassen Methoden die Berechnungen durchführen, um die Daten bereitzustellen.

Die TimeTracker-Klasse bildet den Controller, der die Benutzereingaben verarbeitet und die notwendigen Daten für die View zur Verfügung stellt.

Die View wird durch die Klassen TimeTrackerGUI, Chart, DateTimeFrame und Accordion gebildet. Chart, DateTimeFrame und Accordion sind dabei jeweils Bedienelemente, die in eigene Klassen ausgelagert sind. TimeTrackerGUI ist das gesamte Userinterface, welches die Benutzereingaben an den Controller weiterleitet.

In Abbildung 12 sind die Hauptkomponenten in einem UML-Klassendiagramm dargestellt. In der obersten Ebene sind die Komponenten der MVC-Architektur (View, Controller und Model) dargestellt. Dabei ist nur die Klasse Study als Wurzel des Models aufgeführt. Es ist ersichtlich, dass sämtliche Zugriffe vom User Interface (TimeTrackerGUI) auf die Daten (Study) über den Controller (TimeTracker) führen.

Die Diagramme werden über eine Factory erzeugt. Die Klasse ChartFactory erstellt Instanzen der PieChart- und BurndownChart-Klassen. Der Controller generiert die notwendigen Daten und erstellt durch den Aufruf der ChartFactory eine Chart-Instanz. Von der View wird nur auf die abstrakte Klasse Chart, von welcher PieChart und BurndownChart erben, zugegriffen um die Diagramme darzustellen.

Um Einstellungen, wie z.B. der Dateiname des letzten Trackings, zu speichern verwendet der Controller die Settings-Klasse. Die Klasse verwaltet die Einstellungen in einem Dictionary und die Einstellungen können als JSON- Datei gespeichert und von einer JSON- Datei geladen werden.

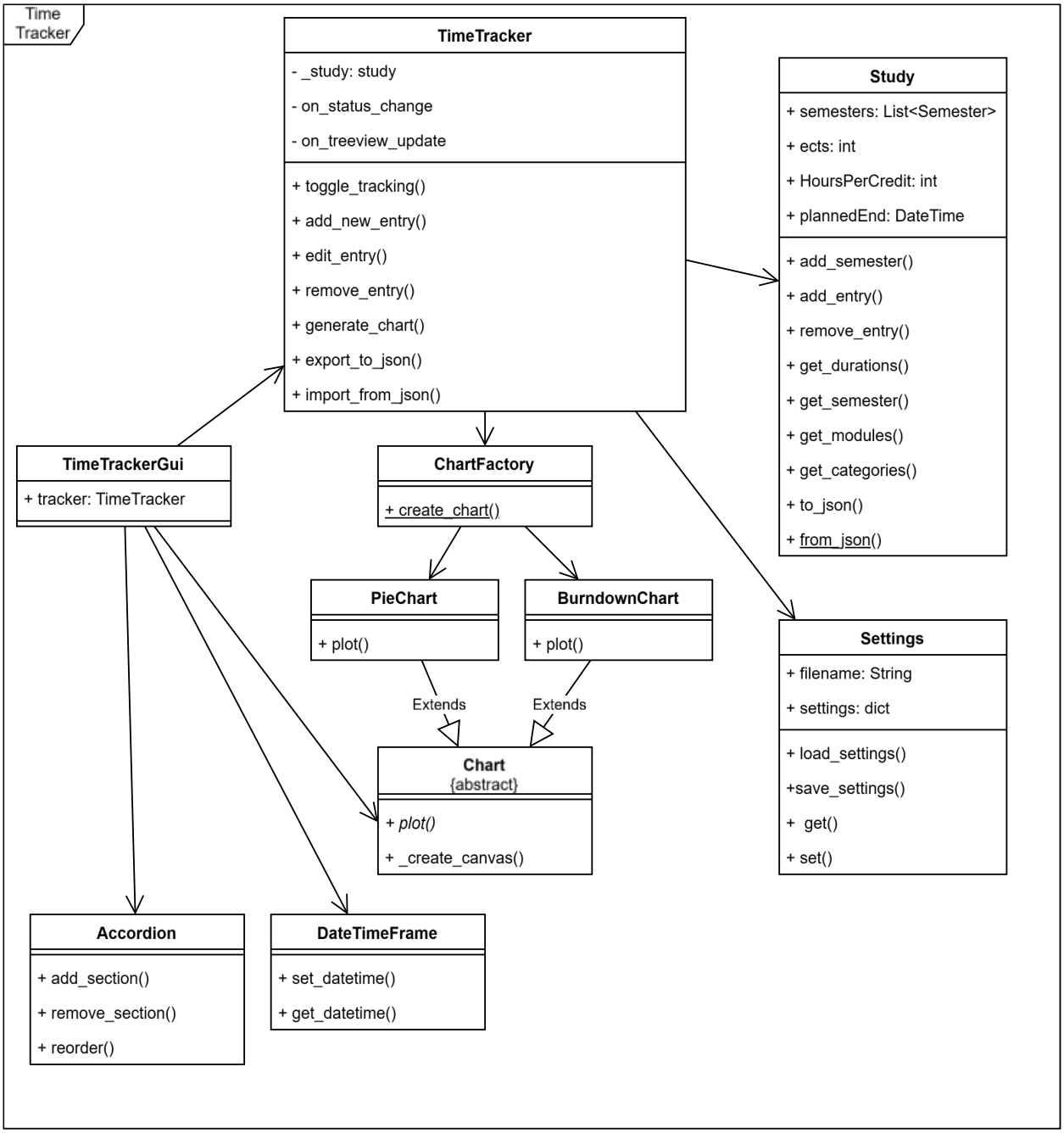

Abbildung 12: UML-Klassendiagramm der Hauptkomponenten

4.4 Verhalten

In Abbildung 13 ist der Ablauf zum Erstellen eines Burndown-Charts in Form eines UML-Sequenzdiagramms dargestellt. Die Sequenz wird durch eine Benutzereingabe gestartet. Dies kann durch den Wechsel des Diagramm-Typs oder durch den Wechsel der zu visualisierenden Daten erfolgen. Die View (TimeTrackerGUI) leitet die Anfrage an den Controller (TimeTracker) weiter, welcher die Daten für den entsprechenden Diagrammtyp generiert und über die ChartFactory eine Instanz des Diagramms (BurndownChart) erstellt. Diese Instanz wird bis zur View zurückgereicht. Die View ruft die Plot-Funktion des Diagramms auf und übergibt ihr das Frame, in welchem die Visualisierung dargestellt werden soll. Das Diagramm stellt dann die Visualisierung auf dem gewünschten Frame dar.

Der Ablauf zur Erstellung eines Pie-Charts unterscheidet sich nur darin, dass ein anderer Diagrammtyp an den Controller übergeben wird. Über die ChartFactory wird dann eine Instanz des PieCharts erstellt, auf welche von der View auf dieselbe Art und Weise zugegriffen wird.

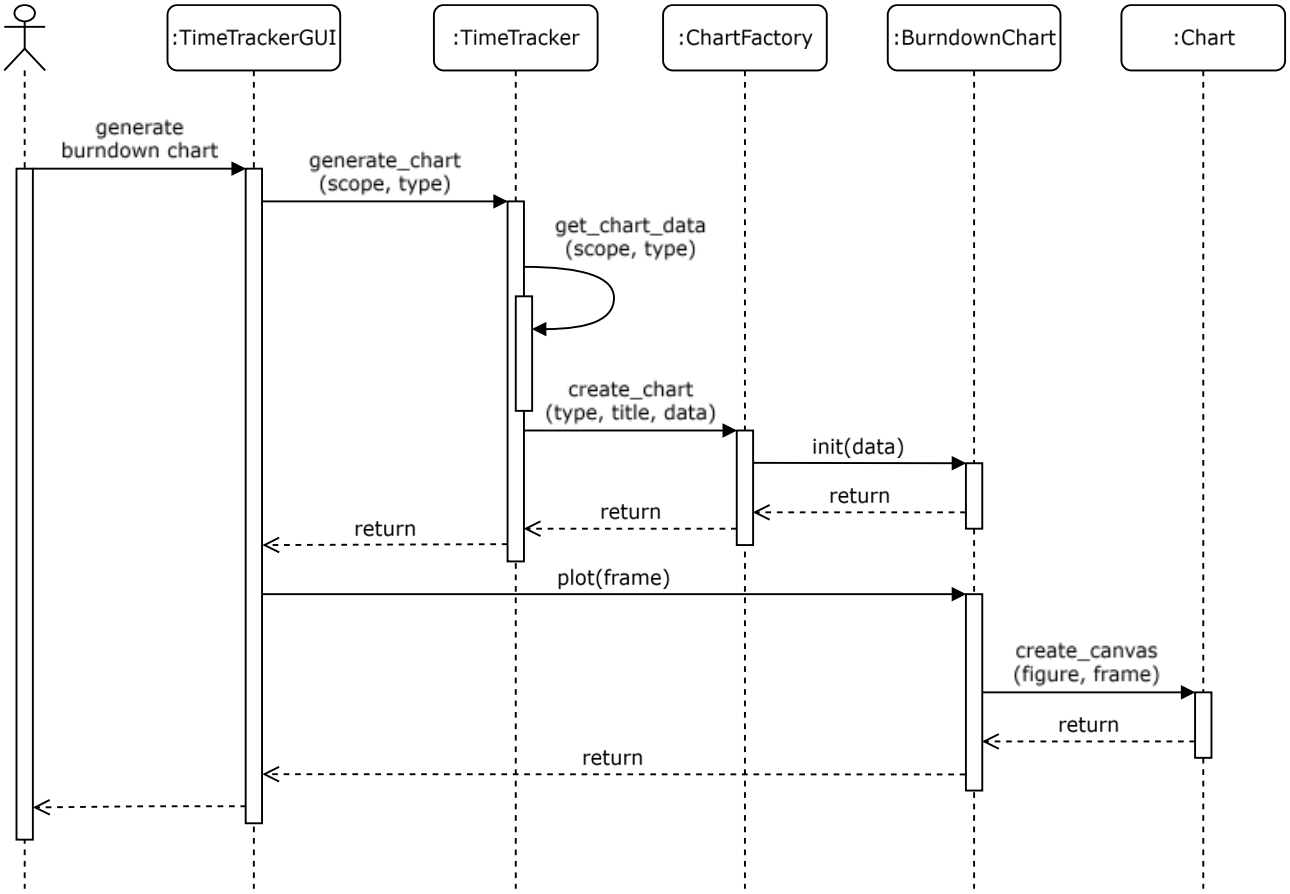

Abbildung 13: UML-Sequenzdiagramm Burndown-Chart erstellen

5 Testdokument

5.1 Teststrategie

Durch die Erstellung von automatisierten Unit-Tests während der Entwicklung können Probleme früh erkannt werden. Es sollen automatisierte Tests für sämtliche Klassen entwickelt werden. Das User Interface bildet dabei eine Ausnahme die manuell getestet wird. Da das User Interface aufgrund der MVC-Architektur keine Geschäftslogik enthält, wird davon ausgegangen, dass etwaige Fehler nur geringe Auswirkungen auf das Projekt haben und die notwendigen Fehlerbehebungen einen geringen Komplexitätsgrad aufweisen.

Die Testabdeckung wird über eine Code-Coverage ausgewertet. Ziel dabei ist es folgende Testabdeckungen zu erreichen. Für das User Interface sollen wo möglich einzelne Tests erstellt werden, die die Grundfunktion der Anwendung prüfen. Die Benutzerdefinierten Bedienelemente sollen eine Testabdeckung von über 80% aufweisen, da sie auch in anderen Projekten verwendbar sein sollen und deshalb die selbstständige Funktion sichergestellt sein muss. Alle anderen Klassen, die die Logik enthalten oder die Daten zur Verfügung stellen, sollen eine Testabdeckung von über 95% aufweisen.

Dadurch, dass der Controller mit automatisierten Tests getestet wird ist auch die Stufe der Integrationstests größtenteils abgedeckt, weil der Controller sowohl mit der View als auch mit dem Model interagieren muss.

Die Systemtests werden am User Interface durchgeführt. Dabei werden alle Benutzerinteraktionen getestet. Es soll speziell Wert auf die Benutzerfreundlichkeit und auf die Performance der Anwendung geachtet werden. Falls spürbare Performancedefizite auftreten soll der Code durch gezielte Profilierung untersucht werden.

5.2 Testprotokoll

ID	**Testfall**	
001	**Anwendung starten**	
	Vorbedingungen	Eine Ausführbare Version der Anwendung (Python Umgebung oder .exe) liegt vor
	Handlung	• Anwendung starten • Neues Studium anlegen • Studium konfigurieren
	Erwartetes Ergebnis	• Es wird ein Dialog zum Starten eines neuen Studiums angezeigt. • Es wird der Dialog zum Auswählen des Speicherorts angezeigt. • Anwendung startet und es könnten Tätigkeiten aufgezeichnet werden.
	Tatsächliches Ergebnis	Neues Studium kann angelegt und gespeichert werden.
002	**Tracking starten**	

	Vorbedingungen	frisch installierte Anwendung starten
	Handlung	• Semester, Modul, und Kategorie wählen • Tracking starten
	Erwartetes Ergebnis	• Tracking wird gestartet • Eintrag wird in der Liste angelegt • Die aktuelle Dauer wird angezeigt.
	Tatsächliches Ergebnis	Das Tracking startet und die Informationen werden angezeigt.
003	**Tracking stoppen**	
	Vorbedingungen	Testfall 002
	Handlung	Tracking stoppen
	Erwartetes Ergebnis	• Tracking wird gestoppt • Die Dauer des Listeneintrags wird angepasst • Die aktuelle Dauer wird entfernt
	Tatsächliches Ergebnis	Das Tracking stoppt und die Informationen werden aktualisiert.
004	**Eintrag bearbeiten**	
	Vorbedingungen	Ein bereits aufgezeichneter Eintrag ist in der Tabelle
	Handlung	• Eintrag durch Doppelklick öffnen • Alle Felder bearbeiten • Speichern • Dialog schließen
	Erwartetes Ergebnis	• Bearbeitungsdialog wird angezeigt • Dialog bleibt nach dem Speichern geöffnet • Eintrag hat in der Tabelle andere Werte (möglicherweise muss die Filterung aktualisiert werden)
	Tatsächliches Ergebnis	Der Dialog wird geöffnet und der Eintrag kann bearbeitet werden. Das Öffnen ist sehr langsam (dauert einige Sekunden). Das Fenster wird auf dem Hauptbildschirm oben links dargestellt.
005	**Neuen Eintrag hinzufügen**	
	Vorbedingungen	Ein bereits aufgezeichneter Eintrag ist in der Tabelle
	Handlung	• Eintrag durch Doppelklick öffnen • Felder bearbeiten • Als neuen Eintrag speichern • Dialog schließen
	Erwartetes Ergebnis	• Bearbeitungsdialog wird angezeigt • Dialog bleibt nach dem Speichern geöffnet • Ein neuer Eintrag wurde zur Tabelle hinzugefügt (möglicherweise muss die Filterung aktualisiert werden) • Die Comboboxen enthalten auch die hinzugefügten Daten im Dropdown

	Tatsächliches Ergebnis	Der Dialog wird geöffnet und der bearbeitete Eintrag kann als neuer Eintrag gespeichert werden. Auch hier ist das Öffnen des Dialogs langsam.
006	**Eintrag löschen**	
	Vorbedingungen	Ein bereits aufgezeichneter Eintrag ist in der Tabelle
	Handlung	• Eintrag durch Doppelklick öffnen • Löschen • Dialog schließen
	Erwartetes Ergebnis	• Bearbeitungsdialog wird angezeigt • Dialog bleibt nach dem Löschen geöffnet • Eintrag wurde aus der Tabelle entfernt
	Tatsächliches Ergebnis	Der Dialog wird geöffnet und der Eintrag kann gelöscht werden. Auch hier ist das Öffnen des Dialogs langsam
007	**Modul abschließen**	
	Vorbedingungen	Es wurde bereits eine Tätigkeit aufgezeichnet. Die Tätigkeit ist über die Comboboxen ausgewählt.
	Handlung	• Modul durch Klick auf den Button finish Module beenden • Tätigkeit durch Doppelklick in der Tabelle auswählen
	Erwartetes Ergebnis	• Das Modul wird beendet. • Im geöffneten Dialog ist der Endzeitpunkt des Moduls eingetragen.
	Tatsächliches Ergebnis	Das Modul wird beendet.
008	**Studium bearbeiten**	
	Vorbedingungen	Anwendung ist gestartet
	Handlung	• Mit Edit -> Edit Study das Studium bearbeiten
	Erwartetes Ergebnis	• Es wird ein Dialog geöffnet - ECTS können geändert werden - Das Ende kann bearbeitet werden
	Tatsächliches Ergebnis	Der Titel im Dialog ist falsch. Die Werte können bearbeitet werden. Das Fenster wird auf dem Hauptbildschirm oben links dargestellt.
009	**Semester bearbeiten**	
	Vorbedingungen	Es sind Daten mit mindestens zwei unterschiedlichen Semestern verfügbar.
	Handlung	• Mit Edit -> Edit Semester die Semester bearbeiten
	Erwartetes Ergebnis	• Es wird ein Dialog geöffnet - Alle Semester werden angezeigt - Die Felder können bearbeitet werden.

	Tatsächliches Ergebnis	Die Felder können bearbeitet werden. Das Fenster wird auf dem Hauptbildschirm oben links dargestellt.
010	**Studium Burndown-Chart anzeigen**	
	Vorbedingungen	Es wurden bereits mehrere Tätigkeiten aufgezeichnet. Es sind mindestens 2 unterschiedliche Semester in den Daten vorhanden. Mindestens ein Semester hat abgeschlossene Module unter sich.
	Handlung	• Analyze View öffnen • Burndown-Chart auswählen
	Erwartetes Ergebnis	• Es wird standardmäßig der zuletzt verwendete Diagrammtyp angezeigt. Falls es das erste Diagramm ist, das angezeigt wird, wird das Burndown-Chart angezeigt. • Das Akkordion zeigt die eingefügten Semester an. • Das Burndown-Chart wird dargestellt. - Alle Beschriftungen sind lesbar - Das Ende der Ideallinie entspricht dem geplanten Studiumsende. - Für abgeschlossene Module wird ein Datenpunkt angezeigt.
	Tatsächliches Ergebnis	Die Daten werden korrekt dargestellt.
011	**Studium Pie-Chart anzeigen**	
	Vorbedingungen	Testfall 010
	Handlung	• Pie-Chart auswählen
	Erwartetes Ergebnis	• Es wird das Pie-Chart angezeigt - Es sind alle Daten korrekt dargestellt - Alle Beschriftungen sind lesbar
	Tatsächliches Ergebnis	Die Daten werden korrekt dargestellt.
012	**Semester Burndown-Chart anzeigen**	
	Vorbedingungen	Es wurden bereits mehrere Tätigkeiten aufgezeichnet. Es ist mindestens ein abgeschlossenes und ein nicht abgeschlossenes Modul innerhalb eines Semesters in den Daten vorhanden.
	Handlung	• Analyze View öffnen • Burndown-Chart auswählen • Das Semester im Akkordion auswählen
	Erwartetes Ergebnis	• Es wird standardmäßig der zuletzt verwendete Diagrammtyp angezeigt. • Das Akkordion zeigt die eingefügten Semester an. - Durch Aufklappen werden die eingefügten Module angezeigt. • Das Burndown-Chart wird dargestellt. - Die Ideallinie verbindet die Anzahl ECTS und das geplante Ende des Semesters - Für die abgeschlossenen Module sind Datenpunkte sichtbar. - Alle Beschriftungen sind lesbar

	Tatsächliches Ergebnis	Die Daten werden korrekt dargestellt.
013	**Semester Pie-Chart anzeigen**	
	Vorbedingungen	Testfall 012
	Handlung	• Pie-Chart auswählen
	Erwartetes Ergebnis	• Es wird das Pie-Chart angezeigt - Es sind alle Daten korrekt dargestellt - Alle Beschriftungen sind lesbar
	Tatsächliches Ergebnis	Die Daten werden korrekt dargestellt.
014	**Modul Burndown-Chart anzeigen**	
	Vorbedingungen	Es wurden bereits mehrere Tätigkeiten aufgezeichnet. Es sind mindestens 2 unterschiedliche Kategorien innerhalb eines Moduls in den Daten vorhanden.
	Handlung	• Analyze View öffnen • Burndown-Chart auswählen • Das Semester im Akkordion ausklappen • Das Modul im Akkordion auswählen
	Erwartetes Ergebnis	• Es wird standardmäßig das Pie-Chart angezeigt - Die eingefügten Semester sind dort sichtbar • Das Akkordion zeigt die eingefügten Semester an. - Durch Aufklappen werden die eingefügten Module angezeigt • Es wird eine Fehlermeldung angezeigt, dass die Daten für diesen Diagrammtyp nicht verfügbar sind.
	Tatsächliches Ergebnis	Tippfehler in der Fehlermeldung.
015	**Modul Pie-Chart anzeigen**	
	Vorbedingungen	Testfall 014
	Handlung	• Pie-Chart auswählen
	Erwartetes Ergebnis	• Es wird das Pie-Chart angezeigt - Es sind alle Daten korrekt dargestellt - Alle Beschriftungen sind lesbar
	Tatsächliches Ergebnis	Die Daten werden korrekt dargestellt.
016	**Studium speichern**	
	Vorbedingungen	Die Anwendung läuft und es wurden Tätigkeiten aufgezeichnet
	Handlung	• Mit File -> Save as an einem Dateipfad speichern. • Einen zusätzlichen Eintrag hinzufügen • Mit File -> Save as mit einem anderen Dateinamen speichern
	Erwartetes Ergebnis	• Am initial ausgewählten Speicherort ist die Datei immer noch vorhanden.

		• Am ausgewählten Speicherort ist die zuvor gespeicherte Datei vorhanden.
	Tatsächliches Ergebnis	Es sind beide Dateien vorhanden.
017	**Studium öffnen**	
	Vorbedingungen	Testfall 016
	Handlung	• Mit File -> Open die initial erstellte Datei öffnen • Mit File -> Open die in Testfall 016 gespeicherte Datei öffnen
	Erwartetes Ergebnis	• Es wird die jeweilige Datei geöffnet und die Daten werden angezeigt
	Tatsächliches Ergebnis	Die unterschiedlichen Dateien können geöffnet werden und die Daten werden aktualisiert.
018	**Anwendung schließen**	
	Vorbedingungen	Es sind nicht gespeicherte Aufzeichnungsdaten vorhanden
	Handlung	• Anwendung schließen • Anwendung erneut starten
	Erwartetes Ergebnis	• Alle Daten sind immer noch vorhanden • Das letzte aktive Tracking ist ausgewählt
	Tatsächliches Ergebnis	Die Daten wurden gespeichert und sind vorhanden.
019	**Daten filtern**	
	Vorbedingungen	Es sind Daten mit unterschiedlichen Semestern und Modulen vorhanden.
	Handlung	• Über die Comboboxen und durch Texteingabe Daten wählen • Unterschiedliche Kombinationen von Semester, Modul und Kategorie
	Erwartetes Ergebnis	• Die Tabelle wird aufgrund der Datenauswahl gefiltert
	Tatsächliches Ergebnis	Die Tabelle wird entsprechend der Auswahl gefiltert.

In Abbildung 14 ist die Testabdeckung, die mit den automatisierten Tests erreicht wird dargestellt. Zur Auswertung wurde das Modul coverage verwendet.

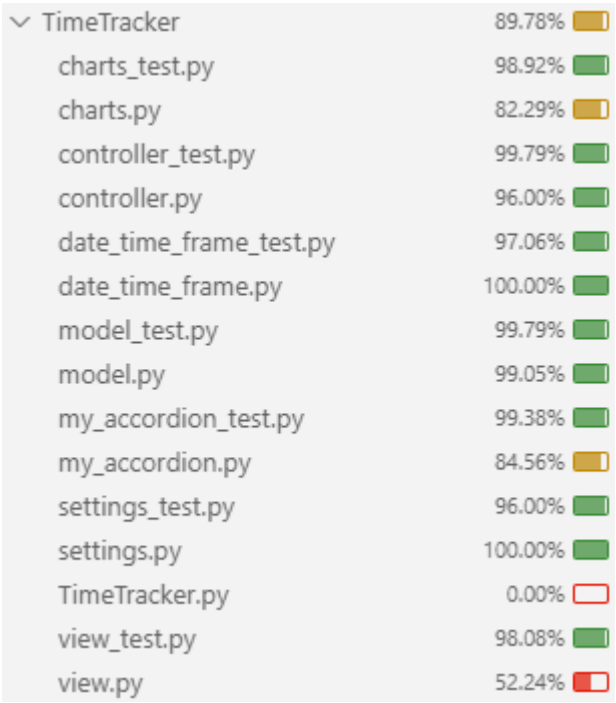

Abbildung 14: Testabdeckung der Unittests

Aus der ersten Testausführung ergaben sich folgende Erkenntnisse:

- Das Öffnen des „Eintrag bearbeiten Dialogs“ ist langsam. Eine Untersuchung des Codes zeigte, dass das Laden des DateTimeFrames und insbesondere des DateEntry lange dauert.
- Sämtliche Dialoge werden in der oberen linken Ecke des Hauptbildschirms dargestellt unabhängig davon wo das Hauptfenster der Anwendung auf dem Bildschirm positioniert ist.
- Standardwerte für Module sollen über eine Konfiguration einstellbar sein.
- Der Titel im Dialog zum Bearbeiten des Studiums ist falsch.
- Die Fehlermeldung, die angezeigt wird, wenn ein Diagrammtyp für die ausgewählten Daten nicht erstellt werden kann, enthält einen Tippfehler.

Die Fehler wurden behoben und bei einer weiteren Testausführung wurden nur die fehlgeschlagenen Tests und die automatisierten Unittests erneut durchgeführt. Dabei konnten keine weiteren Probleme identifiziert werden.

6 Abstract

Für das Modul *Projekt: Software Engineering* wurde eine Softwareanwendung konzipiert und entwickelt. Während der Durchführung mussten in drei Phasen (Konzeption, Erarbeitung und Reflexion sowie Finalisierung) sämtliche Schritte des Softwareentwicklungsprozess durchlaufen werden.

Als Student und Angestellter, der in seiner Laufbahn nur mit Softwareentwicklung und Testing innerhalb des Softwareentwicklungsprozesses Berührungspunkte hatte, ist ein Projekt, bei dem sämtliche Rollen und Verantwortungen übernommen werden müssen, eine gute Gelegenheit um Softwareprojekte aus dem Blickwinkel unterschiedlicher Akteure zu betrachten. Zusätzlich gibt es die Möglichkeit die Schnittstellen zwischen unterschiedlichen Projektteammitgliedern selbst auszutesten und ein Verständnis für andere Rollen zu entwickeln.

6.1 Entwicklungsverlauf

Die initiale Projektleitung durch Verfassen der Projektübersicht sowie die Analyse der Projektrisiken und Erstellung des Zeitplans lief gut und war schnell erledigt. Im weiteren Verlauf des Projekts geriet die Rolle des Projektleiters aus dem Fokus, was zu erheblichen Verzögerungen und mehrfachen Anpassungen im Zeitplan führte. Durch die zeitliche Verschiebung musste eine Änderung der Aufgabenstellung in Kauf genommen werden, wodurch eine geringe Anpassung der Architektur und eine umfängliche Refaktorisierung des Sourcecodes notwendig wurden.

Das Verfassen der Anforderungen war dahingehend interessant, dass die Anforderungen dokumentiert werden mussten. Bei bisherigen Projektarbeiten wurde das nie durchgeführt und somit hatten bewusste oder unbewusste Anforderungsänderungen keinen Einfluss auf das Projekt. Im vorliegenden Projekt waren keine Anforderungsänderungen notwendig, wobei die Anforderungen in der Spezifikationsphase bestimmt granularer verfasst werden hätten können. Selbst die Änderung der Architektur hatte keine Auswirkung auf die bereits verfassten Anforderungen, was aber darauf zurückzuführen ist, dass die erste Architektur zeitlich nach den Anforderungen verfasst wurde.

Mit dem Erstellen der Systemarchitektur wird der Grundstein für die technische Ausarbeitung des Projekts gelegt. Dazu muss der Architekt den Gesamtüberblick über die Applikation haben. Zusätzlich müssen Aspekte, wie Wartbarkeit, Skalierbarkeit und Portabilität, die auch erst nach dem Abschluss der Projektarbeit in Kraft treten können, berücksichtigt werden. Die Arbeit mit UML-Diagrammen war dabei hilfreich um Gedanken und Abläufe zu verschriftlichen und zu visualisieren.

Die Entwicklung der Anwendung war dahingehend anspruchsvoll, dass die selbst verfassten Anforderungen beachtet werden mussten. Von anderen Projektarbeiten ist man es gewohnt, die Anforderungen während der Entwicklung auszulegen. Da die Anforderungen in diesem Projekt verschriftlicht wurden, war das nicht möglich und somit war die Entwicklungsarbeit realitätsnaher gestaltet. Das Entwickeln von Unittests während der Ausarbeitung der Applikation erwies sich als

besonders hilfreich, vor allem während der Refaktorisierung des gesamten Sourcecodes nach der Anpassung der Architektur.

Die Position des Testers wurde erst nach der Entwicklung des Sourcecodes wahrgenommen. realistischer wäre es gewesen den Testplan und die aus den Anforderungen abgeleiteten Tests während oder vor der Entwicklung der Anwendung zu erstellen, wie es in Softwareentwicklungsprojekten üblich ist. Dadurch hätte auch eine bessere Kontrolle für die Erfüllung der Anforderungen erfolgen können. So wurden während der Entwicklung kleine Funktionstests durchgeführt, die nicht im Testprotokoll aufscheinen.

6.2 Was verlief gut?

Die Anforderungen konnten eingehalten werden und es mussten keine Anforderungen umformuliert werden.

Besonders gut verlief die Verwendung von automatisierten Unittests, um frühzeitig Fehler durch Änderungen in verschiedenen Codeteilen zu entdecken und zu beheben. Die Zusammenarbeit von Entwicklung und Test verlief auch positiv, da sowohl der Entwickler als auch der Tester, da diese Rollen von derselben Person ausgeführt wurden, ein umfangreiches Systemwissen hatten. Dieser Umstand hat aber ausgeschlossen, dass der Tester einen Blackbox Test durchführen kann, bei dem er keinen Einblick in die Funktionsweise der Anwendung hat.

6.3 Was könnte man beim nächsten Projekt besser machen? Welche wichtigen Erkenntnisse wurden gewonnen?

Zweifelsohne hat das Zeitmanagement in diesem Projekt versagt. Die Rolle des Projektleiters wurde nach der initialen Projektplanung nicht mehr wahrgenommen. Dadurch wurde nur die technische Weiterführung des Projekts ohne Rücksicht auf zeitliche Aspekte vorangetrieben. Dies hatte zur Folge, dass der Zeitplan mehrfach nicht eingehalten werden konnte. Für weitere Projektarbeiten sollte der Zeitplan konsequent überprüft werden und einzelne Arbeitspakete abgeschlossen werden bevor neue begonnen werden.

Aus der Softwareentwicklung bin ich es, sowohl in der Konzeption als auch in der Entwicklung von Software und von Tests, gewohnt mit Reviews zu arbeiten, um die Arbeitsergebnisse zu verbessern und Unsicherheiten zu beseitigen. In diesem Projekt ist dies aufgrund der Prüfungsleistung nicht möglich. Es wurde nur das Feedback nach der Abgabe der einzelnen Phasen eingeholt und eingearbeitet. Es hätte die Möglichkeit bestanden durch zusätzliche Rückfragen Unsicherheiten vorzeitig zu beseitigen. Solche Möglichkeiten sollten bei zukünftigen Projektarbeiten genutzt werden.