



Hochschule Macromedia für angewandte Wissenschaften
University of Applied Sciences

Kursbezeichnung: Aufbaumodul Programmieren
Prüfer: Thomas Berger

Vom Studierenden auszufüllen:

300164
Matrikelnummer

D-PBf DT DFO 6d 24W
Kohorte

Balthasar
Nachname

Leif
Vorname

Die Arbeit wird eingereicht als:
(Tragen Sie bitte in die zutreffende Box den Buchstaben X ein)

- Einzelarbeit
 Gruppenarbeit

Trifft nur auf Gruppenarbeiten zu: (Nur bei Gruppenarbeiten auszufüllen)

Falls Sie eine Gruppenarbeit einreichen, dann müssen bitte die Vor- und Nachnamen aller Gruppenmitglieder aufgeführt werden. Die Namen sind von den jeweiligen Gruppenmitgliedern selbst elektronisch einzutragen. Durch Eintrag des Namens wird bestätigt, dass der/die Studierende mit der Abgabe der Arbeit in der vorliegenden Form einverstanden ist. Ferner wird mit Eintrag des Namens erklärt, die Projektarbeit (bei Gruppenarbeit: den von dem/der jeweiligen Studierenden erstellten und entsprechend gekennzeichneten Teil der Arbeit selbständig und ohne fremde Hilfe angefertigt zu haben. Dabei hat der/die Studierende sich keiner anderen Hilfsmittel bedient als derjenigen, die im beigefügten Quellenverzeichnis genannt sind. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind von dem/der Studierenden als solche kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

1) _____
2) _____
3) _____
4) _____

5) _____
6) _____
7) _____
8) _____

Bewertung der Gruppenarbeiten:

(Tragen Sie bitte in die zutreffende Box den Buchstaben X ein)

- Ich beantrage bei meiner Gruppenarbeit eine Individualbewertung
 Ich beantrage bei meiner Gruppenarbeit eine Gruppenbewertung

Eine Individualbewertung bedeutet, dass jedes Gruppenmitglied eine individuelle Note bekommt und eine Gruppenbewertung bedeutet, dass jedes Gruppenmitglied eine identische Note bekommt.

Leverkusen, 07.06.2025
Ort/Datum

Leif Balthasar
Vollständiger Vor- und Nachname

Vom Prüfer auszufüllen: (Freitext für die Zweitkorrektur)

Studiengang:
Digital Technologies & Coding B.Sc.

Studienrichtung:
Digital Forensics

Lehrveranstaltung:
Aufbaumodul Programmieren

Entwicklung eines objektorientierten Library Management Systems

Verfasser: Leif Balthasar
Matrikelnummer: 300164
Prüfer/in: Thomas Berger

Die Projektarbeit wurde am 27.06.2025 bei der Hochschule
Macromedia in Düsseldorf digital eingereicht.

Inhaltsverzeichnis

1. Projektidee	1
2. Motivation	1
3. Projektziele	1
4. Rollen und Rechte.....	1
5. Programmstruktur	2
6. Technische Umsetzung.....	2
7. Datenfluss	2
8. UML-Diagramm.....	2
9. Besonderheiten.....	3
10. Weiterentwicklungspotential	3
11. Fazit	3
Quellenverzeichnis.....	4
KI-Verzeichnis.....	5

Konzept: Library Management System

1. Projektidee

Ziel des Projekts ist die Entwicklung eines textbasierten Bibliotheksverwaltungssystems in Python. Nutzer können verschiedene Medientypen – wie Bücher, DVDs und Magazine – ausleihen und zurückgeben. Ein integriertes Rollen- und Rechtesystem sorgt dabei für eine differenzierte Zugriffssteuerung je nach Benutzerrolle.

2. Motivation

Eine effiziente Verwaltung der Mediennutzung ist in Bibliotheken essenziell. Dieses Projekt bildet ein vereinfachtes Bibliothekssystem ab und dient gleichzeitig als praxisnahe Übung zur Anwendung objektorientierter Programmierung, Dateiverwaltung und benutzerfreundlicher Terminalinteraktion. Darüber hinaus vermittelt es Grundlagen der Softwarearchitektur und zeigt den praktischen Umgang mit strukturierten Daten im JSON-Format.

3. Projektziele

- Entwicklung eines menügesteuerten Programms in Python
- Verwaltung von Benutzern mit unterschiedlichen Rollen
- Ausleihe und Rückgabe von Medien
- Persistente Datenspeicherung und -ladung über JSON-Dateien
- Anwendung objektorientierter Prinzipien (Vererbung, Klassen, Methoden)
- Einheitliche und intuitive Benutzerführung über das Terminal

4. Rollen und Rechte

Das System unterscheidet zwischen drei Rollen mit abgestuften Rechten:

Rolle	Rechte
User	Medien anzeigen, ausleihen, zurückgeben
Verwaltung	Zusätzlich: Medien verwalten, Benutzer hinzufügen oder entfernen
Admin	Vollzugriff, inkl. Rollenänderungen und Entfernen von Medien oder Benutzern

5. Programmstruktur

Modul/Datei	Funktion
LibrarySystem.py	Einstiegspunkt mit Hauptmenü und Steuerungslogik
external_functions.py	Enthält alle zentralen Klassen, Methoden und Menüfunktionen
media.json	Permanente Speicherung aller Medienobjekte
users.json	Permanente Speicherung aller Benutzerinformationen

6. Technische Umsetzung

- **Sprache:** Python 3
- **Benutzeroberfläche:** Textbasiertes Menü im Terminal
- **Datenhaltung:** JSON-Dateien für dauerhafte Speicherung
- **Datenmodelle:** User, Book, DVD, Magazine, MediaItem, Author
- **Designprinzipien:**
 - Objektorientierte Programmierung (Vererbung, Kapselung)
 - Trennung von Logik und Benutzeroberfläche
 - Rollenbasiertes Zugriffssystem
 - Verwendung von UUIDs zur eindeutigen Identifikation von Medien

7. Datenfluss

- **Beim Programmstart:** Einlesen von media.json und users.json
- **Während der Nutzung:** Interaktion über Menüführung, Datenverarbeitung im RAM
- **Beim Beenden:** Änderungen werden zurück in die JSON-Dateien geschrieben

8. UML-Diagramm

Zur Visualisierung der objektorientierten Struktur wurde ein UML-Klassendiagramm erstellt. Es zeigt die wichtigsten Klassen des Systems, ihre Attribute, Methoden und Vererbungsbeziehungen.

Das zentrale Element ist die Basisklasse MediaItem, von der die spezialisierten Klassen Book, DVD und Magazine erben. Jede dieser Klassen besitzt eigene zusätzliche Attribute, etwa isbn bei Büchern oder duration bei DVDs. Die Methoden borrow() und return_item() sind gemeinsam in der Oberklasse definiert, wodurch Wiederverwendbarkeit und Strukturklarheit erreicht werden.

Die Klasse User enthält zentrale Benutzerdaten und verweist auf ausgeliehene Medien inklusive Zeitstempeln. Von User erbt die erweiterte Klasse Author, welche über zusätzliche Eigenschaften (biography, writtenBooks) verfügt. Die Methoden borrowItem() und returnItem() ermöglichen die Interaktion mit Medienobjekten.

Die Klasse LibrarySystem fungiert als Verwaltungseinheit für alle Benutzer und Medien. Sie enthält Methoden zur Verwaltung von Objekten, zum Einlesen und Speichern von Daten sowie zur Rollenänderung.

Dieses UML-Diagramm diente als Grundlage für die strukturierte Entwicklung der Software und half, die Klassenzusammenhänge frühzeitig klar zu definieren.

9. Besonderheiten

- Konsolenfreundliche Darstellung mit Farben, Symbolen und Animationen
- Zeitstempel bei jeder Ausleihe zur Nachverfolgung
- Versteckte Funktionen (Easter-Egg durch Eingabe von _EE_)
- Stabile Bedienung durch Fehlerbehandlung und Abbruchlogik

10. Weiterentwicklungsdimensionen

- Ergänzung eines Statistikmoduls (z. B. meistgeliehene Medien)
- Einführung eines Mahnwesens mit Rückgabefristen
- Passwortverschlüsselung zur Erhöhung der Datensicherheit
- Entwicklung einer grafischen Benutzeroberfläche (z. B. mit Tkinter oder PyQt)

11. Fazit

Dieses Projekt war nicht nur eine technische Umsetzung, sondern auch eine kreative und motivierende Erfahrung. Die Entwicklung eines eigenen Bibliotheksverwaltungssystems mit Python hat gezeigt, wie erfüllend es sein kann, ein Programm Schritt für Schritt wachsen zu sehen – von den ersten Klassen bis hin zur fertigen Anwendung mit Nutzerrollen, Medientypen und persistenter Datenspeicherung.

Besonders die Arbeit mit objektorientierter Programmierung ermöglichte eine klare, strukturierte Denkweise, die sich durch das gesamte Projekt zieht. Jede Funktion, jede Klasse hatte ihren Platz – und es war spürbar, wie aus abstrakten Konzepten nach und nach ein funktionierendes System wurde. Dabei war nicht nur das Programmieren an sich spannend, sondern auch die Herausforderung, Benutzerfreundlichkeit, Fehlerbehandlung und Datenstruktur so zu gestalten, dass das System intuitiv bedienbar und robust bleibt.

Die Möglichkeit, zwischen verschiedenen Benutzerrollen zu unterscheiden, Medien zu verleihen und sogar eine kleine visuelle Benutzerführung mit Unicode-Symbolen und Farbcodes einzubauen, machte das Projekt nicht nur lehrreich, sondern auch richtig schön. Es war eine Freude zu sehen, wie all die kleinen Bausteine zusammenspielen – und wie durch jeden Entwicklungsschritt das System ein Stück "lebendiger" wurde.

Darüber hinaus zeigt das Projekt, dass selbst komplexere Anwendungen mit Python erreichbar sind – auch für Lernende. Es motiviert, weiterzumachen, mehr zu entdecken und eigene Ideen umzusetzen. Genau darin liegt sein besonderer Wert: Es verbindet technische Kompetenz mit echter Begeisterung fürs Programmieren.

Quellenverzeichnis

ChatGPT. (o. D.). *ChatGPT*. OpenAI. <https://chatgpt.com/>

GeeksforGeeks. (o. D.). *Object-Oriented Programming (OOPs) in Python*. <https://www.geeksforgeeks.org/python-object-oriented-programming/>

Lutz, M. (2013). *Learning Python* (5. Aufl.). O'Reilly Media.

MySQL. (o. D.). *MySQL 8.0 Reference Manual*. Oracle Corporation. <https://dev.mysql.com/doc/refman/8.0/en/>

Programmieren Starten. (o. D.). *YouTube – Programmieren Starten*. YouTube. <https://www.youtube.com/@ProgrammierenStarten>

SQLite. (o. D.). *SQLite Documentation*. <https://www.sqlite.org/docs.html>

Stack Overflow. (o. D.). *Questions*. Stack Exchange Inc. <https://stackoverflow.com/questions>

W3Schools. (o. D.). *Python Tutorial*. <https://www.w3schools.com/python/>

Wayne, G. (2020). *Clean Architecture for Python Developers*. Leanpub. <https://leanpub.com/clean-architecture-python>

KI-Verzeichnis

Prompt 1

Zeile:

Zeile 135–175 in external_functions.py (Funktion menu_delete_article)

Wortlaut des Prompts:

„Korrigiere meine Funktion zum Löschen von Medien aus dem Bibliothekssystem. Sie soll alle Medientypen (Buch, DVD, Magazin) korrekt anzeigen und nach Benutzereingabe löschen. Der Ablauf soll robust sein, fehlerhafte Eingaben abfangen und die Ausgabe visuell verständlich gestalten.“

Übernahme sowie Erklärung der Herangehensweise:

Die ursprüngliche Funktion war funktional, aber unvollständig strukturiert, inkonsistent bei der Statusanzeige und hatte keine einheitliche Fehlerbehandlung. Mit Hilfe von ChatGPT wurde die Funktion vollständig überarbeitet:

- **Einführung klarer Strukturen:** Jeder Medientyp wird in einem eigenen Block abgefragt, gelistet und separat verarbeitet.
- **Bessere Nutzerführung:** Durch Unicode-Symbole (📘, 🎬, 📖), klare Rückmeldungen und farbliche Hervorhebungen.
- **Fehlerrobustheit:** Leere Eingaben oder ungültige Titel führen nicht mehr zum Absturz, sondern zu klaren Hinweisen.
- **Objektprüfung:** Löschung erfolgt nur bei vorhandenem Objekttyp + Titelübereinstimmung.

Prompt 2

Zeile:

Zeile 103–180 in class_librarySystem.py
(Funktion load_users_from_json)

Wortlaut des Prompts:

„Wie kann ich aus einer JSON-Datei Benutzer wiederherstellen, inklusive ihrer ausgeliehenen Bücher und der zugehörigen Zeitstempel? Die Objekte sollen korrekt rekonstruiert werden und alle Relationen müssen wiederhergestellt sein.“

Übernahme sowie Erklärung der Herangehensweise:

Diese Funktion war technisch sehr anspruchsvoll. Die KI half, die **logische Verknüpfung zwischen Benutzern, ausgeliehenen Büchern und Zeitstempeln** korrekt herzustellen:

- **Typprüfungen:** Nur Bücher mit ISBN werden ausgeliehen, andere Medientypen ignoriert.
- **Datenbindung:** Die Zeitstempel werden per item.itemID im Dictionary borrowTimestamps gespeichert.
- **Fehlervermeidung:** Schutz bei fehlerhaften oder fehlenden Einträgen durch try-Blöcke und Fallbacks.

-
- **Doppelte Schleifenlogik:** Erst wird nach dem passenden Buch gesucht, dann die Zeitstempel zugewiesen – das wäre manuell leicht fehleranfällig gewesen.

Prompt 3

Zeile:

Zeile 203–275 in main.py

(Funktion main_menu mit Rollensteuerung)

Wortlaut des Prompts:

„Ich möchte ein dynamisches Hauptmenü mit unterschiedlichen Rechten für admin, verwaltung und normale Benutzer. Kannst du mir helfen, das übersichtlich, robust und nutzerfreundlich zu gestalten?“

Übernahme sowie Erklärung der Herangehensweise:

Die KI unterstützte bei der **Strukturierung eines mehrstufigen, rollenbasierten Menüs** mit diesen Aspekten:

- **Zentrale Menüführung:** Je nach Rolle wird ein anderer Block des Menüs gezeigt.
- **Farbkodierung & Unicode-Symbole:** Übersicht und Lesbarkeit wurden deutlich erhöht.
- **Easter Egg Integration:** Sonderkommando _EE_ wurde in alle Rollen integriert.
- **Fehlerabfang:** Ungültige Eingaben führen zu Feedback, aber nicht zum Absturz.
- **Rückgabe-Logik:** Durch return "switch" wird der Log-out technisch elegant gelöst.

Prompt 4

Zeile:

Zeile 316–396 in external_functions.py

(Funktion menu_borrow_article)

Wortlaut des Prompts:

„Ich brauche ein robustes Ausleihsystem, bei dem Benutzer ein verfügbares Medium auswählen können – sortiert nach Typ. Es soll dabei geprüft werden, ob z. B. Bücher nur begrenzt ausgeliehen werden dürfen. Die Eingabe soll fehlerresistent sein.“

Übernahme sowie Erklärung der Herangehensweise:

Die KI half hier, eine **mehrstufige, benutzergesteuerte Auswahl** zu entwickeln:

- **Medientypauswahl mit Icons:** Der User wählt erst Buch/DVD/Magazin.
- **Verfügbarkeitsprüfung:** Nur freie Medien werden angezeigt.
- **Limitprüfung:** Bücher sind auf 3 Ausleihen pro Benutzer begrenzt.
- **Titelmatching:** Unabhängig von Groß-/Kleinschreibung.
- **Elegantes Feedback:** Klare Rückmeldungen bei Abbrüchen, Fehleingaben und Erfolgen.

Prompt 5

Zeile:

Zeile 23–82 in class_librarySystem.py
(Funktion load_media_from_json)

Wortlaut des Prompts:

„Ich möchte aus einer JSON-Datei automatisch unterschiedliche Medientypen wie Bücher, DVDs und Magazine laden und sie korrekt rekonstruieren. Dabei sollen unbekannte Typen abgefangen und übersprungen werden.“

Übernahme sowie Erklärung der Herangehensweise:

Die KI unterstützte bei der sauberer **Objekterzeugung aus dynamischen Daten**:

- **Typerkennung mit `type.lower()`** zur Fallunterscheidung.
- **Zuweisung der optionalen Felder mit `getattr()`** und Standardwerten.
- **Fehlerresistenz:** Unerkannte Typen werden erkannt und protokolliert.
- **Verfügbarkeitsstatus** (`available`) wird korrekt berücksichtigt und gesetzt.
- **Lesbarkeit:** Die Datenstruktur im Code bleibt trotz Mehrtyp-Handling übersichtlich.

Eidesstattliche Erklärung

Ich, Leif Balthasar
geboren am 12.08.2002

erkläre hiermit, die vorliegende Arbeit selbständig und ohne fremde Hilfe angefertigt zu haben. Dabei habe ich mich keiner anderen Hilfsmittel bedient als derjenigen, die im beigefügten Quellenverzeichnis genannt sind.

Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind von mir als solche kenntlich gemacht.

Düsseldorf, den 07.06.2025
Studienort

.....
Unterschrift Studierende/r (= Verfasser/in)
