

Dokumentation Informatik II

**Entwicklung eines Intelligenten Empfehlungssystems**

Vorlesungsfach: Informatik 2

Datum: 15.06.2025

Studiengang: TEL24B

Name des prüfenden Dozenten: Prof. Dr. Ossmane Krini

Name der Studenten: Carina Halter: 2000130

Laura Mirwald: 3816633

Semian Finné: 8543533

Tristan Bihler: 5893076

Inhaltsverzeichnis

[1. Projektidee & Zieldefinition 3](#_Toc200925410)

[2. Anforderungsanalyse & Konzeptentwicklung 3](#_Toc200925411)

[Programmiersprache und Entwicklungsumgebung 3](#_Toc200925412)

[Datenbanken 4](#_Toc200925413)

[Bibliotheken 5](#_Toc200925414)

[Vorläufiger Programmablaufplan 5](#_Toc200925415)

[GUI - Layout 7](#_Toc200925416)

[Zeitplan 9](#_Toc200925417)

[3. Architektur- & Systemdesign 11](#_Toc200925418)

[MVC Architektur Model-View-Controller 11](#_Toc200925419)

[Fertiges Design der Benutzeroberfläche 14](#_Toc200925420)

[4. Implementierung / Codierung 17](#_Toc200925421)

[Programmablaufplan 17](#_Toc200925422)

[Algorithmus 20](#_Toc200925423)

[5. Testen & Debugging 23](#_Toc200925424)

[Testen 23](#_Toc200925425)

[Debugging 23](#_Toc200925426)

[6. Bugs & Zukünftiges 24](#_Toc200925427)

[Bugs 24](#_Toc200925428)

[Zukünftiges 27](#_Toc200925429)

[7. Reflexion & Nachbereitung 28](#_Toc200925430)

[8. Quellenverzeichnis 29](#_Toc200925431)

[10. Anhang 0](#_Toc200925432)

[Beispiele für ausgefüllte Umfragebögen 0](#_Toc200925433)

[Code 3](#_Toc200925434)

Abbildungsverzeichnis

[Abbildung 1: Vorläufiger Programmablaufplan 7](#_Toc200925435)

[Abbildung 2: Login-Fenster im Ganzen 8](#_Toc200925436)

[Abbildung 3: Login-Fenster vergrößert 8](#_Toc200925437)

[Abbildung 4: Hauptfenster - Filmauswahl, Beschreibung, Empfehlungen 9](#_Toc200925438)

[Abbildung 5: Favoriten-Fenster 10](#_Toc200925439)

[Abbildung 6: Zeitplan des Projektes "Filmempfehlungssystem" 11](#_Toc200925440)

[Abbildung 7: MVC-Flussdiagramm - ursprüngliche Version 12](#_Toc200925441)

[Abbildung 8: MVC-Flussdiagramm - heutige Definition 13](#_Toc200925442)

[Abbildung 9: Fehlermeldung - User existiert nicht 15](#_Toc200925443)

[Abbildung 10: Login-Maske fertiger GUI 15](#_Toc200925444)

[Abbildung 11: Fehlermeldung - User existiert bereits 15](#_Toc200925445)

[Abbildung 12: Hauptmenü - Empfehlungen 16](#_Toc200925446)

[Abbildung 13: Meine Favoriten & Empfehlungen auf verhaltensähnliche Nutzer 17](#_Toc200925447)

[Abbildung 14: Programmablaufplan - Login 18](#_Toc200925448)

[Abbildung 15: Programmablaufplan - Main 19](#_Toc200925449)

[Abbildung 16: Programmablaufplan - Favoriten 20](#_Toc200925450)

[Abbildung 17: Content-based Algorithm 22](#_Toc200925451)

[Abbildung 18: User Collaborative Algorithm 23](#_Toc200925452)

[Abbildung 19: Bug bei der Fehlermeldung doppelter Benutzernamen Registrierung 27](#_Toc200925453)

[Abbildung 20: Bug - Beschreibungsfeld variiert -> großer Abstand 27](#_Toc200925454)

[Abbildung 21:: Bug - Beschreibungsfeld variiert -> kleiner Abstand 28](#_Toc200925455)

# 1. Projektidee & Zieldefinition

In der heutigen digitalen Welt sind Online-Dienste fester Bestandteil unseres Alltags.  
Ob beim Einkaufen, Video-Streaming oder in sozialen Netzwerken - personalisierte Werbung ist auf unzählige Online-Plattformen kaum wegzudenken. Hier liegt das Problem, denn wie finden Nutzer unter der Informationenflut im Netz genau das, was sie wirklich interessiert und ihren individuellen Vorlieben entspricht?  
Die Antwort ist ein Empfehlungssystem.

Das Ziel unseres Projekts ist es, dieses Problem zu lösen und ein solches Empfehlungssystem zu entwickeln und zu implementieren. Es soll in der Lage sein, basierend auf dem historischen Verhalten und den Präferenzen von Nutzern maßgeschneiderte Empfehlungen zu generieren und zu speichern. Dies beinhaltet die Konzeption und Implementierung einer geeigneten Datenstruktur für Nutzerpräferenzen, sowie die Entwicklung eines Algorithmus zur Berechnung von Ähnlichkeiten zwischen Produkten.

Wir haben uns für die konkrete Entwicklung eines Empfehlungssystems für einen Streaming-Dienst entschieden. Hierfür muss zum Beispiel ein Weg gefunden werden, um herauszufinden welche Filme einem ähnlichen Genre angehören und ob diese dem Nutzer gefallen können oder eher nicht. Außerdem soll es eine Möglichkeit geben, dass Filme in einer Favoritenliste gespeichert werden können und zusätzlich darauf Empfehlungen angezeigt werden.

# 2. Anforderungsanalyse & Konzeptentwicklung

## Programmiersprache und Entwicklungsumgebung

Durch unser gewähltes Projekt ist es eine Vorgabe, dieses in der Programmiersprache Python umzusetzen. Jedoch haben wir uns ganz gezielt dafür entschieden. Zum einen sollte das erste Projekt in C++ geschrieben werden und da ein Projekt in C diesem sehr ähneln würde, tendierten wir eher zur Python-Alternative. Dadurch können wir unser erst frisch erlerntes Wissen in der etwas anderen Programmiersprache erweitern und festigen. Zum anderen hat Python auch einige Vorteile. Python ist sehr Programmierfreundlich, da es relativ einfach und leicht verständlich aufgebaut ist. Außerdem ist es gut dafür geeignet Daten einfach und in einer schnellen Entwicklungszeit zu veranschaulichen. In Bezug auf unser Projekt ist es natürlich auch von Vorteil, dass die Programmiersprache selbst schon viel in Listen arbeitet. Diese benötigen wir später mehrmals zum Verwalten unserer Film-Auswahl, sowie Favoriten bzw. den Empfehlungen.

Als Entwicklungsumgebung haben wir uns für Visual Studio Code entschieden. Da wir diese bereits im 1. Semester installiert und bereits für Python in den vier Vorlesungen des zweiten Semesters initialisiert hatten, war diese Wahl Ideal für uns. Wir können direkt mit dem Coden beginnen, ohne weitere Compiler, Packages oder sonstige Installationen tätigen zu müssen.

## Datenbanken

Um unser Filmempfehlungssystem realistisch zu gestalten und auch mit richtigen Informationen zu füllen, haben wir uns zur Realisierung für JSON-Dateien entschieden.

Warum genau eine JSON-Datei und keine andere?   
Ursprünglich war die Idee mit Firebase zu arbeiten, welche JSON-Dateien unterstützt. Allerdings haben wir uns aufgrund kostenpflichtiger Funktionen, die wir für unsere Umsetzung benötigen würden, gegen Firebase entschieden.   
Da wir uns bereits durch die ursprünglich geplante Datenbank «Firebase» mit JSON-Dateien auseinandergesetzt und sie sich für unsere Anwendungen bewährt hatten, wollen wir diese weiterhin nutzen.

Was ist eine JSON-Datei?  
Eine JSON-Datei ist eine rein textbasierte Datei, welche Daten im JavaScript Object Notation Format speichert. Ursprünglich stammt sie von JavaScript, ist jedoch auch in vielen weiteren Programmiersprachen verwendbar, darunter auch Python.

Häufig wird eine solche Datei als Zwischenspeicher benutzt, um Daten zwischen einer Webanwendung (wie z.B. einer Website) und einem Server zu übertragen. Das bedeutet, dass Informationen strukturiert gespeichert und gesendet werden, sodass der Server diese problemlos interpretieren kann.

Zu den Vorteilen einer JSON-Datei zählt die einfache und schnelle Einrichtung. Unter anderem auch dadurch, dass sie sowohl für Maschinen als auch für Menschen lesbar ist. Des Weiteren ist sie eine gute Wahl, wenn man statische Daten benutzt, also wenn Daten selten geändert werden. Allerdings sollte beachtet werden, dass bei großen Datenmengen die Performance stark beeinflusst wird. Denn jedes Mal, wenn Daten aus einer JSON-Datei aufgerufen werden, muss die Datei neu geladen werden. Außerdem benötigt sie viel Speicherplatz, da sie aus alphanumerischen Zeichen besteht.

Für unser Projekt wollen wir die Filminformationen, wie Titel, Beschreibung, Genre und YouTube Trailer aus einer TMDB (The Movie Database) mithilfe einer API in unsere JSON-Datei speichern.

Eine API (Application Programming Interface) dient als Schnittstelle, über die wir gezielt die Filminformationen von der Website abrufen und verarbeiten können.

## Bibliotheken

Unser Filmempfehlungssystem soll über eine sich öffnende grafische Benutzeroberfläche (GUI ≙ Graphical User Interface) gesteuert und ausgegeben werden. Dafür soll die Bibliothek «Tkinter» verwendet werden.

Tkinter ist ein GUI-Packages, welches bereits in der Standardbibliothek von der uns genutzten Programmiersprache Python enthalten ist. Mit dem Modul können grafische Benutzeroberflächen, wie Fenster, Buttons und Eingabefelder erstellt werden. Außerdem ist es auf verschiedenen Betriebssystemen direkt anwendbar, weshalb keine weitere Installation benötigt wird.

Die Bibliothek ist für unser Projekt gut anzuwenden, da sie noch weitere Vorteile mit sich bringt:

* Einfache Syntax/Bedienung:

Tkinter ist leicht zu erlernen und man muss den Hintergrund, auf der die Bibliothek basiert, nicht genauer verstehen.

* Übersichtliche Struktur:

Außerdem ist die Struktur funktionierender GUI’s einfach gehalten und kann mit kurzen Codes beschrieben werden.

* Flexible Positionierung:

Auch bei der Positionierung wird zum Beispiel deutlich, dass Tkinter einfach gehalten ist. Denn die Positionen unserer gewünschten Funktionen können auf drei verschiedene Arten festgelegt werden: Entweder können die Aktionen einfach im Feld per «pick and place» (direkte Platzierung innerhalb des Fensters) definiert werden, über Zeilen und Spalten oder bei der genauen Variante über fest definierte Positionen und Größe, die explizit gesetzt werden müssen.

Durch die einfache Struktur und dennoch umfangreichen Funktionen, sollte Tkinter für unseren Anwendungsfall gut geeignet sein, ein einfach gehaltenes und intuitives Userinterface für unser Projekt erstellen zu können.

## Vorläufiger Programmablaufplan

Nachdem nun das technische Konzept gegeben ist, haben wir uns an die detaillierteren Details des Projektes gesetzt. Zunächst wurde ein grober Programmablaufplan erstellt, der die Funktionsweise unseres zu erstellenden Codes beschreibt.

Wird unser Programm ausgeführt, so wird sich ein Fenster öffnen. In diesem sich der Benutzer entweder mit einem bereits vorhandenen Benutzernamen anmelden oder neu registrieren kann. Bei der Registrierung kann der Name frei gewählt werden und wird anschließend in unsere JSON-Datei gespeichert. Anschließend kann der User sich mit seinem Account in unser Filmeempfehlungssystem anmelden.

Nach einem erfolgreichen Login wird der User zum Main-Fenster weitergeleitet. In dem verschiedene Ausgaben angezeigt werden, darunter eine Filmliste, Filminformationen, sowie die vom Algorithmus generierten Empfehlungen, die aus den bereits markierten Filmen entstehen. Bei den Filminformationen wäre es sinnvoll weiteren Funktionen zu haben, wie zum Beispiel das Hinzufügen von Filmen in eine Favoritenliste und ein Button, welcher zum Trailer der entsprechenden Auswahl öffnet. Außerdem soll es eine Möglichkeit geben, zur eigenen Favoritenliste wechseln zu können. Die Liste zeigt ausschließlich die vom User markierten Filme und ermöglicht es, diese aus der aus der Liste zu entfernen.

Ein Bild, das Text, Diagramm, parallel, Plan enthält.

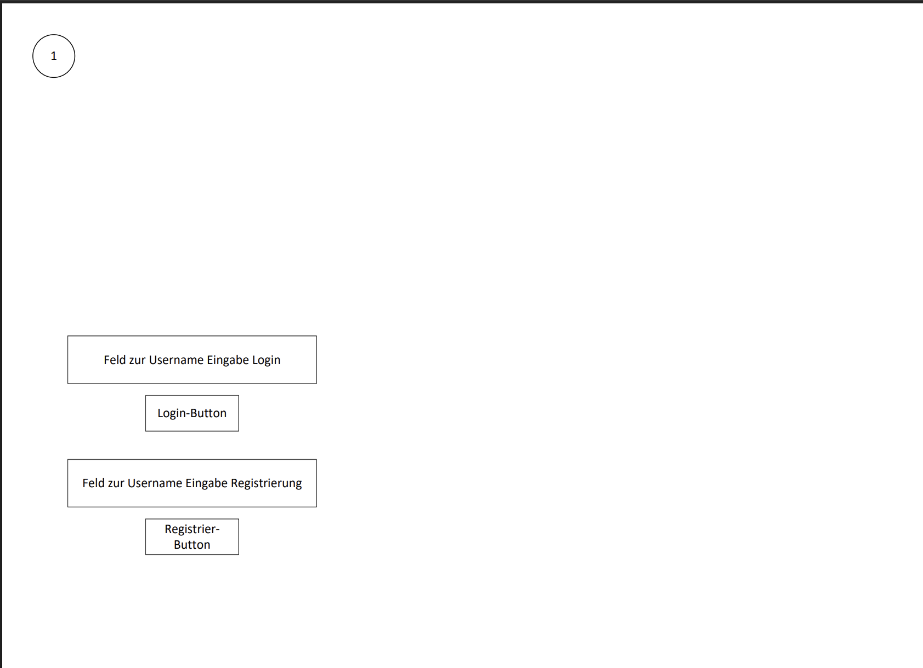
KI-generierte Inhalte können fehlerhaft sein.

Abbildung 1: Vorläufiger Programmablaufplan

## GUI - Layout

Anschließend wurde ein grobes Konzept erstellt, wie später unser Ausgabe-Fenster (GUI) aussehen könnte.

Nach dem Starten des Programmes soll sich ein Fenster öffnen, auf diesem soll man sich mittels einem Benutzernamen anmelden können. Ist man noch nicht registriert, so kann dieses in einem Feld darunter geschehen. Die Bestätigung der Eingabe soll jeweils über einen Button erfolgen.



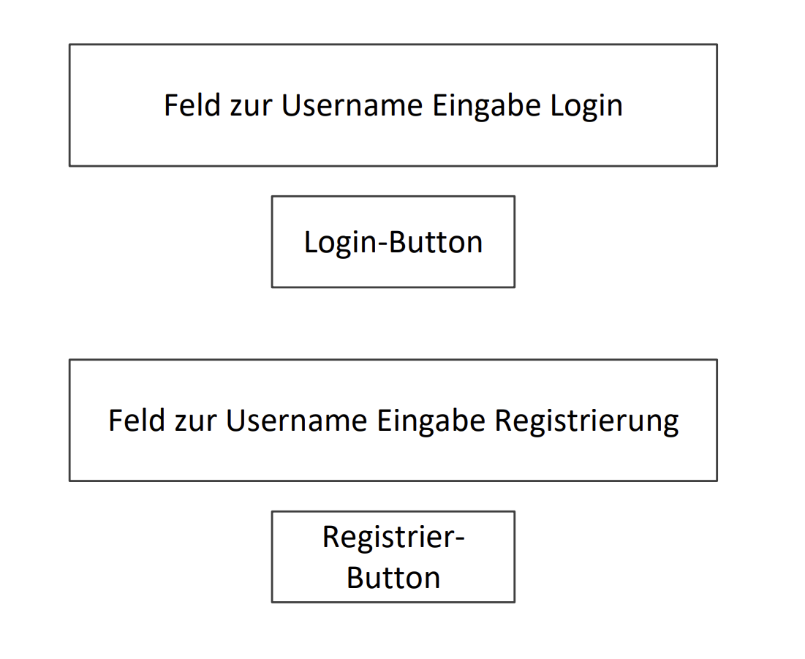


Abbildung 2: Login-Fenster im Ganzen

Abbildung 3: Login-Fenster vergrößert

Nachdem der Login erfolgreich war, kommen wir zu unserem Hauptfenster. In diesem soll es oben links eine Suchleiste geben, indem spezielle Filme nach ihrem Namen gesucht werden können. Direkt darunter werden alle verfügbaren Filme angezeigt. Sowohl die, nach denen gezielt gesucht wurde, als auch eine vollständig aufgeführte Liste, falls keine Suche erfolgte. In dem Feld rechts daneben erscheinen die Filmempfehlungen. Diese werden anhand der markierten Favoriten generiert. Unterhalb der beiden anderen Feldern gibt es eine weitere Ausgabe. In dieser werden Beschreibungen zu ausgewählten Filmen ausgegeben. Des Weiteren gibt noch folgende Features:

* Ein Button, welcher direkt zum Trailer auf YouTube weiterleitet
* Eine Schaltfläche, mit der ein Film zu Favoriten hinzugefügt werden kann
* Optional wird ein Filmcover neben der Beschreibung ausgegeben

Dazu könnte im Hauptfenster noch ein Drop-Down-Menü erscheinen, beidem zu den Favoriten und z.B. einer Freundesliste gewechselt werden könnte. Wobei die Freundesliste optional ist und wahrscheinlich aus Zeitgründen schon zuvor eher eine Erweiterung wäre.

Abbildung 4: Hauptfenster - Filmauswahl, Beschreibung, Empfehlungen

Ein Bild, das Text, Screenshot, Diagramm, Rechteck enthält.

KI-generierte Inhalte können fehlerhaft sein.

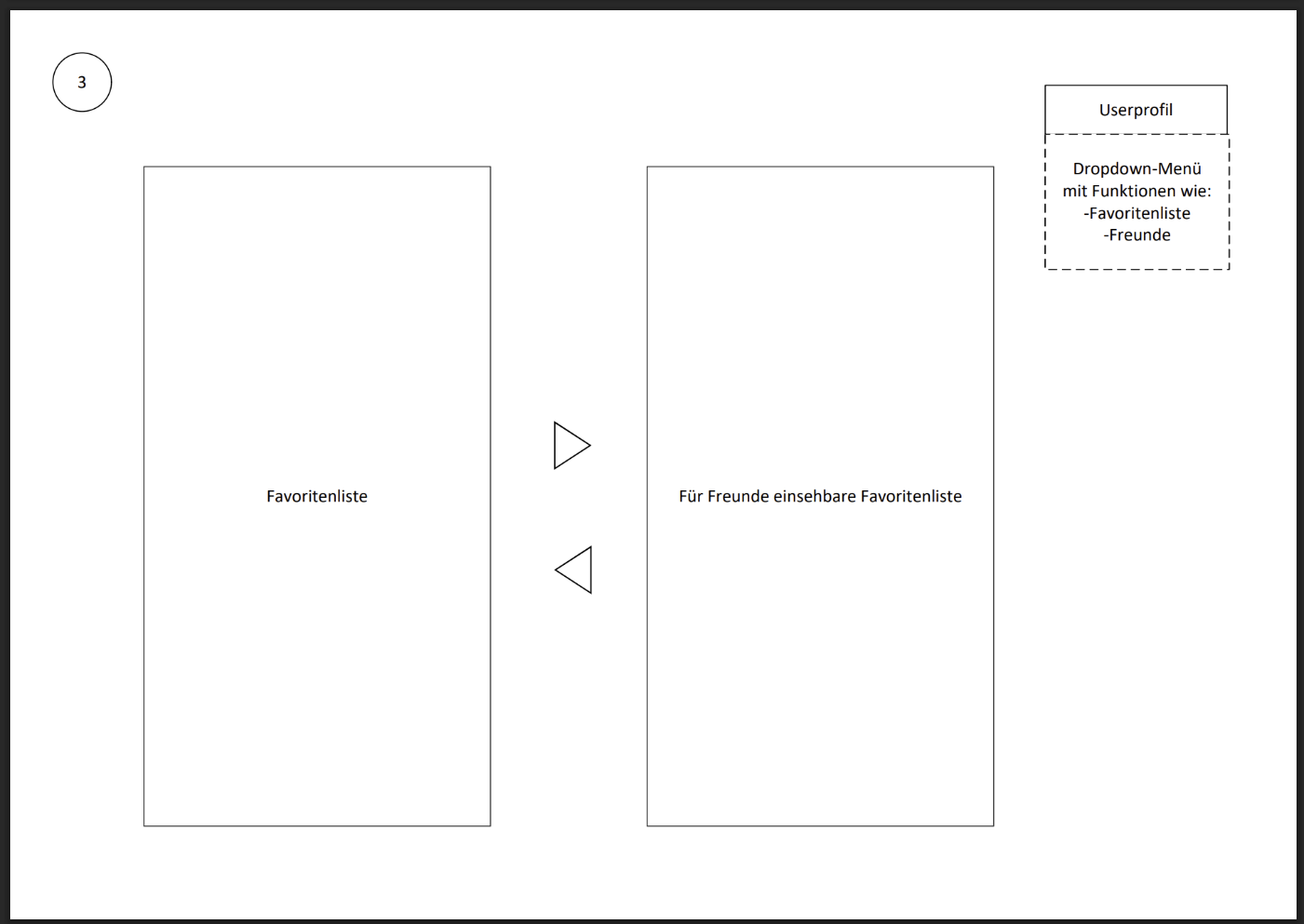
Als letztes hätten wir nun noch das Fenster, bei dem bereits als Favoriten gewählte Filme angezeigt werden. Hierbei gibt es lediglich links ein Feld, bei dem diese eingesehen werden können. Rechts besteht eine weitere Ausgabe, in der Filme gelistet sind, welche wir persönlich, also unabhängig des Algorithmus, unseren Freunden empfehlen und teilen möchten. Das Drop-Down-Menü beidem die Fenster und Aktionen gewechselt werden kann, behält seinen Platz rechts oben im Eck.

Abbildung 5: Favoriten-Fenster

## Zeitplan

Um eine bessere Struktur und auch Übersicht über unser Projekt zu erhalten, haben wir uns einen groben Zeitplan erstellt. Dieser wurde immer wieder überprüft und kontrolliert, ob die Ziele eingehalten und die Aufgaben erledigt sind. Außerdem haben wir uns in regelmäßigen Abständen gemeinsam getroffen, um an unserem Projekt weiterzuarbeiten.   
Ist bei einer/einem ein Problem aufgetreten, für das keine Lösung gefunden wurde, so haben wir uns immer geholfen, auch unabhängig von unseren Treffen.

Auf der nächsten Seite kann unser Zeitplan eingesehen werden:

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

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 6: Zeitplan des Projektes "Filmempfehlungssystem"

# 3. Architektur- & Systemdesign

## MVC Architektur Model-View-Controller

Einst für die Programmiersprache „Smalltalk-76“ in den späten 70ern vom Norweger Trygve Reenskaug entwickelt, ist die MVC-Architektur heute eine der verbreitetsten Möglichkeiten, den Code einer Applikation zu strukturieren. Die MVC-Struktur besteht aus drei Komponenten: Model, Controller und View.

Model:  
Das Model beinhaltet die eigentliche Logik des Programms. Neben dem Ausführen von Methoden, greift diese Komponente auf Datenquellen wie Datenbanken und APIs zu. Die dort gespeicherten Informationen werden vom Model verarbeitet und aktualisiert.

View:  
In der View-Klasse sind alle Methoden zum Aufbau einer grafischen Oberfläche implementiert. In ihr ist das Layout der Anzeigeelemente wie Buttons und Listen festgelegt. Außerdem werden in Ihr Usereingaben entgegengenommen und an den Controller weitergeleitet. Sie dient der Visualisierung der von Model aufbereiteten Daten. In der modernen MVC-Architektur kommuniziert die View-Klasse nicht mehr direkt mit dem Model.

Controller:  
Der Controller bildet das Bindeglied der MVC-Architektur. Er leitet die Useranfragen und eingaben an das Model. Meldet die Model-Klasse das Daten aktualisiert wurden, übergibt er den Befehl zur Aktualisierung der Anzeige an View. Außerdem wird im Controller die App gestartet, hier werden die Initialisierungen durchgeführt.

In der ursprünglichen Version kommunizierten diese Komponenten untereinander.

Ein Bild, das Text, Screenshot, Schrift, Rechteck enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 7: MVC-Flussdiagramm - ursprüngliche Version

In der heutigen Programmentwicklung wird versucht, Programme und Klassen möglichst modular anzulegen. Aus diesem Grund wurde die MVC-Architektur noch einmal optimiert. In der so genannten Apple MVC findet keine direkte Kommunikation zwischen Model und View mehr statt. Das macht die Programmstruktur noch übersichtlicher und ermöglicht es die einzelnen Klassen für sich zu testen, zu duplizieren und in anderen Projekten wieder zu verwenden. Das spart nicht nur Zeit, sondern auch Kosten.

Ein Bild, das Text, Screenshot, Rechteck, Klebezettel enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 8: MVC-Flussdiagramm - heutige Definition

Ein weiterer Vorteil dieser Struktur ist ihre Einfachheit, sie ist auch für Programmeinsteiger gut verständlich und erleichtert spätere Anpassungen. Durch die klare Einteilung fällt die Aufgabenzuweisung leicht, was zu einem übersichtlichen und *wartbaren* Code führt. Außerdem reduziert sie das Risiko von Fehlern erheblich.

Bei der Implementierung ist es jedoch entscheidend, dieses Schema konsequent einzuhalten. Andernfalls können die Vorteile dieser Architektur nicht zum Tragen kommen. Besonders häufig tritt das Problem auf, dass in der MVC-Architektur, Controller und View sehr eng miteinander verflochten werden. In einer modularen Lösung sollte die Kopplung jedoch stets möglichst lose gehalten werden. Eine enge Verknüpfung erschwert nicht nur die Wiederverwendbarkeit der Komponenten, sondern auch das Debugging erheblich.

Wir haben uns für die MVC-Struktur entschieden, da sie auf uns sehr intuitiv wirkt und es die Zusammenarbeit von mehreren Entwicklern an einem gemeinsamen Code erheblich erleichtert. Durch die klare Trennung der Komponenten konnten wir die Aufgaben unter uns gut aufteilen. Zudem ermöglicht uns die Architektur, die Applikation zu einem späteren Zeitpunkt zu erweitern – wie im Kapitel *„Zukünftiges“* genauer beschrieben. Mit der MVC-Architektur schaffen wir eine gute Grundlage, auf welche wir unser Programm weiter aufbauen können.

Vorteile:

Gute für den Einstieg in die Programm Architektur.-> simpel

Wenigere code benötigt als in anderen Architekturen -> übersichtlicher

Klare Trennung der aufgaben Bereiche, da jede Komponente ihre eigene Zuständigkeit hat.

Sehr weit verbreitet, um nicht zu sagen „gängigste“ Architektur in Neuentwicklungen von Mobile App.

Nachteile:

Es geschieht schnell, dass in der MVC-Architektur, Controller und View sehr eng miteinander verwoben sind. Das untergräbt ein wichtiges Ziel der MVC-Architektur, die klare Trennung der Zuständigkeiten der modularen Komponenten. Dies erschwert die Wiederverwendung vor allem der Controller Klasse, wenn zum Beispiel Views nicht mehr online, sondern auf dem Desktop anzeigen soll. Muss Oft auch Controller modifiziert werden. In einer Modularen Lösung sollte die Kopplung immer möglichst lose gehalten werden. Außerdem erschwert es das Debuggen, wenn man nicht mehr genau unterscheiden kann, wo der Fehler geschieht.

„Massive View Controller“: Dadurch, dass dem Controller zu viele Aufgabe zugeteilt werden, besteht die Gefahr der Überladung. -> Dem Controller werden Aufgaben zugeteilt, die nicht in seinen Bereich fallen Macht code unübersichtlich, schwerer zu überarbeiten

* Jede Komponente solle nur eine Aufgabe haben: Anzeigen-Steuern-Verarbeiten

## Fertiges Design der Benutzeroberfläche

Ein Bild, das Text, Screenshot, Schrift, Betriebssystem enthält.

KI-generierte Inhalte können fehlerhaft sein.Ein Bild, das Text, Screenshot, Schrift, Betriebssystem enthält.

KI-generierte Inhalte können fehlerhaft sein.Das endgültige Design unserer GUI für das Filmempfehlungssystem unterscheidet sich nicht viel von unserem zuvor skizzierten Aufbau. Wie geplant bestehen die Fenster aus drei verschiedenen Benutzeroberflächen.   
Es beginnt auf der ersten Seite mit dem Login. Hierbei haben wir die Interaktionsfelder lediglich ganz nach oben in die Mitte positioniert. Ansonsten ist es dabeigeblieben, dass wir einmal ein Login als Eingabefeld, sowie ein Registrierungsfeld mit jeweiligen Bestätigungsbutton haben. Als Änderung kam hier hinzu, dass sich ein neues kleines Fenster öffnet, sobald versucht wird einen Namen zu registrieren, welcher bereits vorhanden ist oder der Benutzer noch nicht in der Datenbank existiert.

Abbildung 9: Fehlermeldung - User existiert nicht

Abbildung 10: Login-Maske fertiger GUI

Ein Bild, das Screenshot, Text, Display, Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 11: Fehlermeldung - User existiert bereits

Im Hauptfenster haben wir ganz oben die Suchleiste, in der, in der darunterliegenden Filmliste nach Filmen gesucht werden kann. Direkt daneben sind die Empfehlungen als Ausgabe. Außerdem befindet sich wie in der Planung definiert eine Beschreibung, sowie zwei Buttons für den Trailer und das Hinzufügen der Favoriten. Geändert hat sich leider, dass wir kein Cover der entsprechenden Filme hinzufügen konnten. Dafür hätten wir eine Open Source Datenbank benötigt, die uns die Bilder zur Verfügung stellt. Dies für jeden Film händisch zu suchen wäre sehr zeitaufwendig gewesen, weshalb wir den Fokus lieber auf andere Funktionen und Details legten. Außerdem hat sich geändert, dass wir das Profil des Users nicht anzeigen, hierbei haben wir uns geeinigt, dass direkt zu den Favoriten gewechselt werden kann und nicht erst über ein Drop-Down-Menü.

Abbildung 12: Hauptmenü - Empfehlungen

Ein Bild, das Text, Elektronik, Screenshot, Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

In der letzten Ansicht «Meine Favoriten» ist einheitlich auch kein Profil, stattdessen direkt der Button, der einen zurück ins Hauptmenü leitet. Ansonsten haben wir eine Liste, welche die vom User gewählten Favoriten ausgibt, sowie eine Erweiterung, dass von Ähnlichen Verhaltens-Profilen dessen Favoriten als Empfehlung darstellt.

Darunter erneut das Beschreibungsfeld, mit den zwei Buttons für Favoriten und dem Link zum Trailer auf YouTube. Dies haben wir hinzugefügt, damit aus der Empfehlungsliste ähnlicher User, der Film direkt ausgewählt werden kann und nicht erst nochmal zurück ins Hauptmenü gewechselt werden muss, um dort den Film zu suchen und dessen Informationen zu gelangen.

Abbildung 13: Meine Favoriten & Empfehlungen auf verhaltensähnliche Nutzer

Ein Bild, das Text, Elektronik, Screenshot, Display enthält.

KI-generierte Inhalte können fehlerhaft sein.

# 4. Implementierung / Codierung

## Programmablaufplan

Die genaue Struktur des Codes haben wir übersichtlich in einem Programmablaufplan (PAP) dargestellt, somit wird der Aufbau, sowie unsere Funktionen und deren Aufgaben gut visualisiert. Außerdem haben wir einen gut kommentierten Code im Anhang, bzw. in den Ordner unser Abgabe beigefügt.

Da der Programmablaufplan leider nicht im Ganzen hier eingefügt werden kann (wäre sonst viel zu klein und nicht mehr leserlich), haben wir diesen für die Doku nochmals unterteilt.

Sollte dieser komplett und aneinanderhängend angesehen werden, so wird auch dieser im Ordner beim Code enthalten sein.

Login:

Ein Bild, das Text, Screenshot, Diagramm, Plan enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 14: Programmablaufplan - Login

Main:

Ein Bild, das Text, Diagramm, Screenshot, Plan enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 15: Programmablaufplan - Main

Favoriten:

Ein Bild, das Text, Screenshot, Diagramm, Plan enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 16: Programmablaufplan - Favoriten

## Algorithmus

Wir haben ein Empfehlungssystem entwickelt, das den Content-based Algorithm mit dem User Collaborative Algorithm kombiniert. Unser Ziel ist es, dem eingeloggten Nutzer Filme vorzuschlagen, die auch den Präferenzen anderer ähnlicher Nutzer entsprechen, wobei wir insbesondere auf Übereinstimmungen in den Genre-Kategorien achten.

Das Ganze läuft wie folgt ab:

1.Der Content-based Algorithm:

Um die individuellen Vorlieben der Nutzer zu berücksichtigen und basierend darauf Filmempfehlungen anzuzeigen, verwenden wir einen Inhaltsbasierten Algorithmus. Dies beginnt damit, dass die von dem Nutzer favorisierten Filme in einer Liste, der Favoritenliste gespeichert werden. Danach muss mithilfe des Filmtitels herausgefunden werden, welchen Genres die Favoriten angehören.  
Hierfür wird in der JSON-Datei mit den Filmdaten nach dem Titel des favorisierten Filmes gesucht und dann das dazugehörige Genre ausgelesen. Es ist zu beachten, dass ein Film auch mehrere Genres haben kann. In diesem Fall wird jedes einzelne Genre ausgelesen.  
Dieses Vorgehen wird auf jeden Film aus der Favoritenliste angewendet. Die zugeordneten Genres werden nun in eine neue List, die Genreliste geschrieben. Im darauffolgenden Schritt sollen nun ähnliche Filme herausgefiltert werden, welche dem Nutzer gefallen könnten.  
Dies kann durch einen Genre Counter realisiert werden. Dabei wird gezählt, wie häufig einzelne Genres in der Favoritenliste vertreten sind. Wenn beispielsweise zwei Filme dem Genre „Drama“ zugeordnet sind, wird die Anzahl für „Drama“ entsprechend um zwei erhöht. Falls ein Film mehreren Kategorien angehört, beispielsweise „Action“ und „Drama“, wird für beide Genres der Zähler jeweils um eine Stelle hochgezählt. Anschließend wird geprüft, ob ein Film bereits in der Favoritenliste enthalten ist, wenn das der Fall ist, werden solche Filme nicht erneut als Empfehlung angezeigt.

Um zu ermöglichen die Empfehlungen nach Relevanz zu ordnen, arbeiten wir mit einem Score. Dieser gibt an, wie viele Übereinstimmungen zwischen den bevorzugten Genres des Nutzers und den Filmen aus der allgemeinen JSON-Filmdatenbank existieren. Der Film mit der höchsten Übereinstimmung wird an erster Stelle in der Empfehlungsliste angezeigt. Die resultierenden Empfehlungen werden schließlich in einer Liste gespeichert und dem Nutzer ausgegeben. Das bedeutet im Klartext: Je höher der Score, desto weiter oben erscheint der Film in der Liste.

Durch dieses Verfahren bietet das Empfehlungssystem präzise und personalisierte Vorschläge, die auf den individuellen Filmgeschmack des Nutzers abgestimmt sind.

Ein Bild, das Text, Screenshot, Diagramm, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.Der Algorithmus grafisch dargestellt:

Abbildung 17: Content-based Algorithm

2.Der User Collaborative Algorithm:

Damit den Nutzern zusätzlich Filme empfohlen werden können, welche auch deren Freunden gefallen, verwenden wir einen benutzerbasierenden Algorithmus. Im Code beginnen wir damit, die Daten der Filme und der gesamten Nutzerbasis zu laden und in entsprechende Variablen zu speichern. Anschließend ermitteln wir für jeden Nutzer dessen bevorzugte Genres und speichern diese als separate Listen. Für den eingeloggten Nutzer extrahieren wir dessen Genres und favorisierten Filme. Danach durchlaufen wir alle anderen Nutzer, wobei der eingeloggte Nutzer dabei ausgeschlossen wird. Für jeden dieser Nutzer überprüfen wir die Ähnlichkeit zum eingeloggten Nutzer und fügen diesen Wert einer speziellen Ähnlichkeitsliste hinzu. Mithilfe der Jaccard-Formel können wir die Ähnlichkeitswerte für die Benutzer erstellen und danach ansteigend nach dem Wert sortieren, sodass der ähnlichste Nutzer an erster Stelle steht.   
Die Jaccard-Formel oder auch jaccard simularity dient zur Berechnung von Ähnlichkeiten zweier Mengen. Sie wird definiert als das Verhältnis der Größe der Schnittmenge zur Größe der Vereinigungsmenge dieser beiden Mengen:

J(A, B) = | A∩B | / | A∪B |   
| A∩B | ist die Anzahl der gemeinsamen Elemente (Schnittmenge).  
∣A∪B∣ ist die Anzahl der insgesamt vorhandenen unterschiedlichen Elemente (Vereinigungsmenge).

In unserem System wäre das zum Beispiel, wenn zwei Nutzer folgenden Lieblingsfilmen haben:

Nutzer A: {Romantik, Action, Drama} und   
Nutzer B: {Action, Drama, Thriller}

Hier wäre die Schnittmenge {Action, Drama} und bekommt dadurch die Größe 2 und die Vereinigungsmenge wäre {Romantik, Action, Drama, Thriller}, somit die Größe 4. Wenn man das in die Formel einsetzt, rechnet man 2/4 und bekommt einen Wer von 50% heraus. Dies sagt uns, dass die Jaccard-Ähnlichkeit 0,5 (also 50 %) beträgt.  
Sie kann zwischen 0 und 1 liegen und je höher die Zahl, desto ähnlicher sind die beiden Datensätze.  
Im Anschluss entnehmen wir diesem ähnlichsten Nutzer dann dessen Filmempfehlungen, die anschließend an den Controller zurückgegeben werden.

Der Algorithmus grafisch dargestellt:

Ein Bild, das Text, Screenshot, Diagramm, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 18: User Collaborative Algorithm

# 5. Testen & Debugging

## Testen

## Debugging

Debugging: Fehlerbehebung in Programmen:  
Debugging ist der Prozess, Fehler aus einem Programm zu entfernen. Dabei versucht man, mögliche Fehlerquellen zu identifizieren, diese Fehler anzeigen zu lassen und anschließend zu beheben. Hierfür haben wir unter anderem die klassische print()-Funktion verwendet.  
Allerdings stößt die alleinige Nutzung von print() an ihre Grenzen, insbesondere wenn das Programm nach einem Fehler abstürzt.

Erweiterte Fehlerbehandlung mit try: und excpet:

Um Abstürze zu vermeiden, haben wir häufig die Konstrukte try: und except Exception as e: eingesetzt. Mit diesen Funktionen kann ein Programmabschnitt (im try-Block) „versucht“ werden. Sollte dabei ein Problem auftreten, wird dieses „abgefangen“ (im except-Block). Im except-Block können wir den Fehler dann mit print(e) ausgeben. Der große Vorteil dieser Methode ist, dass das Programm nicht abstürzt. Es gibt lediglich den Fehler aus und überspringt den restlichen Teil der Funktion, sodass das Programm mit anderen Funktionen fortfahren kann.

Gezieltes Fehlermanagement mit raise Exception(„text“):

Zusätzlich haben wir das raise Exception(„text“) -Statement genutzt. Dies ermöglicht es, Funktionen, bei denen ein bestimmtes, vom Programm nicht unterstütztes Ereignis erwartet wird, abzubrechen und das Problem zu benennen. Dieses Problem kann dann wie oben beschrieben mit try und except erfasst und ausgegeben werden.

Effektivste Fehleranalyse mit dem Python-Debugger:

Sollten all diese Methoden nicht ausreichen, können wir auf den Python-Debugger zurückgreifen. Die Debugger-Übersicht von Visual Studio Code ist hierbei sehr hilfreich. Sie ermöglicht es uns, neben den Zeilennummern einen Breakpunkt zu setzen. An diesem Punkt unterbricht das Programm die Ausführung, und wichtige Daten wie relevante Variablen in der jeweiligen Zeile, können in der Übersicht angezeigt werden.

Beispiel:

Das Video demonstriert, wie der Debugger genutzt werden kann, um Fehler zu finden, die beispielsweise durch fehlerhafte Daten entstehen.



# 6. Bugs & Zukünftiges

## Bugs

Wie im Kapitel Testen & Debugging bereits beschrieben, sind im laufe des Projektes einige Bugs aufgetreten. Diese galt es für uns trotz engem Zeitplan so gut es geht zu beheben.

|  |  |  |
| --- | --- | --- |
| Bugs | Status | Lösung / Kommentar |
| Doppelte Filme in der Datenbank | ✓ | Da wir die Filmliste zuerst mittels KI generiert hatten, waren einige Filme doppelt in der Liste vorhanden. Als Lösung haben wir es mit einer API von einer OpenSource Film Datenbank die Filmliste generiert. |
| Auf der GUI haben wir teils englische und teils deutsche | ✓ | Untereinander Einigung, auf welcher Sprache wir alles ausgeben und dass wir diese einheitlich abändern bzw. weiterarbeiten |
| Registrierung funktioniert nicht | ✓ | User.json Datei musste nach einer Registrierung neu geladen werden, sodass die Überprüfung bei der darauffolgenden Anmeldung auch den neuen User in er Datei findet. |
| Schaut man sich eine Filmbeschreibung aus der Filmliste an und klickt anschließend auf einen Film in der Empfehlungsliste, so erscheint keine Beschreibung. Erst beim Wählen eines anderen Filmes erscheint die entsprechende Beschreibung. | ✓ | Hier lag der Fehler vor, dass bei der Initialisierung die Variable nicht als Dictionary deklariert wurde. Nachdem wir dies angepasst haben, war auch dieser Bug behoben. |
| Film kann zunächst aus den Empfehlungen nicht zu Favoriten hinzugefügt werden | ✓ | Der Button wurde zwar angezeigt, jedoch nur ausgegraut. Hierbei mussten wir diesen lediglich auch für die Favoritenliste wieder aktivieren. |
| Zurück-Button von Favoriten zum Hauptmenü ist im Titel drin | ✓ | Verschieben des Buttons, sodass dieser nun vollständig rechts neben dem Titel angezeigt wird. |
| Titel aus den Favoriten werden auch in den Empfehlungen wiedergegeben | ✓ | Im Code wird überprüft, ob die Titel, welcher unser Algorithmus empfehlen würde, schon bereits in den Favoriten markiert wurde. Ist dies der Fall, so wird der Titel übersprungen und nicht mehr ausgegeben. |
| Mehrfache gleiche Benutzernamen Registrierungen sind möglich | ✓ | Auch hier wird wieder überprüft, ob der gewünschte Name, welcher neu angelegt werden soll bereits in der User.json Datei vorhanden ist. Ist der Name bereits angelegt, soll eine Fehlermeldung erscheinen. |
| Bei doppelter Registrierung öffnet sich ein Fenster, dass dieser Name bereits verlegt ist, allerdings erscheint der Text als Fenstername und nicht innerhalb dessen (siehe Bild 19) | ✓ | Lösung beschreiben fehlt noch |
| Film kann mehrfach zu Favoriten hinzugefügt werden | ✓ | Eine weitere Schleife wurde hinzugefügt, die die Titel überprüft, ob der Titel nicht bereits in den Favoriten vorhanden ist. Somit wird der Titel dennoch nur einmal unter Favoriten gelistet. |
| Fenstergröße der Beschreibung variiert anhand der Größe des Textes (siehe Bild 20 & 21) | 🗶 |  |
| Filmbeschreibungen bei wenigen Filmen fehlen | 🗶 |  |

Beispiel eines noch nicht behobenen Bugs:   
Es wird versucht einen Benutzer zu registrieren, welcher bereits in unserer Liste existiert. Somit ist dies nicht mehr möglich und es öffnet sich ein Fenster, indem die Fehlermeldung erscheinen soll. Das Problem ist nur, dass der Text bisher noch nicht ausgegeben wird.

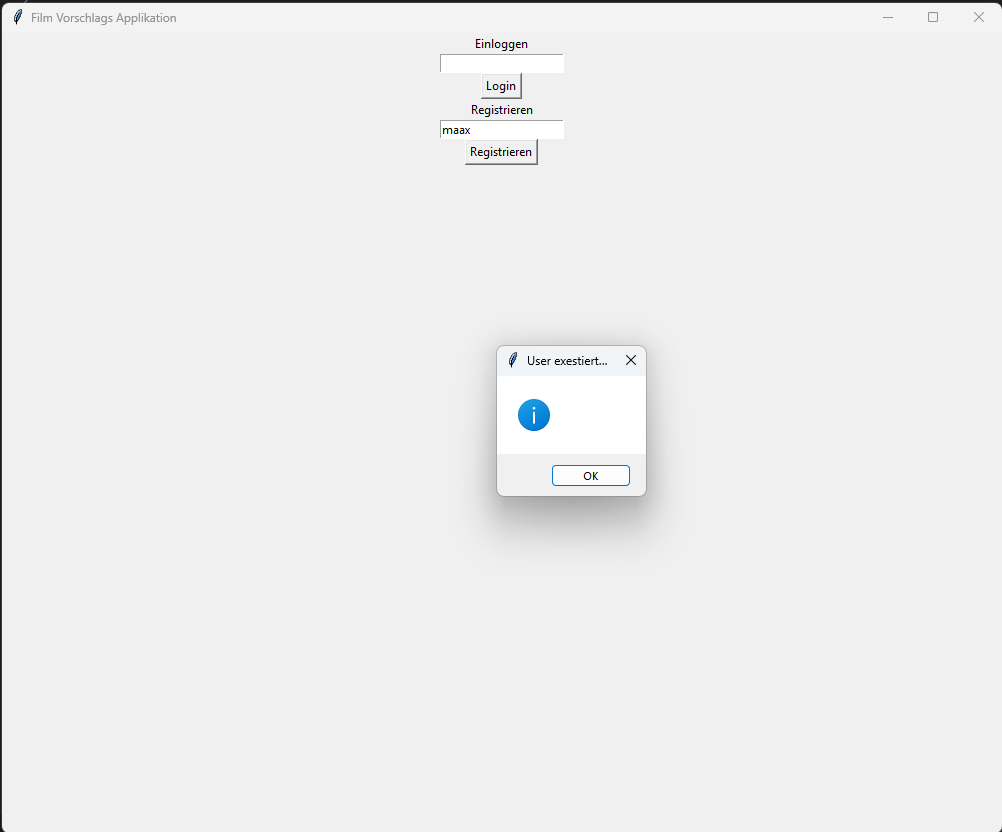
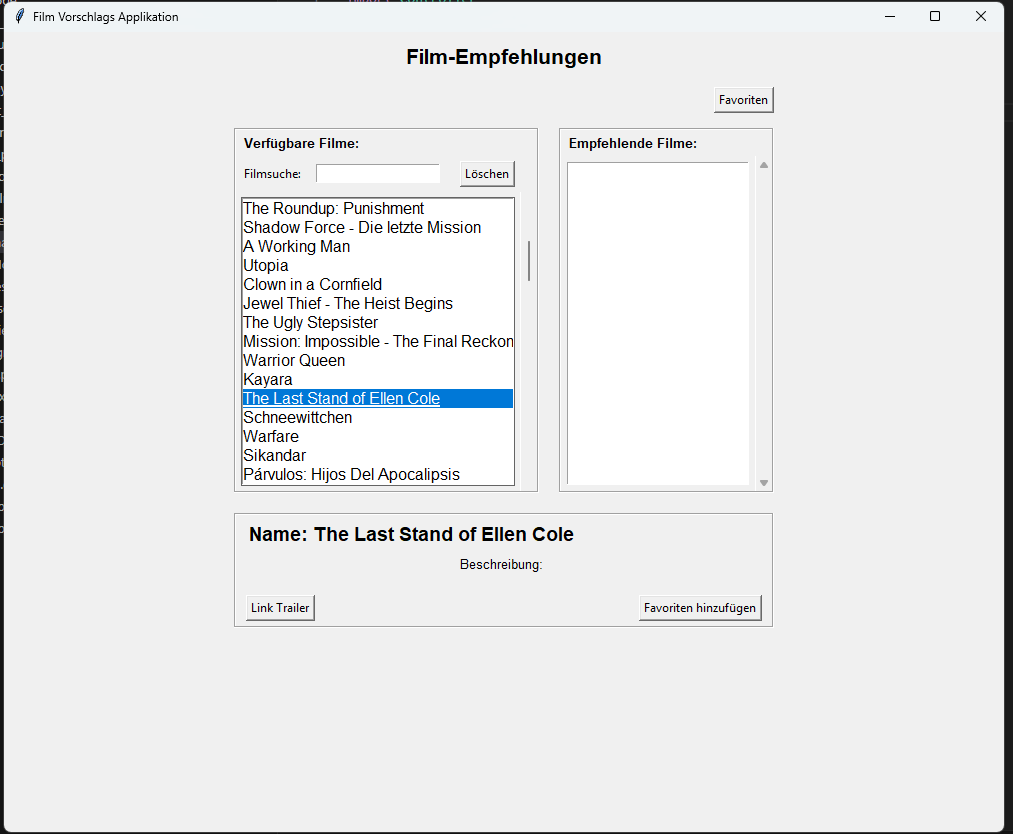


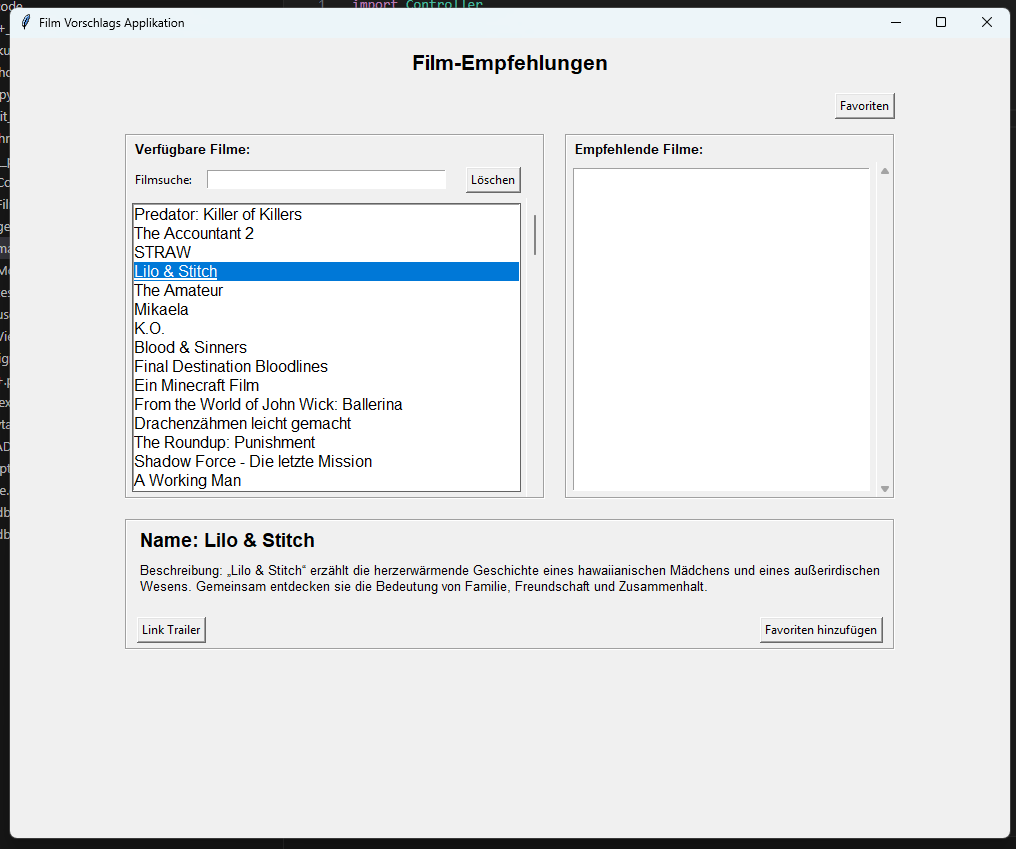
Abbildung 19: Bug bei der Fehlermeldung doppelter Benutzernamen Registrierung

Ein weiteres Beispiel wäre, dass sich die Textausgabe, in der sich die Beschreibung befindet, je nach Länge des Textes sich automatisch anpasst und somit variiert. Dies verdeutlichen wir mit einem weiteren Bug, dass ein paar Filme keine Beschreibung haben. Hierbei sieht man am deutlichsten, dass die Felder sich automatisch etwas anpassen. Positiv ist allerdings, dass sich die Fenster darüber an der gleichen Breite orientieren und somit nicht komisch aussieht:



Großer Abstand

Abbildung 20: Bug - Beschreibungsfeld variiert -> großer Abstand



Kurzer   
Abstand

Abbildung 21:: Bug - Beschreibungsfeld variiert -> kleiner Abstand

## Zukünftiges

Für die zukünftige Entwicklung unseres Projektes konzentrieren wir uns auf die Optimierung und Erweiterung des Systems. Zunächst würden wir bestehende Bugs beheben, um das Programm möglichst zuverlässig und fehlerfrei zu machen.

Des Weiteren würden wir anstatt der JSON-Dateien eine richtige Datenbank (z.B. SQL) implementieren. Dies würde eine automatisierte Aktualisierung der Filmdaten ermöglichen, um den Nutzern stets die neusten Informationen zu garantieren. Außerdem würde es Anpassungen des Programms vereinfachen, da wir Informationsneuheiten nicht mehr manuell eintragen müssten. Ein zusätzlicher Vorteil der Verwendung einer Datenbank liegt im Speicherplatz, da JSON-Dateien aufgrund ihrer alphanummerischen Darstellung mehr Speicherplatz beanspruchen als binäre Datenformate. Ein weiterer Nebeneffekt wäre, dass wir dadurch die Performance unseres Programms verbessern. Durch die Umstellung kann der Speicherbedarf reduziert und die Performance des Programms wesentlich verbessert werden.

Darüber hinaus ermöglicht eine Datenbank mehreren Benutzern gleichzeitig auf die Informationen zuzugreifen, weshalb es das System vielseitiger nutzbar und noch benutzerfreundlicher macht. Zusätzlich ist geplant zu den jeweiligen Filmen auch die Cover ausgeben zu können

Eine weitere geplante Verbesserung wäre die Integration sozialer Funktionen. Nutzer sollen die Möglichkeit haben Freunde zu „adden“, wodurch auch Empfehlungen basierend auf den Vorlieben von Freunden erscheinen. Dadurch entstehen noch individuellere Empfehlungen.

Um die Reichweite und Zugänglichkeit haben wir uns überlegt unser Empfehlungssystem so zu erweitern, damit wir es plattformübergreifend zur Verfügung stellen können. Das bedeutet, dass dem Benutzer nicht nur Filmempfehlungen von einem bestimmten Streamingdienst angezeigt werden, sondern von vielen verschiedenen. Dabei soll natürlich auch deutlich sein, auf welcher Plattform der Film angeschaut werden kann.

Für zukünftige, größere Nutzerdatenbanken planen wir den Einsatz von Machine Learning (ML). Diese Technologie könnten uns dabei helfen, Informationen effizienter und schneller zu verarbeiten. Zum jetzigen Zeitpunkt wäre der Aufwand für die Implementierung von ML – einschließlich zusätzlicher Treiber, Software und Bibliotheken – jedoch unverhältnismäßig hoch und würde sich angesichts unserer aktuellen Datenbankgröße nicht lohnen.

Ähnliches gilt für die Integration einer dedizierten Datenbank wie SQL. Während SQL-Datenbanken für große Datenmengen ideal sind, hätte ihre Einbindung in unserem derzeitigen Stadium die Entwicklung erschwert. Für Entwickler kann es aufwendiger sein, SQL-Datenbankdateien zu lesen und Fehler direkt in der Datenbank zu beheben. Dies hätte wiederum Konsequenzen für die Entwicklung und Fehlerbehebung des Algorithmus haben können. Eine SQL-Datenbank wird erst dann sinnvoll, wenn unsere Datenmengen signifikant anwachsen.

# 7. Reflexion & Nachbereitung

# 8. Quellenverzeichnis

|  |  |
| --- | --- |
| [1] | R. F. García, iOS Architecture Patterns: MVC, MVP, MVVM, VIPER, and VIP in Swift, Apress, 2023. |
| [2] | g. Education, «Vorgehensmodell MVC - Model, View, Controller kurz erklärt,» https://www.youtube.com/watch?v=VGQMQUa\_o5o, 2021. |
| [3] | T. M. Tutorials, «Softwareengineering Tutorial #22 - Model View Controller MVC,» https://www.youtube.com/watch?v=xK-nj3J1zXA, 2018. |
| [4] | W. D. Simplified, «MVC Explained in 4 Minutes,» https://www.youtube.com/watch?v=DUg2SWWK18I, 2019. |
| [5] | «PythonGUIs - Create a GUI Using Tkinter and Python,» 2022. [Online]. Available: https://www.pythonguis.com/tutorials/create-gui-tkinter/#what-is-tkinter. [Zugriff am 14 06 2025]. |
| [6] | «Python - tkinter — Python interface to Tcl/Tk,» [Online]. Available: https://docs.python.org/3/library/tkinter.html. [Zugriff am 14 06 2025]. |
| [7] | «Python Tutorial - Tkinter Tutorial,» 2021. [Online]. Available: https://www.pythontutorial.net/tkinter/. [Zugriff am 14 06 2025]. |
| [8] | «Python-lernen.de - GUI Programmierung über tkinter in Python,» 2025. [Online]. Available: https://www.python-lernen.de/tkinter-gui.htm. [Zugriff am 14 06 2025]. |
| [9] | «heise online - Was sind JSON-Dateien und wie lassen sie sich öffnen?,» [Online]. Available: https://www.heise.de/tipps-tricks/Was-sind-JSON-Dateien-und-wie-lassen-sie-sich-oeffnen-9302292.html. [Zugriff am 13 06 2025]. |
| [10] | «Firebase,» [Online]. Available: https://firebase.google.com/. [Zugriff am 31 05 2025]. |
| [11] | P. D. O. Krini, *Vorgehen - Informatikprojekt Teil 2,* 2025. |
| [12] | «Statologie - Berechnen des Jaccard-Koeffizienten in Python,» [Online]. Available: https://statologie.de/jaccard-koeffizient-python/. [Zugriff am 06 2025]. |

# 10. Anhang

## Beispiele für ausgefüllte Umfragebögen

**Ein Bild, das Logo, Clipart, Grafiken, Symbol enthält.

KI-generierte Inhalte können fehlerhaft sein.Python Programm**

***Bewertungsbogen für Empfehlungssystem***

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bewertungsinformationen | | | | | | | | | | |
|  | | | | | | | | | | |
| Name: Max \_\_\_\_ | | |  | | | | | | |
| Datum: 13.06.2025 |  | | | | Bewertungszeitraum: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | |  |
|  | | | | | | | | | | |
|  | | | | | | | | | | |
| Programm | | | | | | | | | | |
|  | | | | | | | | | | |
|  | | (5) = ausgezeichnet | | (4) = zufrieden stellend | | (3) =  teils teils | (2) =  weniger zufrieden stellend | (1) =  nicht zufrieden stellend | | |
| Übersichtlichkeit / Darstellung | |  | |  | |  |  |  | | |
| Leicht in der Anwendung | |  | |  | |  |  |  | | |
| Funktionalität: | | (5) = relevant | | (4) = weniger relevant | | (3) =  teils teils | (2) = weniger relevant | (1) = nicht relevant | | |
| Wie relevant waren die Empfehlungen für dich? | |  | |  | |  |  |  | | |
|  | |  | |  | |  |  |  | | |
|  | | (5) = ja | | (4) = oft | | (3) =  teils teils | (2) = selten | (1) = Nein | | |
| Wurden dir oft Filme angezeigt, die du bereits in deinen Favoriten hattest? | |  | |  | |  |  |  | | |
| Würdest du das Empfehlungssystem in Zukunft weiterhin nutzen? | |  | |  | |  |  |  | | |
| Hast du das Gefühl, dass deine Eingaben (z.B. Bewertungen, Klicks) einen Einfluss auf die Empfehlungen hatten? | |  | |  | |  |  |  | | |
|  | | | | | | | | | | |

Was hat dir an der Funktionsweise des Empfehlungssystems besonders gut gefallen?

Einfachheit, Klare Strukturierung, Auflistung

Was hat dir an der Funktionsweise des Empfehlungssystems nicht so gut gefallen oder war verwirrend?

Doppelte Filme, Teils eingeschränkte Funktionsweise, wenn Film z.B. zu Favoriten hinzugefügt wurde

oder auf die Vorschläge geklickt wurde, Registrierung geht nicht

Gab es technische Probleme oder Fehler während der Nutzung? Wenn ja, welche?

Wenn die Beschreibung des gesuchten Films einmal angezeigt wurde und man das erste Mal Film

aus den Vorschlägen nimmt, wird dort die Beschreibung nicht angezeigt, erst als man einen Film aus

den nächsten Vorschlägen auswählt. Der Film kann dann nicht direkt zur Watchlist hinzugefügt

werden, nachdem der erste hinzugefügt wurde und man auf ein Filmvorschlag geklickt hat. Erst

nachdem die Favoriten angeklickt wurden, ist dies wieder möglich. Favoriten „Zurück“ Button ist im

Titel drin. Merkliste? Bereits Titel aus den Favoriten werden in den Vorschlägen erneut angezeigt?

Registrierung funktioniert nicht, userID muss händisch angepasst werden, sonst wird das danach

nicht mehr als Login akzeptiert. Mehrfache gleiche User Registrierung? Prüfung ob User bereits

vorhanden? Gleicher Film kann mehrfach auf die Favoritenliste hinzugefügt werden

Doppelte Filme wie z.B. Lilo & Stitch. Fenster wird öfters mal verschoben bzw. Größe geändert.

Welche Funktionen oder Verbesserungen würdest du dir für das Empfehlungssystem wünschen?

Fehler beheben, größere Filmauswahl, Merkliste aktivieren, oder entfernen.

Login bzw. Registrierungsmaske mit Infotext was falsch war oder gemacht werden muss. Buttons

durch Enter bestätigen Film als Favorit hinzugefügt, Meldung über Bestätigung.

Hast du weitere Anmerkungen, Kommentare oder Feedback, dass du uns mitteilen möchtest?

Super Arbeit mit diesem Wissen und für diese kurze Zeit 😉

*Vielen Dank für deine Teilnahme!!*

**Ein Bild, das Logo, Clipart, Grafiken, Symbol enthält.

KI-generierte Inhalte können fehlerhaft sein.Python Programm**

***Bewertungsbogen für Empfehlungssystem***

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bewertungsinformationen | | | | | | | | | | |
|  | | | | | | | | | | |
| Name: Hanspeter Mirwald | | |  | | | | | | |
| Datum: 15.06.2025 |  | | | | Bewertungszeitraum: 30 min | | | |  |
|  | | | | | | | | | | |
|  | | | | | | | | | | |
| Programm | | | | | | | | | | |
|  | | | | | | | | | | |
|  | | (5) = ausgezeichnet | | (4) = zufrieden stellend | | (3) =  teils teils | (2) =  weniger zufrieden stellend | (1) =  nicht zufrieden stellend | | |
| Übersichtlichkeit / Darstellung | |  | |  | |  |  |  | | |
| Leicht in der Anwendung | |  | |  | |  |  |  | | |
| Funktionalität: | | (5) = relevant | | (4) = weniger relevant | | (3) =  teils teils | (2) = weniger relevant | (1) = nicht relevant | | |
| Wie relevant waren die Empfehlungen für dich? | |  | |  | |  |  |  | | |
|  | |  | |  | |  |  |  | | |
|  | | (5) = ja | | (4) = oft | | (3) =  teils teils | (2) = selten | (1) = Nein | | |
| Wurden dir oft Filme angezeigt, die du bereits in deinen Favoriten hattest? | |  | |  | |  |  |  | | |
| Würdest du das Empfehlungssystem in Zukunft weiterhin nutzen? | |  | |  | |  |  |  | | |
| Hast du das Gefühl, dass deine Eingaben (z.B. Bewertungen, Klicks) einen Einfluss auf die Empfehlungen hatten? | |  | |  | |  |  |  | | |
|  | | | | | | | | | | |

Was hat dir an der Funktionsweise des Empfehlungssystems besonders gut gefallen?

Große Auswahl an Vorschlägen

Was hat dir an der Funktionsweise des Empfehlungssystems nicht so gut gefallen oder war verwirrend?

verwirrende Reihenfolge der Filmliste

es fehlen einige Filmbeschreibungen

Gab es technische Probleme oder Fehler während der Nutzung? Wenn ja, welche?

Nein

Welche Funktionen oder Verbesserungen würdest du dir für das Empfehlungssystem wünschen?

Möglichkeit, nach Genre zu filtern und Sortierung nach Alpahbet

Rückmeldung zur erfolgreichen Registrierung des Benutzers

Hast du weitere Anmerkungen, Kommentare oder Feedback, dass du uns mitteilen möchtest?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Vielen Dank für deine Teilnahme!!*

Inhaltsverzeichnis, Abbildungsverzeichnis aktualisieren und Schriftart, sowie Größe anpassen

## Code

Da der Code nur schwer samt den ganzen Kommentaren in die Dokumentation eingefügt werden kann, werden wir diesen, wie den kompletten Programmablaufplan auch, in den Ordner unserer gesamten Abgabe hinzufügen. Somit kann bei Bedarf unser Code und dessen Kommentare angesehen werden. Gegebenenfalls könnte somit auch unser Filme-Empfehlungssytem live getestet werden.