SPECTRUM MOVIE RECOMMENDATION SYSTEM

System Design Document

Version 1.0

Prepared by Group 11:

Adith S Biju

Manav Maniprasad

Eric Oommen Mathew

12/01/2024

Table of Contents

1. Introduction

The System Design Document (SDD) pertains to the development project known as "Spectrum Movie Recommendation System." Spectrum is envisioned as an innovative solution designed to empower users on the customer platform to efficiently explore content through the assistance of our advanced recommendation system. This software, a Hybrid Recommendation System for Movies, seamlessly combines collaborative and content-based filtering techniques within the realm of web-based recommender systems.

Specifically, Spectrum integrates with the well-established TMDB dataset, utilizing a content filtering approach grounded in trained neural networks that represent individual user preferences. This document outlines the design principles, architecture, and guidelines for the development of Spectrum, emphasizing the significance of factors such as supplementary user and item features on the accuracy of our proposed hybrid recommender system.

To optimize system runtime and uncover latent user and item relationships,the system uses Natural Language Processing (NLP) and cosine similarity to find movies that are similar to the ones users have liked in the past. The system's overarching goal is to address the increasing demand for highly personalized and effective content filtering systems in the face of the vast amount of information available online.

Security and Privacy Considerations:

Spectrum is committed to prioritizing user privacy and data security. Throughout the development process, security measures will be implemented to safeguard user information, ensuring compliance with relevant privacy regulations.

1.1. Purpose of the SDD

The purpose of this SDD is to serve as a comprehensive reference guide for the development team, designers, and stakeholders involved in the Specturm project. Specifically, the document aims to:

- Clarify System Architecture: Clearly articulate the architecture, components, and interactions of the Specturm Movie Recommendation System.
- Guide Hybrid Model Implementation: Offer detailed guidelines on the implementation of the hybrid recommendation system, highlighting the integration of collaborative and content-based filtering techniques.
- Demonstrate Prediction Accuracy: Through various experiments, showcase the influence of supplementary user and item features on the prediction accuracy of the hybrid recommender system.

- Highlight System Optimization: Detail the application of Natural Language Processing (NLP) and cosine similarity to find movies that are similar to the ones users have liked in the past.
- Address Privacy and Security: Outline security measures to ensure the protection of user information and compliance with privacy regulations.

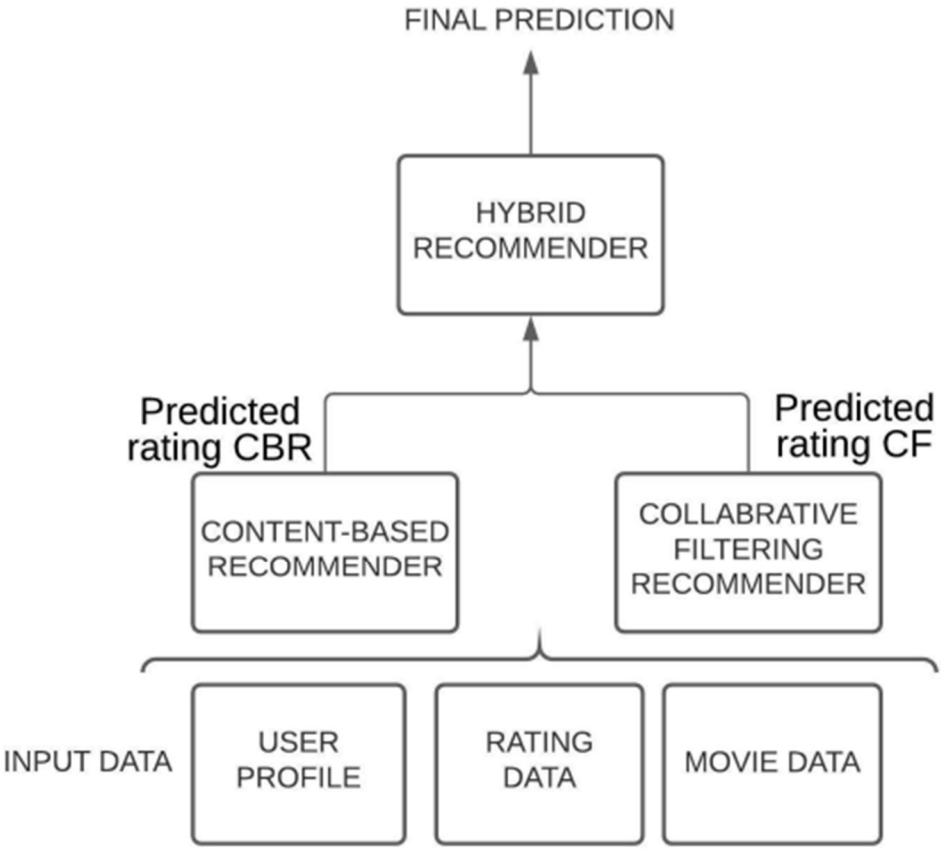
This SDD is tailored to address the unique needs of the Specturm Movie Recommendation System, ensuring that it effectively guides the development process and contributes to the successful implementation of the platform.

2. General Overview and Design Guidelines/Approach

This section describes the principles and strategies to be used as guidelines when designing and implementing our Spectrum Movie Recommendation System.

2.1. General Overview

The Specturm Movie Recommendation System is an innovative platform designed to elevate the user's movie-watching experience by offering personalized recommendations. Employing a Hybrid Recommendation approach, the system seamlessly integrates collaborative and content-based filtering methodologies within a web-based recommender system. Specturm is intricately connected to the TMDB dataset, utilizing advanced neural networks to filter content based on individual user preferences. The design focuses on not only delivering efficient and accurate recommendations but also adapting to evolving user interactions.



2.2. Assumptions/Constraints/Risks

2.2.1. Assumptions

Related Software or Hardware:

Assumption: The system assumes compatibility with modern web browsers.

Dependency: Reliable access to the TMDB dataset for movie information is assumed.

End-User Characteristics:

Assumption: Users will provide accurate information about their movie preferences.

Changes in Functionality:

Assumption: Potential changes in functionality are expected, necessitating adaptability in the system's design.

2.2.2. Constraints

Hardware or Software Environment:
Constraint: The system is designed to operate in a web-based environment.
Impact: Ensures compatibility with mainstream operating systems is assumed.

Performance Requirements:
Constraint: The system must meet predefined performance benchmarks for a seamless user experience.
Impact: Ensures optimal system performance to meet user expectations.
Security Requirements:
Constraint: Adherence to privacy regulations for the protection of user data.
Impact: Imposes strict security measures to safeguard user information.

Data Repository and Distribution:
Constraint: Efficient handling of the TMDB dataset for timely movie updates.
Impact: Ensures the system remains up-to-date with the latest movie information.

Network Communications:
Constraint: Reliable internet connectivity is assumed for accessing external datasets and ensuring system responsiveness.
Impact: Ensures uninterrupted access to external resources for optimal system functionality.

Verification and Validation Requirements:
Constraint: Rigorous testing procedures to validate system accuracy and performance.
Impact: Guarantees the reliability and effectiveness of the Specturm Movie Recommendation System

2.2.3 Risks

1. Uncertain User Adoption:

Risk: There is a risk of uncertain user adoption, potentially impacting the system's overall success.

Mitigation: Conduct user surveys, gather feedback, and implement an intuitive user interface to enhance user engagement.

2. Prediction Accuracy Challenges:

Risk: Achieving high prediction accuracy may pose challenges due to evolving user preferences.

Mitigation: Implement continuous learning algorithms, regularly update recommendation models, and encourage user feedback.

3. Data Security Concerns:

Risk: Potential risks related to data security and privacy breaches.

Mitigation: Implement robust encryption protocols, adhere to privacy regulations, and conduct regular security audits.

4. Dependence on External Datasets:

Risk: Dependency on external datasets, such as TMDB, may lead to disruptions or delays in updates.

Mitigation: Establish contingency plans, implement efficient data synchronization processes, and explore alternative data sources.

5. Technological Changes:

Risk: Rapid technological changes may impact the compatibility and performance of the system.

Mitigation: Stay abreast of technological advancements, plan for regular system updates, and maintain flexibility in the system architecture.

These assumptions, constraints, and risks provide essential context for the design and implementation of the Specturm Movie Recommendation System, guiding the development team in creating a robust, adaptable, and user-centric platform.

3. Design Considerations

3.1. Goals and Guidelines

In the development of Spectrum, our movie recommendation chatbot, our primary goal is to deliver a user-centric experience that prioritizes the seamless and enjoyable interaction of users with the system. We aim to create an interface that is intuitive and user-friendly, fostering engagement and long-term usage. Another crucial goal is to achieve performance excellence by optimizing the system for fast response times. This emphasis on speed is driven by the understanding that quick interactions significantly contribute to user satisfaction. Additionally, we aspire to maintain cross-platform consistency, ensuring that the Spectrum experience remains cohesive across various devices and platforms. This goal is essential to provide users with a uniform and reliable experience, regardless of how they access the chatbot. Ultimately, these goals collectively guide the design and development process, with a focus on user satisfaction, performance optimization, and consistent accessibility.

3.2. Development Methods & Contingencies

In the development of Spectrum, our movie recommendation chatbot, we have chosen to embrace an Agile development approach, which prioritizes adaptability and collaboration throughout the software development life cycle. This methodology allows for iterative development cycles, enabling us to respond effectively to changing requirements and incorporate valuable feedback from stakeholders. As part of our development methodology, we are adhering to object-oriented design principles and leveraging Unified Modeling Language (UML) for visualizing and documenting the software architecture. Object-oriented design provides a modular and scalable foundation, promoting code reuse and maintainability.

Considering potential contingencies, we have identified key areas where challenges may arise, and we have established contingency plans to address them. In the event of external interface challenges, we are prepared to explore alternative communication methods to maintain project continuity. Additionally, if architecture stability becomes a concern, we have mechanisms in place to regularly assess and adjust our technological choices to ensure a stable foundation for development. Recognizing the dynamic nature of software development, our contingency plans include flexibility to accommodate changing requirements by maintaining open lines of communication with stakeholders and fostering a responsive development team.

Resource constraints are acknowledged, and we have contingency plans for effective resource allocation and task prioritization to mitigate any potential impacts. Furthermore, in the face of unforeseen technology emergencies, a rapid response plan is established, allowing for quick adaptation, alternative technology exploration, or the implementation of temporary solutions while addressing the underlying issues.

These development methods and contingencies collectively form a robust framework that not only aligns with industry best practices but also equips the Spectrum project with the flexibility and resilience needed to navigate the complexities of software development successfully.

3.3. Architectural Strategies

- Programming Language & Libraries: The project will use Python libraries, JavaScript, and HTML for implementation.
- Software Reuse: The system will utilize Matrix factorization, Singular Value Decomposition (SVD), and other methods for recommendation algorithms.
- Future Enhancements: Not explicitly mentioned, but the system's performance and recommendation accuracy could be improved over time.
- User Interface Paradigm: The system will have a search function and a link for content providers to add new content.
- Hardware/Software Interfaces: The system will be accessible via any internet-connected hardware and compatible with browsers like Internet Explorer, Mozilla, and Google Chrome1.
- Error Recovery & Data Persistence: The system will use secure sockets for transactions and perform hourly backups to prevent data loss.
- SQL for distributed data management.

3.4 Performance Engineering

In the development of Spectrum, our movie recommendation chatbot, performance engineering plays a pivotal role in shaping the system's design to meet defined scalability and responsiveness expectations. The incorporation of performance requirements into the system's architecture is a foundational aspect, aligning closely with the guidelines outlined in Sections 2.0 and 2.1.1 of the CMS Performance Test Plan and Results Template.

Performance requirements, as outlined in the Requirements Document, have been meticulously analyzed and translated into key design considerations. To address scalability expectations, we have embraced a microservices architecture. This architectural decision allows for independent scaling of specific functionalities, ensuring the system can efficiently handle increased workloads by deploying additional instances of relevant services. Microservices enhance scalability by isolating components and enabling horizontal scaling, perfectly aligning with the scalability requirements specified in the performance criteria.

For responsiveness expectations, particularly in user interactions with the chatbot, our design focuses on implementing efficient algorithms for movie recommendations and leveraging caching strategies to minimize response times. The technology stack, featuring Python for backend development and React for the frontend, has been chosen with a deliberate emphasis on responsiveness. These technologies are known for their efficiency in handling user requests and delivering dynamic user interfaces, meeting the requirements for prompt and engaging user experiences.

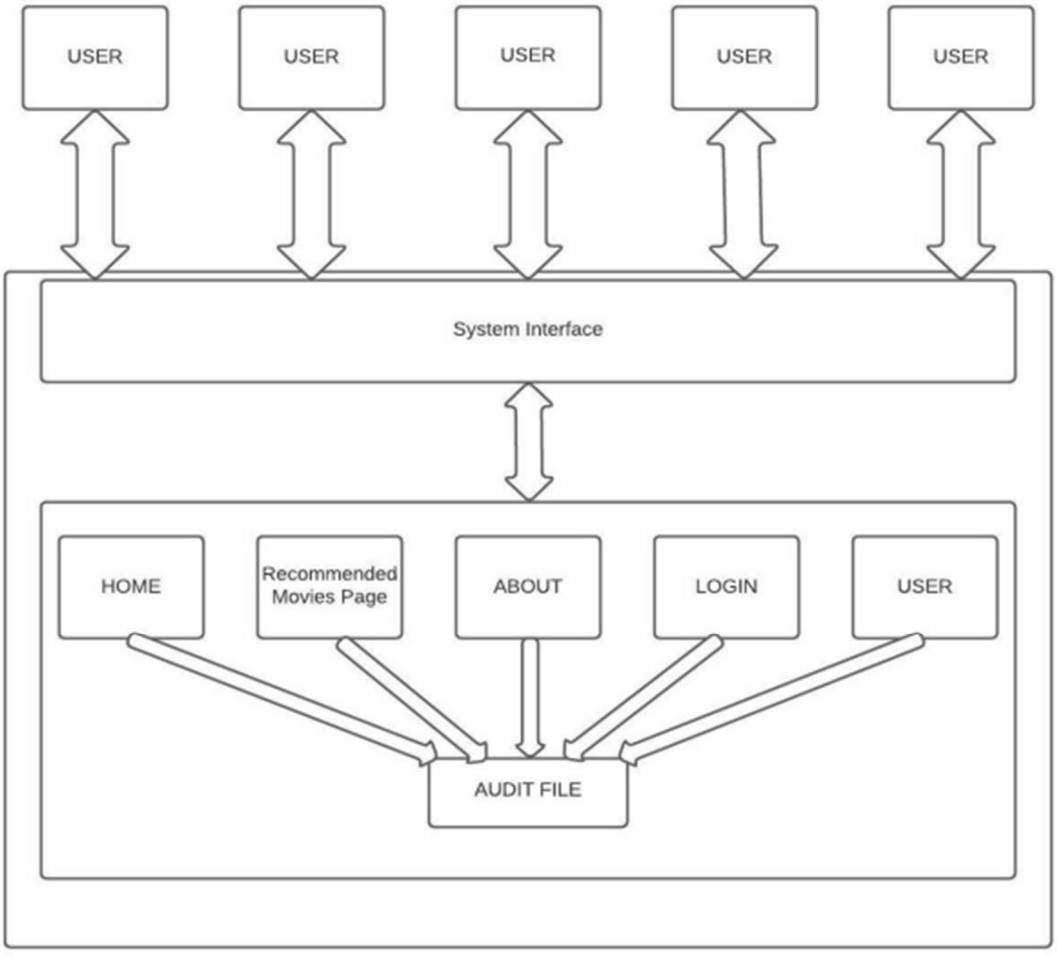
As part of our commitment to performance engineering, we have initiated the development of Production Load Model(s) in preparation for performance testing. This involves constructing realistic load scenarios based on anticipated user behaviors and usage patterns. The load model encompasses various factors, including the number of concurrent users, types of interactions, and the frequency of requests, providing a comprehensive simulation of real-world usage scenarios. This proactive approach ensures that the system is rigorously tested under conditions that closely resemble actual user engagement.

By integrating performance requirements into the system's design and preparing a detailed Production Load Model, our objective is to validate the system's capability to meet scalability and responsiveness expectations. This not only ensures compliance with performance criteria but also enables us to identify and address potential performance bottlenecks proactively. In doing so, Spectrum is poised to deliver a robust and high-performing movie recommendation chatbot that provides a seamless and responsive experience to its users.

4. System Architecture and Architecture Design

The Specturm Movie Recommendation System employs a modular and scalable system architecture designed to deliver personalized movie recommendations efficiently. At the top-most level, the system is decomposed into key components/subsystems, each assigned specific responsibilities to collectively provide the desired functionality. The major responsibilities include user profile management, collaborative filtering, content-based filtering, and data synchronization with external databases, notably TMDB.

4.1. Logical View



4.2. Hardware Architecture

1. Central Processing Unit (CPU):

The brain of the system, responsible for fetching, decoding, and executing instructions. It consists of:

- Control Unit (CU): Decodes instructions and directs other components.
- Arithmetic Logic Unit (ALU): Performs mathematical and logical operations.
- Registers: Temporary storage for data and instructions.

2.Memory:

- Stores data and instructions that the CPU needs to access quickly.
- Random Access Memory (RAM): Volatile memory, loses data when powered off. Used for active programs and data.
- Read-Only Memory (ROM): Non-volatile memory, retains data even when powered off. Stores permanent instructions like the BIOS.

3. Storage:

- Holds data and programs permanently, even when the computer is turned off. Examples include:
- Hard Disk Drive (HDD): Traditional spinning disk storage, slower but cheaper.
- Solid State Drive (SSD): Flash-based storage, faster but more expensive.

4.3. Software Architecture

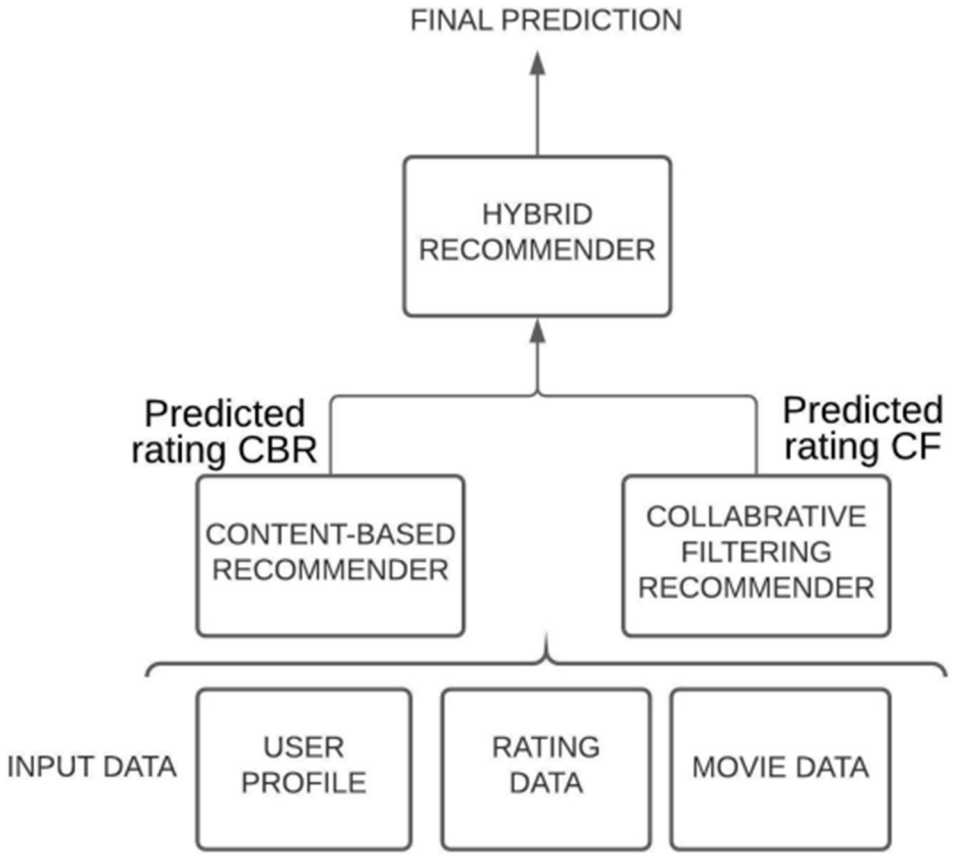
Operating System: The system is designed to run on the Windows operating system for its support and user-friendliness.

Database: SQL database is chosen to store movie details and user preferences.

Programming Languages & Libraries: The project will utilize Python libraries, JavaScript, and HTML.

4.4. System Architecture Diagram

This is how our recommender predits the data based on the necessary user data,rating data and movie data and gives the user a good recommendation.Content based and collaborative filtering recommenders are used as a hybrid recommender to give recommendations.



5. System Design

5.1. Business Requirements

The key business requirements for the Movie Recommendation System include:

- Real-Time Prediction Accuracy: The system must provide accurate movie recommendations in real-time to enhance user experience.

- Ease of Use: The system should be user-friendly, ensuring that users can easily interact with the platform and receive personalized movie suggestions.

- Adaptability to Data Increase: The database design must be scalable to accommodate an increase in data volume for prediction, ensuring continued system performance and accuracy.

5.2. Database Design

This is how we have designed our database in our system.

Data dictionary

	FIELD	TYPE	NULL
User	username	varchar(100)	NO
	email	varchar(100)	NO
	password	varchar(100)	NO
	profile_image	Image	YES
	user_id	int	NO

	FIELD	TYPE	NULL
Ratings	user_id	int	NO
	movie_id	int	NO
	movie_liked/disliked	int	NO

	FIELD	TYPE	NULL
Movies	genres	varchar(100)	NO
	id	int	NO
	title	varchar(100)	NO
	average_rating	float	NO
	Cast	varchar(100)	YES

5.2.1. Data Objects and Resultant Data Structures

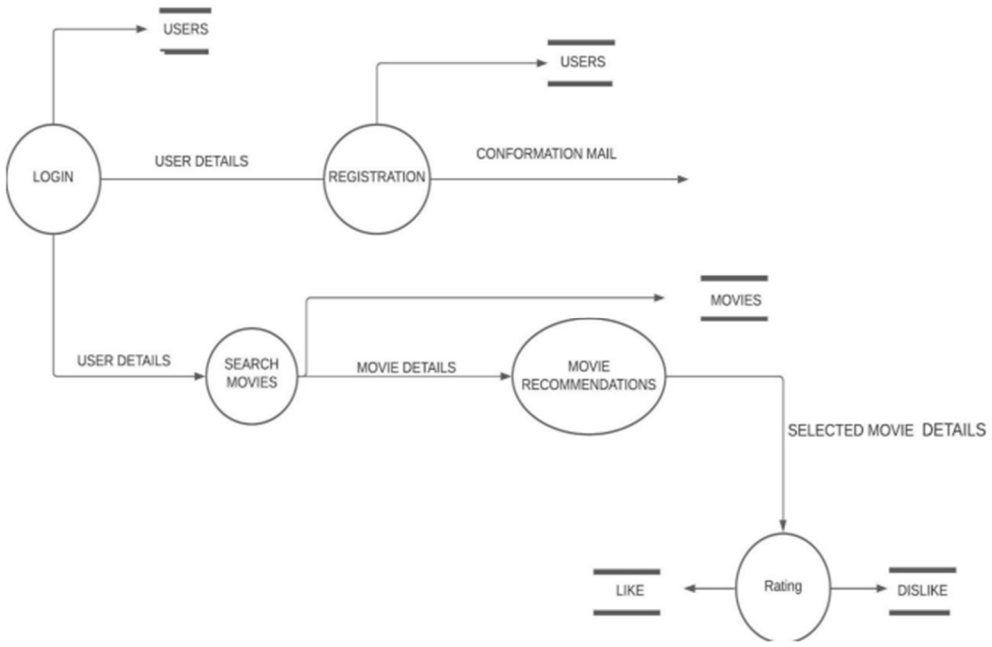
Data Structure: Movie Records

Storage Format: SQL Database Table

Functions:

- Input: Movie Dataset
- Output: Movie Details required for our prediction and recommendation

5.2.2 Inputs&Outputs



5.3 User Interface Design

UI is designed according to the UI design principles.

1. The structured principle: UI is organized in such a way that related things are combined together and unrelated things are separated.

2. The simplicity principle: It is easy to follow the provided interface.In the case of a mistake.,the system displays an error message.

3. The visibility principle: All system functions are available through the UI. It does not overwhelm users with too many alternatives.

4. The feedback principle: Through the system of messages, the design keeps users informed of actions, errors, or exceptions.

5. The reuse principle: In design, the same names were used to perform the same operations with different objects in order to reduce ambiguity

6. Operational Scenarios

Scenario 1: User Registration and Profile Setup

Description:

Event: A new user accesses the Specturm platform.

Action: User initiates the registration process.

Stimuli: The system prompts the user for necessary information (name, email, password).

Interaction: User provides required information and submits the registration form.

Information: The system validates user details and creates a new user profile.

Outcome: User receives a confirmation message, and the profile is set up.

Estimate:

Size: Small (user registration details).

Frequency: Average – multiple new users daily.

Scenario 2: User Movie Rating

Description:

Event: Registered user logs into the Specturm platform.

Action: User navigates to a movie and provides a rating.

Stimuli: The system updates user preferences based on the rating.

Interaction: User submits the rating and potentially adds a review.

Information: The system processes the rating, updates collaborative filtering data.

Outcome: User receives a confirmation, and the movie is added to the user's preferences.

Estimate:

Size: Small (user rating data).

Frequency: Average – multiple ratings per user weekly.

Scenario 3: Movie Recommendation Retrieval

Description:

Event: User logs in and requests movie recommendations.

Action: The system evaluates user preferences (collaborative and content-based filtering).

Stimuli: User preferences trigger the recommendation algorithms.

Interaction: System retrieves and displays recommended movies.

Information: System fetches movie data from the TMDB dataset.

Outcome: User views personalized movie recommendations.

Estimate:

Size: Medium (recommended movie details).

Frequency: Average – multiple requests per user weekly.

Scenario 4: Data Synchronization with TMDB

Description:

Event: System triggers a scheduled update to synchronize with TMDB.

Action: External Data Interface connects to TMDB.

Stimuli: System initiates batch transfers to update movie details.

Interaction: TMDB provides the latest movie information.

Information: System updates its movie database with fresh data.

Outcome: Specturm's movie database is synchronized with TMDB.

Estimate:

Size: Large (complete movie dataset).

Frequency: Periodic – scheduled updates daily.

Scenario 5: User Feedback Submission

Description:

Event: User provides feedback on a recommended movie.

7. Detailed Design

This section delves into the specific hardware and software components of the Specturm Movie Recommendation System, outlining their functionalities and interactions to create a functional product.

7.1. Hardware Detailed Design

Since the application must run over the internet, all the hardware required to be connected to the internet will be a hardware interface for the system. As for e.g. Modem, WAN – LAN, Ethernet Cross-Cable.

7.1.1 Server Components

- Power Input Requirements: 110-240V AC, 50-60Hz
- Signal Impedances and Logic States: Standard TTL levels
- Connectors: Ethernet (RJ45), USB, HDMI
- Memory and/or Storage Space: 32GB RAM, 1TB SSD
- Processor Requirements: Quad-core, 3.0 GHz
- Graphical Representation: See Appendix A
- Cable Type/Length: CAT6 Ethernet cables, varying lengths
- User Interfaces: Web-based management interface
- Hard Drive Specifications: SSD for faster data retrieval
- Monitor Resolution: N/A

7.1.2 User Machine Interface

- Power Input Requirements: N/A (user device dependent)
- Signal Impedances and Logic States: N/A
- Connectors: N/A
- Memory and/or Storage Space: Dependent on the user device
- Processor Requirements: Dependent on the user device
- Graphical Representation: N/A
- Cable Type/Length: N/A
- User Interfaces: Web-based interface
- Hard Drive Specifications: N/A
- Monitor Resolution: Dependent on the user device

7.1.3 Database Server

- Power Input Requirements: 110-240V AC, 50-60Hz
- Signal Impedances and Logic States: Standard TTL levels
- Connectors: Ethernet (RJ45), USB
- Memory and/or Storage Space: 64GB RAM, 2TB SSD
- Processor Requirements: Octa-core, 3.5 GHz
- Graphical Representation: See Appendix B
- Cable Type/Length: CAT6 Ethernet cables, varying lengths
- User Interfaces: Web-based database management interface
- Hard Drive Specifications: SSD for faster data retrieval
- Monitor Resolution: N/A

7.2 Software Detailed Design

We have chosen Windows operating system for its best support and user-friendliness.To save the movie details, users preferences etc, we have chosen SQL database.To implement the project we will use Python libraries,Javascript,Html etc.

7.2.1 Movie Recommendation Algorithm Service

- Service Identifier: MovieRecommendationService
- Classification: Application
- Definition: Generates personalized movie recommendations based on user preferences.
- Requirements: User history, preferences, and real-time data.
- Internal Data Structures: User profiles, movie metadata.
- Constraints: Availability of user data.
- Composition: Utilizes collaborative filtering algorithms.
- Users/Interactions: Collaborates with User Interface service.
- Processing: Algorithmic recommendation engine.
- Interfaces/Exports: Recommendations API.

7.2.2 User Interface Service

- Service Identifier: UserInterfaceService
- Classification: Application
- Definition: Provides a user-friendly interface for interacting with the system.
- Requirements: User input, system feedback.
- Internal Data Structures: UI components, user input data.
- Constraints: UI responsiveness.
- Composition: Utilizes front-end frameworks.
- Users/Interactions: Interacts with Movie Recommendation Algorithm service.
- Processing: Handles user input and displays recommendations.
- Interfaces/Exports: User Interface API.
-

7.3 Performance Detailed Design

This spectrum aims for a responsive and scalable user experience:

7.3.1 Capacity and Volume Requirements/Estimates

- System designed to handle X simultaneous users.

7.3.2 Performance Expectations

- Response time to be under Y seconds.

7.3.3 Availability Requirements

- Targeting 99.9% uptime.

7.3.4 Performance Design to Meet Capacity Requirements

- Load balancing across multiple servers.

7.3.5 Reliability Design to Meet Availability Requirements

- Redundant servers with automatic failover.

7.3.6 Backup, Recovery, and Archive Design

- Daily backups with versioning, off-site storage.

The single point of failure is identified and mitigated through redundancy.

7.4 Internal Communications Detailed Design

The Specturm Movie Recommendation System incorporates multiple components that necessitate seamless internal communication. The following details the internal communication design to facilitate information exchange, command provision, and support input/output functions.

7.4.1 Number of Servers and Clients

- N servers and M clients per network segment.

7.4.2 Specifications for Bus Timing Requirements and Bus Control

- Ethernet communication with standard bus timing.

7.4.3 Format(s) for Data Exchange

- JSON format for data exchange.

7.4.4 Graphical Representation

- D See Appendix I.

7.4.5 LAN Topology

- Star topology for internal communications.

8. System Integrity Controls

Internal Security
Design Specification:
Access Control:
Implementation is done using Role-Based Access Control (RBAC).Specturm employs RBAC to restrict access to critical data items based on users/operators' roles and responsibilities.By implementing RBAC, only authorized personnel have access to sensitive data, enhancing overall system security.
Principle of Least Privilege:
Implementation is done by Granting users/operators the minimum level of access necessary, Spectrum follows the principle of least privilege, ensuring that users/operators only have access essential for their specific duties.Reducing unnecessary access mitigates the risk of unauthorized data exposure or manipulation.
Encryption:
Implementation by Encryption mechanisms during transmission and storage.Specturm employs encryption to safeguard critical data items during both transmission over the network and storage in databases.Encryption adds an additional layer of security, protecting data from unauthorized access or interception.
User Authentication:
Implementation by Strong user authentication methods (e.g., username and password)
Spectrum utilizes robust user authentication methods to ensure secure access, with username and password being primary credentials.Secure authentication methods prevent unauthorized users from gaining access to the system.

Audit Procedures
Design Specification:
Audit Logging:
Comprehensive audit logging procedures is done. The Specturm implements detailed audit logging procedures to capture control, reporting, and retention period requirements for operational and management reports.Comprehensive auditing is essential for monitoring system activities and identifying potential security incidents.
Audit Trail Storage
Secure storage of audit logs.Specturm securely stores audit logs, ensuring their integrity and availability for the required retention period.Secure storage guarantees the preservation and accessibility of audit trails for compliance and investigative purposes.

Automated Alerts:
implementation: Automated alerts for suspicious activities or unauthorized access attempts. The Specturm incorporates automated alert mechanisms to notify relevant parties of any suspicious activities or security breaches. Automated alerts enable prompt response to potential security threats, enhancing overall incident response capabilities.

Application Audit Trails
Design Specification:
Dynamic Auditing
Implementation is done by Dynamic auditing mechanisms for tracking retrieval access to critical data. Specturm dynamically audits retrieval access to designated critical data, monitoring user interactions with sensitive information.Dynamic auditing provides real-time insights into data access patterns, aiding in anomaly detection and security analysis.
Granular Logging
Implementation: Granular logging capturing details such as user identification, network terminal identification, date, time, and data accessed.Specturm ensures granular logging to capture detailed information for each retrieval operation, including user identification and data accessed.Granular logging facilitates thorough analysis and investigation in the event of security incidents.

Standard Tables for Data Validation
Design Specification:
Data Validation Tables
Implemented as Standard tables containing valid values for critical data fields.Specturm defines standard tables to maintain valid values for critical data fields, ensuring data integrity.Data validation tables prevent the entry of incorrect or unauthorized values, contributing to overall data consistency.

Referential Integrity
Implemented by Enforcing referential integrity constraints.The Specturm enforces referential integrity constraints to ensure data consistency across the system, preventing orphaned or inconsistent data.Referential integrity safeguards against data anomalies, maintaining the overall integrity of the system.

The implementation of these system integrity controls is crucial for maintaining the security, integrity, and reliability of the Specturm Movie Recommendation System. Regular monitoring and analysis of audit logs, adherence to access controls, and validation processes contribute to a robust and secure operational environment. The system should be continuously evaluated to adapt to emerging security threats and compliance requirements.
Currently we are yet to implement some of these necessary intergrity controls in the current v1.0 of our Spectrum system.

9. External Interfaces

Interface Architecture

The Specturm Movie Recommendation System interfaces with external systems to enhance its functionality and provide up-to-date movie information. The primary external system is the TMDB (The Movie Database) dataset, which serves as a comprehensive source of movie details. The interface architecture involves batch transfers and queries, ensuring seamless data exchange between Specturm and TMDB.

9.1. Interface Architecture

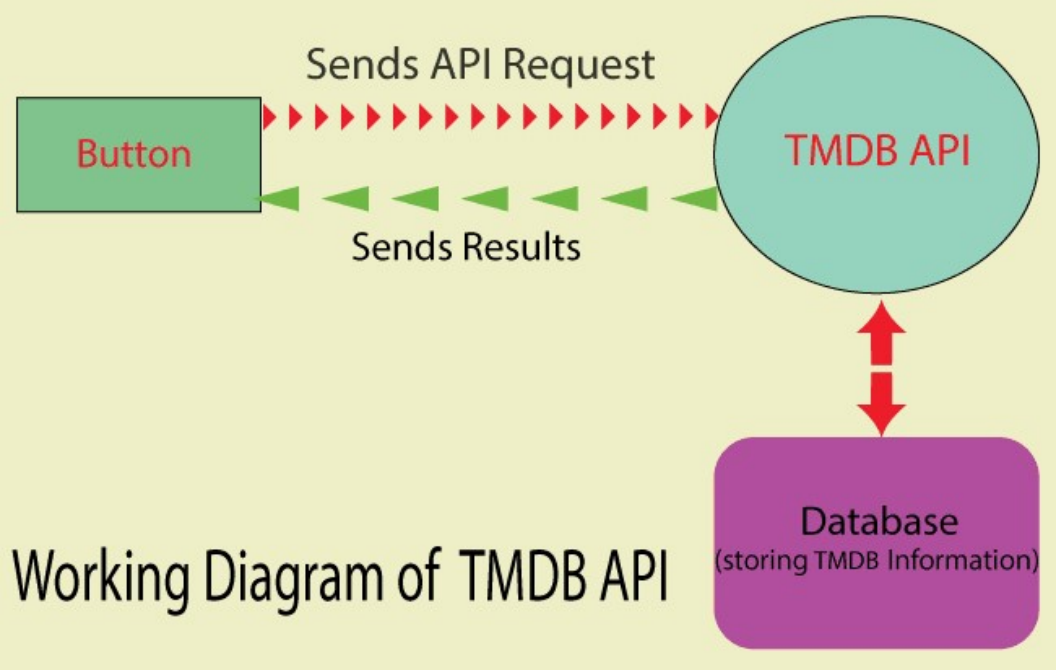
1. TMDB Interface:

Data Flow: Specturm engages in a two-way data flow with The Movie Database (TMDB). It retrieves movie details, such as metadata, ratings, and user reviews, to enrich its recommendation database. Additionally, Specturm periodically updates its movie database with the latest information from TMDB through batch transfers.

Data Format: The data exchanged between Specturm and TMDB is in JSON format, ensuring efficient and structured information exchange.

Interface Type: The interface primarily involves batch transfers for periodic updates and real-time queries for on-demand information retrieval.

Interface Diagram:



So when we search for the movie the above process happens between our spectrum system and TMDB Api and database .

9.2. Interface Detailed Design

1. TMDB Interface Control Document (ICD):

The TMDB ICD serves as a comprehensive guide for the interface between Specturm and TMDB.

Data Formats: It outlines the specific JSON data format used for communication, specifying the structure and encoding standards.

Protocols: The document details the communication protocols, ensuring a standardized and secure interaction between Specturm and TMDB.

Security Measures: The ICD describes the security measures, including authentication mechanisms such as API keys, to ensure the integrity and confidentiality of data during transit.

Batch Transfer Protocols: For batch transfers, the document provides specifics on the frequency, timing, and format of data exchanges to keep the Specturm movie database updated.

Real-time Query Mechanisms: It outlines the methods for real-time queries, ensuring that Specturm can retrieve the latest information from TMDB when users request specific details.

Error Handling: The document includes guidelines for error handling and resolution, ensuring the robustness of the interface under various conditions.

These external interfaces and the associated ICDs are vital components of the Specturm Movie Recommendation System, ensuring seamless communication with external systems, especially TMDB. The detailed documentation supports correct, secure, and efficient data exchange, enhancing the overall functionality and reliability of the movie recommendation system.