Requirements Analysis

The C-Movie platform requires retrieving and aggregating movie ratings from other platforms. The ratings data can be retrieved either through REST APIs, by fetching them from an FTP server, or via an online feed. The data that will be received will be a title, the genre(s), a rating and/or ratings per categories.Categories might differ per provider. Different rating formats need to be normalized across providers in order to calculate a common scale (0-100) result across movies. Other metadata differences across platforms, like genre or title need to be unified according to the most reliable source (per movie). The user, upon a movie search, should be able to see as a result (at least) the title, the genres, a total rating, split in 3 specific categories(performance, screenplay,soundtrack) that are calculated by an in-house algorithm.

Assumptions and Constraints

- C-Movie users' only interaction with the platform will be searching. No new ratings will be added by the users, hence we don't need to create user accounts.
- I assume that not all movies need to be processed at once at the beginning of the MVP. There could be millions of movies around the world released yearly. I assume that will not have a big impact on the business side (especially for an MVP), while on the technical side we will not have to deal with a huge batch processing on start.
- For simplicity, I assume that the first population is made from a chosen list of the most popular movies of the past decades as well as the last few years.
- I assume that users do not expect real-time updates. A few updates per day could fit the requirements.
- I assume that ratings of new movies could radically change over the period of a few months after its first release.
- I assume that the online feed will send some sort of notification to the C-movie platform, to inform for updates. I also assume that the online feed will have a relatively low rate of updates per day.
- I assume that the MVP will have a few thousand users/requests per day.
- I assume that after the first data population, the MVP will receive a few thousand data updates per day.
- I assume that the C-Rating algorithm requires a minimum number of providers per movie to make the calculation of the aggregated ratings, while some providers might also be mandatory.
- I assume that each provider only provides a single type of integration (API, online/offline feed).
- Each provider provides fixed rating categories for their movies.
- I assume that there is a third party service that provides universal movie IDs and deduplication (from quick search this is probably true, e.g EIDR)

Solution

High Level System Description:
In the solution described, I consider the data collection to be the most complicated part. In the flowcharts, I describe 4 different flows of data fetching, in order to account for the different types of data sources, as well as dealing with updates. I also provide a high level system design and DB schemas for the raw (yet normalized) data, and the processed (with C-Ratings) data.
In a nutshell, the system will be receiving data from various sources, through different services. All services will serialize the data and send them to a message queue for normalization. A DB with the normalized data will be populated. A new entry in the DB, will enable a DB trigger to fetch data from missing (API) providers. A new entry or an update will also trigger the processing of the new data from the C-Rating Calculator service, which contains the proprietary C-Rating algorithm. If enough data is available for the aggregated score to be calculated, the Calculator will write the result in the DB with the processed data. The results will be available for the users to search through a web app that is supported by a backend service that communicates with the processed DB and hosts the REST API. If a search is not successful, the backend service will trigger a request for fetching data for the movie that the user is looking for to the providers that are integrated through API. The result will be served asynchronously to the users (e.g with a message, "rating is pending, search again later").
Below, I further describe some decisions taken for the design and the processes.
Data Collection: The system supports three types of data collection.
-*Data are fetched through REST API*, which is triggered from three different flows: a) a user is looking for a movie that we have no rating for, b) the online/offline sources provided new movies, so we request the ratings from API providers too, c) a periodic trigger that request updates for existing movies that are relatively newly released.

- *Data files are fetched from online feeds.* In this case we are waiting for a notification of an RSS feed or equivalent in order to look for new data in an FTP server.
-*Data files are fetched periodically from an offline server.* We can initially populate one in-house offline server with data from popular movies of the past decade, as well as with movies that were released in the last few years. New searches from the users and online feeds will incrementally populate the database with other movies.
Deduplication: In order to have a faster release we decide to hire a third party service that does the deduplication and connects the titles with a common id.
Normalization and DB population: These functions are put in one service, due to their simplicity and continuity but they could also be separate microservices. The DB is able to watch real-time changes in a table and send an event to an external service (e.g the service that triggers further data fetching). However, this is an architecture that I would consider with a grain of salt and I would consult with the engineering team.
C-Rating Calculation: In order to resolve conflicts on the title and the genre of a movie across platforms, we can create a reliability score on the title and genre by comparing commonality across platforms. We can also deterministically enhance the reliability score, with a weight we give to each platform (e.g according to popularity). We can choose to provide a result only if we have ratings from sufficient providers, but we can also use the reliability weight of each provider in order to decide if a result will be reliable (e.g if we have IMDB and RT, we don't need 4 more providers to decide on a score). In regards to the categories rating, we will have to consider the categories each provider gives to calculate our categories score in function to those categories.
C-Movie application: A search for a movie that is not stored in the C-movie database could trigger a real-time request towards the integrated rating platforms, as well as a real-time processing. The user will be warned they need to wait for a result. Additionally, we can use a search engine (e.g Elasticsearch), to help the user type the movie title.
Other considerations:
I would also make sure that monitoring and observation tools are in place, in order to do error-handling control and make adjustments for performance (e.g caching, autoscaling etc) and cost saving. I would also consider some user testing in order to understand the usability of the platform.