1 Phase 1

1.1 Destination Group

In today's vast landscape of streaming platforms and endless content options, keeping track of what you have watched can quickly become overwhelming. That is where my application should steps in to offer invaluable assistance. By utilizing this web-application, users gain their own centralized hub for organizing their entertainment preferences. No more struggling to recall which episode they last watched or forgetting the name of that intriguing new series they stumbled upon. With the ability to easily add, remove, and personalize entries, users can create their own personalized watchlist. Additionally, my web-application enhances engagement by allowing users to rate and jot down personal notes for each series, fostering a deeper connection with their favorite shows and movies.

But of course this webapplication is not needed by everybody it is more for passionate watcher and proably those whoe watch multipe movie/series at the same time.

1.2 Technology Stack

1.2.1 Backend

I've decided to leverage a Node.js due to its non-blocking, event-driven architecture. To manage data, I've chosen MongoDB. Reason for this is its flexible, schema-less nature, allowing for rapid iteration and evolution of the data model. This is particularly beneficial for handling the diverse and dynamic data structures typical in entertainment content.

Additionally, I'll plan to utilize Node Express API as the communication bridge between the frontend and backend, offering RESTful endpoints for CRUD operations like retrieval, update, addition, and removal of entries within the database.

The entire project will be containerized using Docker to ensure a smooth and easy deployment, allowing easy management of dependencies and consistent environments across different stages of development and production.

1.2.2 Frontend

For the Frontend, I plan to utilize only HTML, CSS, and JavaScript. However, for styling purposes, I may incorporate open-source templates/resources if necessary.

1.2.3 Why

This technology stack was chosen to align with my goal of expanding my JavaScript skills, despite being more proficient in Python. Node.js allows for the development of both backend and frontend in the same language, simplifying the development process. Its efficiency and performance in handling I/O-intensive operations make it well-suited for a web application that frequently interacts with a database. MongoDB's flexible schema allows me to focus more on application development rather than dealing with rigid database structures, which is crucial for handling a variety of data types and relationships inherent in tracking entertainment content.

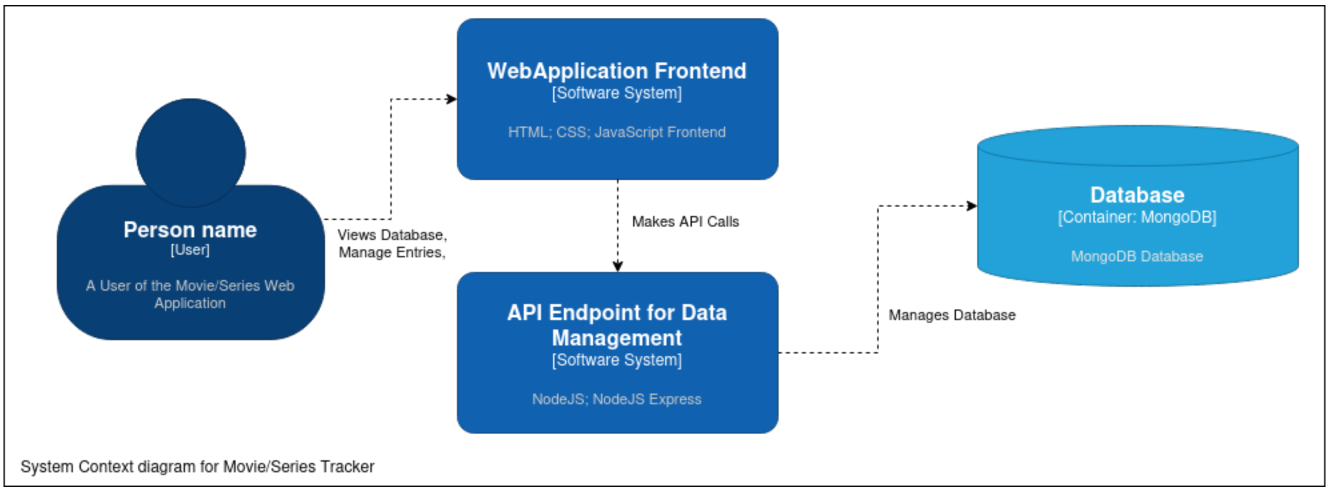
WebApplication Frontend
[Software System]
HTML; CSS; JavaScript Frontend
Person name
[User]
A User of the Movie/Series Web Application
Views Database, Manage Entries,
Makes API Calls
Database
[Container: MongoDB]
MongoDB Database
Manages Database
API Endpoint for Data Management
[Software System]
NodeJS; NodeJS Express
System Context diagram for Movie/Series Tracker

Figure 1: Graphic overview of the technologies chosen