Checkpoint 1 report

Contents

[Introduction 1](#_Toc161557868)

[Project description 1](#_Toc161557869)

[Objectifs analysis 1](#_Toc161557870)

[Uses cases (UML) 2](#_Toc161557871)

[Conceptual and logical database diagrams 2](#_Toc161557872)

[Conceptual design 2](#_Toc161557873)

[Test data identification 3](#_Toc161557874)

[Tools used 3](#_Toc161557875)

[Conclusion 3](#_Toc161557876)

# Introduction

This report provides an overview of the database design for our bDMI project, as well as its functional and non-functional requirements and use cases. A conceptual and logical database schema will set the foundation for the database's structure.

# Project description

The bDMI (IMDb in reverse) project aims to develop a platform that allows users to browse a comprehensive movie and tv series database. The platform will include search, registration, recommendation features, and the possibility of creating a personal watchlist.

# Objectifs analysis

We identified the following objectives:

* Offer an online catalogue of movies and tv series
  + Build a database of movies and tv series
  + Offer a good user experience
    - Create an engaging website
      * Implement a recommendation system
      * Implement a user registration feature
      * Implement a watchlist feature
      * Implement a search and discovery feature

Based on those objectives, we identified the following functional and non-functional needs:

The first functional need is "Allow users to interact with a catalogue of movies and tv series", supported by EF 2: "Allow users to register with the website", EF 5: "Allow users to interact with the service" and EF 4: "Allow users to discover movies and tv series".

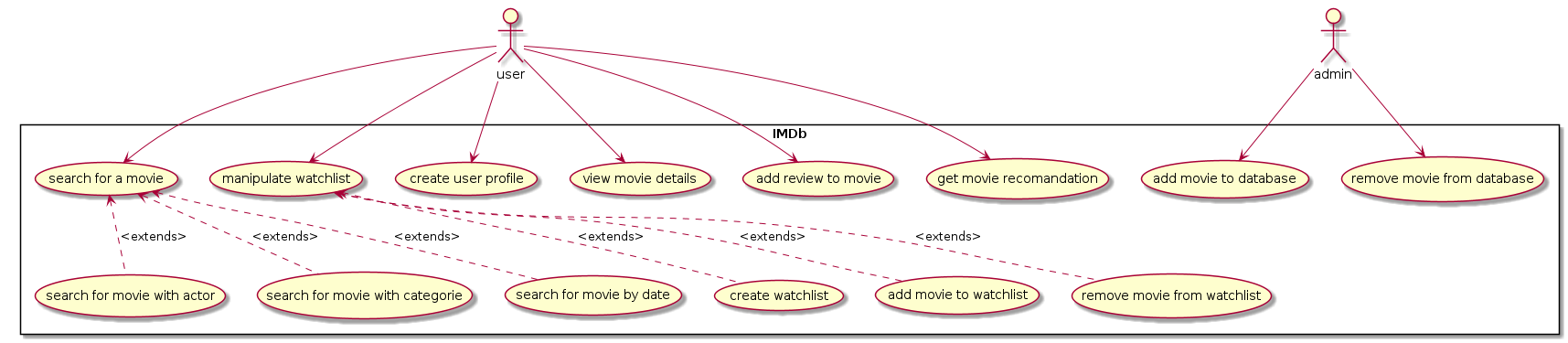
EF5 is made up of the following: "Allow users to create a watchlist", ENF "Users can manipulate their watchlist through adding, removing and viewing the content of their watchlist" and "Allow users to add a review and a rating for a movie".

EF 4 is made up of the following: EF: "The service should provide movie recommendations", ENF: "The service should provide movie recommendations based on user preferences and past activity".

# Uses cases (UML)

We identified the following use cases:

* A user can:
  + create a watchlist
  + add a movie to their watchlist
  + manipulate their watchlist
  + remove a movie from their watchlist
  + create a user profile
  + search for a movie
  + search for a movie using different filters like actor, category, date
  + view a movie's details
  + review a movie
  + get movie recommendations
* An administrator can:
  + add and remove movies to and from the database



# Conceptual and logical database diagrams

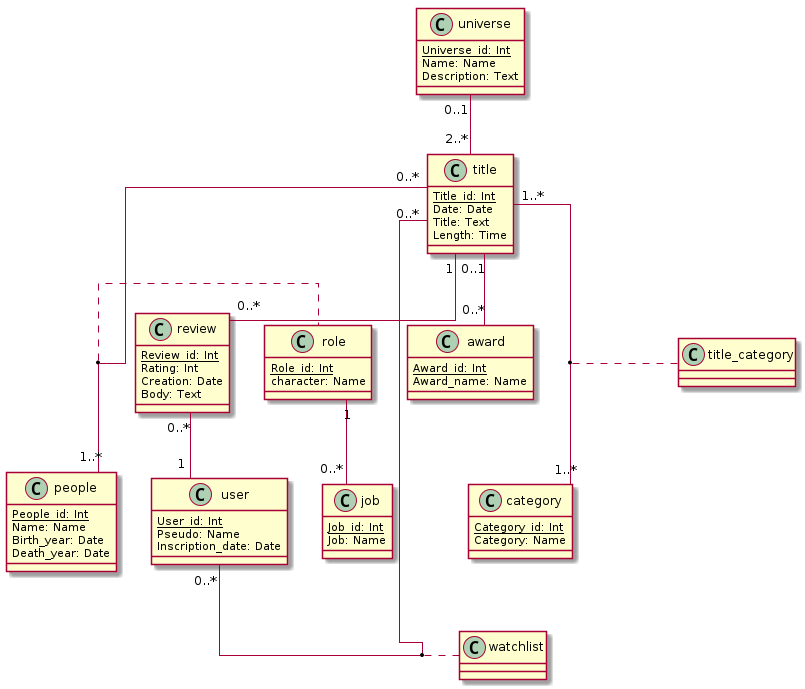
## Conceptual design

We have identified the following entities relevant to our project:

* universe : will contain a bundle of movies belonging to the same universe and its description. For example, Star Wars, Harry Potter, Transformers, Lord Of The Rings etc.
* title : contains all movie titles
* award : contains all movie awards
* category : contains the different categories
* people : contains the people involved with the production/making of the movie (directors, producers, writers, actors, etc.)
* watchlist : contains the watchlist of users
* user : contains all users of the service
* review : contains all reviews of movies and shows
* role : contains the character played by a person in a movie
* job : contains the jobs people had for a movie

Furthermore, we have identified the following relationships between those entities:

* universe "0..1" -- "2..\*" title : for a universe to exist, it must contain at least two movies. A title can belong to no universe.
* title "0..*" -- "1..*" people : a movie necessarily has people (crew and actors).
* title "1" -- "0..\*" review : a movie can have 0 or more reviews.
* title "0..1" -- "0..\*" award : a movie can have 0 or more awards.
* title "1..*" -- "1..*" category : a title must have 1 or more categories, and a category must have at least one title.
* review "0..\*" -- "1" user : a user can write none or more reviews for a title.
* user "0..*" -- "0..*" title | (title, user) .. watchlist : a title and a user are in relation with each other through a watchlist. A user can have no movies in their watchlist or more.
* role "1" -- "0..\*" job : an actor can also be an a director or producer.
* (title, people) .. role : a title and a person will be related through the role table
* (title, category) .. title\_category : a title and its category will be related through the title\_category table.



## Logical design

# Test data identification

# Tools used

We primarily used the Git version control tool.

# Conclusion