Design Report for CS235 Assignment 2

This design report looks into the web application, CS235Flix. More specifically it introduces its cool new feature and describes and justifies the key design decisions made in the application's development.

Cool new feature: Watchlist

The cool new feature that has been implemented in CS235FLIX is a watchlist. This feature was selected because of the additional benefit it provides in terms of user functionality. More specifically, users while browsing often come across movies they would like to watch but just not at that time point. The watchlist allows the user to store these movies for them to watch in the future. To use the watchlist, the user must be registered and logged in. While browsing and searching every movie has a button which allows the user to add the movie to their watchlist. If the user chooses to add the movie to their watchlist, they are redirected to their watchlist so that they can see what all movies they have in their watchlist. Redirecting the user back to their watchlist is done so that the user is aware of the number of movies they have in their watchlist, therefore encouraging them to keep their watchlist to an appropriate length. Furthermore, the user can easily go back to whatever they were browsing or searching by pressing the browser's back button. If the movie is already in the watchlist, the movie will not be added again to the watchlist and the redirection will show the user that the movie is already in their watchlist. The user can view their watchlist from the navigation panel by clicking 'watchlist'. For every movie in the watchlist, there is a button which allows the user to remove the movie from the watchlist. If the user chooses to delete the movie, it is removed from the watchlist and the user is redirected to the watchlist to show them that the movie has been deleted.

Key design decisions

For assignment 2, a key design decision that made its implementation much easier was simplifying the domain model. This simplification involved deleting the genre, actor, and director classes. Therefore, instead of using genre, actor and director objects, strings were used to represent the information relevant to these fields. This meant the domain model was much easier to deal with in the later stages as it only involved the necessary classes. Instead of using the watchlist class, the watchlist attribute and functions which add and remove movies to and from the watchlist and retrieve the watchlist were added to the user class. The movie class was modified to ensure that movie object contains all required information about the movie ie. Rank, title, release year, genres, actors, and director because this was critical for the service layer to be implemented efficiently. The MovieFileCSVReader was modified to ensure that duplicate objects were not created ie. We only create one movie object for each movie. This will also be useful for assignment 3.

Design principles and patterns

The repository pattern has been used in developing this web application. This is because it provided a means to avoid committing to database early on. More specifically, using a repository pattern allows us to comply with the SOLID principle, Dependency Inversion. The repository pattern means that the high-level module ie. the service layer does not depend on a low-level module ie. a database, but rather is dependent on the abstract repository interface (abstraction). The database is also dependent on the abstract repository interface.

The application's architecture complies with the SOLID principle, Single Responsibility. Each of the modules have a well-defined responsibility. Flask is responsible for abstracting above threads, network programming and HTTP protocol. The view layer is where the request handlers are implemented. The service layer implements application logic.

CS235Flix has 4 different blueprints: home, authentication, movies, and utilities. By using blueprints, different application functionality was able to be separated out into modules as each module has a single responsibility. The home blueprint is responsible for serving up the homepage. The authentication blueprint is responsible for registering users, and handling login/logout. The movies blueprint is responsible for handling requests concerning movies. The utilities blueprint contains a useful utility function that is used by the other blueprints. Within each blueprint there is a "webby" component that is responsible for implementing HTTP requests and generate HTML pages and a service component that implements application logic/rules and interacts with the repository and domain model. Consequently, the resulting elements are better organised, easier to work with and text, and offer more scope for reuse. Furthermore, the blueprint implementations maintain a separation of view and service concerns.

Overall, complying with the SOLID principles – Single Responsibility and Dependency Inversion – has been tremendously beneficial for CS235Flix. In terms of development, complying with Single Responsibility meant that the code was easier to follow, understand and debug. Complying with Dependency Inversion means that CS235Flix is a web application that is easier to change and maintain in the future.

Link to github repository: https://github.com/SrishtiT/CS235Flix-A2.git
All relevant files are in the master branch.

Requests: When marking please make the browser zoom to 80% as that is what was defaulted in my computer. Also, I have had issues with my browser as the CSS updates have sometimes failed to carry through. To resolve this, I used the chrome extension 'CSS Reloader'. If the home page does not look like the image below, please contact me at stoo676@aucklanduni.ac.nz as the web application works perfectly on my computer. Also, sometimes my browser required me to press 'logout' before doing anything else on the web application. Please note that this issue was to do with my browser and not my web application.