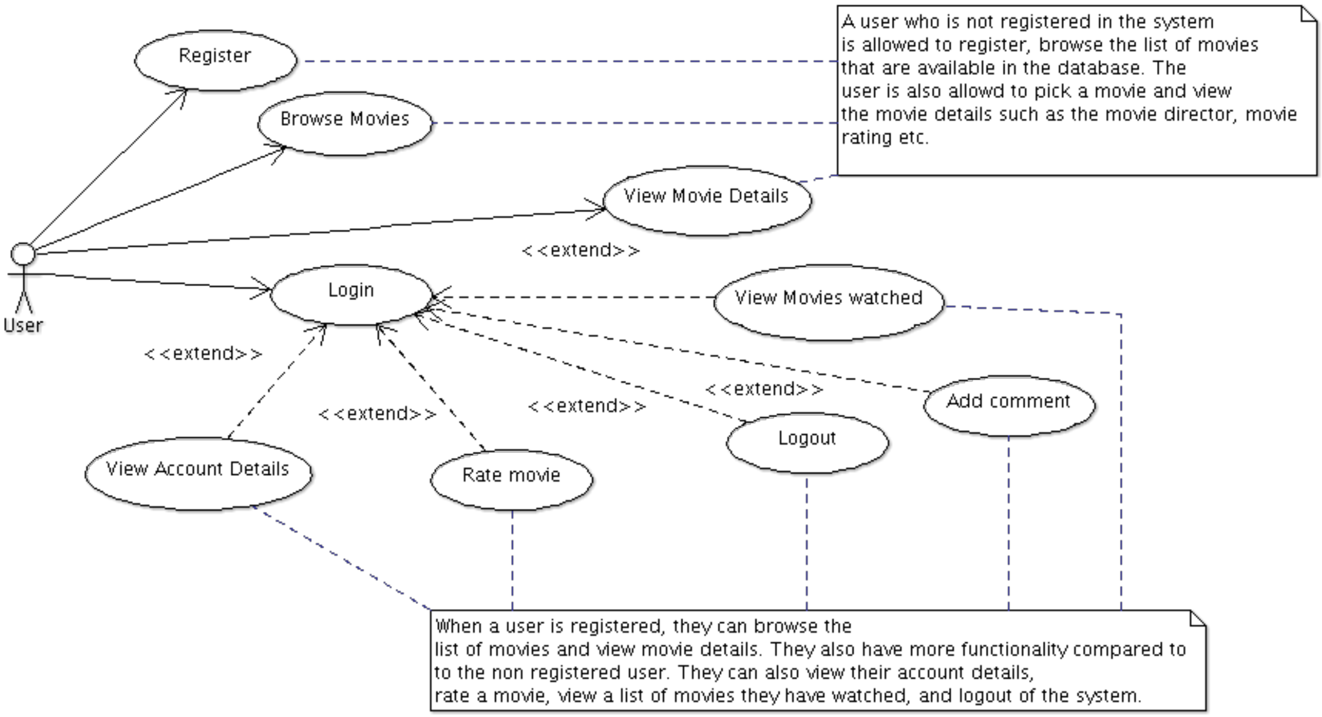
DT228/3 Software Engineering 3
Assignment Deliverable 1

Student Name: Jonathan Riordan
Student ID : C13432152

Video URL: https://youtu.be/vAbWZfc9iOU
For my assignment, I decided to do a movie web application.

**Use Case Diagram:**

The non-registered user is able register, browse movies on the site and to view movie details. These are the only methods avail to a non registered user compared to a register user. A registered user, once logged into the system has the options to view account details, rate a movie, view movies watched, logout and the functions of the non registered user such as browse movies and view movie details.

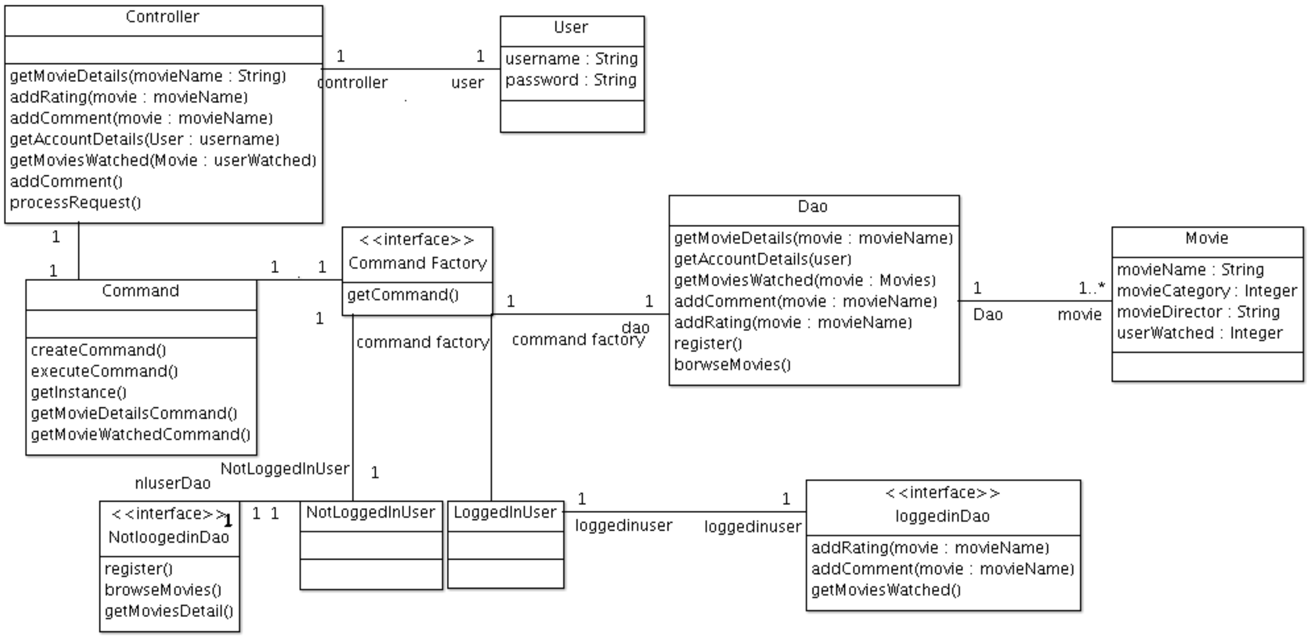


A user who is not registered in the system
is allowed to register, browse the list of movies
that are available in the database. The
user is also allowd to pick a movie and view
the movie details such as the movie director, movie
rating etc.

When a user is registered, they can browse the
list of movies and view movie details. They also have more functionality compared to
to the non registered user. They can also view their account details,
rate a movie, view a list of movies they have watched, and logout of the system.

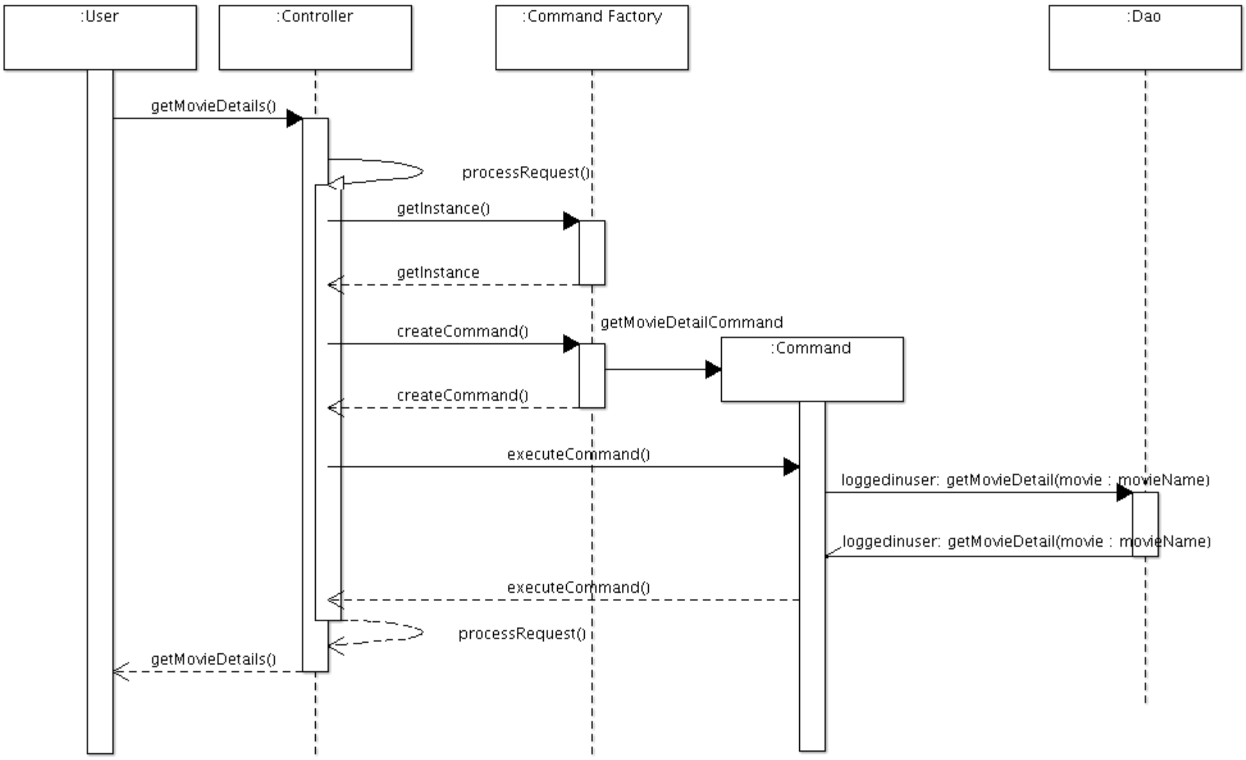
**Class Diagram:**

The relationships through out this class diagram are bidirectional. User has a 1 to 1 relationship to Controller. Controller acts as the functions of the web application. It too has a 1 to 1 relationship to the command class. The command class has the methods of geInstance, createCommand, executeCommand. I will be using the Command factory pattern. Depending on the method, if the method belongs to a non logged in user or logged in user, the relevant information will be displayed to the user, for example a non looked in user can only perform the register browseMovie and getMovieDetail methods. The Dao has a one to many relationship to the class movie. The dao can have many movies while each movie can only have one database.



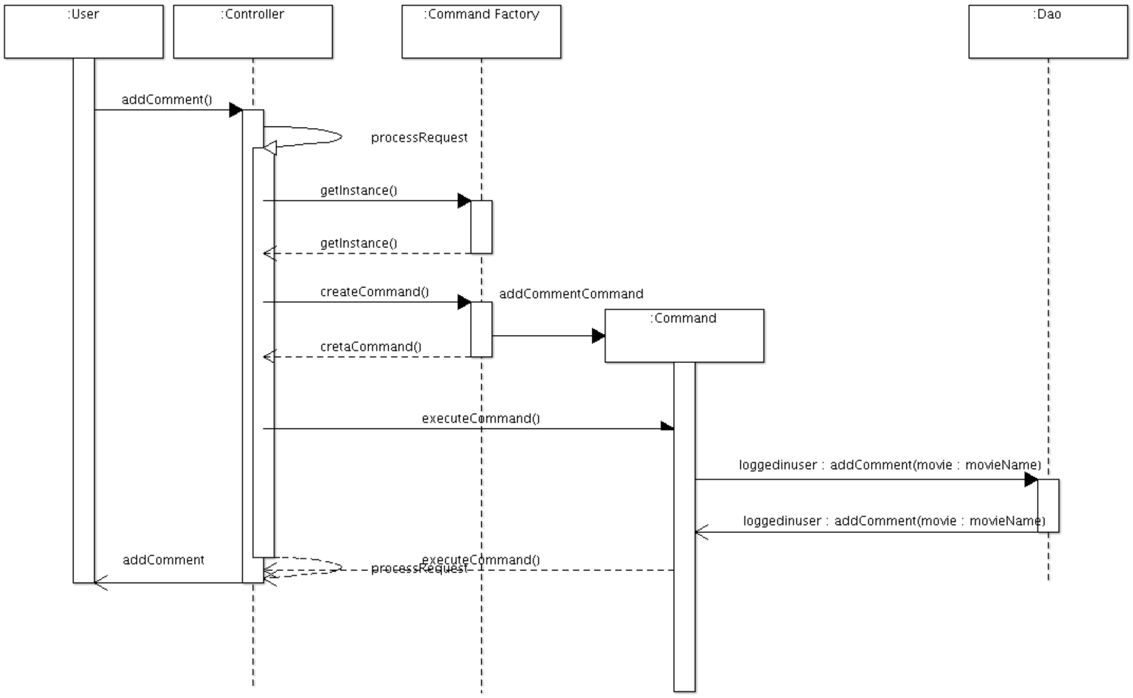
**Get Movie Detail:**

The actor user creates a call getMovieDetail and sends it to the controller. This request is then processed. The instance is then got and a command is created called getMovieDetailCommand. The command is then returned to the requested process and is then executed. The method getMovieDetail is performed.The movie that is selected is identified by the movieName.



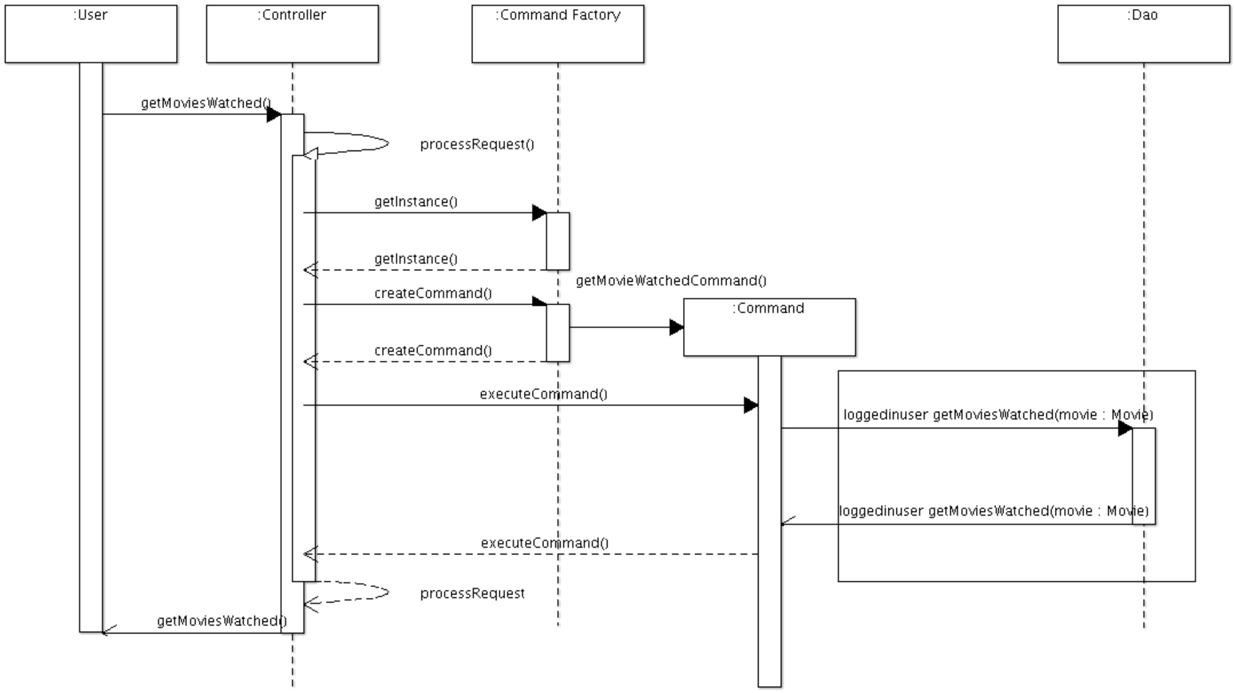
**Add comment Sequence Diagram:**

The actor user creates a call addComment to the controller. This request is then processed. The instance is then got and a command is created called addCommentCommand. The command is then returned to the requested process and is then executed. The method addComment is performed. The movie in which the comment is added to is done by using the movie name as the identifier.



**Get Movie Watched Sequence Diagram:**

The actor user creates a call getMovieWatched to the controller. This request is then processed. The instance is then got and a command is created called getMovieWatchedCommand. The command is then returned to the requested process and is then executed. The method getMovieWatched is performed. A loop is used to print out each movie that the user has watched. This can be achieved in code using boolean's for each movie.


:User
:Controller
:Command Factory
:Dao
getMoviesWatched()
processRequest()
getInstance()
getInstance()
getMovieWatchedCommand()
createCommand()
:Command
createCommand()
executeCommand()
loggedinuser getMoviesWatched(movie : Movie)
loggedinuser getMoviesWatched(movie : Movie)
executeCommand()
processRequest
getMoviesWatched()