|  |
| --- |
| Programming for Web 2.0 |

Requirement specification

*Star Wars Timeline*

*by Jan Weiss and Malte Dammann*

|  |
| --- |
| *Författare:* Malte Dammann, Jan Weiss  *Handledare:*  *Tobias Andersson Gidlund*  *Examinator:*  *Tobias Andersson Gidlund*  *Termin: VT16*  *Ämne: Programming for Web 2.0*  *Kurskod: 2DV512* |



Table of contents

[1 Purpose - 2 -](#_Toc469056950)

[2 Overall description - 3 -](#_Toc469056951)

[2.1 User Interface - 3 -](#_Toc469056952)

[2.1.1 Admin Interface - 3 -](#_Toc469056953)

[2.1.2 Public User Interface - 6 -](#_Toc469056954)

[3 Backend - 7 -](#_Toc469056955)

[4 Frontend - 8 -](#_Toc469056956)

[5 Database - 9 -](#_Toc469056957)

[5.1 The movie database table - 9 -](#_Toc469056958)

[5.2 The user database table - 9 -](#_Toc469056959)

[6 Deployment - 10 -](#_Toc469056960)

# Purpose

The final project of Programming for Web 2.0 is a scrollable timeline with content of our choice. We decided to display all “Star Wars” movies in a scroll view. For each movie facts like description, title, release date, trailer, title picture, etc. are shown.

An admin can login to the dashboard via the login-page. There it is possible to create new movies an to edit or delete existing ones.

The project includes creating the server side and the client side software using a Java-based application server.

# Overall description

## User Interface

In this paragraph the two different user interfaces from the perspective of the administaror and the public user are shown up. There are six main Use Cases, shown in realtion to the user in the Figure 1 Use Cases within the application. Details in the following paragraphbelow. They are described in more detail in the subsections below.

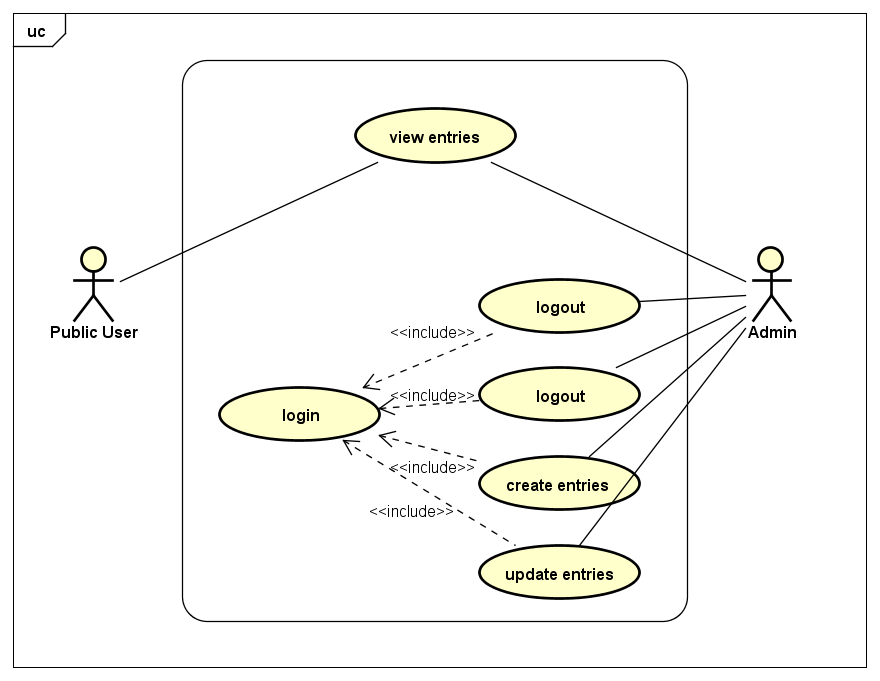


Figure 1 Use Cases within the application. Details in the following paragraph

### Admin Interface

Use Cases within the timeline, done from the Administrator and/ or from the dashboard.

|  |  |
| --- | --- |
| **Name** | D1: create new movie |
| **User** | Administrator |
| **Prerequirement** | Administrator is logged in and viewing the Dashboard |
| **Steps** | Administrator clicks on ‘New Movie’ in the list on the left  An empty form is shown to enter the movie data  Administrator fills in the form and clicks ‘Save’ to save movie  The movie gets stored in the database |
| **Special cases** | If there are fields not filled or filled wrongly (checking data type) an error will be shown and the movie not saved in the database |
| **Postcondition** | The movie is now stored in the database and will be shown on the public user interface, the timeline. After submitting a new movie the site reloads, listing the movie in the list on the left |

|  |  |
| --- | --- |
| **Name** | D2: edit existing movie |
| **User** | Administrator |
| **Prerequirement** | Administrator is logged in and viewing the Dashboard |
| **Steps** | Administrator clicks on a stored movie in the list on the left  A form is shown with the currently stored information about the movie  Administrator does changes within the form and clicks ‘Save’ to save movie  The movie gets edited in the database |
| **Special cases** | If there are fields not filled or filled wrongly (checking data type) an error will be shown and the movie not edited in the database |
| **Postcondition** | The movie is now updated in the database and will be shown like this on the public user interface, the timeline. After editing a movie the site reloads |

|  |  |
| --- | --- |
| **Name** | D3: delete existing movie |
| **User** | Administrator |
| **Prerequirement** | Administrator is logged in and viewing the Dashboard |
| **Steps** | Administrator clicks on a stored movie in the list on the left  A form is shown with the currently stored information about the movie  Administrator clicks on ‘Delete’ underneath the form  A pop-up asks to confirm the deletion, on confirmation  The movie gets deleted in the database |
| **Special cases** | The Administrator cancels from the confirmation-dialogue and the movie will not be deleted |
| **Postcondition** | The movie is now deleted from the database and will not be shown on the public user interface, the timeline, anymore. After deleting a movie the site reloads, not listing the movie in the list on the left anymore |

|  |  |
| --- | --- |
| **Name** | D4: Logout |
| **User** | Administrator |
| **Prerequirement** | Administrator is logged in |
| **Steps** | Administrator clicks on ‘Logout’ in the top-bar on any site within the timeline website  The administrator will be logged out. |
| **Special cases** | - |
| **Postcondition** | The administrator is logged out and can’t watch the dashboard anymore. The public user interface, the timeline, will load |

|  |  |
| --- | --- |
| **Name** | D5: Login |
| **User** | Administrator |
| **Prerequirement** | Administrator is not logged in |
| **Steps** | Administrator clicks on ‘Login’ in the top-bar on the public user interface, the timeline  A new website loads, holding a login-form  Administrator enters his email address and password  Administrator clicks on ‘Login’ underneath the form  Server checks input against database, the administrator gets logged-in with session attribute |
| **Special cases** | If the login credentials are wrong, an error is shown and the administrator does not get logged in |
| **Postcondition** | The administrator is logged in and gets redirected to the dashboard |

### Public User Interface

Use Case performed by an user from the home page, the time line.

|  |  |
| --- | --- |
| **Name** | U1: watching trailer |
| **User** | Administrator or public user |
| **Prerequirement** | User is watching the public user interface, the timeline |
| **Steps** | User browses through the movies on the timeline  User clicks on one of the ‘Trailer’ buttons of a movie  A pop-up opens, holding an embedded YouTube video, the trailer of the movie  User can click play to watch  User can click on ‘X’ or besides pop-up to close the pop-up and stop the video from playing |
| **Special cases** | If there is no video stored for a movie, no video will be shown |
| **Postcondition** | - |

# Backend

We will use Spring MVC and Java Server Pages as our backend.

We will need two main classes for our content: One for the movies containing all attributes of the movies and one class for the administrators which are allowed to access the dashboard and edit the movie database.

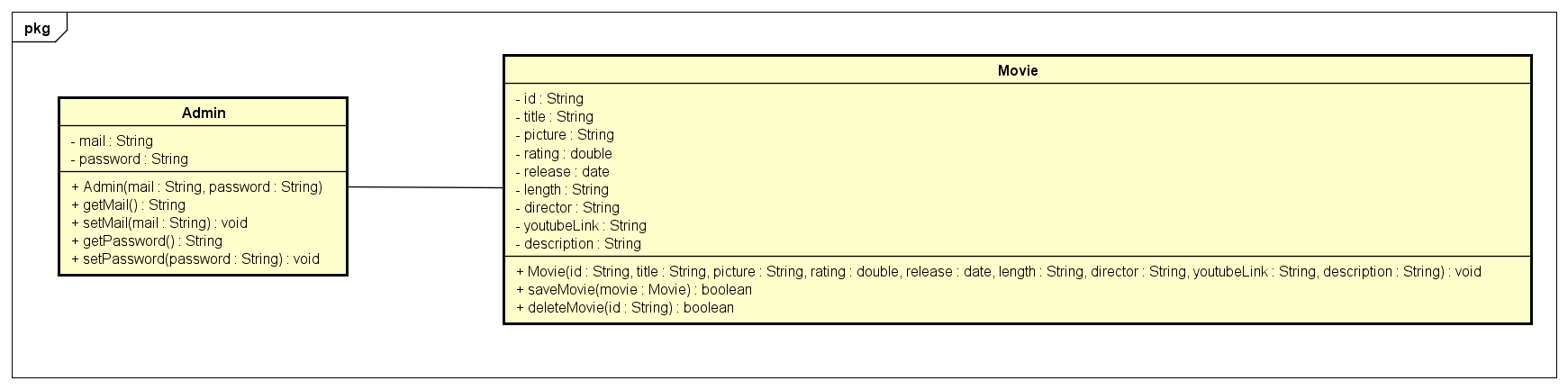


Figure 2 Class diagram

A session manager included in Spring MVC will be used to control the user management. The session manager is used to restrict access to the dashboard to users stored in the database being administrators.

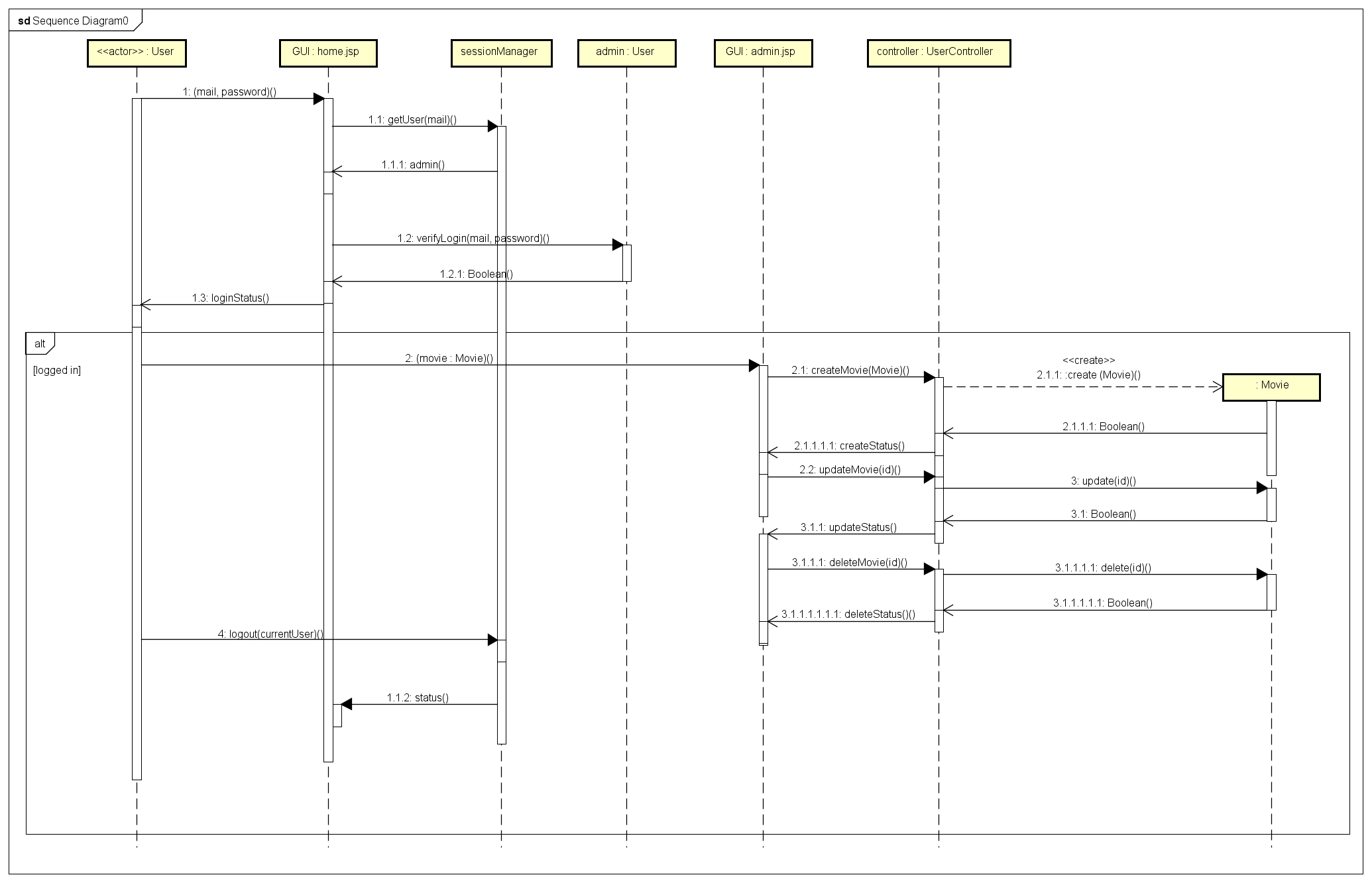


Figure 3 Actions possible to perform within the website. Compare with Figure 1 Use Cases within the application. Details in the following paragraph

# Frontend

Our frontend is created with Java Server Pages with JSP Standard Tag Library (JSTL) and the design will be adjusted using CSS and Twitter Bootstrap and JavaScript, including jQuery.

We will provide three different web pages:

* Homepage (Timeline)
  + Showing the timeline including the Star Wars movies as suggested in the Mock-up
  + Accessible for all users
* Login
  + Showing a login form as suggested in the Mock-up
  + Lets users saved in the database login and access Dashboard afterwards
  + Accessible for all users
* Dashboard
  + The content management dashboard for administrator, showing currently stored Star Wars movies
  + Shows list of all stored movies and allows to create new one, alter existing ones or even delete them
  + Accessible for logged in administrators

# Database

The database is supposed to hold all data needed to run the Star Wars timeline. Within the project we mainly have the information about the different movies. Additionally we need a user management and save the login information within the database.

As database management system we will use MySQL as it an open-source relational database management system which is a well-known and approved management system and easy to use.

We will create two tables: One for the movies and one for the users.

## The movie database table

|  |  |
| --- | --- |
| * id | int |
| * title | varchar |
| * rating | double |
| * releaseDate | varchar |
| * director | varchar |
| * length | varchar |
| * youtubeLink | varchar |
| * description | varchar |
| * image | longtext |

Primary key: id

## The user database table

|  |  |
| --- | --- |
| * id | int |
| * email | varchar |
| * password | varchar |

Primary key: id

# Deployment

We will use XAMPP to create a local server holding our database. XAMPP provides an phpMyAdmin installation we are using to maintain the database. We will provide a SQL document we can use to set up the database including Star Wars movies to easily set up the database from a phpMyAdmin dashboard.

For our server we use Wildfly as a JBoss instance and include it within IntelliJ IDEA which we use as IDE.