2. Analysis

**2.1 Use case diagram**

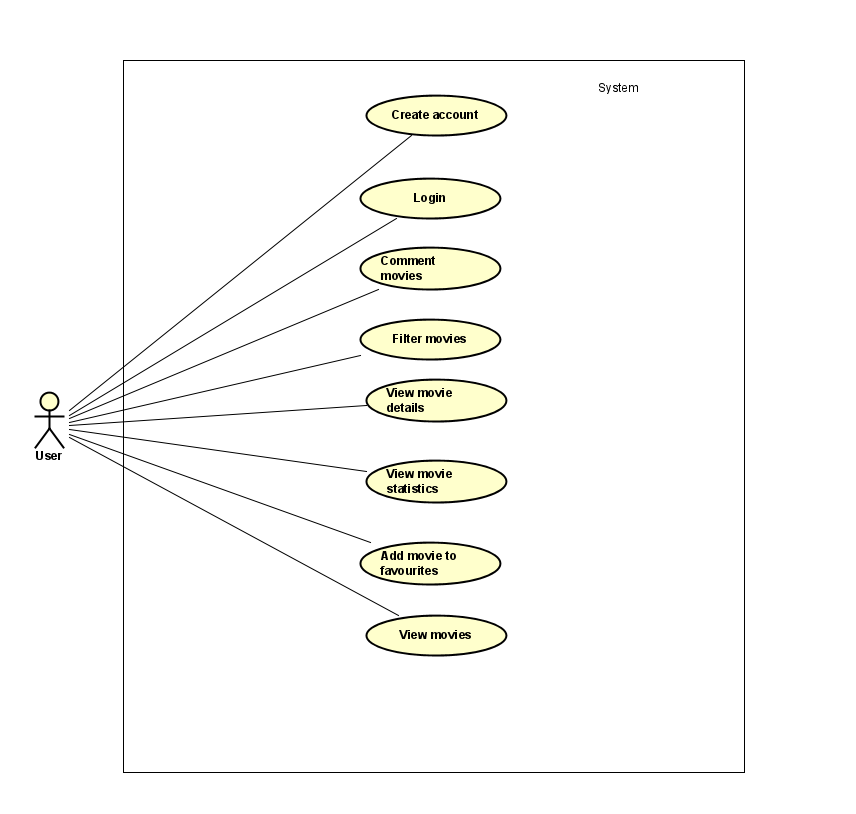


Figure 1 System analysis - Use cases

The system has one actor the User.

The user can create an account, login, write comments on movies, filter movies, view specific movie details, view movie statistics based on: most voted movies, best-rated movies, user’s favourites, add movie to favourites and view movies.

**2.2. Use case description**

Use case description explains a specific use case in full detail. The summary is a short description of the whole use case.

|  |  |  |
| --- | --- | --- |
| **Item** | **Value** | |
| **Use Case** | **View Movies** | |
| **Summary** | Actor is using the search function to find a movie and its details. | |
| **Actor** | User | |
| **Precondition** | The actor must be logged in to the system.  The system must have a movie database available. | |
| **Post condition** |  | |
| **Base Sequence (Main Scenario)** | **Actor** | **System** |
| 1. Select the option to search for movie  2. User fills in the title of the movie  3. Submit to start searching | 1.1 Displays the form of movie title  2.1 System validates  3.1 Displays the movie, including details, that matches the searched information |
| **Branch Sequence 1 (Alternate Scenarios)** | 2a. The movie title does not exist in the database | |
| **Actor** | **System** |
| 2. User fills in the title of the movie  3. Submit to start searching | 2.1 System validates  3.1 Displays the movie, including details, that matches the searched information |
| **Exception Sequence for the Branch Sequence 1**  \* at any time during steps during step 2a.2 Actor can cancel the process  Use case ends. | |
| **Exception Sequence – Base Sequence** | At any step, the actor aborts the process. The use case ends. | |
| **Notes** | The search is based on the title of the movie. The details displayed for a movie are: director, actor, year of production and ratings. | |

**2.3. Activity diagram**

**2.3.1 Search Movie**

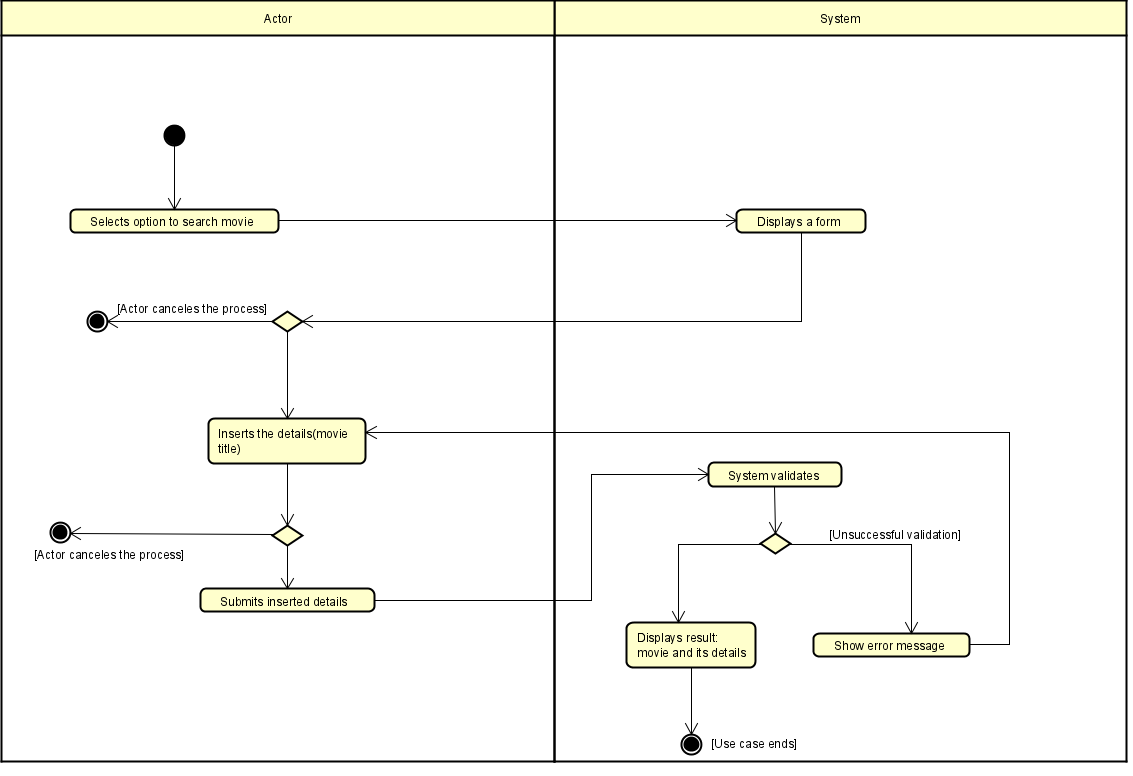
****

Figure 2 Search movie - Activity Diagram

The activity diagram represents the flow of search movies how it works.

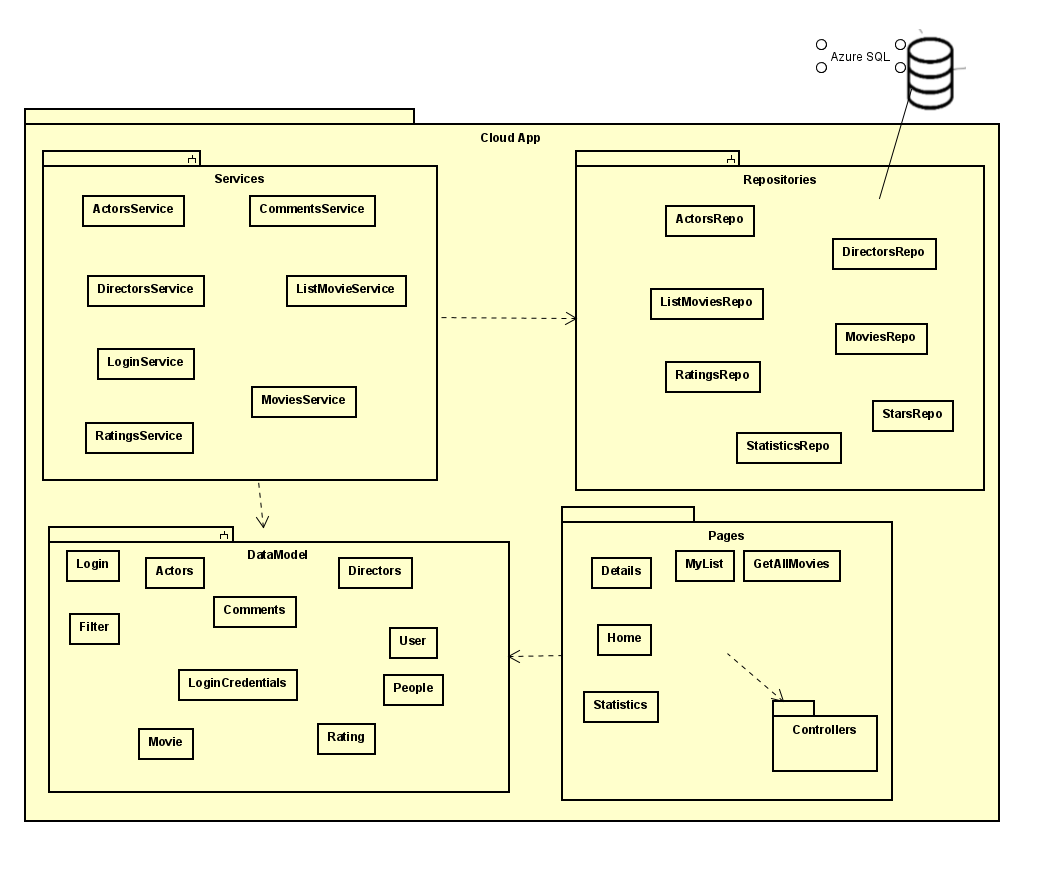
The User selects the option to search a movie and inserts movie details, the system displays the matching movies, and if the system does not find a matching movie, it will display an error message.

3. Design

**3.1 System Overview**

The full system infrastructure is hosted in Azure Cloud.

The WebApp is hosted in a Azure Compute service instance and the database in Azure SQL.

**3.2 Class diagram**

This is the main class diagram the system is organized in 5 main packages:

**Page**: The page class is responsible for the UI design

**Controller**: Each page has it’s own controller that contains all the necessary methods.

**Datamodel**: Each entity in the app has its own data model class.

**Repositories**: All access to the database is through the repository classes.

**Services:** The service classes are called from the controllers to access API and repositories.

**3.4 Design patterns and libraries**

**3.4.1. Razor pages**

Razor Pages is a simplified web application-programming model. It removes much of the ceremony of ASP.NET MVC by adopting a file-based routing approach. Each Razor Pages file found under the Pages directory equates to an endpoint. Razor Pages have an associated C# objected called the page model, which holds each page's behaviour. Additionally, each page works on the limited semantics of HTML, only supporting GET and POST methods.

**3.4.2. Repository pattern**

All operations related to data in Azure SQL are accessed through the repository class to abstract the data sources from the rest of the app.

**3.4.3. Charts**

For a better understanding of the movie statistics values that are displayed in bar charts, using Fusion Charts library.

**3.4.3. Firebase Authentication**

Users account creation and login features achieved using the Firebase API.

**3.5 Data storage**

The provided database source is movies.db, with new entities