PISA UNIVERSITY

DATA MINING AND MACHINE LEARNING MODULES

"PISAFLIX" PROJECT DOCUMENTATION

ACADEMIC YEAR 2019-2020

STEFANO PETROCCHI

SUMMARY

DESIGN DOCUMENT

For sake of brevity only the information about the *Data Mining and Machine Learning* modules are described.

DESCRIPTION

Have you ever found yourself in a gloomy day? Everyone is at home, no one knows what to do and time seems to slow down. That's the perfect time for a movie!

PisaFlix is a platform in which users can find quality and updated **information** regarding movies. It provides a service of **suggestions** to help you to choose what film to watch, based on what you like. PisaFlix has a **comment** section that gives at the users the possibility to create a community around their favourite movies, exchanging opinions and news regarding them. It is also possible to add films to a **favourite** list in order to find them quicker ad receive suggestions. The possibility to see other users favourites it is essential to find new friends with the same cinematic tastes. Lastly it is possible to personalize the **safe search** level to avoid showing content not suitable for children.

PisaFlix offers services that will change the way users approach the world of the movie, providing them everything they need to enjoy at best their passions.

REQUIREMENTS

MAIN ACTORS

The application will interact only with the **users**, distinguished by their privilege level:

- **Normal User**: a normal user of the application with the possibility of basic inaction.

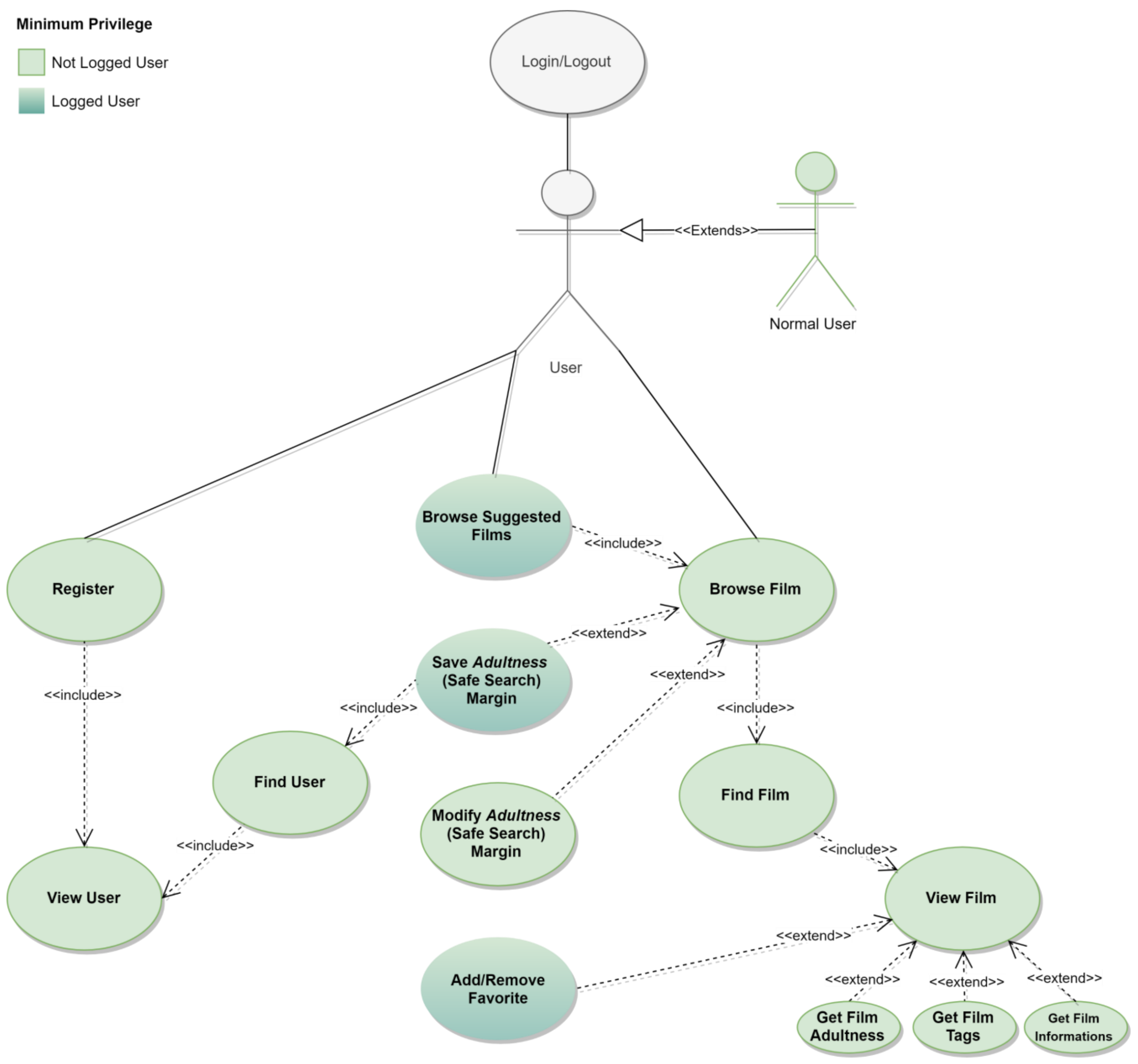
FUNCTIONAL

1. *Users* can **search** for the **movies** available on the platform.
2. *Users* can **view** the information about a specific *Movie*.
3. *Users* can **register** an account on the platform.
4. *Users* can **log in** as *Normal users* on the platform in order to do some specific operations:
   a. If logged a *Normal user* can **add/remove** to **favourite** a *Movie*.
   b. If logged a *Normal user* can **view** his *suggested movies*.
   c. If logged a *Normal user* can **modify** his *safe search level*.

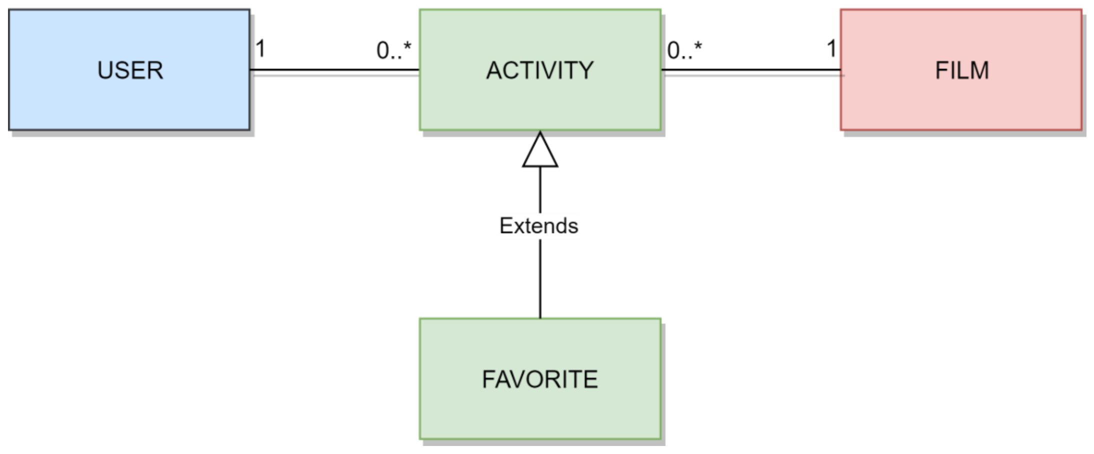
NON-FUNCTIONAL

1. A way to **periodically** recalculate suggested movies must be supplied.
2. A way to **modularly** calculate movies *adultness* must be supplied.

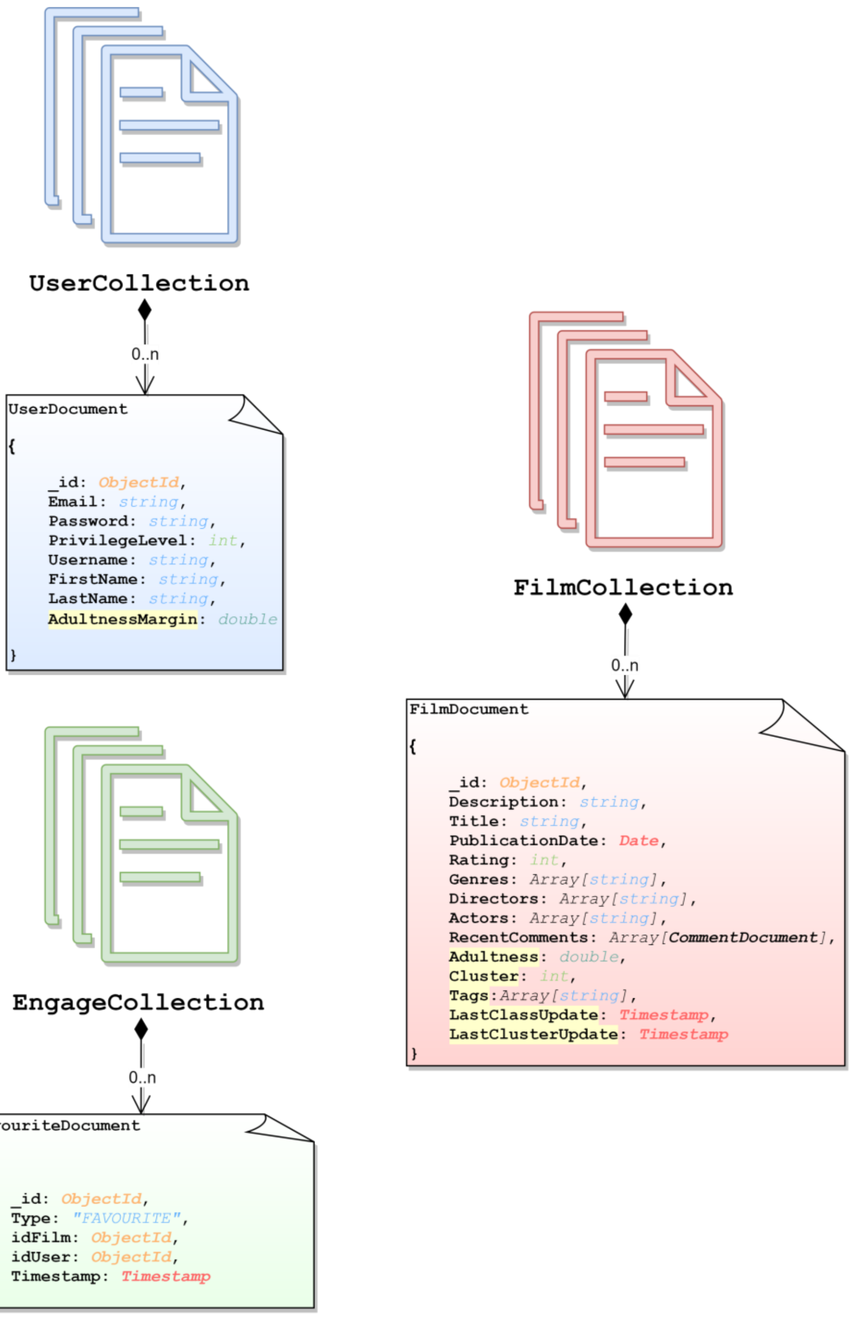
USE CASES



ANALYSIS CLASSES



DATA MODEL



```
UserDocument
{
    _id: ObjectId,
    Email: string,
    Password: string,
    PrivilegeLevel: int,
    Username: string,
    FirstName: string,
    LastName: string,
    AdultnessMargin: double
}
```

```
FilmDocument
{
    _id: ObjectId,
    Description: string,
    Title: string,
    PublicationDate: Date,
    Rating: int,
    Genres: Array[string],
    Directors: Array[string],
    Actors: Array[string],
    RecentComments: Array[CommentDocument],
    Adultness: double,
    Cluster: int,
    Tags:Array[string],
    LastClassUpdate: Timestamp,
    LastClusterUpdate: Timestamp
}
```

```
FavouriteDocument
{
    _id: ObjectId,
    Type: "FAVOURITE",
    idFilm: ObjectId,
    idUser: ObjectId,
    Timestamp: Timestamp
}
```

APPLICATION ARCHITECTURE

Users can use a java application with a **GUI** to take advantage of all the functionalities of the platform.

The client Application is made in *Java* using **JavaFX framework** for the *front-end* and the **MongoDB driver** to manage *back-end* functionalities. **Services** and ***JavaBean* objects** compose the *middleware* infrastructure that connect *front-end* and *back-end.*

INTERFACE DESIGN PATTERN

The graphic user interface was build following the software design pattern of **Model-View-Controller**.

MODEL

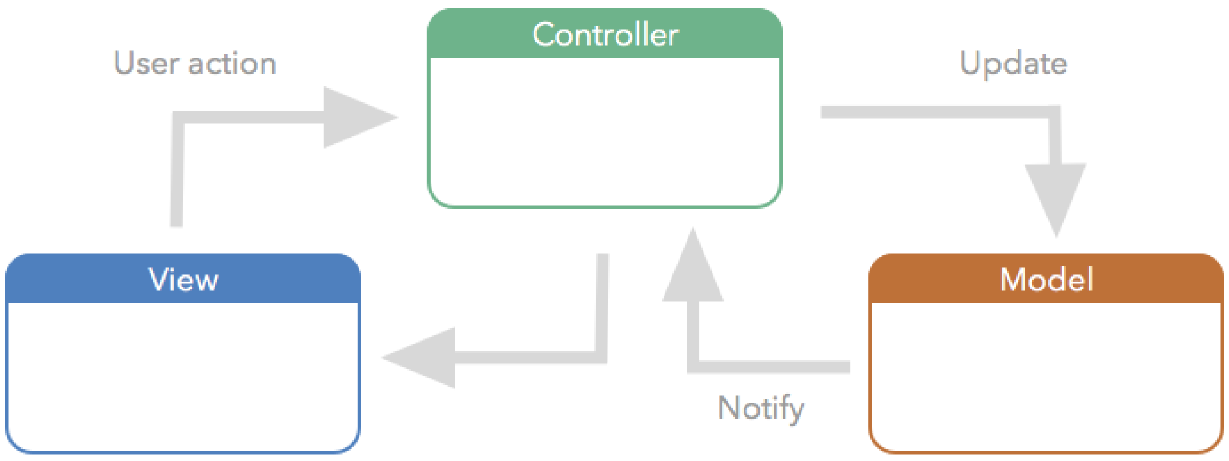
Services module represents the *model* and it's the central component of the pattern. It is the application's dynamic data structure, independent of the user interface. It directly manages logic and rules of the application receiving inputs from the controller. The model is also responsible for managing the application's data in form of JavaBean objects, exchanging them with the controller.

VIEW

The **fxml files** represents the *view* and are responsible for all the components visible in the user's interface.

CONTROLLER

The **page controllers** are the *controller* of the application. They receive inputs from the *view* and convert them into commands for the *model* or *view* itself. Controllers can also validate inputs and data without the intervention of the *model*. Data is exchanged between *model* and *controller* using JavaBean objects.



MACHINE LEARNING

PROGRAMMING LANGUAGE AND LIBRARIES

For the operations of machine learning ***Python*** language has been chosen as it allows easy manipulation of data due to its characteristics and thanks to the libraries that can be used with it.

In particular the libraries used are:

- ***Pandas*** library used to manipulate data in *dataframe* format
- ***Sklearn*** library containing various machine learning algorithms
- ***Scipy*** library containing the optimization algorithms

INSTALLATION

Before using the scripts is needed to install the following libraries in addition to Python 3.8:

- ***Nltk, Numpy, Pandas, Scikit-learn, Scipy, Sklearn*** it is also necessary to download the files containing the ***stop words*** and ***names*** for the vectorization of the film’s description using the following code:

- *nltk.download('punkt')*
- *nltk.download('stopwords')*
- *nltk.download('names')*

MAIN CLASSES AND SCRIPTS

The whole code is commented in detail therefore only the main operations carried out and their results will be highlighted.

The project’s code can be found within the path: ***Task2\PisaFlix\src\main***

PYTHON SCRIPTS (RESOURCES\DATAMINING\SCRIPTS)

PREPROCESSING.PY

It contains the functions for the different **preprocessing** operations.

Its execution produces the file *preprocessedData.csv* which is used for classification, clustering and for the various comparisons.

DIFFERENTIAL_EVOLUTION.PY

Contains the code for **optimizing** the *tf-idf* preprocessing parameters by means of a ***differential evolution*** algorithm

PARAMETER_OPTIMIZATION_CHI2.PY

Contains the code for ***optimizing*** the *chi2* preprocessing parameters by means of a **brute force** algorithm.

CLASSIFIERS_COMPARISON.PY

Uses *preprocessedData.csv* to calculate ***AUC***, ***accuracy*** and ***confusion matrix*** statistics for the comparison of the various **classifiers**.

CLUSTERING_COMPARISON.PY

Uses *preprocessedData.csv* to calculate the clustering algorithms ***elbow graph*** of the ***silhouette*** and the max number of films in each cluster of each clustering algorithms.

CLASSIFIER_CHI2.PY

It **classifies** the films present in *to_be_classified.csv* using the **same vocabulary** as the films in *preprocessedData_chi2.csv* calculated with the *feature selection chi2*.

CLASSIFIER_TF-IDF.PY

It **classifies** the films present in *to_be_classified.csv* using the **same vocabulary** as the films in *preprocessedData_tf-idf.csv* calculated with the *feature selection* of *tf-idf*.

CLASSIFIER.PY

Is the script called by the class *Classifier.java.*

It **preprocess** the films present in *to_be_classified.csv* after merging them with *labelledDat.csv* (the labelled films used for training) to extract the combined features.

Labelled data is then used to **train** the **model** that is used to classify the films.

The result classes are then print in the ***standard output*** in the same order of the films in *to_be_classified.csv.*

CLUSTERIZER.PY

Is the script called by the class *Clusterizer.java.*

It **preprocess** the films present in *to_be_clusterized.csv* and then calculate the ***cluster*** of membership of each film.

The **most important features** of each cluster are also calculated, and all the **results** are written in the file *clustering_results.txt*

JAVA CLASSES

CLASSIFIER.JAVA

Is the class that manages the classification of all the films in the database.

Cyclically take a **sample** of films not yet classified from the database, perform the **classification** by saving the films in *to_be_classified.csv* and calling the script *classifier.py*, **picks up** the results from the *standard output* of the script and then **update** the relative fields of the classified films in the database.

CLUSTERIZER.JAVA

Is the class that manages the clustering of all the films in the database.

It takes **all** the films (or a sample like the classifier) from the database, perform the **clustering** by saving the films in *to_be_clusterized.csv* and calling the script *clusterizer.py*, **picks up** the results from *clustering_results.txt* and then **update** the relative fields of the clustered films in the database.

FILMMANAGER.JAVA

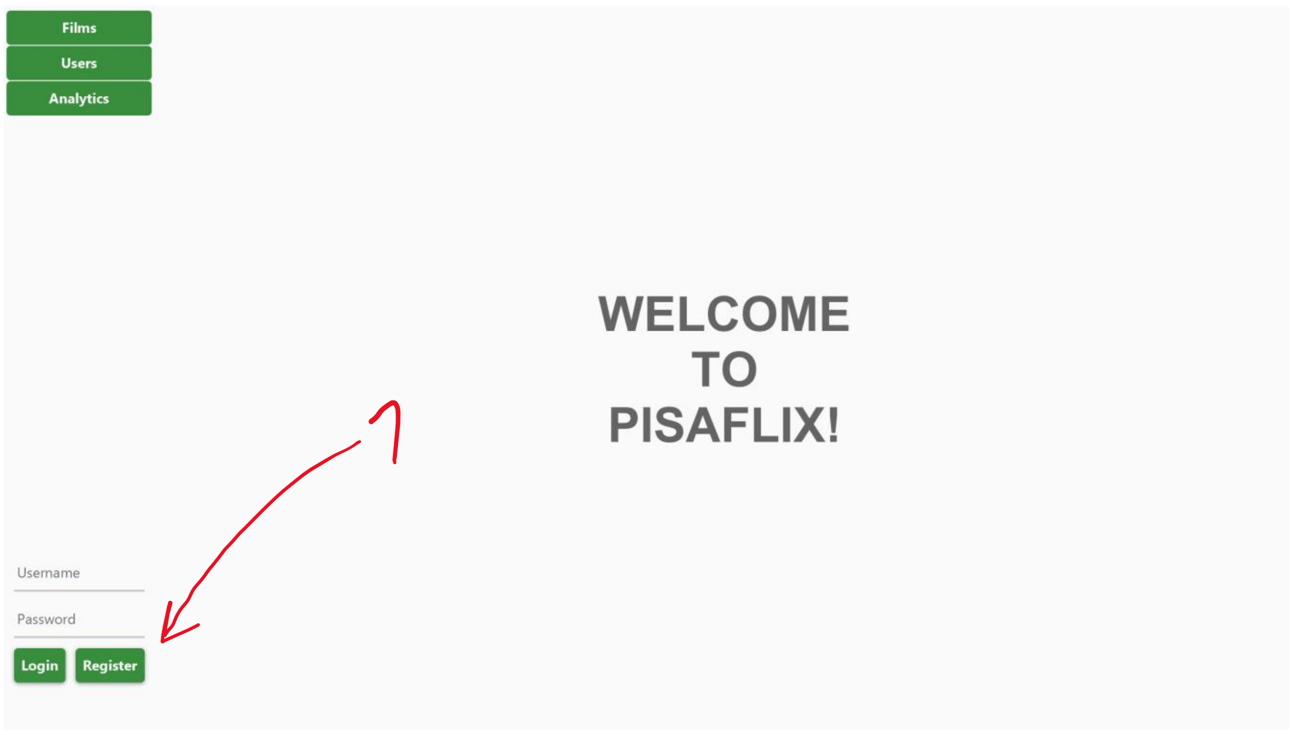
Is the **database manager** for the film's collection

At the end of the file contains all the functions used by the *data mining* tasks to **interface** with the database, it also contains the functions used to collect data for the operation of the modules added to the application.

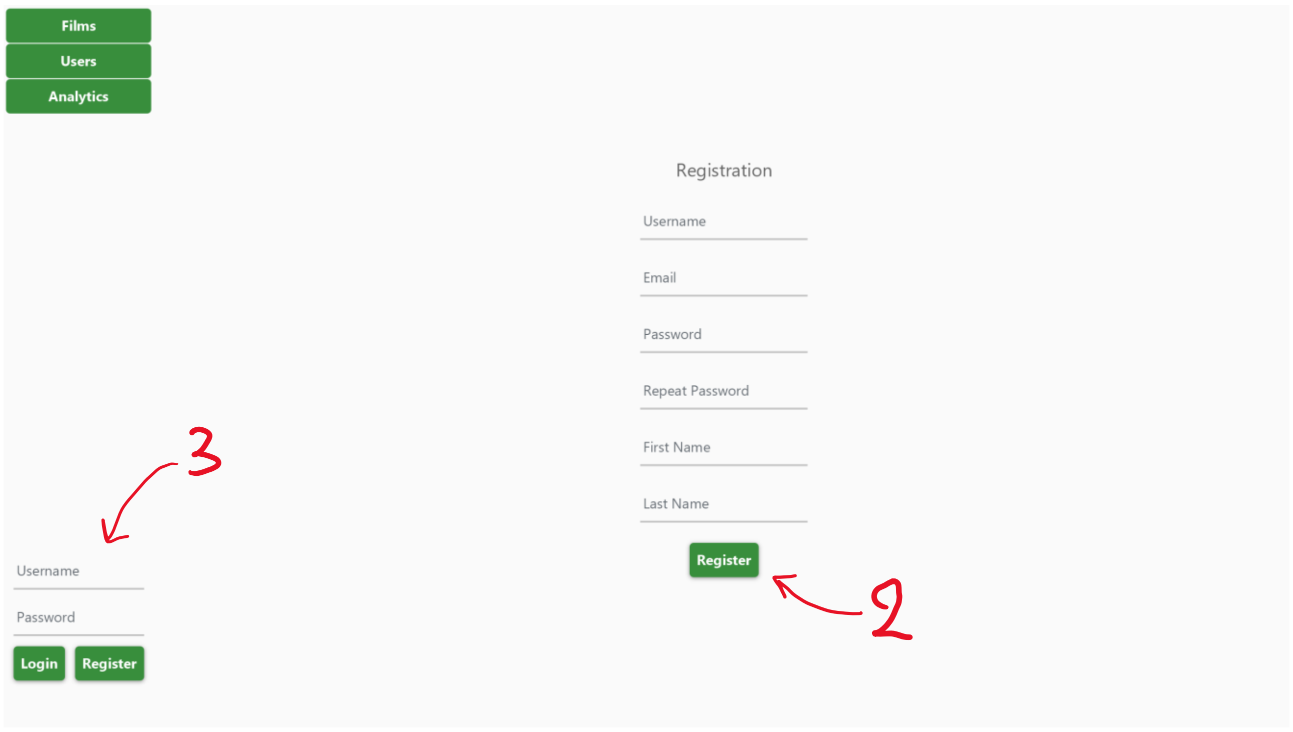
USER MANUAL

REGISTRATION AND LOGIN

1. To open the registration page click on the "Register" button



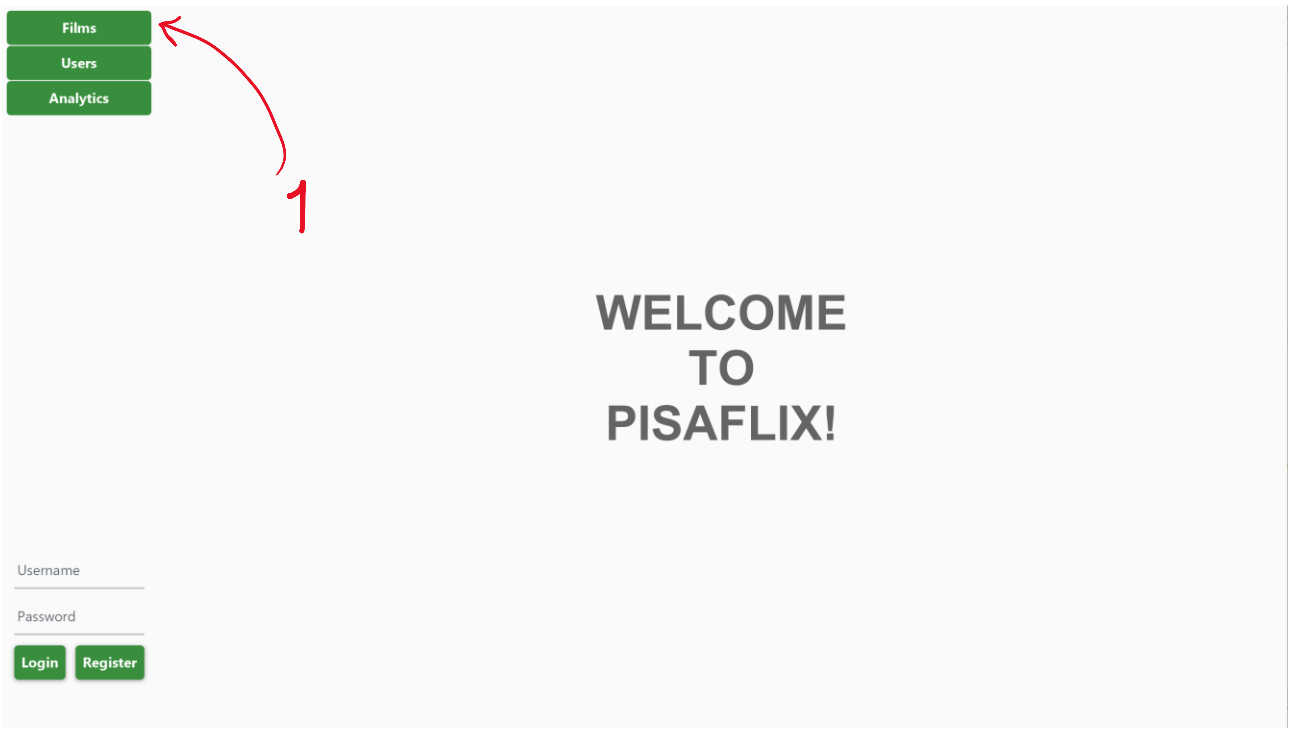
2. Fill the fields with your information and click on "Register"



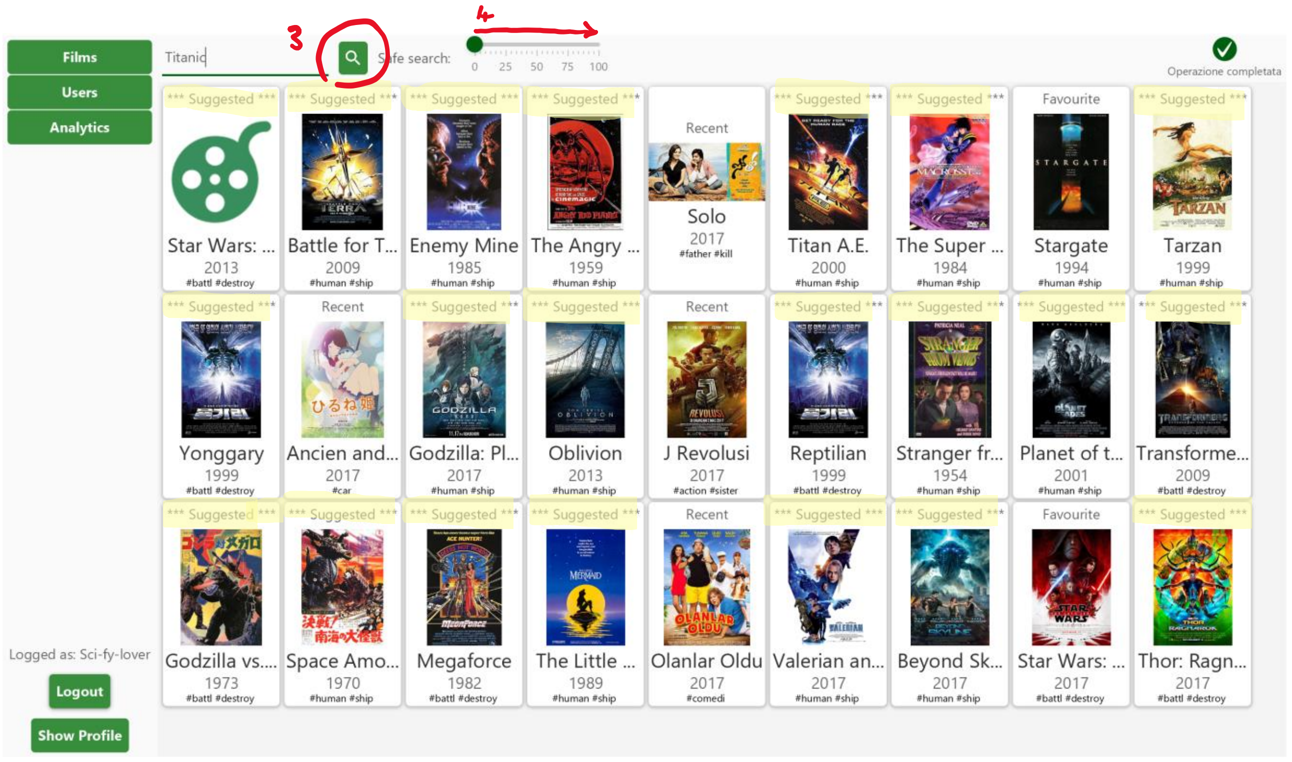
3. Fill the fields and click on "Login" button to login

SAFE SEARCH AND SUGGESTIONS

1. Click on the "Films" button to open the movies search page



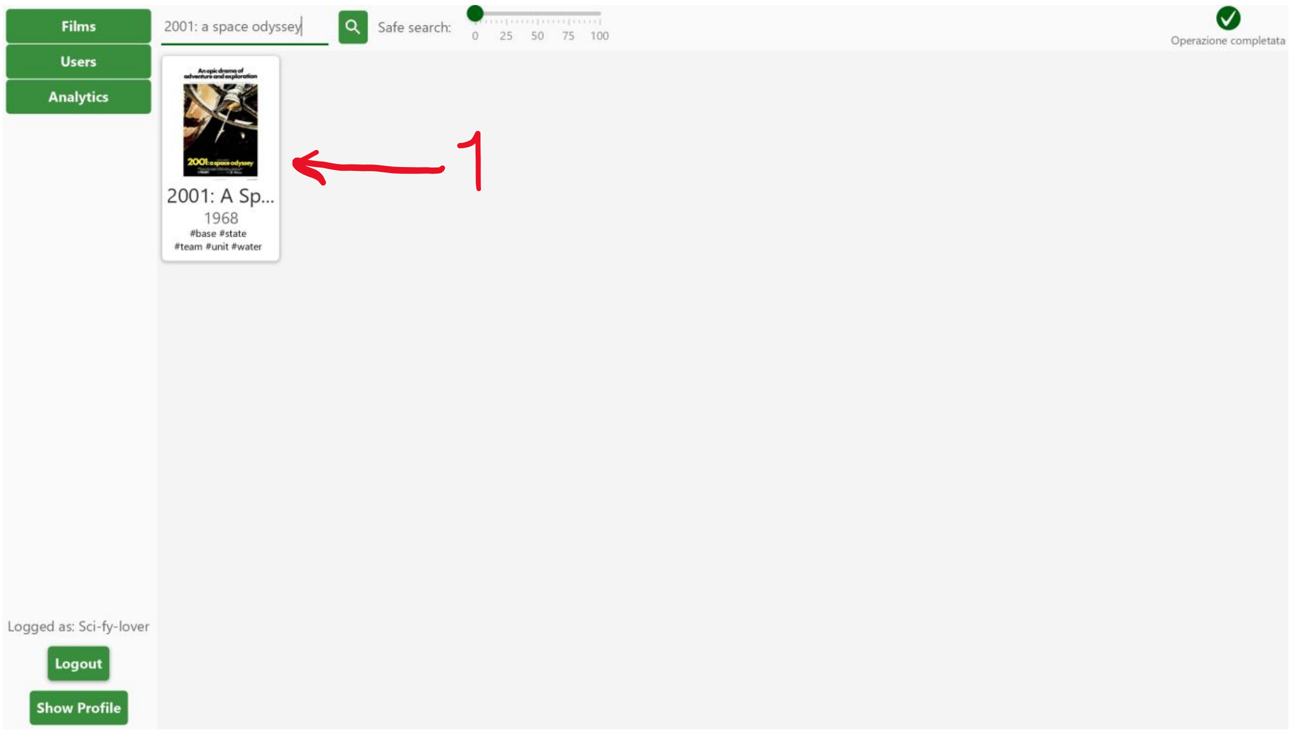
2. If you are logged in you should see suggested movies based on your favourite



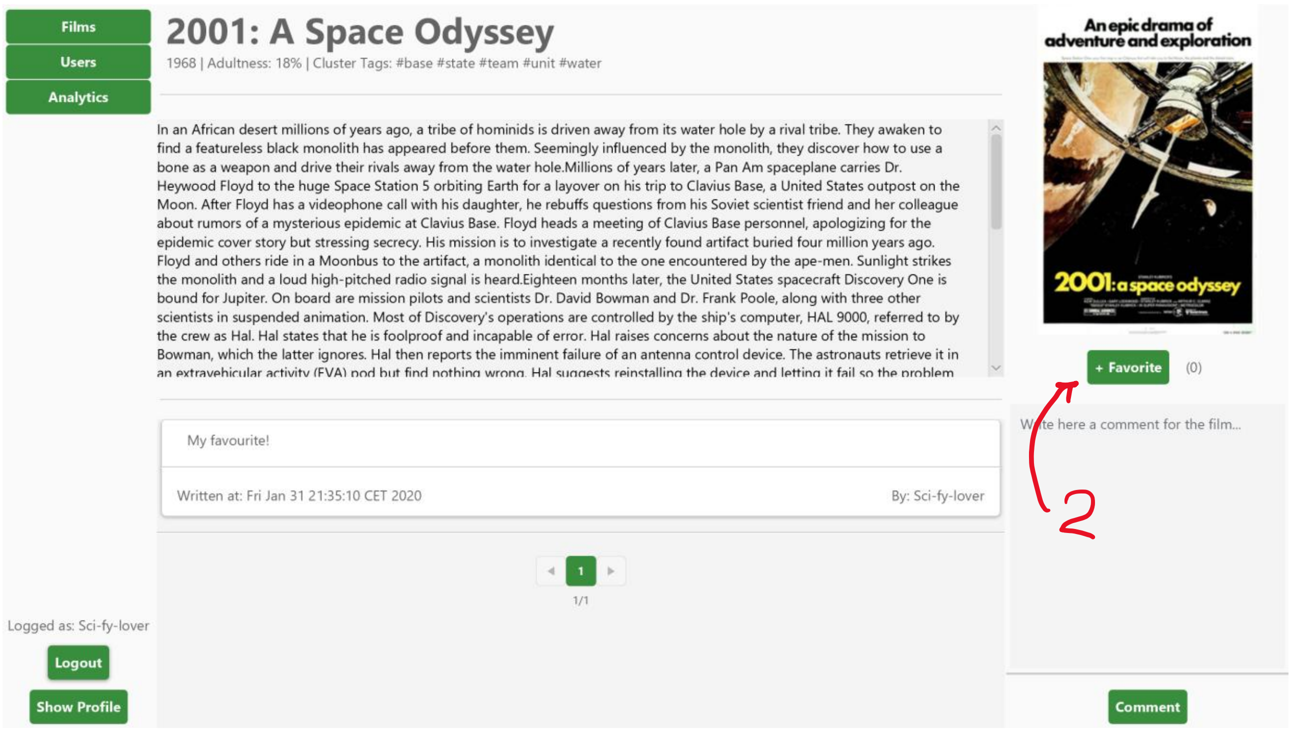
3. To search for a film, type the title or associated tags in the apposite field, then click on the search button
4. To restrict the number of adult content to be displayed, move the slider to the right as desired

ADD FILM TO FAVOURITES

1. To add a film to your favourite list, search for it and then click on its card



2. Click on the "+ Favourite" button


Films
Users
Analytics
2001: A Space Odyssey
1968 | Adultness: 18% | Cluster Tags: #base #state #team #unit #water
+ Favorite
(0)
Write here a comment for the film...
2
Logged as: Sci-fy-lover
Logout
Show Profile
Comment

In an African desert millions of years ago, a tribe of hominids is driven away from its water hole by a rival tribe. They awaken to find a featureless black monolith has appeared before them. Seemingly influenced by the monolith, they discover how to use a bone as a weapon and drive their rivals away from the water hole.Millions of years later, a Pan Am spaceplane carries Dr. Heywood Floyd to the huge Space Station 5 orbiting Earth for a layover on his trip to Clavius Base, a United States outpost on the Moon. After Floyd has a videophone call with his daughter, he rebuffs questions from his Soviet scientist friend and her colleague about rumors of a mysterious epidemic at Clavius Base. Floyd heads a meeting of Clavius Base personnel, apologizing for the epidemic cover story but stressing secrecy. His mission is to investigate a recently found artifact buried four million years ago. Floyd and others ride in a Moonbus to the artifact, a monolith identical to the one encountered by the ape-men. Sunlight strikes the monolith and a loud high-pitched radio signal is heard.Eighteen months later, the United States spacecraft Discovery One is bound for Jupiter. On board are mission pilots and scientists Dr. David Bowman and Dr. Frank Poole, along with three other scientists in suspended animation. Most of Discovery's operations are controlled by the ship's computer, HAL 9000, referred to by the crew as Hal. Hal states that he is foolproof and incapable of error. Hal raises concerns about the nature of the mission to Bowman, which the latter ignores. Hal then reports the imminent failure of an antenna control device. The astronauts retrieve it in an extravehicular activity (EVA) pod but find nothing wrong. Hal suggests reinstalling the device and letting it fail so the problem

My favourite!

Written at: Fri Jan 31 21:35:10 CET 2020 By: Sci-fy-lover

1

1/1