Fake-Amazon Prime

Project: Movie Streaming System

Documentation of the Project

Programming Excercise 2024

Examiner: Serkan Savranoglu

Member:

Jeonghun Kim 1451179

Jakub Wieloch 1354755

Mohammed Taha El Youssefi 1454943

Paolo Yang 1456204

Table of Content

Statutory Declaration

We herewith declare that we have completed the present report independently, without making use of other than the specified literature and aids. All parts that were taken from published and non-published texts either verbally or in substance are clearly marked as such. This report has not been presented to any examination office in the same form.

Frankfurt Am Main, dated 03.07.2024

[illegible]

Signature JeongHun Kim

[illegible]

Signature Jakub Wieloch

[illegible]

Signature Taha El Youssefi

[illegible]

Signature Paolo Yang

Introduction

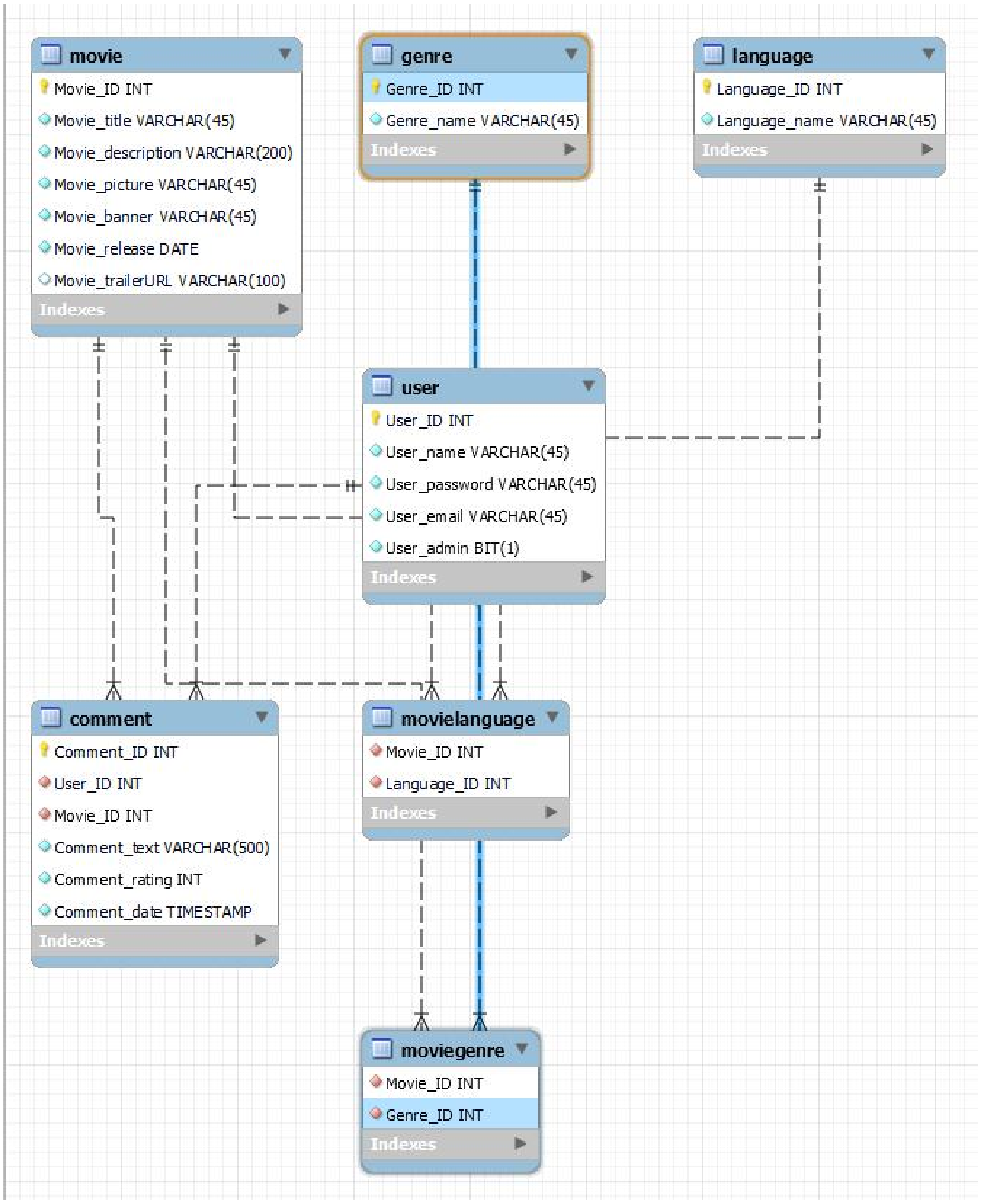
Welcome to our movie streaming service, which aims to improve your entertainment experience. Our goal is to provide a user-friendly solution that makes content discovery easier and enhances the viewing experience. Using a database, we offer a variety of services to both casual viewers and die-hard movie fans.

Our program offers simplicity, efficiency and an amazonlike interface. This interface clearly distinguishes between user roles, resulting in individualized experiences for each viewer. Administrators may manage information and the movie database more efficient, while consumers benefit from an easy browsing experience that is tailored to their interests.

The User-friendly interface is designed to minimize complications and provide a consistent experience for all users. Our program promotes usability and accessibility with straightforward navigation and visually appealing aspects. We help people discover and enjoy movies more quickly by selecting key content and presenting it in a simple and structured manner, rather than being swamped by unnecessary information.

We firmly believe that a well-structured and visually appealing application is essential for increasing user pleasure and engagement. By constantly improving our platform and extending our database, we hope to provide a movie streaming experience that suits our users changing needs.

Organization



Planing 1

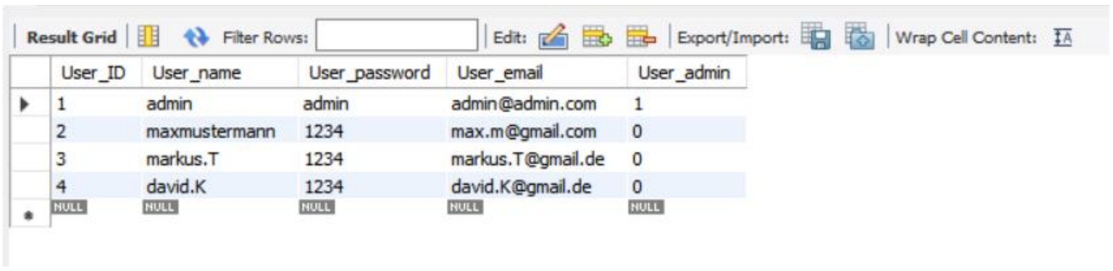
We planed to make an amazon like movie streaming service that users can streaming movies, making comments and choosing movie or subtitle language. The movies are sorted by their genre.
First we started with Java Swing, but after a while we noticed that animations can´t be realized with Java Swing. Then we switched to Javafx

If we had more time, we would have gone farther into improving the platform with more features. For example, we wanted to include many slides with recommended movies to assist viewers find new movies based on their likes. Furthermore, we wanted a dynamic main page with a video playing in the background to create a more engaging and dynamic experience for users while they browsed the material. These enhancements would have dramatically improved the user experience, making our movie streaming service more interactive and user-friendly. We also would have worked on upgrading the design to make the service more user-friendly.
These changes would have made our movie streaming service more enjoyable.

Database

The tables in the Movie Streaming database interact with each other to facilitate various aspects of system.

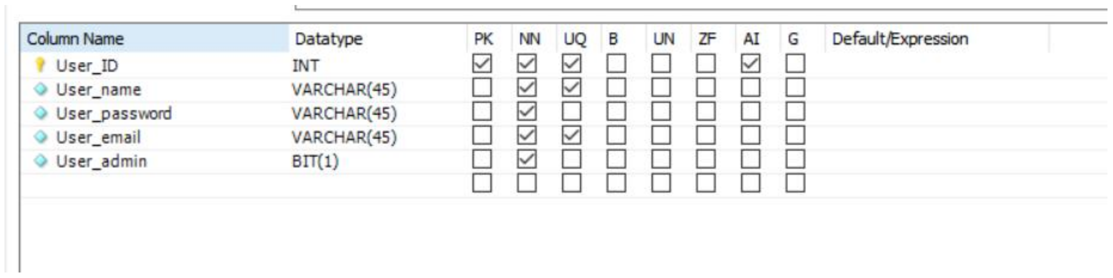
User Table



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

User_ID	User_name	User_password	User_email	User_admin
1	admin	admin	admin@admin.com	1
2	maxmustermann	1234	max.m@gmail.com	0
3	markus.T	1234	markus.T@gmail.de	0
4	david.K	1234	david.K@gmail.de	0
NULL	NULL	NULL	NULL	NULL

User Database 1

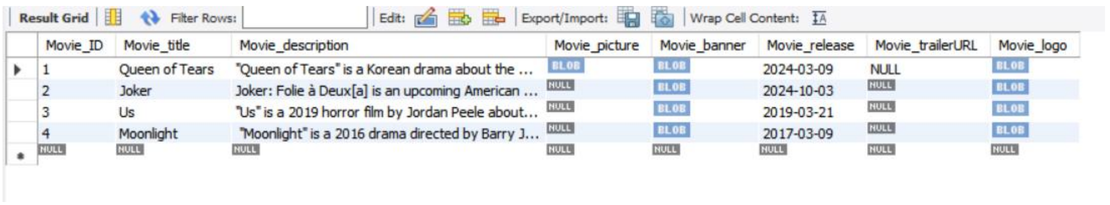


Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
User_ID	INT	✓	✓	✓	☐	☐	☐	✓	☐	
User_name	VARCHAR(45)	☐	✓	✓	☐	☐	☐	☐	☐	
User_password	VARCHAR(45)	☐	✓	☐	☐	☐	☐	☐	☐	
User_email	VARCHAR(45)	☐	✓	✓	☐	☐	☐	☐	☐	
User_admin	BIT(1)	☐	✓	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	☐	

User Database 2

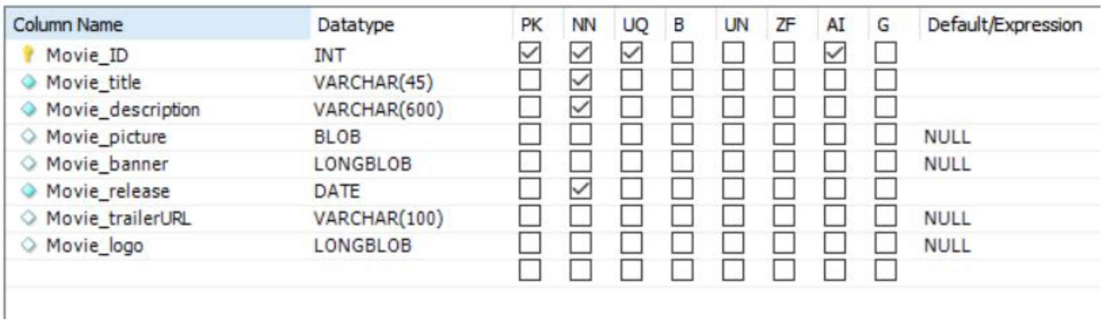
The "User" table represents the administrators and users. It contains informations such as the UserID, Username, email and password. This table allows for the management of administrators and assigning them specific permissions and access rights. The password will be encrypted to guarantee full security, even the administrator can not see the password of the user.

Movie



	Movie_ID	Movie_title	Movie_description	Movie_picture	Movie_banner	Movie_release	Movie_trailerURL	Movie_logo
▶	1	Queen of Tears	"Queen of Tears" is a Korean drama about the ...	BLOB	BLOB	2024-03-09	NULL	BLOB
	2	Joker	Joker: Folie à Deux[a] is an upcoming American ...	NULL	BLOB	2024-10-03	NULL	BLOB
	3	Us	"Us" is a 2019 horror film by Jordan Peele about...	NULL	BLOB	2019-03-21	NULL	BLOB
	4	Moonlight	"Moonlight" is a 2016 drama directed by Barry J...	NULL	BLOB	2017-03-09	NULL	BLOB
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Movie Database 1

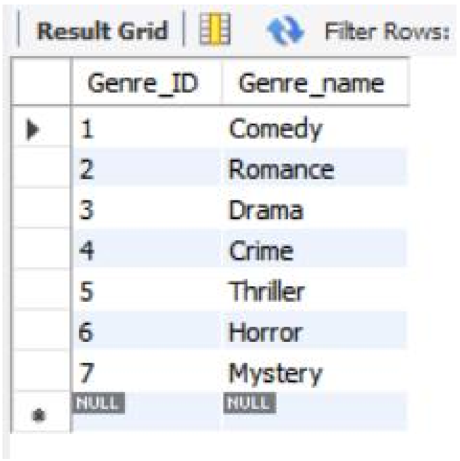


Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
Movie_ID	INT	✓	✓	✓				✓		
Movie_title	VARCHAR(45)		✓							
Movie_description	VARCHAR(600)		✓							
Movie_picture	BLOB									NULL
Movie_banner	LONGBLOB									NULL
Movie_release	DATE		✓							
Movie_trailerURL	VARCHAR(100)									NULL
Movie_logo	LONGBLOB									NULL

Movie Database 2

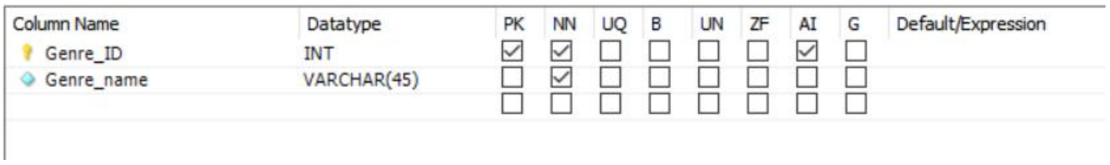
The „Movie“ table represent the Movies. It contains informations such as the MovieID, Title, Description, Release date and Pictures of the movie.

Genre



	Genre_ID	Genre_name
▶	1	Comedy
	2	Romance
	3	Drama
	4	Crime
	5	Thriller
	6	Horror
	7	Mystery
*	NULL	NULL

Genre Database 1

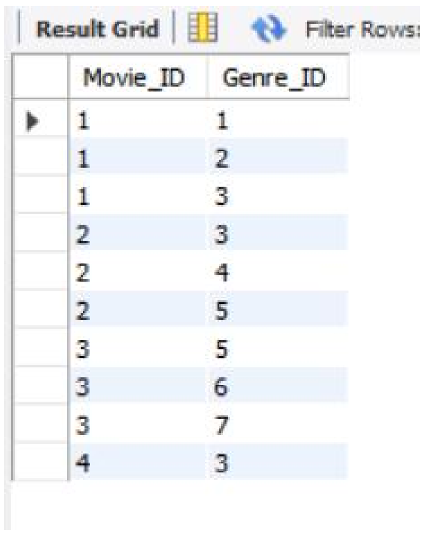


Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
Genre_ID	INT	✓	✓					✓		
Genre_name	VARCHAR(45)		✓							

Genre Database 2

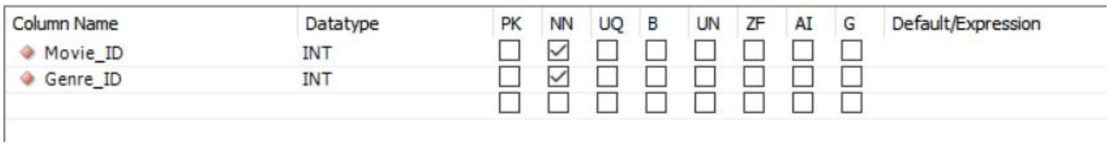
The „Genre“ table represent the Genre. It contains the GenreID and the Genre. There is now seven Genre in our database.

MovieGenre



	Movie_ID	Genre_ID
▶	1	1
	1	2
	1	3
	2	3
	2	4
	2	5
	3	5
	3	6
	3	7
	4	3

Movie Genre Database 1

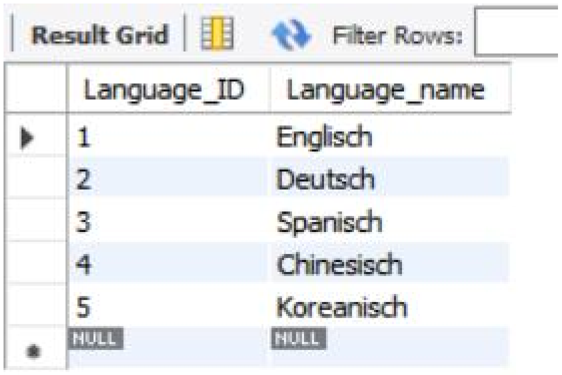


Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
Movie_ID	INT	☐	☑	☐	☐	☐	☐	☐	☐	
Genre_ID	INT	☐	☑	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	☐	

Movie Genre Database 2

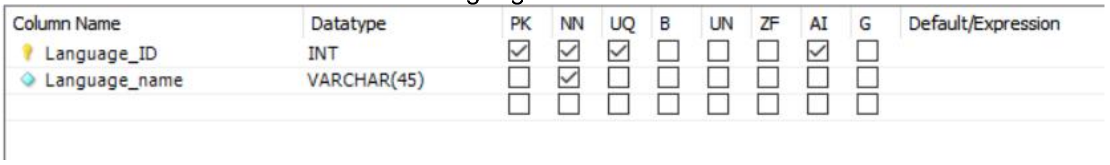
The 'MovieGenre' table represents the genres of the movies, as movies can belong to multiple genre. We add different GenreIDs to the MovieID to specify the genres of the movie.

Language



	Language_ID	Language_name
▶	1	Englisch
	2	Deutsch
	3	Spanisch
	4	Chinesisch
	5	Koreanisch
●	NULL	NULL

Language Database 1

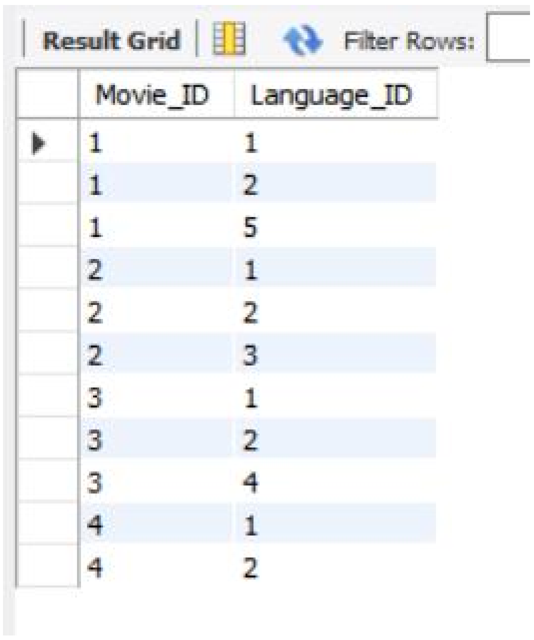


Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
Language_ID	INT	☑	☑	☑	☐	☐	☐	☑	☐	
Language_name	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	☐	

Language Database 2

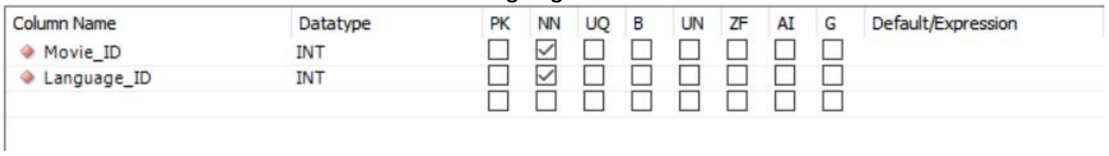
The „Language“ table represent the subtitle language. It contains the LanguageID and the Language. There is now five Languages in our database.

MovieLanguage


Result Grid | Filter Rows:

Movie_ID	Language_ID
1	1
1	2
1	5
2	1
2	2
2	3
3	1
3	2
3	4
4	1
4	2

Movie Language Database 1

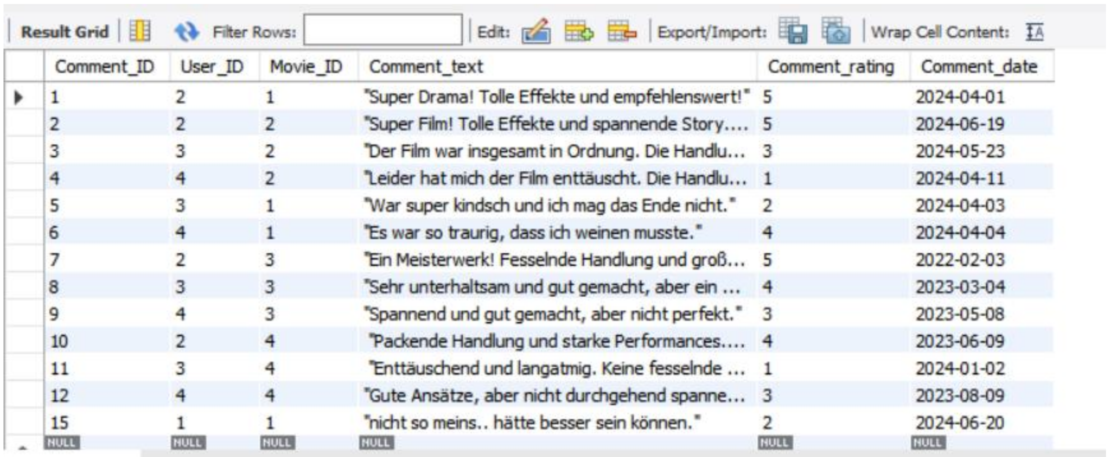


Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
Movie_ID	INT	☐	☑	☐	☐	☐	☐	☐	☐	
Language_ID	INT	☐	☑	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	☐	

Movie Language Database 2

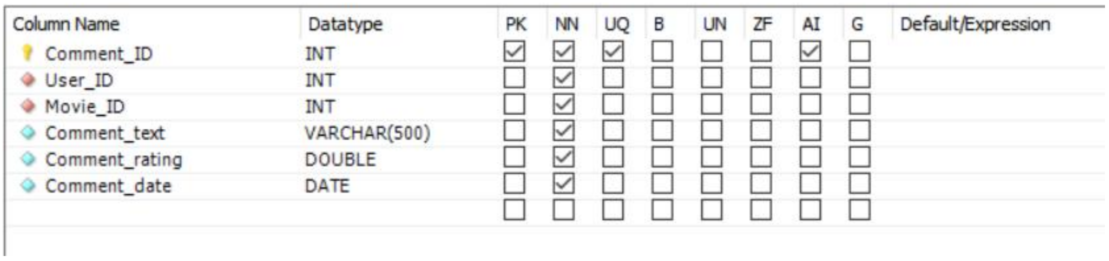
The 'MovieLanguage' table represents the subtitle language of the movies, as subtitle language can belong to multiple movies. We add different LanguageIDs to the MovieID to specify the subtitle language of the movie.

Comment



Comment_ID	User_ID	Movie_ID	Comment_text	Comment_rating	Comment_date
1	2	1	"Super Drama! Tolle Effekte und empfehlenswert!"	5	2024-04-01
2	2	2	"Super Film! Tolle Effekte und spannende Story....	5	2024-06-19
3	3	2	"Der Film war insgesamt in Ordnung. Die Handlu...	3	2024-05-23
4	4	2	"Leider hat mich der Film enttäuscht. Die Handlu...	1	2024-04-11
5	3	1	"War super kindsch und ich mag das Ende nicht."	2	2024-04-03
6	4	1	"Es war so traurig, dass ich weinen musste."	4	2024-04-04
7	2	3	"Ein Meisterwerk! Fesselnde Handlung und groß...	5	2022-02-03
8	3	3	"Sehr unterhaltsam und gut gemacht, aber ein ...	4	2023-03-04
9	4	3	"Spannend und gut gemacht, aber nicht perfekt."	3	2023-05-08
10	2	4	"Packende Handlung und starke Performances....	4	2023-06-09
11	3	4	"Enttäuschend und langatmig. Keine fesselnde ...	1	2024-01-02
12	4	4	"Gute Ansätze, aber nicht durchgehend spanne...	3	2023-08-09
15	1	1	"nicht so meins.. hätte besser sein können."	2	2024-06-20
NULL	NULL	NULL	NULL	NULL	NULL

Comment Database 1

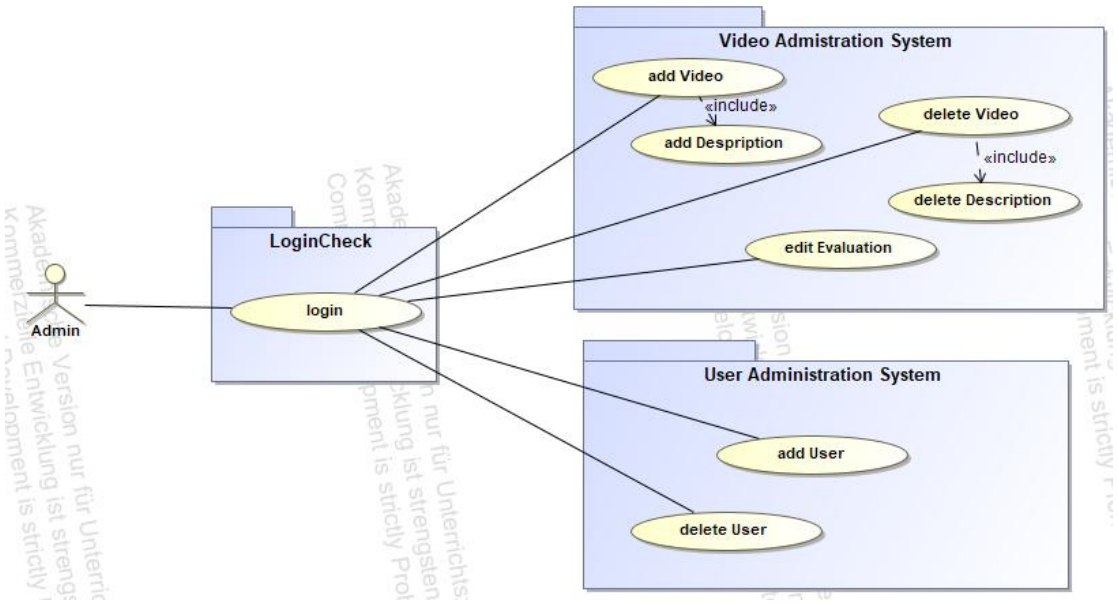


Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
Comment_ID	INT	✓	✓	✓				✓		
User_ID	INT		✓							
Movie_ID	INT		✓							
Comment_text	VARCHAR(500)		✓							
Comment_rating	DOUBLE		✓							
Comment_date	DATE		✓							

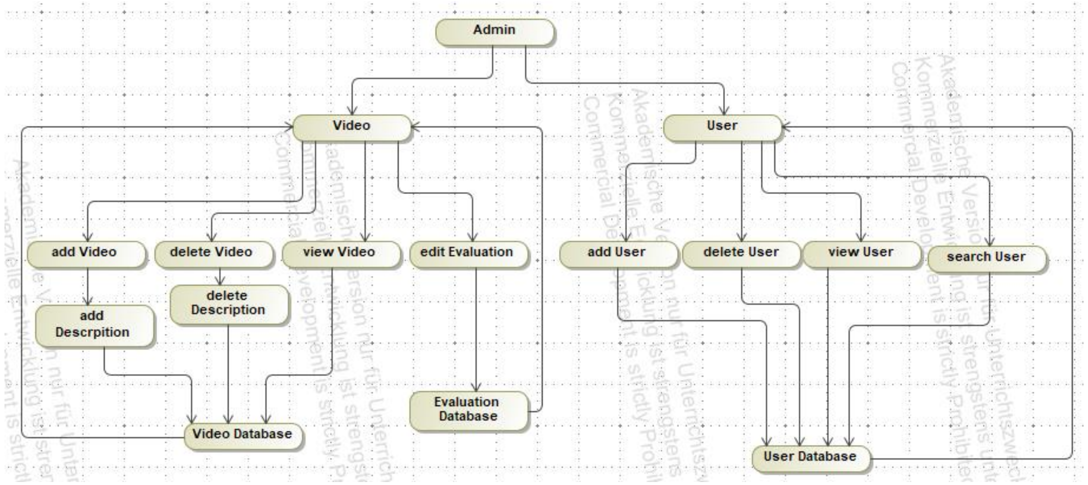
Comment Database 2

The „Comment“ table represent the Comments, Reviews and Evaluations. It contains informations such as the CommentID, UserID, MovieID, Comment Text, Rating and Comment date of the movie.

Function of Administrators



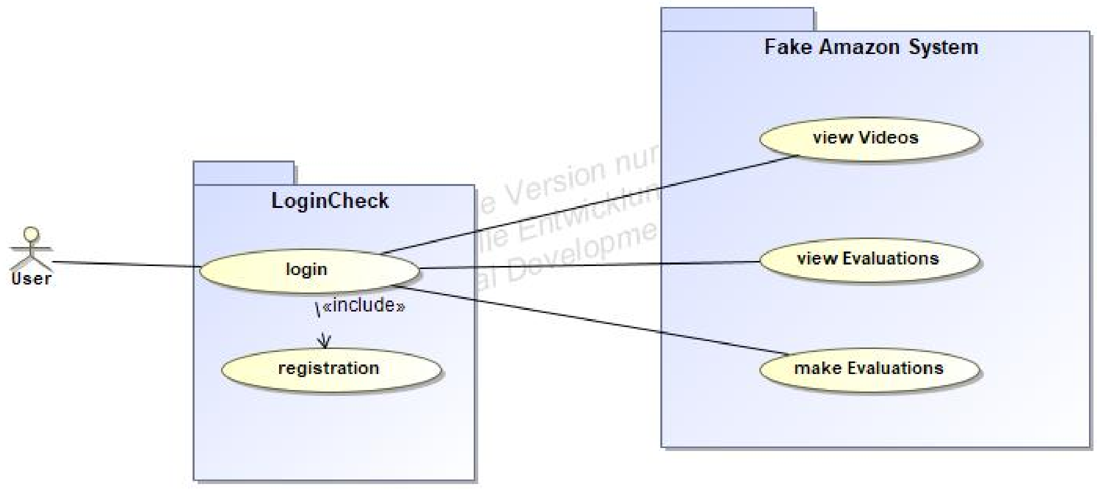
Admin Use Case 1



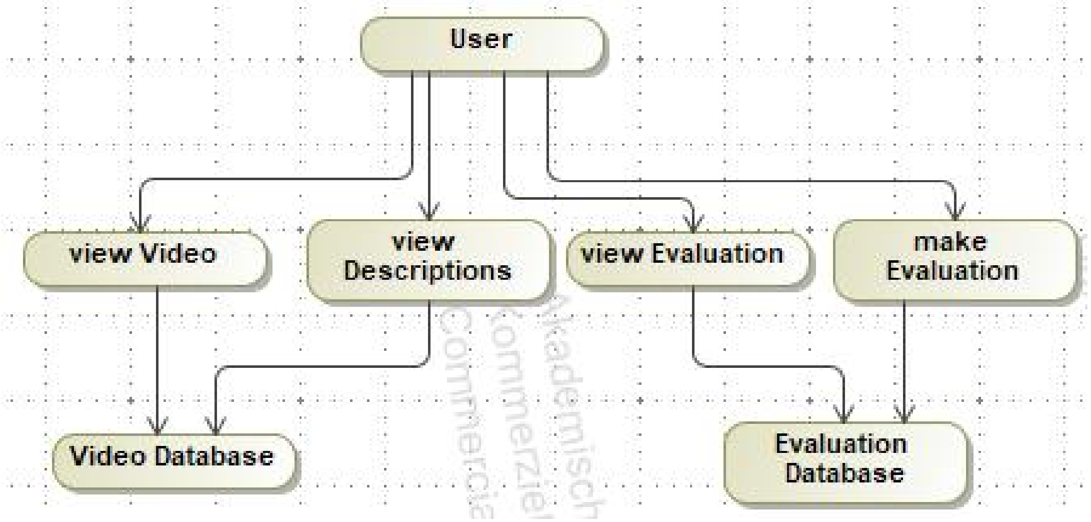
Admin activity 1

The administrator can log in to the system using the LoginCheck to access the video administration or user administration system. That ensures security and offers secure access to administrative content. The administrator can add new videos to the Video Database or delete existing videos from the Video Database to expanding the library or to ensure the content to remain up to date. After adding or deleting videos, the administrator can add descriptions to the videos or delete descriptions from the videos. The administrator also has access to edit the comments, evaluations, or reviews of the videos. That ensures the feedback is appropriate and useful for users.
The administrator has access to the user administration system only through the database. To successfully manage user accounts, the administrator can add or remove existing inactive users from the database. The administrator can access and search for detailed information on users in the database to aid in user management and assistance.

Functions of Users



User Use Case 1



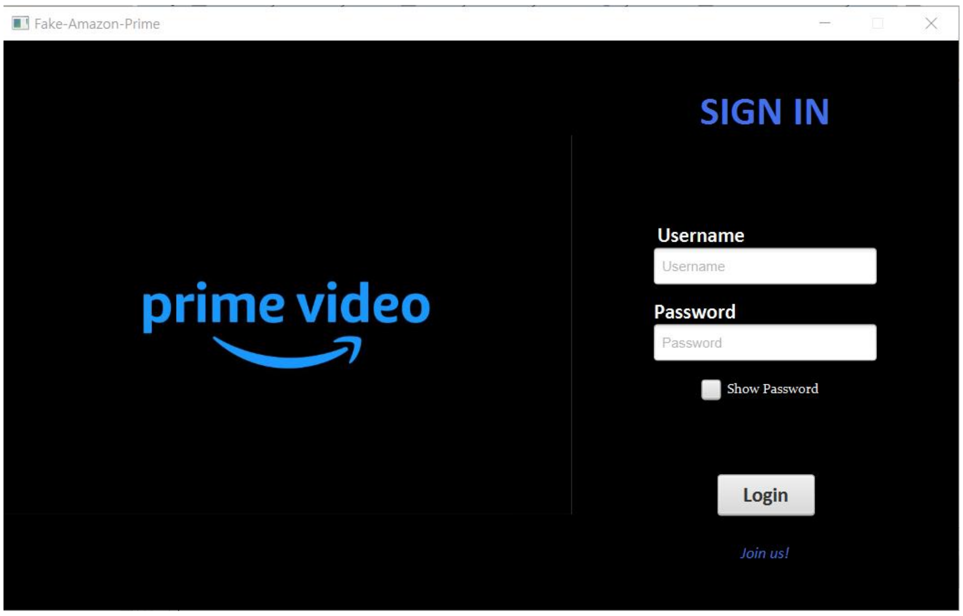
User activity 1

In our movie streaming service, the user has access to several functionalities to enhance the streaming experience and allow the user to interact with the content effectively. The user has to log in to the system through the LoginCheck functionality. That ensures security and offers secure access to personalized account content. If the user does not have an account, there is the possibility to register. The login data will be added to the database.
Once the user is logged into the system, users can browse and watch a wide range of videos available in the video database. Users can also view detailed descriptions of the videos, providing them with additional information about the content.
Users can also browse detailed descriptions of the videos, which give them more information about the topic.
Users can also see comments, evaluations, and reviews for the videos. This feature allows people to make more informed choices about what to watch based on feedback from other viewers. The evaluations are stored in the Evaluation Database. Users can also contribute to the community by leaving

comments, evaluations, or reviews about the videos they have watched. This comment has been stored in the Evaluation Database.

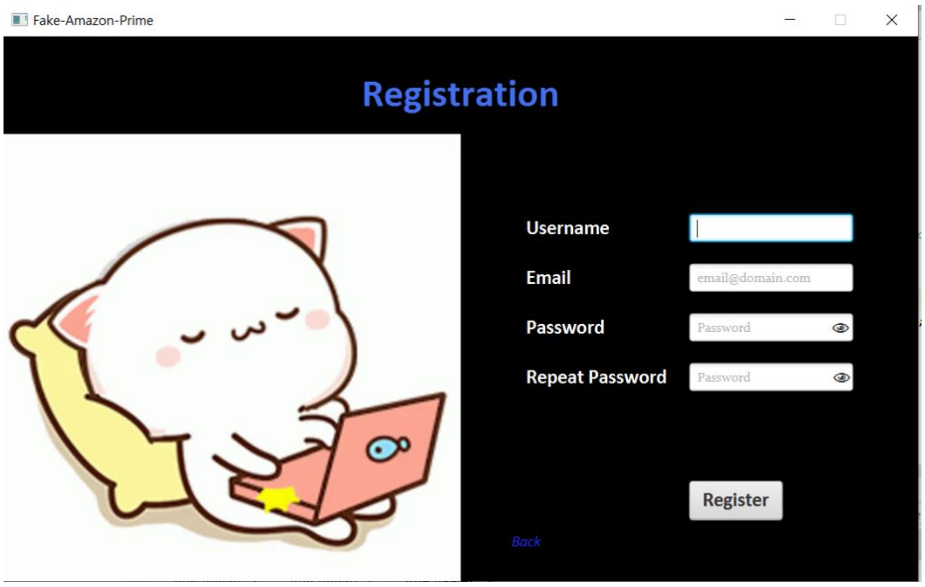
Design

To provide a user-friendly interface for our Movie Streaming system, we decided to create it with JavaFX. Our goal was to create a well-structured and clear user experience for both users and administrators.



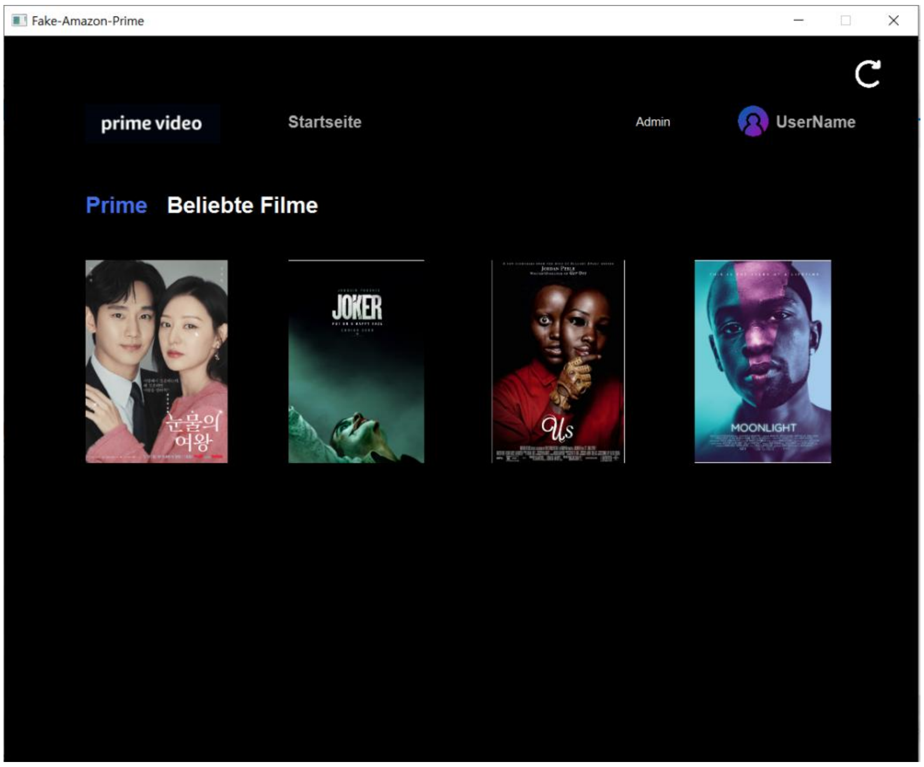
LoginPage 1

The visual design of our interface features a simple black background, which promotes a sense of cleanliness and clarity. To offer visual signals and increase usability, we added blue Logo and headlines and white buttons and menu bars.The use of a consistent color scheme throughout the interface enhances visual coherence and makes it easier for users and administrators to locate and interact with different elements. By opting for a plain and well-structured design, we aim to reduce visual clutter and ensure that users and administrators can focus on the core functionalities of the Movie Streaming system. Our interface design prioritizes usability and simplicity, allowing users and administrators to navigate the system efficiently without being overwhelmed by unnecessary visual elements.



RegistrationPage 1

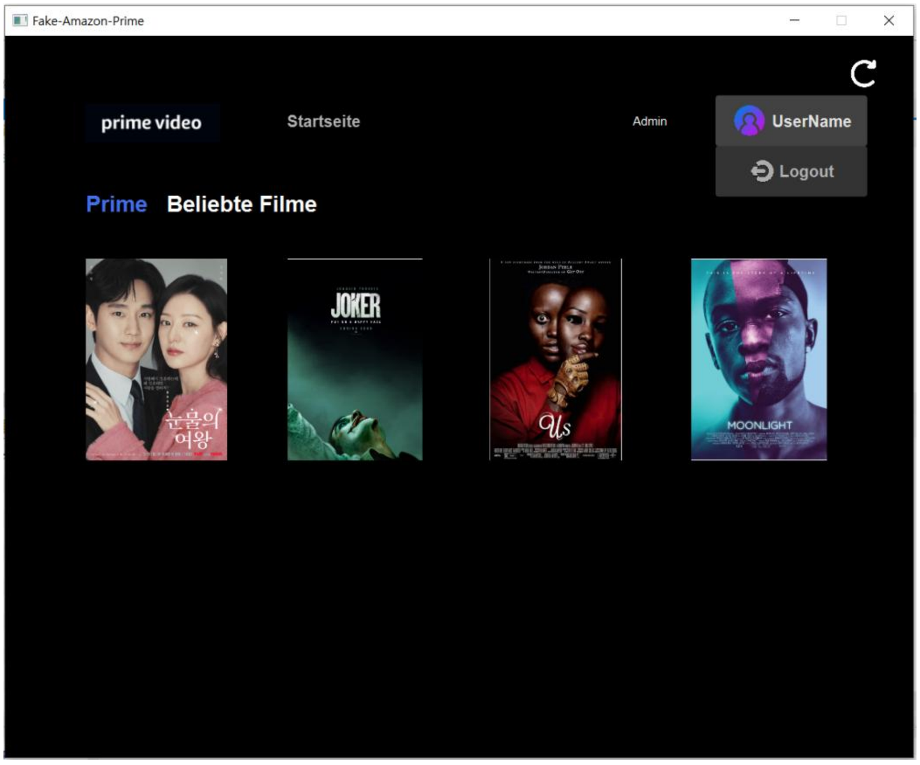
Here is the registration page for users who don’t have an account. The data will be stored in the User Database. The email must be a real email, and the passwords must match.



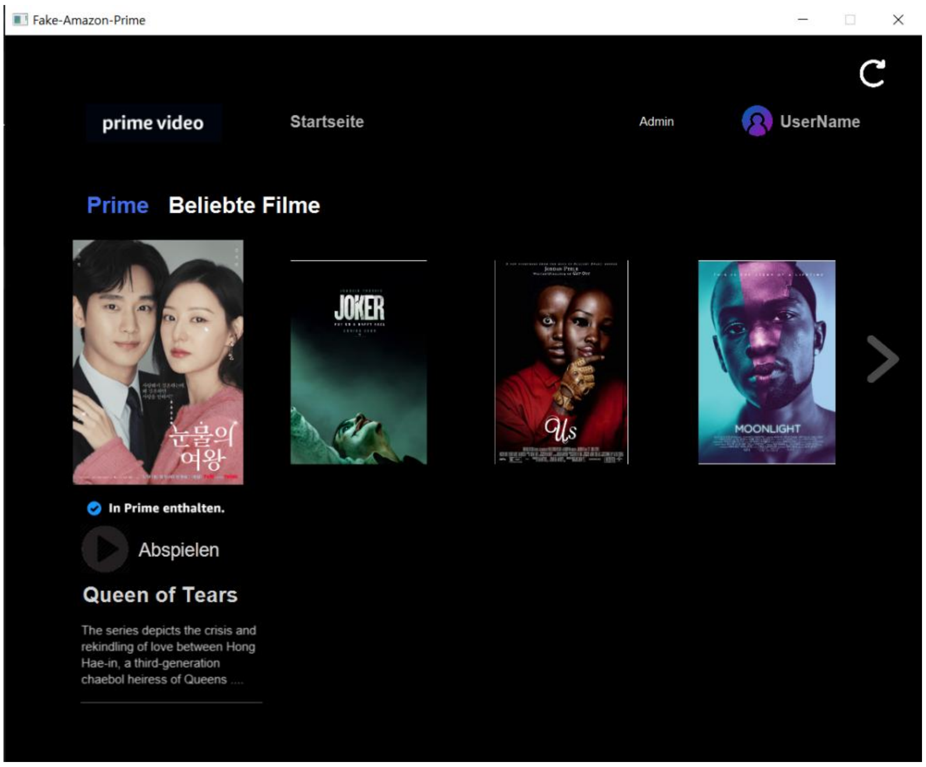
MainPage 1

After the Login you will get to the Mainpage of our Movie Streaming System. At the top, there is a navigation bar. On the left side, there are the logo „prime video“ and a lable „Startseite“. On the right side, there is a button „Admin“ to add or delete video, a user navigation icon to log out of the system and a refresh button to refresh after editing videos.

Below the navigation bar, there is a Slide of recommendations „Beliebte Filme“ displaying four Movies.



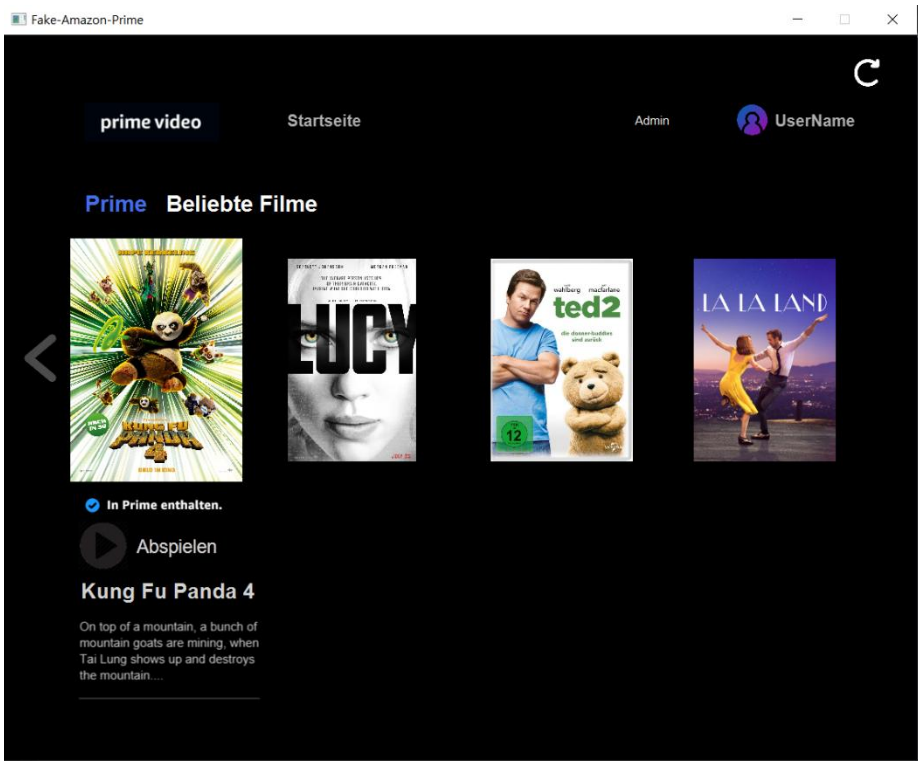
MainPage 2



The series depicts the crisis and rekindling of love between Hong Hae-in, a third-generation chaebol heiress of Queens ....

MainPage 3

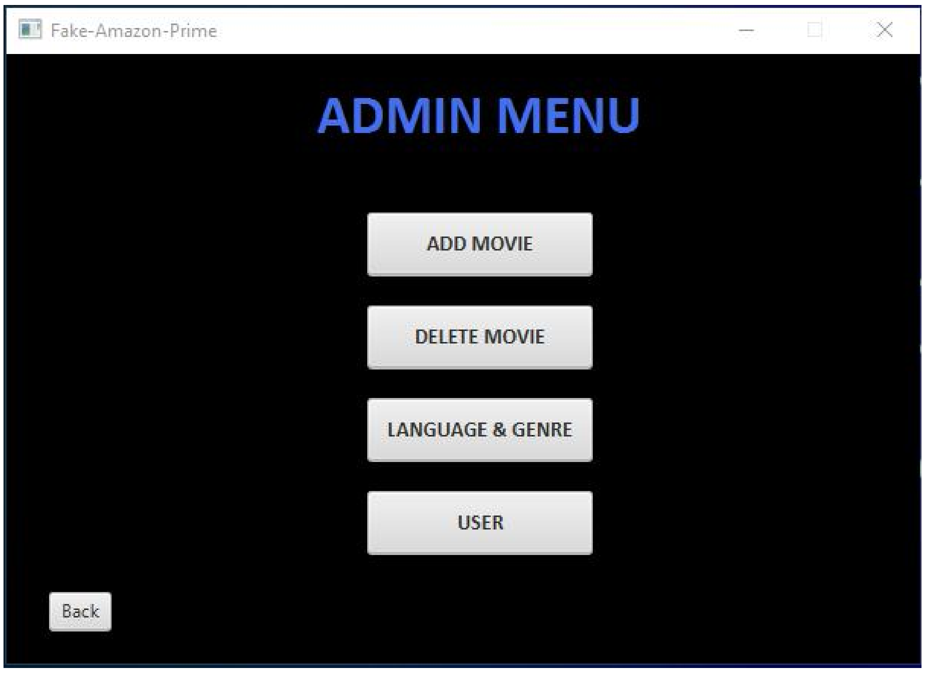
The poster for "Queen of Tears" is highlighted if the user goes with the mouse pad on it and below there is a short description.



On top of a mountain, a bunch of mountain goats are mining, when Tai Lung shows up and destroys the mountain....

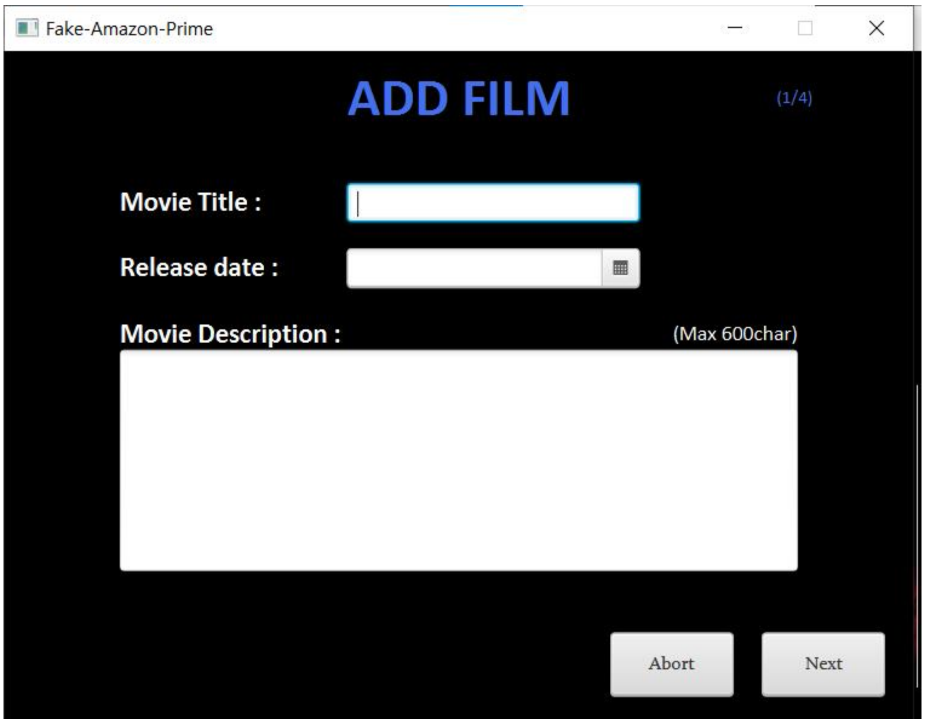
MainPage 4

If the user goes to the slide bar and click on the arrow, it will recommend 4 more Movies.



MainPage 5

If the user has administration rights, the administration button is clickable. On the administration interface page, we have designed a user-friendly layout for adding or deleting movies from the database. There is also the Language&Genre Button for the administrator to select which languages available and in which genre the movie is.


Fake-Amazon-Prime

ADD FILM (1/4)

Movie Title :

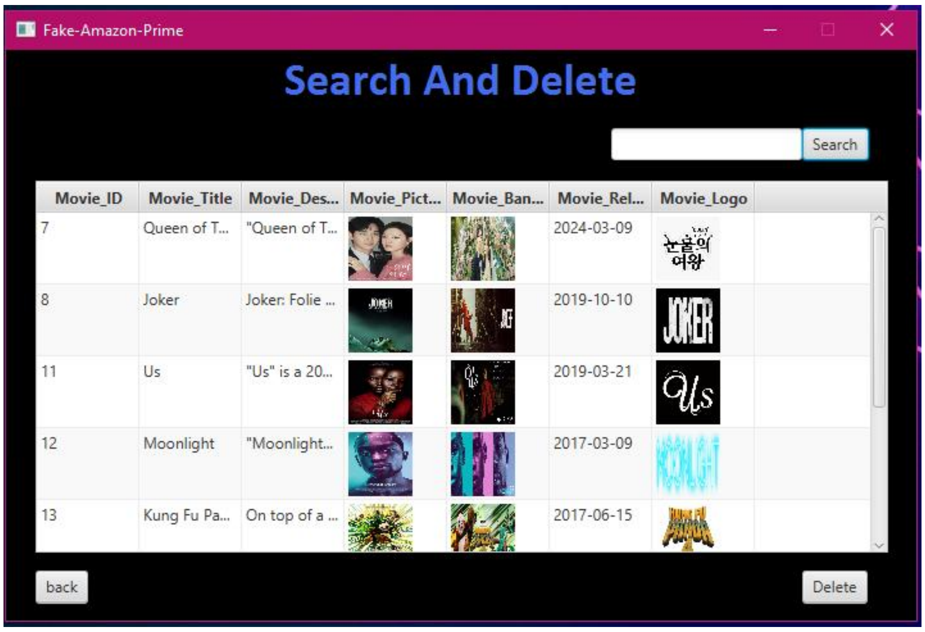
Release date :

Movie Description : (Max 600char)

Abort Next

MainPage 6

Here is the Movie Adding Page. Below the heading, there are three text fields required for adding movie details. After filling out the fields, the administrator can click „Abort“ to cancel the addition or „Next“ to add it to the database.


Fake-Amazon-Prime

Search And Delete

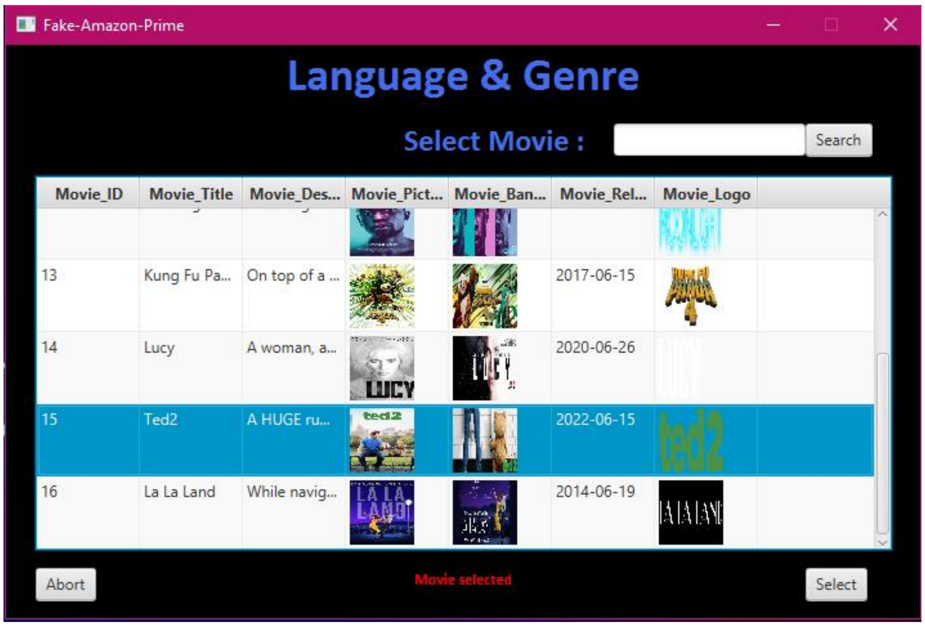
Search

Movie_ID	Movie_Title	Movie_Des...	Movie_Pict...	Movie_Ban...	Movie_Rel...	Movie_Logo
7	Queen of T...	"Queen of T...			2024-03-09	
8	Joker	Joker: Folie ...			2019-10-10	
11	Us	"Us" is a 20...			2019-03-21	
12	Moonlight	"Moonlight...			2017-03-09	
13	Kung Fu Pa...	On top of a ...			2017-06-15	

back Delete

MainPage 7

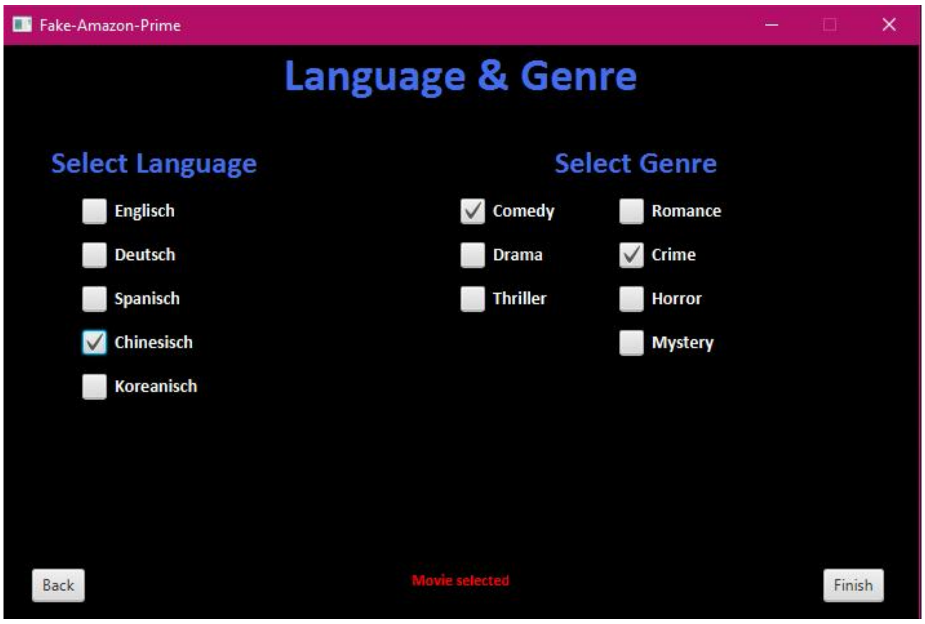
Here is the Movie Deleting Page. Below, there is a table of the movies in our database. The table shows the MovieID, MovieTitle, MovieDescription, MoviePicture, MovieBanner, MovieRelease and MovieLogo. The administrator can search and select which movie to delete.



Movie_ID	Movie_Title	Movie_Des...	Movie_Pict...	Movie_Ban...	Movie_Rel...	Movie_Logo
13	Kung Fu Pa...	On top of a ...			2017-06-15	
14	Lucy	A woman, a...			2020-06-26	
15	Ted2	A HUGE ru...			2022-06-15	
16	La La Land	While navig...			2014-06-19	

MainPage 8

Here is the Language&Genre Page. The Administrator can set which subtitle language is available for the movie and choose the genre of the movie.



Select Language

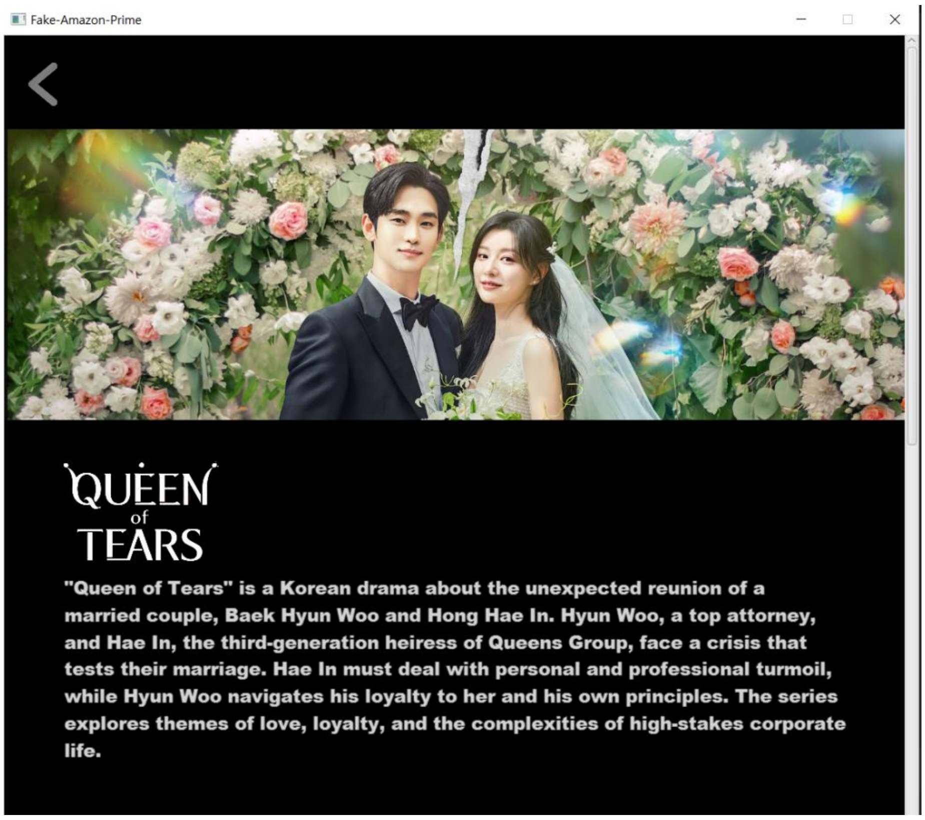
- Englisch
- Deutsch
- Spanisch
- [✓ Chinesisch]
- Koreanisch

Select Genre

- [✓ Comedy]
- Drama
- Thriller
- Romance
- [✓ Crime]
- Horror
- Mystery

MainPage 9

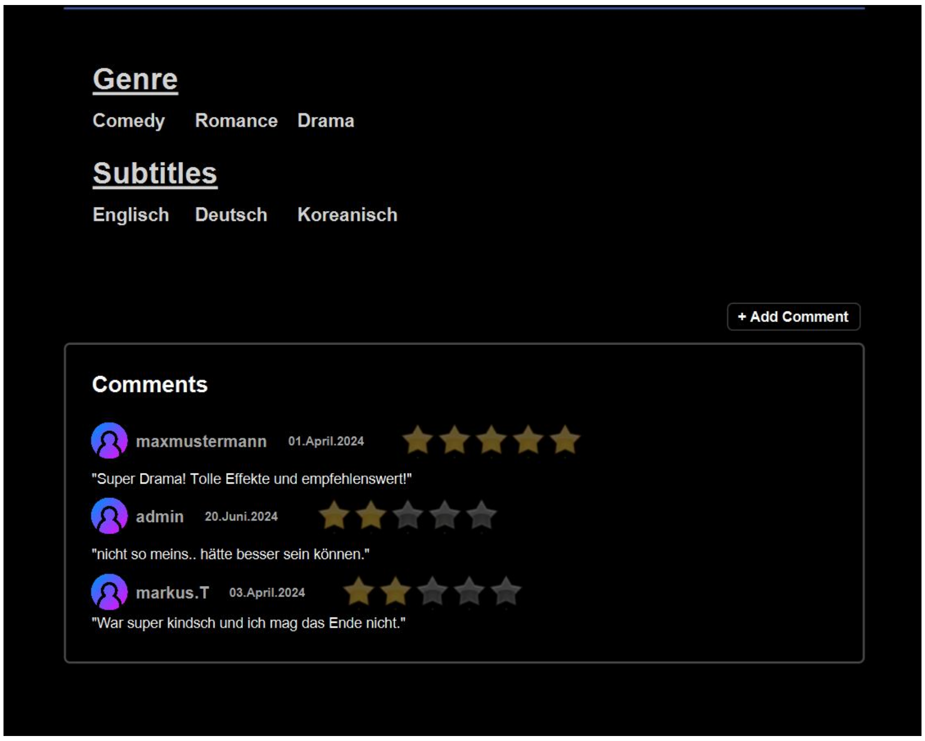
Administrator can set Language and Genre. It will be displayed in the description page.



"Queen of Tears" is a Korean drama about the unexpected reunion of a married couple, Baek Hyun Woo and Hong Hae In. Hyun Woo, a top attorney, and Hae In, the third-generation heiress of Queens Group, face a crisis that tests their marriage. Hae In must deal with personal and professional turmoil, while Hyun Woo navigates his loyalty to her and his own principles. The series explores themes of love, loyalty, and the complexities of high-stakes corporate life.

DescriptionPage 1

If the user clicks on a movie, it will go to the Description Page with a detailed description of the video and a large banner with the movie poster.



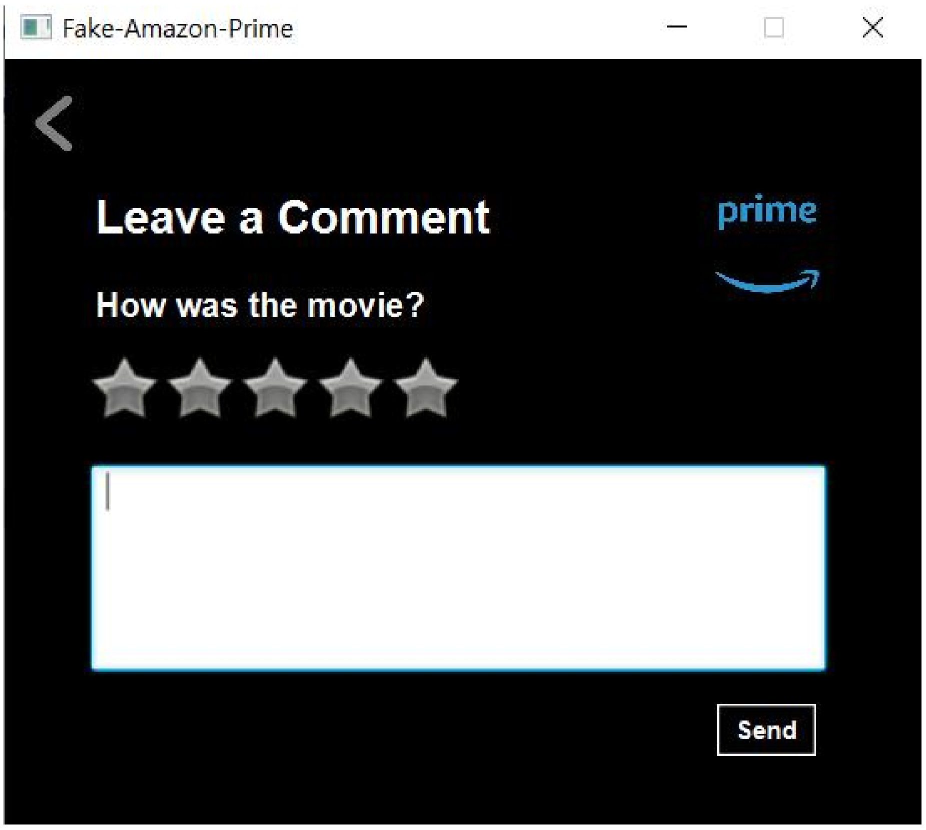
maxmustermann 01.April.2024 ★★★★★
"Super Drama! Tolle Effekte und empfehlenswert!"

admin 20.Juni.2024 ★★☆☆☆
"nicht so meins.. hätte besser sein können."

markus.T 03.April.2024 ★★☆☆☆
"War super kindsch und ich mag das Ende nicht."

DescriptionPage 2

Below the description, there is the genre specialization of the movie and information about the available subtitles. Below that, there is the comment section. The user can read comments from other users to get better insight into the movie. The user can also add their own comment.



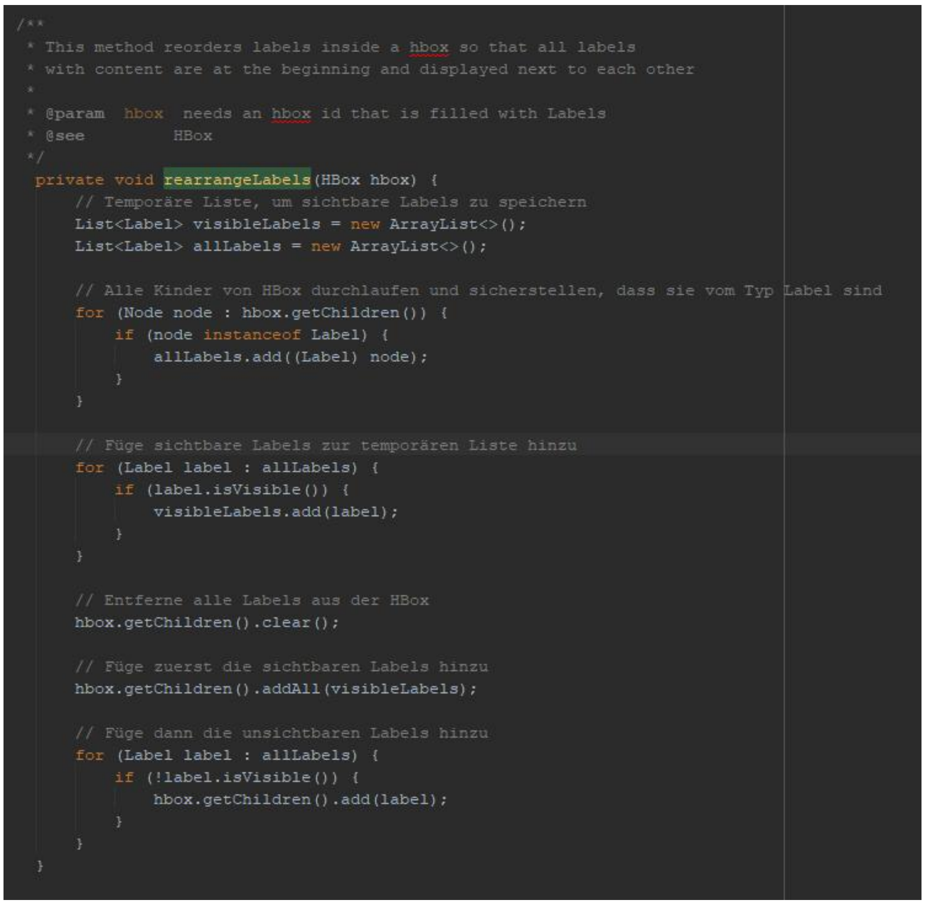
DescriptionPage 3

If the user clicks on the 'Add Comment' button, they can leave their opinion of the movie and rate the movie out of five stars. After sending, the comment will be stored in the Comment Database.

Code

Here are a few code snippets from our code that we programmed, which is not easy to understand. The remaining code is placed separately on an USB stick.

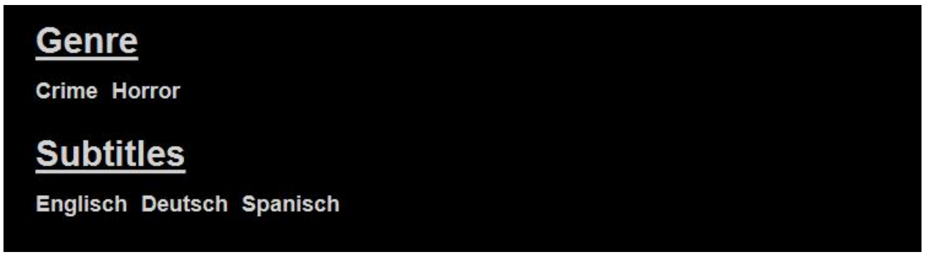
„rearrangeLabels“



```java
/**
 * This method reorders labels inside a hbox so that all labels
 * with content are at the beginning and displayed next to each other
 *
 * @param  hbox  needs an hbox id that is filled with Labels
 * @see         HBox
 */
 private void rearrangeLabels(HBox hbox) {
     // Temporäre Liste, um sichtbare Labels zu speichern
     List<Label> visibleLabels = new ArrayList<>();
     List<Label> allLabels = new ArrayList<>();

     // Alle Kinder von HBox durchlaufen und sicherstellen, dass sie vom Typ Label sind
     for (Node node : hbox.getChildren()) {
         if (node instanceof Label) {
             allLabels.add((Label) node);
         }
     }

     // Füge sichtbare Labels zur temporären Liste hinzu
     for (Label label : allLabels) {
         if (label.isVisible()) {
             visibleLabels.add(label);
         }
     }

     // Entferne alle Labels aus der HBox
     hbox.getChildren().clear();

     // Füge zuerst die sichtbaren Labels hinzu
     hbox.getChildren().addAll(visibleLabels);

     // Füge dann die unsichtbaren Labels hinzu
     for (Label label : allLabels) {
         if (!label.isVisible()) {
             hbox.getChildren().add(label);
         }
     }
 }
```

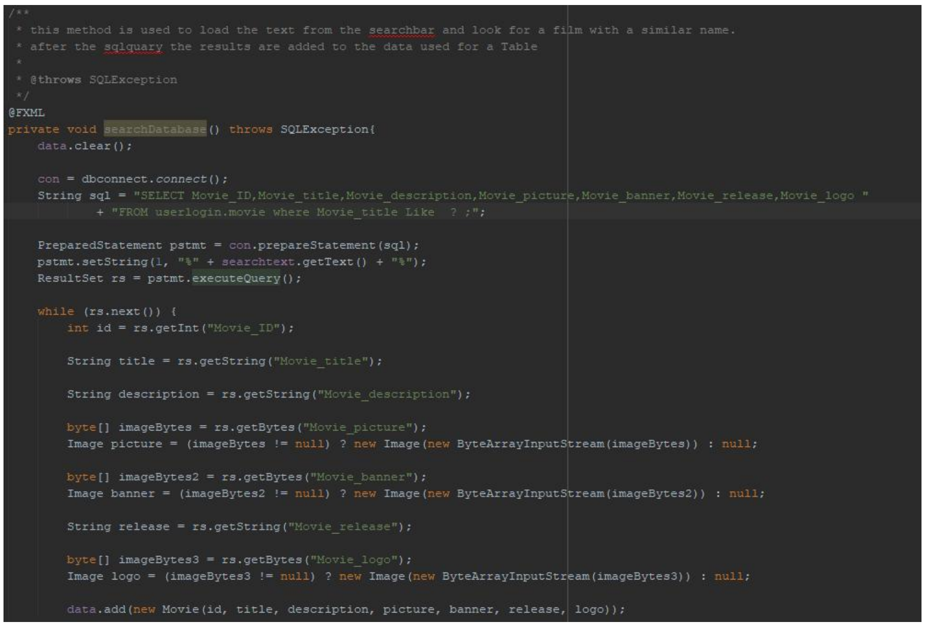
Code 1

This method „rearrangeLabels “ is to rearrange the labels on the displayside so that all visible labels are displayed like Movie Language and Genre.



Code 2

„searchDatabase“

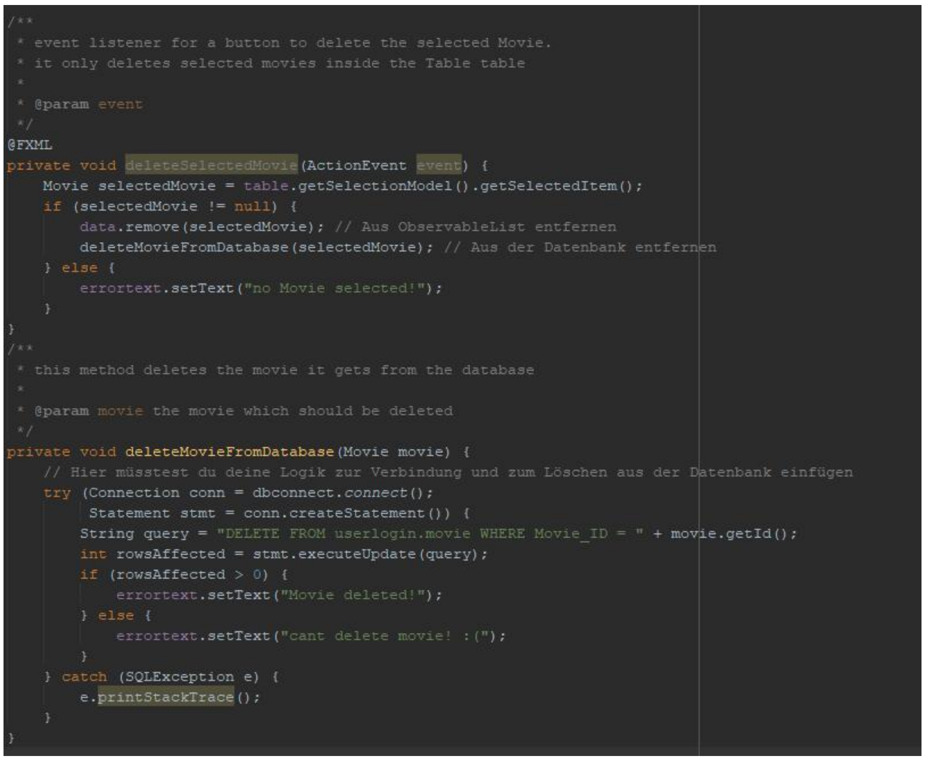


```java
/**
 * this method is used to load the text from the searchbar and look for a film with a similar name.
 * after the sqlquary the results are added to the data used for a Table
 *
 * @throws SQLException
 */
@FXML
private void searchDatabase() throws SQLException{
    data.clear();

    con = dbconnect.connect();
    String sql = "SELECT Movie_ID,Movie_title,Movie_description,Movie_picture,Movie_banner,Movie_release,Movie_logo "
            + "FROM userlogin.movie where Movie_title Like  ? ;";

    PreparedStatement pstmt = con.prepareStatement(sql);
    pstmt.setString(1, "%" + searchtext.getText() + "%");
    ResultSet rs = pstmt.executeQuery();

    while (rs.next()) {
        int id = rs.getInt("Movie_ID");

        String title = rs.getString("Movie_title");

        String description = rs.getString("Movie_description");

        byte[] imageBytes = rs.getBytes("Movie_picture");
        Image picture = (imageBytes != null) ? new Image(new ByteArrayInputStream(imageBytes)) : null;

        byte[] imageBytes2 = rs.getBytes("Movie_banner");
        Image banner = (imageBytes2 != null) ? new Image(new ByteArrayInputStream(imageBytes2)) : null;

        String release = rs.getString("Movie_release");

        byte[] imageBytes3 = rs.getBytes("Movie_logo");
        Image logo = (imageBytes3 != null) ? new Image(new ByteArrayInputStream(imageBytes3)) : null;

        data.add(new Movie(id, title, description, picture, banner, release, logo));
```

Code 3

This method „searchDatabase“ is used to look up for a movie in the database, after we executing a database query. The result will be displayed in the table.

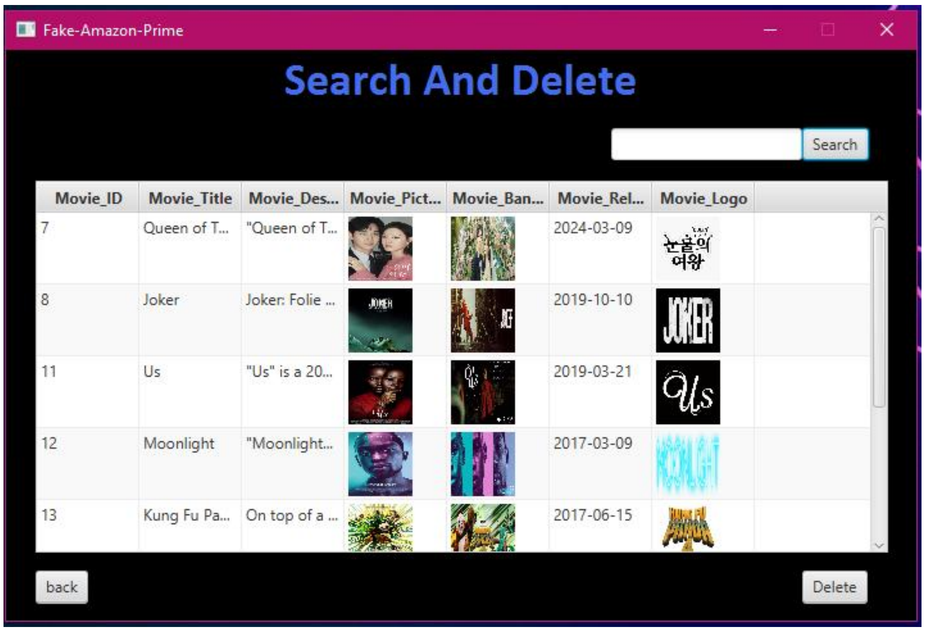
„deleteSelectedMovie“



```java
/**
 * event listener for a button to delete the selected Movie.
 * it only deletes selected movies inside the Table table
 *
 * @param event
 */
@FXML
private void deleteSelectedMovie(ActionEvent event) {
    Movie selectedMovie = table.getSelectionModel().getSelectedItem();
    if (selectedMovie != null) {
        data.remove(selectedMovie); // Aus ObservableList entfernen
        deleteMovieFromDatabase(selectedMovie); // Aus der Datenbank entfernen
    } else {
        errortext.setText("no Movie selected!");
    }
}
/**
 * this method deletes the movie it gets from the database
 *
 * @param movie the movie which should be deleted
 */
private void deleteMovieFromDatabase(Movie movie) {
    // Hier müsstest du deine Logik zur Verbindung und zum Löschen aus der Datenbank einfügen
    try (Connection conn = dbconnect.connect();
         Statement stmt = conn.createStatement()) {
        String query = "DELETE FROM userlogin.movie WHERE Movie_ID = " + movie.getId();
        int rowsAffected = stmt.executeUpdate(query);
        if (rowsAffected > 0) {
            errortext.setText("Movie deleted!");
        } else {
            errortext.setText("cant delete movie! :(");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Code 4

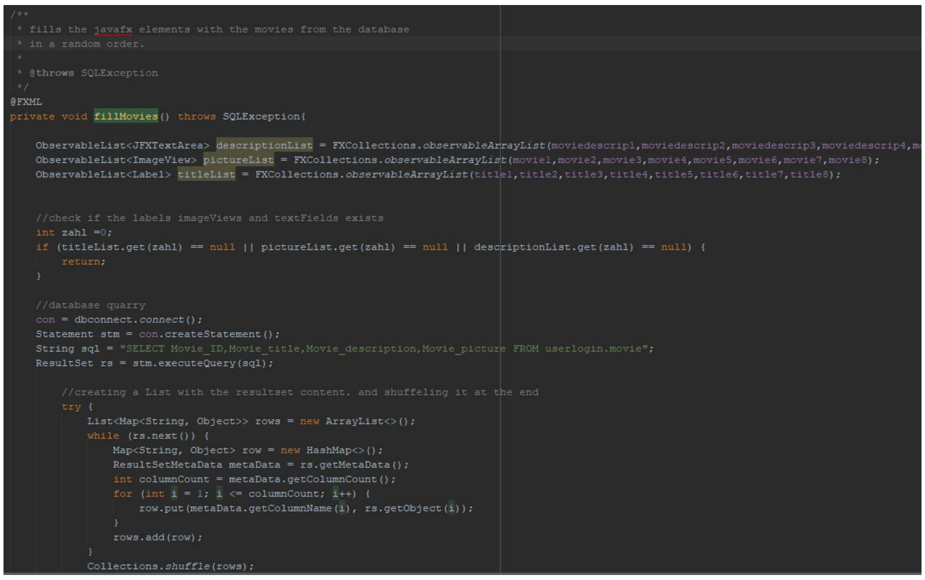
This method "deleteSelectedMovie" identifies and deletes the selected movie from the table and the database.



Movie_ID	Movie_Title	Movie_Des...	Movie_Pict...	Movie_Ban...	Movie_Rel...	Movie_Logo
7	Queen of T...	"Queen of T...			2024-03-09	
8	Joker	Joker: Folie ...			2019-10-10	
11	Us	"Us" is a 20...			2019-03-21	
12	Moonlight	"Moonlight...			2017-03-09	
13	Kung Fu Pa...	On top of a ...			2017-06-15	

Code 5

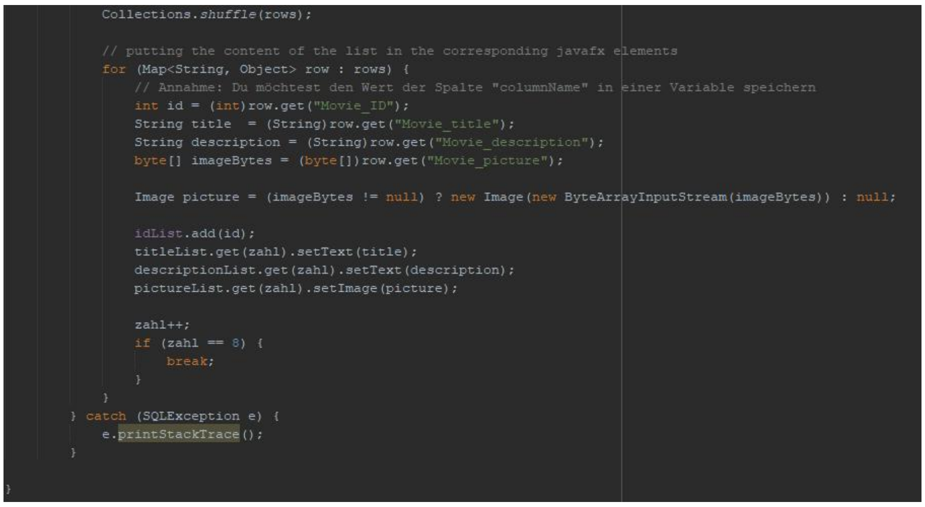
„fillMovies“



```java
/**
 * fills the javafx elements with the movies from the database
 * in a random order.
 *
 * @throws SQLException
 */
@FXML
private void fillMovies() throws SQLException{

    ObservableList<JFXTextArea> descriptionList = FXCollections.observableArrayList(moviedescrip1,moviedescrip2,moviedescrip3,moviedescrip4,mo
    ObservableList<ImageView> pictureList = FXCollections.observableArrayList(movie1,movie2,movie3,movie4,movie5,movie6,movie7,movie8);
    ObservableList<Label> titleList = FXCollections.observableArrayList(title1,title2,title3,title4,title5,title6,title7,title8);


    //check if the labels imageViews and textFields exists
    int zahl =0;
    if (titleList.get(zahl) == null || pictureList.get(zahl) == null || descriptionList.get(zahl) == null) {
        return;
    }

    //database quarry
    con = dbconnect.connect();
    Statement stm = con.createStatement();
    String sql = "SELECT Movie_ID,Movie_title,Movie_description,Movie_picture FROM userlogin.movie";
    ResultSet rs = stm.executeQuery(sql);

        //creating a List with the resultset content. and shuffeling it at the end
        try {
            List<Map<String, Object>> rows = new ArrayList<>();
            while (rs.next()) {
                Map<String, Object> row = new HashMap<>();
                ResultSetMetaData metaData = rs.getMetaData();
                int columnCount = metaData.getColumnCount();
                for (int i = 1; i <= columnCount; i++) {
                    row.put(metaData.getColumnName(i), rs.getObject(i));
                }
                rows.add(row);
            }
            Collections.shuffle(rows);
```

Code 6

This method „fillMovies“ is responsible for the movies being displayed in a random order.



```
            Collections.shuffle(rows);

            // putting the content of the list in the corresponding javafx elements
            for (Map<String, Object> row : rows) {
                // Annahme: Du möchtest den Wert der Spalte "columnName" in einer Variable speichern
                int id = (int)row.get("Movie_ID");
                String title  = (String)row.get("Movie_title");
                String description = (String)row.get("Movie_description");
                byte[] imageBytes = (byte[])row.get("Movie_picture");

                Image picture = (imageBytes != null) ? new Image(new ByteArrayInputStream(imageBytes)) : null;

                idList.add(id);
                titleList.get(zahl).setText(title);
                descriptionList.get(zahl).setText(description);
                pictureList.get(zahl).setImage(picture);

                zahl++;
                if (zahl == 8) {
                    break;
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

}
```

Code 7

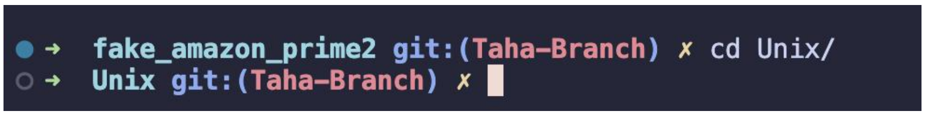
This code also collects movie information from a database query, stores the Movie IDs in an ObservableList, and assigns the resulting material to JavaFX elements for easy access to the movie description pages.

While working on the Deployment side of the Project we concentrated on these Points:

1. Simplicity of Deployment:
To streamline the deployment process and make it as straightforward as possible for users of varying technical expertise.

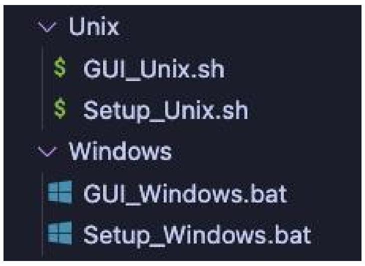
After Cloning the Repo the user will only need to run 2 Scripts.

1. The User needs to first navigate to either the Unix or Windows folder based on the OS he uses:



```
● ➜  fake_amazon_prime2 git:(Taha-Branch) ✗ cd Unix/
○ ➜  Unix git:(Taha-Branch) ✗ 
```

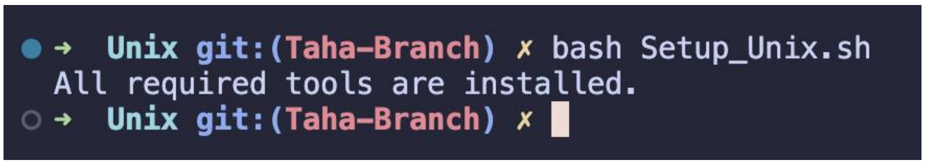
Deployment 1



```
⌄ Unix
    $ GUI_Unix.sh
    $ Setup_Unix.sh
⌄ Windows
    GUI_Windows.bat
    Setup_Windows.bat
```

Deployment 2

2. Launch the Setup Script:



```
● ➜  Unix git:(Taha-Branch) ✗ bash Setup_Unix.sh
  All required tools are installed.
○ ➜  Unix git:(Taha-Branch) ✗ █
```

Deployment 3

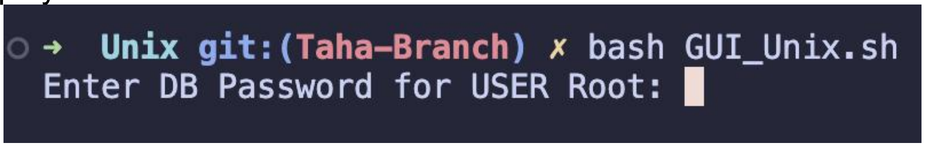
- The *Setup_Unix.sh* bash script will

1. First: make sure that *Docker* is installed in the machine where we want to deploy the project, if not installed the script will provide you with the link where you can install docker from and ask you to rerun the script.

2. Second: it will check if *Maven* is installed in the machine, if not it will download it using the *wget* command and then add it to the /etc/profile.d directory so that you can use the *mvn* command from the terminal.

- After Setting Up everything the script will tell you that all the requirements have been installed.

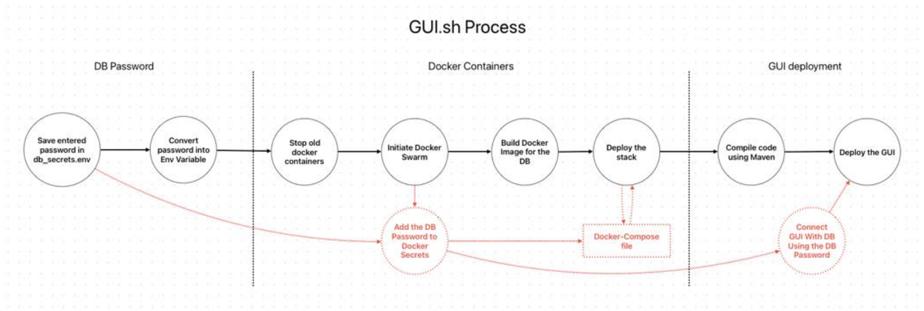
3. Launch the GUI Script and Give a password to be used in the DB Deployment:



```
○ ➜  Unix git:(Taha-Branch) ✗ bash GUI_Unix.sh
  Enter DB Password for USER Root: █
```

Deployment 4

This diagram Explains the process followed by the GUI bash script, which resumes the Deployment Process.


GUI.sh Process
DB Password
Docker Containers
GUI deployment
Save entered password in db_secrets.env
Convert password into Env Variable
Stop old docker containers
Initiate Docker Swarm
Build Docker Image for the DB
Deploy the stack
Compile code using Maven
Deploy the GUI
Add the DB Password to Docker Secrets
Docker-Compose file
Connect GUI With DB Using the DB Password

Deployment 5

- The GUI_Unix.sh bash script will

1. *Move* to the *Databases* directory where the Dockerfile-db is situated

2. Generate an AES Key to be used for the encryption of password. And then it will save it in a temporary .txt file.

3. *Stop and Remove* all containers that are running with the same name as the one we are trying to make.

4. It will build a new docker image named userlogin-db

5. It will run it on port 3308 so that the GUI can get access to it Using the docker-compose.yml file.

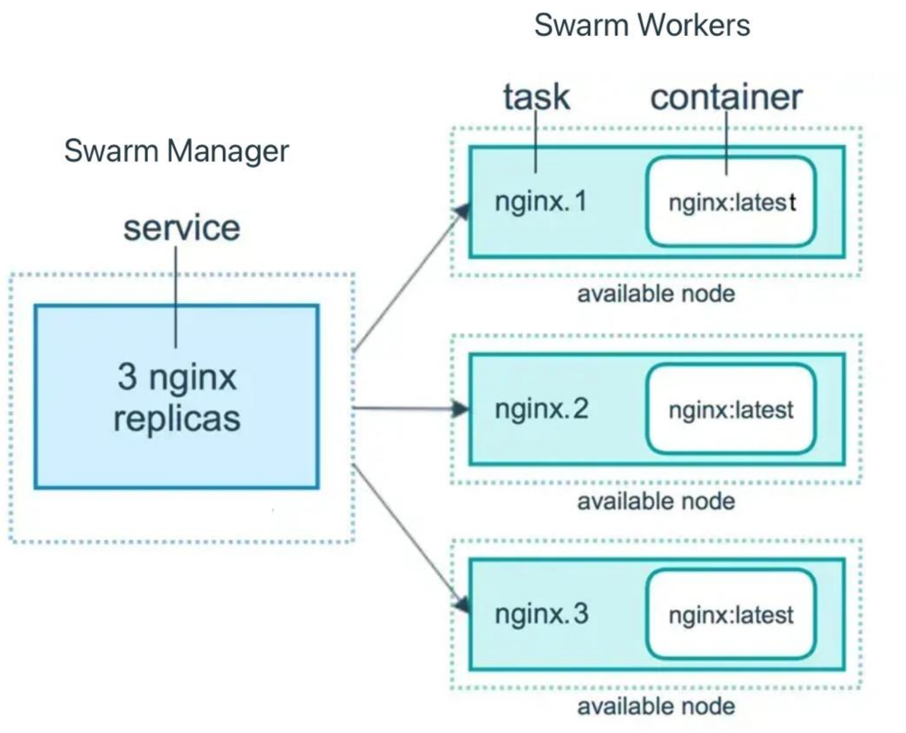
6. It will now go back to the main Directory (/login) where the pom.xml is situated

7.It will run a command that will clean old packages and create a new jar file. And then it will run the javafx application using the

```
mvn javafx:run
```

command

2. Reliability:
To maintain consistent performance and uptime, ensuring the system works correctly and efficiently over time.
We use *Docker Swarm* instead of the normal Docker containers because it will keep respawning every time a container is down. Which will give us a consistent Uptime.



Reliability 1

3. Scalability:
To ensure the system can handle an increasing amount of work, or its potential to accommodate growth.
By using docker containers we were able to insure these key aspects:

1. Lightweight and Efficient:

- Docker containers share the host OS kernel, leading to minimal overhead compared to traditional local Databases. This efficiency allows more containers to run on a single host.

2. Resource Allocation:

- Docker allows setting resource limits (CPU, memory) on containers, enabling efficient utilization of system resources and preventing any single container from exhausting the host's resources.

3. Infrastructure as Code:

- Docker integrates well with CI/CD pipelines, enabling automated deployment and scaling processes. This automation reduces manual intervention and speeds up the scaling process.

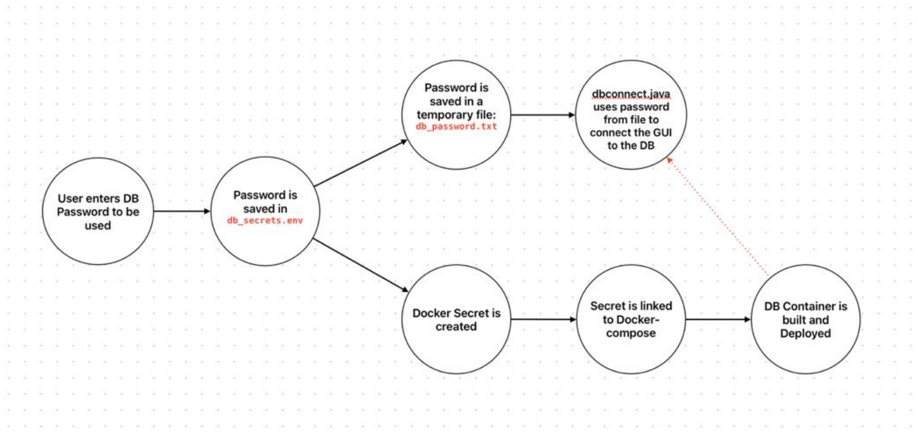
4. Security:
To protect sensitive data and ensure the application is secure from various threats.

1. Penetration test:

- A white box penetration test was conducted to identify and address all vulnerable points in the code.

    a. Secured the app against SQL Injections. By using the *PreparedStatement* function instead of the normally used *Statement* function, it allows us to use parameterized queries, which helps prevent SQL injections by treating user inputs as data not as executable code.
    b. The setString method is used to safely insert the user inputs (user and password) into the query. The ? placeholders in the SQL query are replaced with the actual values of this.getUser() and this.getPassword()
    c. The *executeQuery* method is called on the *PreparedStatement* object to execute the query. The result is stored in a ResultSet.
    d. And finaly we handle the resultes using an if else statement that checks if a record matching the criteria was found, using the rs.next() method.

2. Implemented Docker Secrets:


User enters DB Password to be used
Password is saved in db_secrets.env
Password is saved in a temporary file: db_password.txt
dbconnect.java uses password from file to connect the GUI to the DB
Docker Secret is created
Secret is linked to Docker-compose
DB Container is built and Deployed

Docker Secrets 1

3. *Encrypted all Passwords in the Database.*

- Description of Used Algorithm:

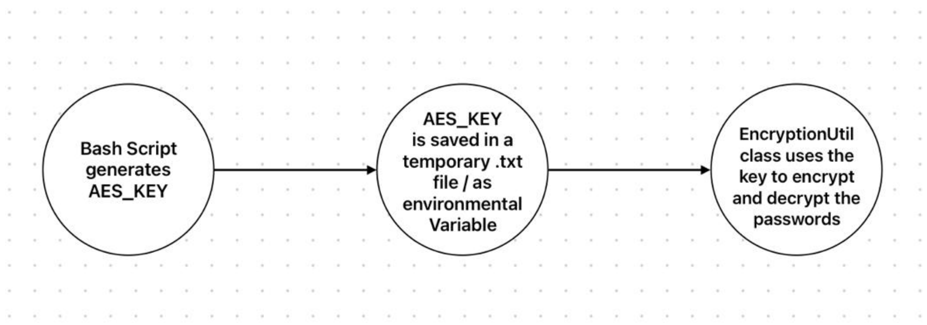
This project implements a user login and registration system with enhanced security measures, including protection against SQL injection and encryption of sensitive information such as passwords.

- Encryption Details:

The primary purpose of encryption in this project is to ensure that sensitive information, specifically user passwords, is not stored in clear text in the database. This enhances the security of the system by protecting user data from being compromised.

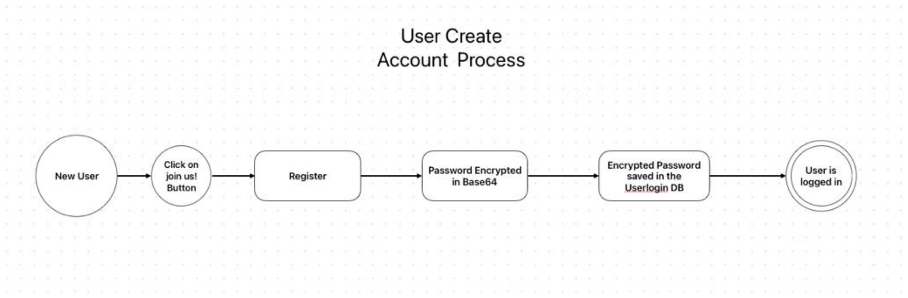
1. Encryption Algorithm: In this project we use the Advanced Encryption Standard (AES) algorithm to encrypt and decrypt data.

2. Key Management: A secret key is used to encrypt and decrypt the data. This key must be securely stored and managed. The Key is being randomly generated by the GUI script and stores it in a temporary file that gets deleted at the end of the process.

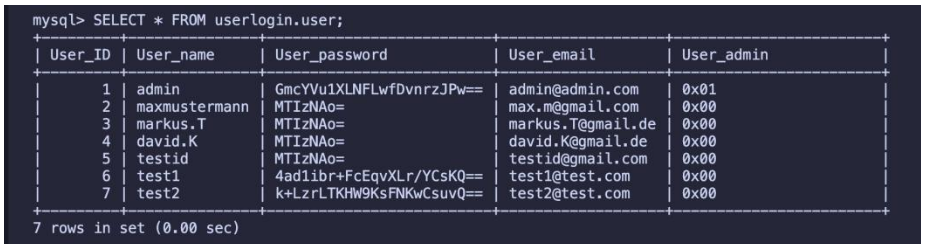

Bash Script generates AES_KEY
AES_KEY is saved in a temporary .txt file / as environmental Variable
EncryptionUtil class uses the key to encrypt and decrypt the passwords

Docker Secrets 1

4. *Encryption Process*: When a user registers, their password is encrypted before being saved to the database.


User Create Account Process
New User
Click on join us! Button
Register
Password Encrypted in Base64
Encrypted Password saved in the Userlogin DB
User is logged in

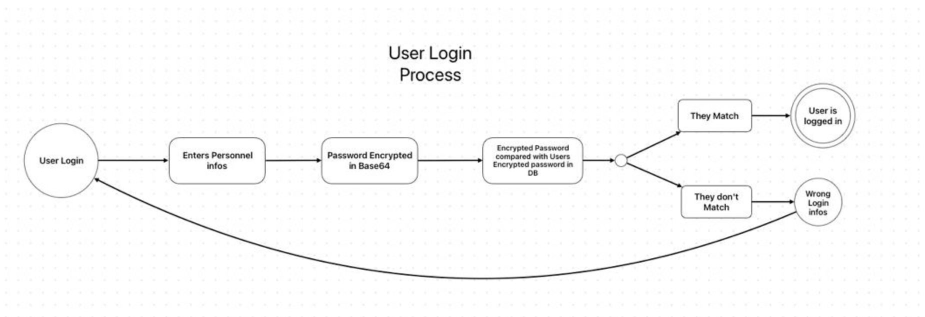
Encryption 1



```
mysql> SELECT * FROM userlogin.user;
+---------+--------------+---------------------------+------------------+----------------------+
| User_ID | User_name    | User_password             | User_email       | User_admin           |
+---------+--------------+---------------------------+------------------+----------------------+
|       1 | admin        | GmcYVu1XLNFLwfDvnrzJPw== | admin@admin.com  | 0x01                 |
|       2 | maxmustermann | MTIzNAo=                 | max.m@gmail.com  | 0x00                 |
|       3 | markus.T     | MTIzNAo=                  | markus.T@gmail.de | 0x00                |
|       4 | david.K      | MTIzNAo=                  | david.K@gmail.de | 0x00                 |
|       5 | testid       | MTIzNAo=                  | testid@gmail.com | 0x00                 |
|       6 | test1        | 4ad1ibr+FcEqvXLr/YCsKQ== | test1@test.com   | 0x00                 |
|       7 | test2        | k+LzrLTKHW9KsFNKwCsuvQ== | test2@test.com   | 0x00                 |
+---------+--------------+---------------------------+------------------+----------------------+
7 rows in set (0.00 sec)
```

Encryption 2

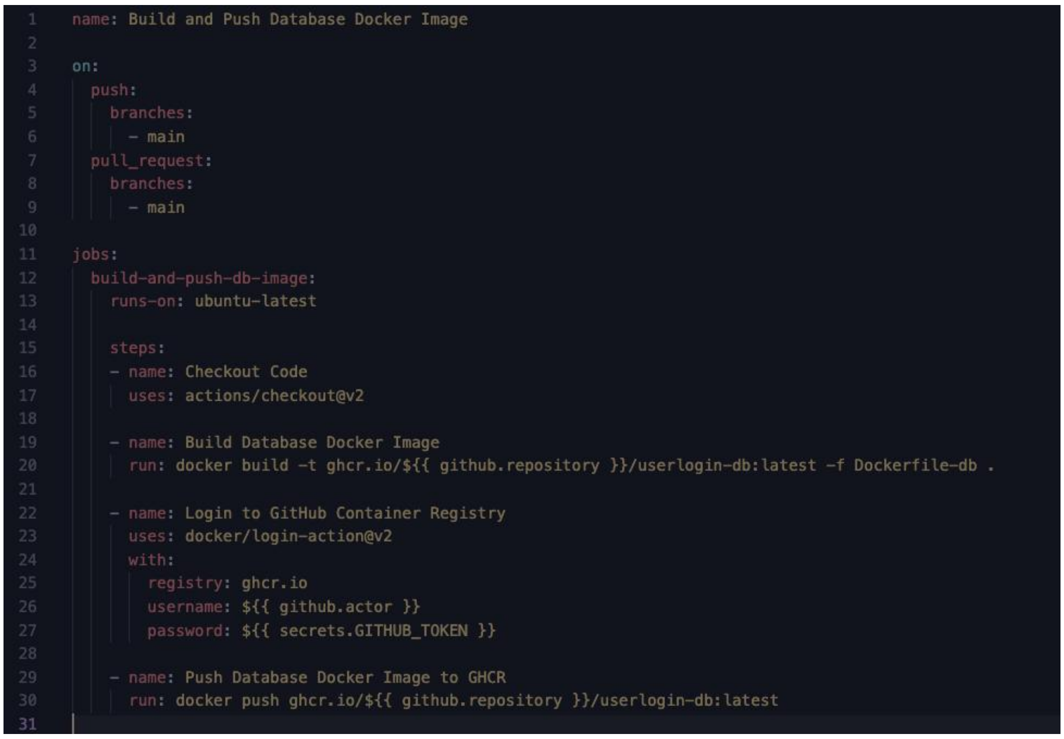
5. *Decryption Process*: When a user logs in, the entered password is encrypted and compared with the stored encrypted password to verify the user's identity.



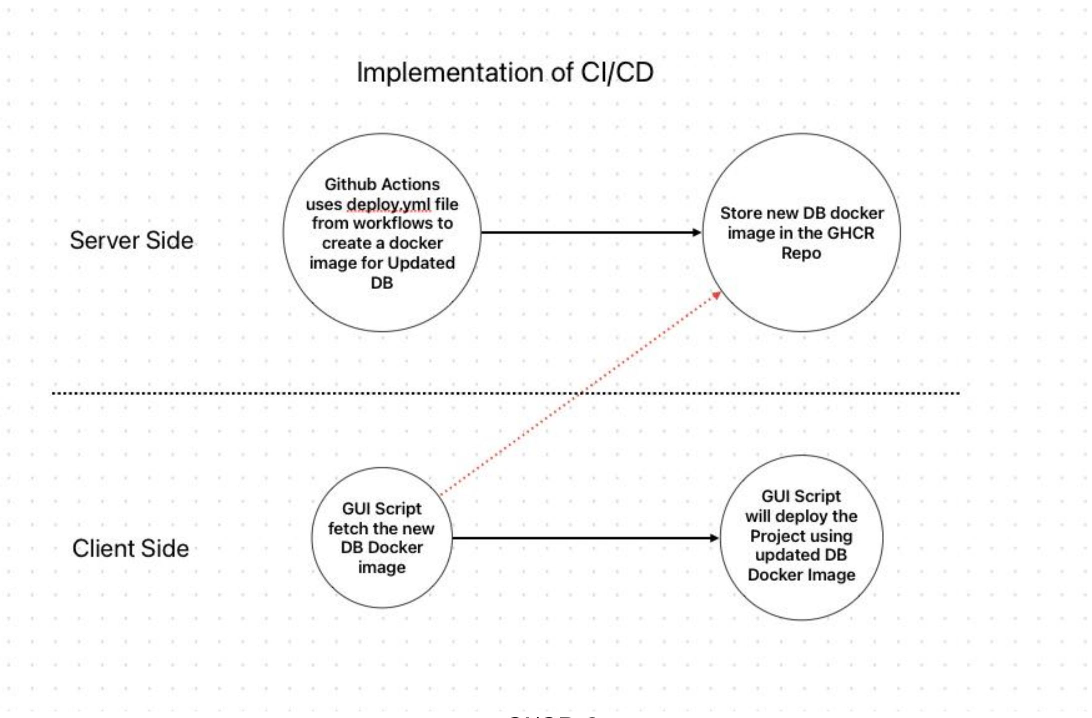
Encryption 3

5. CI/CD (Still not fully working):

- Implemented continuous integration and continuous deployment using GitHub Actions.
- Automated the process of uploading updated Docker images to the GitHub Container Registry (GHCR).


```
name: Build and Push Database Docker Image

on:
  push:
    branches:
      - main
  pull_request:
    branches:
      - main

jobs:
  build-and-push-db-image:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout Code
      uses: actions/checkout@v2

    - name: Build Database Docker Image
      run: docker build -t ghcr.io/${{ github.repository }}/userlogin-db:latest -f Dockerfile-db .

    - name: Login to GitHub Container Registry
      uses: docker/login-action@v2
      with:
        registry: ghcr.io
        username: ${{ github.actor }}
        password: ${{ secrets.GITHUB_TOKEN }}

    - name: Push Database Docker Image to GHCR
      run: docker push ghcr.io/${{ github.repository }}/userlogin-db:latest
```

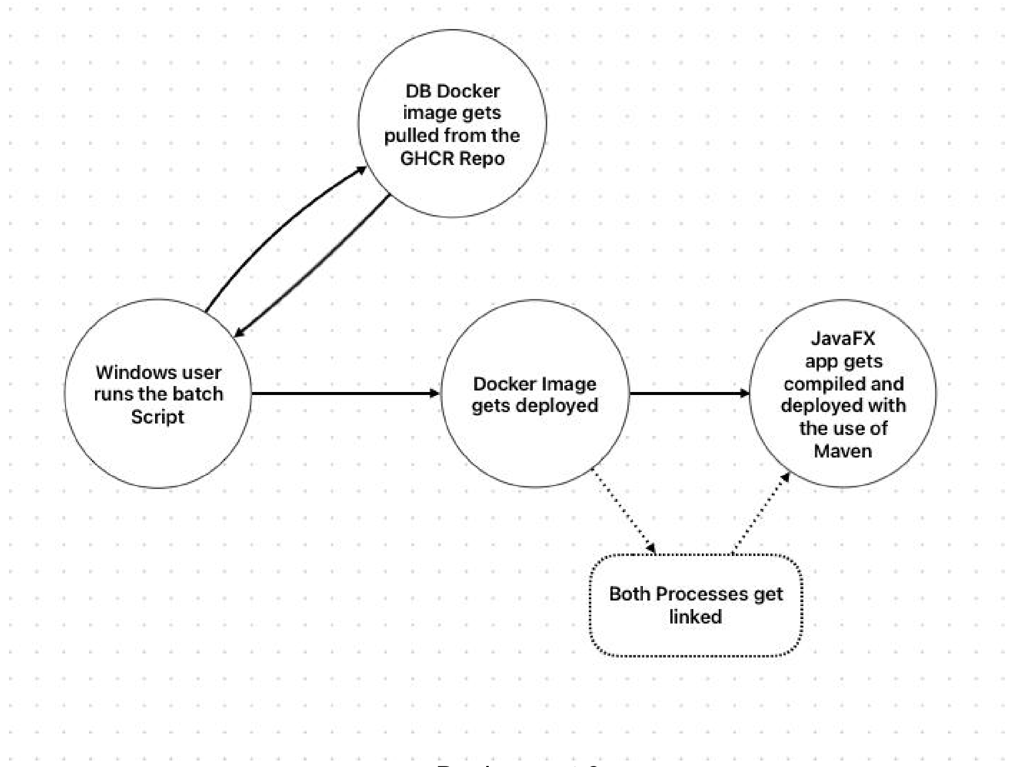
CI/CD 1


Implementation of CI/CD
Server Side
Github Actions uses deploy.yml file from workflows to create a docker image for Updated DB
Store new DB docker image in the GHCR Repo
Client Side
GUI Script fetch the new DB Docker image
GUI Script will deploy the Project using updated DB Docker Image

CI/CD 2

6. A Solution for the Windows batch Problem:

While trying to figure out a solution for the Windows deployment script without having a windows machine to work with, I decided to implement some type of CI/CD, where I will manually upload the working Container image into a GHCR, and then make the Windows batch script fetch and deploy the already made docker image and link the GUI to it.


DB Docker image gets pulled from the GHCR Repo
Windows user runs the batch Script
Docker Image gets deployed
JavaFX app gets compiled and deployed with the use of Maven
Both Processes get linked

Deployment 6

Conclusion

In summary, the programming and development of a convincing and user friendly movie Streaming Service is important for a good entertainment experience. This project successfully addressed this requirement of this modul „Programming Exercise“ by focusing on a simple user interaction and improving content management and guaranteeing a easy to understand management application for administrators.

Fake Amazon uses a secure login system to ensure that only authorized user may access to their accounts. For full security the password will be encrypted, so that the administration do not even see the password to guarantee integrity.

The step-by-step process for adding new movie begins on the Admin page and goes to the dedicated Add Movie page, provides administrators with an easier way to add new movies in three steps. First add movie Title, second add Description and last add movie picture, banner and a logo. A user comment rating option provides a community where other users can get insight into a film, even if they have not seen the movie.

In summary, this project shows the need of a well-designed movie streaming service which is simple, organized, and user-friendly. By implementing such a system, streaming platforms can promote efficient and effective content management, resulting in a lively and exciting entertainment environment.

Sources

- MySQL for the database
- SceneBuilder
- Netbeans
- JavaFX
- Pictures from Netflix and Google

Image directory

Das Inhaltsverzeichnis ist leer, da du keine Absatzstile für das Inhaltsverzeichnis festgelegt hast.