

**King Abdulaziz University**

**Department of Computer Science**

**Faculty of Computing and Information Technology**

**CPCS403**

**Uniplay**

“Track and follow all your favorite shows and movies in one place.”

Final Project

**Submitted by: Raiyan Tarek Subedar**

**ID: 2136108**

Table of Contents

[Problem Definition 3](#_Toc182588999)

[uniplay-production.up.railway.app 3](#_Toc182589000)

[Recommended Solution 3](#_Toc182589001)

[Benefits of the Provided Solution 3](#_Toc182589002)

[Storyboard 4](#_Toc182589003)

[Landing Page 4](#_Toc182589004)

[Home Page 5](#_Toc182589005)

[About Page 6](#_Toc182589006)

[Contact Page 7](#_Toc182589007)

[Login Page 8](#_Toc182589008)

[Register Page 9](#_Toc182589009)

[MovieShow Page 10](#_Toc182589010)

[Service Page 11](#_Toc182589011)

[Sitemap 12](#_Toc182589012)

[PHP Pages 13](#_Toc182589013)

[Database Tables 15](#_Toc182589014)

[Security Measures 20](#_Toc182589015)

[Client-Side and Server-Side Validation 20](#_Toc182589016)

# Problem Definition

Users today face the challenge of managing multiple streaming platforms, each offering different content. Keeping track of favourite shows and movies across platforms like Netflix, Hulu, and Amazon Prime can be difficult. Without a unified system to track release dates, new episodes, and recommendations, users often miss important updates.  
**Uniplay** solves this problem by providing a single platform where users can follow and track their favourite shows and movies, regardless of the streaming service.

# [uniplay-production.up.railway.app](https://uniplay-production.up.railway.app/)

# Recommended Solution

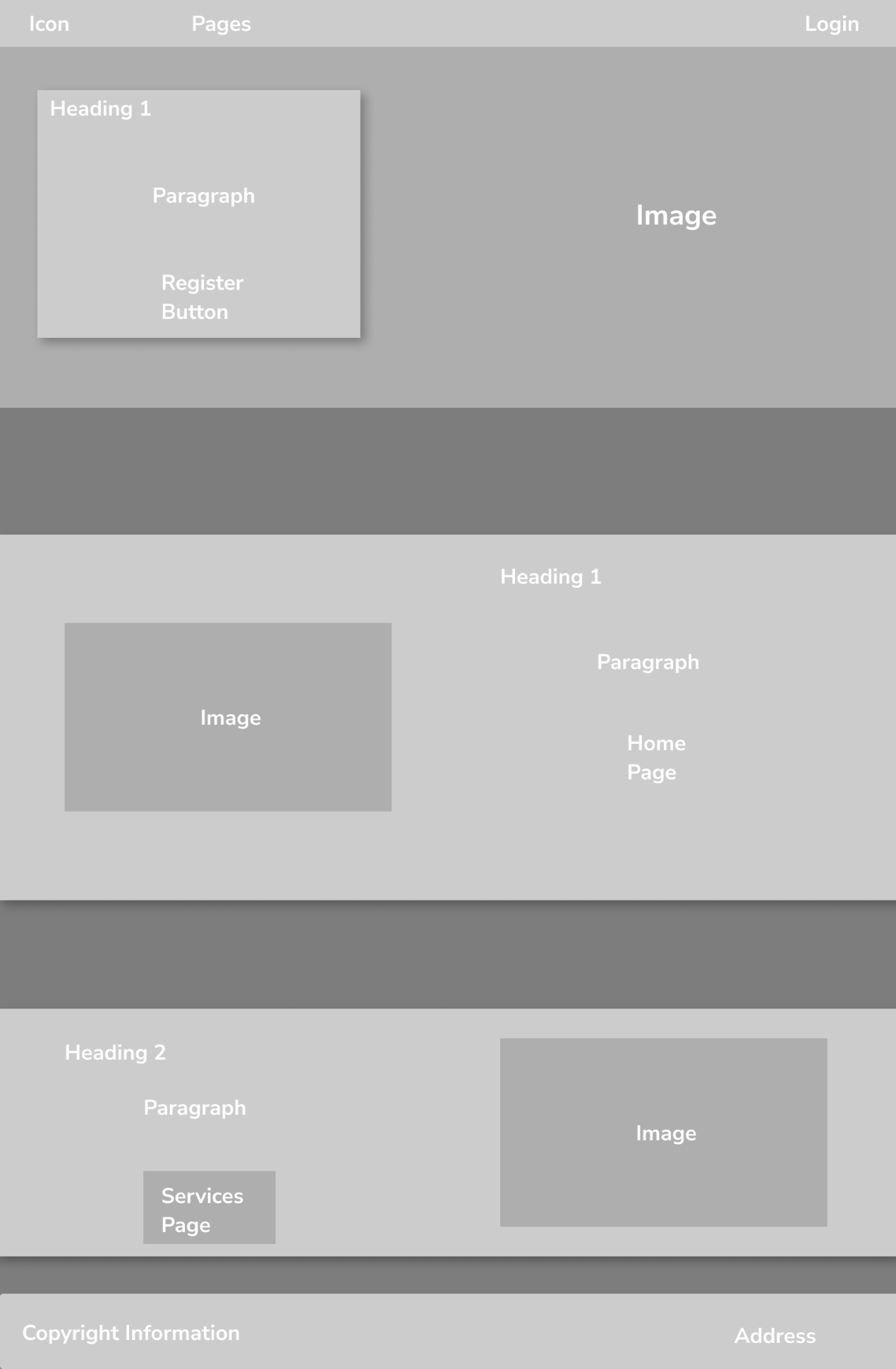
The solution is Uniplay, a platform designed to centralize content tracking from various streaming services. Inspired by IMDb, Uniplay allows users to search for, follow, and receive updates about shows and movies in one place.  
The platform aggregates key information like release dates, episode reminders, and recommendations. While Uniplay doesn’t provide direct streaming, it links users to the relevant platform for watching content, creating a central hub for managing their media preferences.

# Benefits of the Provided Solution

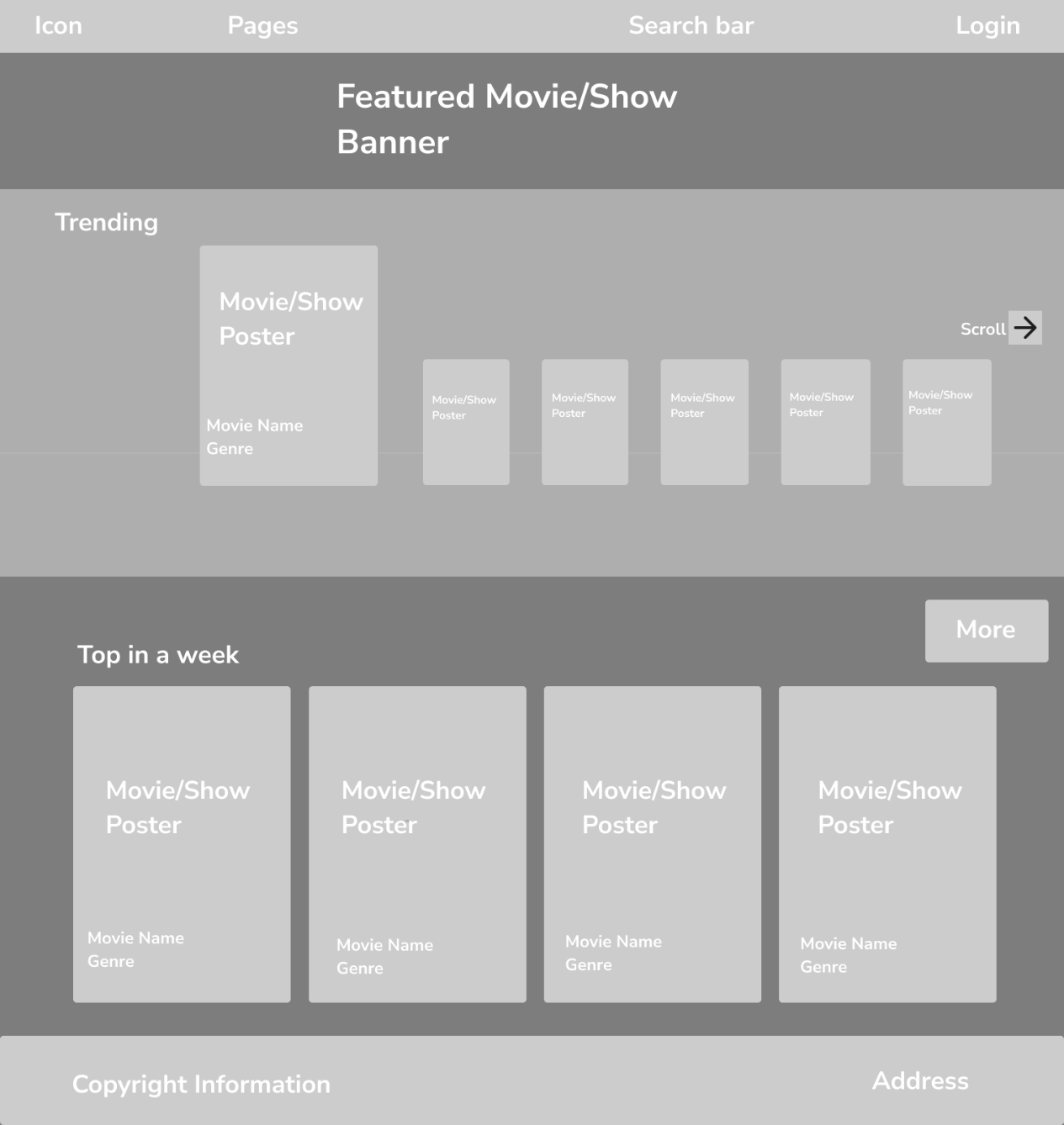
Uniplay simplifies the user experience by bringing all the information into one place. Users no longer need to visit multiple services to keep up with their favorite shows and movies.  
This saves time and reduces the chance of missing important content. Additionally, users can discover new recommendations based on their interests, improving content management and enhancing the overall viewing experience.

# Storyboard

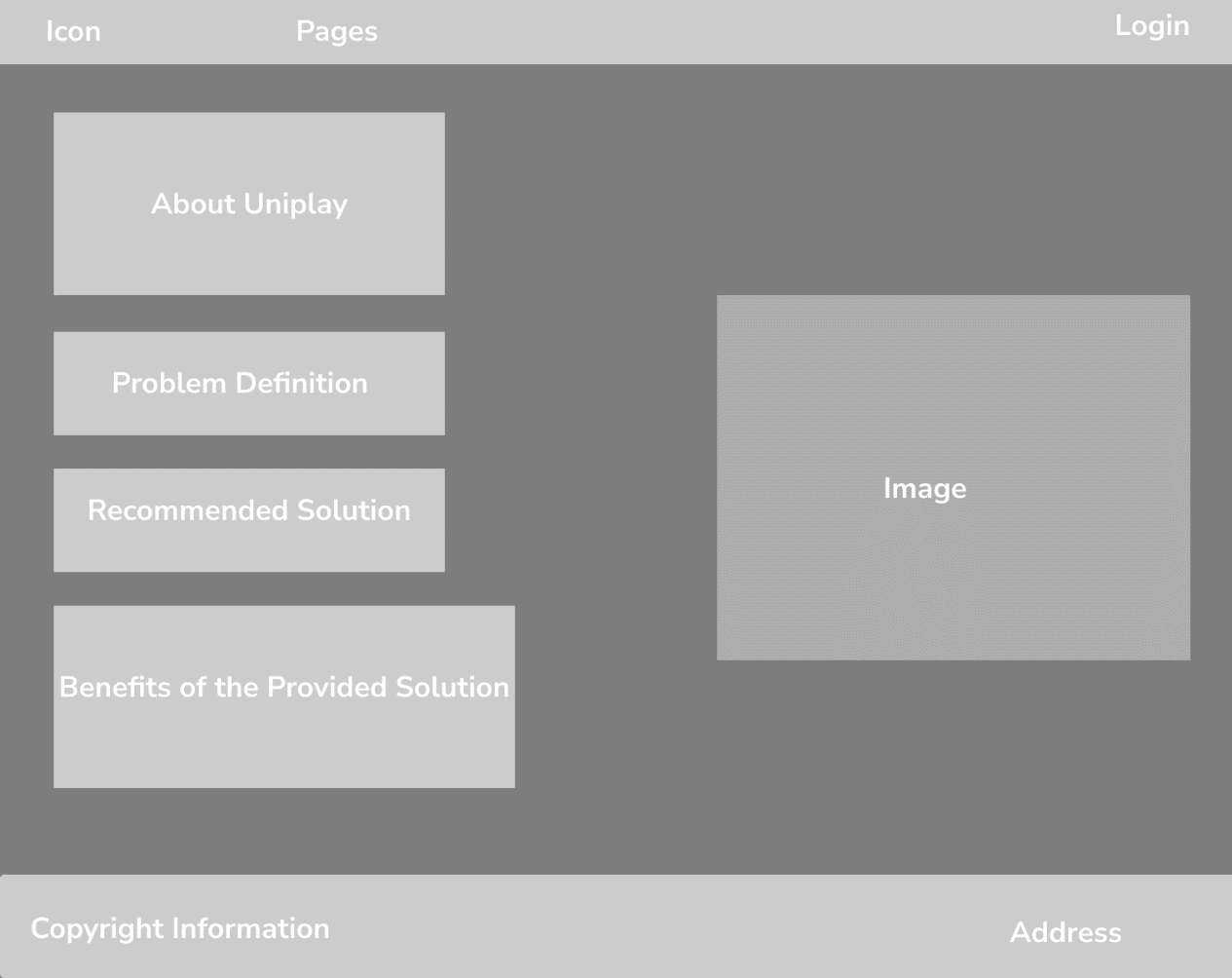
## Landing Page



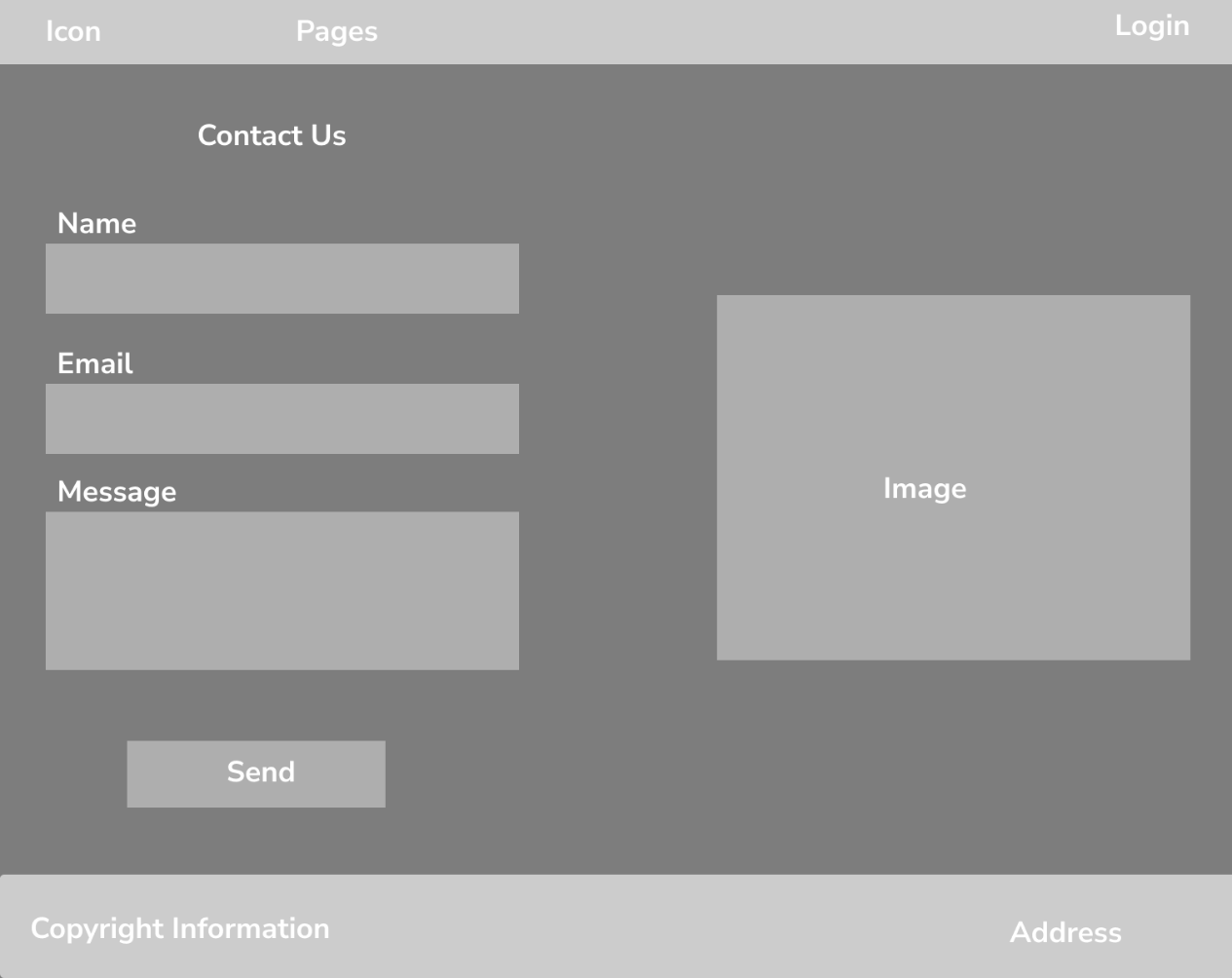
## Home Page



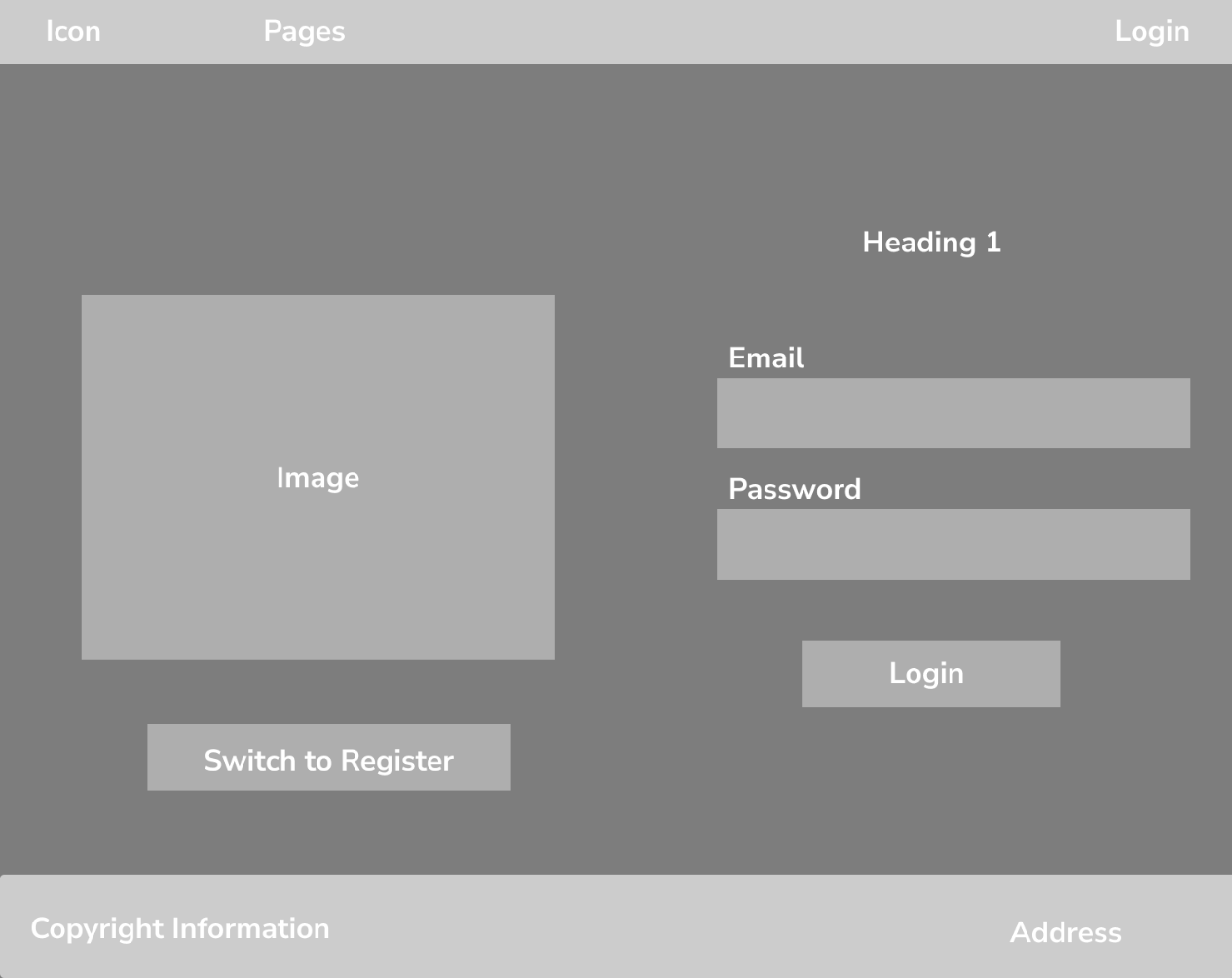
## About Page



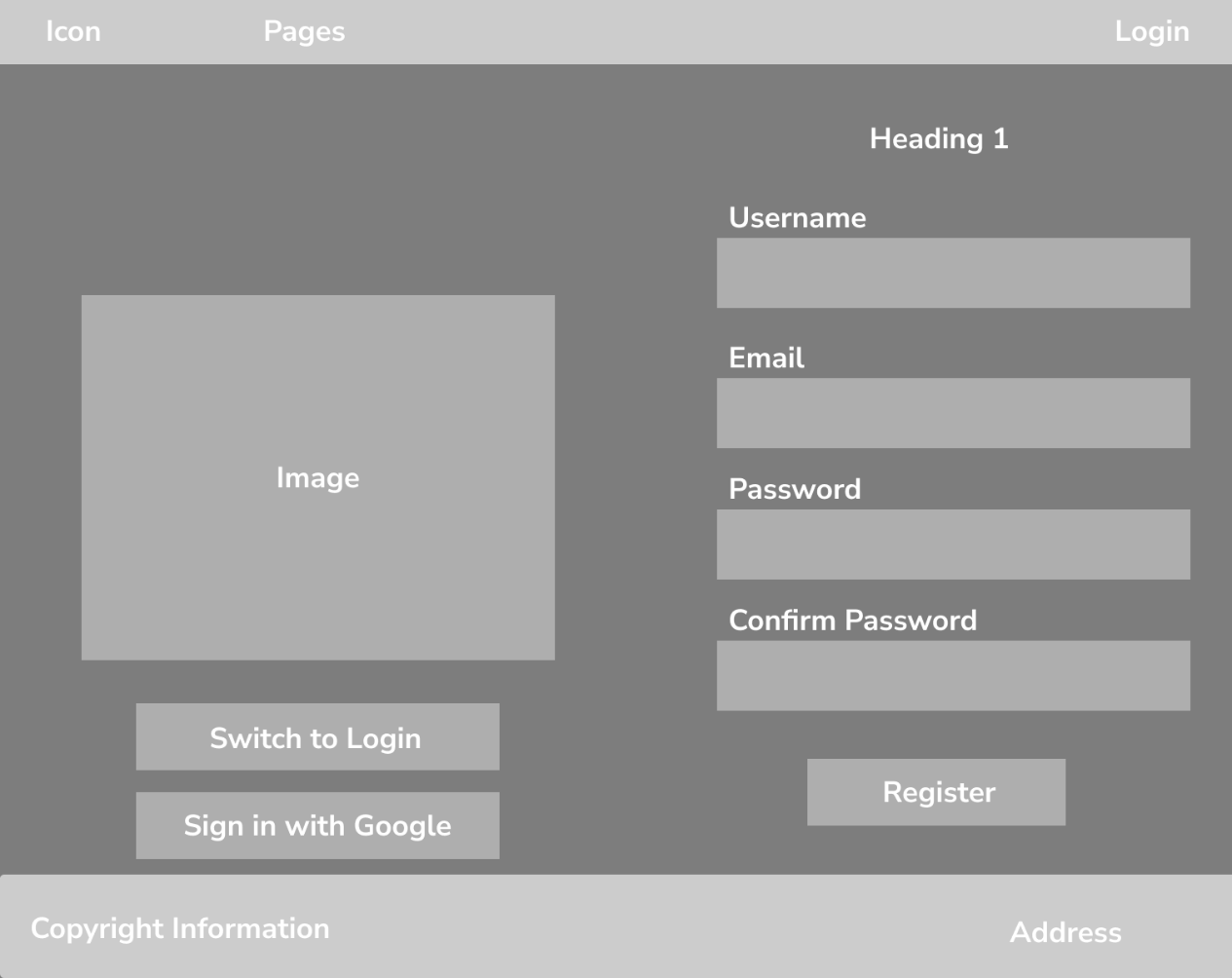
## Contact Page



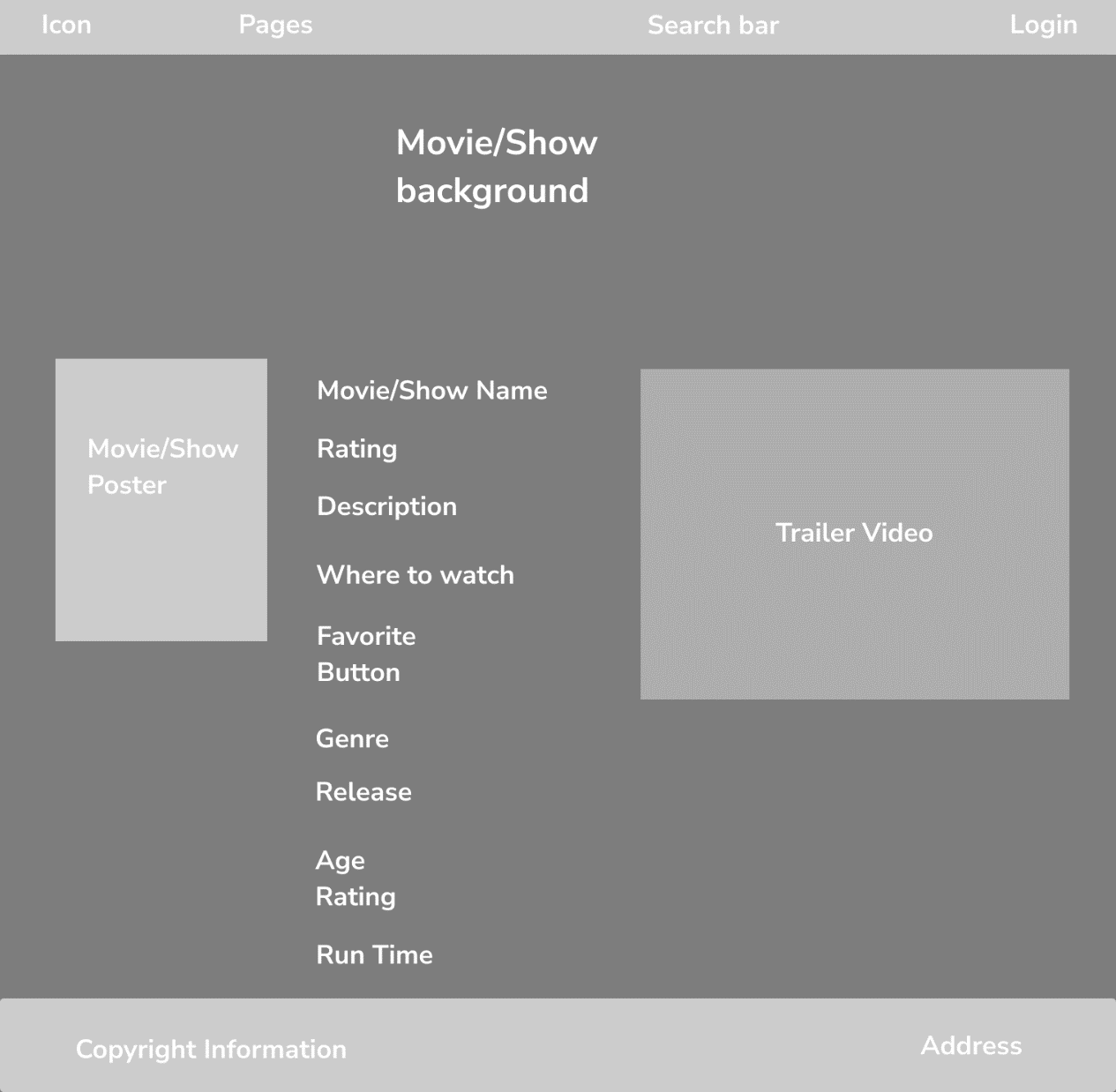
## Login Page



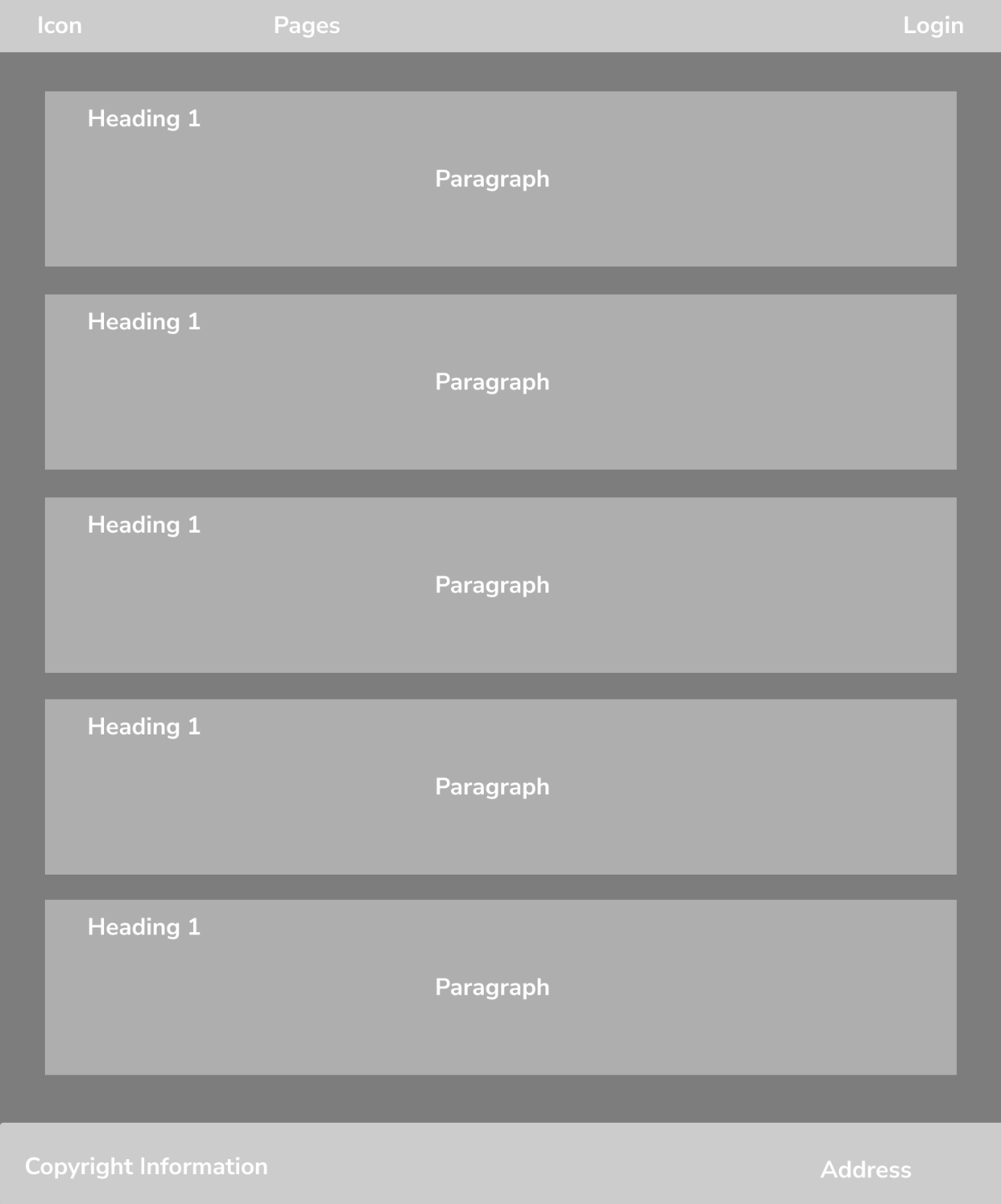
## Register Page



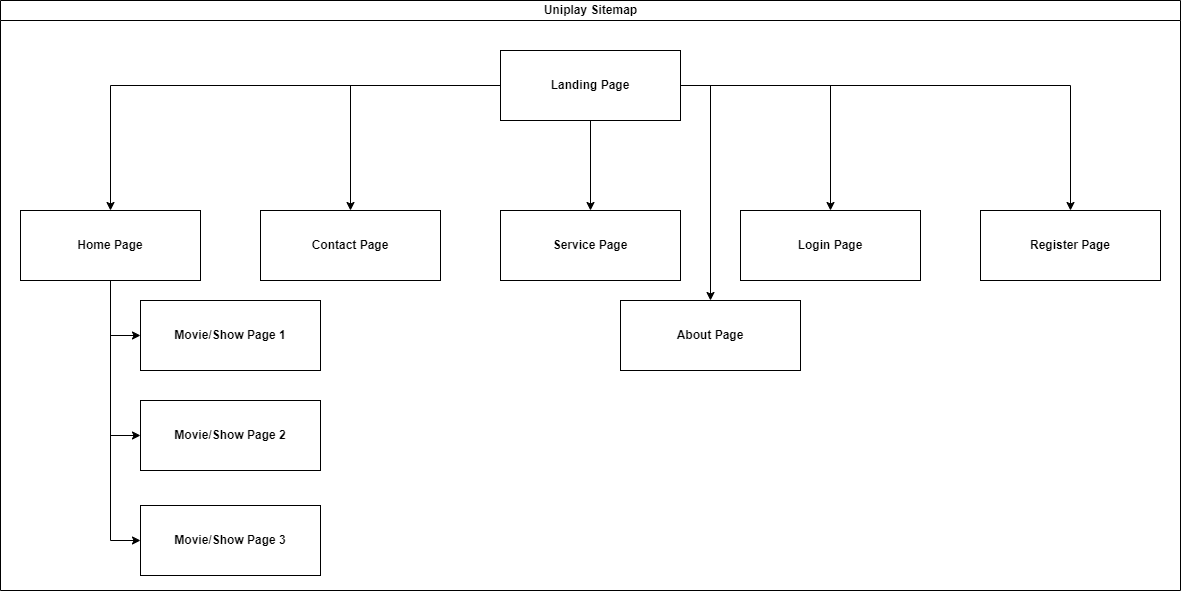
## MovieShow Page



## Service Page



# Sitemap



# PHP Pages

The following new PHP pages have been added to the Uniplay project:

1. **feedback.php**
   * **Functionality**: Allows users to submit feedback about the platform. The form collects personal information, ratings, recommendations, and additional feedback.
   * **Features**:
     + Personal Information Section: Collects name, email, phone, and age.
     + Feedback Section: Collects ratings, recommendations, services used, and additional feedback.
2. **search.php**
   * **Functionality**: Enables users to search for movies or shows. If the media is not found, users can request it.
   * **Features**:
     + Search Form: Allows users to enter a title to search.
     + Results Display: Shows media details if found.
     + Request Form: Allows users to request media if not found.
3. **submit\_feedback.php**
   * **Functionality**: Processes the feedback form submission from feedback.php.
   * **Features**:
     + Validates and sanitizes user inputs.
     + Uses prepared statements to prevent SQL injection.
     + Inserts feedback into the database.
4. **table.php**
   * **Functionality**: Displays a schedule of upcoming movie and show releases.
   * **Features**:
     + **Release Schedule Table**: Shows details such as show/movie title, type, season, episode, release date, time, and platform.
     + **Footer Row**: Indicates more content is coming soon.
5. **video.php**
   * **Functionality**: : Provides a section to watch the latest featured film trailer.
   * **Features**:
     + **Video Section**: Embeds a YouTube video trailer for the featured film.
6. **resume.php**
   * **Functionality**: Displays the user's resume.
   * **Features**:
     + **Resume Section**: Embeds a PDF of the user's resume using the <object> tag.

# Database Tables

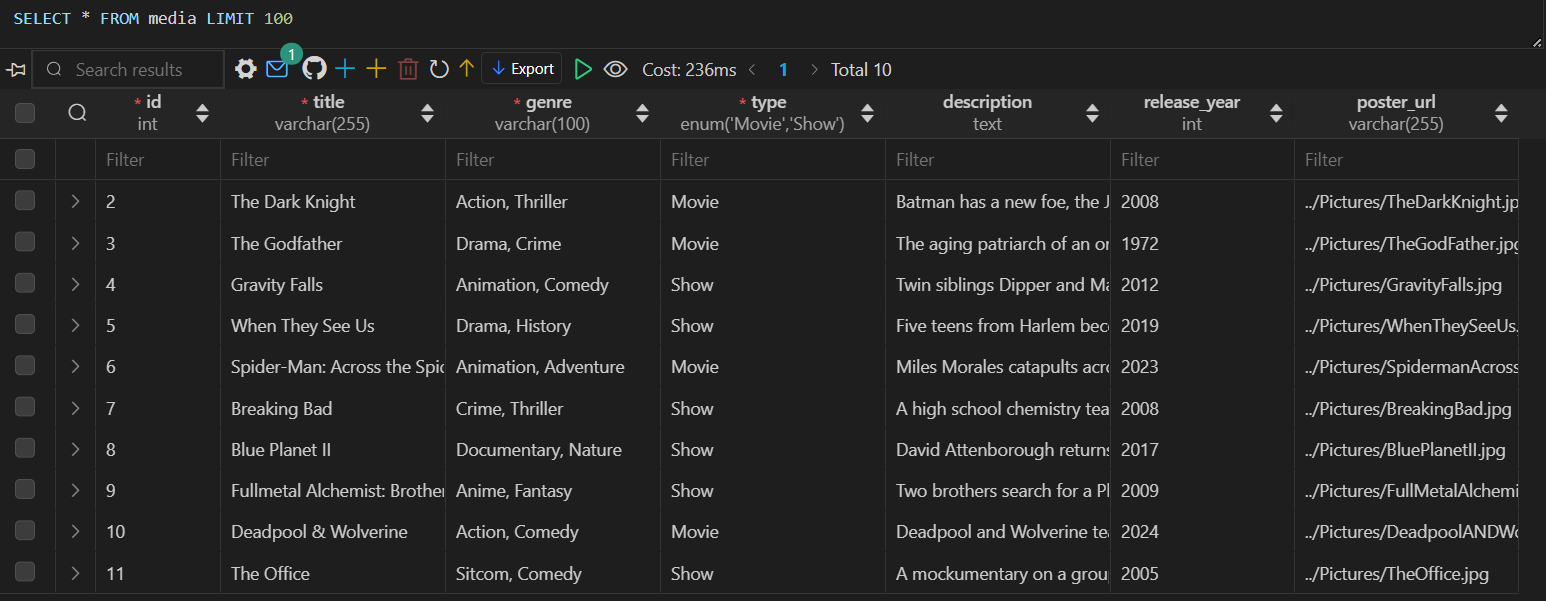
The following database tables have been created to support the functionalities of the Uniplay platform:

**1. media Table**

**Purpose**: Stores information about movies and shows.

**Structure**:

* id (INT, AUTO\_INCREMENT, PRIMARY KEY): Unique identifier for each media item.
* [title](vscode-file://vscode-app/c:/Users/Work/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) (VARCHAR(255)): Title of the media.
* genre (VARCHAR(255)): Genre of the media.
* type (ENUM('Movie', 'Show')): Type of the media (Movie or Show).
* description (TEXT): Description of the media.
* release\_year (INT): Release year of the media.
* poster\_url (VARCHAR(255)): URL of the media's poster image.

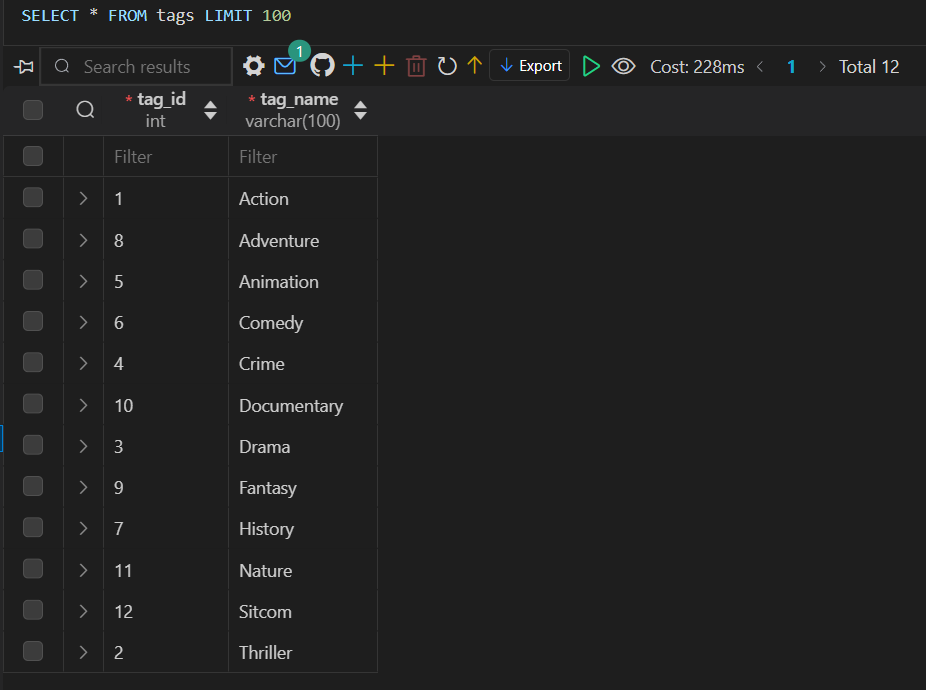


**2. tags Table**

**Purpose**: Stores tags that can be associated with media items.

**Structure**:

* tag\_id (INT, AUTO\_INCREMENT, PRIMARY KEY): Unique identifier for each tag.
* tag\_name (VARCHAR(100), UNIQUE): Name of the tag.



**3. media\_tags Table**

**Purpose**: Creates a many-to-many relationship between media items and tags.

**Structure**:

* media\_id (INT): Foreign key referencing the id column in the media table.
* tag\_id (INT): Foreign key referencing the tag\_id column in the tags table.
* PRIMARY KEY (media\_id, tag\_id): Composite primary key to ensure unique associations.
* FOREIGN KEY (media\_id) REFERENCES media(id) ON DELETE CASCADE: Ensures referential integrity.
* FOREIGN KEY (tag\_id) REFERENCES tags(tag\_id) ON DELETE CASCADE: Ensures referential integrity.

A screenshot of a computer

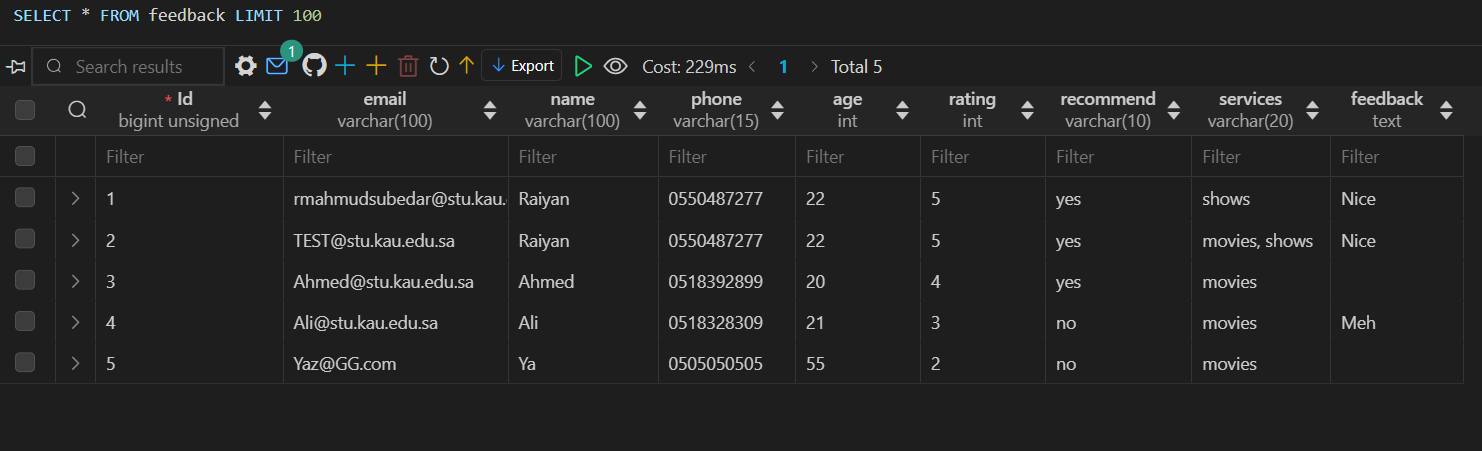
Description automatically generated

**4. feedback Table**

**Purpose**: Stores feedback submitted by users.

**Structure**:

* id (INT, AUTO\_INCREMENT, PRIMARY KEY): Unique identifier for each feedback entry.
* [email](vscode-file://vscode-app/c:/Users/Work/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) (VARCHAR(255)): Email of the user submitting the feedback.
* [name](vscode-file://vscode-app/c:/Users/Work/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) (VARCHAR(255)): Name of the user.
* [phone](vscode-file://vscode-app/c:/Users/Work/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) (VARCHAR(20)): Phone number of the user.
* [age](vscode-file://vscode-app/c:/Users/Work/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) (INT): Age of the user.
* rating (INT): Rating given by the user.
* recommend (VARCHAR(3)): Whether the user would recommend the platform (Yes or No).
* services (VARCHAR(255)): Services used by the user.
* feedback (TEXT): Additional feedback provided by the user.



**5. requests Table**

**Purpose**: Stores media requests submitted by users.

**Structure**:

* request\_id (INT, AUTO\_INCREMENT, PRIMARY KEY): Unique identifier for each request.
* [title](vscode-file://vscode-app/c:/Users/Work/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) (VARCHAR(255)): Title of the requested media.
* genre (VARCHAR(100)): Genre of the requested media.
* type (ENUM('Movie', 'Show')): Type of the requested media (Movie or Show).
* description (TEXT): Description of the requested media.
* release\_year (INT): Release year of the requested media.
* poster\_url (VARCHAR(255)): URL of the poster image for the requested media.
* user\_email (VARCHAR(255)): Email of the user submitting the request.
* request\_date (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP): Date and time when the request was submitted.
* status (ENUM('Pending', 'Reviewed', 'Added'), DEFAULT 'Pending'): Status of the request.

A screenshot of a computer

Description automatically generated

# Security Measures

To ensure the security of the application, the following measures have been implemented:

1. **SQL Injection Prevention**
   * **Prepared Statements**: All database interactions involving user input use prepared statements with parameterized queries. This prevents SQL injection attacks by ensuring that user inputs are treated as data, not executable code.
2. **Cross-Site Scripting (XSS) Prevention**
   * **Output Sanitization**: All data output to the browser is sanitized using functions like htmlspecialchars. This prevents XSS attacks by converting special characters to HTML entities, ensuring that user inputs are not executed as code.

## Client-Side and Server-Side Validation

Validation is performed on both the client and server sides to ensure data integrity and security.

1. **Client-Side Validation**
   * **JavaScript**: Validates inputs before form submission. Checks for empty fields, valid email formats, and other constraints.
   * **Example**:

document.addEventListener("DOMContentLoaded", function () {

    const form = document.querySelector("form");

    form.addEventListener("submit", function (event) {

        let isValid = true;

        const name = document.getElementById("name");

        const email = document.getElementById("email");

        const phone = document.getElementById("phone");

        const age = document.getElementById("age");

        if (name.value.trim() === "" || email.value.trim() === "" || phone.value.trim() === "" || age.value.trim() === "") {

            alert("All fields are required.");

            isValid = false;

        }

        const emailPattern = /^[a-zA-Z0-9.\_-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}$/;

        if (!emailPattern.test(email.value)) {

            alert("Please enter a valid email address.");

            isValid = false;

        }

        if (!isValid) {

            event.preventDefault();

        }

    });

});

1. **Server-Side Validation**
   * **PHP**: Validates and sanitizes inputs on the server side. Ensures that all validations done on the client side are also performed on the server side.
   * **Example**:

<?php

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

    $name = filter\_var($\_POST['name'], FILTER\_SANITIZE\_FULL\_SPECIAL\_CHARS);

    $email = filter\_var($\_POST['email'], FILTER\_SANITIZE\_EMAIL);

    $phone = filter\_var($\_POST['phone'], FILTER\_SANITIZE\_FULL\_SPECIAL\_CHARS);

    $age = filter\_var($\_POST['age'], FILTER\_SANITIZE\_NUMBER\_INT);

    $rating = filter\_var($\_POST['rating'], FILTER\_SANITIZE\_NUMBER\_INT);

    $recommend = filter\_var($\_POST['recommend'], FILTER\_SANITIZE\_FULL\_SPECIAL\_CHARS);

    $services = isset($\_POST['services']) ? implode(", ", array\_map('htmlspecialchars', $\_POST['services'])) : '';

    $feedback = filter\_var($\_POST['feedback'], FILTER\_SANITIZE\_FULL\_SPECIAL\_CHARS);

    $insertQuery = "INSERT INTO feedback (email, name, phone, age, rating, recommend, services, feedback) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";

    $insertStmt = mysqli\_prepare($conn, $insertQuery);

    mysqli\_stmt\_bind\_param($insertStmt, "sssissss", $email, $name, $phone, $age, $rating, $recommend, $services, $feedback);

    if (mysqli\_stmt\_execute($insertStmt)) {

        echo "Feedback submitted successfully!";

    } else {

        echo "Error: Could not submit feedback.";

    }

    mysqli\_stmt\_close($insertStmt);

}