BILKENT UNIVERSITY
ENGINEERING FACULTY
DEPARTMENT OF COMPUTER SCIENCE

# CS353 DATABASE SYSTEMS
# PROJECT DESIGN REPORT

Digital Reading and Sharing Platform
Group 16

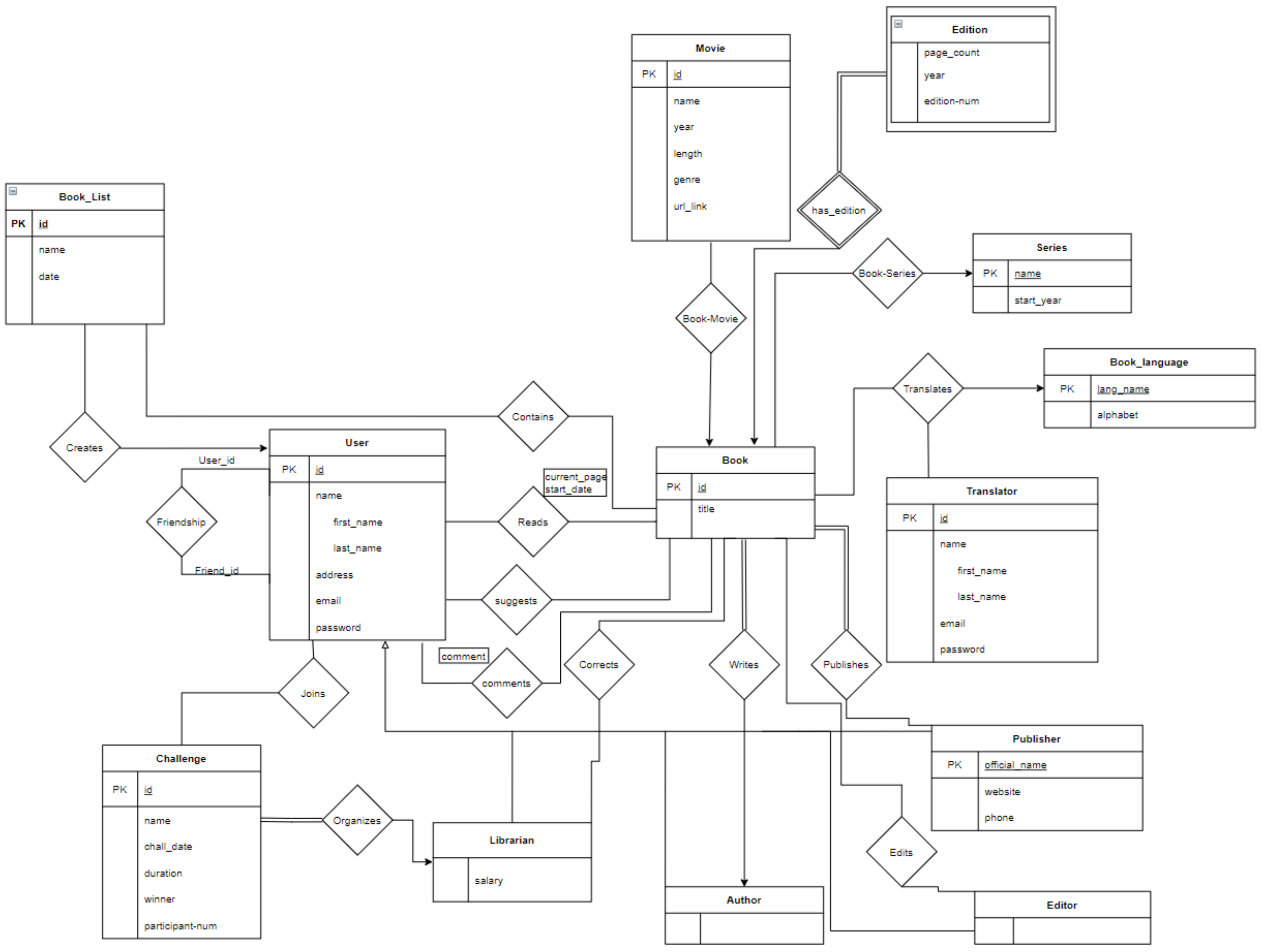
*Barış Tiftik*
*Ege Moroğlu*
*Mehmet Yiğit Harlak*
*Melisa Onaran*

Spring 2021

## Revised E/R Model

Following changes are made according to TA’s feedback and group discussions during the design process in our E/R model which leads to a more detailed and proper structure for our project.

- Edition entity which is a weak entity added and a relation named has_edition is made between Book and Edition entity.
- Specialization relation ISA added between User and Librarian entities.
- Specialization relation ISA added between User and Author entities.
- Specialization relation ISA added between User and Publisher entities.
- Specialization relation ISA added between User and Editor entities.
- Specified attributes of User, Librarian, Author, Editor and Publisher entities are rearranged due to added ISA relations.
- Suggests relation added between User and Book entities to provide user to suggest books.
- Reads relation between User and Book entities is modified to provide a book tracking progress for users.
- Movie entity which is indicating movie of a book and added as an extra functionality is extended to provide a YouTube video as a teaser.
- Comments relation between User and Book entities is modified to provide a comment in a text form



Movie
PK id
name
year
length
genre
url_link
Edition
page_count
year
edition-num
has_edition
Book_List
PK id
name
date
Book-Movie
Book-Series
Series
PK name
start_year
Book_language
PK lang_name
alphabet
Translates
Contains
Creates
User
PK id
name
first_name
last_name
address
email
password
User_id
Friendship
Friend_id
current_page
start_date
Reads
Book
PK id
title
Translator
PK id
name
first_name
last_name
email
password
suggests
comment
comments
Corrects
Writes
Publishes
Joins
Challenge
PK id
name
chall_date
duration
winner
participant-num
Organizes
Librarian
salary
Publisher
PK official_name
website
phone
Edits
Author
Editor

# Table Schemas

Table schemas with its corresponding relational model, functional dependencies, candidate keys and normal form is given in this section.

## User

**Relational Model:**
User(id, first_name, last-name, address, e-mail, password)

**Functional Dependencies:**
id → first_name, last-name, address, e-mail, password
e-mail → first_name, last-name, address, id, password

**Candidate Keys:**
{(id), (e-mail)}

**Normal Form:**
3NF

**Table Definition:**

```
CREATE TABLE User(
        id tinyint not null,
        first_name varchar(25),
        last_name varchar (25),
        address varchar(225),
        email varchar(25) not null,
        password varchar(25) not null,
        PRIMARY KEY (id)
);
```

## Author

**Relational Model:**
Author(id)
FK: id references User

**Functional Dependencies:**
None

**Candidate Keys:**
{(id)}

**Normal Form:**
3NF

**Table Definition:**

```
CREATE TABLE Author(
        id tinyint not null,
        PRIMARY KEY (id),
        FOREIGN key (id) references User (id)
);
```

## Publisher

**Relational Model:**
Publisher(id, official_name, website, phone)
FK: id references User

**Functional Dependencies:**
id, official_name → website, phone

**Candidate Keys:**
{(id,official_name)}

**Normal Form:**
3NF

**Table Definition:**

```
CREATE TABLE Publisher(
        id tinyint not null,
        official_name varchar (25) not null,
        website varchar (30),
        phone bigint,
        PRIMARY KEY (id, official_name),
        FOREIGN KEY (id) references User (id)
);
```

## Book

**Relational Model:**
Book(id, title, serie_name, author_id)
FK: serie_name references Series
FK: author_id references Author

**Functional Dependencies:**
id → title, serie_name, author_id

**Candidate Keys:**
{(id)}

**Normal Form:**
3NF

**Table Definition:**

```
CREATE TABLE Book(
        id tinyint not null,
        title varchar (25) not null,
        author_id tinyint not null,
        serie_name varchar (25),
        PRIMARY KEY (id),
        FOREIGN KEY (serie_name) references Series (name)
        FOREIGN KEY (author_id) references Author (id)
);
```

## Translator

**Relational Model:**
Translator(id, name, e-mail, password)

**Functional Dependencies:**
id → name, e-mail, password
e-mail → name, id, password

**Candidate Keys:**
{(id), (e-mail)}

**Normal Form:**
3NF

**Table Definition:**
```
CREATE TABLE Translator(
        id tinyint not null,
        name varchar (25) not null,
        email varchar (25) not null,
        password varchar (25) not null,
        PRIMARY KEY (id)
);
```

## Librarian

**Relational Model:**
Librarian(id, salary)
FK: id references User

**Functional Dependencies:**
id → salary

**Candidate Keys:**
{(id)}

**Normal Form:**
3NF

**Table Definition:**

```
CREATE TABLE Librarian(
        id tinyint not null,
        salary int not null,
        PRIMARY KEY (id),
        FOREIGN KEY (id) references User (id)
);
```

## Movie

**Relational Model:**

Movie(id, name, year, length, genre, url_link, book_id)
FK: book_id references Book

**Functional Dependencies:**
id → name, year, length, genre, url_link, book_id)

**Candidate Keys:**
{(id)}

**Normal Form:**
3NF

**Table Definition:**

```
CREATE TABLE Movie(
        id tinyint not null,
        name varchar (25) not null,
         year bigint,
         length tinyint,
        genre varchar(8),
        url_link varchar(30),
        book_id tinyint not null,
        PRIMARY KEY (id),
        FOREIGN KEY (book_id) references book (id)
);
```

## Edition

**Relational Model:**
Edition(book_id, edition_num, year, page_count)
FK: book_id references Book

**Functional Dependencies:**
None

**Candidate Keys:**
{(book_id, edition_num, year, page_count)}

**Normal Form:**
3NF

**Table Definition:**
```
CREATE TABLE Edition(
        book_id tinyint not null,
        edition_num tinyint not null,
        page_count smallint not null,
        year smallint not null,
        PRIMARY KEY (book_id, edition_num, page_count, year),
        FOREIGN KEY (book_id) references Book (id)
);
```

## Editor

**Relational Model:**
Editor(<u>id</u>)
FK: id references User

**Functional Dependencies:**
None

**Candidate Keys:**
{(id)}

**Normal Form:**
3NF

**Table Definition:**

```
CREATE TABLE Editor(
            id tinyint not null,
            PRIMARY KEY (id),
            FOREIGN KEY (id) references User (id)
);
```

## Book_Language

**Relational Model:**
Book_Language(<u>lang_name</u>, alphabet)

**Functional Dependencies:**
lang_name → alphabet

**Candidate Keys:**
{(lang_name)}

**Normal Form:**
3NF

**Table Definition:**
```
CREATE TABLE Book_language(
		lang_name varchar (8) not null,
		alphabet varchar (30) not null,
		PRIMARY KEY (lang_name)
);
```

## Series

**Relational Model:**
Series(<u>name</u>, start_year, book_id)
FK: book_id references Book

**Functional Dependencies:**
name → start_year

**Candidate Keys:**
{(name)}
**Normal Form:**
3NF

**Table Definition:**
```
CREATE TABLE Series(
        name varchar (25) not null,
        start_year BIGINT,
        book_id tinyint not null,
        PRIMARY KEY (name),
        FOREIGN KEY (book_id) references Book (id)
);
```

## Challenge

**Relational Model:**
Challenge(<u>id</u>, name, date, duration, winner, particip-num, librar-id)

**Functional Dependencies:**
id → name, date, duration, winner, particip-num, librar-id
FK: winner references User
FK: librarian_id references Librarian

**Candidate Keys:**
{(id)}

**Normal Form:**
3NF

**Table Definition:**

```
CREATE TABLE Challenge(
        id tinyint not null,
        name varchar (25) not null,
        chall_date timestamp not null,
        duration tinyint,
        winner varchar (25),
        particip_num tinyint,
        librarian_id tinyint,
        PRIMARY KEY (id),
        FOREIGN KEY (winner) references User (name),
        FOREIGN KEY (librarian_id) references Librarian (id)
);
```

## Book_list

**Relational Model:**
Book_List (list_id, name, date, user-id)
FK: user_id references User

**Functional Dependencies:**
list_id → name, date, user-id
name, user_id → list_id, date

**Candidate Keys:**
{(list_id), (name, user_id)}

**Normal Form:**
3NF

**Table Definition:**

```
CREATE TABLE Book_list(
		id tinyint not null,
		name varchar (25) not null,
		list_date date,
		user_id tinyint not null,
		PRIMARY KEY (id),
		FOREIGN KEY (user_id) references User (id)
);
```

## Joins

**Relational Model:**
Joins(user_id, chall_id)
FK : user_id references User
FK : chall_id references Challenge

**Functional Dependencies:**
None

**Candidate Keys:**
{(user_id, chall_id)}

**Normal Form:**
3NF

**Table Definition:**
```
CREATE TABLE Joins(
   user_id tinyint not null,
   chall_id tinyint not null,
   PRIMARY KEY (user_id, chall_id),
   FOREIGN KEY (user_id) references User(id),
   FOREIGN KEY (chall_id) references Challange(id)
);
```

## Contains

**Relational Model:**
Contains(list_id, book_id)
FK : list_id to Book_List
FK : book_id to Book

**Functional Dependencies:**
None

**Candidate Keys:**
{(list_id, book_id)}

**Normal Form:**
3NF

**Table Definition:**
```
CREATE TABLE Contains(
   list_id tinyint not null,
   book_id tinyint not null,
   PRIMARY KEY (list_id, book_id)
   FOREIGN KEY (list_id) references Book_List(list_id)
   FOREIGN KEY (book_id) references Book(id)

);
```

## Reads

**Relational Model:**
Reads(<u>user_id, book_id</u>, current_page, start_date)
FK : user_id references User
FK : book_id references Book

**Functional Dependencies:**
user_id, book_id → current_page, start_date

**Candidate Keys:**
{(user_id, book_id)}

**Normal Form:**
3NF

**Table Definition:**
```
CREATE TABLE Reads(
   user_id tinyint not null,
   book_id tinyint not null,
   current_page int,
   start_date date,
   PRIMARY KEY (user_id, book_id),
   FOREIGN KEY (user_id) references User(id),
   FOREIGN KEY (book_id) references Book(id)
);
```

## Suggests

**Relational Model:**
Suggests(user_id, book_id)
FK : user_id references User
FK : book_id references Book
**Functional Dependencies:**
None

**Candidate Keys:**
{(user_id, book_id)}

**Normal Form:**
3NF

**Table Definition:**
```
CREATE TABLE Suggests(
    user_id tinyint not null,
    book_id tinyint not null,
    PRIMARY KEY (user_id, book_id),
    FOREIGN KEY (user_id) references User(id),
    FOREIGN KEY (book_id) references Book(id)
);
```

## Comments

**Relational Model:**
Comments(user_id, book_id, comment)
FK : user_id references User
FK : book_id references Book

**Functional Dependencies:**
user_id, book_id → comment

**Candidate Keys:**
{(user_id, book_id)}

**Normal Form:**
3NF

**Table Definition:**
```
CREATE TABLE Comments(
   user_id tinyint not null,
   book_id tinyint not null,
   comment varchar (5000),
   PRIMARY KEY (user_id, book_id),
   FOREIGN KEY (user_id) references User(id),
   FOREIGN KEY (book_id) references Book(id)

);
```

## Friendship

**Relational Model:**
Friendship(user_id, friend_id)
FK : user_id references User
FK : friend_id references User

**Functional Dependencies:**
None

**Candidate Keys:**
{(user_id, friend_id)}

**Normal Form:**
3NF

**Table Definition:**
```
CREATE TABLE Friendship(
   user_id tinyint not null,
   friend_id tinyint not null,
   PRIMARY KEY (user_id, friend_id),
   FOREIGN KEY (user_id) references User(id),
   FOREIGN KEY (friend_id) references User(id)

);
```

## Corrects

**Relational Model:**
Corrects(<u>librar_id, book_id</u>)
FK : librar_id references Librarian
FK : book-id references Book

**Functional Dependencies:**
None

**Candidate Keys:**
{(librar_id, book_id)}

**Normal Form:**
3NF

**Table Definition:**
```
CREATE TABLE Corrects(
   librar_id tinyint not null,
   book_id tinyint not null,
   PRIMARY KEY (librar_id, book_id),
   FOREIGN KEY (librar_id) references Librarian(id),
   FOREIGN KEY (book_id) references Book(id)

);
```

## Edits

**Relational Model:**
Edits(<u>editor_id, book_id</u>)
FK : editor_id references Editor
FK : book_id references Book

**Functional Dependencies:**
None

**Candidate Keys:**
{(editor_id, book_id)}

**Normal Form:**
3NF

**Table Definition:**
```
CREATE TABLE Edits(
   editor_id tinyint not null,
   book_id tinyint not null,
   PRIMARY KEY (editor_id, book_id),
   FOREIGN KEY (editor_id) references Editor(id),
   FOREIGN KEY (book_id) references Book(id)

);
```

## Publishes

**Relational Model:**
Publishes(publisher_id, publisher_name,  book_id)
FK : publisher_id references Publisher
FK : publisher_name references Publisher
FK : book_id references Book

**Functional Dependencies:**
None

**Candidate Keys:**
{(publisher_id, publisher_name, book_id)}

**Normal Form:**
3NF

**Table Definition:**
```
CREATE TABLE Publishes(
    publisher_id tinyint not null,
    publisher_name varchar (25) not null
    book_id tinyint not null,
    PRIMARY KEY (publisher_id, publisher_name, book_id),
    FOREIGN KEY (publisher_id) references Publisher(id),
    FOREIGN KEY (publisher_name) references Publisher(official_name),
    FOREIGN KEY (book_id) references Book(id)

);
```

## Translates

**Relational Model:**
Translates(<u>translator_id, book_id, lang_name</u>)
FK : translator_id to Translator
FK : book_id to Translator
FK : lang_name to Book_Language

**Functional Dependencies:**
None

**Candidate Keys:**
{(translator_id, book_id, lang_name)}

**Normal Form:**
3NF

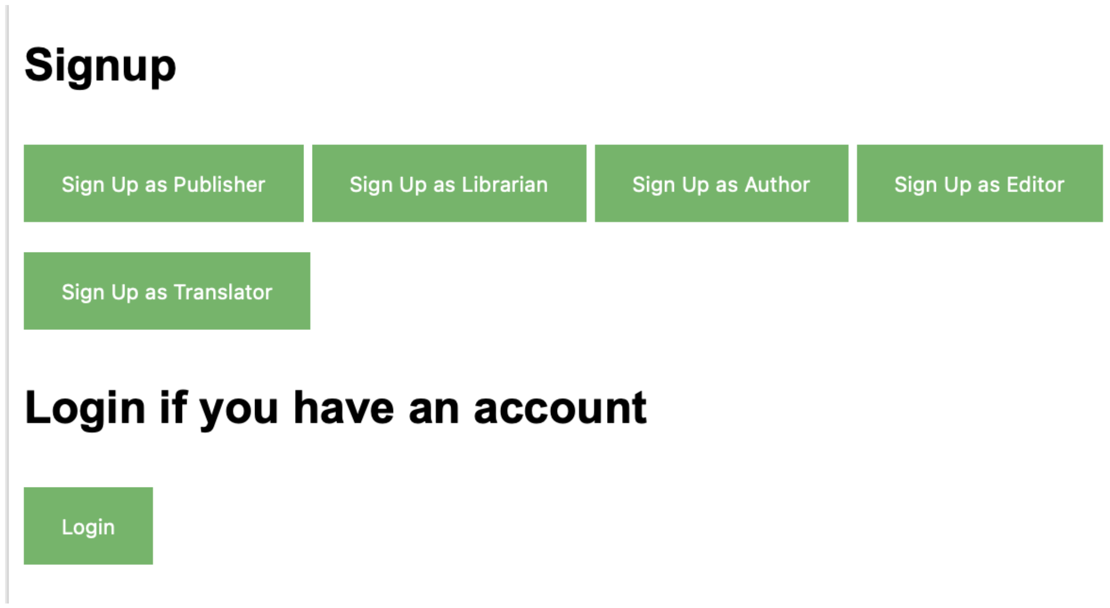
**Table Definition:**
```
CREATE TABLE Translates(
    translator_id tinyint not null,
    book_id tinyint not null,
    lang_name varchar (8) not null
    PRIMARY KEY (translator_id, book_id, lang_name),
    FOREIGN KEY (translator_id) references Translator(id),
    FOREIGN KEY (book_id) references Book(id)
    FOREIGN KEY (lang_name) references Book_Language(lang_name)

);
```

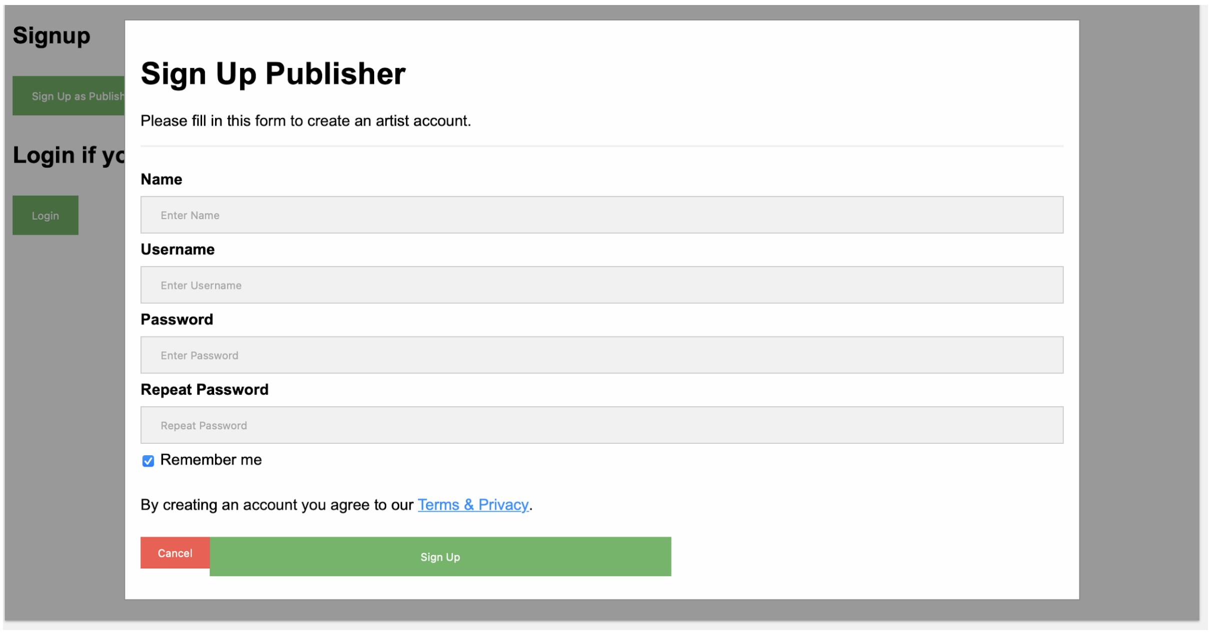
# UI design and Corresponding SQL statements

User interface design and its corresponding SQL statements has shown in this section.

## Sign Up and Login Choice Page



## Publisher Sign Up Page


Signup

Sign Up as Publish

Login if yo

Login

**Sign Up Publisher**

Please fill in this form to create an artist account.

**Name**

Enter Name

**Username**

Enter Username

**Password**

Enter Password

**Repeat Password**

Repeat Password

☑ Remember me

By creating an account you agree to our Terms & Privacy.

Cancel

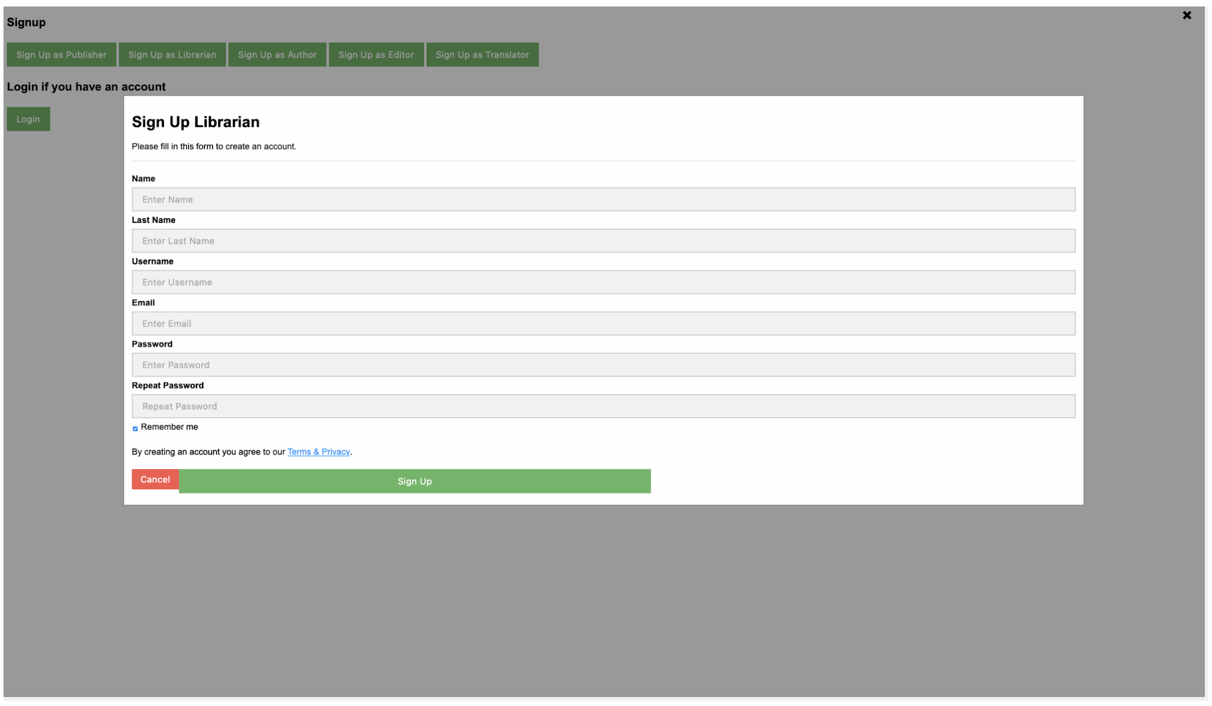
Sign Up

Sign up new Publisher:
ID primary key of the User  is considered as the username in queries below.
Inputs:
@fname, @username, @password, @repeatedpassword

insert into User VALUES(@username, @fname, null, null, null, @password)
WHERE @password = @repeatedpassword

insert into Publisher VALUES(@username, @fname, null, 0)

## Librarian Sign Up Page


Signup

Login if you have an account

Sign Up Librarian

Please fill in this form to create an account.

Name

Enter Name

Last Name

Enter Last Name

Username

Enter Username

Email

Enter Email

Password

Enter Password

Repeat Password

Repeat Password

Remember me

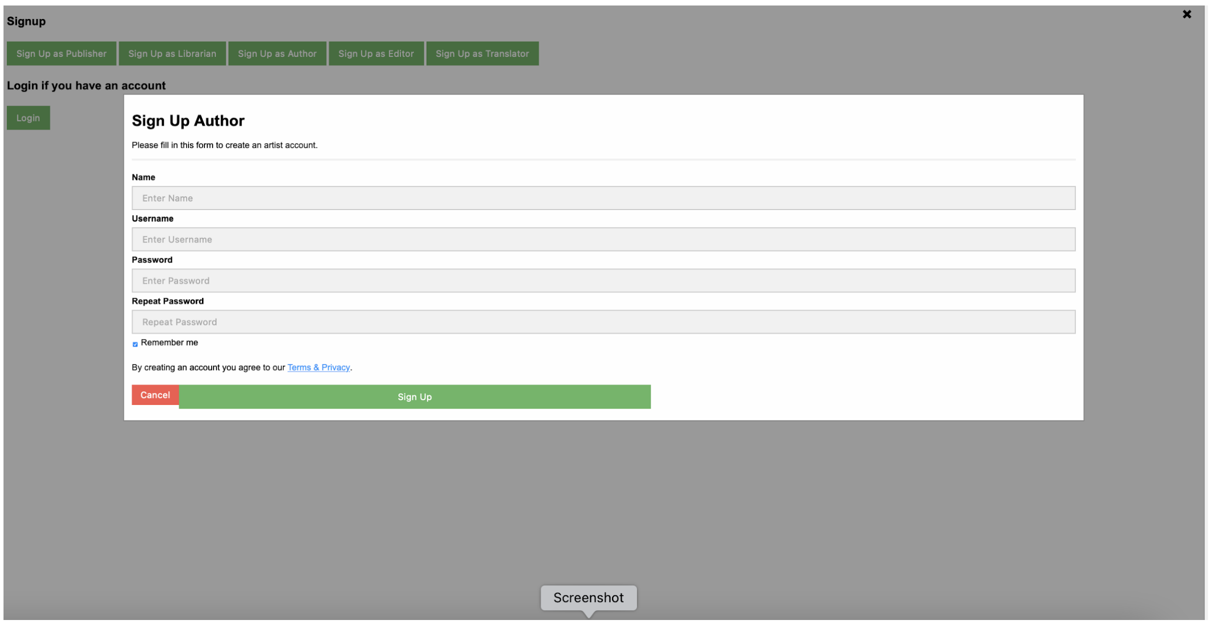
By creating an account you agree to our Terms & Privacy.

Sign up new Librarian:
Inputs:
@fname, @lname, @username, @email, @password, @repeatedpassword

insert into User VALUES(@username, @fname, @lname, null, null, @password)
WHERE @password = @repeatedpassword

insert into Librarian VALUES(@username, 0)

## Author Sign Up Page



Signup

Sign Up as Publisher | Sign Up as Librarian | Sign Up as Author | Sign Up as Editor | Sign Up as Translator

Login if you have an account

Login

Sign Up Author

Please fill in this form to create an artist account.

Name: Enter Name

Username: Enter Username

Password: Enter Password

Repeat Password: Repeat Password

Remember me

By creating an account you agree to our Terms & Privacy.

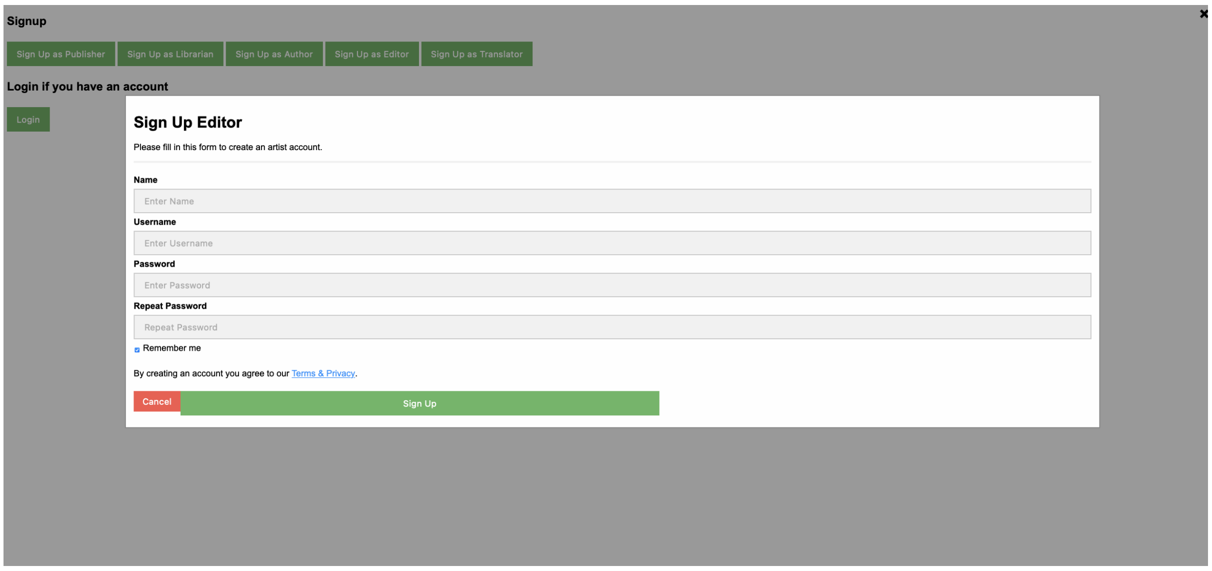
Cancel | Sign Up

Sign up new Author:
Inputs:
@fname, @username,@password, @repeatedpassword

insert into User VALUES(@username, @fname, null, null, null, @password)
WHERE @password = @repeatedpassword

insert into Author VALUES (@username)

## Editor Sign Up Page

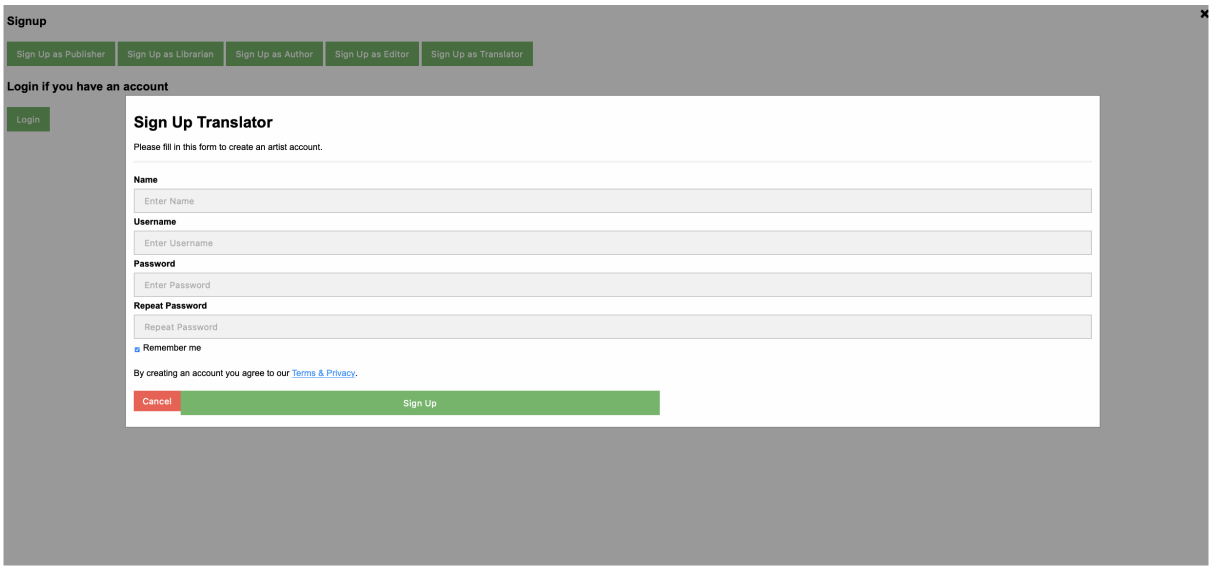


Sign up new Editor:
Inputs:
@fname, @username, @password, @repeatedpassword

insert into User VALUES(@username, @fname, null, null, null, @password)
WHERE @password = @repeatedpassword

insert into Editor VALUES(@username)

## Translator Sign Up Page



Sign Up Translator

Please fill in this form to create an artist account.

Name

Enter Name

Username

Enter Username

Password

Enter Password

Repeat Password

Repeat Password

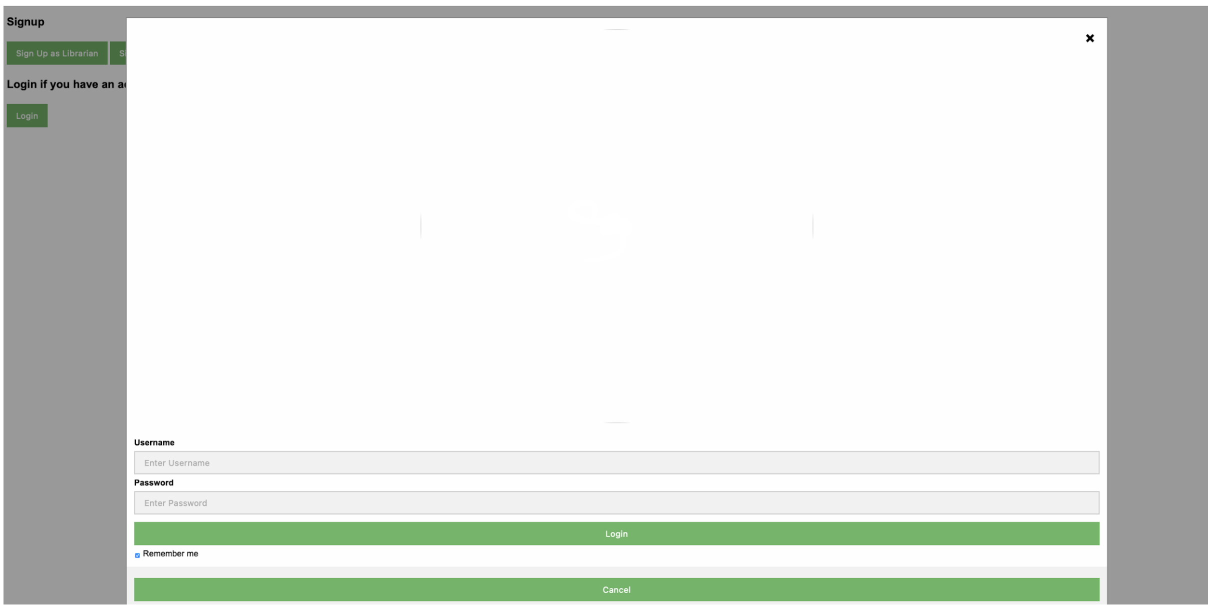
Remember me

By creating an account you agree to our Terms & Privacy.

Sign up new Translator:
Inputs:
@fname, @username, @password, @repeatedpassword

insert into Translator VALUES(@username, @fname, null, @password)
WHERE @password = @repeatedpassword

## Login Page


Signup

Sign Up as Librarian

Login if you have an a

Login

Username

Enter Username

Password

Enter Password

Login

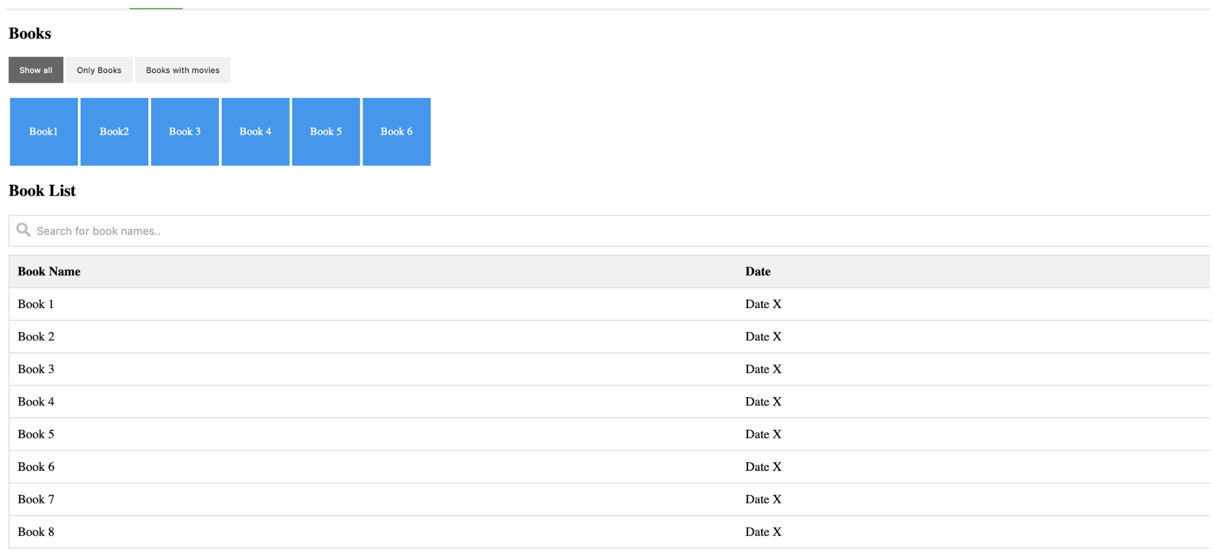
Remember me

Cancel

Sign up new Translator:
Inputs:
@username, @password

SELECT * FROM User
WHERE id = @username and password = @password

## Books Page



**Books**

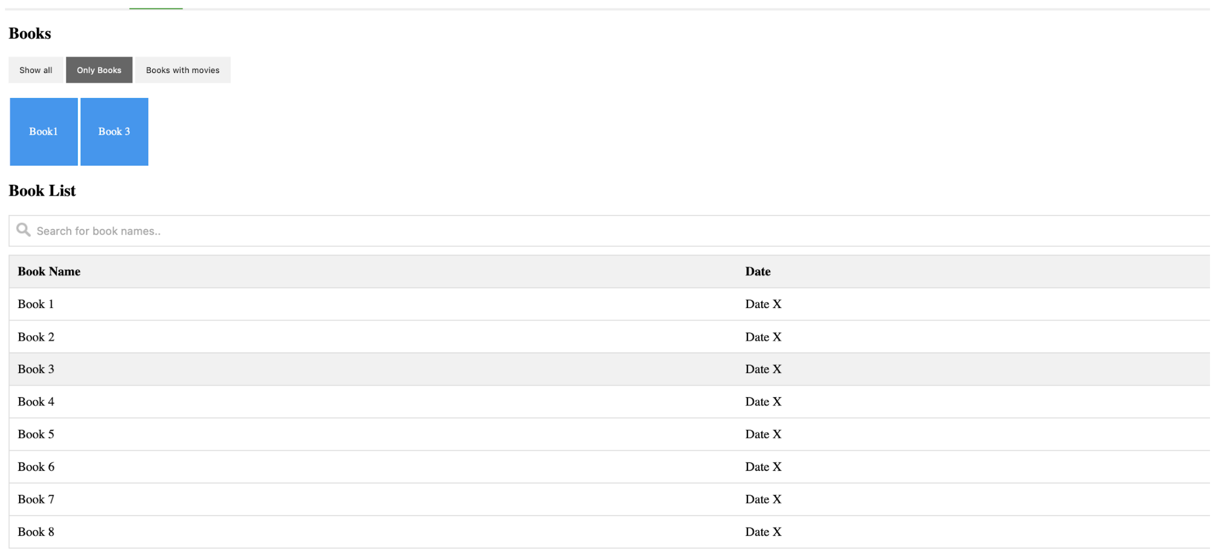
**Book List**

| Book Name | Date |
|---|---|
| Book 1 | Date X |
| Book 2 | Date X |
| Book 3 | Date X |
| Book 4 | Date X |
| Book 5 | Date X |
| Book 6 | Date X |
| Book 7 | Date X |
| Book 8 | Date X |

Display all books of the user X reads:

```
SELECT B.title, B.date
FROM     User AS U, Reads AS R, Books AS B,
WHERE U.id = R.user_id AND U.first_name = ‘X’ AND  R.book_id = B.id;
```

## Books without Movies Page



**Books**

**Book List**

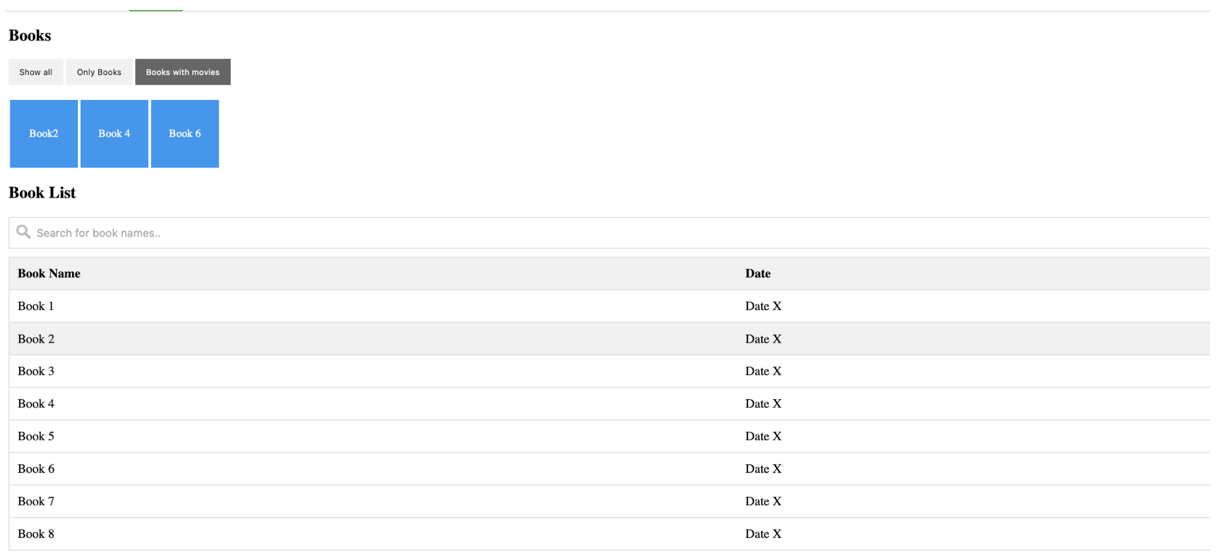
| Book Name | Date |
|---|---|
| Book 1 | Date X |
| Book 2 | Date X |
| Book 3 | Date X |
| Book 4 | Date X |
| Book 5 | Date X |
| Book 6 | Date X |
| Book 7 | Date X |
| Book 8 | Date X |

Display all books of the user X reads which not have movies:

Display all books of the user X reads:

```
( SELECT B.title, B.date
FROM     User AS U, Reads AS R, Books AS B,
WHERE U.id = R.user_id AND U.first_name = ‘X’ AND  R.book_id = B.id )
-
( SELECT B.title, B.date
FROM     User AS U, Reads AS R, Books AS B, Movies  AS M
WHERE U.id = R.user_id AND U.first_name = ‘X’ AND  R.book_id = B.id AND M.book_id =
B.id );
```

## Books with Movies Page


**Books**

Show all | Only Books | Books with movies

Book2 | Book 4 | Book 6

**Book List**

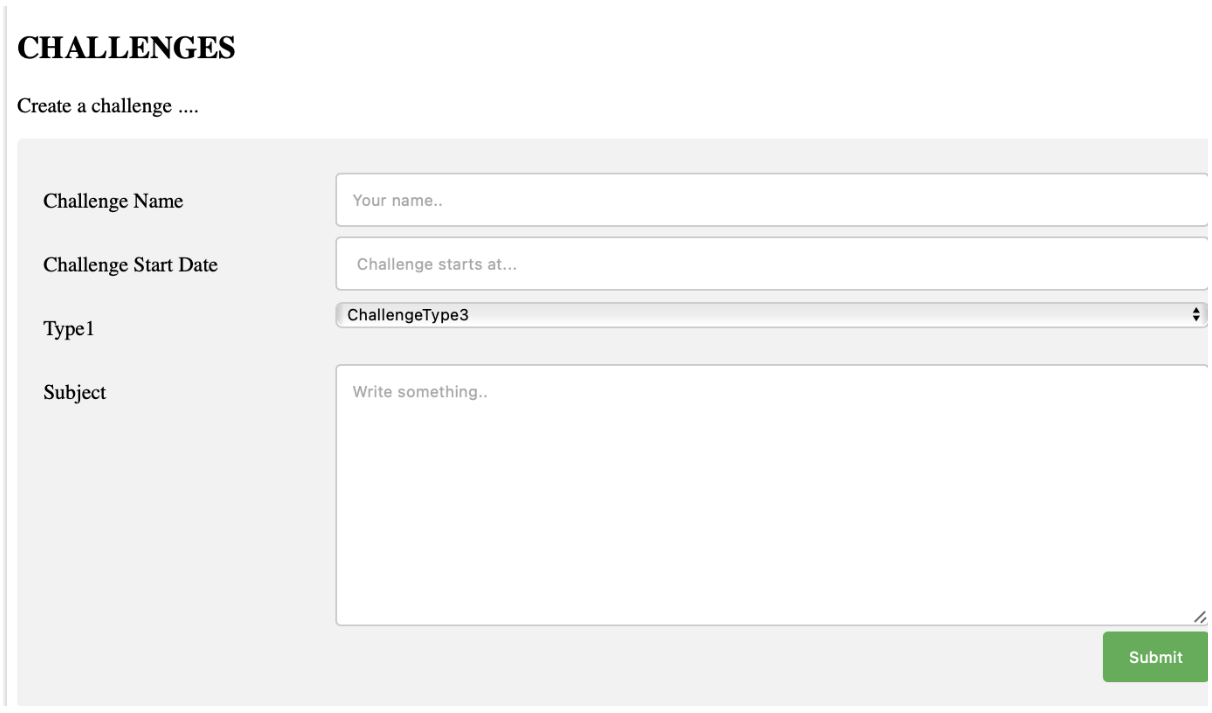
Search for book names..

| Book Name | Date |
|---|---|
| Book 1 | Date X |
| Book 2 | Date X |
| Book 3 | Date X |
| Book 4 | Date X |
| Book 5 | Date X |
| Book 6 | Date X |
| Book 7 | Date X |
| Book 8 | Date X |

Display all books of the user X reads which have movies:

```
SELECT B.title, B.date
FROM     User AS U, Reads AS R, Books AS B, Movies  AS M
WHERE U.id = R.user_id AND U.first_name = ‘X’ AND  R.book_id = B.id AND M.book_id =
B.id;
```

## Challenge Arrangement Page


CHALLENGES

Create a challenge ....

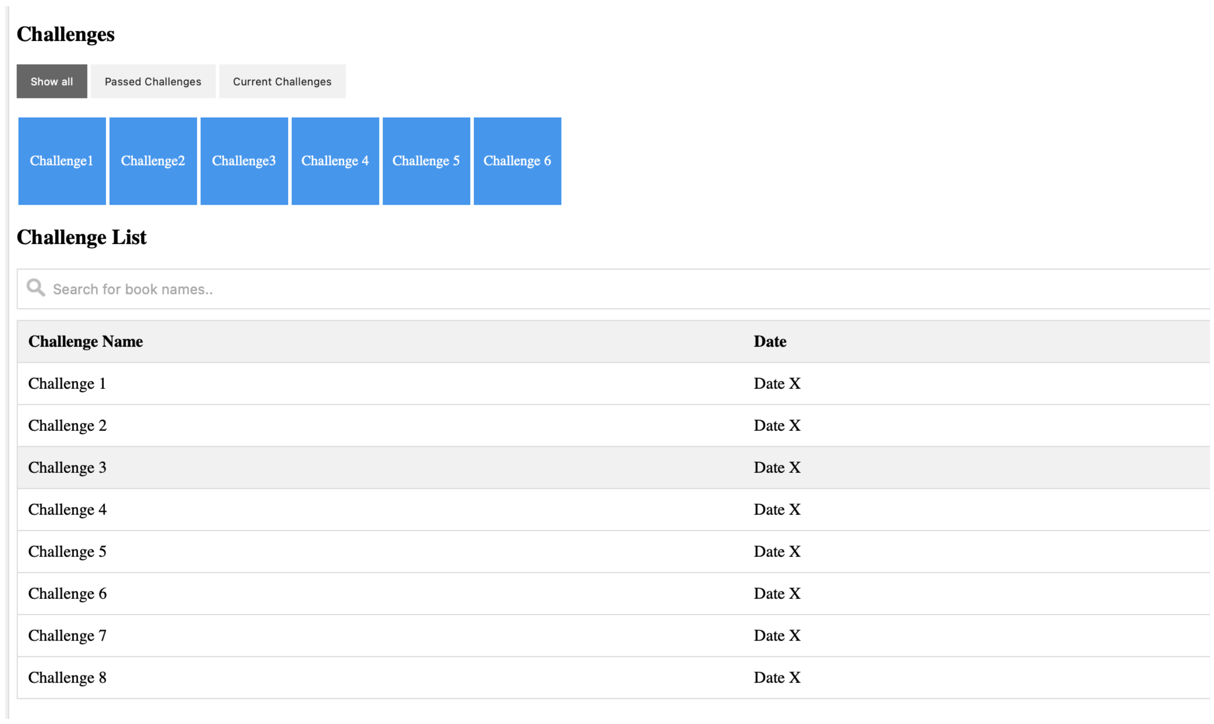
| | |
|---|---|
| Challenge Name | Your name.. |
| Challenge Start Date | Challenge starts at... |
| Type1 | ChallengeType3 |
| Subject | Write something.. |

Get detailed input from librarian to organize a challenge:

Inputs:
@cname, @sdate, @stype, @subject

insert into Challenge VALUES(0, @cname, @sdate, null, null, 0, 0)

## Challenge View Page



| Challenge Name | Date |
|---|---|
| Challenge 1 | Date X |
| Challenge 2 | Date X |
| Challenge 3 | Date X |
| Challenge 4 | Date X |
| Challenge 5 | Date X |
| Challenge 6 | Date X |
| Challenge 7 | Date X |
| Challenge 8 | Date X |

View all challenges arranged (with more than one user) until now:

```
SELECT C.name, C.chall_date
FROM    Challenge AS C
WHERE C.participant_num > 1;
```

## QUERIES

Following SQL queries will be also included in our project and related user interfaces will be implemented in the implementation step of our project.

1- Retrieve book ids user X reads

```
SELECT R.book_id
FROM    User AS U, Reads AS R
WHERE U.id = R.user_id AND U.first_name = ‘X’;
```

2- Retrieve the last page number read user X reads the book X

```
SELECT R.current-page
FROM    User AS U, Reads AS R, Book AS B
WHERE U.id = R.user_id AND R.book_id = B.id AND U.first_name = ‘X’ AND B.title = ‘X’;
```

3- Retrieve book names user X have completed

```
SELECT B.title
FROM    User AS U, Reads AS R, Book AS B, Edition  AS E
WHERE U.id = R.user_id AND R.book_id = B.id AND B.id = E.book_id AND
            U.first_name = ‘X’  AND R.current_page = E.page_count;
```

4- Retrieve book titles user X haven't completed yet

```
SELECT B.title
FROM    User AS U, Reads AS R, Book AS B, Edition  AS E
WHERE U.id = R.user_id AND R.book_id = B.id AND B.id = E.book_id AND
            U.first_name = ‘X’  AND R.current_page < E.page_count;
```

5- Retrieve book titles and edition numbers that are published in X

```
SELECT B.title, E.edition_num
FROM    Book as B, Edition as E
WHERE B.id = E.book_id AND E.year = X;
```

6- Retrieve author names and book counts that have X or more books

```
SELECT A.first_name, COUNT(B.id)
FROM    Author AS A, Book AS B
WHERE A.id = B.author_id
GROUP BY A.id
HAVING COUNT(B.id) >= X;
```

7- Retrieve book names that have movie

```
SELECT B.title
```

```
FROM    Book AS B, Movie as M
WHERE B.id = M.book_id;
```

8-  Retrieve book names that are written in X Language

```
SELECT B.title
FROM    Book AS B, Translates AS T
WHERE B.id = T.book_id AND T.lang_name = ‘X’;
```

9-  Retrieve book names that are published by X (official_name)

```
SELECT B.title
FROM    Book AS B, Publishes AS P
WHERE B.id = P.book_id AND P.publisher_name = ‘X’;
```

10-  Retrieve book names that are edited by X

```
SELECT B.title
FROM    Book AS B, Edits AS EDTS, Editor AS EDTR
WHERE B.id = EDTS.book_id AND EDTS.editor_id = EDTR.id AND EDTR.first_name = ‘X’;
```

11-  Retrieve the book X that is translated by X

```
SELECT B.title
FROM    Book AS B, Translates AS TS, Translator AS  TR
WHERE B.id = TS.book_id AND TS.translator_id = TR.id AND B.title = ‘X’ AND
TR.first_name = ‘X’;
```

12-  Retrieve librarian ids who correct book X

```
SELECT L.id
FROM    Librarian AS L, Corrects AS C, Book as B
WHERE L.id = C.librar_id AND C.book_id = B.id AND B.title = ‘X’;
```

13-  Retrieve challenge names which have no winner and min. ten participants

```
SELECT C.name
FROM    User AS U, Joins AS J, Challenge AS C
WHERE U.id = J.user_id AND J.chall_id = C.id AND C.winner = null
        AND 10 <= ( SELECT COUNT(user_id)
                    FROM    Join AS J2
                    WHERE J2.chall_id = C.id );
```

14-  Retrieve challenge names which have winner X and min. ten participants

```
SELECT C.name
FROM    User AS U, Joins AS J, Challenge AS C
```

```
WHERE U.id = J.user_id AND J.chall_id = C.id AND C.winner = 'X'
        AND 10 <= ( SELECT COUNT(user_id)
                    FROM   Join AS J2
                    WHERE J2.chall_id = C.id );
```

15- Retrieve user X's friend names

```
SELECT U2.first_name
FROM    User AS U1, Friendship AS F, User AS U2
WHERE  U1.id = F.user_id AND F.friend_id = U2.id AND U1.first_name = 'X';
```

16- Retrieve user X's book list names

```
SELECT BL.name
FROM   User AS U, Book_List as BL
WHERE U.id = BL.user_id AND U.first_name = 'X';
```

17- Retrieve book list X's includings

```
SELECT B.title
FROM   Book_List AS BL, Contains AS C, Book AS B
WHERE BL.id = C.list_id AND C.book_id = B.id AND BL.name = 'X';
```

## Website Address

https://github.com/egemoroglu/Digital_Reading_And_Sharing_Platform

## References

[1] “goodreads” [online] available: www.goodreads.com , Accessed March 31, 2021

[2] “Flowchart Maker & Online Diagram Software” [Online] available: app.diagrams.net , Accessed: March 30, 2021