BILKENT UNIVERSITY
ENGINEERING FACULTY
DEPARTMENT OF COMPUTER SCIENCE

CS353 DATABASE SYSTEMS
PROJECT FINAL REPORT

Digital Reading and Sharing Platform
Group 16

Barış Tiftik
Ege Moroğlu
Mehmet Yiğit Harlak
Melisa Onaran

Spring 2021

Application System Description

Digital Reading and Sharing Platform is a web-based application which maintains information and interaction regarding reading habits of the people. System features information about books that a user is currently reading, have been read or s/he will read. Further information related to those books such as its author, page number, edition, movies etc. Additionally, the platform provides users to compete in reading challenges with other users. Those challenges are focused on counting books or the number of pages users read in a time interval and the user who has the most pages becomes the winner.

Six different user domains are included in the system which are user (reader), publisher, librarian, author, editor and translator.

Readers are able to add/remove books to their profile, create booklists, mark their progress on books, rate or comment on books or suggest books on their profile as well.

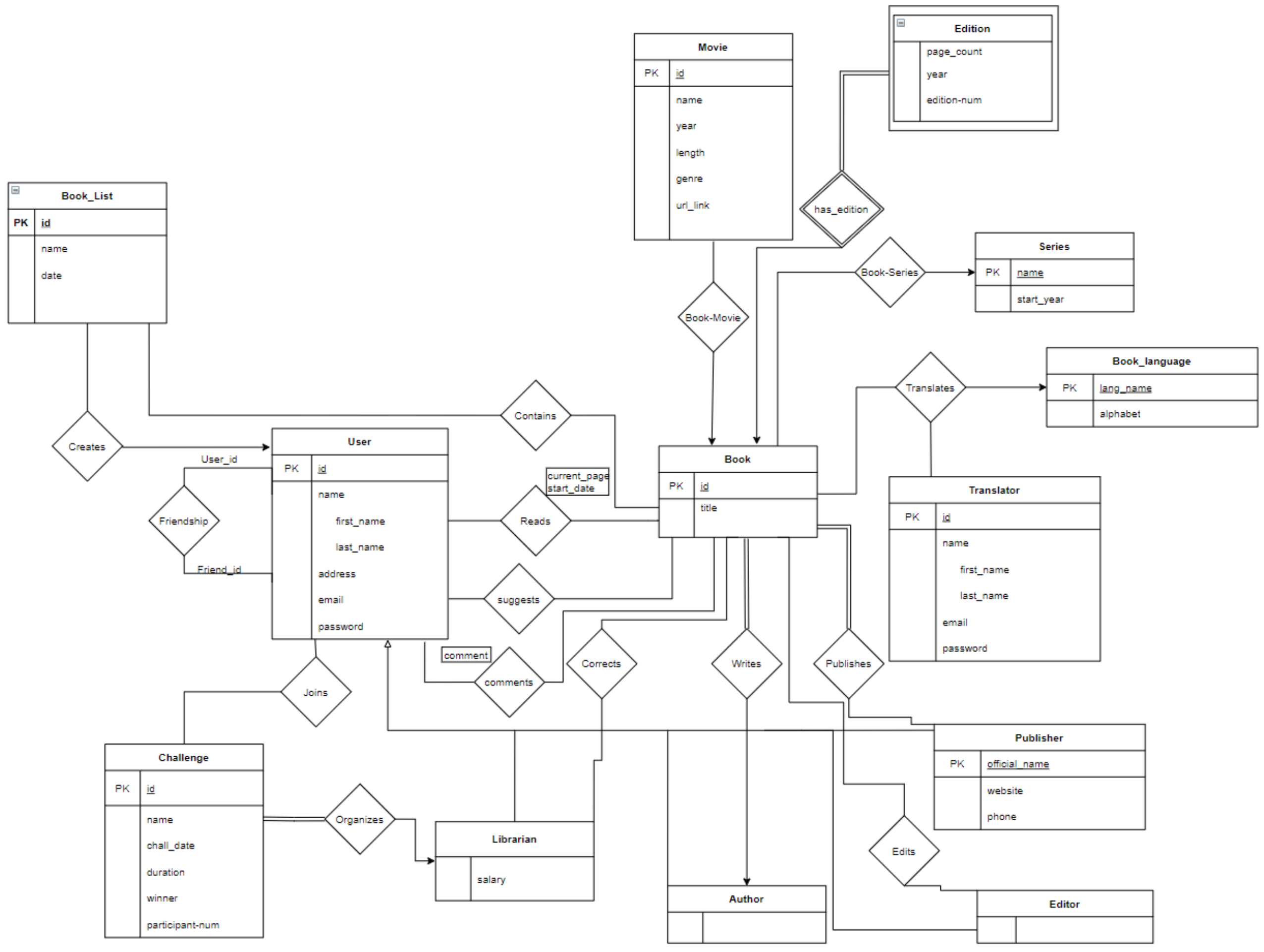
Authors can write and publish their books via publisher. They also are able to read and reply to reviews regarding their books.

Librarians are able to organize reading challenges which readers can compete in.

Editors are authorized to edit published books and its properties such as page number, edition or its movie.

Translators are able to translate books to specific languages.

Final E/R Model



Final Table Schemas

User

Relational Model:

User(id, first_name, last-name, address, e-mail, password)

Author

Relational Model:

Author(id)
FK: id references User

Publisher

Relational Model:

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

Book

Relational Model:

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

Translator

Relational Model:

Translator(id, name, e-mail, password)

Librarian

Relational Model:

Librarian(id, salary)
FK: id references User

Movie

Relational Model:

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

Edition

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

Editor

Relational Model:
Editor(id)
FK: id references User

Book_Language

Relational Model:
Book_Language(lang_name, alphabet)

Series

Relational Model:
Series(name, start_year, book_id)
FK: book_id references Book

Challange

Relational Model:
Challenge(id, name, date, duration, winner, particip-num, librar-id)

Book_list

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

Joins

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

Contains

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

Reads

Relational Model:
Reads(user_id, book_id, current_page, start_date)
FK : user_id references User
FK : book_id references Book

Suggests

Relational Model:
Suggests(user_id, book_id)
FK : user_id references User
FK : book_id references Book

Comments

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

Friendship

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

Corrects

Relational Model:
Corrects(librar_id, book_id)
FK : librar_id references Librarian
FK : book-id references Book

Edits

Relational Model:
Edits(editor_id, book_id)
FK : editor_id references Editor
FK : book_id references Book

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

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

Implementation Details

Back-end

We used java language to implement the back-end of the project, since java can be considered as our main programming language. To make the things a bit easier, we used Spring Framework, especially the annotations in the framework. For instance, “Autowired” annotation helped us on connections of the instances inside different classes. To work efficiently and for the readability of the project, we used spring-mvc (model, view and controller) as a design pattern. The code has five major parts in it which are Model, View, Controller, Dao (Data Access Object) and UserService (View is basically front-end).

Model package contains the objects and users of the project. We implemented classes such as User, Book, BookList inside this package. Model package contains the very basic java code. It only contains the main objects, their constructors and essential functions. We used these functions inside the Dao package.

Inside the Dao package, we have seperated Daos for each of the objects such as UserDao and BookDao. These Daos perform more than one database operation. Inside Daos, we implemented the executions of SQL queries. Registration of the User or addition of a book is executed inside these Daos. Add or select operations are done by the help of the simple set and get function which are implemented inside the model package.

As for the controllers, they are basically the way of connection between front-end and back-end. Inside controllers, annotations of spring framework such as “RequestMapping” and “ModelAttribute” helped us connect the ends without any problem. We needed to implement different controllers for each page since there are different data shown on different pages. For instance, an editor’s view is different than an author’s view. As a result of the difference, we implemented different controllers for each page.

UserService is implemented to use the functions inside Daos that perform query executions. We implemented a reperate package for that to make the code more readable. Since it is a different class, we did not have any confusion while adding new features and testing them.

We implemented the database on MySQL and connected to the localhost. After connecting the local host we connected to the localhost using IntelliJ's "Tool Window". Also, we declared the path of the database inside user-beans of Spring Framework.

Front-end

As mentioned in the back-end section, we used spring-mvc and we implemented the front-end components under the View package. We first implemented the front-end using HTML language. However, later during implementation we needed to shape the code a little bit for it to be suitable for the JSP (Java Server Page) format. We used JSP format to avoid implementing servlets. JSP's "tag" format allowed us to implement java code inside an HTML file. We kept our main HTML format inside the "jsp" format and the parts which execute data transfer using java code inside "tag" format.

We tried to design a simple user interface. Since the purpose of the project is working more on databases, we tried to implement a plain UI for the project. Our other purpose was to make an understandable and easy to use UI.

Advanced Database Components

Views

1) In the challenge list page that is created by librarians, the *Non_Finished_Challenges* view can be displayed by users. This view retrieves challenge names which have winner X and min. ten participants

```
CREATE VIEW Non_Finished_Challenges(challenge_name)
AS 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 );
```

2) In the authors page which is filled by already registered authors, user can view *Famous_Authors* view that retrieves author names and book counts that have 50 or more books

```
CREATE VIEW Famous_Authors(author_name, id)
AS 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) >= 50;
```

3) In the challenges page which was created by librarians. *Popular_Challenges* can be displayed by users and displays challenges to more than ten users and its winner.

```
CREATE VIEW Popular_Challenges(challenge_name)
AS 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 );
```

4) In the books page where all books displayed that are registered in the database, the user can view his uncompleted books as *Incomplete_Books* view.

```
CREATE VIEW Incomplete_Books(title)
AS 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;
```

Triggers

1) The database keeps the book count under some threshold like 100000. When one more book is added to the database, the very first book of the database is deleted to prevent undesirable growth.

```
CREATE TRIGGER keep_book_count AFTER INSERT ON Book
WHEN ( (SELECT COUNT(*)
           FROM Book) > 100000 )
BEGIN
          DELETE TOP 1
          FROM Book
END
```

Reports

1) Monthly Winner Names Report

```
SELECT winner
FROM    Challenge
WHERE winner <> null AND datediff(curdate(), date) <= 30;
```

User Manual

1. User is going to encounter a sign up/login choice page and choose his/her user type if s/he wants to sign up or press the login button if they already have an account.

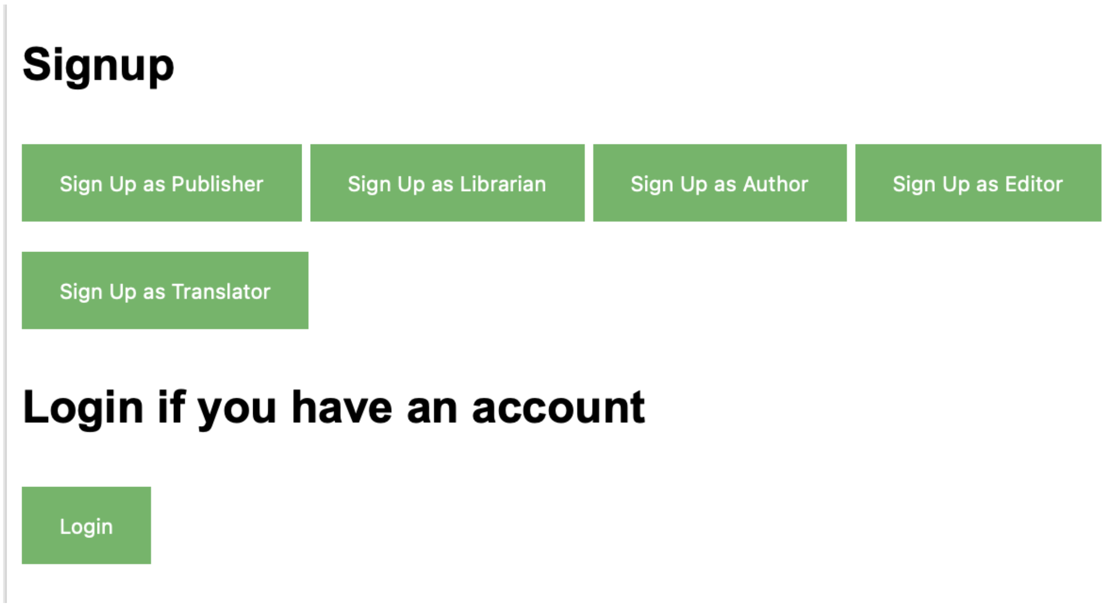
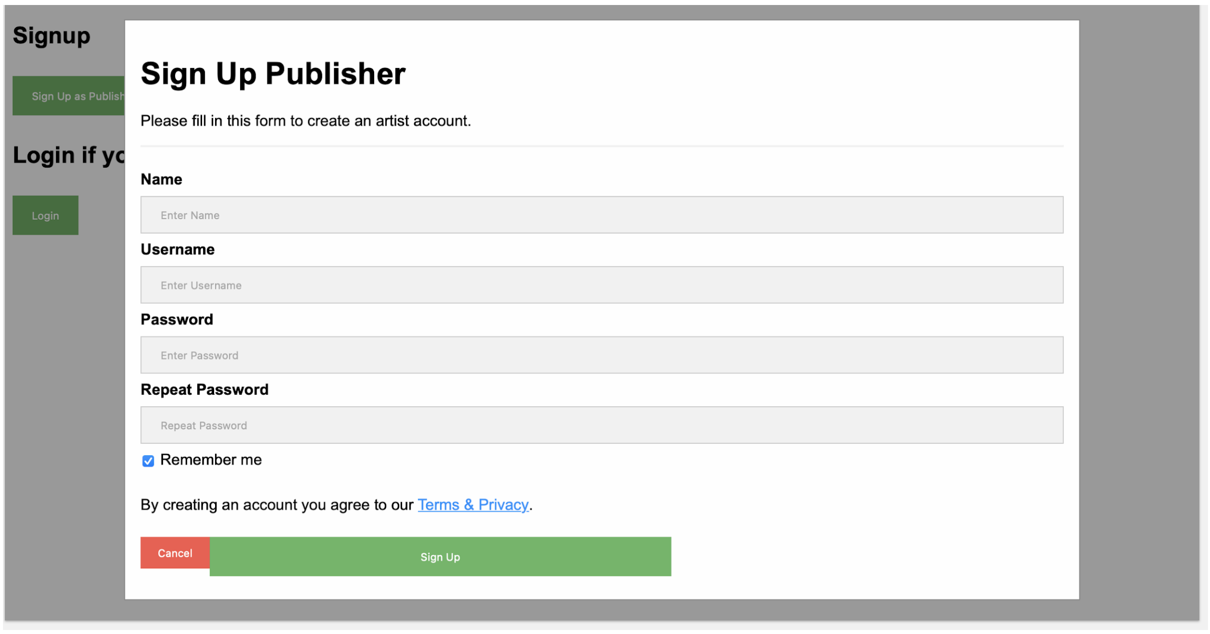


Figure 1: Sign up/Login Page

2. If one of the sign up buttons are clicked, a sign up screen will be displayed according to users choice (publisher, librarian, author, editor, translator)



Sign Up Publisher

Please fill in this form to create an artist account.

Name

Enter Name

Username

Enter Username

Password

Enter Password

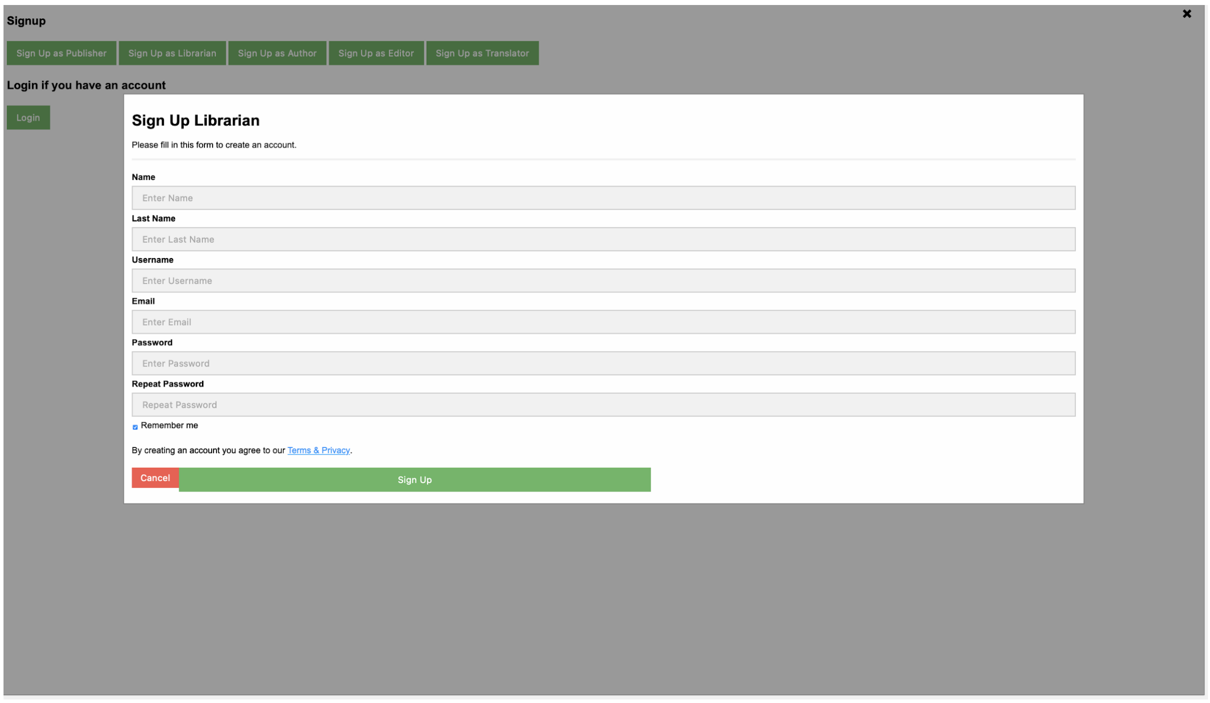
Repeat Password

Repeat Password

[✓] Remember me

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

Figure 2: Publisher 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 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

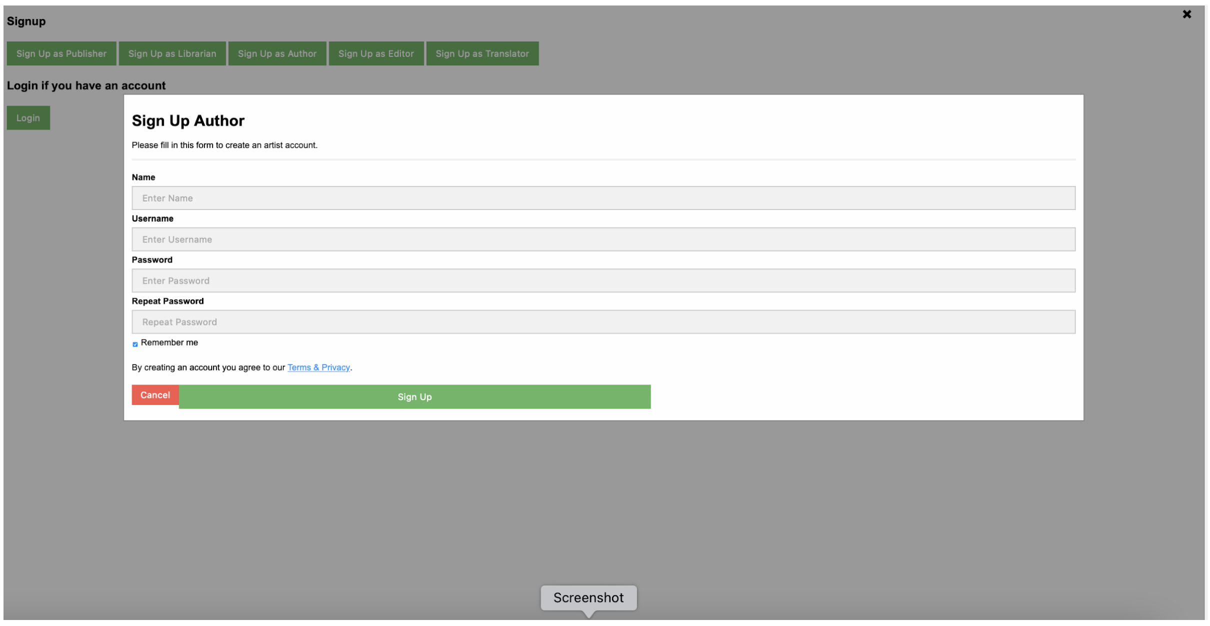
Repeat Password

Remember me

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

Cancel | Sign Up

Figure 3: Librarian 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

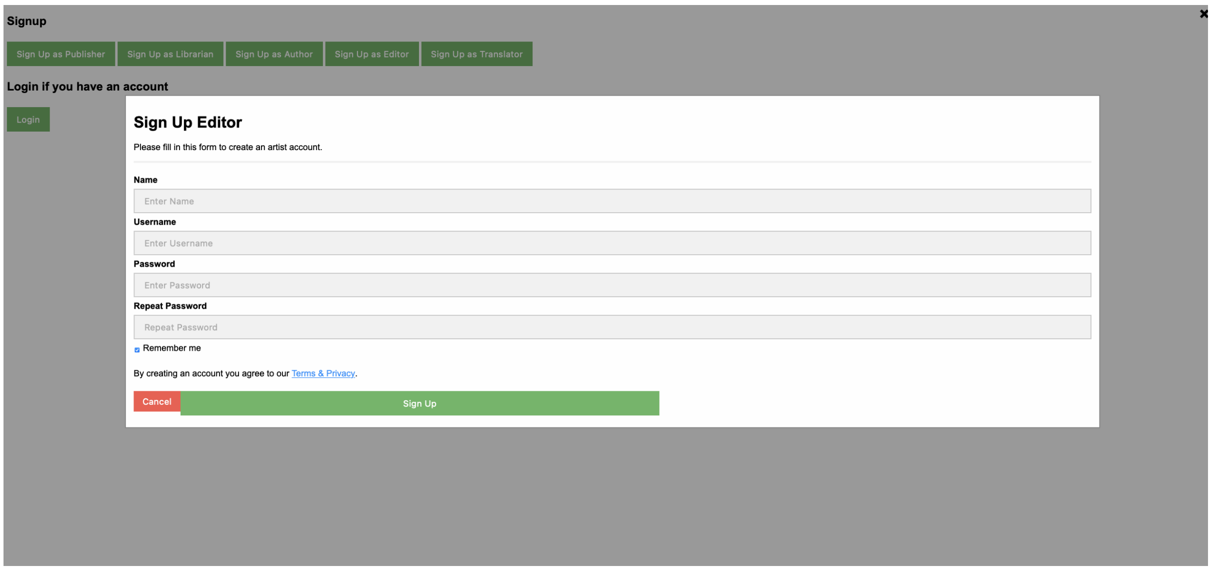
Remember me

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

Cancel | Sign Up

Screenshot

Figure 4: Author Sign up Page



Sign Up Editor

Please fill in this form to create an artist account.

Name

Enter Name

Username

Enter Username

Password

Enter Password

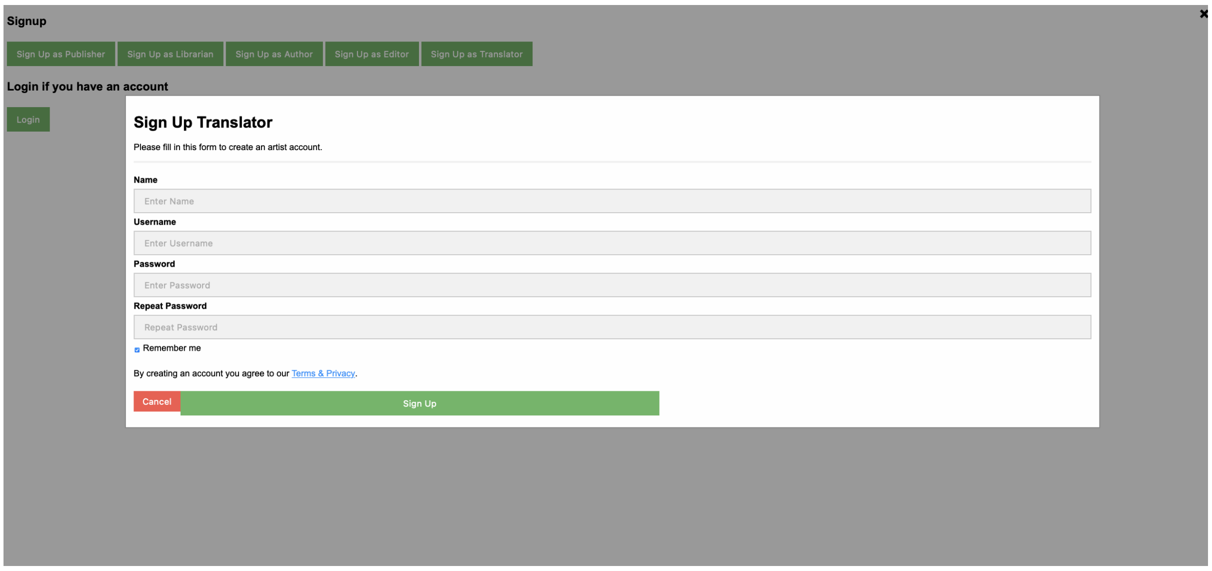
Repeat Password

Repeat Password

Remember me

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

Figure 5: Editor 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

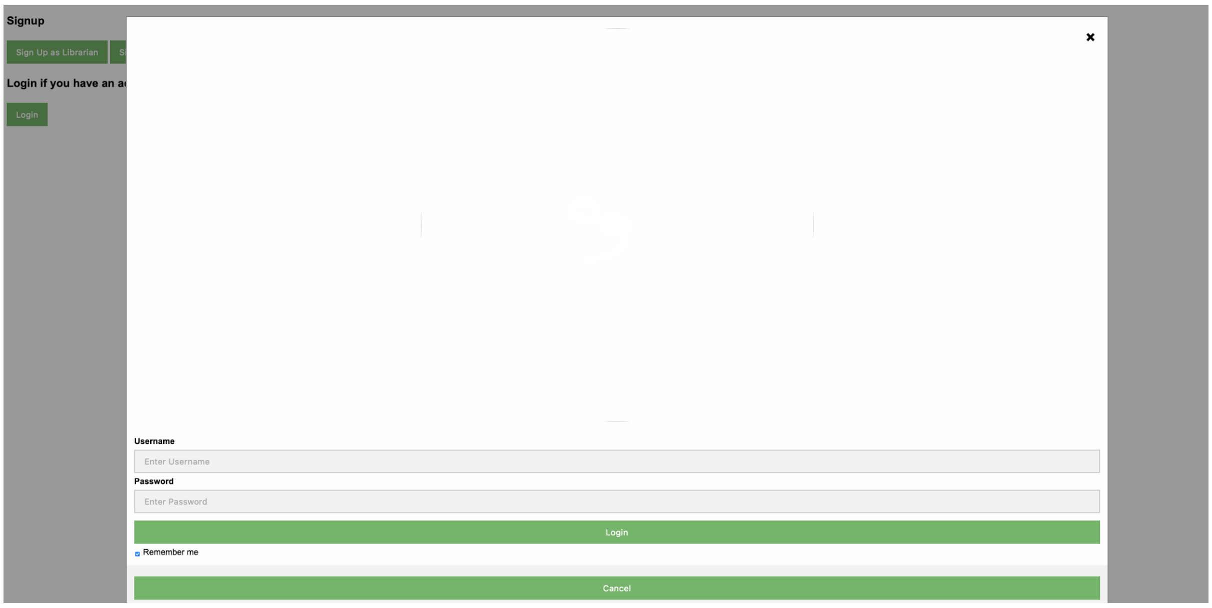
Repeat Password

Repeat Password

Remember me

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

Figure 6: Translator Sign up Page



Signup

Sign Up as Librarian

Login if you have an ac

Login

×

Username

Enter Username

Password

Enter Password

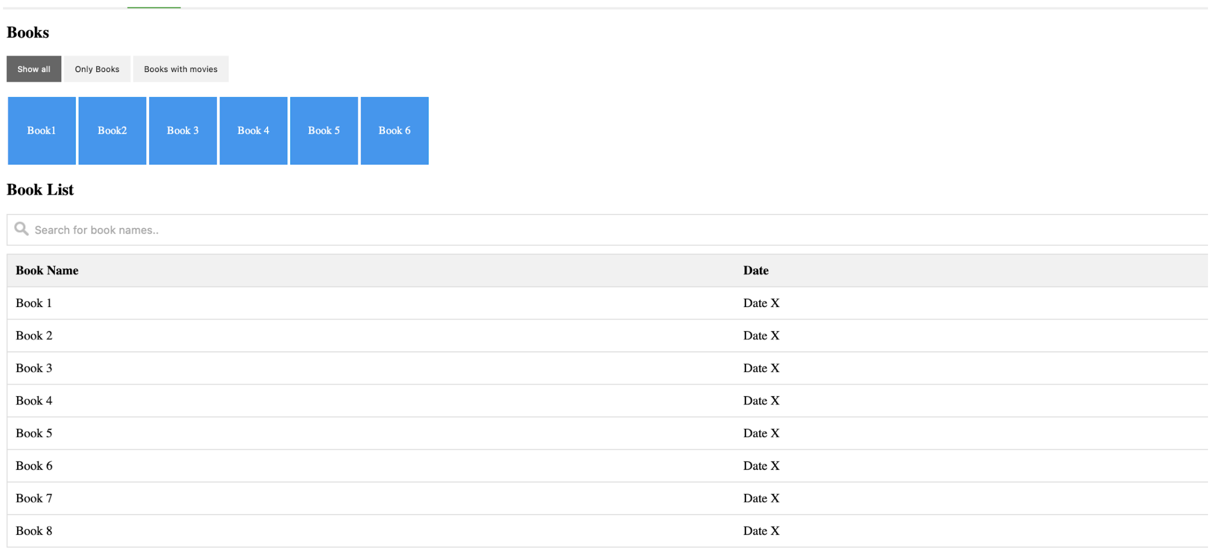
Login

Remember me

Cancel

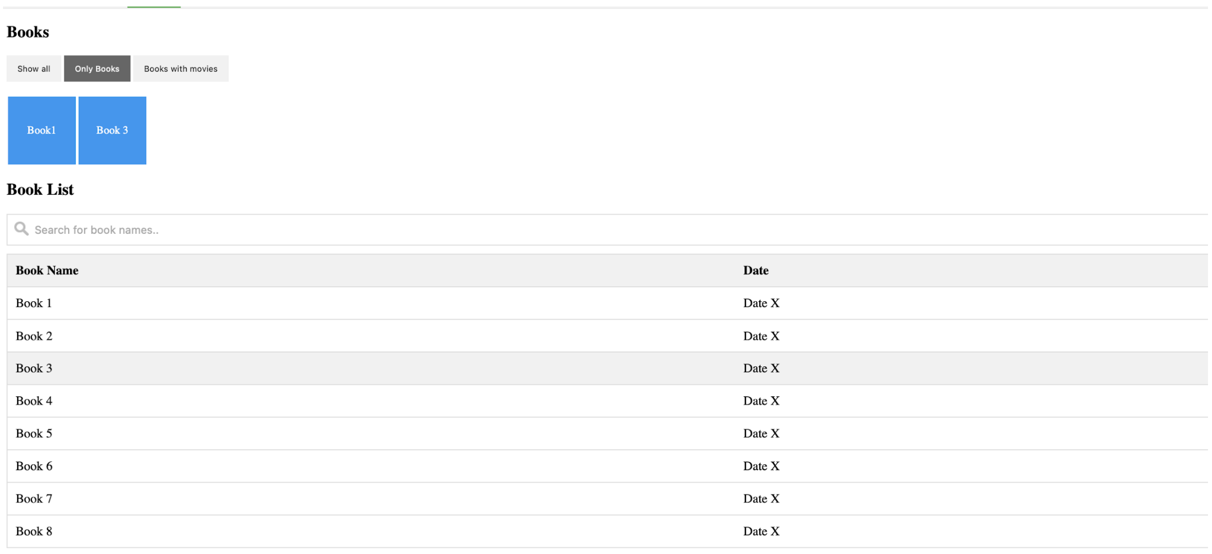
Figure 7: Login Page

3. If a user logins as a reader, s/he will display books in his/her own profile. List of books, books without movies and books that have movies can be displayed using filters.



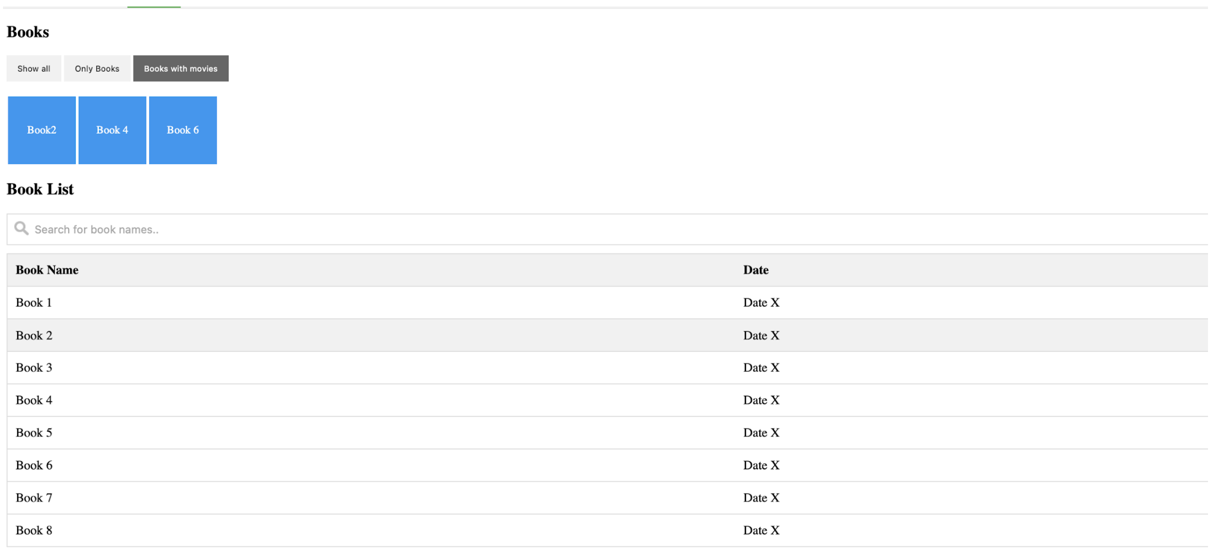
Book Name	Date
Book 1	Date X
Book 2	Date X
Book 3	Date X
Book 4	Date X
Book 5	Date X
Book 6	Date X
Book 7	Date X
Book 8	Date X

Figure 8: All Books Page



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

Figure 9: Books without movies Page


Books

Show all | Only Books | Books with movies

Book2 | Book 4 | Book 6

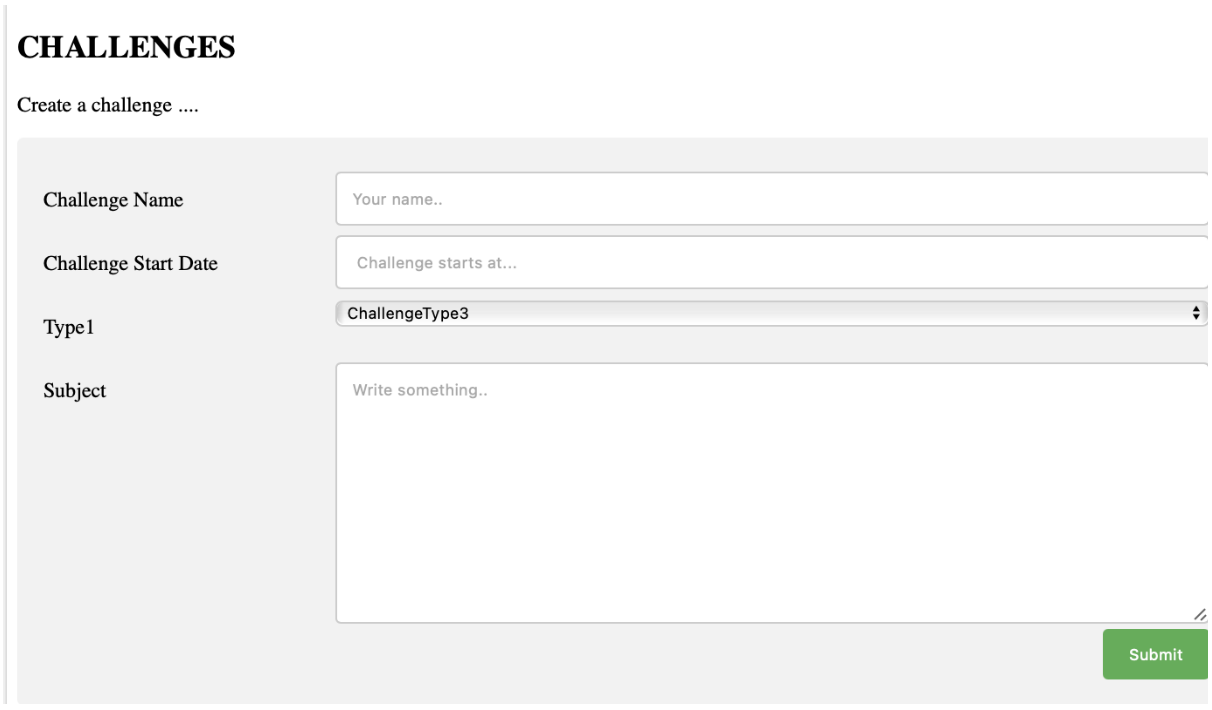
Book List

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

Figure 10: Books with movies Page

4. Librarians can arrange challenges using challenge arrangement page. Challenge name, start date, type and subject must be entered and it can be started by clicking submit button.


CHALLENGES

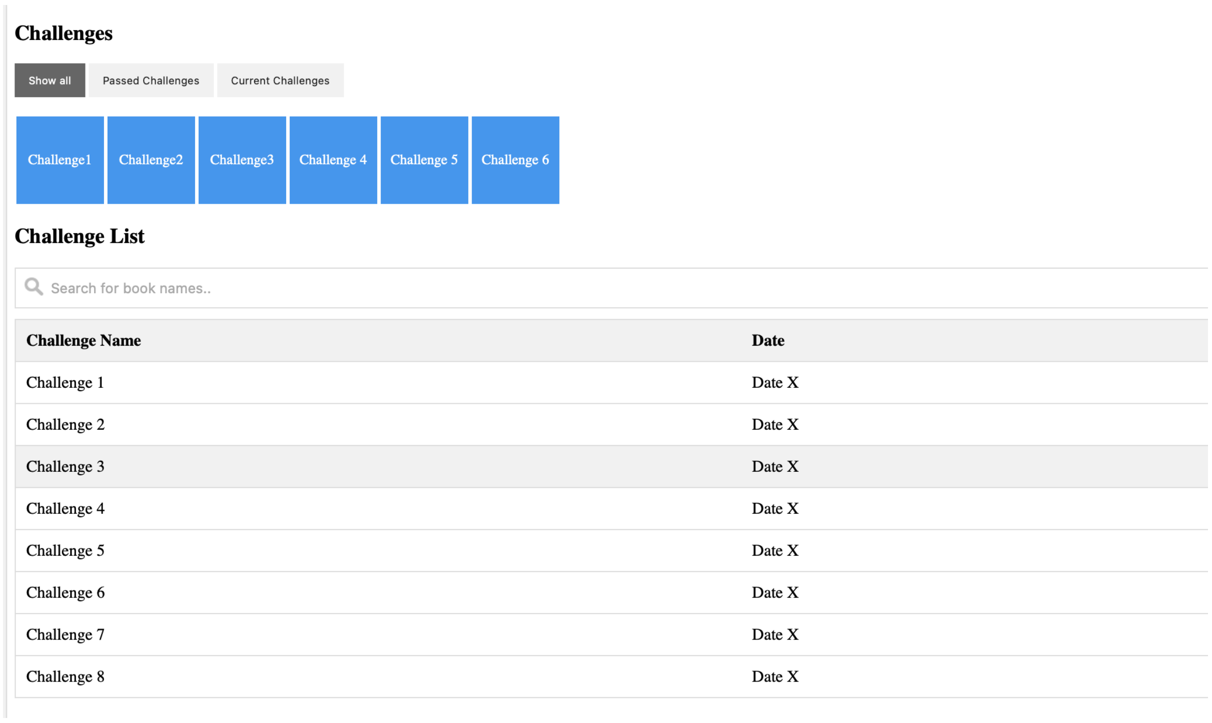
Create a challenge ....

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

Submit

Figure 11: Challenge Arrangement Page

5. Challenges can be viewed by challenge view page by all user types. Active challenges can be filtered as current challenges and past challenges.



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

Figure 12: Challenges Page

Contributions of group members

Barış Tiftik - reports, database and back-end
Ege Moroğlu - reports, Back-end and demo
Melisa Onaran - reports, user interface and front-end
Mehmet Yiğit Harlak - reports, database and back-end

Website

Code: https://github.com/yigitharlak/Digital-Reading-and-Sharing-Platform
Previous Reports: https://github.com/egemoroglu/Digital_Reading_And_Sharing_Platform