CS 353
Database Systems

Spring 2021

Design Report

Team Members

Ahmet Feyzi Halaç	21703026
Ege Şahin	21702300
Göktuğ Gürbüztürk	21702383
Zeynep Cankara	21703381

Section: 1

Group: 10

Instructor: Uğur Güdükbay

1. Introduction:

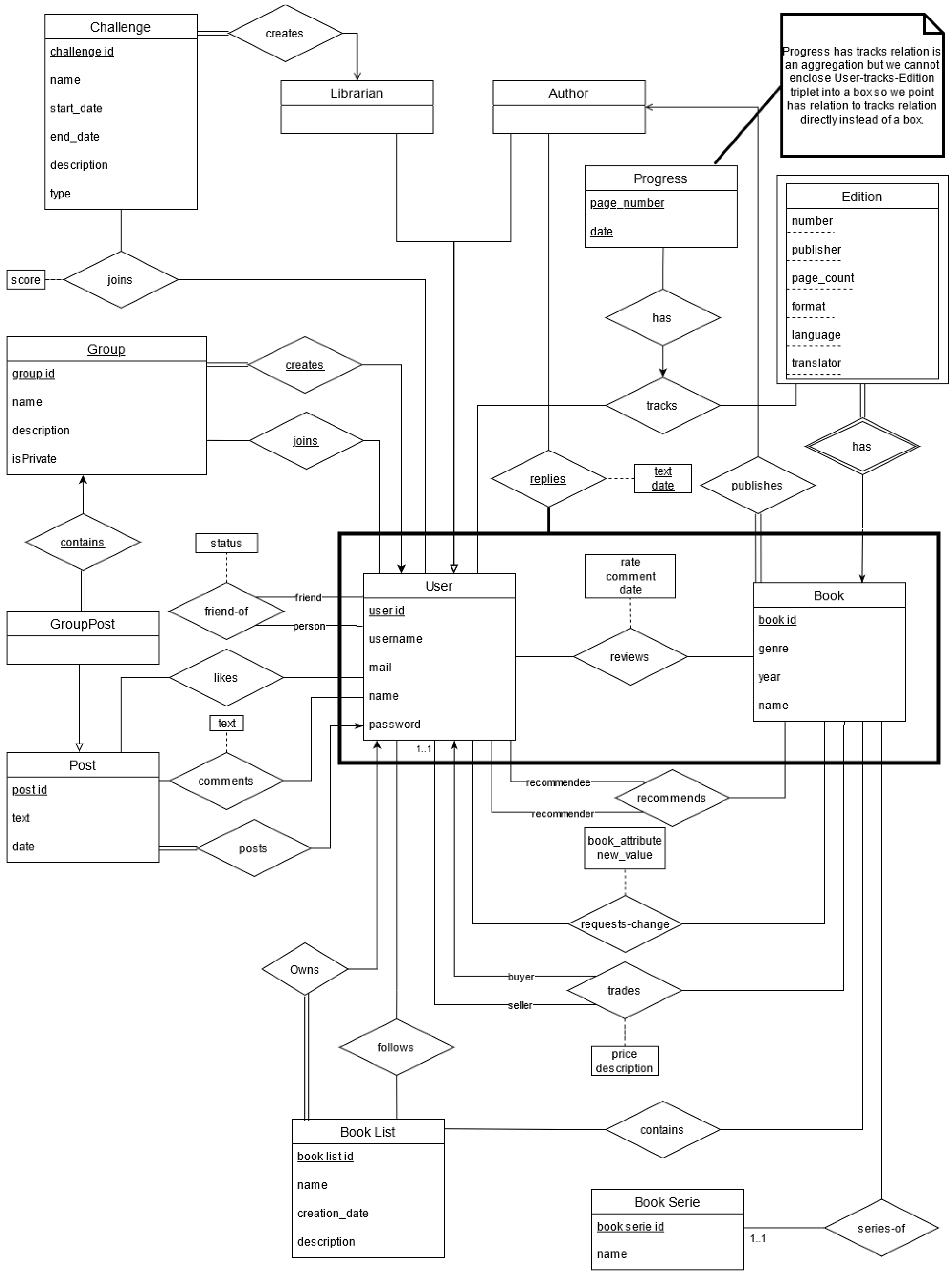
This report is a design document for the Social Cataloging Platform for Books project. This report includes a revised ER model, relational schemas, design of the user interface, and SQL statements that are used in the project.

2. Revised E/R Diagram:

The following changes have been made on the ER diagram based proposal report feedback.

- The Challenge entity has total participation to the created relationship.
- A post can be posted by only one User entity.
- A book list can be owned by only one User entity.

We included the additional functionality of book buying/selling features and the group creation community features within the same ER diagram.


Challenge
challenge id
name
start_date
end_date
description
type
creates
Librarian
Author
Progress has tracks relation is an aggregation but we cannot enclose User-tracks-Edition triplet into a box so we point has relation to tracks relation directly instead of a box.
Progress
page_number
date
Edition
number
publisher
page_count
format
language
translator
score
joins
has
Group
group id
name
description
isPrivate
creates
joins
tracks
has
replies
text
date
publishes
contains
status
friend-of
friend
person
User
user id
username
mail
name
password
rate
comment
date
reviews
Book
book id
genre
year
name
GroupPost
likes
text
comments
Post
post id
text
date
posts
1..1
recommendee
recommender
recommends
book_attribute
new_value
requests-change
buyer
seller
trades
price
description
Owns
follows
Book List
book list id
name
creation_date
description
contains
Book Serie
book serie id
name
1..1
series-of

3. Table Schemas

Challenge(challenge_id, name, start_date, end_date, description, type, creator_id)

```
CREATE TABLE Challenge(
challange_id  INT,
name          VARCHAR(16) NOT NULL,
start_date    DATE NOT NULL,
end_date      DATE NOT NULL,
description   VARCHAR(80),
type          VARCHAR(16),
creator_id    INT NOT NULL,
PRIMARY KEY(challenge_id),
FOREIGN KEY (creator_id) references User);
```

JoinsChallenge(challenge_id, user_id, score)

Foreign key: challenge_id references Challenge relation
user_id references User relation

```
CREATE TABLE JoinsChallenge(
challenge_id  INT,
user_id       INT,
score         INT,
PRIMARY KEY(user_id, challenge_id),
FOREIGN KEY(challenge_id) references Challenge,
FOREIGN KEY(user_id) references User);
```

User(user_id, username, mail, name, password, usertype)

Candidate key: username

```
CREATE TABLE User(
user_id       INT,
username      VARCHAR(16) UNIQUE,
mail          VARCHAR(32) NOT NULL,
name          VARCHAR(16) NOT NULL,
password      VARCHAR(32) NOT NULL,
usertype      VARCHAR(16) NOT NULL,
PRIMARY KEY(user_id),
CHECK (usertype IN (‘LIBRARIAN’, ‘USER’, ‘AUTHOR’));
```

Friend-of(friend_id, person_id, status)

Foreign key: friend_id references user_id from User relation
person_id references user_id from User relation

```
CREATE TABLE Friend-of(
friend_id      INT,
person_id      INT,
status         VARCHAR(16) NOT NULL,
PRIMARY KEY(friend_id, person_id),
FOREIGN KEY(friend_id) references User,
FOREIGN KEY(person_id) references User
CHECK (status IN (‘PENDING’, ‘ACCEPTED’, ‘REJECTED’));
```

Likes(post_id, user_id)

Foreign key: post_id references Post relation
user_id references User relation

```
CREATE TABLE Likes(
post_id        INT,
user_id        INT,
PRIMARY KEY(post_id, person_id),
FOREIGN KEY(post_id) references Post,
FOREIGN KEY(user_id) references User);
```

Post(post_id, text, date, writer_id)

```
CREATE TABLE Post(
post_id        INT,
text           VARCHAR(64),
date           DATE,
writer_id      INT,
PRIMARY KEY(post_id),
FOREIGN KEY(writer_id) references User);
```

Comments(post_id, user_id, text)

Foreign key: post_id references Post relation
user_id references User relation

```
CREATE TABLE Comments(
post_id        INT,
user_id        INT,
text           VARCHAR(64),
PRIMARY KEY(post_id, user_id, text),
FOREIGN KEY(post_id) references Post,
FOREIGN KEY(user_id) references User);
```

Book-List(book_list_id, name, creation_date, description, owner_id)

```
CREATE TABLE Book-List(
book_list_id        INT,
name                VARCHAR(32),
creation_date       DATE,
description         VARCHAR(64),
owner_id            INT,
PRIMARY KEY(book_list_id),
FOREIGN KEY(owner_id) references User);
```

Follows(user_id, book_list_id)

Foreign key: book_list_id references Book_List relation
user_id references User relation

```
CREATE TABLE Follows(
user_id             INT,
book_list_id        INT,
PRIMARY KEY(user_id, book_list_id),
FOREIGN KEY(book_list_id) references Book_List,
FOREIGN KEY(user_id) references User);
```

Book(book_id, genre, year, name, author_id)

Foreign key : user_id references User relation

```
CREATE TABLE Book(
book_id             INT,
genre               VARCHAR(32),
year                INT,
name                VARCHAR(64),
author_id           INT,
PRIMARY KEY(book_id),
FOREIGN KEY(author_id) references User);
```

Contains(book_list_id, book_id)

Foreign key: book_list_id references Book_List relation
book_id references Book relation

```
CREATE TABLE Contains(
book_list_id        INT,
book_id             INT,
PRIMARY KEY(book_list_id, book_id),
FOREIGN KEY(book_list_id) references Book_List,
FOREIGN KEY(book_id) references Book);
```

Tracks(user_id, book_id, number, publisher, page_count, format, language, translator)

Foreign key: book_id references Edition relation
number references Edition relation
publisher references Edition relation
page_count references Edition relation
format references Edition relation
language references Edition relation
translator references Edition relation
user_id references User relation

```
CREATE TABLE Tracks(
user_id             INT,
book_id             INT,
number              INT,
publisher           VARCHAR(64),
page_count          INT,
format              VARCHAR(64),
language            VARCHAR(64),
translator          VARCHAR(64),
PRIMARY KEY(user_id, book_id, number, publisher, page_count, format, language,
translator),
FOREIGN KEY(book_id, number, publisher, page_count, format, language, translator)
references Edition;
```

Progress(page_number, date)

```
CREATE TABLE Progress(
page_number         INT,
date                DATE,
PRIMARY KEY(page_number, date));
```

Reviews(user_id, book_id, rate, comment, date)

Foreign key: book_id references Book relation
user_id references User relation

```
CREATE TABLE Reviews(
user_id      INT,
book_id      INT,
rate         INT,
comment      VARCHAR(200),
date         DATE,
PRIMARY KEY(user_id, book_id),
FOREIGN KEY(user_id) references User,
FOREIGN KEY(book_id) references Book);
```

Recommends(recommendee_id, recommender_id, book_id)

Foreign key: recommendee_id references user_id from User relation
recommender_id references user_id from User relation
book_id references Book relation

```
CREATE TABLE Recommends(
recommendee_id    INT,
recommender_id    INT,
book_id           INT,
PRIMARY KEY(recommendee_id, recommender_id, book_id),
FOREIGN KEY(recommendee_id) references User,
FOREIGN KEY(recommender_id) references User,
FOREIGN KEY(book_id) references Book);
```

Request-Change(user_id, book_id, book_attribute, new_value)

Foreign key: book_id references Book relation
user_id references User relation

```
CREATE TABLE Request-Change(
user_id           INT,
book_id           INT,
book_attribute    VARCHAR(64),
new_value         VARCHAR(64),
PRIMARY KEY(user_id, book_id, book_attribute, new_value),
FOREIGN KEY(user_id) references User,
FOREIGN KEY(book_id) references Book,
CHECK (book_attribute IN (‘genre’, ‘year’, ‘name’));
```

Edition(book_id, number, publisher, page_count, format, language, translator)

Foreign key: book_id references Book relation

```
CREATE TABLE Edition(
book_id      INT,
number       INT,
publisher    VARCHAR(20),
page_count   INT,
format       VARCHAR(20),
language     VARCHAR(20),
translator   VARCHAR(20),
PRIMARY KEY(book_id, number, publisher, page_count, format, language, translator),
FOREIGN KEY(book_id) references Book);
```

Book-Serie(book_serie_id, name)

```
CREATE TABLE Book-Serie(
book_serie_id        INT,
name                 VARCHAR(64),
PRIMARY KEY(book_serie_id));
```

Series-of(book_id, book_serie_id)

Foreign key: book_id references Book relation
book_serie_id references Book-Serie relation

```
CREATE TABLE Series-of(
book_id              INT,
book_serie_id        INT,
PRIMARY KEY(book_id),
FOREIGN KEY(book_id) references Book,
FOREIGN KEY(book_serie_id) references Book-Serie);
```

Replies(user_id, book_id, author_id, text, date)

Foreign key: author_id references User relation
book_id references Reviews relation
user_id references Reviews relation

```
CREATE TABLE Replies(
user_id      INT,
book_id      INT,
date         DATE,
text         VARCHAR(200),
author_id    INT
PRIMARY KEY(user_id, book_id, author_id),
FOREIGN KEY(author_id) references User,
FOREIGN KEY(book_id, user_id) references Reviews);
```

Trades(offer_id, buyer_id, seller_id, price, description, book_id)
Foreign key: buyer_id references user_id from User relation
seller_id references user_id from User relation
book_id references Book relation

```
CREATE TABLE Trades(
offer_id            INT,
buyer_id            INT,
seller_id           INT,
price               REAL,
description         VARCHAR(64),
book_id             INT,
PRIMARY KEY(offer_id),
FOREIGN KEY(buyer_id) references User,
FOREIGN KEY(seller_id) references User,
FOREIGN KEY(book_id) references Book);
```

Group(group_id, name, description, isPrivate, user_id)

```
CREATE TABLE Group(
group_id      INT,
name          VARCHAR(20),
description   VARCHAR(20),
isPrivate     INT,
user_id       INT,
PRIMARY KEY (group_id)
FOREIGN KEY (user_id) references User);
```

JoinsGroup(group_id, user_id)
Foreign key: group_id references Group relation
user_id references User relation

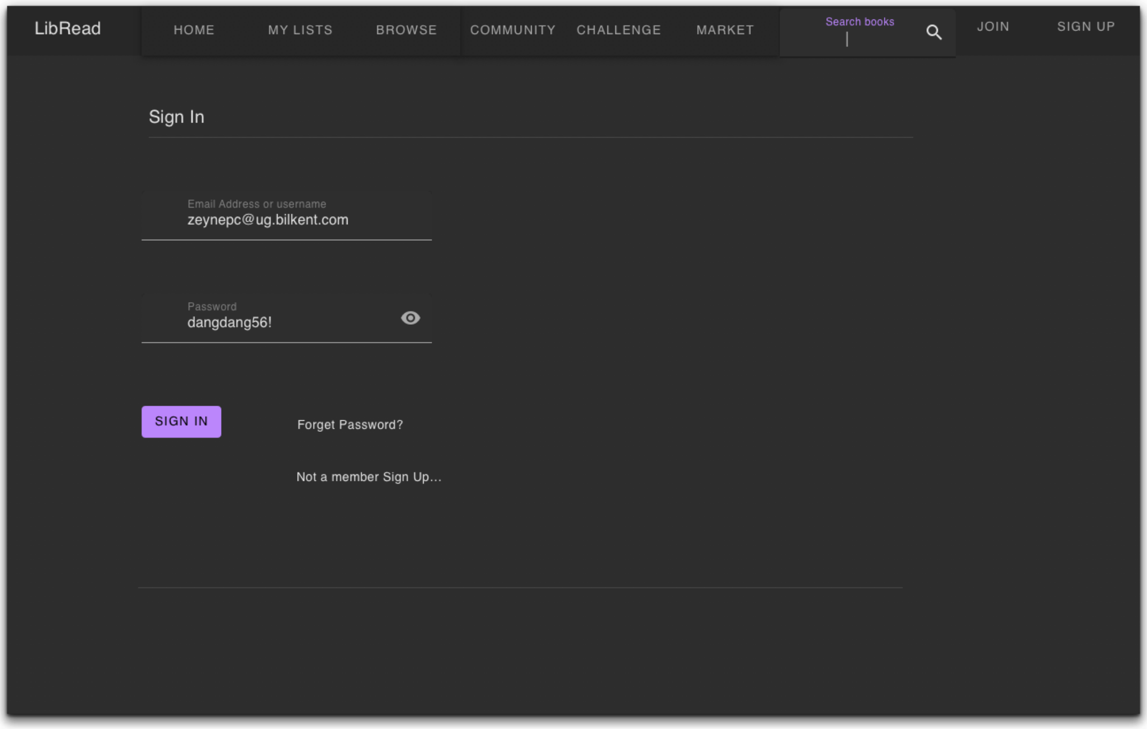
```
CREATE TABLE JoinsGroup(
group_id      INT,
user_id       INT,
PRIMARY KEY(group_id, user_id),
FOREIGN KEY(group_id) references Group,
FOREIGN KEY(user_id) references User);
```

GroupPost(<u>post_id</u>, group_id)
    Foreign key: post_id references Post relation
                 group_id references Group relation

```
CREATE TABLE GroupPost(
post_id      INT,
group_id     INT,
PRIMARY KEY(post_id),
FOREIGN KEY(post_id) references Post,
FOREIGN KEY(group_id) references Group);
```

4. UIs and SQL Queries of Functionalities

4.1 Common Functionality-1

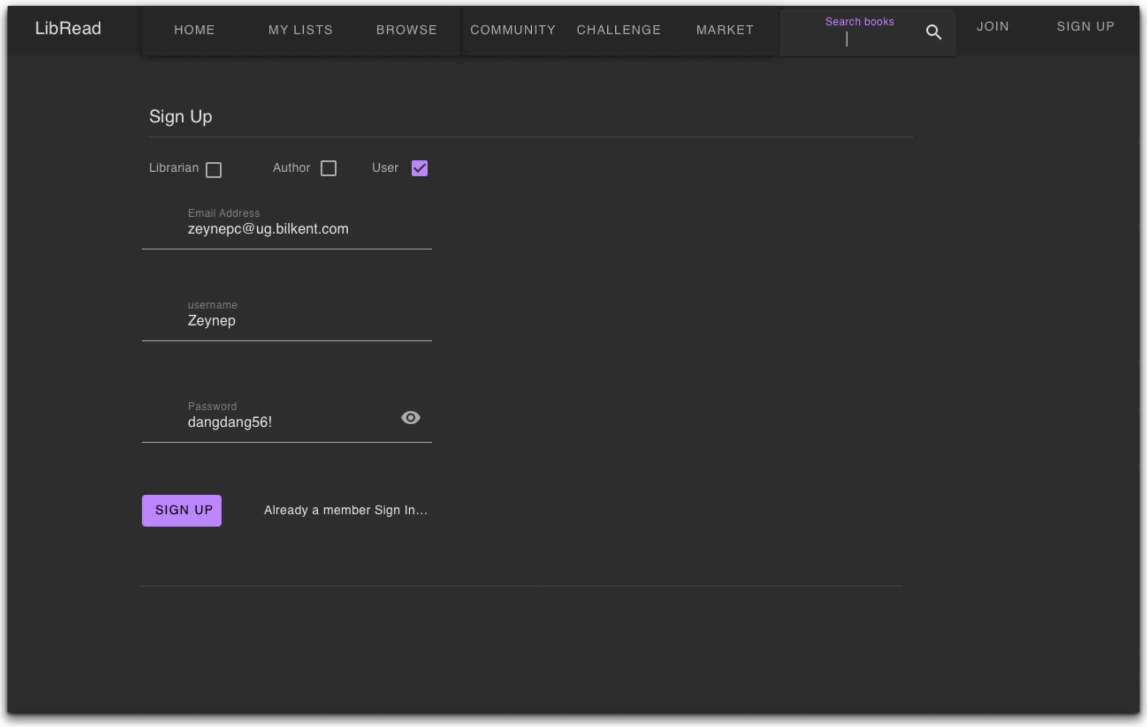


Sign In

Email Address or username
zeynepc@ug.bilkent.com

Password
dangdang56!

Login Validation
SELECT * FROM User
WHERE username = name_input AND password = password_input



Sign Up

Librarian ☐ Author ☐ User ☑

Email Address
zeynepc@ug.bilkent.com

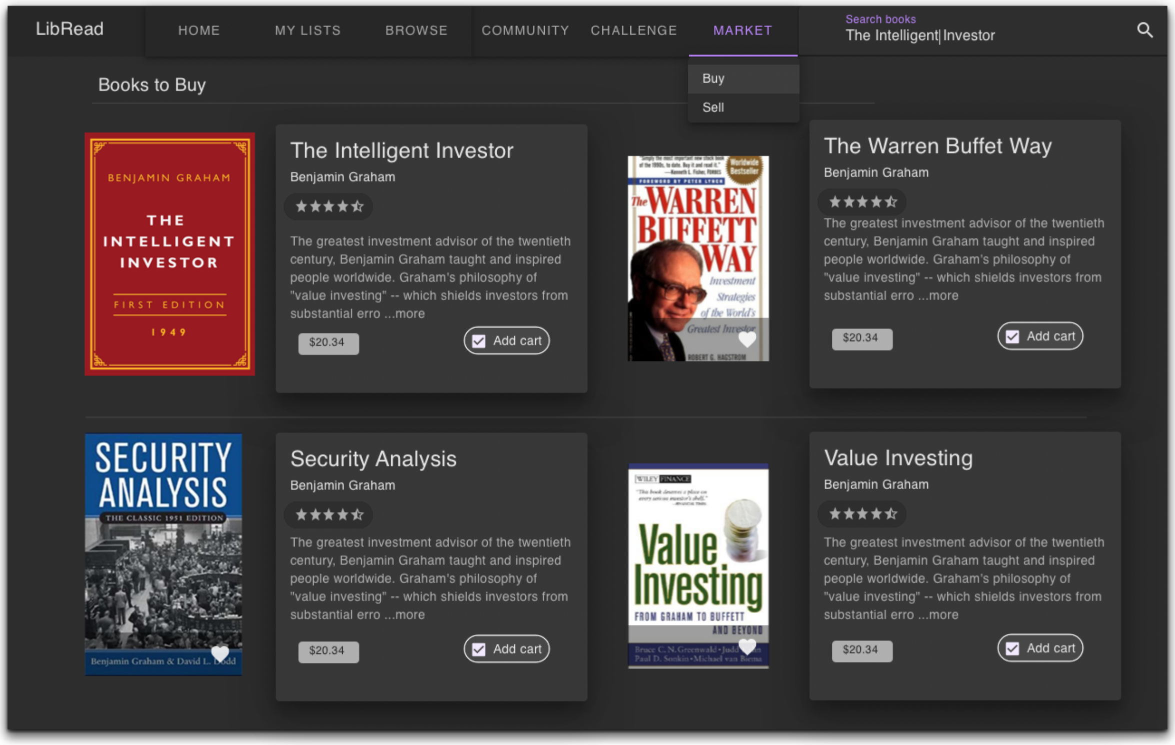
username
Zeynep

Password
dangdang56!

Create New User:
Get all IDs to create new unique ID:
SELECT user_id FROM User
Create user:
INSERT INTO User VALUES
(newID, username_input, mail_input, name_input, password_input, specified_user_type)

4.2 Common Functionality-2 (Additional Functionalities)

4.2.1 Buy/Sell Books



The Intelligent Investor

Books to Buy

The Intelligent Investor
Benjamin Graham
The greatest investment advisor of the twentieth century, Benjamin Graham taught and inspired people worldwide. Graham's philosophy of "value investing" -- which shields investors from substantial erro ...more
$20.34

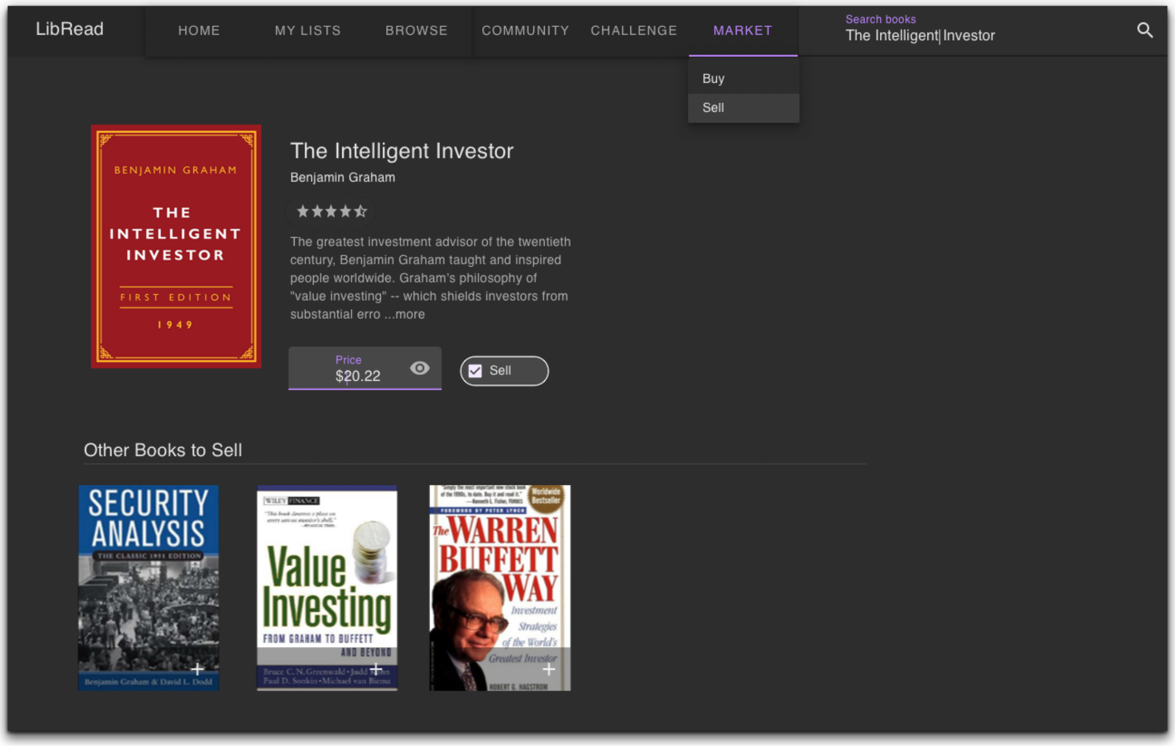
The Warren Buffet Way
Benjamin Graham
The greatest investment advisor of the twentieth century, Benjamin Graham taught and inspired people worldwide. Graham's philosophy of "value investing" -- which shields investors from substantial erro ...more
$20.34

Security Analysis
Benjamin Graham
The greatest investment advisor of the twentieth century, Benjamin Graham taught and inspired people worldwide. Graham's philosophy of "value investing" -- which shields investors from substantial erro ...more
$20.34

Value Investing
Benjamin Graham
The greatest investment advisor of the twentieth century, Benjamin Graham taught and inspired people worldwide. Graham's philosophy of "value investing" -- which shields investors from substantial erro ...more
$20.34

List all offers for a book
SELECT * FROM Trades
WHERE book_id = specified_books_id AND buyer_id IS NULL

Buy a book
UPDATE Trades
SET buyer_id = current_users_id
WHERE offer_id = specified_offer_id



The Intelligent Investor

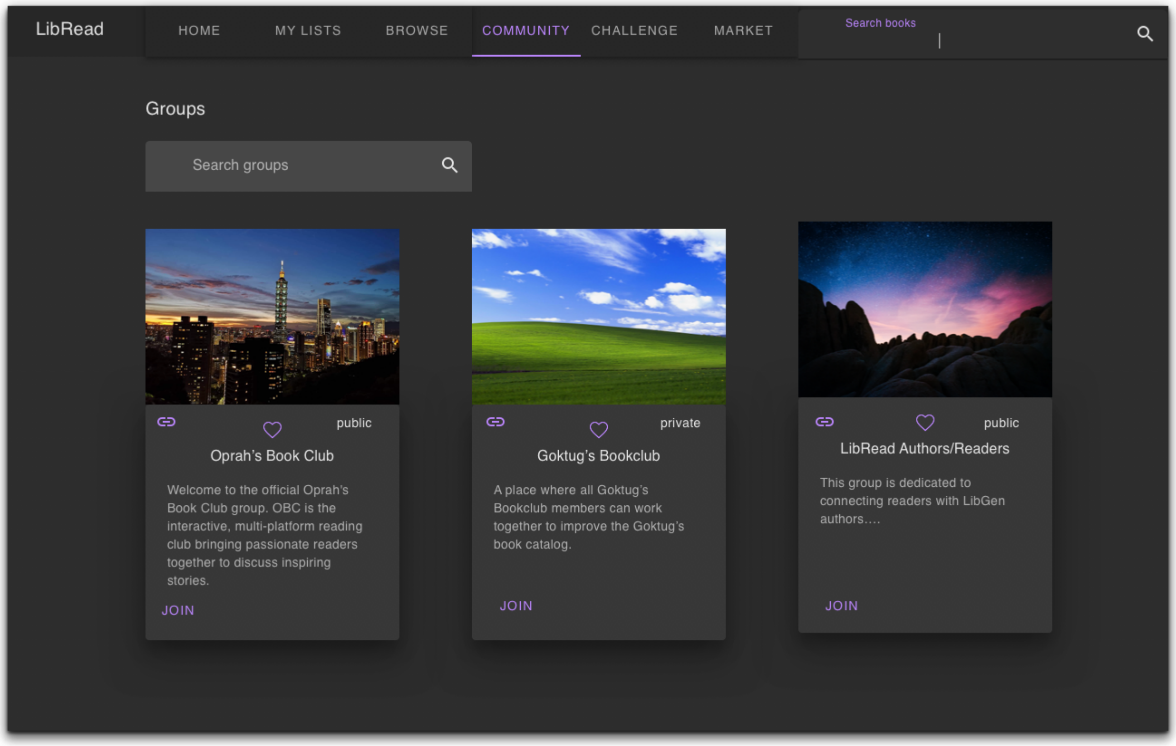
Benjamin Graham

The greatest investment advisor of the twentieth century, Benjamin Graham taught and inspired people worldwide. Graham's philosophy of "value investing" -- which shields investors from substantial erro ...more

Sell a book

INSERT INTO Trades VALUES
(newID, NULL, current_users_id, specified_price, description, specified_book_id)

4.2.2 Groups



Oprah's Book Club

Welcome to the official Oprah's Book Club group. OBC is the interactive, multi-platform reading club bringing passionate readers together to discuss inspiring stories.

Goktug's Bookclub

A place where all Goktug's Bookclub members can work together to improve the Goktug's book catalog.

LibRead Authors/Readers

This group is dedicated to connecting readers with LibGen authors....

Create group
```
INSERT INTO Group VALUES
(newID, specified_name, description, isPrivate, current_users_id)
```

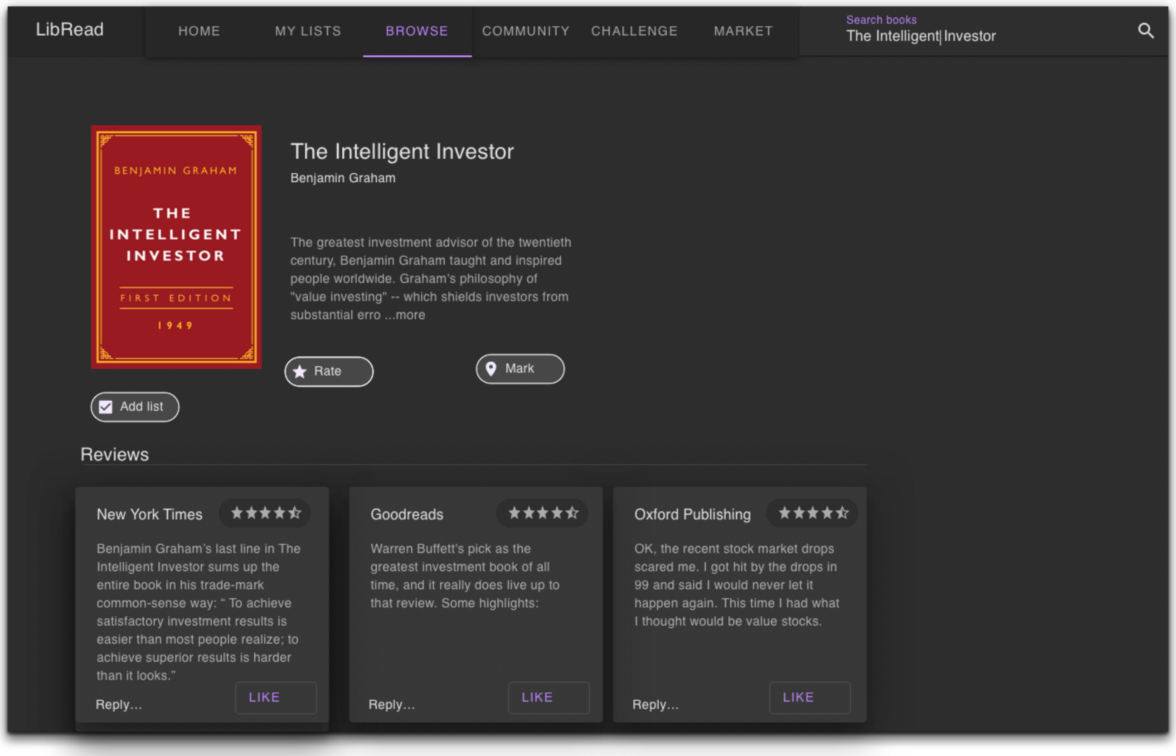
Search groups
```
SELECT * FROM Group
WHERE name LIKE ‘%searched_name%’
```

Join a group
```
INSERT INTO JoinsGroup VALUES
(specified_groups_id, current_users_id)
```

Create new post in a group
```
INSERT INTO Post VALUES
(newID, specified_text, current_date)
INSERT INTO GroupPost VALUES
(newID, specified_group_id)
```

List all posts of a group
```
SELECT text, date FROM
(SELECT post_id FROM GroupPost WHERE group_id = specified_group_id)
NATURAL JOIN Post
```

4.3 Track a book progress



The Intelligent Investor
Benjamin Graham

The greatest investment advisor of the twentieth century, Benjamin Graham taught and inspired people worldwide. Graham's philosophy of "value investing" -- which shields investors from substantial erro ...more

Reviews

New York Times ★★★★☆
Benjamin Graham's last line in The Intelligent Investor sums up the entire book in his trade-mark common-sense way: " To achieve satisfactory investment results is easier than most people realize; to achieve superior results is harder than it looks."

Goodreads ★★★★☆
Warren Buffett's pick as the greatest investment book of all time, and it really does live up to that review. Some highlights:

Oxford Publishing ★★★★☆
OK, the recent stock market drops scared me. I got hit by the drops in 99 and said I would never let it happen again. This time I had what I thought would be value stocks.

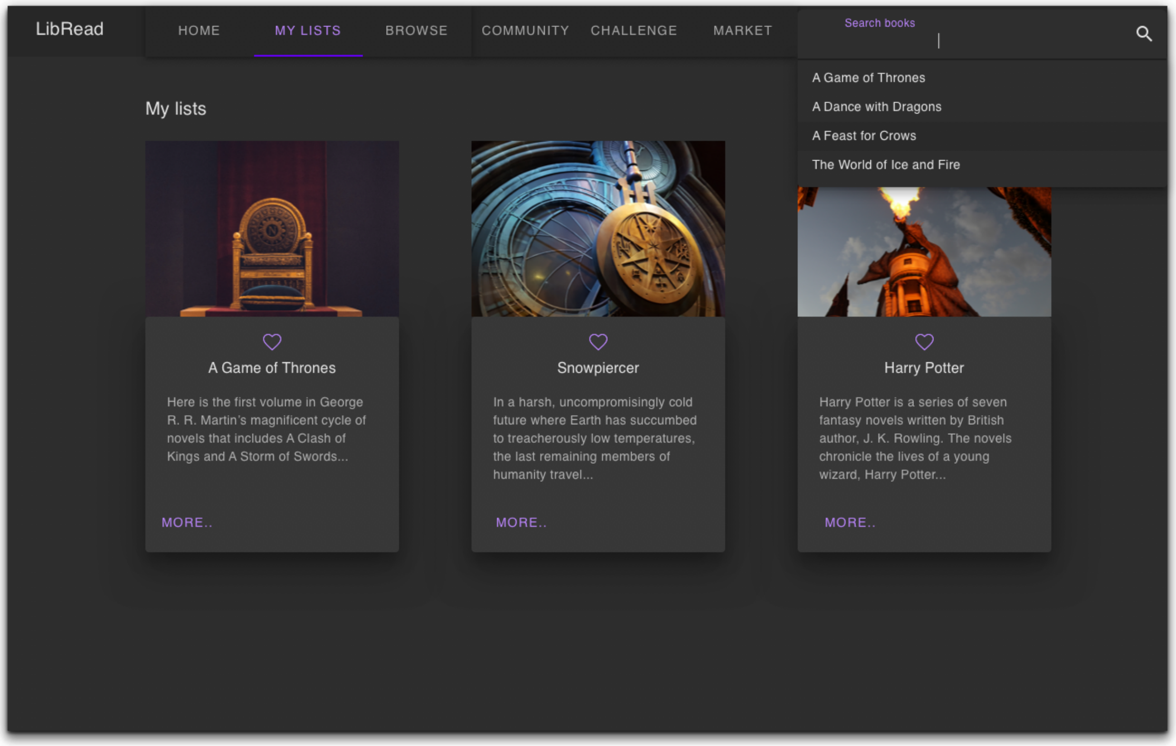
List all available books and apply filters (genre, author, keyword)
SELECT * FROM Book
WHERE genre like '%specified keyword%' or
  author like '%specified keyword%'

Get all editions of a book:
SELECT * FROM Book NATURAL JOIN Edition
WHERE book_id = specified_id

Select a book (and the edition) and start tracking
INSERT INTO Tracks VALUES
(user_id, book_id, number, publisher, page_count, format, language, translator)

Add progress to a book track
INSERT INTO Progress VALUES
(user_id, book_id, number, publisher, page_count, format, language, translator, page_progress, date)

Get all progress information for a track
SELECT page_progress, date FROM Progress
WHERE (user_id, book_id, number, publisher, page_count, format, language, translator, page_progress, date) = (specified_user_id, specified_book_id, specified_number, specified_publisher, specified_page_count, specified_format, specified_language, specified_translator)



A Game of Thrones

Here is the first volume in George R. R. Martin's magnificent cycle of novels that includes A Clash of Kings and A Storm of Swords...

Snowpiercer

In a harsh, uncompromisingly cold future where Earth has succumbed to treacherously low temperatures, the last remaining members of humanity travel...

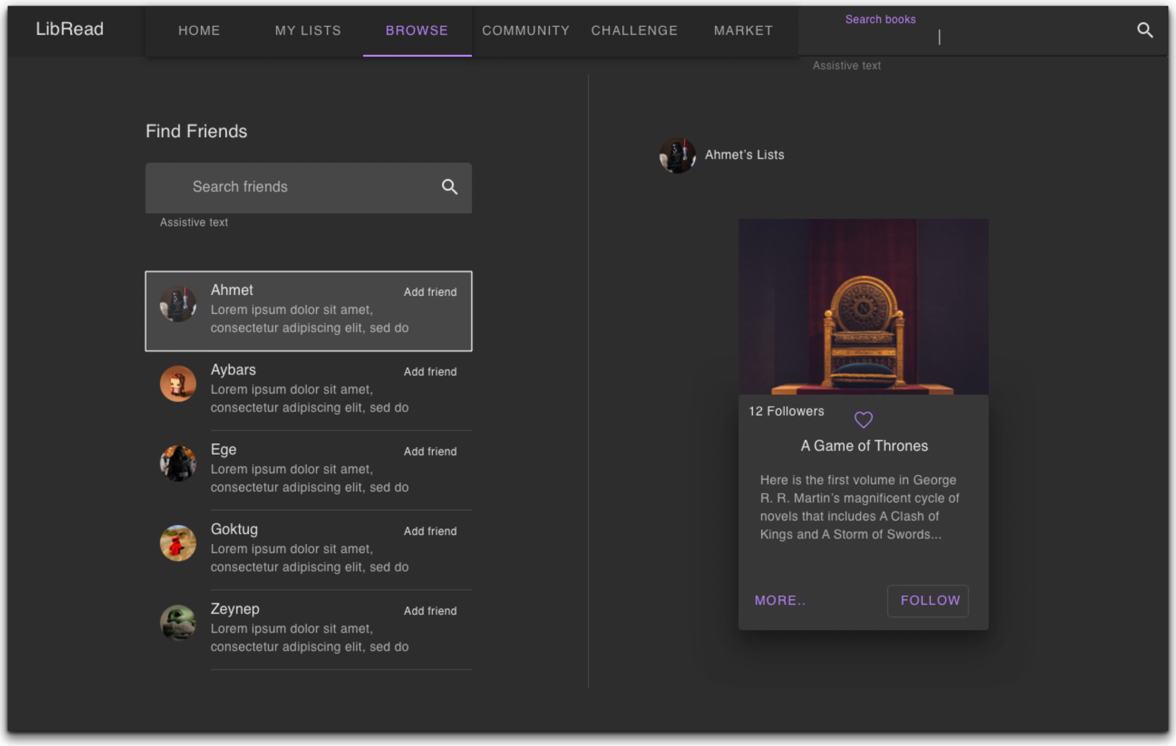
Harry Potter

Harry Potter is a series of seven fantasy novels written by British author, J. K. Rowling. The novels chronicle the lives of a young wizard, Harry Potter...

Get all booklist IDs to create new unique ID
SELECT book_list_id FROM Book-List

Create book list
INSERT INTO Book-List VALUES
(newID, specified_name, current_date, description)

Add book to book-list
INSERT INTO Contains VALUES
(specified_booklist_id, book_id)


LibRead
HOME MY LISTS BROWSE COMMUNITY CHALLENGE MARKET
Search books
Assistive text

Find Friends
Search friends
Assistive text

Ahmet — Add friend
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do

Aybars — Add friend
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do

Ege — Add friend
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do

Goktug — Add friend
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do

Zeynep — Add friend
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do

Ahmet's Lists

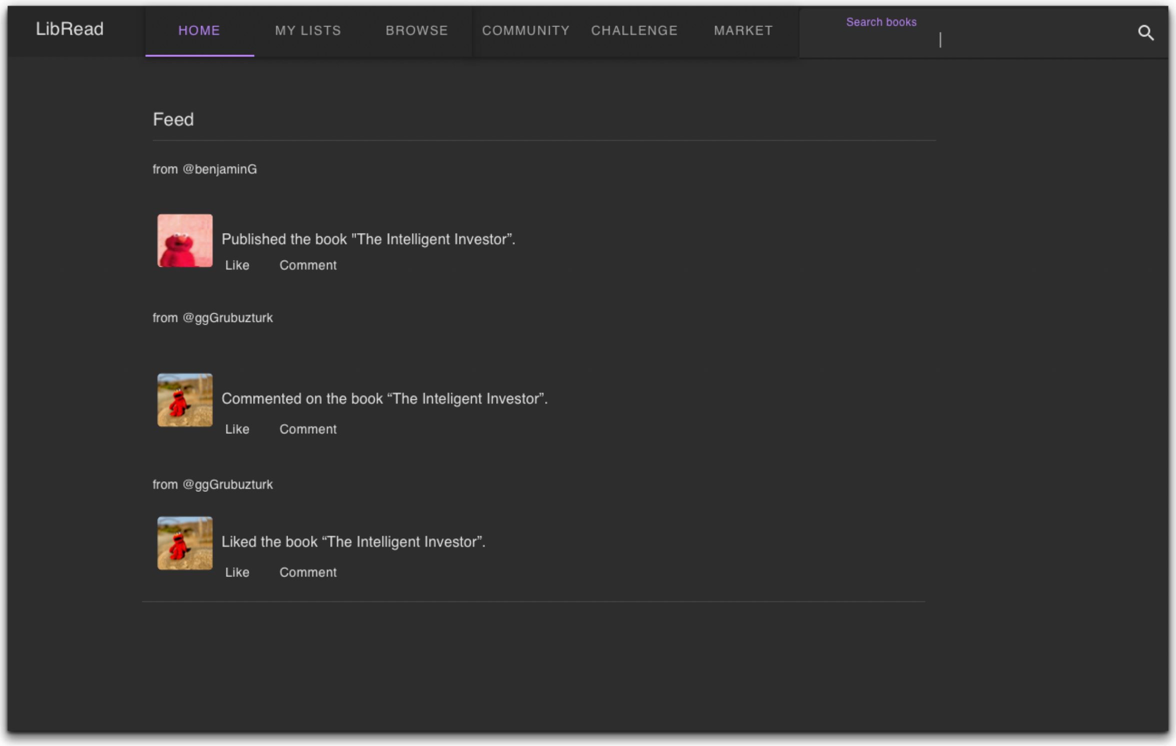
12 Followers
A Game of Thrones
Here is the first volume in George R. R. Martin's magnificent cycle of novels that includes A Clash of Kings and A Storm of Swords...
MORE.. FOLLOW

List all users
SELECT * FROM User

Send friend request
INSERT INTO Friend-of VALUES
(selected_users_id, current_user_id, 'PENDING')

List all friend requests
SELECT * FROM Friend-of, User
WHERE person_id = current_users_id
AND status = 'PENDING' AND user_id = person_id

Accept friend request
UPDATE Friend-of
SET status = 'ACCEPTED'
WHERE person_id = current_users_id AND friend_id = specified_friends_id



Feed

from @benjaminG

Published the book "The Intelligent Investor".

from @ggGrubuzturk

Commented on the book "The Inteligent Investor".

from @ggGrubuzturk

Liked the book "The Intelligent Investor".

Like their posts
INSERT INTO Likes VALUES
(specified_post_id, current_users_id)

Comment on their posts
INSERT INTO Comments VALUES
(specified_post_id, current_users_id, comment)

Get when specified book is read by user

```
SELECT date FROM Tracks NATURAL JOIN Progress
WHERE user_id = current_users_id
        AND book_id = specified_book_id
        AND page_count = page_progress
```

Recommend a book to friends
INSERT INTO Recommends VALUES
(recommended_users_id, current_users_id, book_id)