BİLKENT UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CS353

Project Design Report

Group 11

Aybars Altınışık 21601054

Bulut Gözübüyük 21702771

Denizhan Kemeröz 21703471

Muharrem Berk Yıldız 21802492

Contents

1. Revised E/R Model

- Weak entity set Edition added.
- Changed joins to joins_challenge and cardinality to many to one and has additional attribute book_read.
- Admin table is deleted so the whole system is managed by the librarian, so Librarian does not have is_verified.
- Recommend relation added.
- Rates relation renamed as review and has three additional attributes (reply, comment, rating).
- Additional attributes (request_msg, approved) added and type is removed from the requests relation.
- Progress table has additional attributes called page_number ,date and progress amount is deleted.
- Book_list table has additional attributes called book_count.
- Some attributes from the Book table are moved to the Edition table.
- Changed mark_progress cardinality.
- post_commnet has one to many cardinality.
- Post table has a text attribute.
- Progress_comment to Progress cardinality changed to many to 1
- friend_of relation has accepted attribute

- progress to mark_progress cardinality changed from 1 to total many

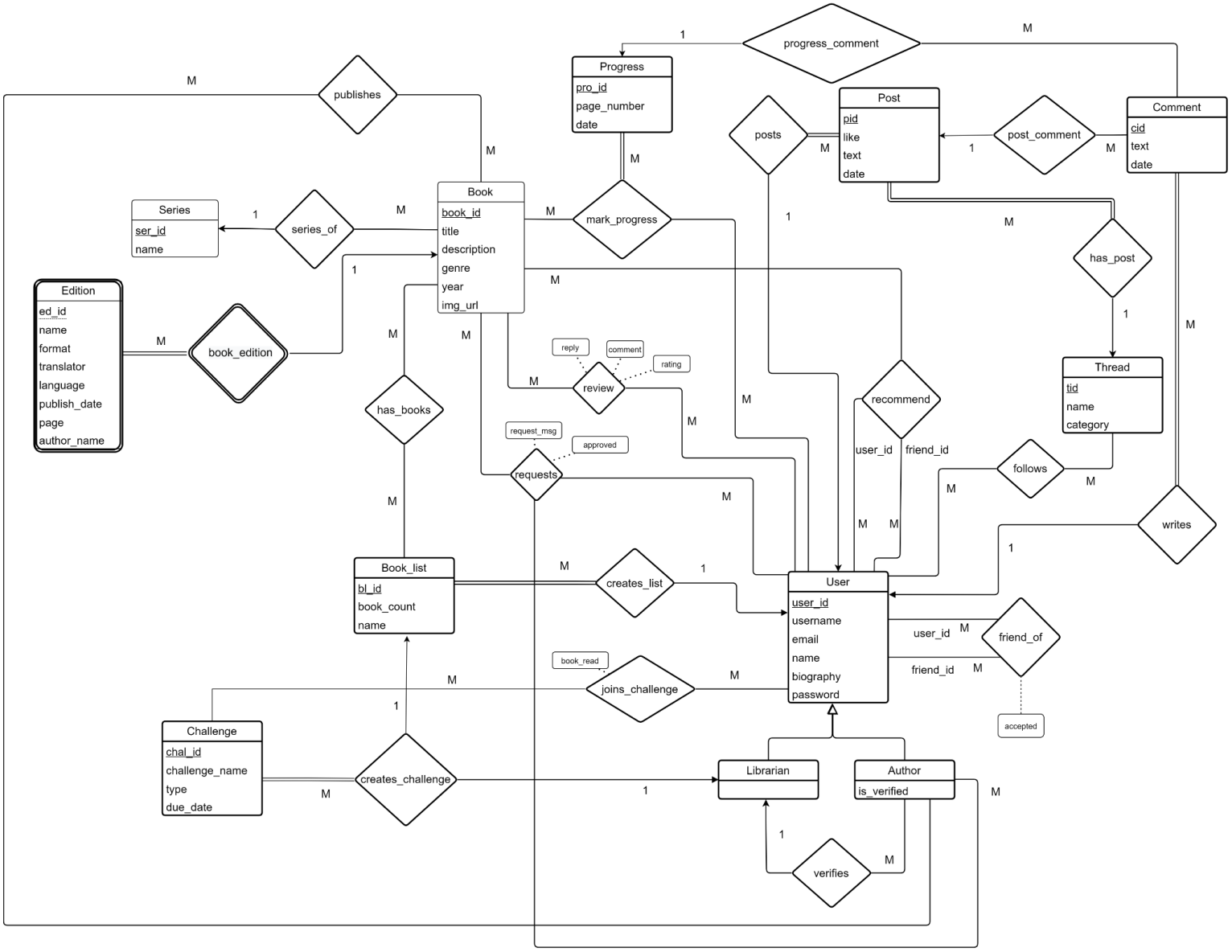

Figure 1: E/R Diagram

Above is the Entity Relationship Diagram of our project. It also can be seen as full size from our web page which can be found in section 4.

2. Table Schemas

2.1. User

Relational Model

user(<u>user_id</u>, user_name, email, name, biography, password)

Primary and Foreign Keys

Primary key(s): user_id

Foreign key(s): ----

Table Declaration:

```
CREATE TABLE User(
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    user_name VARCHAR(32) NOT NULL UNIQUE,
    email VARCHAR(32) NOT NULL UNIQUE,
    name VARCHAR(32) NOT NULL,
    biography VARCHAR(32) DEFAULT NULL,
    password VARCHAR(32) NOT NULL
);
```

2.2. friend_of

Relational Model

friend_of(<u>user_id</u>, <u>friend_id,</u> accepted)

Primary and Foreign Keys

Primary key(s): user_id, friend_id

Foreign key(s):

- user_id - Foreign key to User table
- friend_id - Foreign key to User table

Table Declaration:

```
CREATE TABLE friend_of(
    user_id INT PRIMARY KEY,
    FOREIGN KEY(user_id) REFERENCES User(user_id),
    ON DELETE CASCADE,
    friend_id INT PRIMARY KEY,
    FOREIGN KEY(friend_id) REFERENCES User(user_id)
    ON DELETE CASCADE,
    accepted BIT DEFAULT 0
);
```

2.3. Librarian

Relational Model

librarian(<u>user_id</u>)

Primary and Foreign Keys

Primary key(s): user_id

Foreign key(s):

user_id - Foreign key to User table

Table Declaration:

```
CREATE TABLE Librarian(
    user_id INT PRIMARY KEY,
    FOREIGN KEY(user_id) REFERENCES User(user_id),
    ON DELETE CASCADE
);
```

2.4. Author

Relational Model

author(<u>user_id</u>, is_verified, verifier_id)

Primary and Foreign Keys

Primary key(s): user_id

Foreign key(s):

user_id - Foreign key to User table

verifier_id - Foreign key to Librarian table

Table Declaration:

```
CREATE TABLE Author(
    user_id INT PRIMARY KEY,
    FOREIGN KEY(user_id) REFERENCES User(user_id),
    ON DELETE CASCADE,
    is_verified BIT DEFAULT 0,
    verfier_id INT NOT NULL
);
```

2.5. publishes

Relational Model

publishes(<u>author_id</u>, <u>book_id</u>)

Primary and Foreign Keys

Primary key(s): author_id, book_id

Foreign key(s):

author_id - Foreign key to Author table (user_id)

book_id - Foreign key to Book table

Table Declaration:

```
CREATE TABLE publishes(
    author_id INT PRIMARY KEY,
    FOREIGN KEY(author_id) REFERENCES Author(user_id),
    ON DELETE CASCADE,
    book_id INT PRIMARY KEY,
    FOREIGN KEY(book_id) REFERENCES Book(book_id)
    ON DELETE CASCADE,
);
```

2.6. Book

Relational Model

book(<u>book_id</u>, title, description, genre, year, img_url)

Primary and Foreign Keys

Primary key(s): book_id

Foreign key(s): ----

Table Declaration:

```
CREATE TABLE Book(
    book_id INT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(32) NOT NULL,
    description VARCHAR(256) NOT NULL,
    genre VARCHAR(32) NOT NULL,
    year INT NOT NULL,
    img_url VARCHAR(64) DEFAULT NULL
);
```

2.7. Edition

Relational Model

edition(book_id, ed_id, name, format, translator, language, publish_date, page, author_name)

Primary and Foreign Keys

Primary key(s): book_id, ed_id

Foreign key(s):

book_id - Foreign key to Book table

Table Declaration:

```
CREATE TABLE Edition(
    book_id INT PRIMARY KEY,
    FOREIGN KEY(book_id) REFERENCES Book(book_id),
    ON DELETE CASCADE,
    ed_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(32) NOT NULL,
    format VARCHAR(32) NOT NULL,
    translator VARCHAR(32) DEFAULT NULL,
    language VARCHAR(32) NOT NULL,
    publish_date DATE DEFAULT CURRENT_TIMESTAMP,
    page INT NOT NULL,
    author_name VARCHAR(32) NOT NULL
);
```

2.8. review

Relational Model

review(user_id, book_id, rating, comment, reply)

Primary and Foreign Keys

Primary key(s): user_id, book_id

Foreign key(s):

user_id - Foreign key to User table

book_id - Foreign key to Book table

Table Declaration:

```
CREATE TABLE review(
    user_id INT PRIMARY KEY,
    FOREIGN KEY(user_id) REFERENCES User(user_id),
    ON DELETE CASCADE,
    book_id INT PRIMARY KEY
    FOREIGN KEY(book_id ) REFERENCES Book(book_id ),
    ON DELETE CASCADE,
    rating INT NOT NULL,
    comment VARCHAR(256) NOT NULL,
    reply VARCHAR(256) DEFAULT NULL
);
```

2.9. Series

Relational Model

series(ser_id, name)

Primary and Foreign Keys

Primary key(s): ser_id

Foreign key(s): ----

Table Declaration:

```
CREATE TABLE Series(
    ser_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(32) NOT NULL
);
```

2.10. series_of

Relational Model

series_of(<u>book_id</u>, ser_id)

Primary and Foreign Keys

Primary key(s): book_id

Foreign key(s):

series_id - Foreign key to Series table

book_id - Foreign key to Book table

Table Declaration:

```
CREATE TABLE series_of(
    book_id INT PRIMARY KEY,
    FOREIGN KEY(book_id ) REFERENCES Book(book_id ),
    ON DELETE CASCADE,
    FOREIGN KEY(ser_id ) REFERENCES Book(ser_id ),
    ON DELETE CASCADE
);
```

2.11. Progress

Relational Model

progress(pro_id, page_number, date)

Primary and Foreign Keys

Primary key(s): pro_id

Foreign key(s): ----

Table Declaration:

```
CREATE TABLE Progress(
        pro_id INT PRIMARY KEY AUTO_INCREMENT,
        page_number INT NOT NULL,
        date DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

2.12. progress_comment

Relational Model

progress_comment(cid, pro_id)

Primary and Foreign Keys

Primary key(s): cid

Foreign key(s):

pro_id - Foreign key to Progress table

cid - Foreign key to Comment table

Table Declaration:

```
CREATE TABLE progress_comment(
        cid INT PRIMARY KEY,
        FOREIGN KEY(cid ) REFERENCES Comment(cid ),
```

```
        ON DELETE CASCADE,
        FOREIGN KEY(pro_id ) REFERENCES Progress(pro_id ),
        ON DELETE CASCADE
);
```

2.13. mark_progress

Relational Model

mark_progress(<u>user_id</u>, <u>book_id</u>, <u>pro_id</u>)

Primary and Foreign Keys

Primary key(s): user_id, book_id, pro_id

Foreign key(s):

- user_id - Foreign key to User table
- book_id - Foreign key to Book table
- pro_id - Foreign key to Progress table

Table Declaration:

```
CREATE TABLE mark_progress(
        user_id INT PRIMARY KEY,
        FOREIGN KEY(user_id) REFERENCES User(user_id),
        ON DELETE CASCADE,
        book_id INT PRIMARY KEY,
        FOREIGN KEY(book_id) REFERENCES Book(book_id),
        ON DELETE CASCADE,
        pro_id INT PRIMARY KEY,
        FOREIGN KEY(pro_id) REFERENCES Progress(pro_id),
        ON DELETE CASCADE
);
```

2.14. Comment

Relational Model

comment(cid, text, date, user_id)

Primary and Foreign Keys

Primary key(s): cid

Foreign key(s):

user_id- Foreign key to User table

Table Declaration:

```
CREATE TABLE Comment(
    cid INT PRIMARY KEY AUTO_INCREMENT,
    text VARCHAR(256) NOT NULL,
    date DATE DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY(user_id) REFERENCES User(user_id),
    ON DELETE CASCADE
);
```

2.15. recommend

Relational Model

recommend(user_id, friend_id, book_id)

Primary and Foreign Keys

Primary key(s): user_id, friend_id, book_id

Foreign key(s):

user_id - Foreign key to User table

friend_id - Foreign key to User table (user_id)

book_id - Foreign key to Book table

Table Declaration:

```
CREATE TABLE recommend(
    user_id INT PRIMARY KEY,
    FOREIGN KEY(user_id) REFERENCES User(user_id),
    ON DELETE CASCADE,
    book_id INT PRIMARY KEY,
    FOREIGN KEY(friend_id) REFERENCES User(user_id),
    ON DELETE CASCADE,
    friend_id INT PRIMARY KEY,
    FOREIGN KEY(book_id) REFERENCES Book(book_id),
    ON DELETE CASCADE
);
```

2.16. joins_challenge

Relational Model

joins_challenge(<u>chal_id</u>, <u>user_id,</u> book_read)

Primary and Foreign Keys

Primary key(s): chal_id, user_id

Foreign key(s):

- user_id - Foreign key to User table
- chal_id - Foreign key to Challenge table

Table Declaration:

```
CREATE TABLE joins_challenge(
    user_id INT PRIMARY KEY,
    FOREIGN KEY(user_id) REFERENCES User(user_id),
```

```
        ON DELETE CASCADE,
        chal_id INT PRIMARY KEY,
        FOREIGN KEY(chal_id) REFERENCES Challenge(chal_id),
        ON DELETE CASCADE,
        book_read INT DEFAULT 0
);
```

2.17. Challenge

Relational Model

challenge(chal_id, challenge_name, due_date, bl_id, librarian_id)

Primary and Foreign Keys

Primary key(s): chal_id

Foreign key(s):

bl_id - Foreign key to Book_list table

librarian_id - Foreign key to Librarian table (user_id)

Table Declaration:

```
CREATE TABLE Challenge(
        chal_id INT PRIMARY KEY AUTO_INCREMENT,
        bl_id INT PRIMARY KEY,
        FOREIGN KEY(bl_id) REFERENCES Book_list(bl_id),
        ON DELETE CASCADE,
        challenge_name VARCHAR(32) NOT NULL,
        due_date DATE NOT NULL,
        FOREIGN KEY(librarian_id) REFERENCES Librarian(user_id),
        ON DELETE CASCADE
);
```

2.18. requests

Relational Model

requests(<u>book_id</u>, <u>user_id</u>, <u>librarian_id</u>, request_msg, approved)

Primary and Foreign Keys

Primary key(s): book_id, user_id, librarian_id

Foreign key(s):

user_id - Foreign key to User table

book_id - Foreign key to Book table

librarian_id - Foreign key to Librarian table (user_id)

Table Declaration:

```
CREATE TABLE requests(
    book_id INT PRIMARY KEY,
    FOREIGN KEY(book_id) REFERENCES Book(book_id),
    ON DELETE CASCADE,
    user_id INT PRIMARY KEY,
    FOREIGN KEY(user_id) REFERENCES User(user_id),
    ON DELETE CASCADE,
    librarian_id INT PRIMARY KEY,
    FOREIGN KEY(librarian_id) REFERENCES Librarian(user_id),
    ON DELETE CASCADE,
    request_msg VARCHAR(256) NOT NULL,
    approved BIT DEFAULT 0
);
```

2.19. has_books

Relational Model

has_books(book_id, bl_id)

Primary and Foreign Keys

Primary key(s): book_id, bl_id

Foreign key(s):

book_id - Foreign key to Book table

bl_id - Foreign key to Book_list table

Table Declaration:

```
CREATE TABLE has_books(
    book_id INT PRIMARY KEY,
    FOREIGN KEY(book_id) REFERENCES Book(book_id),
    bl_id INT PRIMARY KEY,
    FOREIGN KEY(bl_id) REFERENCES Book_list(bl_id)
);
```

2.20. Book_list

Relational Model

Book_list(bl_id, name, book_count)

Primary and Foreign Keys

Primary key(s): bl_id

Foreign key(s): ----

Table Declaration:

```
CREATE TABLE Book_list(
    bl_id INT PRIMARY KEY AUTO_INCREMENT,
```

```
    name VARCHAR(32) NOT NULL,
    book_count INT DEFAULT 0
);
```

2.21. Thread

Relational Model

thread(tid, name, category)

Primary and Foreign Keys

Primary key(s): tid

Foreign key(s): ----

Table Declaration:

```
CREATE TABLE Thread(
    tid INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(32) NOT NULL,
    category  VARCHAR(32) NOT NULL
);
```

2.22. follows

Relational Model

follows(user_id, tid)

Primary and Foreign Keys

Primary key(s): user_id, tid

Foreign key(s):

user_id - Foreign key to User table

tid - Foreign key to Thread table

Table Declaration:

```
CREATE TABLE follows(
    user_id INT PRIMARY KEY,
    FOREIGN KEY(user_id) REFERENCES User(user_id),
    ON DELETE CASCADE,
    tid INT PRIMARY KEY,
    FOREIGN KEY(tid) REFERENCES Thread(tid),
    ON DELETE CASCADE
);
```

2.23. Post

Relational Model

post(<u>pid</u>, like, text, date, user_id, t_id)

Primary and Foreign Keys

Primary key(s): pid

Foreign key(s):

- user_id - Foreign key to User table
- t_id - Foreign key to Thread table

Table Declaration:

```
CREATE TABLE Post(
    pid INT PRIMARY KEY AUTO_INCREMENT,
    like INT DEFAULT 0,
    text VARCHAR(256) DEFAULT NULL,
    date DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY(user_id) REFERENCES User(user_id),
```

```
        ON DELETE CASCADE,
        FOREIGN KEY(tid) REFERENCES Thread(tid),
        ON DELETE CASCADE
);
```

2.24. post_comment

Relational Model

post_comment(cid, pid)

Primary and Foreign Keys

Primary key(s): cid

Foreign key(s):

pid - Foreign key to post table

cid - Foreign key to comment table

Table Declaration:

```
CREATE TABLE post_comment(
        cid INT PRIMARY KEY,
        FOREIGN KEY(cid) REFERENCES Comment(cid),
        ON DELETE CASCADE,
        FOREIGN KEY(pid) REFERENCES Post(pid),
        ON DELETE CASCADE
);
```

3. User Interface Design and SQL Statements

3.1. Home Page

figure 1: Home page

3.2. Login Page

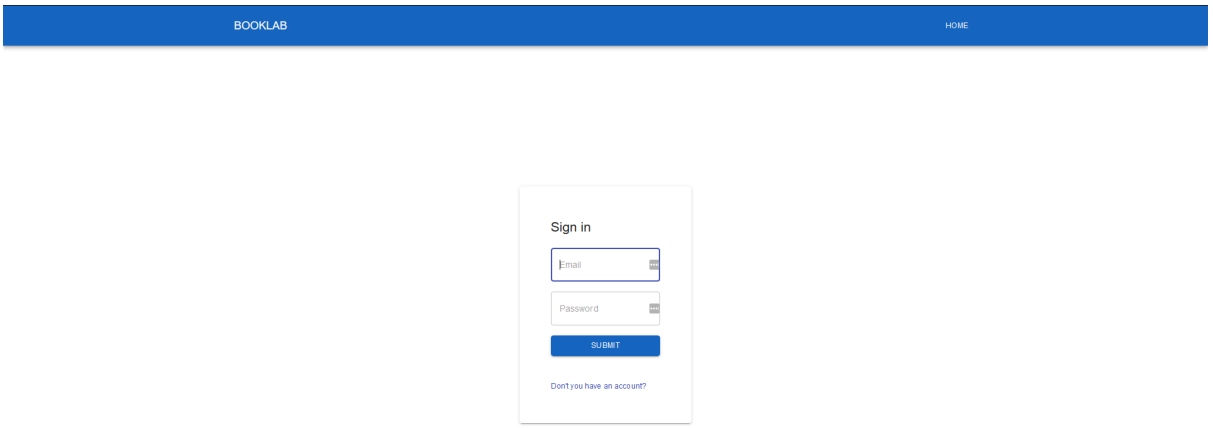

figure 2: Login page

User Inputs: @email, @password

SQL Statements:

On Login Button:

SELECT user_id

FROM User

WHERE @password = password and @email = email

3.3. Create Account Page

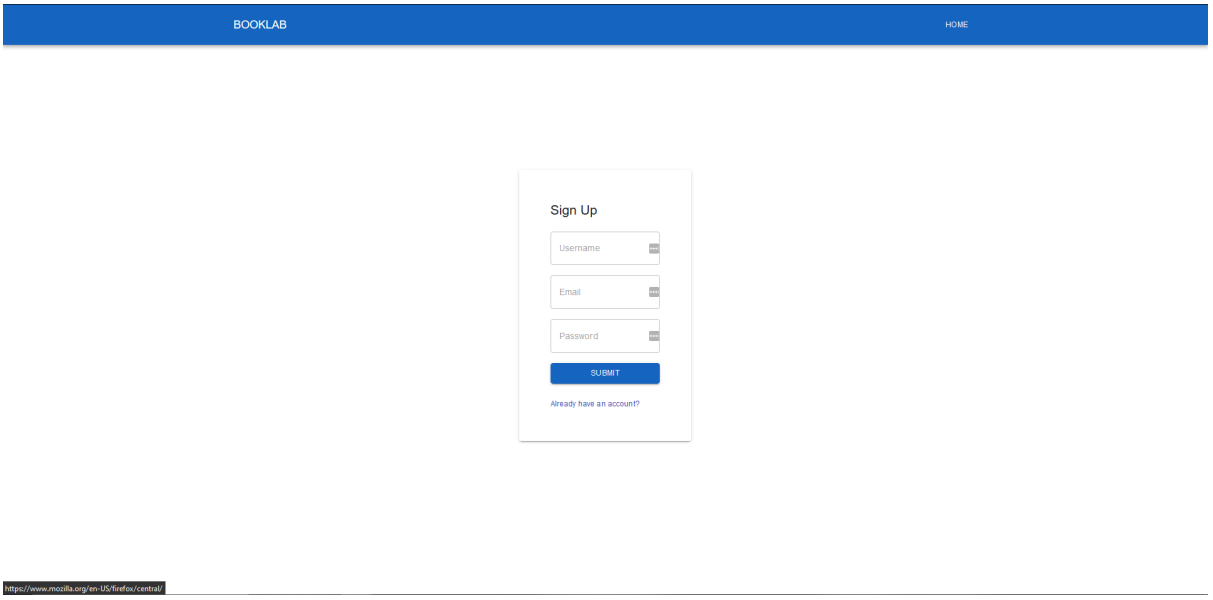

figure 3: Sign up page

User Inputs: @email, @password, @username, @name, @biography

SQL Statements:

On Register Button:

INSERT INTO User(username, email, name, biography, password)

VALUES (@username, @email, @name, @biography, @password)

3.4. My Books Page

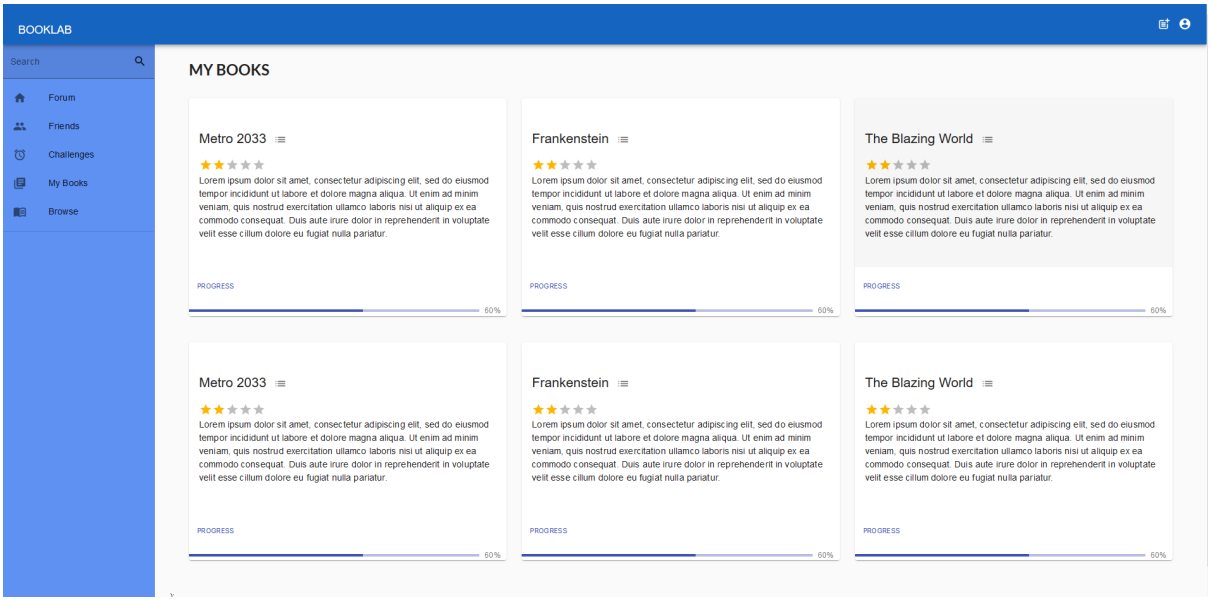

figure 4: My Books page

Session Inputs: @user_id, @friend_id, @book_id

SQL Statements:

On Initial Listing:

SELECT title, description, page_number, rating

FROM mark_progress natural join Book natural join

Progress natural join review

WHERE @user_id = user_id

On recommend a book button:

INSERT INTO recommend(user_id, friend_id, book_id)

VALUES(@user_id, @friend_id, @book_id)

3.5. Book Search

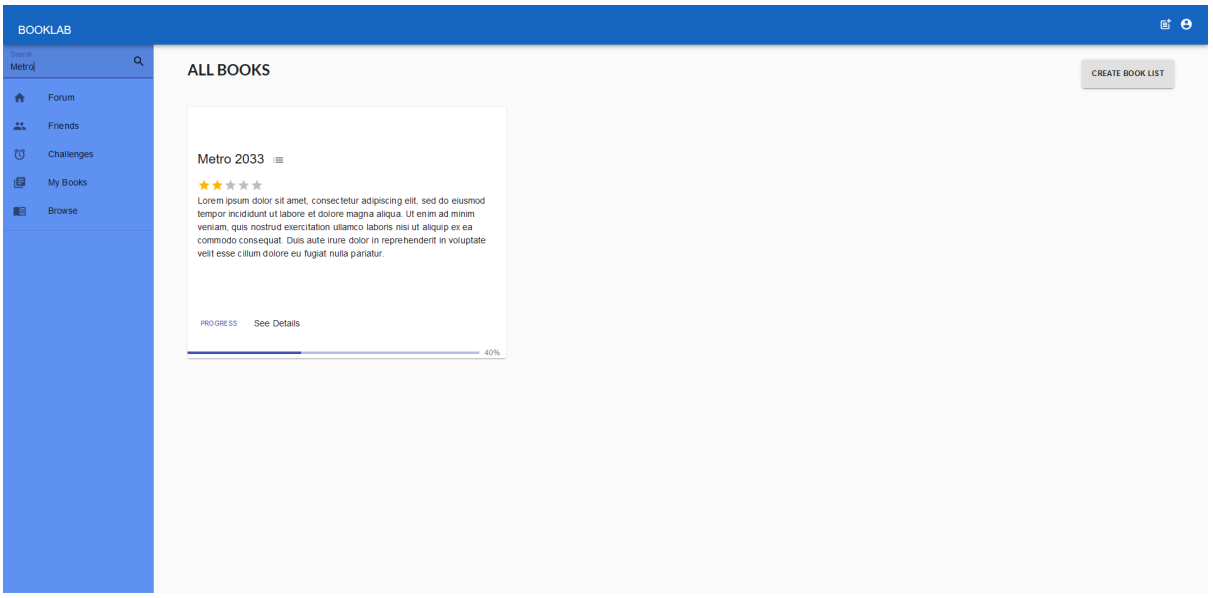

figure 5: Browse page

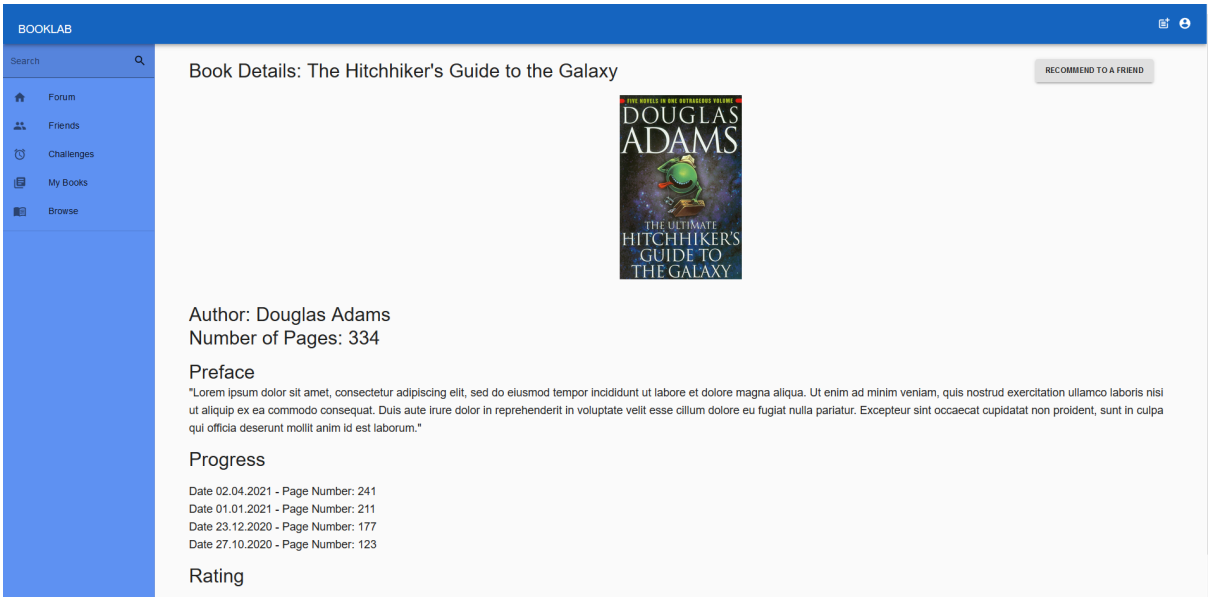

figure 6: Book details page

User Inputs: @title, @pdate, @genre, @author, @edition_name

Session Inputs: @user_id, @pro_id, @currentDate, @book_id, @ed_id

SQL Statements:

On searching:

```
SELECT title, description, year, page_number, page, rating
FROM Book natural join Edition natural join review
WHERE title like %@title% or name like
%@edition_name% or publish_date = @pdate = or genre
= @genre or @author = author_name
```

On tracking:

```
SELECT title, description, img_url, page_number, date,
rating
FROM Book natural join mark_progress natural join
Progress natural join review natural join Edition
WHERE book_id = @book_id and ed_id = @ed_id
```

3.6. Create List

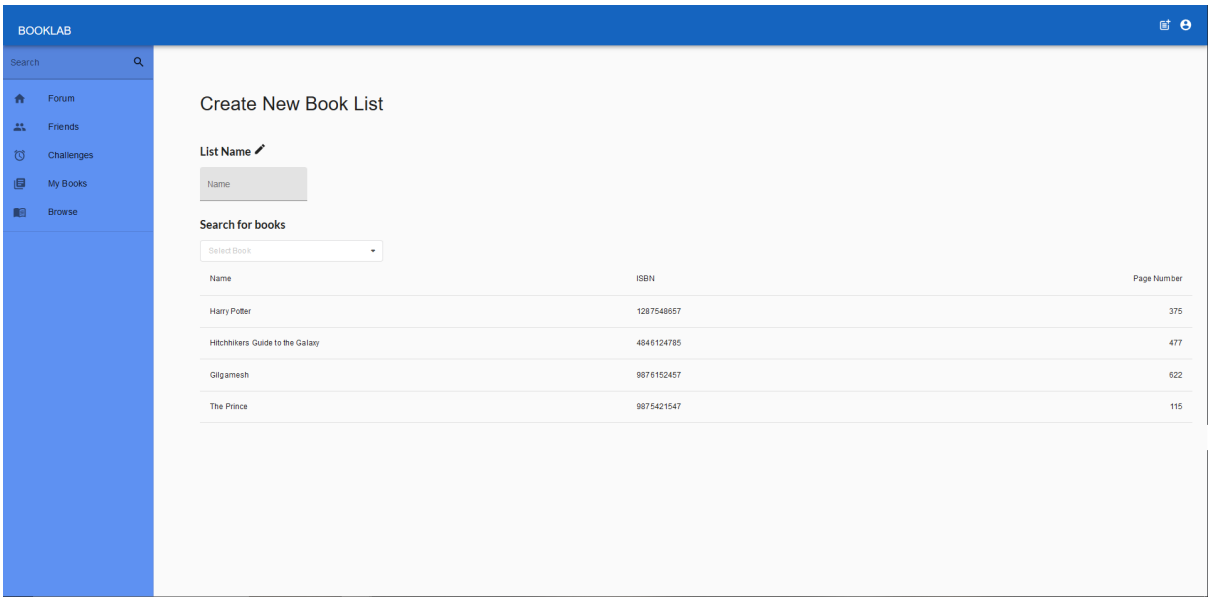

figure 7: Create book list page

User Inputs: @name

SQL Statements:

On Create List Button:

INSERT INTO Book_list (name)

VALUES (@name)

3.7. Add Book to the List

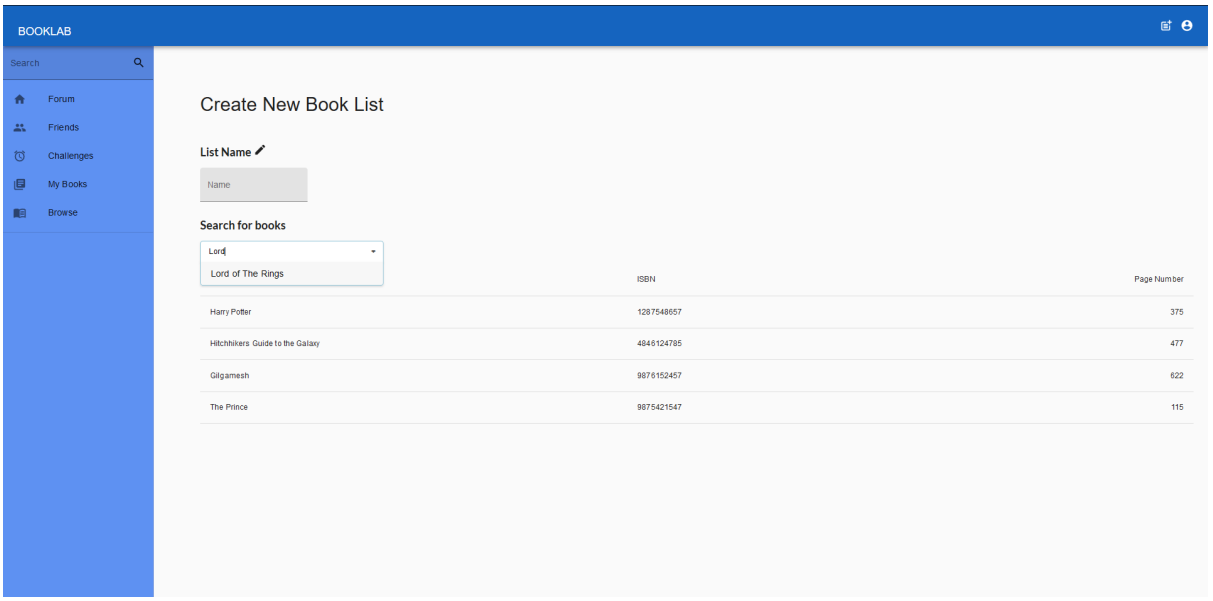

figure 8: Add book to book list

Session Inputs: @user_id, @book_id, @bl_id

SQL Statements:

Add books to the list:

INSERT INTO has_books(book_id, bl_id)

VALUES (@book_id, @bl_id)

3.8. Add Friends

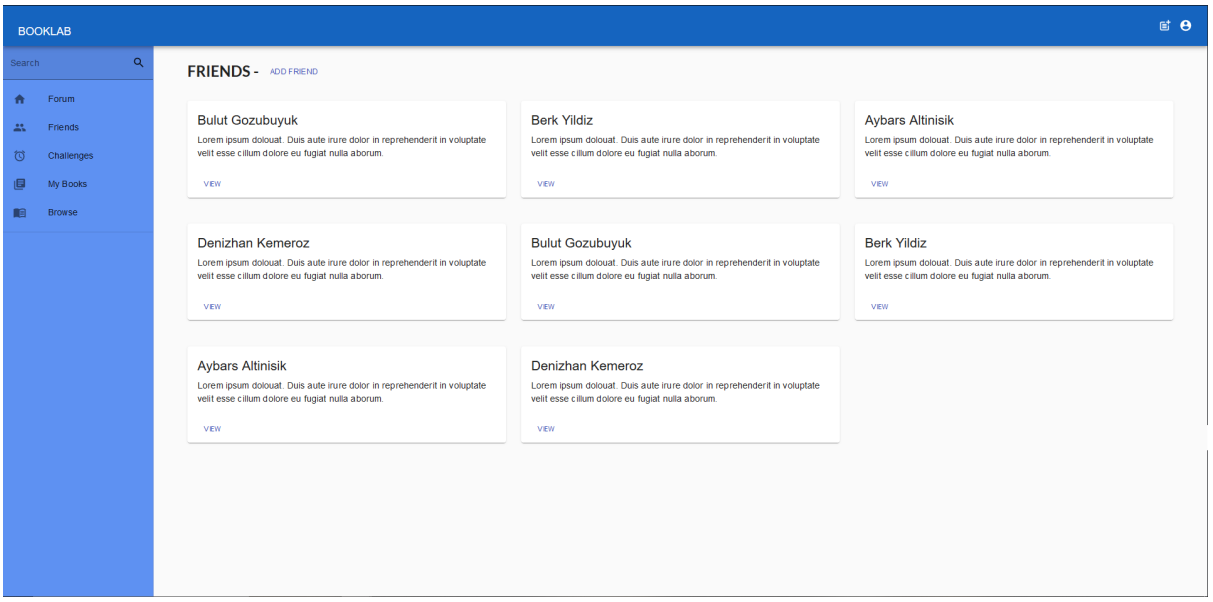

figure 9: My friends page

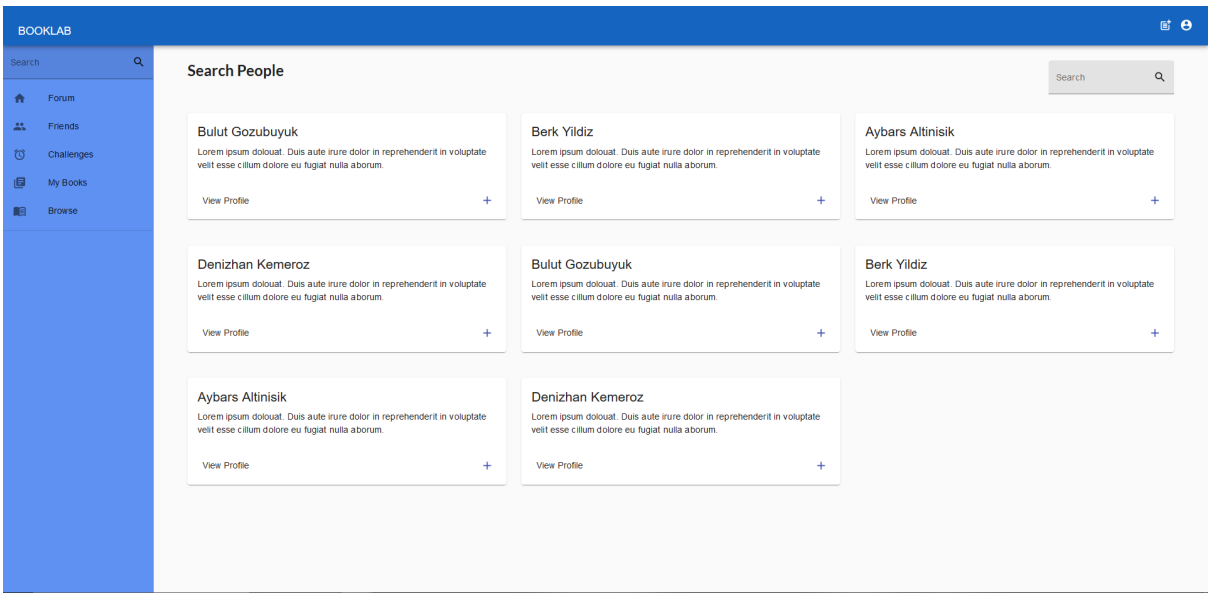

figure 10: Search for friends page

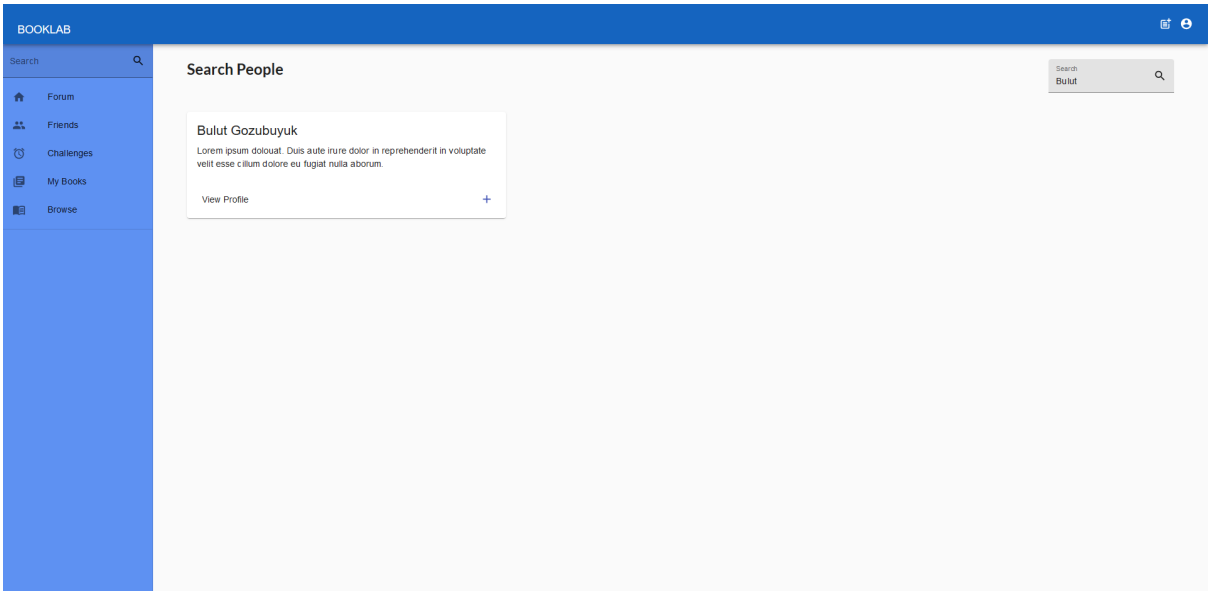

figure 11: Searching for bulut

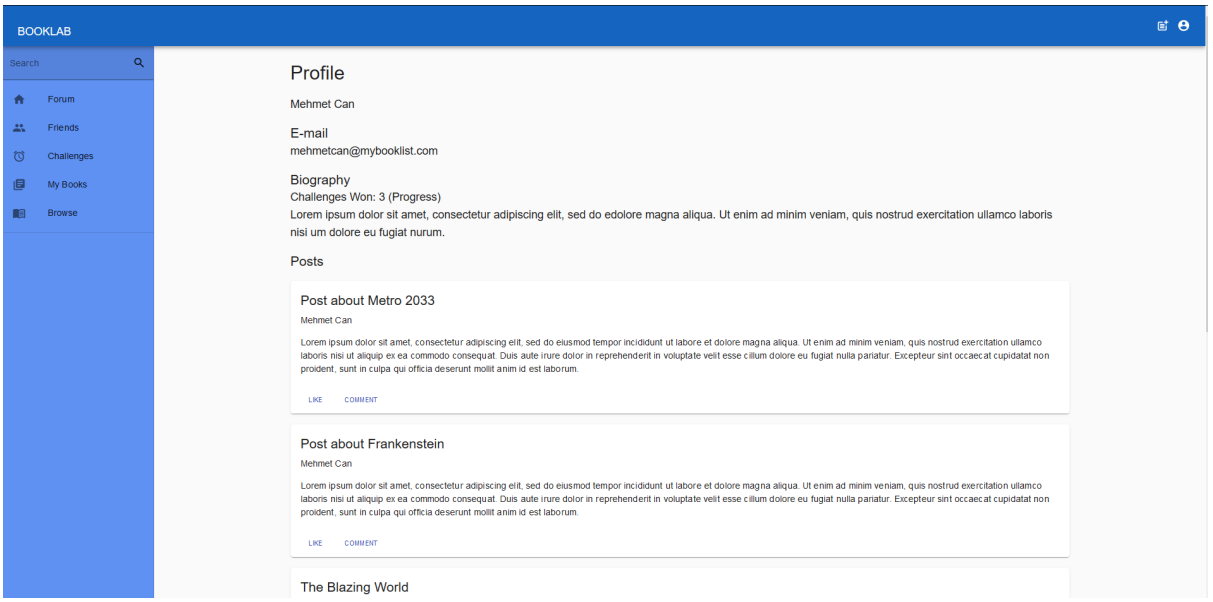

figure 12: Profile of a friend

Session Inputs: @user_id, @friend_id

SQL Statements:

On Add Friend Button:

INSERT INTO friend_of(user_id, friend_id, 0)

VALUES (@user_id, @friend_id, @accepted)

On Accept Request Button:

UPDATE friend_of

SET accepted = 1

WHERE user_id = @user_id AND @friend_id = friend_id

3.9. Challenges Page

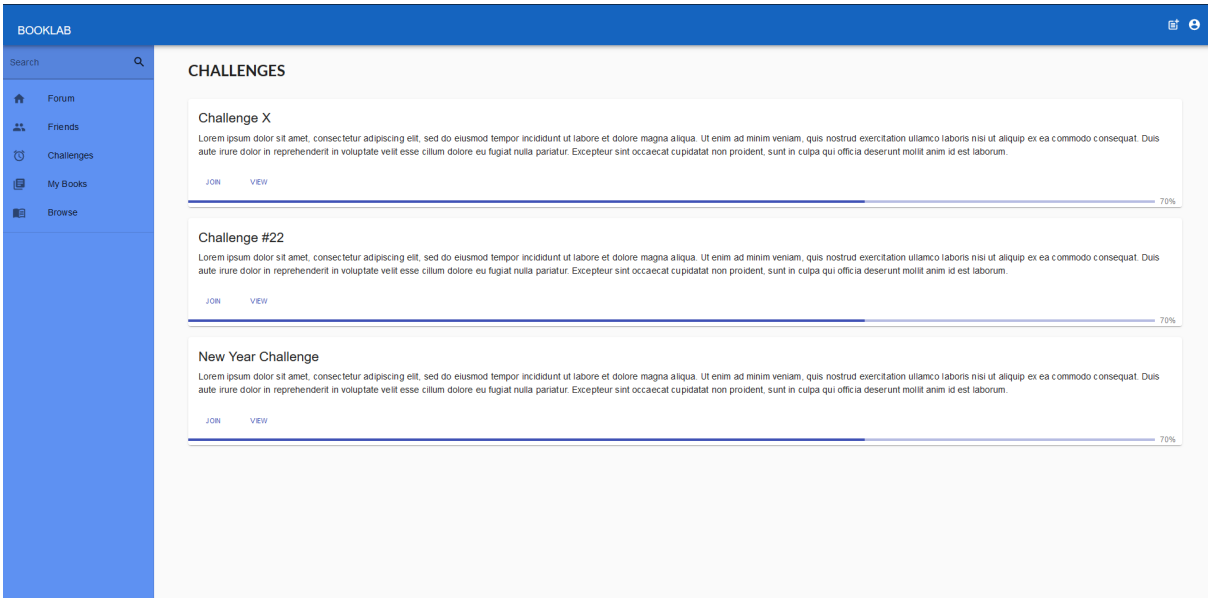

figure 13: Challenges page

User Inputs: @title, @pdate, @genre, @author, @edition_name

SQL Statements:

Listing items:

SELECT challenge_name, due_date, type, book_count

FROM Challenge natural join creates_challenge natural

join Book_list

Joining a challenge:

```
INSERT INTO joins_challenge(user_id, chal_id,
book_read)
VALUES(@user_id, @chal_id, 0)
```

3.10. Forum

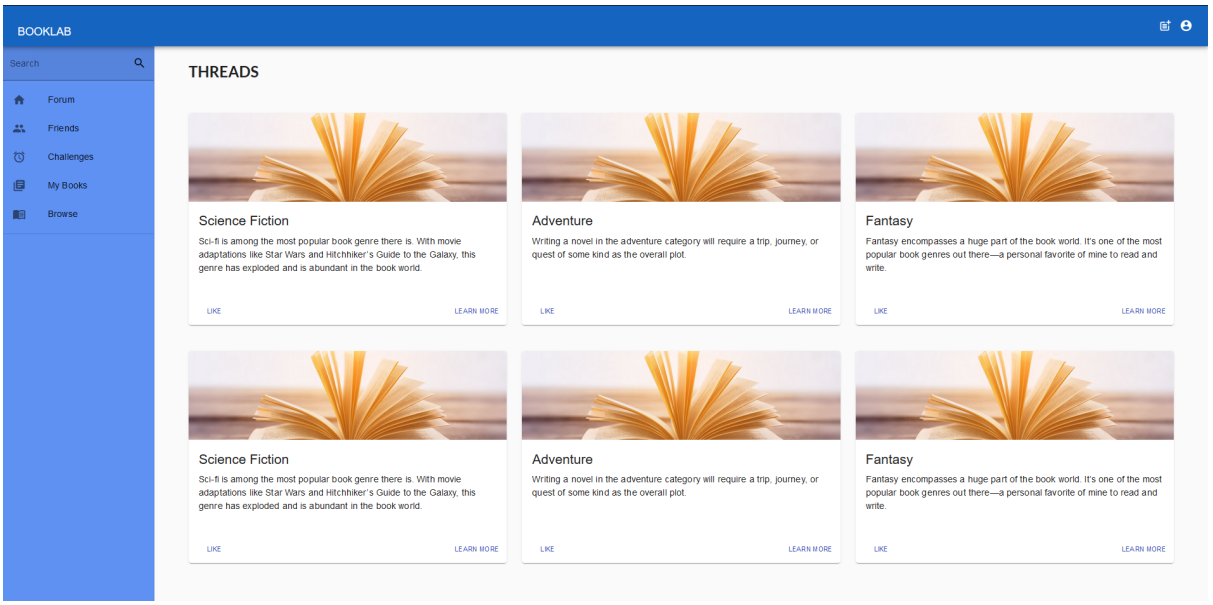

figure 14: Forum page (Threads)

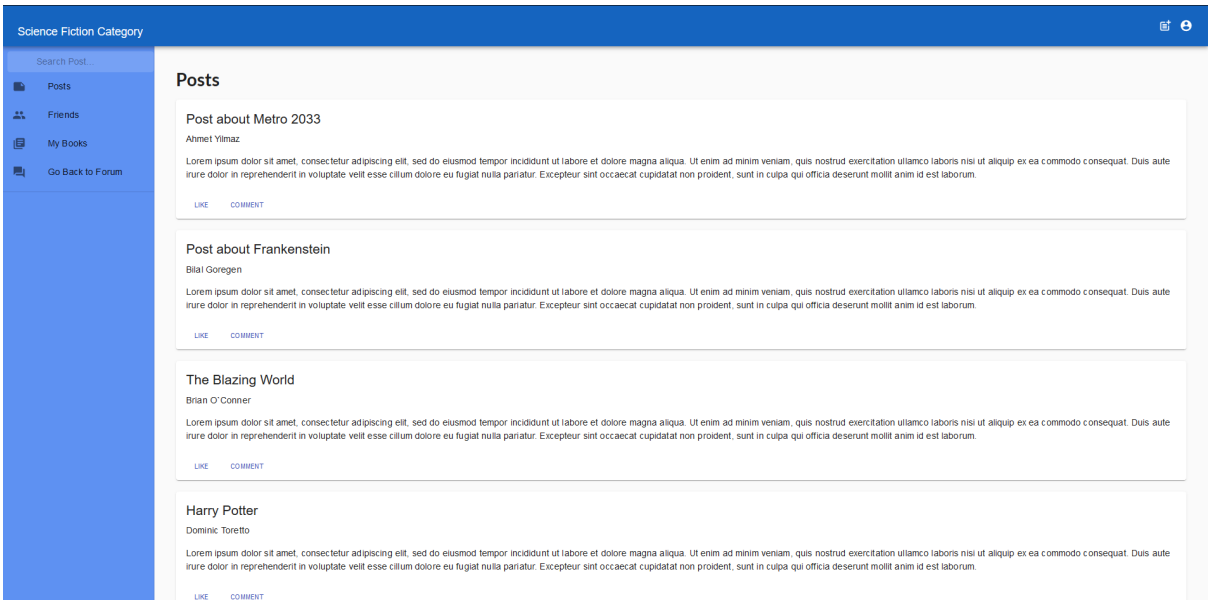

figure 15: Posts for Science Fiction thread

User Inputs: @text

Session Inputs: @pid, @currentDate, @tid, @cid, @user_id

SQL Statements:

Listing threads:

```
SELECT name, category
FROM Thread
```

Listing followed threads:

```
SELECT name, category
FROM Thread natural join follows
```

Listing posts:

```
SELECT username, name, text, date, like
FROM Post natural join posts natural join User
WHERE tid = @tid
```

Listing post's comments:

```
SELECT username, name, text, date
FROM Comment natural join writes natural join User
WHERE @pid = Comment.pid
```

Posting a comment:

```
INSERT INTO Comment(text, date, user_id)
```

```
VALUES(@text, @currentDate, @user_id)

INSERT INTO post_comment(pid, cid)
VALUES(@pid, @cid)
```

Posting a post:

```
INSERT INTO Post(like, text, date, user_id, tid)
VALUES(0, @text, @currentDate, @user_id, @tid)
```

Liking a post:

```
UPDATE Post
SET like = like + 1
WHERE @pid = pid
```

Remove a like from post:

```
UPDATE Post
SET like = like - 1
WHERE @pid = pid
```

4. Project Web Page

- https://cs353group11.github.io/