
CSE414 DATABASE SYSTEMS SEMESTER PROJECT

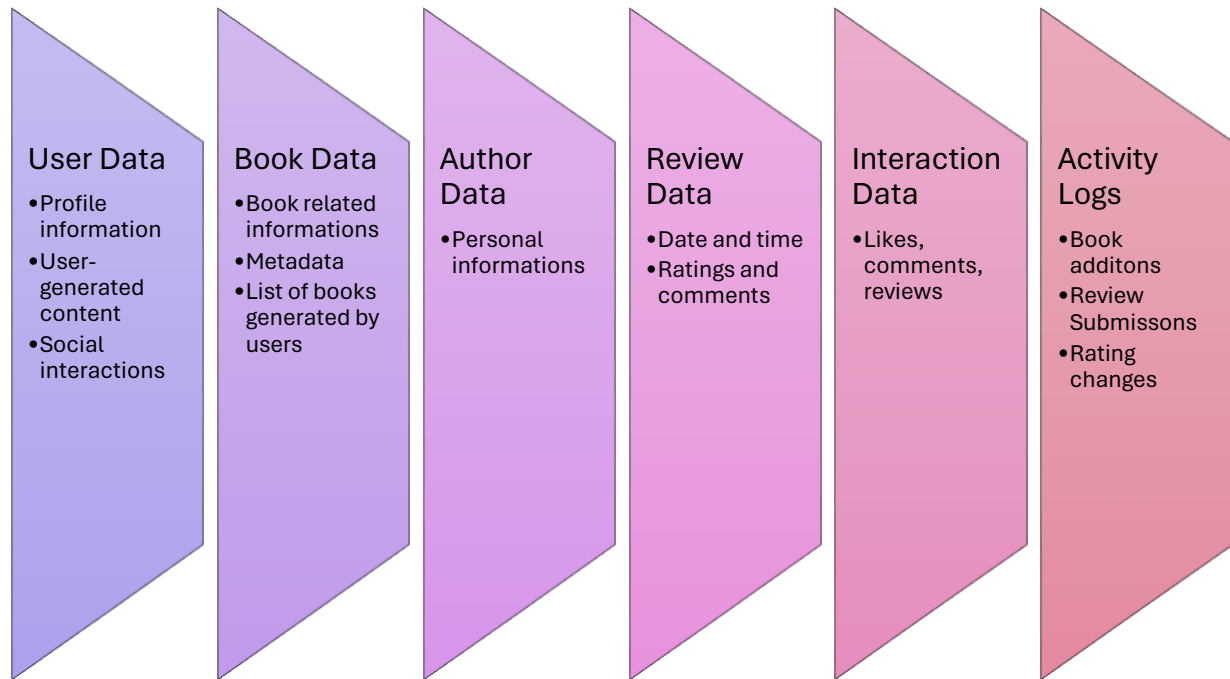
GOODREADS BACKEND

AYŞE GÜL DEMİRBİLEK

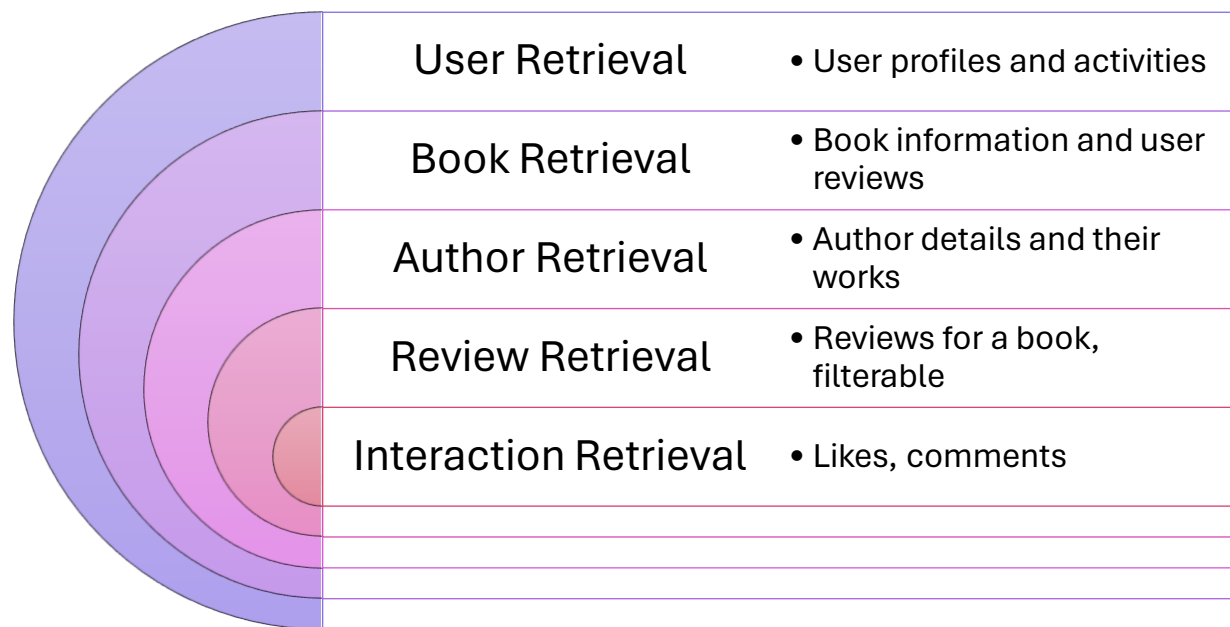
1801042088

USER REQUIREMENTS

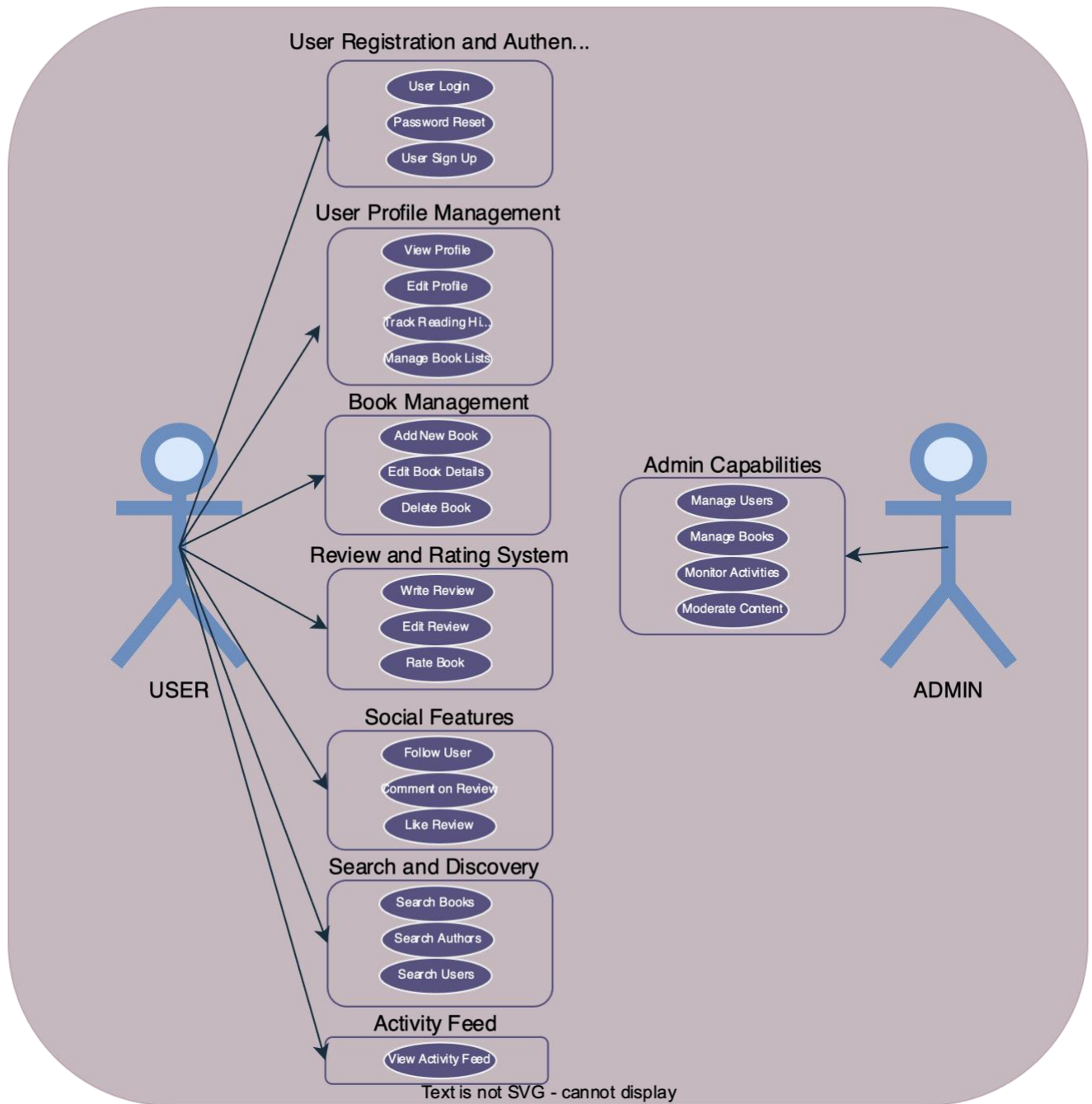
DATA STORAGE



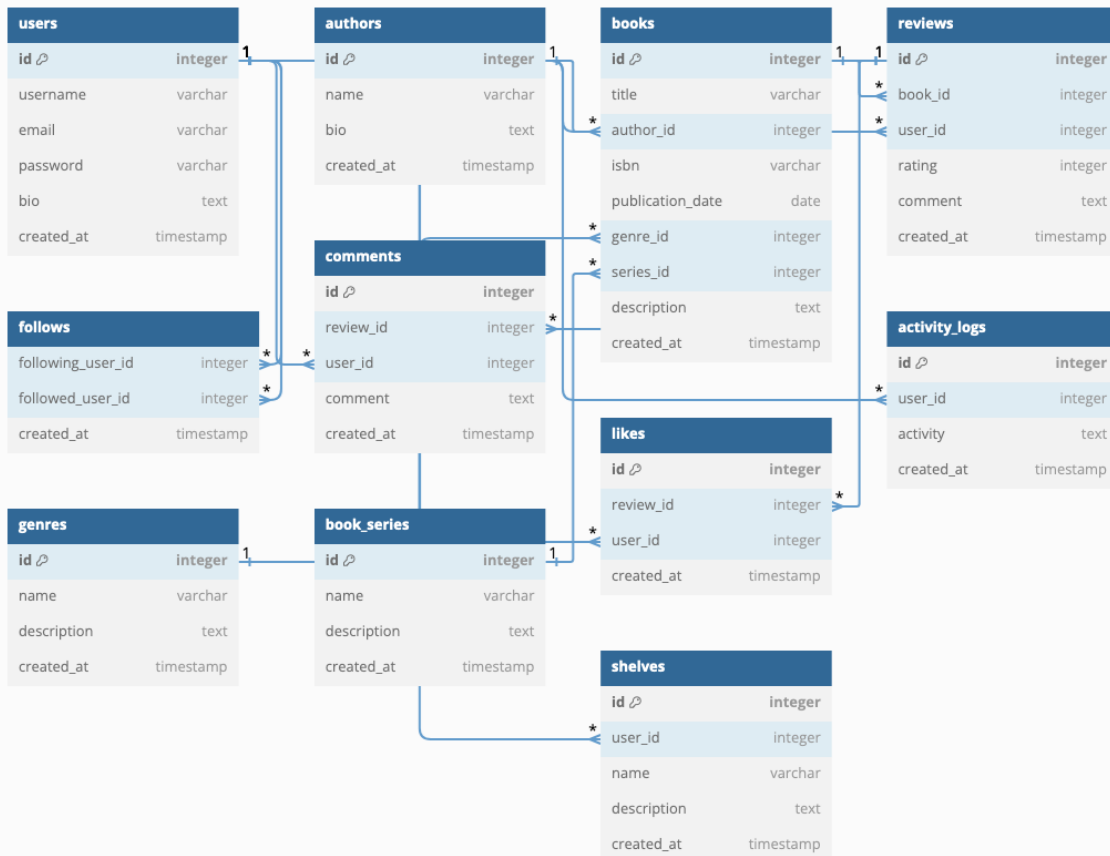
RETRIEVAL CAPACITIES



SPECIFIC FUNCTIONALITIES DESIRED BY USERS



ENTITY-RELATIONSHIP (E-R) DIAGRAM



- Each user (users) can write multiple reviews (reviews), and each review is linked to a specific book (books) and a user.
- Books are authored by authors (authors), creating a many-to-one relationship between books and authors.
- Users can follow other users through the follows table, establishing self-referential relationships within the users table.
- Reviews can receive multiple comments (comments) and likes (likes), both linked to users and reviews, representing many-to-one relationships.
- Additionally, user activities are logged in the activity_logs table, recording various actions performed by users, linked back to the users table.

FUNCTIONAL DEPENDENCIES & DATABASE SCHEMA

| Tables | Fields | Explanation |
|---------------|--------------------------|--|
| Users | user_id | user_id uniquely determines the username, email, password, and created_at attributes for each user. |
| | username | |
| | email | |
| | password | |
| | created_at | |
| Authors | author_id | author_id uniquely determines the name, bio, and created_at attributes for each author. |
| | name | |
| | bio | |
| | created_at | |
| Books | book_id | book_id uniquely determines the title, author_id, isbn, publication_date, genre_id, series_id, description, and created_at attributes for each book. |
| | title | |
| | author_id | |
| | publication_date, | |
| | isbn | |
| | genre_id | |
| | series_id | |
| | description | |
| | created_at | |
| Reviews | review_id | review_id uniquely determines the book_id, user_id, rating, comment, and created_at attributes for each review. |
| | book_id | |
| | user_id | |
| | rating | |
| | comment | |
| | created_at | |
| Follows | following_user_id | The combination of following_user_id and followed_user_id uniquely determines the created_at attribute for each follow relationship. |
| | followed_user_id | |
| | created_at | |
| Comments | comment_id | comment_id uniquely determines the review_id, user_id, comment, and created_at attributes for each comment. |
| | review_id | |
| | user_id | |
| | comment | |
| | created_at | |
| Likes | like_id | like_id uniquely determines the review_id, user_id, and created_at attributes for each like. |
| | review_id | |
| | user_id | |
| | created_at | |
| Activity Logs | activity_log_id | activity_log_id uniquely determines the user_id, activity, and created_at attributes for each activity log. |
| | user_id | |
| | activity | |
| Genres | genre_id | genre_id uniquely determines the name, description, and created_at attributes for each genre. |
| | name | |
| | description | |
| | created_at | |
| Book_Series | series_id | series_id uniquely determines the name, description, and created_at attributes for each book series. |
| | name | |
| | description | |
| | created_at | |
| Shelves | shelf_id | shelf_id uniquely determines the user_id, name, description, and created_at attributes for each shelf. |
| | user_id | |
| | name | |
| | description | |
| | created_at | |

NORMALIZATION

All the tables are created aligning with 3NF rules, their id's are primary keys and all other attributes are fully functionally dependent on it. So I represent example of compositions which are not in any normalization form:

| book_id | book_title | author_name | author_bio |
|---------|------------|-------------|------------|
| 1 | Book 1 | Author 1 | Bio 1 |
| 2 | Book 2 | Author 1 | Bio 1 |
| 3 | Book 3 | Author 2 | Bio 2 |

2NF/3NF (Eliminate Partial Dependency):

| book_id | book_title | author_id |
|---------|------------|-----------|
| 1 | Book 1 | 1 |
| 2 | Book 2 | 1 |
| 3 | Book 3 | 2 |

| author_id | author_name | author_bio |
|-----------|-------------|------------|
| 1 | Book 1 | Bio 1 |
| 2 | Book 2 | Bio 2 |

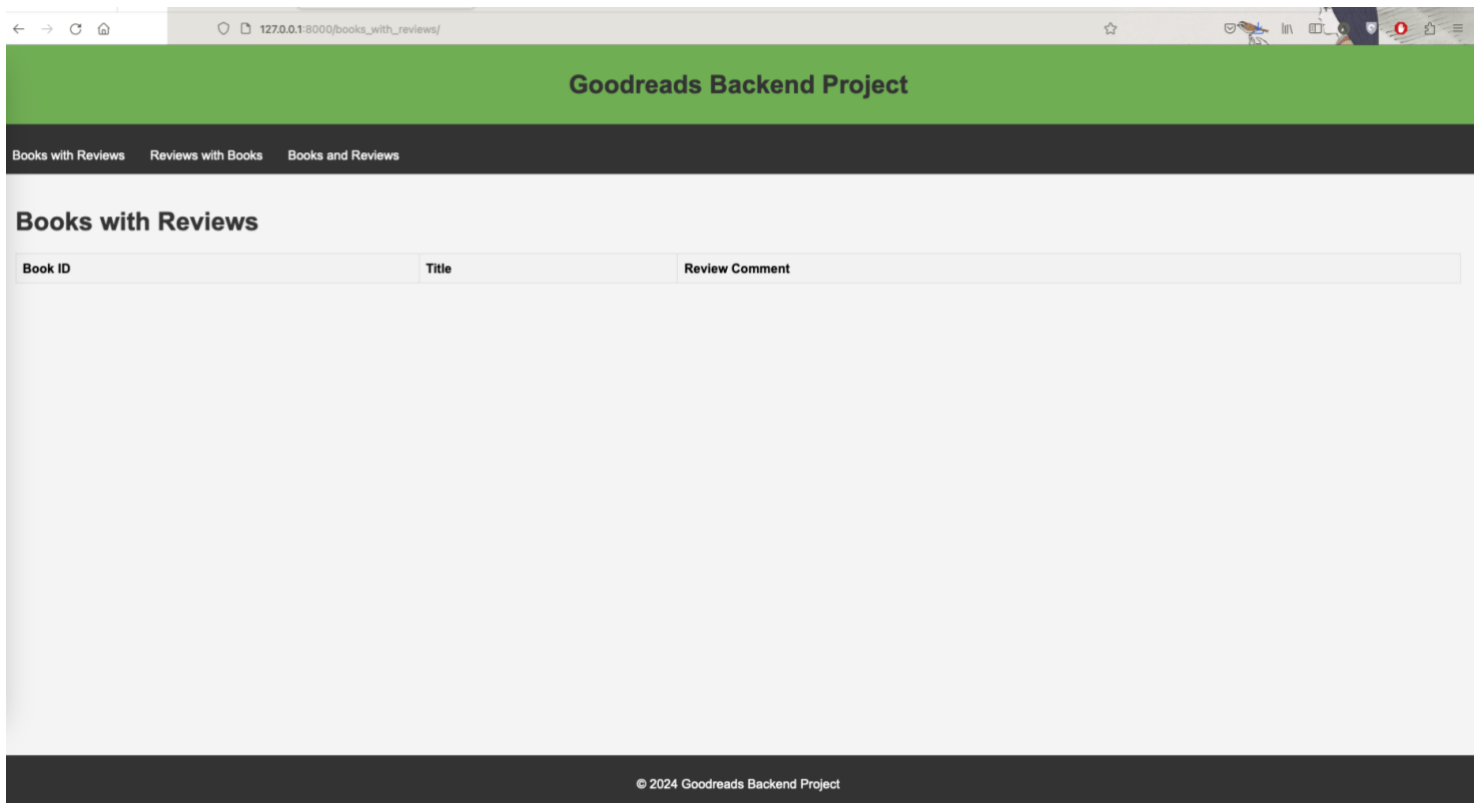
DATABASE INTEGRATION

```
ayseguldemirbilek — psql -p5432 goodreads_db — 80x36
Last login: Thu Jun 13 22:56:39 on ttys007
"/Applications/Postgres.app/Contents/Versions/15/bin/psql" -p5432 "goodreads_db"
ayseguldemirbilek@Ayses-MacBook-Pro ~ % "/Applications/Postgres.app/Contents/Versions/15/bin/psql" -p5432 "goodreads_db"
psql (15.7 (Postgres.app))
Type "help" for help.

goodreads_db=# \dt
               List of relations
Schema |      Name      | Type | Owner
-----|-----|-----|-----
public | Books_activitylog | table | goodreads_user
public | Books_author    | table | goodreads_user
public | Books_book      | table | goodreads_user
public | Books_bookseries | table | goodreads_user
public | Books_comment   | table | goodreads_user
public | Books_follow    | table | goodreads_user
public | Books_genre     | table | goodreads_user
public | Books_like      | table | goodreads_user
public | Books_review    | table | goodreads_user
public | Books_shelf     | table | goodreads_user
public | auth_group      | table | goodreads_user
public | auth_group_permissions | table | goodreads_user
public | auth_permission | table | goodreads_user
public | auth_user       | table | goodreads_user
public | auth_user_groups | table | goodreads_user
public | auth_user_user_permissions | table | goodreads_user
public | django_admin_log | table | goodreads_user
public | django_content_type | table | goodreads_user
public | django_migrations | table | goodreads_user
public | django_session  | table | goodreads_user
(20 rows)

goodreads_db=#
```

USER INTERFACE



TRIGGERS

| TRIGGERS | Trigger Name | Function | Purpose |
|--|-----------------------------------|---------------------------|---|
| Update Review Count for a Book | review_count_trigger | update_review_count | Increment the review count for a book whenever a new review is added. |
| Log User Activities | log_activity_trigger | log_user_activity | Log any activity performed by a user into the Books_activitylog table. |
| Update Last Activity Date | update_last_activity_trigger | update_last_activity | Update the last activity date of a user whenever they perform any action. |
| Prevent Duplicate Follows | prevent_duplicate_follows_trigger | prevent_duplicate_follows | Prevent a user from following the same user more than once. |
| Set Default Shelf for New Users | create_default_shelf_trigger | create_default_shelf | Automatically create a default shelf for a new user when they sign up. |

Books_review table and includes a list of triggers defined on the table, such as review_count_trigger.

```

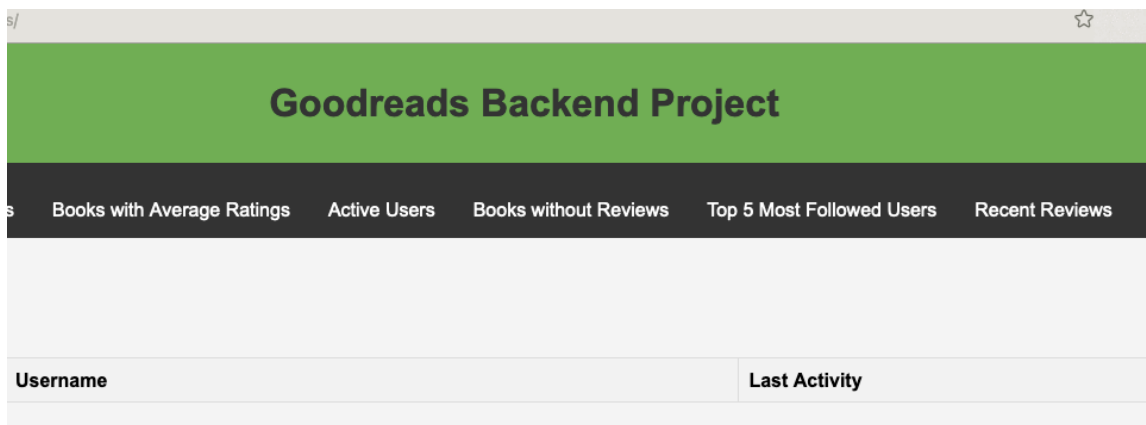
~
goodreads_db=# \d "Books_review"
                                Table "public.Books_review"
  Column      |      Type      | Collation | Nullable |      Default
-----|-----|-----|-----|-----
id             | bigint         |           | not null | generated by default as identity
rating         | integer        |           | not null |
comment        | text           |           | not null |
created_at     | timestamp with time zone |           | not null |
book_id        | bigint         |           | not null |
user_id        | integer        |           | not null |
Indexes:
    "Books_review_pkey" PRIMARY KEY, btree (id)
    "Books_review_book_id_eab688e2" btree (book_id)
    "Books_review_user_id_8660a1c0" btree (user_id)
Foreign-key constraints:
    "Books_review_book_id_eab688e2_fk_Books_book_id" FOREIGN KEY (book_id) REFERENCES "Books_book"(id) DEFERRABLE INITIALLY DEFERRED
    "Books_review_user_id_8660a1c0_fk_auth_user_id" FOREIGN KEY (user_id) REFERENCES auth_user(id) DEFERRABLE INITIALLY DEFERRED
Referenced by:
    TABLE ""Books_comment"" CONSTRAINT "Books_comment_review_id_3dc96480_fk_Books_review_id" FOREIGN KEY (review_id) REFERENCES "Books_review"(id) DEFERRABLE INITIALLY DEFERRED
    TABLE ""Books_like"" CONSTRAINT "Books_like_review_id_d78e9d9b_fk_Books_review_id" FOREIGN KEY (review_id) REFERENCES "Books_review"(id) DEFERRABLE INITIALLY DEFERRED
Triggers:
    log_activity_trigger AFTER INSERT OR DELETE OR UPDATE ON "Books_review" FOR EACH ROW EXECUTE FUNCTION log_user_activity()
    review_count_trigger AFTER INSERT ON "Books_review" FOR EACH ROW EXECUTE FUNCTION update_review_count()
    update_last_activity_trigger AFTER INSERT OR UPDATE ON "Books_review" FOR EACH ROW EXECUTE FUNCTION update_last_activity()
goodreads_db=#

```


VIEWS

| VIEWS | Utility |
|---------------------------------------|---|
| List Books with Their Average Ratings | This view provides an easy way to see the average rating of each book. |
| List Active Users | This view helps identify users who are actively engaging with the platform. |
| List Books Without Reviews | This view highlights books that have not been reviewed yet. |
| List Top 5 Most Followed Users | This view identifies the most popular users on the platform. |
| List Recent Reviews | This view provides an up-to-date list of recent reviews. |

In the user interface, these views are displayed under the corresponding tabs.



The views also can be checked from psql console.

```
ayseguldemirbilek — psql -p5432 goodreads_db — 69x13
goodreads_db=# \dv
              List of relations
Schema |      Name      | Type | Owner
-----+-----+-----+-----
public | active_users   | view | goodreads_user
public | books_with_avg_ratings | view | goodreads_user
public | books_without_reviews | view | goodreads_user
public | recent_reviews  | view | goodreads_user
public | top_5_most_followed_users | view | goodreads_user
(5 rows)

goodreads_db=#
```

SQL STATEMENTS

```
GoodreadsBackend

settings.py M  sql_statements U x  models.py M  utils.py ...  sql_statements U x

Goodreads > sql_statements
Run on active connection | Select block

1 CREATE TABLE Books_author (
2   id SERIAL PRIMARY KEY,
3   name VARCHAR(255) NOT NULL,
4   bio TEXT,
5   created_at TIMESTAMPT WITH TIME ZONE DEFAULT
6   CURRENT_TIMESTAMP
7 );
8 CREATE TABLE Books_genre (
9   id SERIAL PRIMARY KEY,
10  name VARCHAR(255) NOT NULL,
11  description TEXT,
12  created_at TIMESTAMPT WITH TIME ZONE DEFAULT
13  CURRENT_TIMESTAMP
14 );
15 CREATE TABLE Books_bookseries (
16   id SERIAL PRIMARY KEY,
17   name VARCHAR(255) NOT NULL,
18   description TEXT,
19   created_at TIMESTAMPT WITH TIME ZONE DEFAULT
20   CURRENT_TIMESTAMP
21 );
22 CREATE TABLE Books_book (
23   id SERIAL PRIMARY KEY,
24   title VARCHAR(255) NOT NULL,
25   author_id INTEGER REFERENCES Books_author(id),
26   isbn VARCHAR(13) NOT NULL,
27   publication_date DATE NOT NULL,
28   genre_id INTEGER REFERENCES Books_genre(id),
29   series_id INTEGER REFERENCES Books_bookseries(id)
30   NULL,
31   description TEXT NOT NULL,
32   created_at TIMESTAMPT WITH TIME ZONE DEFAULT
33   CURRENT_TIMESTAMP
34 );
35 CREATE TABLE Books_review (
36   id SERIAL PRIMARY KEY,
37   book_id INTEGER REFERENCES Books_book(id),
38   user_id INTEGER REFERENCES auth_user(id),
39   rating INTEGER NOT NULL,
40   comment TEXT NOT NULL,
41   created_at TIMESTAMPT WITH TIME ZONE DEFAULT
42   CURRENT_TIMESTAMP
43 );
44 CREATE TABLE Books_follow (
45   id SERIAL PRIMARY KEY,
46   following_user_id INTEGER REFERENCES auth_user(id),
47   followed_user_id INTEGER REFERENCES auth_user(id),
48   created_at TIMESTAMPT WITH TIME ZONE DEFAULT
49   CURRENT_TIMESTAMP
50 );
51 CREATE TABLE Books_comment (
52   id SERIAL PRIMARY KEY,
53   review_id INTEGER REFERENCES Books_review(id),
54   user_id INTEGER REFERENCES auth_user(id),
55   comment TEXT NOT NULL,
56   created_at TIMESTAMPT WITH TIME ZONE DEFAULT
57   CURRENT_TIMESTAMP
58 );
59 CREATE TABLE Books_activitylog (
60   id SERIAL PRIMARY KEY,
61   user_id INTEGER REFERENCES auth_user(id),
62   activity TEXT NOT NULL,
63   created_at TIMESTAMPT WITH TIME ZONE DEFAULT
64   CURRENT_TIMESTAMP
65 );
66 CREATE TABLE Books_shelf (
67   id SERIAL PRIMARY KEY,
68   user_id INTEGER REFERENCES auth_user(id),
69   name VARCHAR(255) NOT NULL,
70   description TEXT,
71   created_at TIMESTAMPT WITH TIME ZONE DEFAULT
72   CURRENT_TIMESTAMP
73 );
74 CREATE TABLE Books_shelfbooks (
75   id SERIAL PRIMARY KEY,
76   shelf_id INTEGER REFERENCES Books_shelf(id),
77   book_id INTEGER REFERENCES Books_book(id),
78   added_at TIMESTAMPT WITH TIME ZONE DEFAULT
79   CURRENT_TIMESTAMP
80 );
81 -- Insert authors
82 INSERT INTO Books_author (name, bio) VALUES ('Author
83 One', 'Bio of author one');
84 INSERT INTO Books_author (name, bio) VALUES ('Author
85 Two', 'Bio of author two');
86 -- Insert genres
87 INSERT INTO Books_genre (name, description) VALUES
88 ('Genre One', 'Description of genre one');
89 INSERT INTO Books_genre (name, description) VALUES
90 ('Genre Two', 'Description of genre two');
91 -- Insert books
92 INSERT INTO Books_book (title, author_id, isbn, publication_date,
93 genre_id, description) VALUES
94 ('Book One', 1, '1234567890123', '2023-01-01', 1, 'Description of book
95 one');
96 INSERT INTO Books_book (title, author_id, isbn, publication_date,
97 genre_id, description) VALUES
98 ('Book Two', 2, '1234567890123', '2023-01-01', 2, 'Description of book
99 two');
100 -- Insert users
101 INSERT INTO auth_user (username, password, is_superuser, is_staff,
102 is_active) VALUES
103 ('user1', 'pbkdf2_sha256$320000$9qIzlp... (hashed password for user1)',
104 FALSE, FALSE, TRUE);
105 INSERT INTO auth_user (username, password, is_superuser, is_staff,
106 is_active) VALUES
107 ('user2', 'pbkdf2_sha256$320000$9qIzlp... (hashed password for user2)',
108 FALSE, FALSE, TRUE);
109 -- Insert reviews
110 INSERT INTO Books_review (book_id, user_id, rating, comment) VALUES
111 (1, 1, 5, 'Great book!');
112 INSERT INTO Books_review (book_id, user_id, rating, comment) VALUES
113 (2, 2, 4, 'Good book!');
114 -- Insert follows
115 INSERT INTO Books_follow (following_user_id, followed_user_id) VALUES
116 (1, 2);
117 INSERT INTO Books_follow (following_user_id, followed_user_id) VALUES
118 (2, 1);
119 -- Insert comments
120 INSERT INTO Books_comment (review_id, user_id, comment) VALUES
121 (1, 1, 'Nice review!');
122 INSERT INTO Books_comment (review_id, user_id, comment) VALUES
123 (2, 2, 'Thanks for the review!');
124 -- Insert likes
125 INSERT INTO Books_like (review_id, user_id) VALUES
126 (1, 1);
127 INSERT INTO Books_like (review_id, user_id) VALUES
128 (2, 2);
129 -- Insert activity logs
130 INSERT INTO Books_activitylog (user_id, activity) VALUES
131 (1, 'Added a review');
132 INSERT INTO Books_activitylog (user_id, activity) VALUES
133 (2, 'Liked a review');
134 -- Insert shelves
135 INSERT INTO Books_shelf (user_id, name, description) VALUES
136 (1, 'Favorite Books', 'Books I love');
137 INSERT INTO Books_shelf (user_id, name, description) VALUES
138 (2, 'To Read', 'Books I want to read');
139 -- Insert shelf books
140 INSERT INTO Books_shelfbooks (shelf_id, book_id) VALUES
141 (1, 1);
142 INSERT INTO Books_shelfbooks (shelf_id, book_id) VALUES
143 (2, 2);

PROBLEMS OUTPUT TERMINAL PORTS SQL CONSOLE COMMENTS DEBUG CONSOLE

(GoodreadsVenv) ayseguldemirbilek@Ayse-MacBook-Pro Goodreads % python manage.py migrate
Operations to perform:
  Apply all migrations: Books, admin, auth, contenttypes, sessions
Running migrations:
  Applying Books.0004_shelfbooks... OK
(GoodreadsVenv) ayseguldemirbilek@Ayse-MacBook-Pro Goodreads %
```

