Recommender Companion

- Ashi Sinha, Chiranjeewee Koirala, Yukti Sharma, Vrinda Goel

Objective:

Our web application provides personalized book and movie recommendations. The app will allow users to input ratings for at least three books and movies they've experienced, and based on these preferences, generate tailored recommendations. The aim is to simplify content discovery by suggesting books and movies that match the user's distinct taste.

Data Sources:

- We use 2 datasets for our source information, both of which have been sourced from Kaggle:
    - Movies - The Ultimate 1 Million Movie Dataset ([Link](#))
    - Books - Book Recommendation Dataset ([Link](#))
- The movies dataset emulates "The Movies Database" (TMDB) repository, while the books dataset emulates the Goodreads repository.
- Both the dataset required significant clean up due to null/missing values, data corruption, duplicate values etc.

Data Pre-processing:

- Movies dataset originally had **1,023,264** entries, while the Books dataset originally had **271,360** entries.
- Cleanup:
  - Selected only the necessary columns from each of the datasets and dropped the rest (Eg: tagline, popularity, IMDB ID)
  - Missing/Null value removal: Removed several thousand rows in both the datasets
  - Data Corruption: Found columns like "Year of Publication" in the books dataset containing information of the publishing house instead; removed such rows in both datasets.
  - Data Truncation: Due to limitation of resources, we could not work with files over 100+ MB, so we truncated the movies file by selecting random entries to an acceptable size.

Infra Setup

Basic Setup:

- **Docker**: Containerised the application to ensure consistency across different environments.
  - We have defined services like the frontend, databases, and other dependencies in the docker compose file.
  - The Docker Compose file manages and simplifies the setup process by allowing all services to be launched with a single command.
- **MySQL Database**: Utilized as the primary database system.
  - Configured within Docker for development and testing purposes.
- **Flyway**: Database migration tool integrated for version control and schema migration.
  - Ensures database schema is up-to-date and consistent across deployments.
  - Configured to run migrations automatically at service startup.
- **Microsoft Authentication**: Integrates Microsoft OAuth for secure user authentication.
  - Utilizes Microsoft Graph API to fetch user details like email, display name, and roles.

Basic Setup:

- **Backend Technology**:
  - Python with Flask framework for the API server.
  - SQLAlchemy ORM for database interactions.
  - Secured API routes with user and admin middleware for authentication and authorization.
- **Frontend Technology**:
  - Angular for the client-side framework.
  - Communicates with the backend through RESTful APIs.
  - Implements reactive forms and client-side routing.
- **Security Features**:
  - Utilizes HTTPS for secure communication between client and server.
  - JWT tokens used for maintaining user sessions and securing API endpoints.
- **OpenAI GPT 3.5 Turbo API**: Used for Recommendation generation

Data Model

We have 5 tables in our database:

- **Movies** table which acts as the repository of the TMBD repo.

ID	Title	Release Date	Language	Genres	Cast	Director	Poster Path

- **Books** table which acts as the repository of the GoodReads repo.

ISBN	Title	Author	Year of Publication	Image URL

- **User** table which stores information of all active users.

ID	Display Name	Email	Age	Created At	Onboarding Completed

- **User Watched Movies** table which stores the information about movies watched by every user.

UUID	Email	Movie ID	User Rating

- **User Read Books** table which stores the information about books read by every user.

UUID	Email	Book ISBN	User Rating

Features

Authentication

- **User Interaction** - Completed
  - Both admins and regular users are presented with a "Login with Microsoft" button.
  - When clicked, the user is redirected to Microsoft's login page
- **Microsoft Authentication** - Completed
  - User enters their Microsoft credentials
  - Microsoft verifies the user's identity
- **Redirection to Our Application** - Completed
  - Microsoft redirects the user back to our application with an authorization code

Authentication

- **Backend Processing** - Completed
  - Our frontend sends the authorization code to our backend
  - Backend exchanges the code for an ID token from Microsoft
- **Token Verification** - Completed
  - Backend verifies the ID token's signature using Microsoft's public key
  - Backend checks that the token is intended for our application
- **Session Establishment** - Completed
  - Backend sets the ID token as a cookie to maintain the user's session

Authentication

- **Role Determination** - Completed

  - Backend checks the ID token for an "admin" role

  - Sends role information back to the frontend

- **User Interface** - Completed

  - Frontend displays either the admin or user interface based on the

    role

User Home Page (Read):

- The user can see all the books and movies they have added to their profile - Completed

- The content is displayed in a **paginated, tabbed** format, allowing for easy navigation and browsing of their collection - Completed

- The user sees the following buttons:

    - Add movies - Completed

    - Add books - Completed

    - Generate recommendations - Completed

    - Delete profile - Completed

Add Movie / Book (Create):

- During signup, users are required to provide ratings for at least 3 books and movies they've watched/read - Completed

- After initial sign up, users can continue adding more movies and books to their profile - Completed

- Our app will include an **autocomplete** feature that enables users to find titles quickly as they type and the results are **ranked, and paginated** - Completed

- When they add a movie/book to the list, they can also add their rating for the same to indicate how much they like/dislike it - Completed

Delete Book / Movie (Delete):

- Users have the flexibility to remove any books or movies from their profile. This helps them maintain an up-to-date list of rated content, ensuring that the recommendations remain aligned with the current preferences of the user - Completed

Generate Recommendations:

- The user can click a button to generate recommendations for books and movies similar to the ones they have already watched, and liked/disliked - Completed

- The app will use GPT to generate a customised and personalised list of recommendations for the user while factoring their likes and dislikes - Completed

- The prompt will be engineered and experimented with to generate the best possible results for the user - Completed

- The user's age is also factored into the recommendations to ensure age-appropriate recommendations - Completed

Filter Recommendations:

- The users can also filter the type of recommendations before they are generated.

- Some of these filters include:
  - Generate recommendations for only books, only movies, or for both.
    - If the user requests recommendations for both movies and books, the app organizes the results into **3 separate tabs: Books, Movies, and All (a combined view of both movies and books)** - Completed

Filter Recommendations:

- ○ Set how many recommendations they want to generate - Set to 10 due since we decided to use GPT API paid credits (instead of mock server) and were tied up by limited resources.

- ○ Choose if they want to generate recommendations based on their entire history, or only based on the history of either books or movies - Completed

Admin Profile / Panel:

- There will be an admin role to help maintain and update the users database, movie and book repositories - Completed

- Functionalities:
    - Add Movies - Completed
    - Add Books - Completed
    - Delete User Account - Gave authority to User (Completed)

User Profile:

- When the user signs up, their personal information including name and age is taken/entered - Completed

- The user has an option to delete their profile - Completed

Testing

Backend Testing:

All test cases pass successfully!

```
(myenv) vrinda@vrinda-OMEN-Laptop-15-en0xxx:~/Documents/Courses/CS520/Recommender-Companion/app$ pytest tests/test.py -v
================================================= test session starts =================================================
platform linux -- Python 3.12.3, pytest-8.3.4, pluggy-1.5.0 -- /home/vrinda/Documents/Courses/CS520/Project/Recommender-Companion/myenv/bin/python3
cachedir: .pytest_cache
rootdir: /home/vrinda/Documents/Courses/CS520/Recommender-Companion/app
plugins: anyio-4.6.2.post1, cov-6.0.0
collected 77 items

tests/test.py::test_health_check PASSED                                                                          [  1%]
tests/test.py::test_get_listings_authorized PASSED                                                               [  2%]
tests/test.py::test_add_movie_rating PASSED                                                                      [  3%]
tests/test.py::test_add_book_rating PASSED                                                                       [  5%]
tests/test.py::test_delete_rating PASSED                                                                         [  6%]
tests/test.py::test_update_rating PASSED                                                                         [  7%]
tests/test.py::test_onboarding_status PASSED                                                                     [  9%]
tests/test.py::test_duplicate_ratings PASSED                                                                     [ 10%]
tests/test.py::test_account_management PASSED                                                                    [ 11%]
tests/test.py::test_auth_login PASSED                                                                            [ 12%]
tests/test.py::test_user_logout PASSED                                                                           [ 14%]
tests/test.py::test_global_search PASSED                                                                         [ 15%]
tests/test.py::test_user_items_list PASSED                                                                       [ 16%]
tests/test.py::test_user_progress PASSED                                                                         [ 18%]
tests/test.py::test_get_user_info PASSED                                                                         [ 19%]
tests/test.py::test_search_items PASSED                                                                          [ 20%]
tests/test.py::test_recommendations PASSED                                                                       [ 22%]
tests/test.py::test_recommendations_by_type PASSED                                                               [ 23%]
tests/test.py::test_add_rating_out_of_range PASSED                                                               [ 24%]
tests/test.py::test_listings_pagination_and_filters PASSED                                                       [ 25%]
tests/test.py::test_update_rating_error_cases PASSED                                                             [ 27%]
tests/test.py::test_recommendations_error_cases PASSED                                                           [ 28%]
tests/test.py::test_various_routes_coverage PASSED                                                               [ 29%]
tests/test.py::test_auth_and_user_management PASSED                                                              [ 31%]
tests/test.py::test_auth_endpoints_coverage PASSED                                                               [ 32%]
tests/test.py::test_middleware_paths PASSED                                                                      [ 33%]
tests/test.py::test_routes_edge_cases PASSED                                                                     [ 35%]
tests/test.py::test_error_scenarios PASSED                                                                       [ 36%]
tests/test.py::test_add_duplicate_rating PASSED                                                                  [ 37%]
tests/test.py::test_add_rating_with_invalid_value PASSED                                                         [ 38%]
tests/test.py::test_listings_pagination PASSED                                                                   [ 40%]
tests/test.py::test_add_and_delete_ratings PASSED                                                                [ 41%]
tests/test.py::test_missing_state_in_callback PASSED                                                             [ 42%]
tests/test.py::test_get_user_missing_in_db PASSED                                                                [ 44%]
tests/test.py::test_listings_with_invalid_query_params PASSED                                                    [ 45%]
tests/test.py::test_invalid_token_decoding PASSED                                                                [ 46%]
tests/test.py::test_incomplete_onboarding PASSED                                                                 [ 48%]
tests/test.py::test_missing_email_in_token PASSED                                                                [ 49%]
tests/test.py::test_callback_invalid_state PASSED                                                                [ 50%]
tests/test.py::test_invalid_type_in_listings PASSED                                                              [ 51%]
tests/test.py::test_empty_listings PASSED                                                                        [ 53%]
tests/test.py::test_get_microsoft_signing_keys_success PASSED                                                    [ 54%]
tests/test.py::test_verify_token_signature_invalid_kid PASSED                                                    [ 55%]
tests/test.py::test_verify_token_signature_network_error PASSED                                                  [ 57%]
tests/test.py::test_verify_token_signature_invalid_format PASSED                                                 [ 58%]
tests/test.py::test_callback_missing_code PASSED                                                                 [ 59%]
tests/test.py::test_verify_token_signature_valid_token PASSED                                                    [ 61%]
tests/test.py::test_is_admin_user_with_roles PASSED                                                              [ 62%]
tests/test.py::test_is_admin_user_without_roles PASSED                                                           [ 63%]
tests/test.py::test_is_admin_user_invalid_token PASSED                                                           [ 64%]
tests/test.py::test_get_token_from_code_success PASSED                                                           [ 66%]
tests/test.py::test_get_token_from_code_failure PASSED                                                           [ 67%]
tests/test.py::test_get_token_expiry_valid_token PASSED                                                          [ 68%]
tests/test.py::test_get_token_expiry_invalid_token PASSED                                                        [ 70%]
tests/test.py::test_listings_route_movies PASSED                                                                 [ 71%]
tests/test.py::test_listings_route_invalid_tab_type PASSED                                                       [ 72%]
tests/test.py::test_listings_route_search_query PASSED                                                           [ 74%]
tests/test.py::test_get_user_items_movies_and_books PASSED                                                       [ 75%]
tests/test.py::test_404_error_handler PASSED                                                                     [ 76%]
tests/test.py::test_missing_access_token PASSED                                                                  [ 77%]
tests/test.py::test_is_admin_user_invalid_role PASSED                                                            [ 79%]
tests/test.py::test_get_token_expiry_missing_exp PASSED                                                          [ 80%]
tests/test.py::test_verify_token_signature_invalid_signature PASSED                                              [ 81%]
tests/test.py::test_advanced_search_movies PASSED                                                                [ 83%]
tests/test.py::test_advanced_search_books PASSED                                                                 [ 84%]
tests/test.py::test_pagination_edge_cases PASSED                                                                 [ 85%]
tests/test.py::test_empty_results PASSED                                                                         [ 87%]
tests/test.py::test_ordering_logic PASSED                                                                        [ 88%]
tests/test.py::test_invalid_route_method PASSED                                                                  [ 89%]
tests/test.py::test_empty_query_params PASSED                                                                    [ 90%]
tests/test.py::test_listings_combined_filters PASSED                                                             [ 92%]
tests/test.py::test_missing_required_body_fields PASSED                                                          [ 93%]
tests/test.py::test_invalid_item_type PASSED                                                                     [ 94%]
tests/test.py::test_delete_nonexistent_rating PASSED                                                             [ 96%]
tests/test.py::test_search_query_no_results PASSED                                                               [ 97%]
tests/test.py::test_init_debug_mode PASSED                                                                       [ 98%]
tests/test.py::test_user_interactions_with_system PASSED                                                         [100%]
```
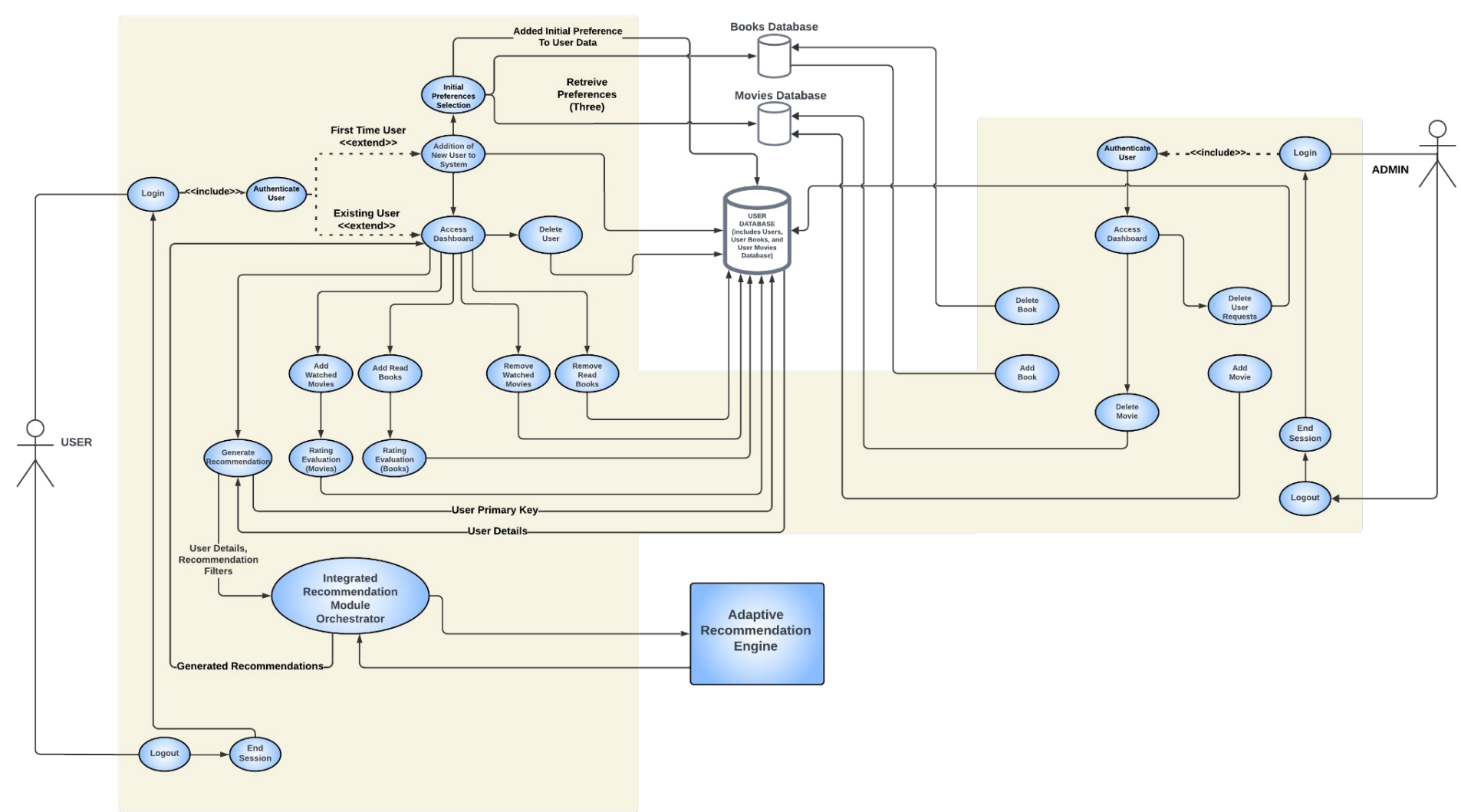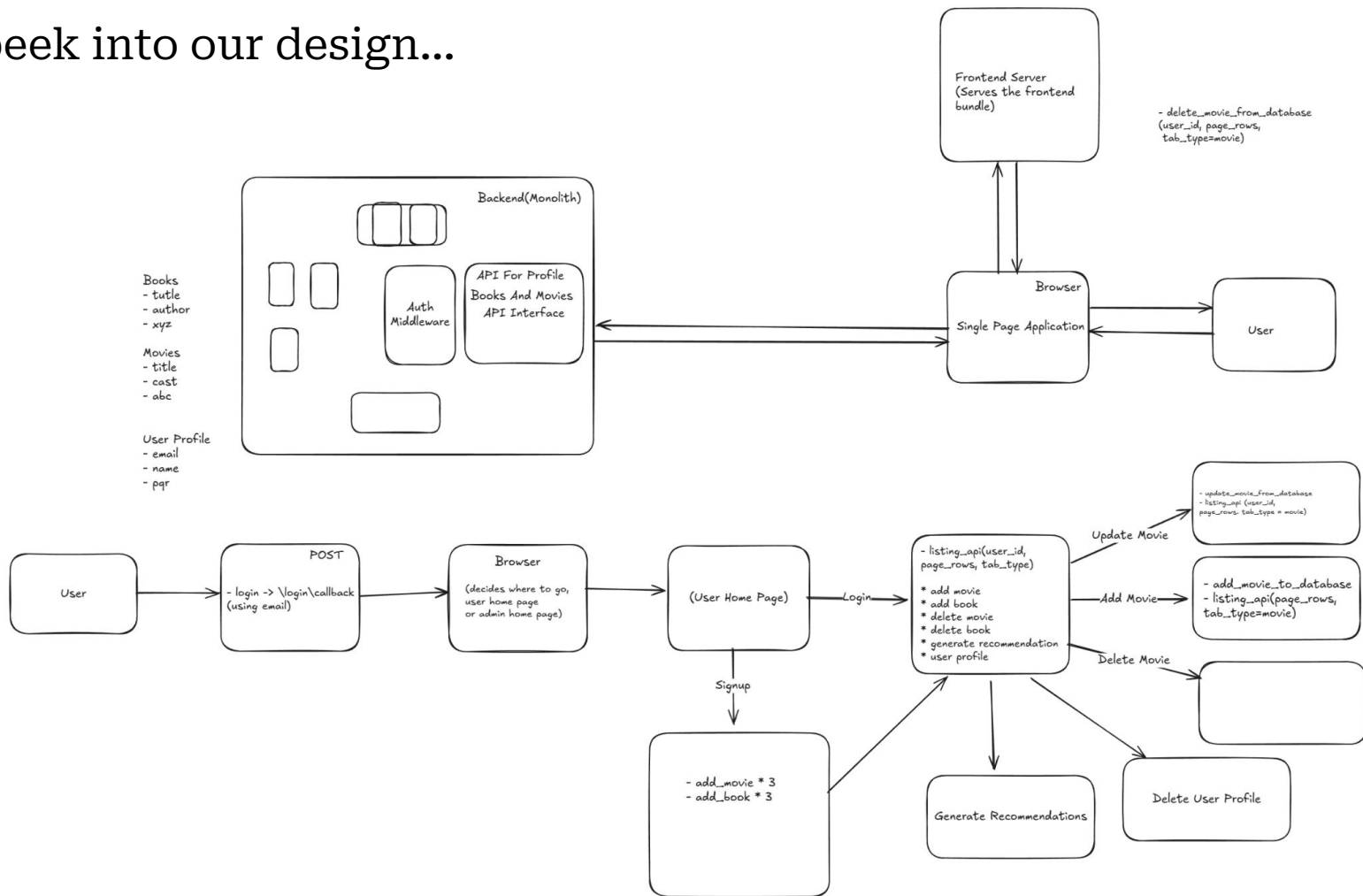
Backend Testing:

The test cases provide **over 90%** coverage. Some files were harder to cover, since their functionalities involved knowing information about GPT's internal tokenisation structure for mocking, which is not well documented.

```
---------- coverage: platform linux, python 3.12.3-final-0 -----------
Name                    Stmts   Miss  Cover
-------------------------------------------
__init__.py                14      0   100%
auth.py                   169     36    79%
config.py                  20      0   100%
extensions.py               2      0   100%
middleware.py              43      8    81%
models.py                  47      2    96%
routes.py                 253     79    69%
tests/__init__.py           0      0   100%
tests/test.py             740      3    99%
-------------------------------------------
TOTAL                    1288    128    90%
```

Diagrams

A sneak peek into our design...

Frontend Server
(Serves the frontend bundle)

- delete_movie_from_database
(user_id, page_rows,
tab_type=movie)

Backend(Monolith)

Books
- tutle
- author
- xyz

Movies
- title
- cast
- abc

User Profile
- email
- name
- pqr

Auth
Middleware

API For Profile
Books And Movies
API Interface

Browser

Single Page Application

User

User

POST

- login -> \login\callback
(using email)

Browser

(decides where to go,
user home page
or admin home page)

(User Home Page)

Login

- listing_api(user_id,
page_rows, tab_type)

* add movie
* add book
* delete movie
* delete book
* generate recommendation
* user profile

Update Movie

- update_movie_from_database
- listing_api (user_id,
page_rows, tab_type = movie)

Add Movie

- add_movie_to_database
- listing_api(page_rows,
tab_type=movie)

Delete Movie

Signup

- add_movie * 3
- add_book * 3

Generate Recommendations

Delete User Profile

Links:

User Manual

Github Link