Team Sea Cream Jasmine

Michelle Chu
James Ooi
Jeng-Rung Tu
Karsten Widjanarko
Yuteng Wu

Introduction

An·i·me : a style of Japanese film and television animation, typically aimed at adults as well as children.

- Anime database website
  - Helps you find the perfect anime you're looking for
  - Shows data analytics and rankings of anime through different filters
- Web application:
  - Frontend: React.JS / Flutter
  - Backend: Django
  - Languages: Python, CSS, Javascript

Dataset

- We decided to use an anime database from Kaggle that web scraped data from MyAnimeList.com
- This data set contains information about 17,562 anime and the preference from 325,772 different users including details like
  - Ratings of animes
  - Information about anime (genre, studio, etc.)
  - Anime list per user (watching, dropped, finished, etc)
- We modified the csv to delete the “adult” anime
- https://www.kaggle.com/hernan4444/anime-recommendation-database-2020?select=animelist.csv

Features – General

- **Edit** existing anime from database
- **Delete** existing anime from database
- **Add** a new anime to the database
- **Backup** local variable data changes to server
- **Import** server changes to local variable data
- Search anime by name
- Search anime by genre
- Search anime by studio
- Search anime by score

Features – Analytics

1. Find the **top 100 most popular anime from selected genre**
2. Find the **top 100 most popular anime from selected type**
3. Find the **average duration for the top 10 anime from all genres**
4. Find the **top 100 anime with the highest completion from selected genre**
5. Find the **top 100 highest average scoring anime from selected studio**
6. Find the **top 100 highest average scoring anime from selected type**
7. Find the **top 100 highest average scoring anime from selected rating**

Demo Time

Major Design – Functions

- **Parser Functions**
  - with open(path, 'r', encoding="UTF-8") as file
    - Using UTF-8 to read the Japanese character
  - generateJson() to convert CSV file into Json local variable

- **Modifying Functions**
  - delete(target_name)
  - update(anime_name, score, ranking, episodes, type, popularity, genre, studio)
  - insert(anime_name, score, ranking, episodes, type, popularity, genre, studio)
  - backup_data()
  - import_data()

Major Design – Analytic Functions

1. **sort_by_score_genre(target_genres)** (Applied IA)
   - return a list of sorted animes by score based on Score
2. **sort_by_score_type(target_type)** (Applied IA)
   - return a list of animes sorted by score based on Type
3. **average_duration_by_top(target_genre)**
   - return a list of animes sorted by Average Duration
4. **sort_by_completion_rate(target_genre)**
   - return the anime has the highest completion rate based on Genre
5. **top_highest_average_anime_by_studio(studio)**
   - return a list of animes sorted by average score based on Studio
6. **top_highest_average_anime_by_type(anime_type)**
   - return a list of animes sorted by average score based on Type
7. **top_highest_average_anime_by_rating(rating)**
   - return a list of animes sorted by average score based on Rating

Major Design – Functions

- **handle_change(action,anime_obj)**
- To speed up the process for our incremental analytics, we decided to save the previous searched/sorted lists in a local map:

```
prev_sorted_data = {
    "genres" : {
        "action": [anime data sorted by score],
        "space": [anime data sorted by score],
    },
    "type" : {
        "TV": [anime data sorted by score],
    },
}
```

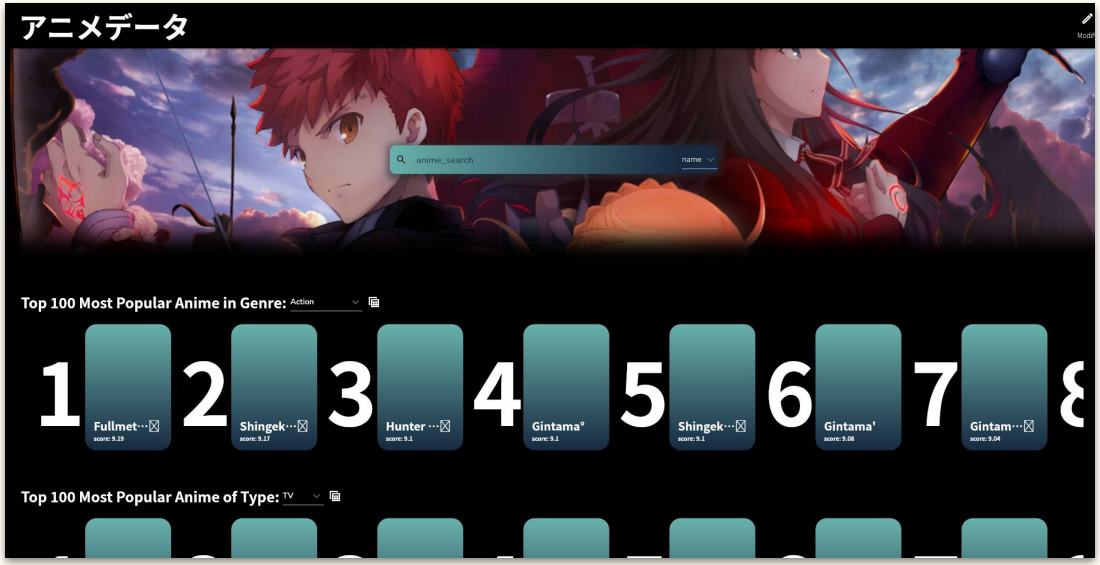
- If there are any changes like edit, insert, delete, we created a function called *handle_change(action,anime_obj)* to handle it where it will just modify one entry in the map

Major Design – Incremental Analytics

- **How we improved performance of analytics**
  - We decided to store the results of the previous searches and previous sorts that we computed into a local map
  - For any future search that uses the same keywords, we can just pull up the results from our local map in constant time
  - When there are any changes to our map, we perform binary search to see where to insert an entry into our sorted list in our local map

Major Design – Polishes after last sprint

- Began porting frontend to Flutter



- Add Table for search results

Major Design – Challenges

- One of the biggest challenges was learning how to do frontend code
  - Learned how to use React.js and Flutter for frontend
  - Learned how to connect frontend with backend
- Another challenge was trying to make the frontend table on the analytics page
  - We solve it by passing in the url parameter and call the table with the given url

Q&A Time