MangaVerse

Feyzan Colak, Flavio Messina, Noemi Cherchi
Large-scale and multi-structured databases project
2023-2024

Contents

Introduction

MangaVerse is a project developed for the Large-scale and multi-structured databases course of the University of Pisa. This web application aims to provide users with a comprehensive platform to explore, search, and interact with a vast collection of manga and anime. Users can register, personalize their profiles, and engage in a community. The platform also offers a set of features for registered users, such as liking media contents, following other users, adding reviews and ratings. Additionally, users receive personalized suggestions based on their preferences and current trends. The site manager has access to detailed analytics about media contents and user activities.

Analysis

Actors

- **Unregistered User**: A visitor who has not created an account on the platform. They can browse media contents, search and filter, view media content details, and register/login.
- **Registered User**: A user who has created an account on the platform. They can perform all the actions of an unregistered user, as well as logout, manage their profile, explore other user profiles, interact with media contents/users, review media contents, and receive advanced recommendations.
- **Manager**: A registered user with administrative features. They can access an analytics dashboard, manage user accounts, content entries, and monitor trends.

Requirements

Unregistered User:

- Browse Media Contents:
  - View a list of available manga and anime on the home page.
  - Access basic details about each media content without logging in.
- Search and Filter:
  - Use the search bar to find specific manga or anime by title.
  - Utilize basic filtering options to refine the media content list.
- View Media Content Details:
  - Click on a media content to view detailed information, including synopsis and genre.
- Register/Login:
  - Access a registration page to create a new account.
  - Use valid credentials to log into the account.
- Explore Features:
  - Access information about the features available to registered users.

Registered User:

- Browse Media Contents:
  - View a list of available manga and anime on the home page.
  - Access basic details about each media content without logging in.
- Search and Filter:
  - Use the search bar to find specific media content by title.
  - Utilize basic filtering options to refine the media content list.
- View Media Content Details:
  - Click on a media content to view detailed information, including synopsis and genre.
- Logout:

  - Ends the user's session.

- Profile Management:
  - Edit and update personal information (e.g., profile picture, bio).
  - Change account password.
- Explore Other User Profiles:
  - View profiles of other registered users.
  - See their liked manga, anime and reviews.
- Interact with Media Contents/Users:
  - Like or dislike manga and anime to indicate preferences.
  - Follow/unfollow other users.
- Review Media Contents:
  - Add reviews and ratings to manga and anime.
  - View and edit own reviews.
- Advanced Recommendations:
  - Receive more refined media content suggestions based on detailed user interactions.
  - Receive users suggestions based on common interests.

Manager(Registered User with Administrative Features):

- Analytics Dashboard:
  - Access a comprehensive analytics dashboard with data on user engagement and media contents trends.
- User Management:
  - View and manage user accounts, including account activation and deactivation.
- Content Management:
  - Manage media content entries, including adding new manga and anime, updating information, and removing entries if necessary.
- Monitor Trends:
  - Monitor trends in user interactions, popular genres, and trending manga.

Non Functional Requirements

Performance

- Response Time: The system should have low latency, with pages loading within an acceptable timeframe.
- Scalability: The system should be able to handle an increasing number of users and data without significant degradation in performance.
- Concurrency: The application should support multiple users simultaneously without performance bottlenecks. For very high traffic scenarios, acceptable delays may be introduced.

Security

- Data Encryption: All user data, including passwords, should be securely encrypted during transmission and storage.

User Interface

- Responsiveness: The user interface should be responsive, providing a consistent and seamless experience across various devices and screen sizes.
- Intuitiveness: The interface should be user-friendly, with clear navigation and easily understandable features.

UML use case diagram

UML class analysis

UML class diagram

Data Modeling

Data Collection

Sources: https://www.kaggle.com/datasets/dbdmobile/myanimelist-dataset?select=users-score-2023.csv, MyAnimeList.net, anilist.com, kitsu.io, livechart.me, anime-planet.com, nofity.moe, anisearch.com, anidb.net

Description: Manga, users and scores datasets were collected from MyAnimeList.net site using the official API and another unofficial API (Jikan). The anime datasets were collected from all the sources.

Variety: The datasets contain a variety of data types, including text, numbers, and dates. Anime are collected from 8 different sources. All the information is collected in 4 different csv files.

Volume: The datasets contain a large volume of data, with thousands of entries for anime, manga, users, and scores. The total size of the datasets is around 3 GB.

Data Cleaning and Preprocessing

Python scripts were used to clean and preprocess the data. The following steps were performed: reviews were created by merging the users and scores datasets, and creating comments about the media contents; the anime dataset was created by putting togethere the diffrerent sources; the manga dataset was created from MyAnimeList.net; the users dataset was cleaned and missing information, like email and password, was added.

Design

The web application needs to handle a big amount of data, so we decided to use a combination of different databases to store and manage the data. We will use a document database to store users, media contents and reviews data, and a graph database to store relationships between users and media content. This will allow us to efficiently store and retrieve data, as well as handle complex relationships between data.

Document Database

For the document database, we will use MongoDB. MongoDB is a NoSQL database that stores data in flexible, JSON-like documents. It is a popular choice for applications that require flexibility and scalability. MongoDB is a document database, which means it stores data in JSON-like documents. These documents are flexible, meaning they can have different fields and structures. This makes MongoDB a good choice for applications that require flexibility in their data model. MongoDB is also a scalable database, meaning it can handle large amounts of data and traffic. It is designed to scale out, meaning you can add more servers to handle more traffic. This makes MongoDB a good choice for applications that need to scale quickly.

Collections The database will have the following collections:

- Anime: This collection will store information about anime, such as titles, tags, and synopsis.
- Manga: This collection will store information about manga, such as titles, genres, and authors.
- Reviews: This collection will store user ratings and comments for media content.
- Users: This collection will store user data, such as usernames, passwords, email addresses, gender and location.

MongoDB document example

Anime:

```
{
  "_id": "65789bb52f5d29465d0abcfb",
  "title": "0",
  "type": "SPECIAL",
  "episodes": 1,
  "status": "FINISHED",
  "picture": "https://cdn.myanimelist.net/images/anime/12/81160.jpg",
  "tags": [
    "drama",
    "female protagonist",
    "indefinite",
    "music",
    "present"
  ],
  "producers": "Sony Music Entertainment",
  "studios": "Minakata Laboratory",
  "synopsis": "This music video tells how a shy girl with a secret love and curiosity...",
  "latest_reviews": [
    {
      "id": "657b301306c134f18884924c",
      "date": "2023-10-03T22:00:00.000+00:00",
      "rating": 4,
      "user": {
        "id": "6577877ce68376234760745c",
        "username": "Tolstij_Trofim",
        "picture": "https://thypix.com/wp-content/uploads/2021/10/manga-profile-picture
    -10..."
      }
    },
  ],
  "anime_season": {
    "season": "FALL",
    "year": 2013
  },
  "average_rating": 6.7,
  "avg_rating_last_update": true,
  "likes": 4
}
```

Manga:

```json
{
  "_id": "657ac61bb34f5514b91ea223",
  "title": "Berserk",
  "type": "MANGA",
  "status": "ONGOING",
  "genres": [
    "Action",
    "Adventure",
    "Award Winning",
    "Drama",
    "Fantasy",
    "Horror",
    "Supernatural"
  ],
  "themes": [
    "Gore",
    "Military",
    "Mythology",
    "Psychological"
  ],
  "demographics": [
    "SEINEN"
  ],
  "authors": [
    {
      "id": 1868,
      "role": "Story & Art",
      "name": "Kentarou Miura"
    },
    {
      "serializations": "Young Animal"
    }
  ],
  "synopsis": "Guts, a former mercenary now known as the \"Black Swordsman,\" is out fo...
    ",
  "title_english": "Berserk",
  "start_date": "1989-08-25T00:00:00.000+00:00",
  "picture": "https://cdn.myanimelist.net/images/manga/1/157897l.jpg",
  "average_rating": 3.33,
  "latest_reviews": [
    {
      "user": {
        "id": "6577877be683762347605ce7",
        "username": "calamity_razes",
        "picture": "https://imgbox.com/7MaTkBQR"
      },
      "date": "2012-12-15T00:00:00.000+00:00",
      "comment": "An insult to the art of manga; avoid at all costs.",
      "id": "657b302206c134f18886f5ef"
    },
  ],
  "anime_season": {
    "season": "FALL",
    "year": 2013
  },
  "average_rating": 6.7,
  "avg_rating_last_update": true,
  "likes": 4
}
```

Reviews:

```
{
  "_id": "657b300806c134f18882f2f1",
  "user": {
    "id": "6577877be68376234760596d",
    "username": "Dragon_Empress",
    "picture": "images/account-icon.png",
    "location": "Columbus, Georgia",
    "birthday": "1987-07-29T00:00:00.000+00:00",
    "rating": 7
  },
  "anime": {
    "id": "65789bbc2f5d29465d0b18b7",
    "title": "Slayers Revolution",
    "date": "2023-07-23T06:27:54.000+00:00",
    "comment": "Above-average quality in animation and soundtrack."
  }
}

```

Users:

```
{
  "_id": "6577877be683762347605859",
  "email": "xdavis@example.com",
  "password": "290cb38a679d5eb68d11b9ea1e21f48234eba6de19f95612dbcb70ce0c7e4e78",
  "description": "Liberating the mind from stress with the power of anime zen.",
  "picture": "https://thypix.com/wp-content/uploads/2021/10/manga-profile-picture-44",
  "username": "Xinil",
  "gender": "Male",
  "birthday": "1985-03-04T00:00:00.000+00:00",
  "location": "Libya",
  "joined_on": "2014-05-29T00:00:00.000+00:00",
  "app_rating": 5,
  "followed": 40,
  "followers": 29
}

```

The field "`app_rating`" is used to know the general satisfaction of the user with the application.

Indexes

We created two indexes in the reviews collection to improve query performance. One for the users id and another for the anime and manga id. This will allow us to quickly retrieve reviews for a specific user or media content.

MongoDB queries

Some of the most important MongoDB queries for analytic and suggestion porpouses.

USERS:

- GetDistribution query to get the user's location, gender, birthday year that gave the highest rating to the application:

```
// Match stage to filter documents where 'criteriaOfSearch' exists
db.collection.aggregate([
    {
        $match: {
            [criteriaOfSearch]: { $exists: true }
        }
    },
    // Project stage to include 'criteriaOfSearch' and 'app_rating' fields
    {
        $project: {
            [criteriaOfSearch]: 1,
            app_rating: 1
        }
    },
    // Group stage to count occurrences of each 'criteriaOfSearch'
    {
        $group: {
            _id: "$" + criteriaOfSearch,
            count: { $sum: 1 }
        }
    },
    // Sort stage to sort documents by 'count' in descending order
    {
        $sort: {
            count: -1
        }
    }
]);

```

Listing 3.1: GetDistribution

ANIME/MANGA:

- GetBestCriteria query, the criteria can be genres, demographics, themes, authors and serialization for manga; tags, producers, studios for anime:

```
db.collection.aggregate([
    // Match stage to filter documents where 'criteria' exists and 'average_rating' is
  not null
    {
        $match: {
            criteria: { $exists: true },
            average_rating: { $ne: null }
        }
    },
    // Unwind stage to deconstruct the 'criteria' array field
    {
        $unwind: "$" + criteria
    },
    // Group stage to calculate the average rating for each criteria
    {
        $group: {
            _id: "$" + criteria,
            criteria_average_rating: { $avg: "$average_rating" }
        }
    },
    // Sort stage to sort documents by 'criteria_average_rating' in descending order
    {
        $sort: {
            criteria_average_rating: -1
        }
    },
    // Skip stage to skip the first 'pageOffset' documents
```

```
        {
            $skip: pageOffset
        },
        // Limit stage to limit the results to 25 documents
        {
            $limit: 25
        }
    ]);

```

Listing 3.2: GetBestCriteria

REVIEWS:

- GetMediaContentRatingByYear query to get the average rating of media content by year:

```
// Match stage to filter documents based on specified conditions
db.collection.aggregate([
    {
        $match: {
            [`${nodeType}.id`]: new ObjectId(mediaContentId),
            rating: { $exists: true },
            date: { $gte: startDate, $lt: endDate }
        }
    },
    // Group stage to group documents by year and calculate the average rating
    {
        $group: {
            _id: { $year: "$date" },
            average_rating: { $avg: "$rating" }
        }
    },
    // Project stage to shape the output documents
    {
        $project: {
            _id: 0,
            year: "$_id",
            average_rating: 1
        }
    },
    // Sort stage to sort documents by year in ascending order
    {
        $sort: { year: 1 }
    }
]);

```

Listing 3.3: GetMediaContentRatingByYear

- SuggestMediaContent query to suggest media content based on common criteria, like birthday or location:

```
db.collection.aggregate([
{
  // Match documents based on a dynamic user criteria
  $match: {
    ["user." + criteriaType]: criteriaValue
  }
},
{
  // Group documents by node type ID and calculate the first title and average rating
  $group: {
    _id: "$" + nodeType + ".id", // Group by the node type's ID
    title: { $first: "$" + nodeType + ".title" }, // Get the first title in the group
    average_rating: { $avg: "$rating" } // Calculate the average rating for the group
  }
},
{
  // Sort the grouped documents by average rating in descending order
  $sort: { average_rating: -1 }
  },
  {
    // Limit the number of results to the page size constant
    $limit: Constants.PAGE_SIZE
  }
]);
```

Listing 3.4: SuggestMediaContent

CRUD operations

- Create: This operation will allow users to create new documents in the database. For example, users can create new reviews for anime and manga.
- Read: This operation will allow users to read documents from the database. For example, users can read information about anime and manga and about other users.
- Update: This operation will allow users to update documents in the database. For example, users can update their reviews for anime and manga, they can also update their own profile, the manager can update media contents.
- Delete: This operation will allow users to delete documents from the database. For example, users can delete their reviews for anime and manga, the manager can delete media contents.

Graph Database

For the graph database, we will use Neo4j. Neo4j is a graph database that stores data in nodes and relationships. It is a popular choice for applications that require complex relationships between data. Neo4j is a graph database, which means it stores data in nodes and relationships. Nodes represent entities, such as users or products, and relationships represent connections between nodes. This makes Neo4j a good choice for applications that require complex relationships between data. Neo4j is also a scalable database, meaning it can handle large amounts of data and traffic. It is designed to scale out, meaning you can add more servers to handle more traffic. This makes Neo4j a good choice for applications that need to scale quickly.

Nodes

The database will have the following nodes:

- User: This node will store information about users, such as id, usernames, and picture.
- Anime: This node will store information about anime, such as id, titles and picture.
- Manga: This node will store information about manga, such as id, titles and picture.

Relationships

The database will have the following relationships:

- LIKE: This relationship will connect users to anime and manga nodes. It will store the date when the user liked the media content.
- FOLLOW: This relationship will connect users to other users.

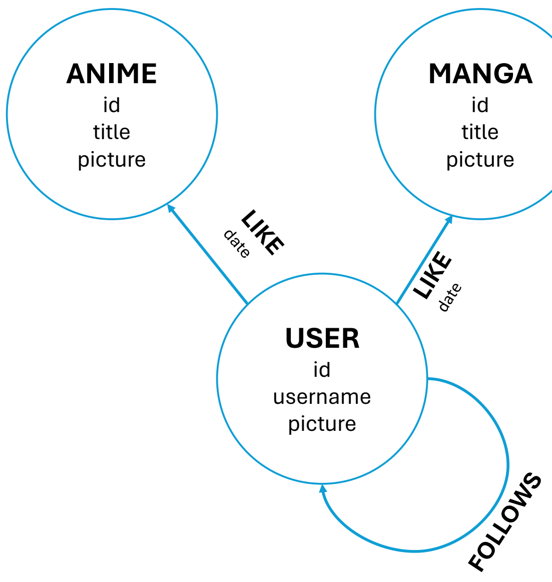

Figure 3.1: GraphDB

GraphDB queries

Some of the most important Neo4j queries for analytic and suggestion porpouses.

USERS:

- Suggest other users followings in common:

```

MATCH (u:User {id: $userId})-[:FOLLOWS]->(following:User)<-[:FOLLOWS]-(suggested:User)
WHERE NOT (u)-[:FOLLOWS]->(suggested) AND u <> suggested
WITH suggested, COUNT(DISTINCT following) AS commonFollowers
WHERE commonFollowers > 5
RETURN suggested as user, commonFollowers
ORDER BY commonFollowers DESC
LIMIT $n
```

Listing 3.5: SuggestUsersByCommonFollowings

- Suggest other users likes in common:

```
  MATCH (u:User {id: $userId})-[r:LIKE]->(media:Manga)<-[:LIKE]-(suggested:User)
  WHERE u <> suggested AND r.date >= $date
  WITH suggested, COUNT(DISTINCT media) AS commonLikes
  WHERE commonLikes > $min
  RETURN suggested AS user, commonLikes
  ORDER BY commonLikes DESC
  LIMIT $n
```

Listing 3.6: SuggestUsersByCommonLikes

ANIME/MANGA:

- Suggest anime and manga based on following in common:

```
  MATCH (u:User {id: $userId})-[:FOLLOWS]->(f:User)-[r:LIKE]->(a:Anime)
  WHERE NOT (u)-[:LIKE]->(a) AND r.date >= $startDate
  WITH a, COUNT(DISTINCT f) AS num_likes
  RETURN a AS anime
  ORDER BY num_likes DESC
  LIMIT $n
```

Listing 3.7: GetSuggestedByFollowings

- Suggest anime and manga based on likes in common between followers:

```
  MATCH (u:User {id: $userId})-[r1:LIKE]->(a:Anime)<-[:LIKE]-(f:User)
  WHERE r1.date >= $startDate
  WITH u, f, COUNT(a) AS common_likes
  ORDER BY common_likes DESC
  LIMIT 20
  MATCH (f)-[:LIKE]->(a2:Anime)
  WHERE NOT (u)-[:LIKE]->(a2)
  WITH a2, COUNT(DISTINCT f) AS num_likes
  RETURN a2 AS anime
  ORDER BY num_likes DESC
  LIMIT $n
```

Listing 3.8: GetSuggestedByLikes

- Get the trend of media contents in a specific year based on the number of likes:

```
  MATCH (a:Anime)<-[r:LIKE]-(u:User)
  WHERE r.date >= $startDate AND r.date < $endDate
  WITH a, count(r) AS numLikes
  ORDER BY numLikes DESC
  RETURN a AS anime, numLikes
  LIMIT $n
```

Listing 3.9: GetTrendMediaContentByYear

- Get the general trend of media contents based on the number of likes:

```
MATCH (u:User)-[r:LIKE]->(a:Anime)
WHERE r.date >= $startDate
WITH a, COUNT(r) AS numLikes
ORDER BY numLikes DESC
RETURN a AS anime, numLikes
LIMIT $n
```

Listing 3.10: GetMediaContentTrendByLikes

CRUD operations

- Create: This operation will allow users to create new nodes and relationships in the database. For example, users can create new relationships between users and media content:

  An user can LIKE a media content:

```
    MATCH (u:User {id: $userId}), (a:Anime {id: $animeId})
    WHERE NOT (u)-[:LIKE]->(a)
    CREATE (u)-[r:LIKE {date: $date} ]->(a)
    RETURN r

```

Listing 3.11: Create Like Relationship

  A user can FOLLOW another user:

```
    MATCH (u:User {id: $userId}), (f:User {id: $followedUserId})
    WHERE NOT (u)-[:FOLLOWS]->(f)
    CREATE (u)-[r:FOLLOWS]->(f)
    RETURN r

```

Listing 3.12: Create Follow Relationship

- Read: This operation will allow users to read nodes and relationships from the database. For example, users can read information about anime and manga and relationships between users and media content. An user can read the list of liked media contents:

```

    MATCH (u:User {id: $userId})-[:LIKE]->(a:Anime)
    RETURN a

```

Listing 3.13: Read Liked Media Contents

  An user can read the list of followers:

```
    MATCH (u:User {id: $userId})<-[:FOLLOWS]-(f:User)
    RETURN f

```

Listing 3.14: Read Followers

- Update: This operation will allow users to update nodes and relationships in the database. For example, users can update their likes for anime and manga and relationships between users.

- Delete: This operation will allow users to delete nodes and relationships from the database. For example, users can delete their likes for anime and manga and relationships between users.

  An user can unlike a media content:

```
    MATCH (u:User {id: $userId})-[r:LIKE]->(a:Anime {id: $animeId})
    DELETE r
    RETURN r

```

Listing 3.15: Delete Like Relationship

An user can unfollow another user:

```
MATCH (:User {id: $followerUserId})-[r:FOLLOWS]->(:User {id: $followingUserId})
DELETE r
RETURN r

```

Listing 3.16: Delete Follow Relationship

Redundancy

The performance of the application is critical, so we need to ensure that the system is highly available and fault-tolerant. To achieve this, we gave priority to fast responses, rather than reducing memory consumption.

Latest reviews

In the anime and manga collections, there's a field containing the latest 5 reviews written for that specific media content, in this way it's fast to retrieve.

Average rating

In the anime and manga collections, there's a field containing the average rating of the media content, this field is updated every time a new review is written.

Number of likes

In the anime and manga collections, there's a field containing the number of likes, this field is updated every time a new like relationship is created or deleted.

Followers and Followings

In the user collection, there are fields containing the number of followers and followings, this field is updated every time a new follow relationship is created or deleted.

User field in Reviews

In the reviews collection, there's a field containing the user data, such as id, username, picture, and also location and birthday, which are used for suggestion porpouses.

Replicas

The performance of the application is critical, so we need to ensure that the system is highly available and fault-tolerant. To achieve this, we will use a replication strategy for both the document and graph databases. This will allow us to have multiple copies of the data, so if one server fails, the system can continue to operate without any downtime.

Implementation

Implementation 1

Implementation 2

Conclusion

Future work