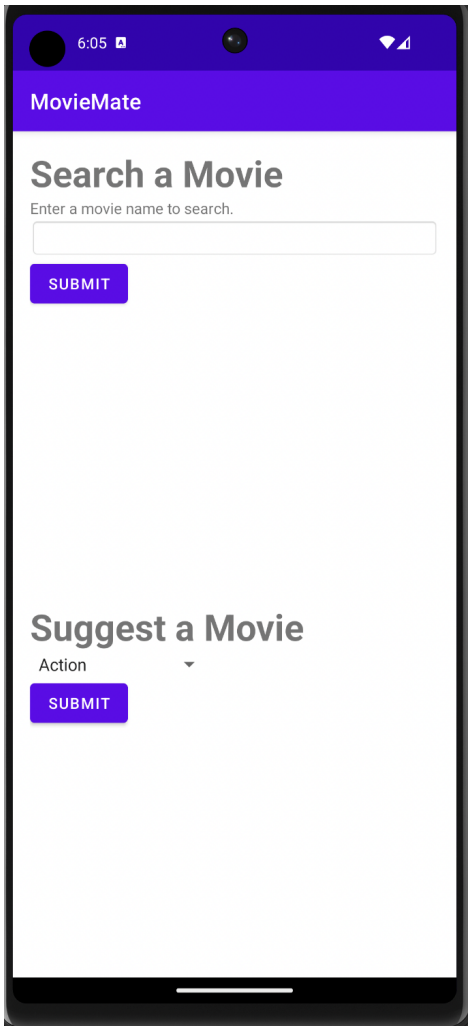
Praveen Ramesh - pramesh2@andrew.cmu.edu

Project4Task2 - MovieMate Android Application

Name : Praveen Ramesh
Andrew ID: pramesh2@andrew.cmu.edu

Description:
MovieMate is an Android app that allows users to search for movie information and receive movie suggestions based on their chosen genre. When searching for a movie, the app displays details such as the movie title, overview, release date, and an official poster. When suggesting a movie, the app provides a movie within the specified genre, accompanied by its poster, title, and average rating.

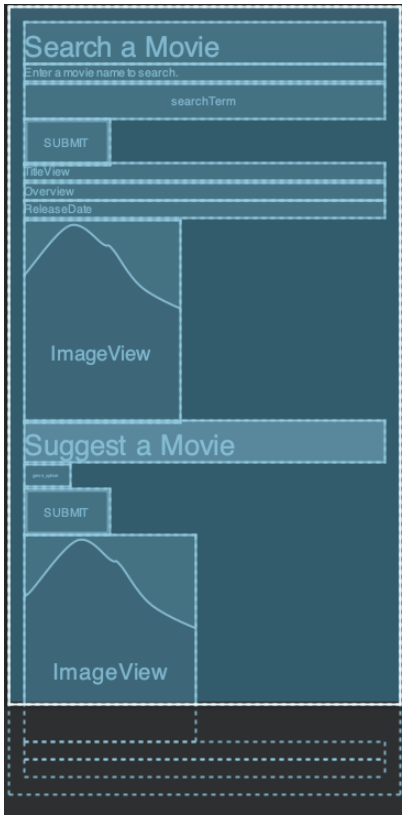
Here is the screenshot of MovieMate application taken from the emulator:



1.Implement a native Android application:

a. Has at least three different kinds of Views in your Layout

MovieMate features a variety of views in its layout, including TextView, EditText, ImageView, Button, and Spinner. These elements are organized within a LinearLayout. Detailed information about each view can be found in the `content_main.xml` file within the Android Studio project.

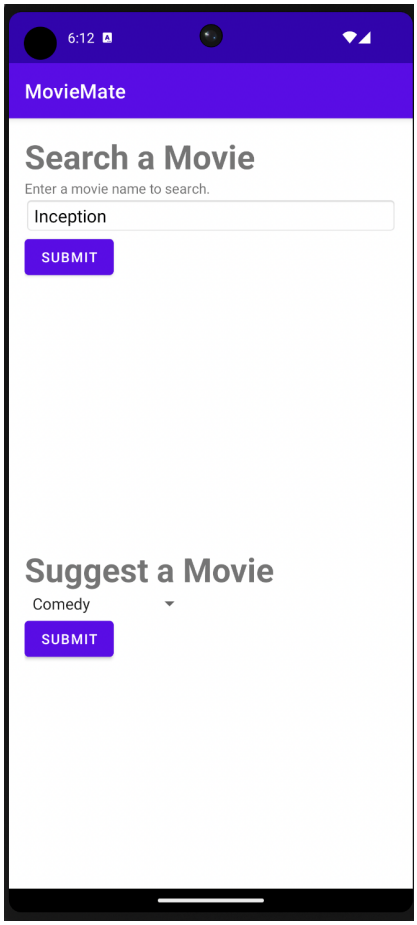


b. Requires input from the user

MovieMate facilitates user input through two methods:

1. **For searching :** Users can enter a movie name using a TextView to search for specific movie information.
2. **For suggestions :** Users can choose a genre from a dropdown list, allowing the app to suggest a movie within the selected genre.

Here is the screenshot for showing the input from the user:



c. Makes an HTTP request (using an appropriate HTTP method) to your web service

MovieMate makes a HTTP GET request either to get the search result or suggestion result from the deployed web service.

Here is the URL for search GET request:

https://pramesh20508-refactored-adventure-667grjxpgg6c5q4q-8080.preview.app.github.dev/MovieMateServlet/search?query=Inception

Here is the URL for suggest GET request:

https://pramesh20508-refactored-adventure-667grjxpgg6c5q4q-8080.preview.app.github.dev/MovieMateServlet/suggest?genre=Action

Snapshot of where the HTTP request is made from the application in GetMovie.java file:

```
private String hitWebServer(URL url){
    try {
        //create a HttpURLConnection
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        //set the request method to GET
        conn.setRequestMethod("GET");
        //set the request property to accept application/json
        conn.setRequestProperty("Accept", "application/json");
        //get the responseText from getResponseBody method
        String responseText = getResponseBody(conn);
        //returns the responseText
        return  responseText;
      //catch clause
    } catch (ProtocolException e) {
        return null;
    } catch (MalformedURLException e) {
        return null;
    } catch (UnsupportedEncodingException e) {
        return null;
    } catch (IOException e) {
        return null;
    }
}
```

d. Receives and parses an XML or JSON formatted reply from your web service:

JSON response for search a movie:

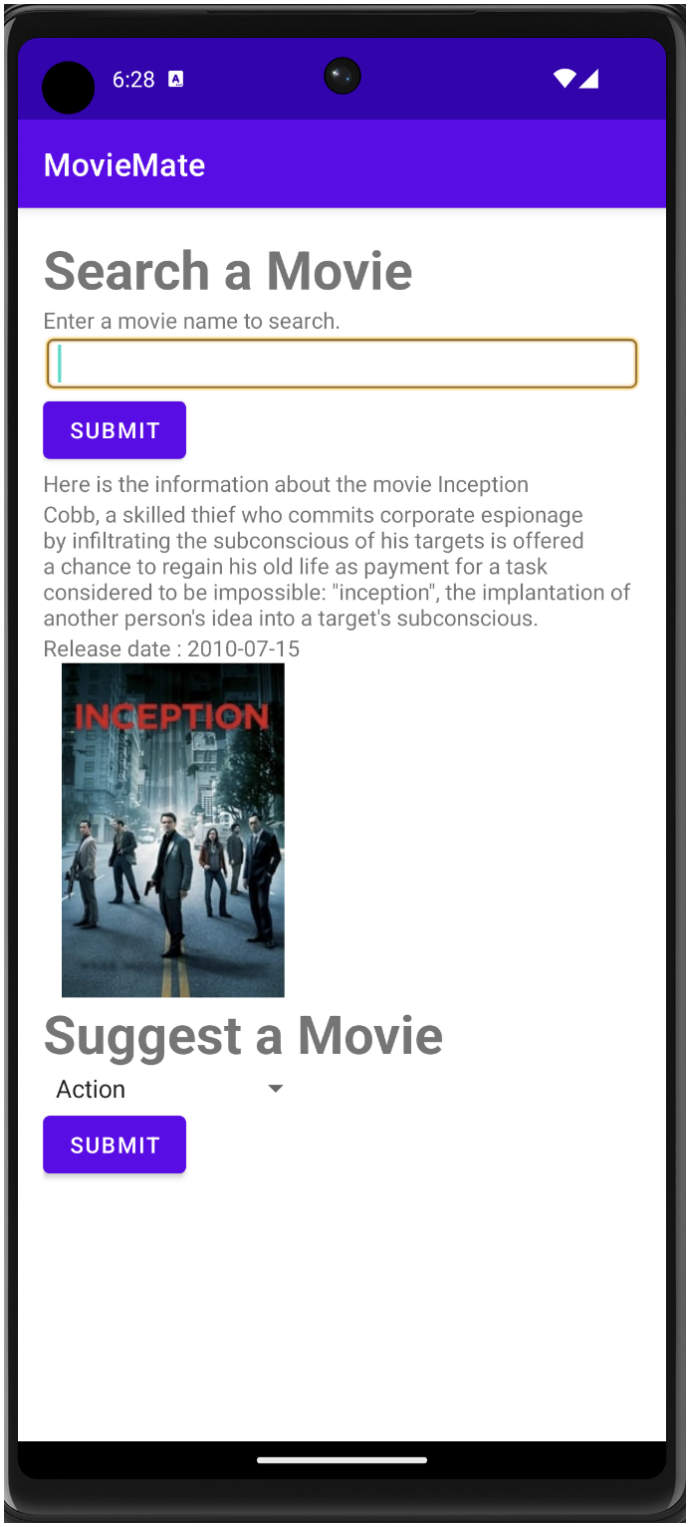

```
{
  "title": "Inception",
  "poster_path": "/edv5CZvWj09upOsy2Y6IwDhK8bt.jpg",
  "vote_average": 8.361,
  "release_date": "2010-07-15",
  "overview": "Cobb, a skilled thief who commits corporate espionage by infiltrating the subconscious of his targets is offered a chance to regain his old life as payment for a task considered to be impossible: \"inception\", the implantation of another person's idea into a target's subconscious."
}
```


JSON response for suggest a movie:

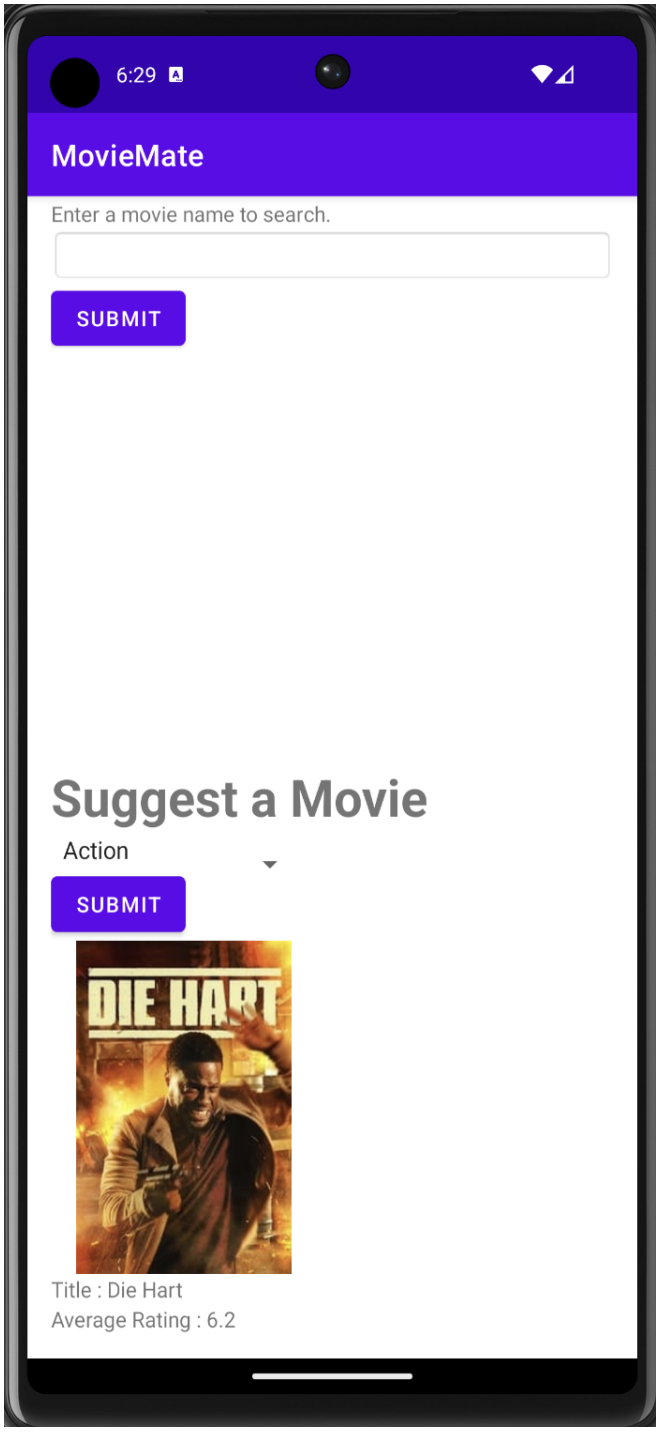

```
{
  "title": "Black Panther: Wakanda Forever",
  "poster_path": "/sv1xJUazXeYqALzczSZ3O6nkH75.jpg",
  "vote_average": 7.3
}
```

e. Displays new information to the user

Output of searching a movie: (Title, overview, release date and poster are shown)

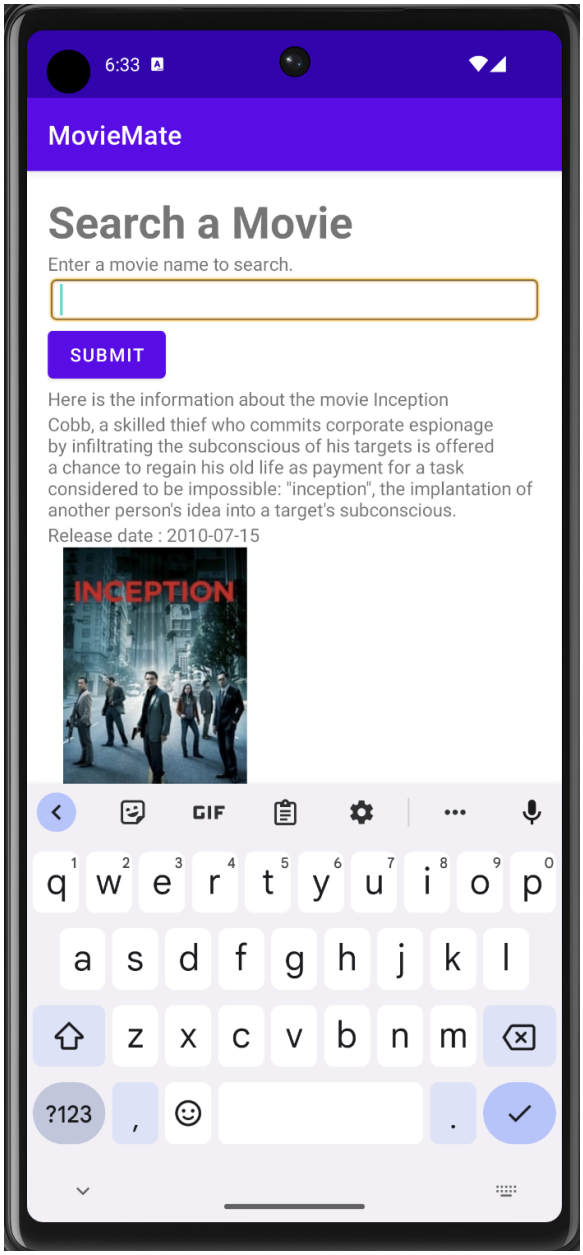


Output of suggesting a movie: (Poster, title, and rating are displayed)



f. Is repeatable (I.e. the user can repeatedly reuse the application without restarting it.)

Yes, user can repeatedly query or ask for suggestions without restarting the application



2. Implement a web service

The webservice is created using HTTPServlet class and is deployed in the codespace using a docker image.

The URL of the webservice: (The default page is Dashboard analytics page)
https://pramesh20508-refactored-adventure-667grjxpgg6c5q4q-8080.preview.app.github.dev

The project directory name is Project4WebService.

a. Implement a simple (can be a single path) API.

The screenshot below demonstrates the implementation of two API request; one for searching and one for providing suggestions

```
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    //requestPath gets the request path
    String requestPath = request.getPathInfo();
    //get the requestTimestamp
    LocalDateTime requestTimestamp = LocalDateTime.now();
    //get the path to redirect it to the required endpoint
    if (requestPath == null || requestPath.equals("/")) {
        //if request path is null call showDashboard to display the dashboard
        showDashboard(request, response);
    }
    //if requestPath start with search call searchMovie
    else if (requestPath.startsWith("/search")) {
        //Initialize the query
        String query = request.getParameter( s: "query");
        //Create a document to store the logs
        //eventType is search
        movieMateDocument = new Document("eventType", "search")
                .append("requestElement", query)
                .append("requestTimestamp", requestTimestamp.toString())
                .append("mobileModel",request.getHeader( s: "User-Agent"));
        //call searchMovie method
        searchMovie(response, query);
    }
    //if requestPath start with search call suggestMovie
    else if(requestPath.startsWith("/suggest")){
        //Initialize the genreString
        String genreString= request.getParameter( s: "genre");
        //Create a document to store the logs
        //eventType is suggestion
```

ol ▶ Run TODO Problems Terminal Services Profiler Dependencies

b. Receives an HTTP request from the native Android application

The web server receives an HTTP request with the argument “search” for searching and the passes the query to the searchMovie method

The webserver also receives an HTTP request with the argument “suggest” for suggesting a movie and passes the genre for suggestion to suggestMovie method in `MovieMateServlet.java` file

Endpoint for dashboard is “/MovieMateServlet”

c. Executes business logic appropriate to your application. This includes fetching XML or JSON information from some 3rd party API and processing the response.

Business logic is written in the MovieMateServlet.java file using two main endpoints which calls the third party TMDB API for movie searching and suggestion. It also makes use of Genre.java class to map the genre names with genre code of TMDB.

Searching logic using TMDB search API : (Please refer to MovieMateServlet.java searchMovie() method for full code)

```
public void  searchMovie(HttpServletResponse response, String query) {
    try {
        //create an UTF-8 encodedQuery
        String encodedQuery = URLEncoder.encode(query,  enc: "UTF-8");
        encodedQuery = encodedQuery.replace( target: "+",  replacement: "%20");
        //Create a TMDB URL for searching a movie
        URL url = new URL( spec: "https://api.themoviedb.org/3/search/movie?api_key=" + TMDB_key + "&language=en-US&query=" + encodedQuery);
        //get the start time
        long startTime = System.currentTimeMillis();
        //get the responseText by calling doThirdPartyAPICall with the URL
        String responseText = doThirdPartyAPICall(url);
        //set the endTime
        long endTime = System.currentTimeMillis();
        // set the latency
        long latency = endTime - startTime;
        //declare a GSON object
        Gson gson = new Gson();
        System.out.println(responseText);
        //convert gson to JsonObject
        JsonObject jsonObject = gson.fromJson(responseText.toString(), JsonObject.class);
        //search for results and parse it to JsonArray
        JsonArray resultsArray = jsonObject.getAsJsonArray( memberName: "results");
        //send the response only when there is atleast one element in resultsArray
        if (resultsArray.size() != 0) {
            //get the topmost search result - TMDB sorts it according to the match percentage with the input query
            JsonObject firstResult = resultsArray.get(0).getAsJsonObject();
            //create a responseJsonObject object
            JsonObject responseJsonObject = new JsonObject();
            //add title to responseJsonObject and sets only the required information to the response JSON
            responseJsonObject.addProperty( property: "title", firstResult.get("title").getAsString());
            //add poster_path to responseJsonObject
            responseJsonObject.addProperty( property: "poster_path", firstResult.get("poster_path").getAsString());
```

Suggesting logic using TMDB suggest API : (Please refer to MovieMateServlet.java suggestMovie() method):

```java
/**
 searchMovieSends does a GET request to TMDb API to retrieve a list of popular movies that belong to the genre
 from the request parameter, then randomly selects one movie from the list and returns the selected movie's title,
 poster path, and vote average in JSON format. Similar to seaarchMovie methods it stores the log to the mongoDB
 using the insertMovieMateDocument helper method
@param response the HttpServletResponse object
@param genreString the genre string used to suggest a movie
*/
1 usage
public void suggestMovie(HttpServletResponse response, String genreString){
        try {
            //Get the corresponding genreCode from the getGenreMap method in genre
            String genreCode = genre.getGenreMap().get(genreString).toString();
            //Initialize the UTF encoded encodedgenreCode
            String encodedgenreCode = URLEncoder.encode(genreCode, enc: "UTF-8");
            encodedgenreCode = encodedgenreCode.replace( target: "+", replacement: "%20");
            //Initialise the TMDB URL to suggest a movie based on a genre
            URL url = new URL( spec: "https://api.themoviedb.org/3/discover/movie?api_key="+TMDB_key+"&language=en-US&sort_by=popularity.desc&include_adult=false&p
            //Initialze the start time
            long startTime = System.currentTimeMillis();
            //call the doThirdPartyAPICall method to get the JSON response from TMDB
            String responseText = doThirdPartyAPICall(url);
            //Initialze the endTime
            long endTime = System.currentTimeMillis();
            //Calculate the latency
            long latency = endTime - startTime;
            //Initialize the GSON object
            Gson gson = new Gson();
            System.out.println(responseText);
            //Parse the responseText to JsonObject
            JsonObject jsonObject = gson.fromJson(responseText.toString(), JsonObject.class);
            //search for results and parse it to JsonArray
```

d. Replies to the Android application with an XML or JSON formatted response. The schema of the response can be of your own design.

As mentioned earlier, the JSON response are sent to the android application only with the required data:

JSON response for search a movie: (Only the required data for searching)


```
{
  "title": "Inception",
  "poster_path": "/edv5CZvWj09upOsy2Y6IwDhK8bt.jpg",
  "vote_average": 8.361,
  "release_date": "2010-07-15",
  "overview": "Cobb, a skilled thief who commits corporate espionage by infiltrating the subconscious of his targets is offered a chance to regain his old life as payment for a task considered to be impossible: \"inception\", the implantation of another person's idea into a target's subconscious."
}
```

JSON response for suggest a movie:: (Only the required data for suggesting)


```
{
  "title": "Black Panther: Wakanda Forever",
  "poster_path": "/sv1xJUazXeYqALzczSZ3O6nkH75.jpg",
  "vote_average": 7.3
}
```


4. Log useful information

The following log informations are stored for each android user interaction with the web service:

1. Event Type(search or suggest) - Whether the GET request is for search or suggest service
2. Request Element - Movie title for a search request or Genre for a suggest request
3. Request timestamp - Time of the request made by the user
4. Mobile Model - User agent of the device making the request is stored
5. Latency - Latency of the service provided by the web server is stored
6. Title - Title of the searched movie or suggested movie is stored
7. Poster path - The path of the movie poster is stored for both search and suggest request
8. Voting Average - Rating of the searched or suggested movie is stored
9. Release date - Release date of the searched movie is stored whereas for suggestions the release dates are stored as empty string
10. Overview - Overview about the searched movie is stored for search request whereas the overview is stored as empty for suggest request

    NOTE: Release date and overview is only stored for search request and for suggest requests it's stored as empty string

These logs provide useful information for further analytics that can be performed with the MovieMate web service.

For instance, the latency is used to compute average latency for operational analytics. Similarly, Request Element is used to get the top searched movie and top suggestions asked by the user.

5. Store the log information in a database

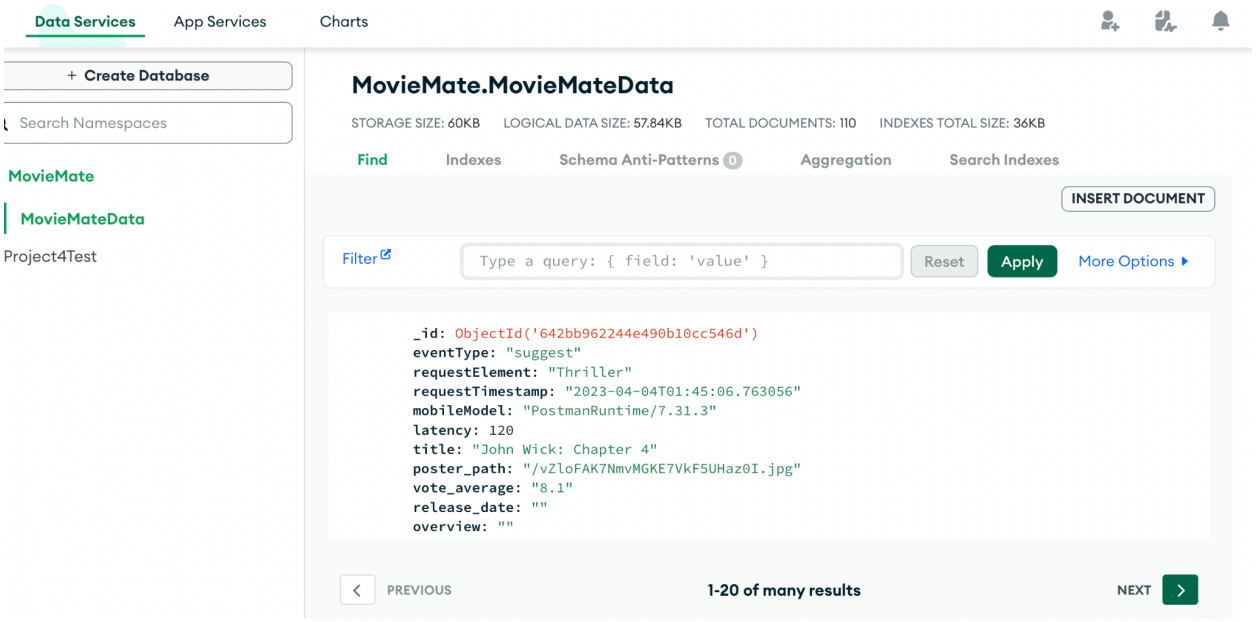
MongoDB Atlas connection string:

"mongodb://pramesh2:pramesh2@ac-gx9xm3y-shard-00-00.qlzkwa8.mongodb.net:27017,ac-gx9xm3y-shard-00-01.qlzkwa8.mongodb.net:27017,ac-gx9xm3y-shard-00-02.qlzkwa8.mongodb.net:27017/test?w=majority&retryWrites=true&tls=true&authMechanism=SCRAM-SHA-1"

The documents are inserted to the database collection “MovieMateData” using the MongoDB.java class. Please refer to the class for the document insertion code.

The logs are stored in MongoDB database called “MovieMate” and the collection name is “MovieMateData”

Screenshot of documents stored in collection of MongoDB:



6. Display operations analytics and full logs on a web-based dashboard

The logic for operational analytics of MovieMate can be found in DashBoardAnalytics.java class. All the three operational metrics - average latency, top searched movie and top suggestions asked by the user are calculated in three separate methods in the class.

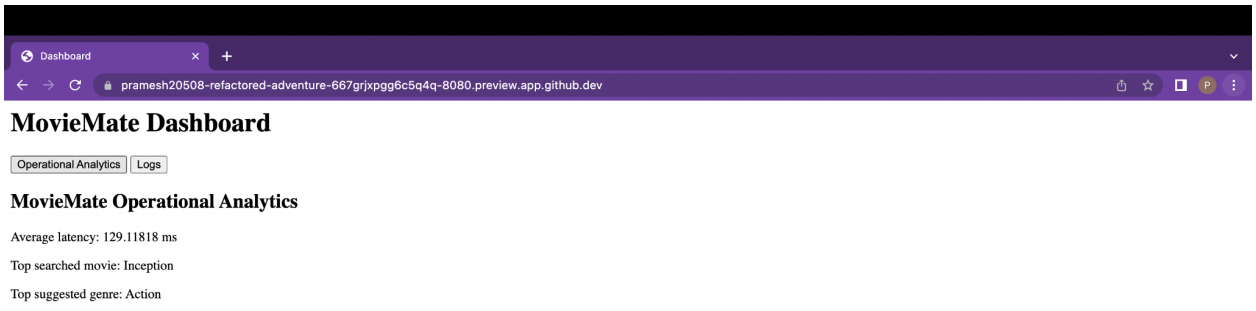
a. A unique URL addresses a web interface dashboard for the web service.

Dashboard Endpoint - /MovieMateServlet
URL -
https://pramesh20508-refactored-adventure-667grjxpgg6c5q4q-8080.preview.app.github.dev

Or

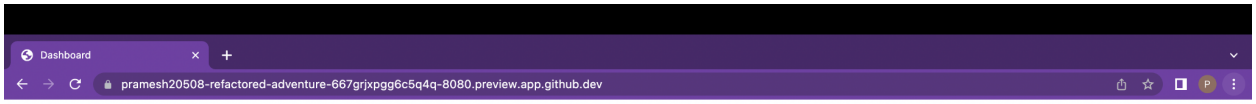
https://pramesh20508-refactored-adventure-667grjxpgg6c5q4q-8080.preview.app.github.dev/MovieMateServlet

b. Screenshot of MovieMate's Operational Analytics:



c. Screenshot of MovieMate's logs:

NOTE: All the logs are displayed in the ascending order of Request Timestamp. Latest request can be found at the bottom of the table.



MovieMate Dashboard

Operational Analytics | Logs

MovieMate Logs

Event Type	Request Element	Request Timestamp	Mobile Model	Latency	Title	Poster Path	Vote Average	Release Date	Overview
suggest	Thriller	2023-04-04T01:45:06.763056	PostmanRuntime/7.31.3	120	John Wick: Chapter 4	/vZloFAK7NmvMGKE7VkF5UHaz0I.jpg	8.1		
search	Gravity	2023-04-04T01:46:18.405129	Dalvik/2.1.0 (Linux; U; Android 13; sdk_gphone64_arm64 Build/TE1A.220922.021)	173	Gravity	/wmUeEacsFZzDndaeOtNNmy26rYJ.jpg	7.16	2013-10-03	Dr. Ryan Stone, a brilliant medical engineer on her first Shuttle mission, with veteran astronaut Matt Kowalsky in command of his last flight before retiring. But on a seemingly routine spacewalk, disaster strikes. The Shuttle is destroyed, leaving Stone and Kowalsky completely alone-tethered to nothing but each other and spiraling out into the blackness of space. The deafening silence tells them they have lost any link to Earth and any chance for rescue. As fear turns to panic, every gulp of air eats away at what little oxygen is left. But the only way home may be to go further out into the terrifying expanse of space.
suggest	Western	2023-04-04T01:46:27.234095	Dalvik/2.1.0 (Linux; U; Android 13; sdk_gphone64_arm64 Build/TE1A.220922.021)	115	The Good, the Bad and the Ugly	/bX2xnavhMYjWDoZp1VM6VnU1xwe.jpg	8.5		
search	master	2023-04-04T01:55:28.880362	Dalvik/2.1.0 (Linux; U; Android 13; sdk_gphone64_arm64 Build/TE1A.220922.021)	192	Master	/gxbwRHsQ2v6DQv28ttp7pIx7Utj.jpg	5.3	2022-03-18	At an elite New England university built on the site of a Salem-era gallows hill, two black women strive to find their place. Navigating politics and privilege, they encounter increasingly terrifying manifestations of the school's haunted past��? and present.
suggest	Animation	2023-04-04T01:56:11.011060	Dalvik/2.1.0 (Linux; U; Android 13; sdk_gphone64_arm64 Build/TE1A.220922.021)	101	Minions: The Rise of Gru	/wKiOkZTN9lUUUNZLmtnwubZYONg.jpg	7.5		
search	inception	2023-04-04T03:05:54.888585	Dalvik/2.1.0 (Linux; U; Android 13; sdk_gphone64_arm64 Build/TE1A.220922.021)	159	Inception	/edv5CZvWj09upOsy2Y6IwDhK8bt.jpg	8.362	2010-07-15	Cobb, a skilled thief who commits corporate espionage by infiltrating the subconscious of his targets is offered a chance to regain his old life as payment for a task considered to be impossible: "inception", the implantation of another person's idea into a target's subconscious.
search	inception	2023-04-04T03:21:42.119298	Dalvik/2.1.0 (Linux; U; Android 13; sdk_gphone64_arm64 Build/TE1A.220922.021)	150	Inception	/edv5CZvWj09upOsy2Y6IwDhK8bt.jpg	8.362	2010-07-15	Cobb, a skilled thief who commits corporate espionage by infiltrating the subconscious of his targets is offered a chance to regain his old life as payment for a task considered to be impossible: "inception", the implantation of another person's idea into a target's subconscious.