Project 4

Name: Yi Guo

Email Address: yiguo@andrew.cmu.edu

Description

My application takes a search string from the user, and uses it to fetch book details from the Open Library API. The fetched information includes the picture of the book cover and book properties like title, authors, number of pages, ISBN10, publishers, publish date and book description. The cover picture and book information will be displayed in the Android App to the user.

API Documentation: https://openlibrary.org/developers/api

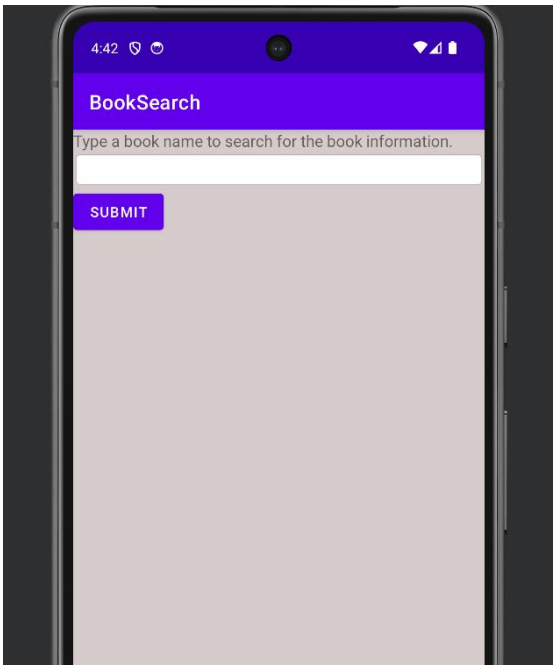
1. Implement a native Android application

The name of my native Android application project in Android Studio is BookSearchApp.

a. Has at least three different kinds of Views in your Layout (TextView, EditText, ImageView, or anything that extends android.view.View).

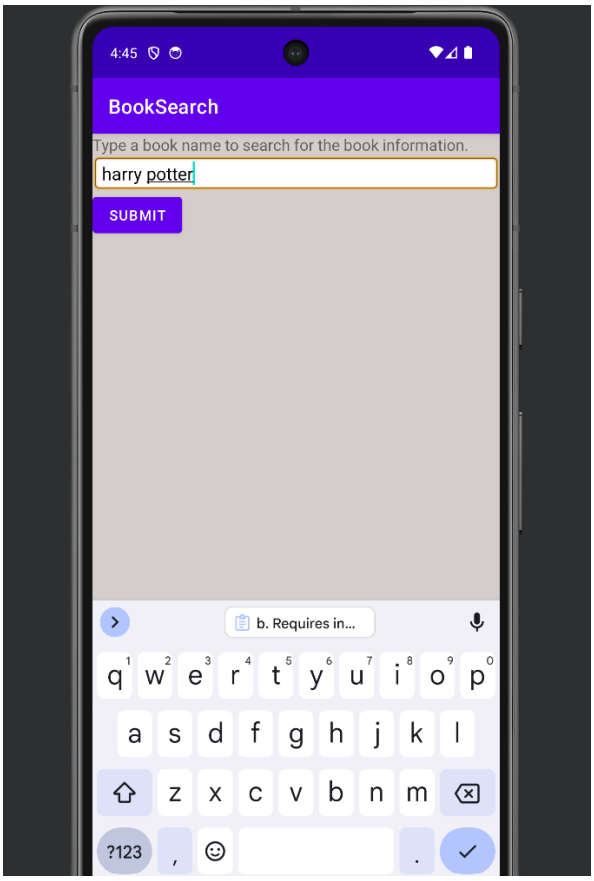
My application uses TextView (for displaying instructions and book details), EditText (for allowing user to prompt search word), ImageView (for displaying book cover) and Button (for submitting search terms).

Here is a screenshot of the layout.



b. Requires input from the user

Here is a screenshot of user searching for a book named harry potter



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

My application does an HTTP GET request in GetBook.java. The HTTP request is: “https://silver-winner-5gqp7gr4qjgp276g5-8080.app.github.dev/getBooks ?searchTerm=” + searchTerm + "&device=" + (Build.MANUFACTURER + " " + Build.DEVICE)

where searchTerm is the user’s input used for search and the device is automatically identified when user clicks submit. It is identified using Build.MANUFACTURER and Build.DEVICE.

The site to query in codespace may change if the space is restarted.

The search method makes this request to my web application and receives the returned JSON string.

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

An example of the JSON formatted reply is {"title":"Harry Potter and the Sorcerer's Stone","authors":"J. K. Rowling","numberOfPages":150,"ISBN10":"8186775609","publishers":"Manjul Publishing House","publishDate":"","picture":"https://covers.openlibrary.org/b/id/8267065-L.jpg","description":""}

Note: when the third-party API does not have certain information such as the publish date and description, the value of them will be set to an empty string. They will not be displayed in the front end if there is no such information.

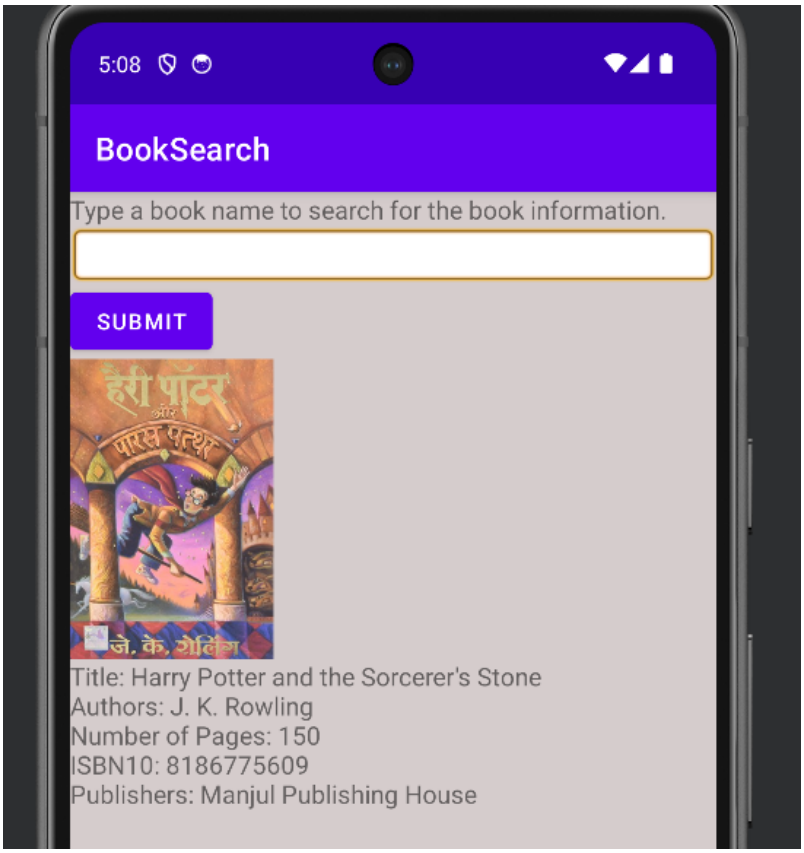
The JSON will later be parsed into a Book class object using a parseJSON method in GetBook.java. The method will also call the getRemoteImage method to get a bitmap representation of the picture if there exists a picture of the book cover.

```
public class Book {
    4 usages
    String title;
    4 usages
    String authors;
    4 usages
    String numberOfPages;
    4 usages
    String ISBN10;
    4 usages
    String publishers;
    4 usages
    String publishDate;
    4 usages
    String description;
    2 usages
    String picture;
}
```

e. Displays new information to the user

Here is a screenshot after the book information has been returned.

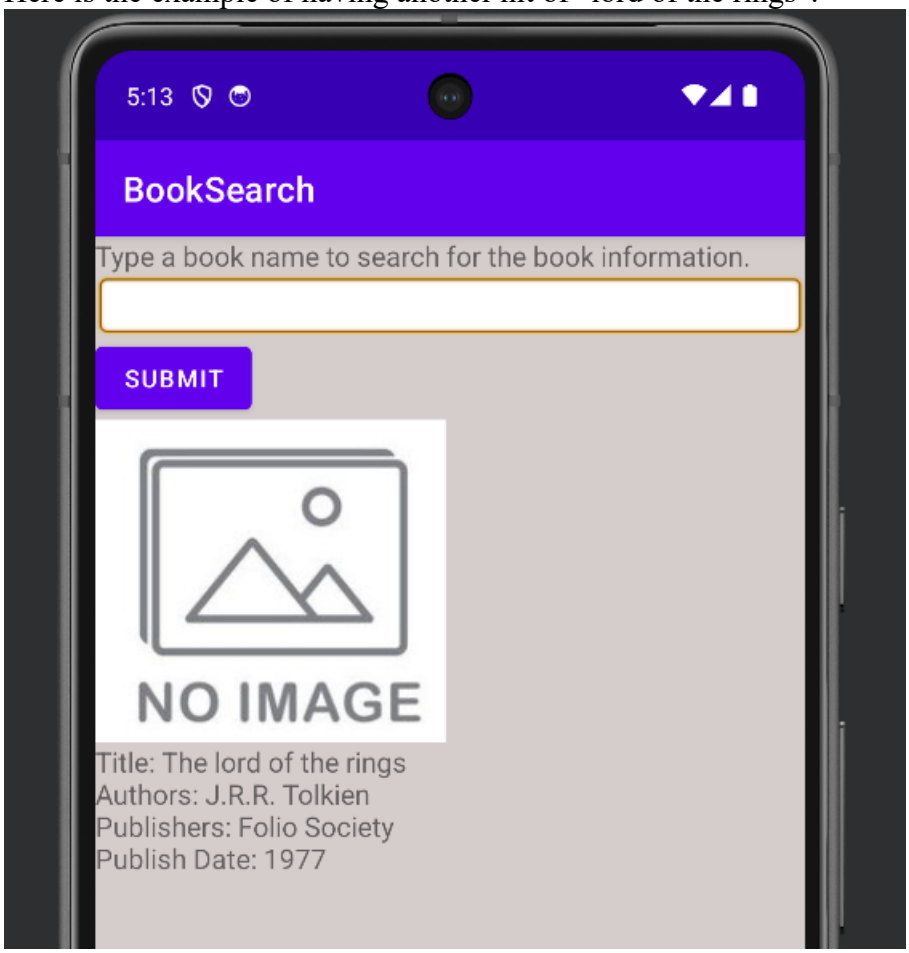
Note: It may take some time for the App to return and search before displaying information. Please wait patiently.



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

The user can type in another search term and hit Submit button again. After each search, the EditText view will be set blank and it can accept new terms.

Here is the example of having another hit of “lord of the rings”.



Note: If there is no image of the book cover in the Open Library API, the user will see a default NO IMAGE picture. (the resource is placed in the res/drawable)

2. Implement a web service

The web service is called BookService.

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

The web service implements the Open Library API and uses its search API and book details API to get the information.

API Documentation: https://openlibrary.org/developers/api

b. Receives an HTTP request from the native Android application
BookServlet.java receives the HTTP GET request from the BookSearchApp with the parameter searchTerm and device. The doGet method is override. It passes the search term to the model function and device is used for logging information.

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

The logic of searching is that the service will first call the searchID method in the BookModel.java to search using the Open Library API. It gets the top one result of search and then randomly selects one from the top 100 editions of it.
Then it returns an Open Library ID which is then fed into the searchDetails method. It searches for the selected book details and description using the ID just found.

For search:

```
String searchURL = "https://openlibrary.org/search.json?title=" + searchTag;
```

For book details:

```
String searchURL = "https://openlibrary.org/api/books?&bibkeys=OLID:" + ID + "&jscmd=data&format=json";
```

For description:

```
searchURL = "https://openlibrary.org/works/" + ID + ".json";
```

Use these endpoints, the service will get JSON format reply. To process the book details, if there is no information found in the JSON file, the string will be set to empty. Also, I only fetch useful information that will be sent back to the mobile app for displaying, including title, authors, number of pages, ISBN10, publishers, publish date, book cover picture and book description. Other information will be discarded.

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

The information of the book fetched using the logic above will be formatted into a new JSON object and the JSON formatted string is sent back to the Android application.

An example of the JSON formatted response is {"title":"Harry Potter and the Sorcerer's Stone","authors":"J. K. Rowling","numberOfPages":150,"ISBN10":"8186775609","publishers":"Manjul Publishing House","publishDate":"","picture":"https://covers.openlibrary.org/b/id/8267065-L.jpg","description":""}

It selects and passes only title, authors, number of pages, ISBN10, publishers, publish date, book cover picture and book description data.

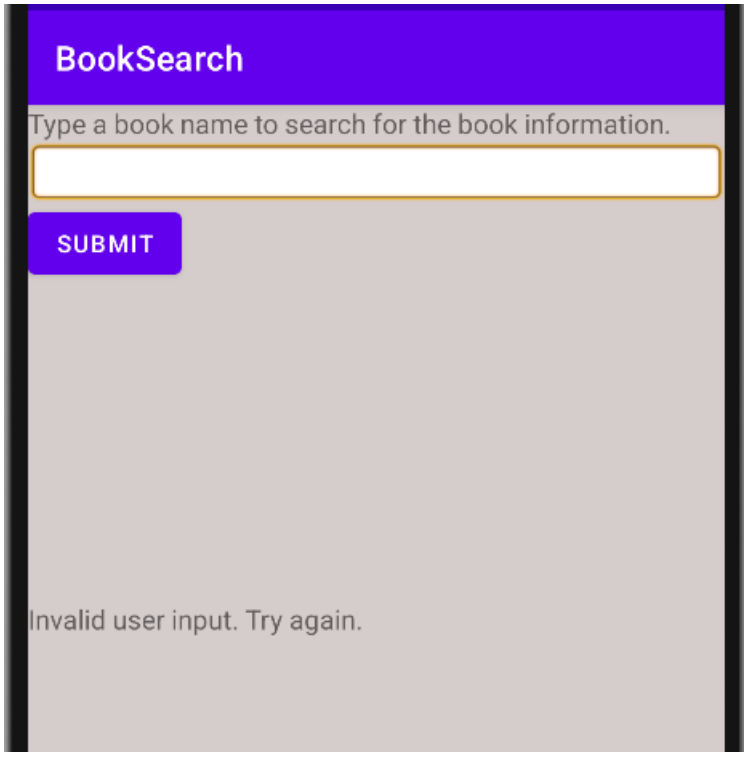
3. Handle error conditions

The application handles:

1. Invalid mobile app input

   This is handled in the android application. If the user gives an empty string of a string will all blank spaces, the app will give “Invalid user input. Try again.”

   Upper case string will be transformed into lower case and then be split using blank spaces to get them into words. This input will not be sent to web service and not be logged.



2. Invalid server-side input.
   This is handled in the Servlet side in BookServlet.java. It will also check with the search term to see if it is empty or with all blank spaces.
   If yes, it will reply with Invalid server-side input error message. This will not be logged.

3. Mobile app network failure, unable to reach server
   This is handled in the Android application. It will try to make HTTP GET request, if this fails or we get code other than 200, the application will display “Connection error.”.

4. Third-party API unavailable.
   This is handled in the BookModel.java. If the fetch method that get the third-party API contents returns an empty string, it shows that the third-party API is not available. This will be logged and displayed to the application.

5. Third-party API invalid data.
   This is handled in the BookModel.java. If using the common structure to parse JSON information from the API fails, it shows that third-party API has invalid data. This will be logged and displayed to the application.

6. No result found.
   If we can not search for any results using the search term, the servlet will give responses “No result found.” This will be logged and displayed to the application.

4. Logging Useful Information

6 pieces of information are logged for each request/reply with the mobile phone. This includes:

1. **Search term:** the search term entered by the user (in lower case). [related to information about request from the mobile phone]

2. **Request Device:** the android device sending the search request. [related to information about request from the mobile phone]

3. **Mobile Request Time:** time when the Android app makes the search request [related to information about request from the mobile phone]

4. **API Request Site:** the primary site to make request to get book details [related to information about the request to the 3rd party API]

5. **Search Time:** The time used for making requests to API and getting responses from API. [related to information about the request and reply to the 3rd party API]

6. **Search Result**: The JSON formatted reply sent back to the Android application. [related to information about the reply to the mobile phone]

This meets with the standards of 6 pieces of information. It includes information about the request from the mobile phone, information about the request and reply to the 3rd party API, and information about the reply to the mobile phone.

I choose to log these because logging search term helps me understand my user groups and what books users care most about. They can be used for further analysis. Request time is like an entry point to the web service. The API request site gives information about which API endpoint is used for fetching details. We could go to the site to check if the app is parsing and getting information currently. Search Time is of concern because we don't want too much delay in searching process. The users can see how quick the web service is responding when they access the dashboard. The search result is used to see what is sent back to the android application, we can also use this to check if the app is displaying correctly.

5. Storing the log information in a database

The log information is stored in a mongoDB noSQL database. The database name is yiguoProject4 and collection name is LogData.

Atlas connection string:

mongodb://yiguo:3e6aAWKpZlPlx1Nh@ac-g0ihedn-shard-00-02.b0ltwju.mongodb.net:27017,ac-g0ihedn-shard-00-01.b0ltwju.mongodb.net:27017,ac-g0ihedn-shard-00-00.b0ltwju.mongodb.net:27017/Cluster0?w=majority&retryWrites=true&tls=true&authMechanism=SCRAM-SHA-1

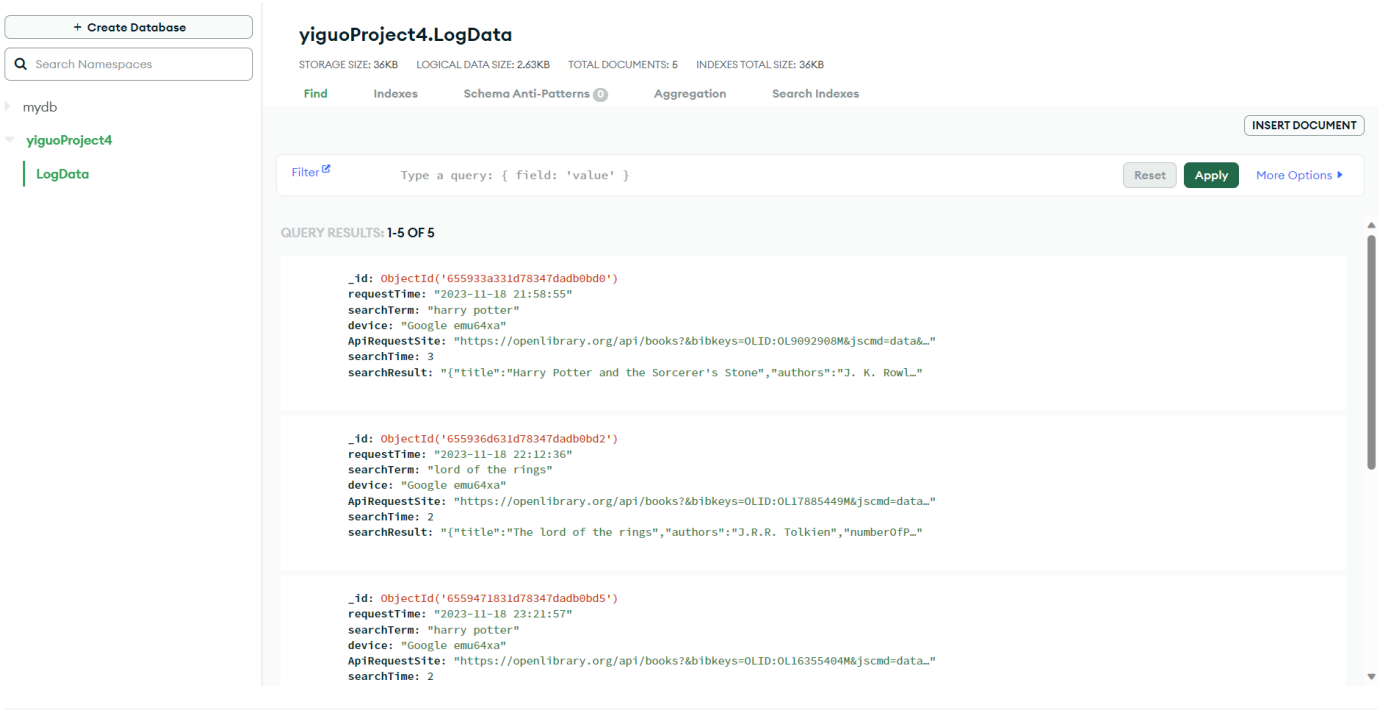
Three shards:

ac-g0ihedn-shard-00-01.b0ltwju.mongodb.net:27017

ac-g0ihedn-shard-00-02.b0ltwju.mongodb.net:27017

ac-g0ihedn-shard-00-00.b0ltwju.mongodb.net:27017

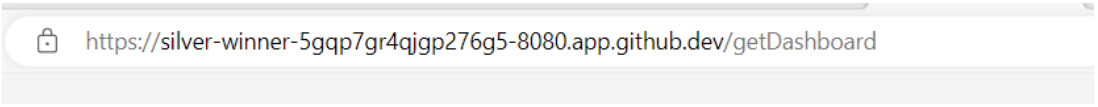
Here is a snapshot of the database collection data.



The web service connects to the database and stores information using the logging() method in the BookModel.java. It retrieves the information from the database using the dashboardAnalysis() method to perform analysis and format log table for the dashboard.

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

a. A unique URL addresses a web interface dashboard for the web service.
The unique URL is /getDashboard. This view is formatted in the dashboard.jsp from the web service.



b. The dashboard displays at least 3 interesting operations analytics.
All analysis are performed in the dashboardAnalysis() method in BookModel.java.

1. Most Popular Search Term: This gives the top term that is searched most frequently by users.
2. Most Popular Device: This gives the most frequently used device for searching.
3. Recent Search Time: This gives the search time of the most recent query. (in seconds)

Note: when there are ties, we select the most recent one for the most popular term / device.

Book Search Dashboard

Book Search Analysis

Most Popular Search Term: harry potter

Most Popular Device: Google emu64xa

Recent Search Time (Seconds): 2

Book Search Log

Mobile Request Time	Search Term	Device	API Request Site	Search Time	Search Result
2023-11-18 21:58:55	harry potter	Google emu64xa	https://openlibrary.org/api/books?&bibkeys=OLID:OL9092908M&jscmd=data&format=json	3	"title":"Harry Potter and the Sorcerer's Stone","authors":"J. K. Rowling","numberOfPages":150,"ISBN10":"8186775609","publishers":"Manjul Publishing House","publishDate":"","picture":"https://covers.openlibrary.org/b/id/8267065-L.jpg","description":""
2023-11-18 22:12:36	lord of the rings	Google emu64xa	https://openlibrary.org/api/books?&bibkeys=OLID:OL17885449M&jscmd=data&format=json	2	"title":"The lord of the rings","authors":"J.R.R. Tolkien","numberOfPages":"","ISBN10":"","publishers":"Folio Society","publishDate":"1977","picture":"","description":""
2023-11-18 23:21:57	harry potter	Google emu64xa	https://openlibrary.org/api/books?&bibkeys=OLID:OL16355404M&jscmd=data&format=json	2	"title":"Harī Pottā to kenja no ishi =","authors":"J. K. Rowling","numberOfPages":462,"ISBN10":"4915512371","publishers":"Seizansha","publishDate":"1999","picture":"","description":"Harry Potter thinks he is an ordinary boy until he is rescued by an owl, taken to Hogwarts School of Witchcraft and Wizardry, learns to play Quidditch and does battle in a deadly duel. The reason: Harry Potter is a wizard."
2023-11-18 23:22:14	harry potter	Google emu64xa	https://openlibrary.org/api/books?&bibkeys=OLID:OL13639000M&jscmd=data&format=json	2	"title":"Harry Potter and the Philosopher's Stone","authors":"J. K. Rowling","numberOfPages":223,"ISBN10":"1551923963","publishers":"Raincoast Books","publishDate":"2000","picture":"https://covers.openlibrary.org/b/id/771854-L.jpg","description":"Rescued from the outrageous neglect of his aunt and uncle, a young boy with a great destiny proves his worth while attending Hogwarts School for Wizards and Witches."
2023-11-18 23:22:41	fox	Google emu64xa	https://openlibrary.org/api/books?&bibkeys=OLID:OL18176186M&jscmd=data&format=json	2	"title":"Fantastic Mr Fox","authors":"Roald Dahl","numberOfPages":81,"ISBN10":"","publishers":"Tetley GB Ltd","publishDate":"1997","picture":"","description":""

c. The dashboard displays formatted full logs. (See above)

The logs are well formatted in a HTML table. The table displays the six pieces of information logged, including request time, search term, device, api request site, search time and search result.

Each time the web service receives a request from the mobile application, there will be a new record added to the table.

7. Deploy the web service to GitHub Codespaces

I created a ROOT.war for my web service and created a codespace using it. The URL is https://silver-winner-5gqp7gr4qjgp276g5-8080.app.github.dev.

Below is the screenshot.