Project 4 Task 2 – The Harry Potter Character Encyclopedia App

by Aastha Shah(AndrewID: aasthash)

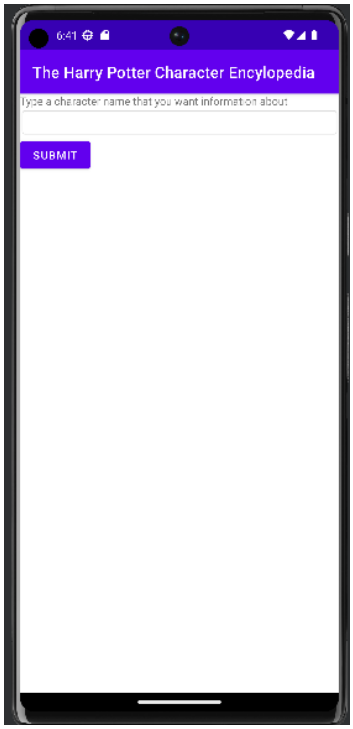
Description:
My mobile app will be like a character encyclopedia for Harry Potter characters.
It will prompt the user to type the character they want to see the information about and will display information like:
Name, Hogwarts House, Date of Birth, Ancestry, Patronus, Actor Name, Eye Color, Hair Color.
The app will also display an image of the character.

Here is how my application meets the task requirements

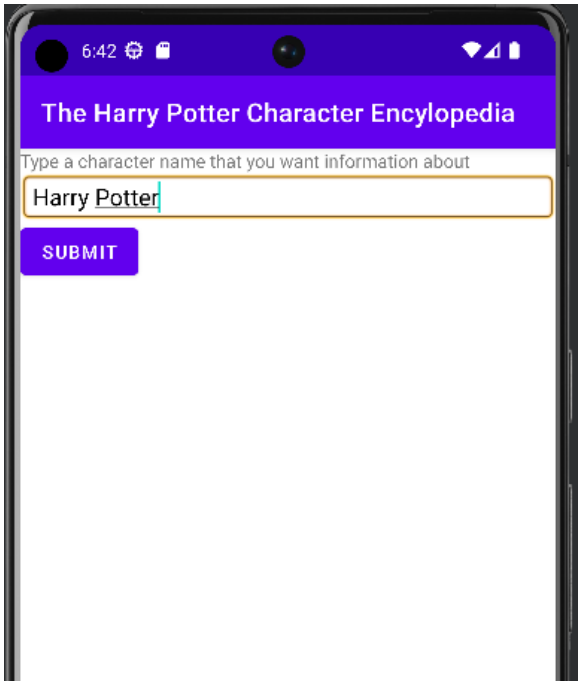
1. Implement a native Android application

The name of my native Android application project in Android Studio is: HPCharacters

a. Has at least three different kinds of views in your Layout
My application uses TextView, EditText, Button, and ImageView.
See content_main.xml for details of how they are incorporated into the LinearLayout.
Here is a screenshot of the layout before the character information has been fetched.



b. Requires input from the user Here is a screenshot of the user searching for information about the character “Harry Potter”



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

My application does an HTTP GET request in GetCharacterDetails.java. The HTTP request is:

https://aasthashah0897-didactic-space-winner-pv5wr4746jp2r65g-8080.preview.app.github.dev/hp-character-encyclopedia?searchChar=+searchChar

where **searchChar** is the character the user wants to search for.
eg:
https://aasthashah0897-didactic-space-winner-pv5wr4746jp2r65g-8080.preview.app.github.dev/hp-character-encyclopedia?searchChar=Harry%20Potter

The search method makes this request of my web application, parses the returned JSON, fetches the required information about the character in the form of a Response class.

The doInBackground method calls the search method and also calls the getRemoteImage method to fetch the Bitmap Image of the character using the imageURL that came in the JSON response from the Servlet.

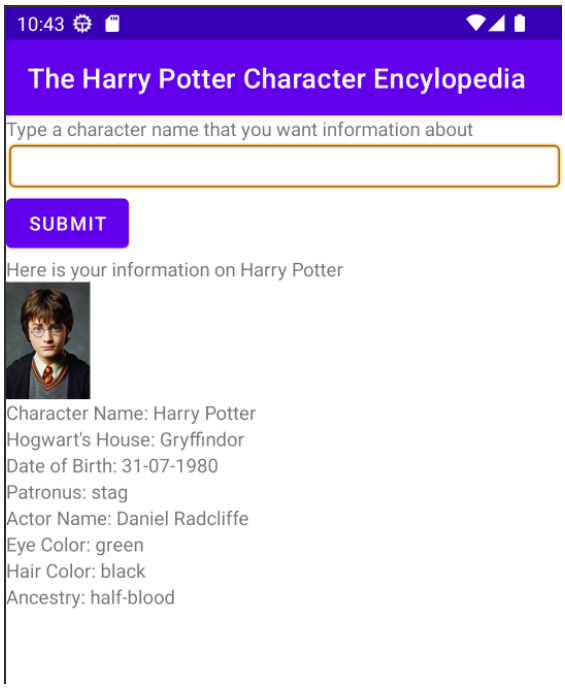
d. Receives and parses an XML or JSON formatted reply from the web service

An example of a JSON reply is:

{"charName":"Harry
Potter","house":"Gryffindor","dob":"31-07-1980","ancestry":"half-blood","imageURL":"https://ik.imagekit.io/hpapi/harry.jpg","patronus":"stag","actorName":"Daniel
Radcliffe","eyeColor":"green","hairColor":"black"}

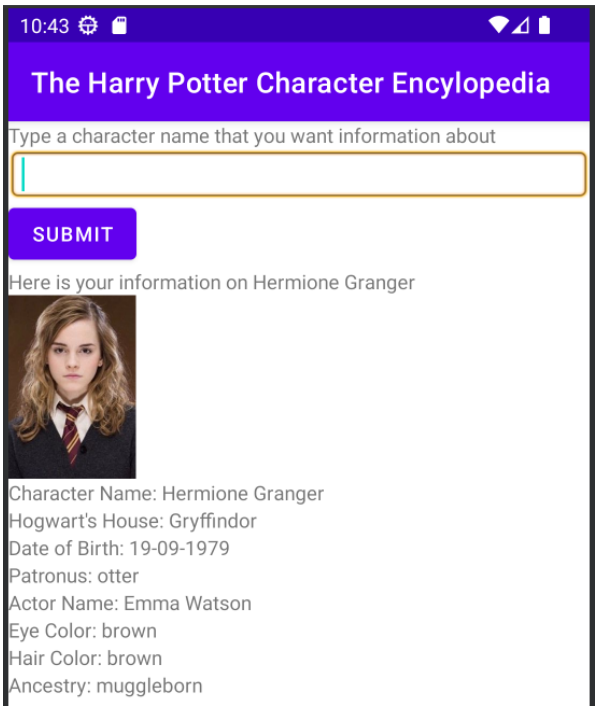
e. Displays new information to the user

Here is the screen shot after the information about the character has been returned.



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

The user can type in another character name and hit Submit. Here is an example of having typed in "Hermione Granger".



2. Implement a web application, deployed to Codespaces

The URL of my web service deployed to Codespaces is:
https://aasthashah0897-didactic-space-winner-pv5wr4746jp2r65g-8080.preview.app.github.dev/hp-character-encyclopedia

The project directory name is project-4-task-2-aasthashah0897

a. Using an HttpServlet to implement a simple (can be a single path) API

In my web app project:
Model: HPModel.java
View: index.jsp - for the json response and dashboard.jsp for displaying the Dashboard
Controller: HPServlet.java

b. Receives an HTTP request from the native Android application.

HPServlet.java receives the HTTP GET request with the argument "searchChar".
It passes this search string on to the model.

c. Executes business logic appropriate to your application

HPModel.java makes an HTTP request to:
https://aasthashah0897-didactic-space-winner-pv5wr4746jp2r65g-8080.preview.app.github.dev/hp-character-encyclopedia?searchChar=Harry%20Potter (Here searchChar will be the character name from the user request)

It then parses the JSON response and extracts the parts it needs to respond to the Android application.

d. Replies to the Android application with an XML or JSON formatted response.

index.jsp formats the response to the mobile application in a simple JSON format of my own design:

{"charName":"Harry Potter","house":"Gryffindor","dob":"31-07-1980","ancestry":"half-blood","imageURL":"https://ik.imagekit.io/hpapi/harry.jpg","patronus":"stag","actorName":"Daniel Radcliffe","eyeColor":"green","hairColor":"black"}

3. Handle error conditions

The application handles error conditions like:
No response from 3rd Party API, Bad Request, Mistakes in user input, Missing values of a few parameters for some characters, etc

4. Log useful information - Itemize what information you log and why you chose it.

I am tracking the following information in the logs:

1. Request Number - The number of the request in a particular session
2. Device Type - The device type with which the request is made
3. Time Stamp of the request
4. Query Parameter of the search request
5. API Lag - Processing time taken to process the request
6. Response Status - Whether the request was a “Success” or a “Failure”

I chose to log this information because it will be helpful to debug or resolve issues related to the application and track where and when did the error happen.

5. Store the log information in a database - Give your Atlas connection string with the three shards

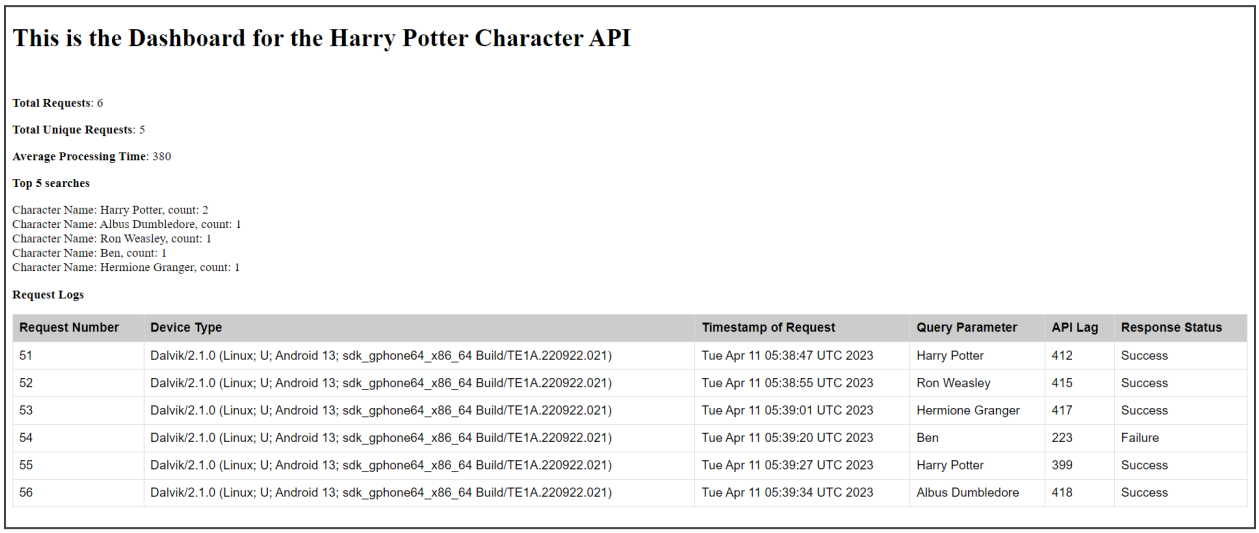
MongoDB connection String:

mongodb://aasthash:DSProject4Task2@ac-q5uyi2g-shard-00-00.djtke9x.mongodb.net:27017,ac-q5uyi2g-shard-00-01.djtke9x.mongodb.net:27017,ac-q5uyi2g-shard-00-02.djtke9x.mongodb.net:27017/test?w=majority&retryWrites=true&tls=true&authMechanism=SCRAM-SHA-1"

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

The Dashboard also displays some analytic information about the logs.
It displays the following:

- Total Number of Requests sent to the Server
- Total number of unique requests
- Average processing time for each request
- Top 5 character searches and the number of times the character has been searched.



This is the Dashboard for the Harry Potter Character API

Total Requests: 6

Total Unique Requests: 5

Average Processing Time: 380

Top 5 searches

Character Name: Harry Potter, count: 2
Character Name: Albus Dumbledore, count: 1
Character Name: Ron Weasley, count: 1
Character Name: Ben, count: 1
Character Name: Hermione Granger, count: 1

Request Logs

Request Number	Device Type	Timestamp of Request	Query Parameter	API Lag	Response Status
51	Dalvik/2.1.0 (Linux; U; Android 13; sdk_gphone64_x86_64 Build/TE1A.220922.021)	Tue Apr 11 05:38:47 UTC 2023	Harry Potter	412	Success
52	Dalvik/2.1.0 (Linux; U; Android 13; sdk_gphone64_x86_64 Build/TE1A.220922.021)	Tue Apr 11 05:38:55 UTC 2023	Ron Weasley	415	Success
53	Dalvik/2.1.0 (Linux; U; Android 13; sdk_gphone64_x86_64 Build/TE1A.220922.021)	Tue Apr 11 05:39:01 UTC 2023	Hermione Granger	417	Success
54	Dalvik/2.1.0 (Linux; U; Android 13; sdk_gphone64_x86_64 Build/TE1A.220922.021)	Tue Apr 11 05:39:20 UTC 2023	Ben	223	Failure
55	Dalvik/2.1.0 (Linux; U; Android 13; sdk_gphone64_x86_64 Build/TE1A.220922.021)	Tue Apr 11 05:39:27 UTC 2023	Harry Potter	399	Success
56	Dalvik/2.1.0 (Linux; U; Android 13; sdk_gphone64_x86_64 Build/TE1A.220922.021)	Tue Apr 11 05:39:34 UTC 2023	Albus Dumbledore	418	Success