

Introduction

Generally, [reverse image search](#) refers to a specific case of Information Retrieval (IR) process. In the early days, search engines such as Google or Bing heavily relied on keyword-spotting techniques that used dictionaries of known words and their meanings when parsing users' text input. To parse an image, on the other hand, an IR process has to use more complex methods than keyword-spotting in order to extract useful information. These methods range from extracting colours and edges to recognising objects in images (e.g. people, buildings, cars, etc.).

This exam is individual or group based (1-3 students) written home examination. The exam focuses on building an Android application to perform reverse image search and organise obtained data in a virtual Android device (emulator) or a physical phone. Students are provided with an AWS server instance running at <http://api-edu.gtl.ai/api/v1/imagesearch> with four end-points that were created specifically for this course. The server continuously runs a Flask Python server instance that accepts an image on one end-point (/upload) and performs reverse image search on another three end-points (/google, /bing, and /tineye). Figure 1 attempts to visualise the server's end-points. It is strongly encouraged that students use [Postman](#) software to test the end-points' inputs and outputs during your development. Alternatively, you could also use [curl](#).

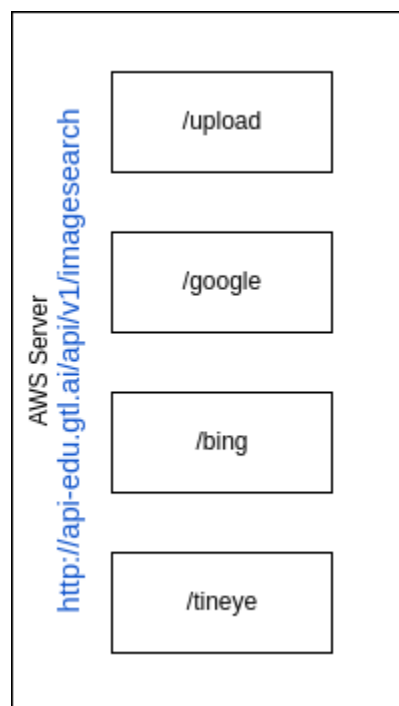



Figure 1: Four AWS Server's end-point paths

Suggested Screen Diagrams

Table 1 below provides a suggestion to how an app for reverse image search may look like. However, there is no specification neither for how a resulting application should look like nor what it should do. Therefore, students have complete freedom to what their application could look like and what purpose it is going to serve.

Table 1: Suggested Screen Diagrams

UI Design Suggestion	Explanation
	<p>Screen 1 (blue rectangle - selected button):</p> <p>The main launcher screen should be where user is going to select a picture from internal storage or take a picture with camera (camera is not required since it has not been covered in class and it is assumed that students will be working with an emulator. However, it is reasonable to use a camera on a physical device when working with images).</p> <p>Once the image is selected, user should be able to adjust the area on the image either by zooming in or out of a specific area on the image or using a bounding box to select an area of interest.</p> <p>Once the user is satisfied with the selected and adjusted image, the image can then be sent to the server in order to find similar images</p>





Application Requirements

The list below specifies hard thresholds when evaluating your submissions. This means that not meeting these requirements will have a high impact on the final grade.

1. At least one list must be handled by a RecyclerView
2. Use intents to pass data
3. Use non-UI-blocking requests to the server
4. User SQLite database to store data
5. Create at least one reasonably complex custom view
6. Use Kotlin as a main programming language (some Java is acceptable when re-using the code found online)
7. Make use of all the end-points on the server
8. Exception handling (e.g. no network connection)
9. Make sure to handle all the Android lifecycle states (the app will be paused, resumed, stopped, etc. during testing)
10. Make use of res/drawable, res/layout, and res/values

Additional Tasks

List below specifies additional challenges, that are also rewarded with points(see Table 3 for more info)

1. Use third-party libraries
2. Extra effort in processing images will be rewarded
3. Comprehensive design will be rewarded
4. Use of Android OS services will be rewarded (e.g. battery level, microphone, GPS, etc.)
5. Compress images to save memory
6. Use bound service for downloading the data periodically
7. Make your own drawables (Hand/Adobe painted or Drawable-programmed in Android Studio)

API

As the Introduction section and Figure 1 suggest, the API's path is:

<http://api-edu.gtl.ai/api/v1/imagesearch/> (don't click, you will get 404). The AWS server that is used for this exam is a relatively small machine. This means that it takes time for the machine to crunch the numbers, which in turn means that there is a delay between each request and reply. Be patient with the server and try not to hack it - it will have an effect on your grade.

If you are impatient, you can always set up a mock server locally that spits out stored output that is similar to that of the AWS server or inject the server data directly into your Kotlin methods.

This is what you would usually do at a workplace to reduce the load on the server during the prototyping phase.

Table 2: API Specification

End-point	HTTP Method	Input	Output
http://api-edu.gtl.ai/api/v1/imagesearch/upload	POST	Content-Disposition: form-data Content-Type: image/jpeg Form data: Rfc2387 standard	https://gtl-bucket.s3.amazonaws.com/automatically_generated_guid.jpg
http://api-edu.gtl.ai/api/v1/imagesearch/bing	GET	Request parameter: url	[{ "store_link": "http://...", "name": "", "domain": "", "identifier": "", "tracking_id": "", "thumbnail_link": "http://...", "description": "", "image_link": "http://...", "current_date": "" }, {"store_link": "...", {...}, ...]
http://api-edu.gtl.ai/api/v1/imagesearch/tineye	GET	Request parameter: url	
http://api-edu.gtl.ai/api/v1/imagesearch/google	GET	Request parameter: url	

Table 2 specifies the API's inputs and outputs. For example, server's end-point /upload expects a multi-part form of image/jpeg type as an input. Once the correct data is sent to the server, the server stores the image and returns a url for the newly stored image on the server. The user should assume that the image is going to be removed from the server at some point and cannot rely on the url being live for a long time. Next, server's end-point /bing and /tineye and /google all have the same specification. This end-point expects a url parameter (e.g. http://api-edu.gtl.ai/api/v1/imagesearch/google?url=https://www.url_of_an_image.png), which will trigger reverse image search on the server. Once the server is done, it will return a list of JSON objects. Each JSON objects contains few key-value pairs. You should only be interested in `thumbnail_link` and `image_link` keys (these are most likely to be the same), however, this may not always be the case.

Examples (You can also download a Postman workspace:
<https://www.getpostman.com/collections/8c0e815075309ef7f1c4>)

1. Upload image using cURL: `curl -F "image=@/path/to/image/on/your/pc/some_image.png" http://api-edu.gtl.ai/api/v1/imagesearch/upload`

Returns URL of your image stored on the server

2. Get list of similar images using Bing: `curl http://api-edu.gtl.ai/api/v1/imagesearch/bing?url=https://www.google.com/images/branding/googlelogo/1x/googlelogo_color_272x92dp.png`

Returns a list of JSON objects with each JSON object containing `image_link` and `thumbnail_link` key-value pairs

Suggestions

- Use [Fast Android Networking Library](#) (Gradle: implementation 'com.amitshekhar.android:android-networking:1.0.2')
- Make a copy of server responses during the development process to reduce the load on the server
- Prototype quick
- Prioritise NullPointerExceptions. These are the worst and have huge impact on application usability
- Split code according to the [MVC](#) guidelines

Report

Each group is expected to submit a concise report of 3 pages (excluding images/diagrams and code), where you will present your project and what you used the reverse image search for. For the technical part of the report, you will have to document the programming choices you made (e.g. library used, splitting functionality into classes, etc). In addition, you will have to provide a discussion for why a certain UI component, pattern, framework, etc. has been used and what are the possible alternatives. Reflect on pros and cons for your chosen approaches.

Submission

Each group is required to submit their source code and a group report in a .zip file. Both in code and in the report students are required to put comments and/or explicitly state which of the sub-requirements from Table 3 they are targeting.

Evaluation

Table 3 gives some insight into what each group will be evaluated on. The report is weighted the most as your discussion and reflection is paramount. Hard sub-requirements are not dependent on the size of the group, while the soft requirements are scalable based on the group size. In general, it is expected more from a group with 3 students than from a group with 1 student.

Table 3: List of sub-requirements

#	Sub-Requirement	Points (Total 100)
1	Hard: Show that you understand when it is appropriate to use the Elvis operator by pointing to parts of code where it was used and discussing what edge cases it is meant to handle	8
2	Hard: Make use of lambdas and higher order functions when processing data that is similar, but not exactly the same. Discuss the code in the report as well as pros, cons, and alternative code structure	8
3	Soft: Make (or extend existing) classes with methods and companion classes to process images. Make use of constructors, overloading, properties, overriding, and interfaces where it is appropriate. Discuss pros, cons, and alternative code structure in the report by referencing your code	8
4	Soft: Implement your own RecyclerView adapter and discuss the view you used for each row in detail in the report	8
5	Soft: Have more than one Activity / Fragment (up to you which one you use). Explain why you decided to use Activities and/or Fragments	3
6	Hard: Pass the data as Parcelable between Activities / fragments	4
7	Hard: Make use of threads or coroutines to make asynchronous operations. Your UI should never freeze until some operation is being executed. Discuss in the report your async code and try to time it by using Android Studio tools or timer in the code for each iteration of your operation. What is the best/worst case scenario of the operation you decided to make asynchronous?	4
8	Hard: Store and retrieve images as blobs in/from the SQLite database. Describe and discuss the database	8

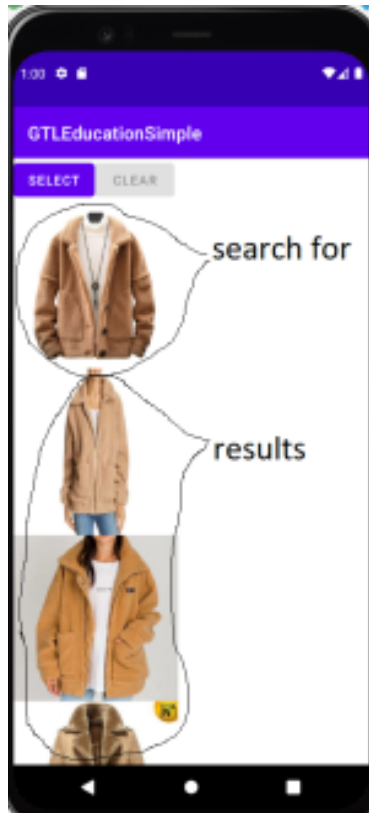
	structure in detail. What are the pros / cons / alternatives considering a commercial version of your application	
9	Soft: Extend a 3rd-party complex view or make your own. Document what parts of the code you have written yourself if the view is downloaded from somewhere.	8
10	Hard: Create callbacks across the application to let the parent class / view know when something is changing. Write about all your callbacks in the report and why they are useful for the information flow in your app	3
11	Report with the following structure: 1. Introduction - introduce your idea for the application. What reverse image search can be used for. What you used reverse image search for in your app / previously. What have you noticed / observed when getting results from the server and why you think you observed those things 2. Overall code structure (classes, interfaces, abstracts, xml, etc.). Class diagram would be helpful here 3. Implementation / Design decisions 4. Discussion - a) what you did not do, b) what you did not do (but wanted to do) and why, c) and what you would have done differently if you had the same task again 5. Conclusion 6. References	35
12	How well you followed the rules / guidelines / suggestions / how attentive you were when reading this task description	3
13	Additional challenges	3 points: Use third-party libraries 3 points: Extra effort in processing images 3 points: Comprehensive design will be rewarded 3 points: Use of Android OS services will be rewarded (e.g. battery level, microphone, GPS, etc.)

		3 points: Compress images to save memory 3 points: Use bound service for downloading the data periodically 6 points: Make your own drawables (Hand/Adobe painted or Drawable-programmed in Android Studio)
--	--	--

After the first round of grading all the groups, there will be an inter-group adjustment process based on the linear transformation formula. This is to make sure that the average is adjusted in case if all groups struggle a lot or all groups outperform

$(b - a)(x - \min x) / (\max x - \min x) + a$, where
 $a = \text{mean}(x) - \text{std}(x)$ and $b = \text{mean}(x) + \text{std}(x)$

Still Feeling Confused



In case you are still stuck, dazed, and confused and cannot imagine what you should do, here is an example application that uses the API from Section N to search for similar clothes. In the image, the “search for” picture is selected from the internal storage on the phone and “results” pictures are returned from the server