Glide: A Big Picture

Marc Andrada, Duo Chai, Soobin Choi

Rationale for Resubmission:

When writing up our initial report, we were not thorough enough in determining key developers for our system. We were initially unsure of how to approach this question and thought we could just generalize developers as those who we saw working on the project as contributors including ourselves. However we should have gone deeper into determining these key developers. We have since updated that section of our report filling in the gaps with more concrete examples indexed below:

1. Determined the main contributor of the project, giving his name and an explanation as to why he is considered a key developer.
2. Discovered additional names of people who have contributed to the project in some capacity. We learned that a key developer isn't only about those who improve the source code, but help to improve the overall project. This means there were some key developers we hadn't even considered before taking a deeper look into documentation (README.md).
3. Recognized the transfer of project ownership from bumptech to being integrated into Google.
4. Gave more rationale as to why we would be considered key developers as well.

Glide Stakeholders

Organizations or individuals that are either actively using Glide and submitting issues to Glide or are interested in or intend to implement parts of the API in their own projects are directly affected by Glide's action, such as changes in Glide's source code. As such, users of the system are stakeholders.

Key developers in Bumptech (developer company) are directly affecting the Glide system through actively maintaining and contributing to its code base, thus are key stakeholders of the system.

Glide system provides sample use cases of the library that test on a series of resource suppliers such as Giphy, Imgur, and Flickr. Each of these suppliers are platforms for media display and sharing. Since Glide depends on these supplies to provide media resources, such platforms can be considered relevant stakeholders of the Glide system.

Glide Functionalities

Overall Domain

Since the Glide system supports managing media resources that are primarily displayed through a user interface, the overall domain of Glide would involve systems that rely on media resource management and have end users interacting with the system.

Essential Functional Aspects

1) Image caching, processing, and displaying is an essential function in the Glide system, supporting a simple and smooth way in which media resources of varying file types get loaded and displayed to end-users.

2) Media resource processing is another essential function that enables file resources to be properly read and converted into Android-compatible files.

Non-Functional Aspects

1) Performance: The speed at which our system processes and displays media resources.
2) Scalability: How our system manages a larger amount of media resources and what to adjust if this API was utilized as part of a larger system
3) Usability: The ease of use and implementation of the library, as well as the documentation provided to assist with the usability of Glide.

Uniqueness

1) Glide provides an easy and simple way, using a single line of code, to load and display a media file
2) Glide makes this process smooth and error free

Key Glide Developers

The first thing we considered when determining key developers were those that upkeep the system itself. They would be the "gate keepers" of the project, ensuring that each pull request is thoroughly examined before allowing the changes to be added to the system. Therefore, the first name that comes to mind is Sam Judd who appears on GitHub to be the main contributor with over 1600 commits. There are a little over 100 other contributors to the system, but none of them have close to as many commits as Sam Judd. Sam appears to actively comment on recent pull requests sharing his advice and determining whether the particular request will be merged.

Through the README.md file provided, we have determined a few other names, which may be considered key developers for the system. Particularly, these individuals have contributed to Glide in various ways. Some implemented particular functionality within the system. For example Jake Wharton and "the Android team" have contributed to the implementation of Glide's disk caching. Additionally, Dave Smith is credited with developing the GIF decoder Glide utilizes. Others, such as Chris Banes, focused on the build scripts. He created the following gradle-mvn-push script. Finally, some key developers have worked on the UI/UX components such as the branding/logo of the product. In this instance, Corey Hall helped design Glide's logo. All of these people have contributed to the overall project and thus should be considered key developers as well.

Although Glide was originally a project of the company, Bumptech, it appears to have been integrated into Google per the README.md file provided with the system. As a result, further contributions to this system require new developers to sign Google's individual contributor license agreement. Those using Google applications, which utilize Glide's library should also be considered key developers of the system. The integration of this library must be handled in a way, which would allow other related applications to run cohesively.

On a broader scope, we can consider our team to be developers of Glide. We have been monitoring this system for the past few months and are actively looking at current and ongoing issues. We have a goal to contribute by resolving an issue, big or small and thus we too would have a stake in this system as developers.

1 Glide Issue on Github: https://github.com/bumptech/glide/issues?page=1&q=is%3Aissue+is%3Aopen

Issues[1]

1. Can't load imgur image
   https://github.com/bumptech/glide/issues/4092
   → This issue is concerned with loading images specifically from the internet, which can either be an issue with the file source itself or an issue with receiving the image from the internet. We could consider related pathways.

2. GIF not loaded when the file path includes Korean name
   https://github.com/bumptech/glide/issues/3675
   → We can look within the file reading/loading functionality of our system and see if it supports the multiple languages when reading a file name. If there is not, our system is limited in terms of the languages that it supports.
   Also when interpreting file names. We might need to take a look at how Glide interprets input Strings - for instance, is it using an UTF-8 way or something else?

3. Load local gif cause GlideException: Failed to load resource
   https://github.com/bumptech/glide/issues/3664
   →According to its failure log, we can identify the issue in failed loadpath and failed decodepath. We should take a look at load class (RequestBuilder) and decode class (decoder) of this media file type, GifDrawable to locate where the error was generated and debug it.

4. Wrong Image is being displayed on some devices in certain cases
   https://github.com/bumptech/glide/issues/3544
   → One potential solution may lie in the system's xml documentation since the user was having issues with image display on different devices. In Android Studio, different devices may require different layout files to load the image properly.

5. An image cannot be loaded on Android 7.1.1
   https://github.com/bumptech/glide/issues/3161
   → Since the image file can be loaded by BitmapFactory, but cannot by Glide, we might need to take a look at when GlideRequest.load() takes in a file path and how it interprets the media object type from the file path. Based on the problem description it seems like GlideRequest works fine when the media file type is specified as Bitmap. So we might need to improve how GlideRequest recognizes different media types by default.

1 Glide Issue on Github: https://github.com/bumptech/glide/issues?page=1&q=is%3Aissue+is%3Aopen