Rationale for Resubmission:
Our initial report format was closer to that of a documentation. We included UML diagrams without necessarily referring to them or describing what is being presented. Further, we included a lot of code snippets that do not enhance the understanding of the overall functionality of each feature. Lastly, our original report lacked structure. Below is an index of the changes we made to fill the deficit in our original report:

1. Restructured our report into three parts; Feature Overview, Relevant Components, and Component Behaviors
2. Replaced code snippets with pseudocode and used abstractions to describe the behaviors of each component.
3. We elaborate on the relationship of the components to other classes in the system through the use of UML diagrams and Sequence Diagrams, referring specifically to each relationship discussed.
4. Added figure labels and created a section for all figures mentioned in the report.

Media Resource Load and Display

Feature Overview

Glide is an open source media management and image loading framework, therefore image display is an essential part of the system's overall functionality. Glide aims to make media fetching, resizing, and displaying a smoother experience for the user. To generate that experience, the system builds requests for each resource. Such feature to build requests can be implemented in any android activity to handle various components of the image loading and displaying processes.

In the Glide system, such request mentioned above is referred to as a GlideRequest. GlideRequests can be specified to address a particular type of media file (i.e. GIF, video, drawable, Bitmap) to be loaded and displayed. This makes the GlideRequest class the most important part of our feature.

Relevant Components

Figures 1 and 2 illustrate that both RequestBuilder and RequestManager classes are relevant to the GlideRequest class. Figure 2 demonstrates that the GlideRequest extends the RequestBuilder class as well as the RequestManager class.

Figure 3 shows that the RequestBuilder class provides multiple basic methods to different GlideRequest classes dealing with different media file types that extend RequestBuilder. The RequestBuilder enables the system to handle the request of loading and displaying different types of media.

Such a relationship is similar in the way that RequestManager is connected to the overall system. As illustrated in Figure 4, the GlideRequest classes, each of which addresses a varying

file type, extend the RequestManager. RequestManager enables the system to manage the request of loading and displaying different types of media.

Component Behaviors

To load media resources using RequestBuilder and RequestManager, the user needs to call a request of type GlideRequest <TranscodeType>, which extends RequestBuilder<TranscodeType> class and calls its load() method. The GlideRequest class also overwrites some basic methods from its parents, such as placeholder(), error(), transition(), listener(), addlistener() etc., altering the way a media file is loaded, displayed, and monitored.

With this feature, GlideRequest is able to use a simple and fluent API that allows users to maximize a request but adding multiple attributes in a single line. The pseudo code below shows how the user can load an image by simply creating a GlideRequest within an android activity.

'GlideRequest.placeholder().error().transition().listener()'

After adding all the attributes above, we can call the load() method to load the media, applying the attributes we mentioned, by using

'GlideRequest.load(uri).into(ImageView)'

Since the main class of this feature, GlideRequest, extends RequestBuilder and RequestManager classes. It is helpful to examine how they are implemented and used in the overall system.

1) *RequestBuilder.java (/glide/library/src/main/java/com/bumptech/glide/)*, handles various components of a resource such as resource loading, display, monitoring, etc.
   Key methods:
   - RequestBuilder.placeholder() method will return a new RequestBuilder with an attribute holding the place for the loading media.
   - RequestBuilder.error() method will return a new RequestBuilder with an attribute that will display a resource if the load fails.
   - RequestBuilder.listener() method will return a new RequestBuilder with an attribute that will monitor the resource load.
   - RequestBuilder.load() method will return a new RequestBuilder to load the given uri.
   - RequestBuilder.into() method will set the ImageView the resource will be loaded into

2) *RequestManager.java (/glide/library/src/main/java/com/bumptech/glide/)* manages various operations of a GlideRequest such as starting, stopping, restarting, etc.
   Key methods:

- *RequestManager*.pauseRequest() method will cancel any load in progress without clearing the resource of complete request.
- *RequestManager*.resumeRequest() method will restart any loads that haven't been completed yet.
- *RequestManager*.onStart() method makes a lifecycle callback that registers for events.
- *RequestManager*.onStop() method makes a lifecycle callback that unregisters for events.
- *RequestManager*.onDestroy() method makes a lifecycle callback that cancels all requests in progress and clear all recycled resources for all completed requests.

Based on the implementation of RequestGuilder, RequestManager, and their relationships to GlideRequests, we can confirm that the components are essential in supporting the functionality of the media loading and displaying feature.

A relevant feature would be proper image file processing, as we cannot expect to load an image correctly using this system without file processing. From our examination of the encoder, decoder, and transcoder components of this system (reported in feature 1 of assignment 2), we have determined that they are essential to achieving this goal. These interfaces allow for modularity of these file processing methods and allow users to make customized implementations with varying image file types.

Figures

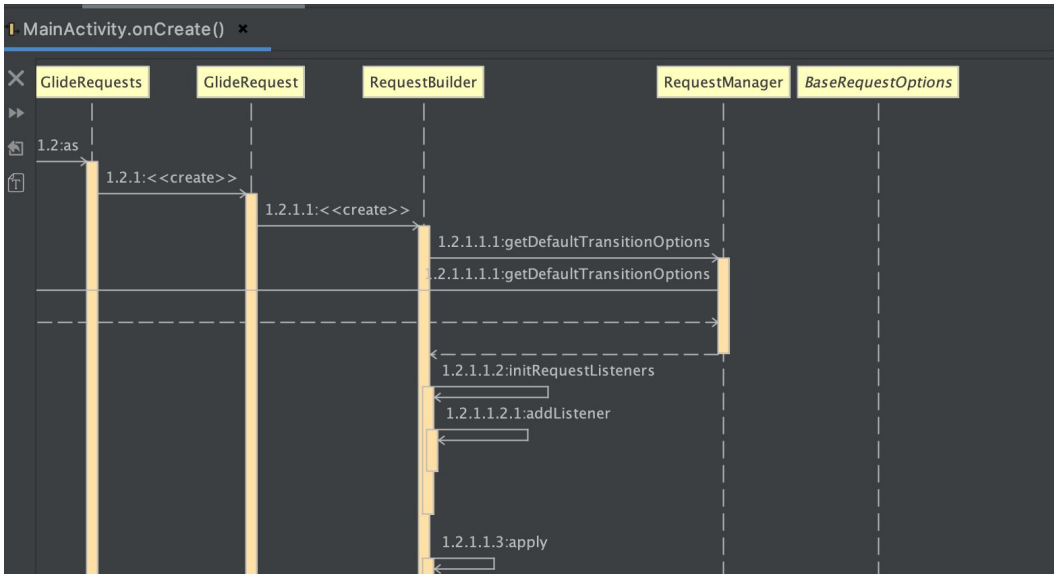

Figure 1: Sequence Diagram of Method Calls to RequestBuilder and RequestManager

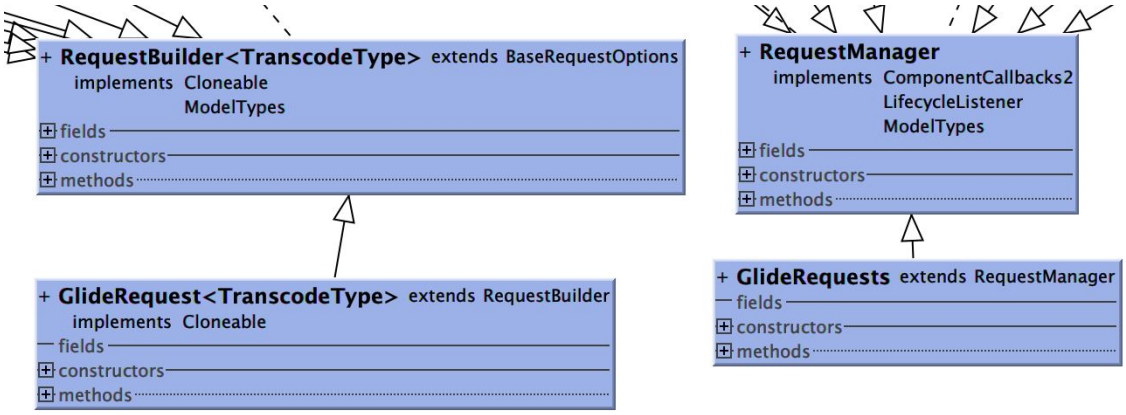

Figure 2: GlideRequest Class Extending RequestBuilder and RequestManager Class

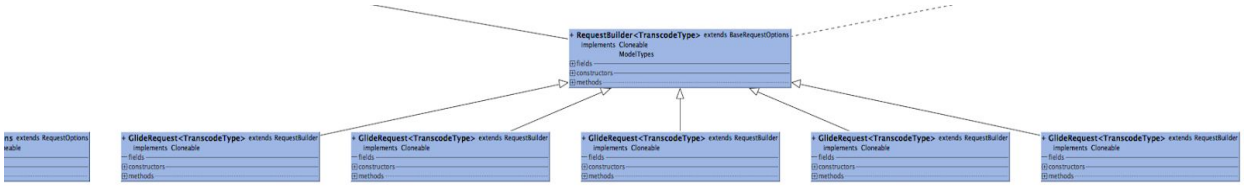

Figure 3: GlideRequest Classes Extending RequestBuilder Class

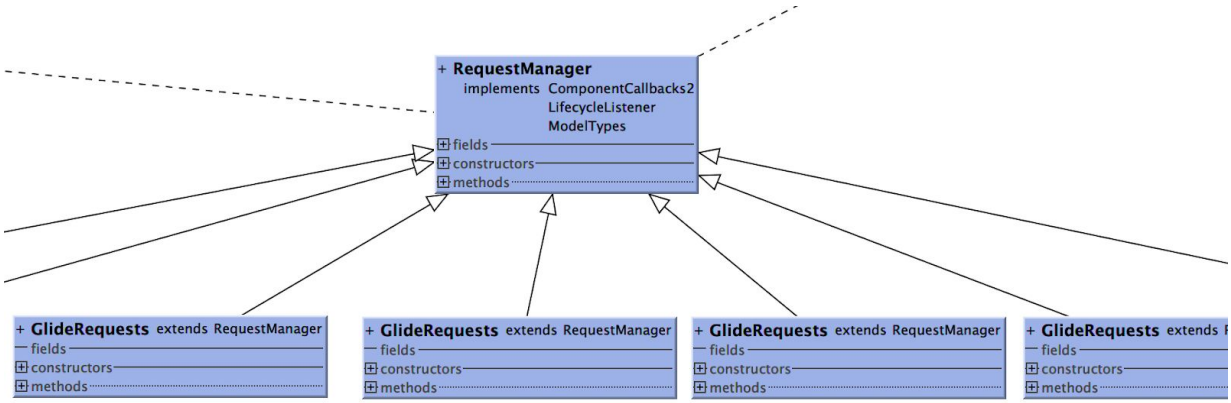

Figure 4: GlideRequest Classes Extending RequestManager Class