CprE 339 - Software Architecture
Final Project
Joseph Elliott
Evan Woodring
Mitchell Borman

# Deliverable #1 - Project Proposal and Group Information

## Textual Description Of What The Project Is About

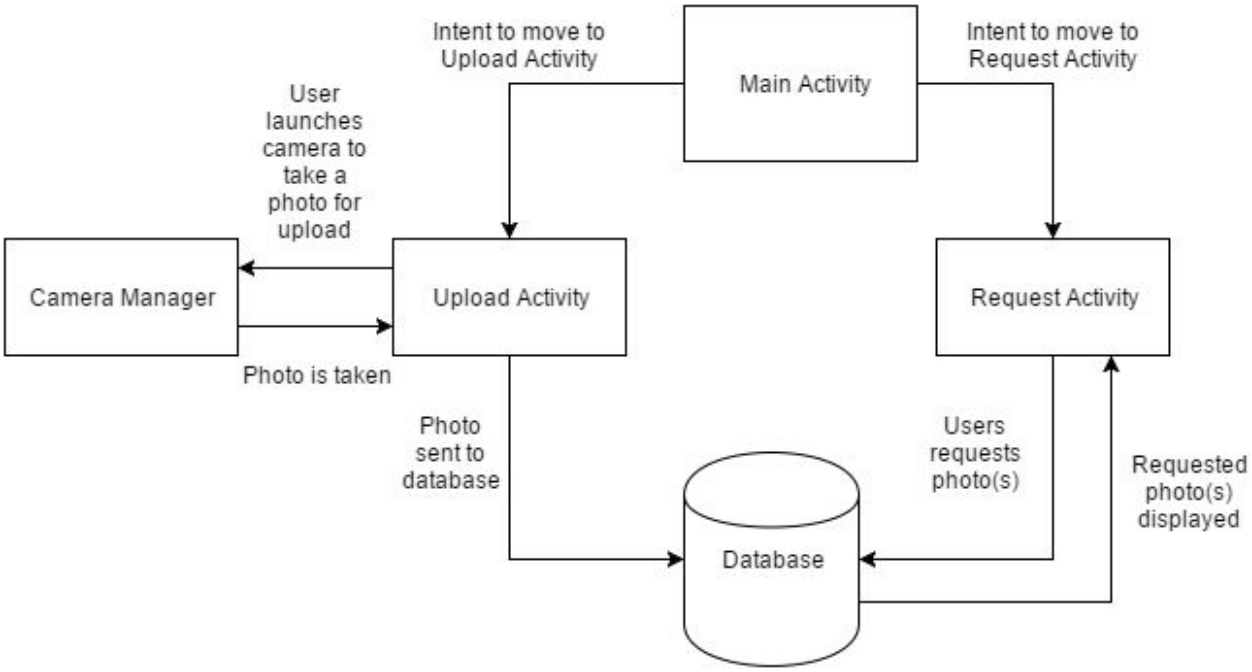
This is a basic Client-Server Android project where the user can take a picture with a phone, and then send that picture to a server. The user may also request a previously uploaded file from the server and view it.

## Group Information - Group Member Names

Joseph Elliott
Evan Woodring
Mitchell Borman

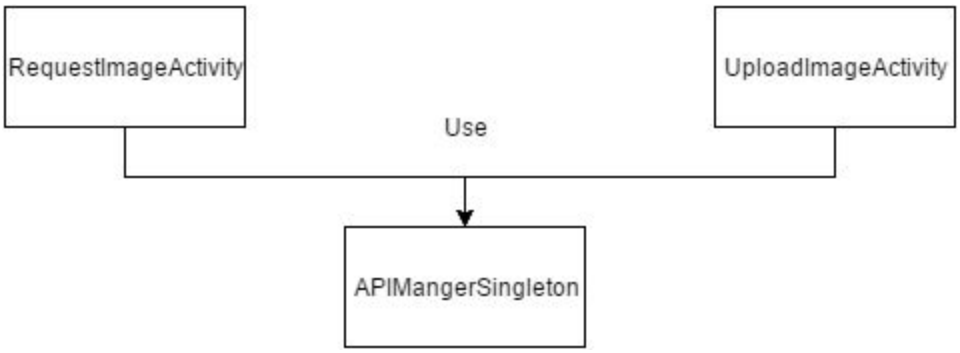
# Deliverable #2 - High-Level Software Architecture

## Connector-Component Diagram




The APIManager handles all of the requests, so whenever the request or upload activity wishes to send a request to the backend, they have to use the APIManager.

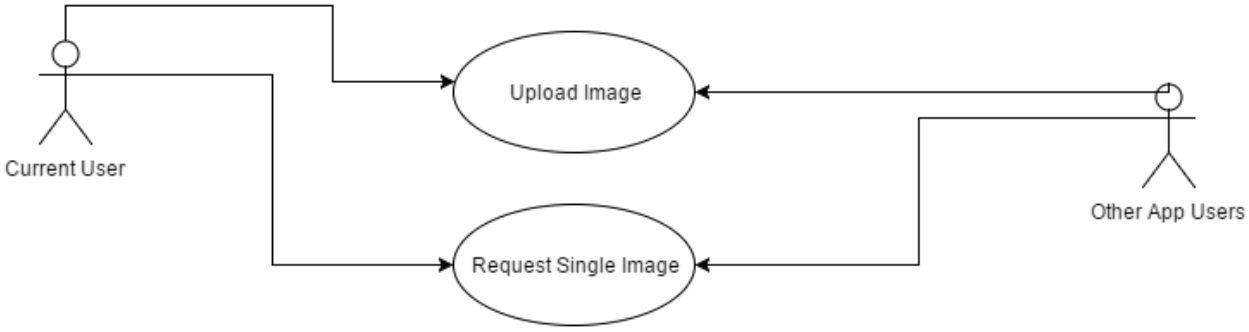
## Module Diagram




Uses diagram showing the relationship between both requesting and uploading an image and the APIManager.

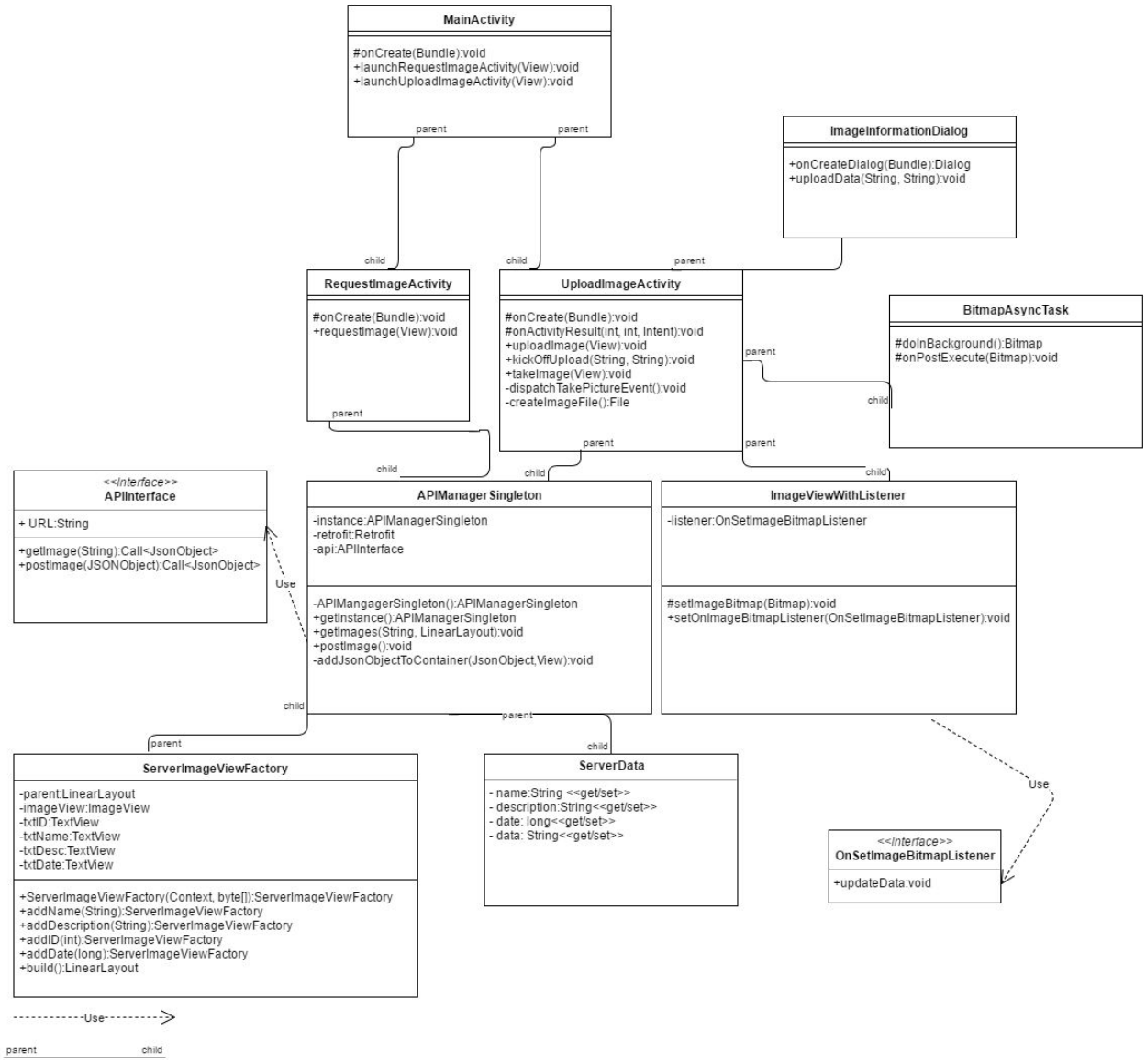
# Deliverable #3 - Detailed Design With UML Diagrams

## Use-Case Diagrams




At any given moment, only one actor will be interacting with a given instance of the app. However, other users may be interacting with the app as well, so we have created an actor to represent the other users of the app.

# Class Diagrams




# Detailed Class Diagrams - 3 Design Patterns Implemented

As seen in the image above, the three design patterns implement are:

Singleton - APIManagerSingleton

- The idea behind this singleton is to reduce the resources required to create the API manager.

Factory - ServerImageViewFactory

- The idea behind this factory is to have a class available where customization is easy. The programmer may not want to display all image details all the time, so this class makes this construction easy.

Listener - ImageViewWithListener

- The idea behind this listener is to constantly update the Base64 encoded string whenever the bitmap within the ImageView changes. While I could update the Base64 string whenever I manually update the code, this guarantees instances the encoded string updating where that code doesn't execute.

# Deliverable #4 - Your Entire Code

## Your Code (A Github Link)

https://github.com/Joseph-W-E/FileMe

## Two Locations Where You Implement Defensive Programming

Location 1:

- In Server/api.rb, we implemented defensive programming by guaranteeing the JSON request format. This is done by demanding specific elements.

Location 2:

- In Server/database.db, we implemented the database in a way to guarantee information entered into the database isn't null. It is not required to have a description for an image, but everything else is required.

## Two Locations Where You Might Have Corrected Code Smells

Location 1:

- In Server/api.rb, we avoid the Dead Code code smell. At first, our ambitions were higher, but given the requirements, we knew we wouldn't use all of the requests that we had written. Therefore, we removed the code for these requests.

Location 2:

- In FileMe/app/src/main/java/com/example/joey/fileme/api/ServerData.java, we avoid the Data Class code smell. At first, we created this class to have a sensible way to extract data from a GET request. However, we realized this wouldn't suffice. Thankfully, we found a way to map GET requests to a class, making this class less of a DataClass and more of a mapper.

## Locations That Are Responsible For Inter-Process / Threads Communication

APIManagerSingleton - All API calls are async tasks.
BitmapAsyncTask - A custom class for moving the Bitmap operations to a background thread. This class was chosen over the Thread class because views can only be updated from the main thread. AsyncTask allows for this via onPostExecute, but Thread does not.

## Three Locations Where You Implement Three Design Patterns

See above.
The three locations are:

- Singleton - APIManagerSingleton
- Factory - ServerImageViewFactory
- Listener - ImageViewWithListener

# Deliverable #5 - Demo video

## What The Video Should Include:

- Describe what your Java/Android application does
- Its design (talk through all your four deliverables)
- Show how your code works for 2 or 3 example inputs (based on your use-case diagrams)

Video Link: