CS5542: Project Report 2

Cognitive Visual Recognition Tracker

By Matthew Velazquez

1. Project Goal and Objectives

1.1 - Primary Objectives

One of the main objectives of CVRT is for physicians to be able to determine trends from patient data that could either be applicable to the individual patient, or to many patients if an aggregate is formed from many individual datasets. On an individual level, these metrics would provide a way for the physician to monitor daily cognitive capability, whereas on a grander scale, these joint datasets could be used to provide better overall treatment for the disease with the future inclusion of predictive analytics. As for short-term advantages, since this monitoring would be implemented daily, delirium can be more quickly identified as it could be the logical result of a sharp drop in image recognition capability.

1.2 - Significance

While there are many demos available in regards to the topic of Visual Question Answering, very few exist that have applicable results for the medical field. With this in mind, CVRT was designed to build on the existing Visual Question Answering space, but to produce practical, medically relevant data as a result. This application is significant because it provides a unique way to track the progression of Alzheimer’s disease based on image recognition in a more data-driven way than was previously possible.

1.3 – Features

* Random pictures are displayed in the application with a question-answer focused interface
* Users can ask questions about those images in order to discover their identities/characteristics. (i.e. What is happening in this picture? Where is this? What item is on the table? etc.)
* Ability to track metrics for the individual patient’s image recognition ability over time.
* Short-term monitoring that can detect delirium and advise for emergent medical treatment.
* Possible ability to import custom images (i.e. A familiar location or relative’s home)
* Possible inclusion to aggregate multiple patients’ data to allow for better predictive analytics

A common scenario for this application would be a family wishing to track the cognitive status of a loved one with dementia or Alzheimer’s disease. This individual would be the primary user as they would be the one launching the application in order to ask questions. The application will allow them to skip as many pictures as they’d like, or they can stay focused on one particular image until their curiosities are satisfied. The application will provide the primary user with answers to their questions while simultaneously logging results to a database. This database would then allow for family members to view an HTML page in order to receive metrics on the individual’s cognitive visual recognition health.

2. Approach

2.1 – Data Sources

The primary data source when the application launches will be an image database repository that family members can submit to via the HTML page. These would most likely be common locations that the patient should be familiar with to hopefully gauge if they’ve retained their familiarity with those locations. If not, how much has it declined? Once the application has loaded, the relevant information will come via the Google Voice Speech Recognizer as the patient asks questions concerning the image. This will be stored in a variable that is then passed to the rest of the application’s workflow.

2.2 – Analytical Tools and Tasks

The main analytic tool that will be used for the metrics component of this application will be done with the Google Charting API. This API allows for our stored database information (imageID, question, answer) to be rendered as user-friendly charts that will be displayed via an HTML page. Bar graphs and line charts will be featured heavily to measure the frequency of the patient’s questions in relation to the image that they decided to focus upon. Over time, this can be leveraged to determine trends that could possibly predict the patient’s rate of cognitive visual recognition impairment. In addition to identifying trends, these types of graphs will help recognize outliers which could signify the sudden emergence of delirium.

2.3 – Expected Inputs/Outputs

The primary input for CVRT (Cognitive Visual Recognition Tracker) are the questions that are asked via voice recognition by the patient. After this speech is recognized and assigned to a variable, it is then passed to the Spark Server alongside the image details to perform the image analysis. This combination of inputs is then used to generate an appropriate answer to the user’s question as the output. Another output that can be expected is the write to the database of the image ID, question, and answer. These will then output to an HTML page for statistical viewing.

3. Related Work

In regards to the Visual Question Answering space, similar work has been done related to the questions being asked of an image with answers generated in response. These demos exist online and serve as a form of introduction to the domain itself, however they do not apply to the medical field. There is also research that has been done regarding impaired visual recognition as a way of predicting the onset of Alzheimer’s, but this research has not been leveraged in a deep-learning application.

Visual Question Answering Demo: <http://visualqa.csail.mit.edu/>

Visual recognition memory in Alzheimer’s disease: repetition-lag effects: <https://www.ncbi.nlm.nih.gov/pubmed/18568983>

Impaired visual recognition memory predicts Alzheimer’s disease in amnestic mild cognitive impairment:

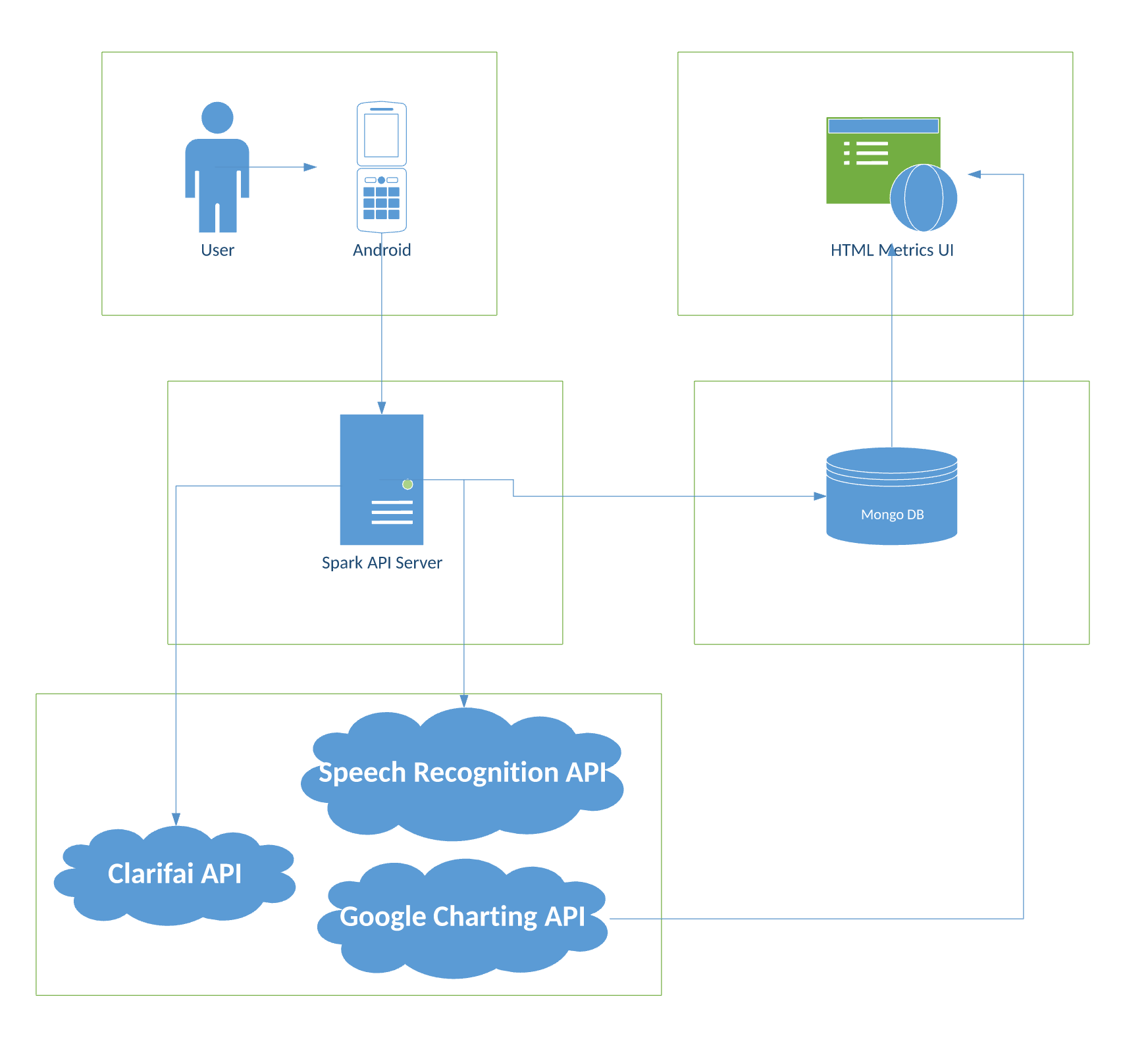
<https://www.ncbi.nlm.nih.gov/pubmed/23572062>

4. Application Specification

4.1 – System Specification/Existing Services Used

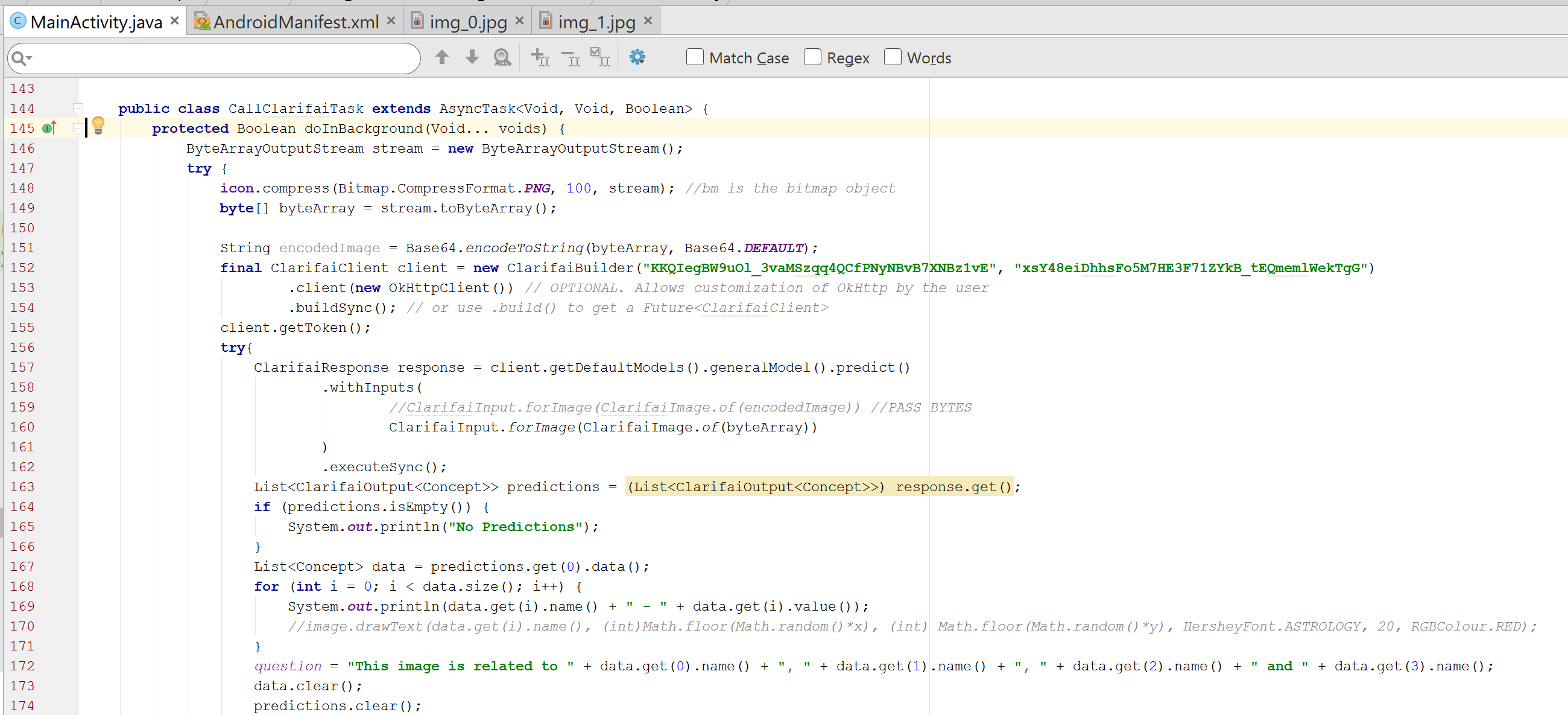
|  |  |  |
| --- | --- | --- |
| Service Name | URL | Description |
| Spark API | https://spark.apache.org/docs/latest/api.html | Allows for deep learning |
| SQLite(or MongoDB) | https://www.mongodb.com/ | Stores data as text in the cloud |
| Android | https://developer.android.com/index.html | Primary OS for mobile platforms |
| Google Charting API | https://developers.google.com/chart/ | Charting engine for data rendering |
| Clarifai API | https://www.clarifai.com/api | Provides image analysis |
| Google Speech Recognition API | https://cloud.google.com/speech/ | Provides Speech Recognition |

4.2 – Software Architecture



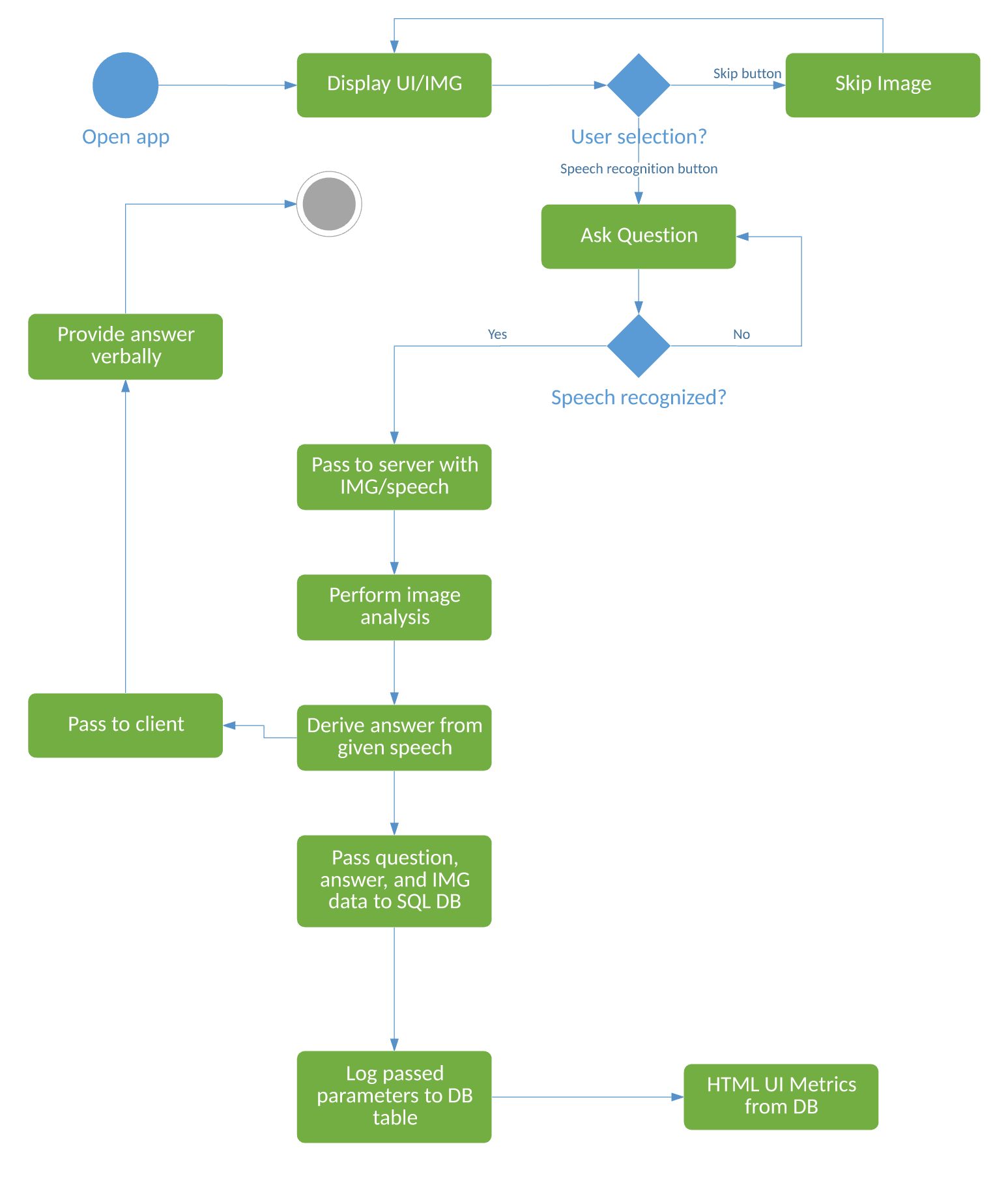
4.3 – Algorithms/Models

Currently the image recognition is being done via the Clarifai API. Once an image has been selected from the user’s phone, it imports as a bitmap and is then converted into a byte array. After this conversion, it is then compressed in order to pass to the Clarifai API. This is done as a background Async task so that it does not interfere with the processes occurring on the main thread.



4.4 – Workflow

The activity diagram below gives a good example of the typical workflow that a user can be expected to experience while using CVRT. The speech recognition button triggers the first of our API usages and prepares the intent for the ‘Ask Question’ node. The question variable is then assigned based on whether the ‘Speech Recognized’ node can interpret the user’s speech or not.

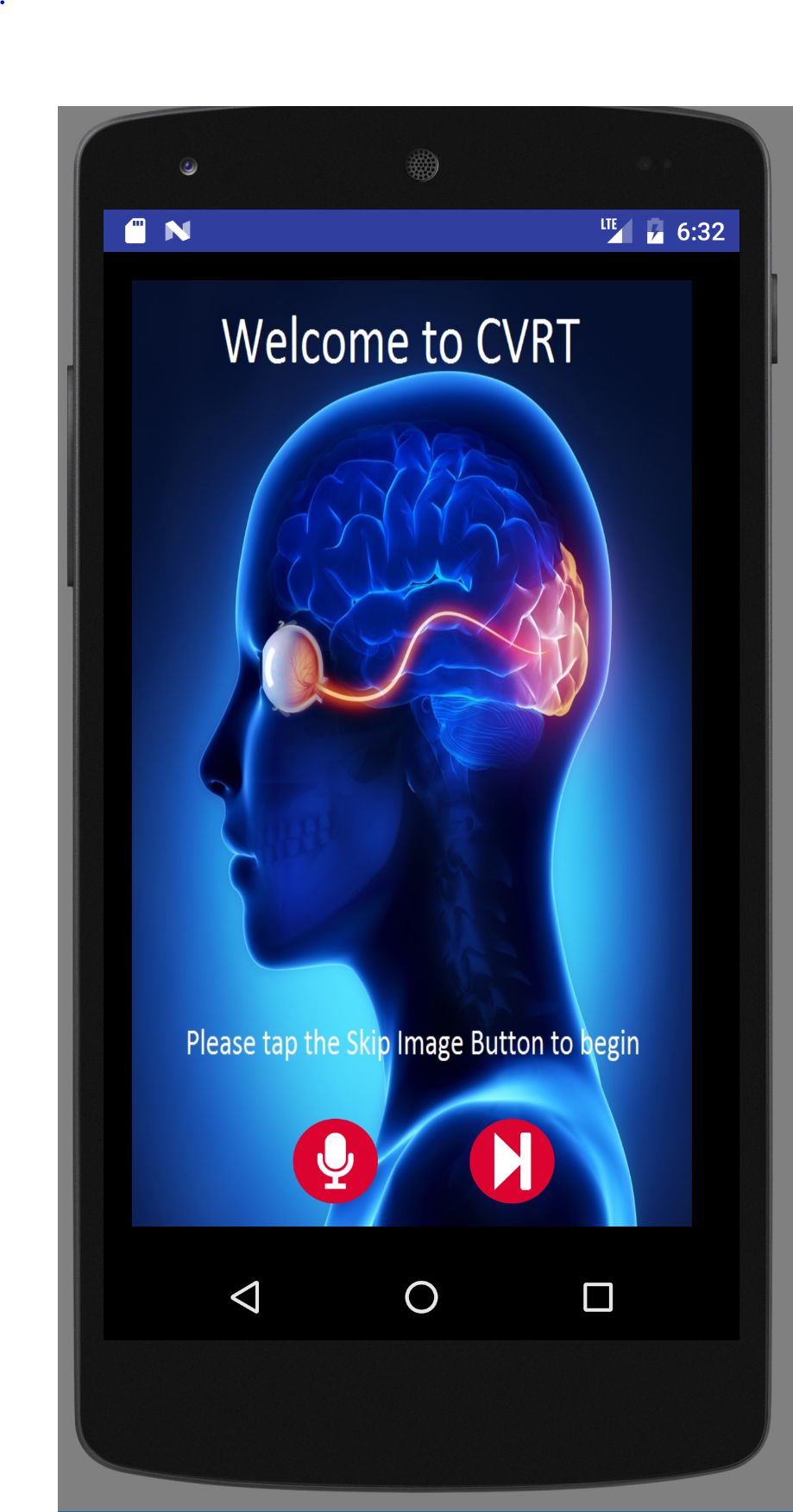


4.5 – Feature Specification

* Images are randomly displayed on UI and remain until user interaction
* User is able to skip images
* User can trigger question input via voice recognition
* Image analysis can be performed on custom images
* Answer can be derived from combination of user’s question and image analysis
* Image ID, question, and answer can be stored in database
* HTML page can read from database, call Google Charting API, and provide meaningful metrics
* Answer can be provided verbally to the user

5. Implementation/Documentation

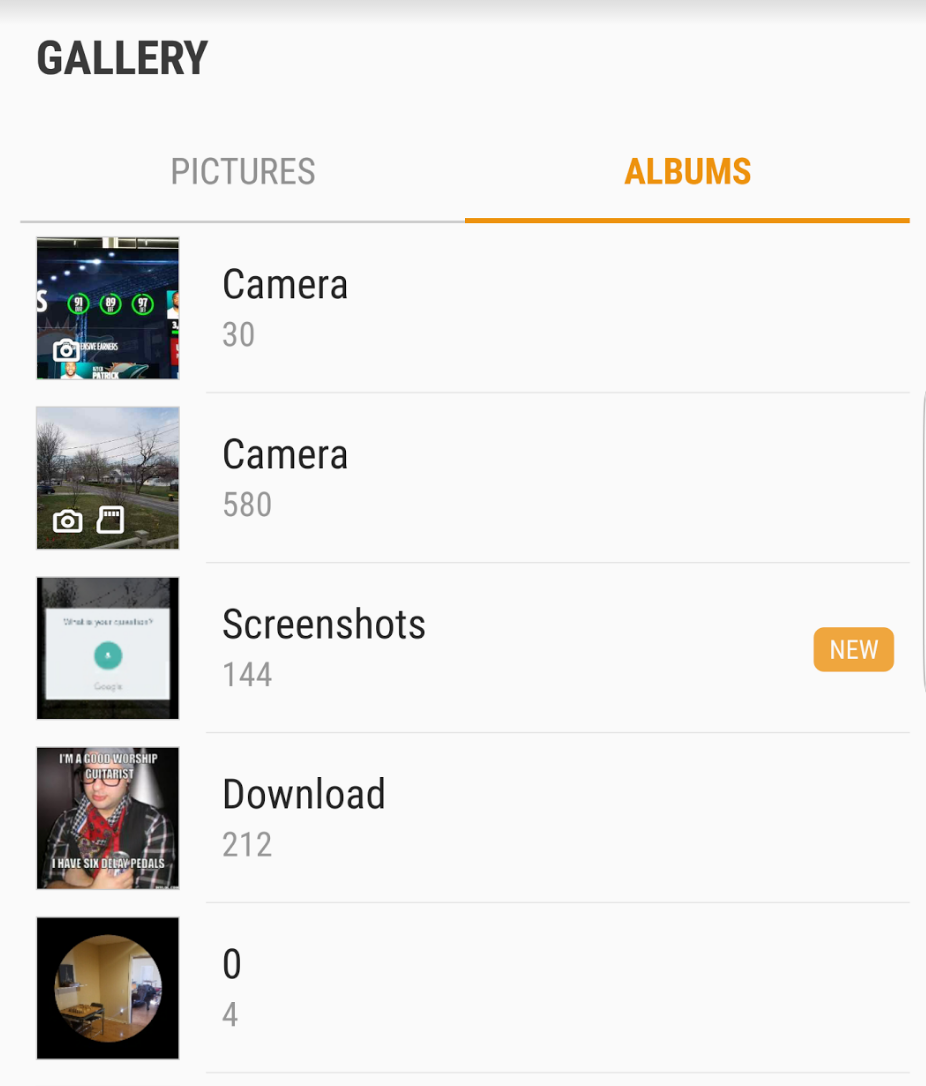
The implementation of CVRT begins with the user opening the application on their Android device. This presents them with a simple UI that has two options (as seen below):



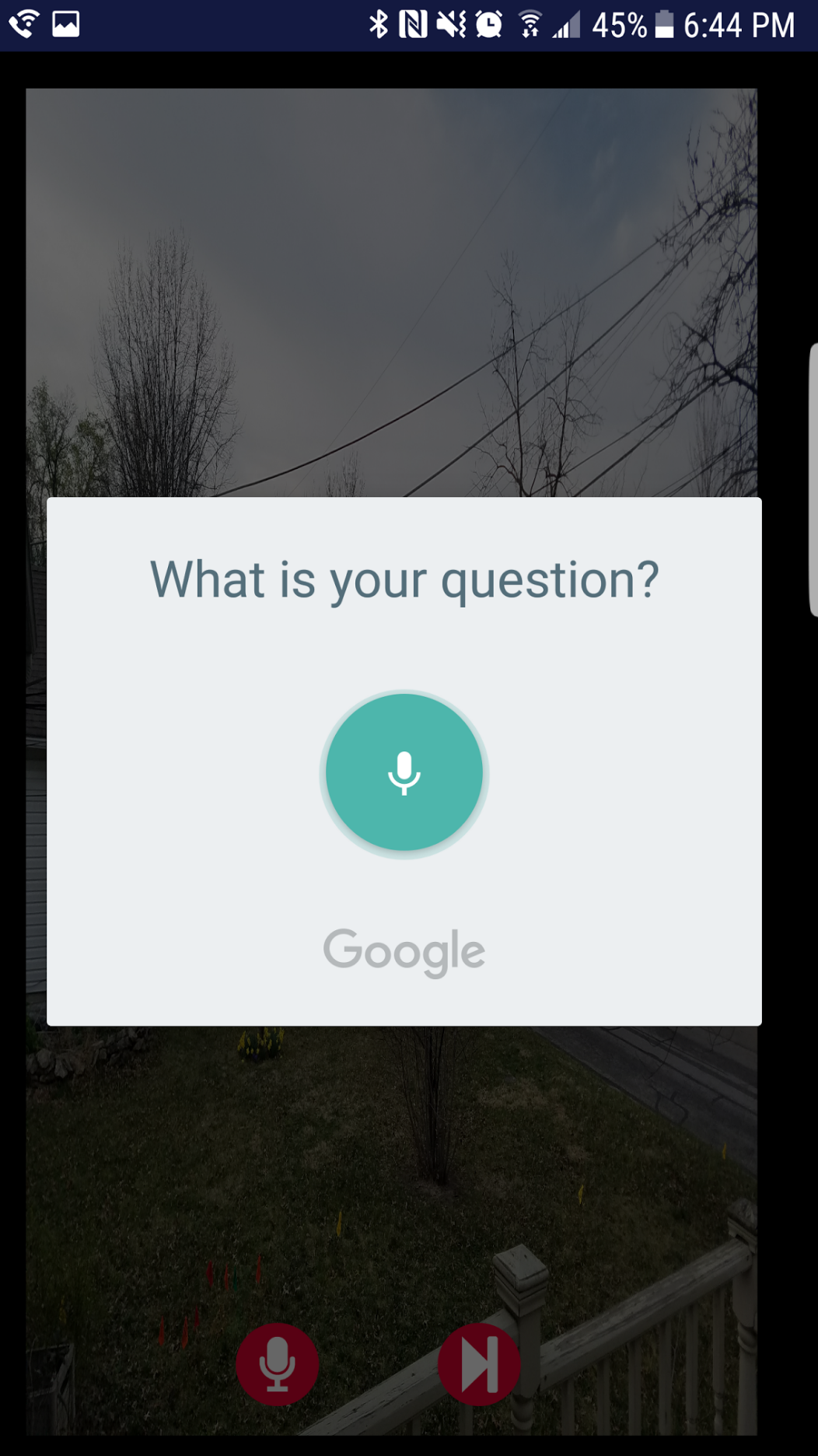
The button on the right acts as a ‘Skip’ option which prompts the user to select a new image from their phone’s gallery in case they are not interested in questioning the existing image. This gallery selection is implemented with the following algorithm upon clicking the button:



This will allow for increased flexibility as the user can choose any image that is on their phone. This functionality is exemplified by the screenshot below of the selection screen.



The button on the left side of the UI acts as a ‘Ask Question’ option and allows the user to verbally dictate a question that they have concerning the image. By clicking this option, the following dialog will display:



If no speech is recognized, then CVRT will prompt the user to speak again. Once speech has been detected, the speech is converted to a string and then stored within a variable. This ‘question’ variable, along with the image information, will be passed to the Spark API Server. The Android application will then create an Async task to leverage the Clarifai API for image analysis, and then provide an answer based on its’ findings. This answer will be verbally returned to the user on the main UI where they can choose to continue asking questions, thus repeating the process, until they are satisfied.

An example of this implementation is as follows. After choosing a picture in my phone’s gallery of a woman on a bicycle, I then tapped on the ‘Ask a Question’ button. This triggered the “What is your question?” speech prompt, which I answered with “Tell me about this picture.” After passing this to the Clarifai API, the application provides the user with a verbal answer. Also, included with the verbal response is a toast message with the same results in case the user is deaf or hard of hearing. An example of these returned results can be seen in the following screenshot:



After the answer has been returned to the user; the image ID, question, and answer will be stored into a database table. This will then be leveraged by an HTML page that takes the database information, calls the Google Charting API, and then displays the report metrics. This web portal will be viewable to the patient as well as any family members/doctors that would be interested in the individual’s cognitive progress/decline.

6. Project Management

6.1 – Project Plan

As I am the only member of this team, task responsibilities as well as contributions are 100% done by myself. Because of this, I must keep my pace quick in regards to development/testing. My plan is to always have a stable build at each project report stage, and to make substantial feature implementation that builds on one another, as opposed to having disparate features that are poorly actualized. By the second iteration, I am hoping to have integrated the Spark Server fully with the existing Android code, and to then have an object formed that would be ready for a database update. For Iteration 3, I would expand the application to include the database relationship and begin work on the Google Charting API functionality. On the final iteration, I am planning to have metrics displayed on the web portal based on the Charting API work done in iteration 3. This would also have the final completed algorithm for the question answering as well as any neural network tweaks that need to be done as the course progresses.

6.2 – Work Completed

So far, I have completed the Android user interface and provided the entry points to the application’s main functionality. I was able to implement an algorithm that completes the Skip button’s functionality so now the remaining workflow focus is on the ‘Ask Question’ feature. On this front, I was able to add the API code to launch the Google Speech Recognition prompt, and to then store that recognized speech as a string variable. As a means for testing, I have also added the code required for the application to verbally speak back to the user. Code has also been added that links the Spark Server, which was also designed, to the Android application; albeit currently with only GET functionality tested.

For Iteration #2, I changed the UI to include a redesign of the home page and buttons. By making both buttons larger and red, they are now more pronounced and friendlier to the user. I also replaced the Skip button’s shuffle functionality, which only shuffled between pre-existing images, to a user gallery selection so that any image can be used for the application. Because of this lack of a default pre-existing image, I added a home screen that prompts the user with app instructions and waits for their input. During this iteration I also added the Clarifai functionality so that any image can be passed from the Android app to the API in order to receive image recognition results. As a result of this, I added the functionality for the application to speak the answer back to the user so that the interaction is more conversational in nature. Finally, I added error handling to properly route any issues experienced with the app itself.

6.3 – Pending Work

For the next iteration, I will be focusing on the Spark to Android relationship. I will need to find a way of passing the image data as a base64 string from the Android application to the Spark Server. Once I am able to receive the data on the server, I will add the image analysis components to my Spark code as well as the write to the database. The database itself also needs to be configured to support the three columns that are required (imageID, question, answer). There is also work to be done in regards to deriving an answer from a given question. I will need to do further research on this to see how this can best be accomplished. Finally, the HTML page, as well as the Google Charting API code, will need to be created in order to provide the required metrics functionality.

6.4 – Issues/Concerns

My current issue is related to Android being able to push the image data in a proper format to the Spark Server. I believe that this can best be done via a Base64 string as seen in class, however, I need to continue to research the proper methods to accomplish this within my existing code. The other concern is more of an unknown. I would like to do similar question/answering as seen in the Visual Question Answering demo (linked in Related Work), but I am not sure if this is possible with dynamic image analysis. I will need to spend time reviewing their github to see how I can best implement a form of this intelligent answering into CVRT.