**Los Angeles Photo Archive (LAPA)**

**Final Project Report**

****

**Faculty Advisors:**

Dr. Russell Abbott & Dr. Jungsoo (Sue) Lim

**CSULA**

**Industry Liaisons:**

Ted Allen & Alisa Blake

**LA City Bureau of Engineering**

***Prepared by:***

Brian Kan, Bernard Bollinger III, Phillip Han, Sharon Lake, Araceli Lopez, Patricia Luz, Kaila Mayho

***Table of Contents***

[**Introduction**](#_tqm8fkv30fna) **2**

[**Related Technologies**](#_3hovv3fctf0d) **3**

[**System Architecture**](#_cupl0hf00op) **4**

[**Results and Conclusions**](#_q70rmgajy8am) **5**

[**References**](#_ums8kq2dpezi) **5**

# Introduction

We worked with the Los Angeles City Bureau of Engineering to develop an Android application that would meet their Sidewalk Repair Project needs. At the moment, the LA City Engineers are using folder names to keep track of Repair Task photos. Such photos will have information such as Before/After, Repair Task ID, Work Status, etc. Our system, complete with an Azure backend and an Android application frontend, allows them to use their Work/Personal Android phones to upload photos with various metadata fields and tie it to an image ID, where it can be pulled later on.

Our key design principle was to create a functional, bug free experience for the end user. We ended up overhauling the existing UI and created a way for users to interact and pass information to the Android application using a procedurally-generated link. Our overall design made it much easier for LA City Engineers to login and fill out various fields of data associated with their work.

# Related Technologies

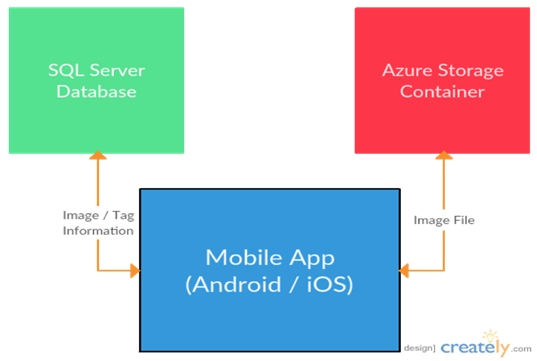
**TIF, EXIF, GIF, JPEG**

One of the technologies that the Bureau of Engineering used during their tests, was using the metadata fields baked into various types of image formats. This didn’t work well because it was hard to pull the metadata from these images on the Android phone, as well as to write this data.

**Azure + Android**

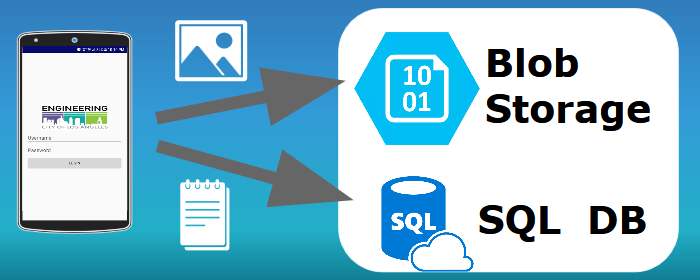
This was our implementation. This implementation is optimal as long as images are properly stored in the Azure Blob and has SQL links pointing to it. Removal of any links in the SQL database will result in overhead within the Azure Blob storage itself ( think, null pointers, or images without pointers ).

# System Architecture



The overall system consisted of 4 components:

1. LAPA Android Application
2. Azure Web Service API
3. Azure Blob Storage
4. Azure SQL Server



This is the general layout of how the app works. The photos are uploaded to the blob storage and metadata is stored on the SQL database. The Azure Web Service API exists in between the Android App and two storage sites. It allows us to interact with the storage sites without the need for any login credentials, only using an API key generated on the Azure website.

To implement the entire system, the Azure Web Service API first had to be published to via Visual Studio. The code was written in C#. Then, we set up the Azure SQL server using our Azure credentials. We also set up After that, we could make calls from the Android application for data via our Azure service adapter.

# Results and Conclusions

Overall, the product was a success. Microsoft Azure, Visual Studio and Android seemed to be interesting technologies and the app turned out to be what the Bureau of Engineering asked for.

However, team cohesion and knowledge was an issue. Teammates were assigned tasks such as learning Azure, iOS Swift and various testing suites such as RoboElectric, Mockito and Espresso and failed to do so.

In the future, things such as offline caching, improved searches by metadata, and geolocational searching can be added.

# References