Skin Cancer Diagnosis using Artificial Intelligence on the Cloud

Team 45

Advisor:
Ashraf Gaffar

Team:
Megan Eberle
Breann Grant
Bariture Ibaakee
Alexander Lafontaine
Evan Nim
Abigail Thompson

Introduction

This project is meant to aid in the process of detecting skin cancer using just an image of the area of skin in question. Powered by Artificial Intelligence (AI), it will aid doctors in their visual analysis of an affected area. As an android application with the AI model hosted on the cloud, it will be easily accessed from anywhere in the world.

This is a tool intended for only doctors. Because it only uses an image, our accuracy is meant to be only slightly better than a doctor's visual assessment, which is about 55-60% accurate. Our goal is not to diagnose a patient, but to supplement a doctor's assessment.

Additionally, we were tasked with comparing two different Cloud Providers for their effectiveness in this application. This comparison is meant to serve as an aid for future developers hoping to achieve the same goal.

Prior Work and Solutions:

Currently there are numerous AI models available and specifically there are already existing AI models that are able to detect cancer.

We will be using existing models to create our model. The shortcoming that current models face that we hope to address with our model is accessibility. We will be using cloud computing to help make our model available to a wider range of people. People all over the world will be able to use our model.

Pros:

-Our model will provide more accessibility by incorporating cloud computing and a user interface

Cons:

-We don't have access to the same sophisticated datasets that other models may be trained on

The National Institute of Dental and Craniofacial Research supported an international study on cancer detection using AI/deep learning. The AI model they created can detect cancer by checking abnormalities in cell size and structure. This study successfully determined the feasibility of using AI to detect cancer cells. In addition to this, the National Cancer Institute has written an article on how AI models are already currently in use. Dr. Ismail Baris Turkbey is a radiologist who with the help of his National Cancer Institute team trained an algorithm that is capable of detecting prostate cancer when given an MRI scan. Figure 1 below shows two MRI prostate scans on top of each other. Both scans have cancer present. The right side of the pictures show where the AI model was able to successfully detect cancer. It is able to determine the specific area in which the cancer is present. This was done by training the model on what symptoms are common in cells with cancer.

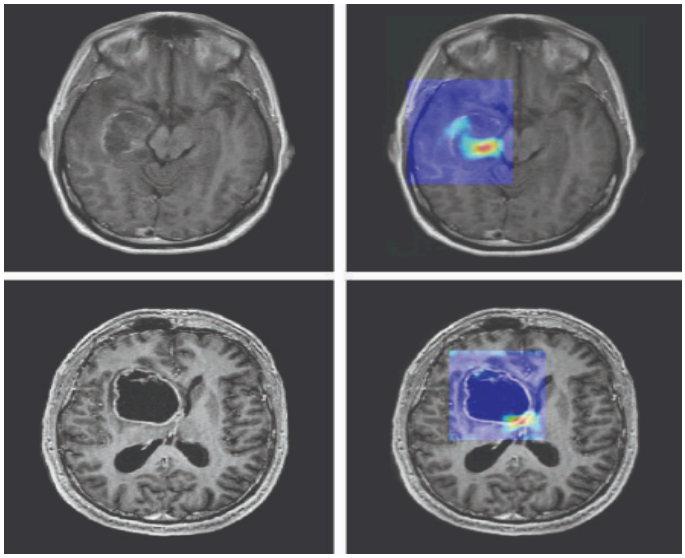


Figure 1 (National Cancer Institute)

Sources:

https://www.nidcr.nih.gov/news-events/2020/exploring-ai-cancer-diagnosis

> "Exploring AI for Cancer Diagnosis." *National Institute of Dental and Craniofacial Research*, U.S. Department of Health and Human Services, www.nidcr.nih.gov/news-events/2020/exploring-ai-cancer-diagnosis. Accessed 18 Oct. 2023.

https://www.cancer.gov/news-events/cancer-currents-blog/2022/artificial-intelligence-cancer-imaging

September 27, 2023, et al. "Can Artificial Intelligence Help See Cancer in New Ways?" National Cancer Institute, www.cancer.gov/news-events/cancer-currents-blog/2022/artificial-intelligence-cancer-imaging. Accessed 18 Oct. 2023.

Revised Design

Requirements:

Functional Requirements:

- The product shall have an Android app to access the model.
- The product shall be accessible from around the world.
- The product shall have the ability to obtain results (cancerous or noncancerous) from an image.

Resource Requirements:

- The product shall run on a device with an Android operating system and camera.
- The model shall be run on a cloud platform.

Nonfunctional Requirements:

- The images obtained through the Android app shall not be saved to any database.
- The User Interface shall be easy and intuitive to navigate and use.

Engineering Standards

- IEEE 730
- IEEE 828
- IEEE 29148
- IEEE 1012
- IEEE 16326
- IEEE 24748
- ISO/IEC 29119

Security Concerns and Countermeasures

1. Sensitive patient data: We do not collect or save data on patients. This includes the images taken through our app and sent to the AI model. No patient images are saved.
2. User authentication: We require an account with a username and password to limit access to the application.

Design Evolution

Frontend Evolution

We originally planned to build a website that would access the AI model, send images, and receive results. We later decided to develop a mobile application in order to utilize the camera functionality included on cell phones. This was meant to be an iOS application, but our team has only Windows computers. After research and trial and error developing in Swift on a Windows machine, we concluded that it would not be feasible for us to develop an iOS application with the resources we had and transitioned to developing an Android application.

Cloud Server/AI Model Evolution

The A.I. model has been successfully changed to a Computer Vision model. After various tests and research, the Xception model was chosen as the objective model to implement. Thus, we

implemented a small version of the Xception model to start writing the code structure and everything necessary to train the model and make predictions with it. After a successful code implementation, the A.I. model was transitioned to a more standard version of the Xception model with an added layer at the end for image classification.

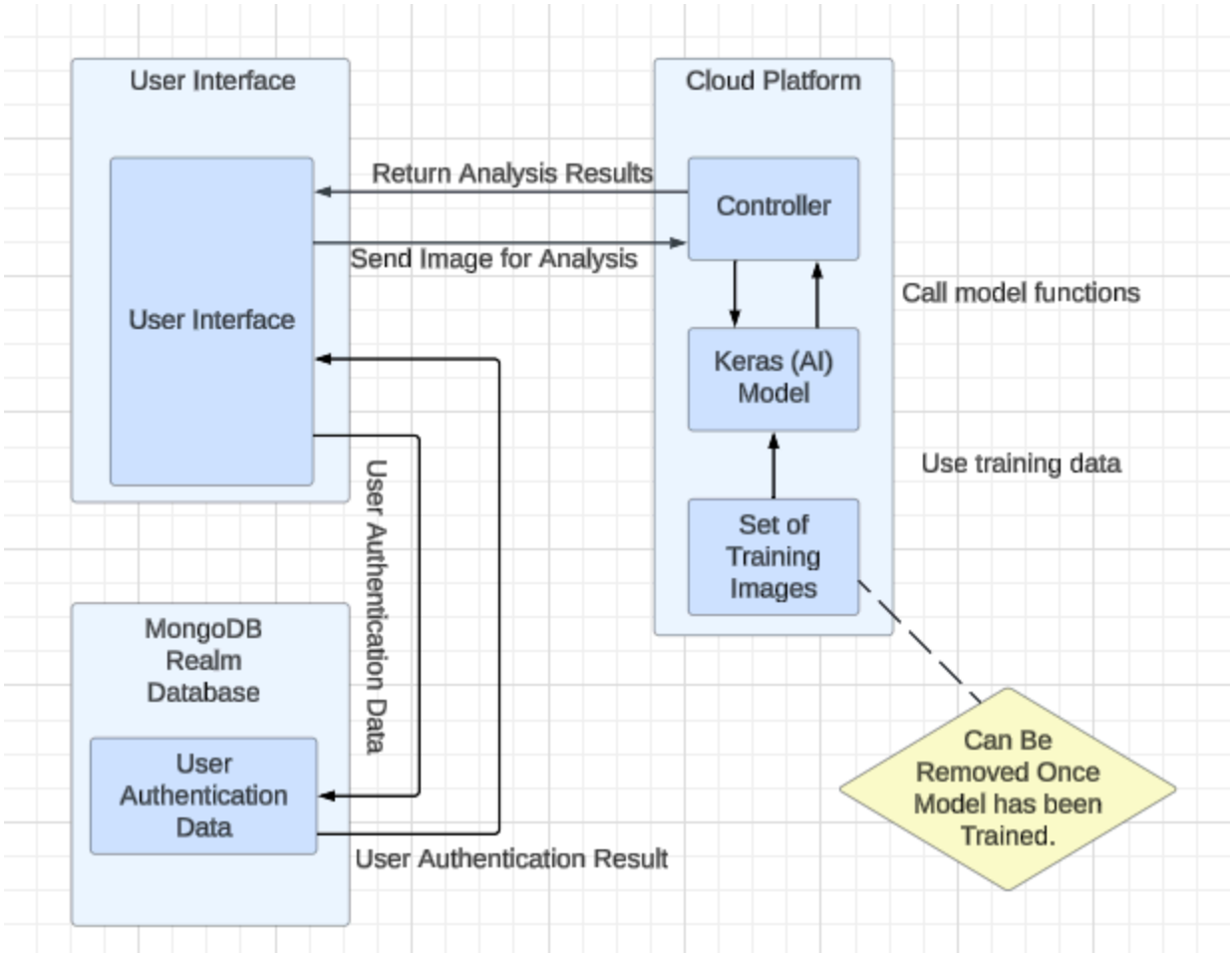
Additionally, there were some modifications added to code based on previous feedback. One of them was using a bigger set of data to avoid overfitting the model quickly. Another modification was the use of data augmentation to train the model. Lastly, we did research in order to pick and choose the best options for the model. All of these modifications helped increase the overall accuracy and reduce the loss of the model in both the training and validation parts of the model's training.

Backend Storage Evolution

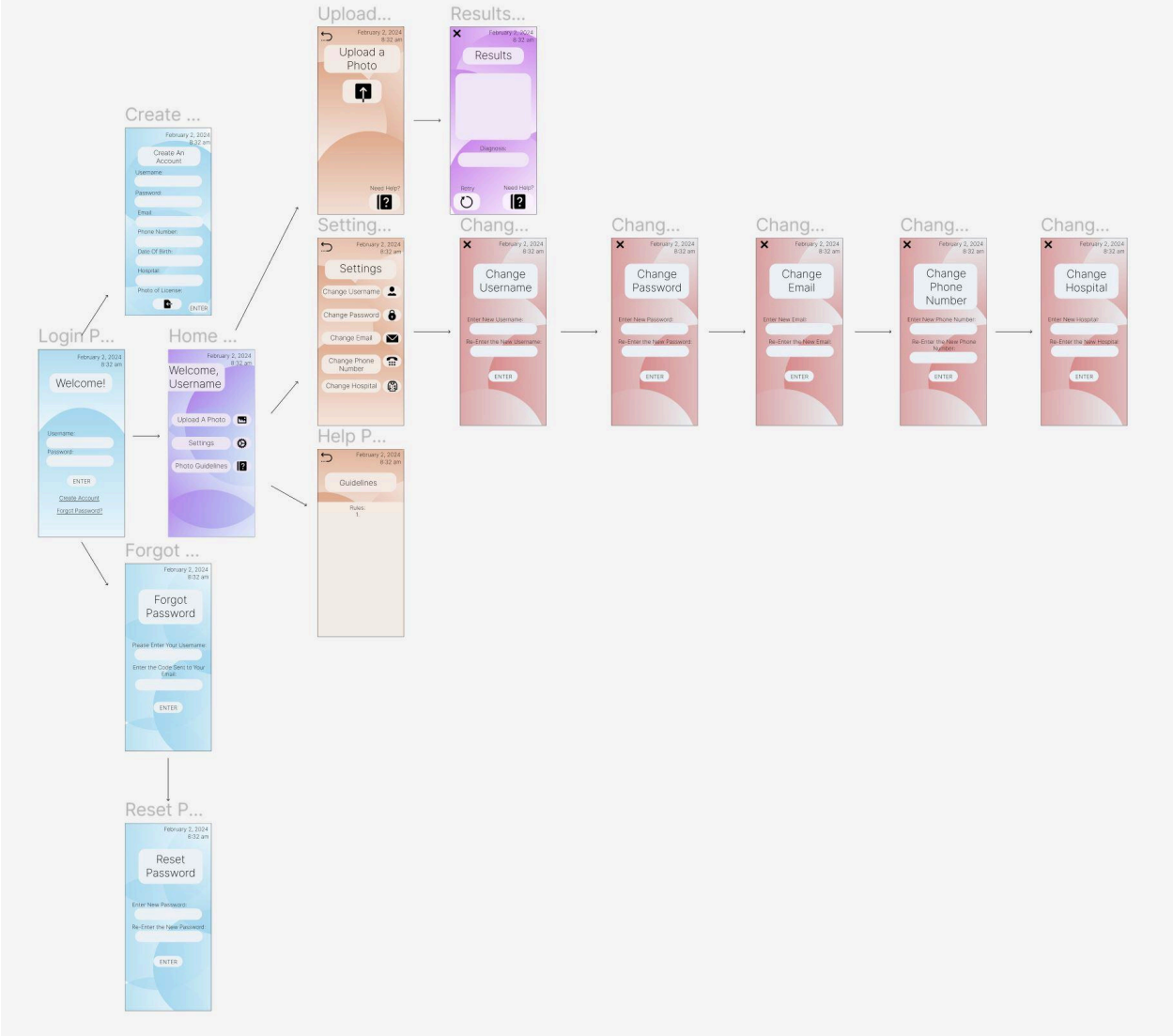
Instead of dedicating a database on a server for the user account information, we decided to utilize MongoDB Realm since the application is on an android studio application and hence would work better and easier with a database specifically catered towards android applications.

Implementation Details

Detailed Design


User Interface
User Interface
Cloud Platform
Return Analysis Results
Controller
Send Image for Analysis
Call model functions
Keras (AI) Model
Use training data
Set of Training Images
User Authentication Data
MongoDB Realm Database
User Authentication Data
Can Be Removed Once Model has been Trained.
User Authentication Result

Description of functionality



Log in or Create Account

Doctors can log in to our app by providing a username and password. They can also create an account by providing personal information and verifying that they are a doctor. They can also change a forgotten password.

Forgot or Reset Password

If a password is forgotten, the user can click on the Forgot Password button on the login page. This allows us to authenticate that they are the right user by asking for their username and email. Then, they will be able to go to the Reset Password page where they can officially reset their password.

Home Page

This is where the user is able to access all of the other options of the application, such as taking a photo, accessing settings, and viewing the image guidelines.

Settings

In the settings page, users can change their account information.

Change Username, Password, etc.

This is where the user is able to change their username, password, email, phone number, and hospital.

Photo Upload

Users can take a photo of an area of skin directly from the app. They can retake the photo or send it to the AI model. This prevents users from storing photos of a patient's skin in the cell phone system. They then see the results of the image.

Results

The results page displays the results given by our AI depending on the photo given.

Image Guidelines

This page gives users a few guidelines for taking photos to receive the best results.

Notes on Implementation

The 30,000 image collection was obtained from the International Skin Imaging Collaboration (ISIC), who provide skin image collections for A.I. training. On top of that, shuffling and data augmentation was used in the image collection to provide more variety to the A.I. model.

Keras and Tensorflow libraries were used to prepare the training dataset and the Xception model. It is also used to make predictions of new data. Although we ideally would have trained our model in the GPU for efficiency, the A.I. was trained in the CPU due to problems with the Tensorflow GPU library not installing properly, Keras and Tensorflow GPU library coordination problems, and the limited budget for this project.

AWS and GCP were used for a small comparison to see the price, the usage, and the time it takes to train per iteration. GCP is used for the final version of the project. Comparisons and recommendations are detailed below.

Testing

Process

An ec2 instance from AWS and a compute engine from GCP were used. These were the only two instances used in order to keep the comparison as similar as possible. The following table shows the specifications chosen for each instance so that they are as similar as possible:

Specifications	EC2 (AWS)	CE (GCP)
VCPUs	8	8
Memory (GBs)	32	32
Disk (GBs)	70	70

Due to budget constraints, both models were trained for 5 iterations with a batch of 1. The same file structure was used in both linux instances. In other words, everything was selected so that it was as close as possible to each other.

In the code, other functions were added for testing and taking measurements, like the total time for each activity, dates for start and end, model summary, and the history of the model.

After the setup, the code was run in both instances for 5 iterations to collect data.

Results

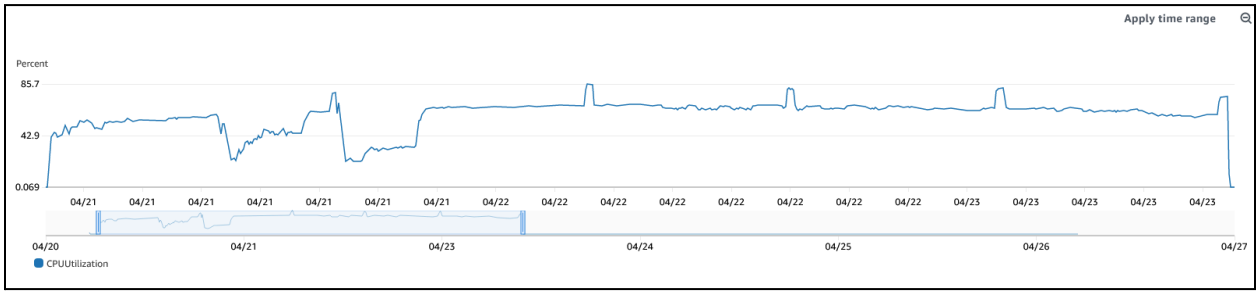
The following table shows the time it took for each training activity to complete:

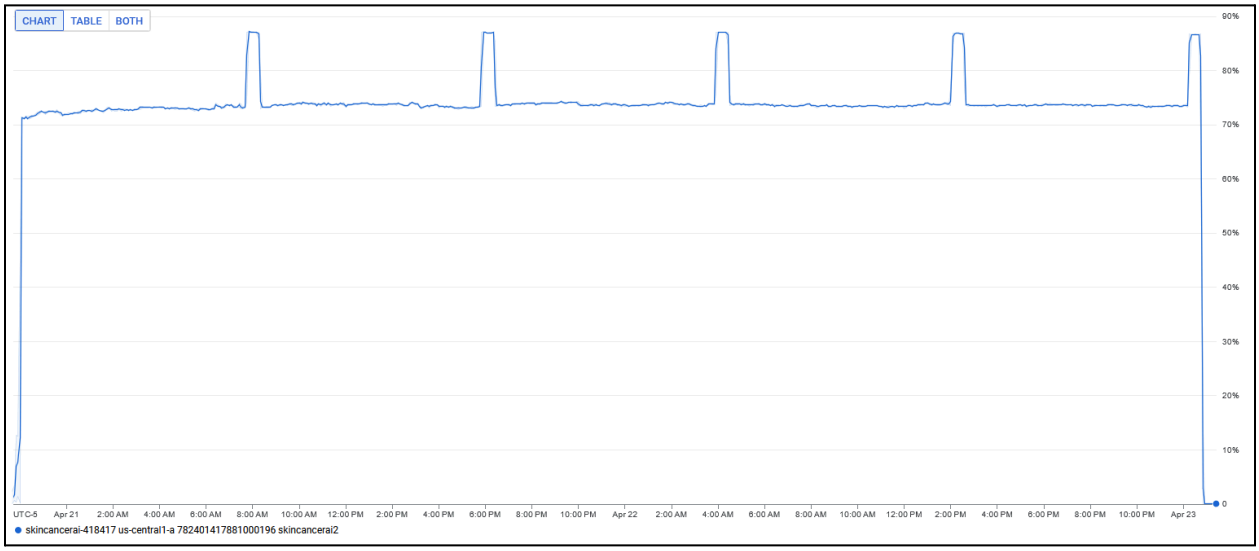
Activity	**AWS Time (seconds)**	**GCP Time (seconds)**	**AWS Time (hours)**	**GCP Time (hours)**
Dataset Generation	4.6235	0.991		
Dataset PreProcessing	0.5168	0.4031		
Model Building	2.0074	1.1855		
Model Training	216159.9248	183061.7765	60.04442356	50.85049347
Model Saving	0.9682	0.8918		

Given the hours above, some approximate calculations can be made about how many hours each iteration took:

	AWS	**GCP**
Hours per Iteration	12.00888471	10.17009869

The following is the CPU usage during the whole process (top graph is AWS and bottom graph is GCP):





Lastly for cloud platforms, the following are the approximate prices for the use of the following cloud computing resources for 5 training iterations (the recommended amount of iterations is between 25 to 50):

	AWS	**GCP**
Cost (US Dollar)	$49.80	$57.50

Due to budget constraints, the A.I. model was trained on a local computer for 25 iterations. It achieved a 98% accuracy with a loss that fluctuates below a loss of 1. However, there may be a chance that the model is biased towards the benign label because of the difference between benign set of images and malignant set of images.

Broader Context

Area	Description	Examples

Public health, safety, and welfare	Our project impacts any person who may have skin cancer as well as doctors by giving them a less invasive option to identify skin cancer.	Reduces (but does not eliminate) need for invasive procedures.
Global, cultural, and social	People who live in regions that get more sun may be more impacted by this project.	People closer to the equator may get more sun than those who live further away.
Environmental	Our project could contribute to climate change by burning fossil fuels and increasing greenhouse gasses.	Since AI uses a lot of computing, it uses a significant amount of energy.
Economic	Our project could decrease the financial responsibility of patients and insurance providers.	Our project provides a non-invasive method of detecting skin cancer, which is generally less expensive.

Conclusions

With regards to performance, GCP offered better results with the limited amount of data we had. It proved to be a better choice for training with a CPU. It was faster at loading the data and training model. A drawback with GCP is price. In our case, the price difference was up to 8 dollars. While this isn't a relatively significant amount, over time the price difference does make a considerable difference.

An important potential consideration for future technical development with our app would be adding encryption and decryption to the overall application for the pictures. When a picture is sent to the cloud, it would be encrypted on the android studio side and then decrypted before being used by the model. A future project would also benefit from having a budget to use, test, and compare more expensive resources in the cloud, like instances with GPUs or TPUs.

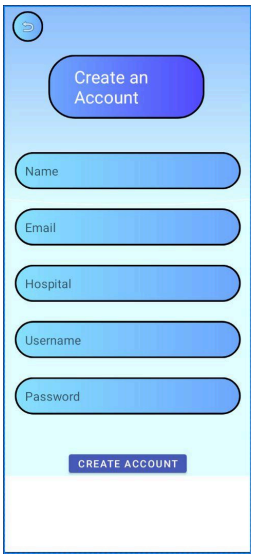
Appendix 1

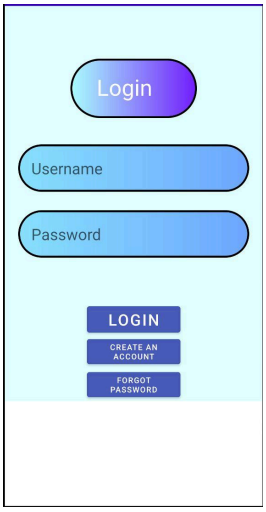
Requirements to setup/demo/test:

1. Install Android Studio
2. Clone Git repository (https://git.ece.iastate.edu/sd/sdmay24-45) to local file system.
3. Open the Android Studio project located at sdmay24-45/Frontend/app.

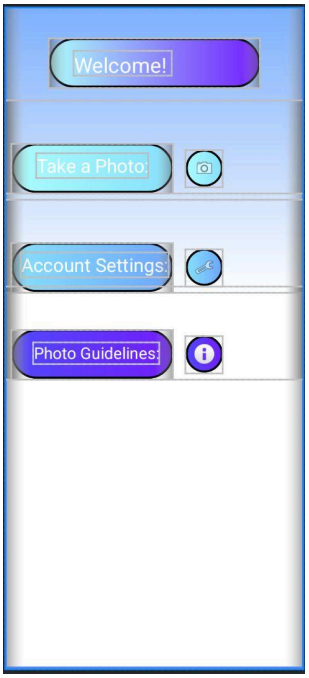
4. Create a device through Android Studio - On the right hand side, click Device Manager, then the '+' button at the top of the panel. Follow the prompts to create a device.
5. In the very top toolbar of Android Studio, click the green play button to run the application.

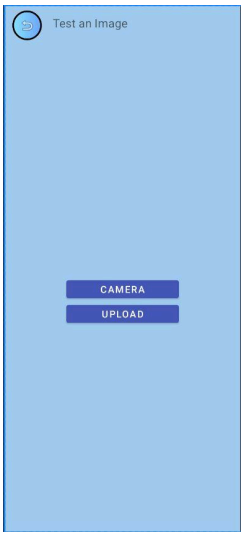
Using the application:

1. Create an Account
   a. 

2. Login
   a. 

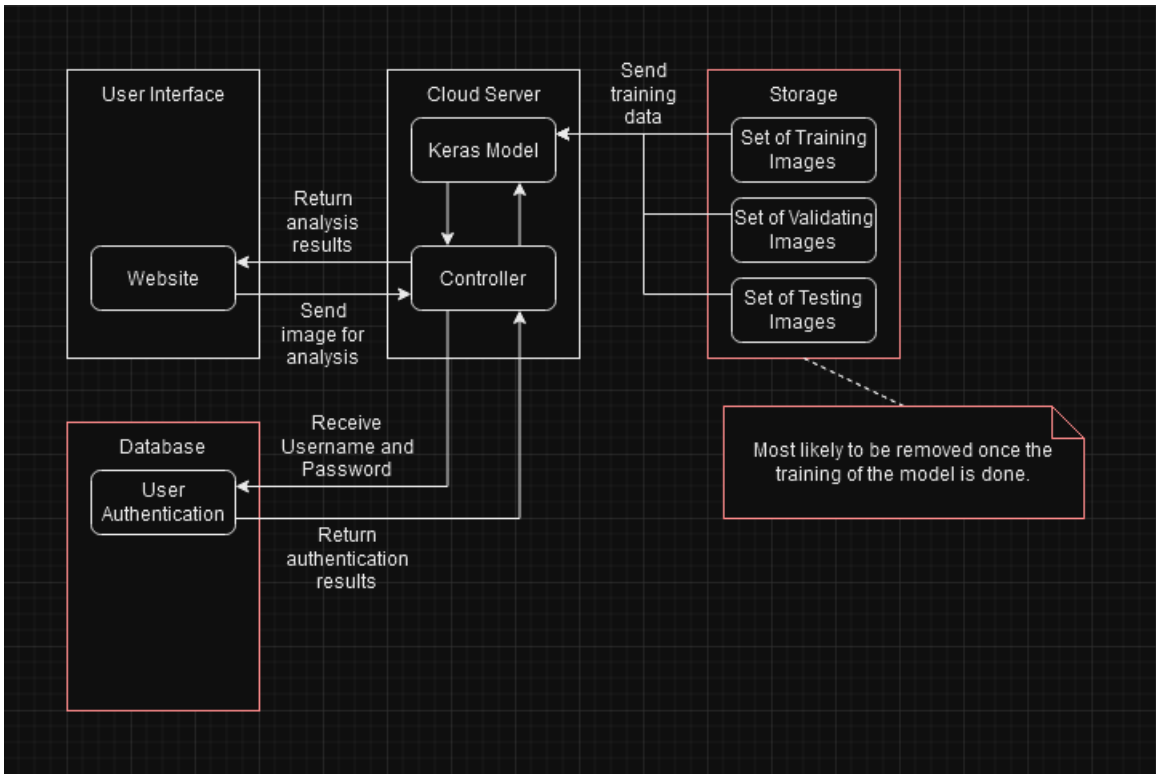
3. Select "Take a Photo"

a. 

4. Take a Photo
   a. 

   b. You can either upload a photo, or use your camera.
   c. Receive your results through the app

Appendix 2



Initial design before client specifications and requirements changed. Initially it would have been 1 cloud platform, but later it was needed to compare two cloud platforms, so the design had to be changed so that the design for both platforms were as similar as possible. Aside from that, the sets of images were decreased to only the training set because of techniques that allowed the code to validate using a sample from the training data.

Appendix 3

We looked into other potential ways we could implement our app, such as creating an IOS app using Swift.

Appendix 4

All code for this project can be found at: https://git.ece.iastate.edu/sd/sdmay24-45