12-8-2023

Chronic Kidney Disease Android Application

Paul Le

SANTA CLARA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Date: December 8, 2023

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Paul Le

ENTITLED

Chronic Kidney Disease Android Application

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

David C. Anastasiu (Dec 8, 2023 09:36 PST)

Thesis Advisor

Silvia Figueira

Silvia Figueira (Dec 8, 2023 09:46 PST)

Department Chair

Chronic Kidney Disease Android Application

by

Paul Le

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Science and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
December 8, 2023

Chronic Kidney Disease Android Application

Paul Le

Department of Computer Science and Engineering
Santa Clara University
December 8, 2023

ABSTRACT

Chronic kidney disease is increasingly recognized as a leading public health problem over the world that affects more than 10 percent of the population worldwide, where electrolytes and wastes can build up in your system. Kidney failure might not be noticeable until more advanced stages where it may then become fatal if not for artificial filtering or a transplant. As a result, it is important to detect kidney disease early on to prevent it from progressing to kidney failure. The current main test of the disease is a blood test that measures the levels of a waste product called creatine and needs information such as age, size, gender, and ethnicity. They may be uncomfortable, can lead to infections, and are inconvenient and expensive.

I will re-engineer an Android application for Chronic Kidney Disease detection by working on test strip detection zone localization, detection zone focus, capture quality, and dynamic model loading. This uses a smartphone's camera and allows users to manually focus on an area of the view to analyze. The camera detects where the test strip and its detection zone is and checks if it is in focus. The pixels are sent to the machine learning algorithm. The application can quickly determine the health of a users kidney and can display it. By only requiring a few drops of blood and an Android smartphone, it is very important for those who cannot afford insurance or live in developing countries. This can make a huge difference in early detection of CDK in these areas where people would otherwise disregard the tests in fear of not having enough money.

ACKNOWLEDGMENTS

I would like to thank my advisor, David C. Anastasiu, for guiding me throughout this entire project.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Problem

Our kidneys are an essential part of the human body that specializes in the filtration and removal of toxins and wastes. Chronic kidney disease is increasingly recognized as a leading public health problem over the world that affects more than 10 percent of the population worldwide. Although we can survive with just one kidney, when the amount of functioning kidney tissue is diminished significantly, there is a case of chronic kidney disease, where electrolytes and wastes can build up in your system. The danger one must account for is that kidney failure might not be noticeable until more advanced stages where it may then become fatal if not for artificial filtering or a transplant. Some symptoms include nausea, vomiting, fatigue, cramps, high blood pressure, and chest pain.

It is important to detect kidney disease early on to prevent it from progressing to kidney failure. The current main test of the disease is a blood test that measures the levels of a waste product called creatinine. With information such as age, size, gender, and ethnicity, the doctor is able to determine the millimeters of waste that one's kidneys should be able to filter in a minute. In general, if a person's filtration rate is less than 90 milliliters per minute, it may be a cause for concern. Other tests include urine tests, ultrasound scans, MRI scans, CT scans, and kidney biopsies. However, blood tests may be uncomfortable for those who have to take out their own blood, can lead to infections, and are inconvenient and expensive as it requires trained medical personnel. This requires a visit to the doctor, a trip to a lab to extract the blood, and the use of expensive machines that most people cannot afford, which takes up a lot of time.

1.2 Solution

I will re-engineer an Android-based experimentation application that will be used for Chronic Kidney Disease detection by improving components of the app, including test strip detection zone localization, detection zone focus, capture quality, and dynamic model loading. Using a smartphone's camera directly in the application, users will be able to manually focus on an area of the view in order to take sharp photos of the test strip that will then be analyzed. The camera has to detect where the test strip is and where the detection zone of it is and then check if it is in focus.

If so, the pixels of the detection zone will be sent to the machine learning algorithm to be analyzed to get the results to the user. By scanning the test strip, the application will be able to determine quickly and accurately the health of the user's kidney, and all of this can be done at home with the images being stored securely online. This will provide a massive advantage over blood tests as it requires only a few drops of blood compared to a vial in a blood test. In addition, this app can be used in existing inexpensive phones with test strips that are less than a dollar each, making it very important for those who cannot afford insurance or live in developing countries. This can make a huge difference in early detection of CDK in these areas where people would otherwise disregard the tests in fear of not having enough money.

1.3 Functional Requirements

The Kidney Health App must be able to log users in with a password and email in order to save and store information that is specific to each person. It must be able to take photos using the smartphone's camera and to focus on a given part of the screen. It must check that the area is in focus or it will change the focus factor until it is focused, then it can take a picture. To do this, the algorithm will be taking screenshots from the camera. Given a screenshot that was just taken, the app has to see if it is in focus and find out where the detection zone is. This is done by detecting where the test strip is and where the detection zone of the test strip is. If it is detected, then it will be checked to see if it is in focus. The pixels will then be extracted and sent to the machine learning algorithm to get the prediction with the results. If the user wants to save the image, then they will be able to save it. If the detection zone is inadequate, then the app will try again from the beginning. In addition, if the detection zone is not in focus with the algorithm, the user will be able to use manual focus. This is done by taking the center point of the detection zone, deciding which way to change the aperture of the camera, and the function telling us if we are in focus by how much. Otherwise once it is finished, the user will be displayed a screen that displays all of the tests that have been taken.

1.4 Constraints and Technologies Used

This app will be constrained by the Android operating system using the Camera2 API. It will be able to access functions that are specific to the Camera2 API to access the manual focus mode of an Android smartphone. Development of the Android application will be done through Android Studio using the Kotlin language. This application will not run on Apple devices that are running the IOS operating system. It will make use of computer vision algorithms, deep learning prediction algorithms, and database storage. The application will only be designed to work with Android smartphones.

1.5 Use Cases and Design Rationale

Once finished, this application will be used in lower income areas where people cannot afford expensive hospital blood tests. The test strips that are less than a dollar each require only a few drops of blood compared to a vial, and this makes it very important for those who cannot afford insurance or live in developing countries. Android phones are much more widespread in these areas due to the lower cost, and making them quick and inexpensive takes out the time spent on making a trip to the hospital and waiting for a result. On the other hand, iPhones tend to have higher cost, leading to less customers in these areas. Current blood tests require trained medical personnel, including a visit to the doctor, the lab for extraction of the blood and the use of expensive machines. This takes up a great deal of time and money that many people in low income countries may not have or be able to afford. The application will be able to deliver the results at home in a matter of minutes.

1.6 Risk Analysis

Issue	Probability	Severity	Solution
Unable to manually focus	Likely	Not Severe	Use Autofocus instead
Falling behind on app development	A Little Likely	Slightly Severe	Create and update timeline often
Lacking direction	Unlikely	Severe	Consult with Dr. Anastasiu

1.7 Diagram and Timeline

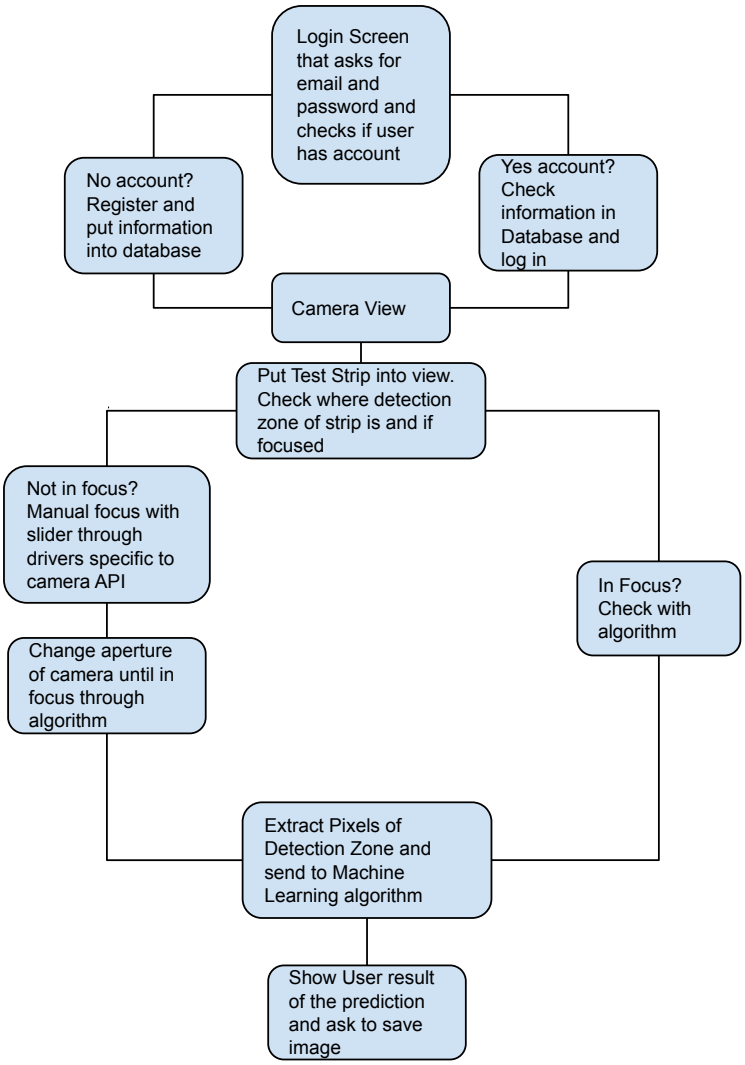


Figure 1.1: Diagram flow chart

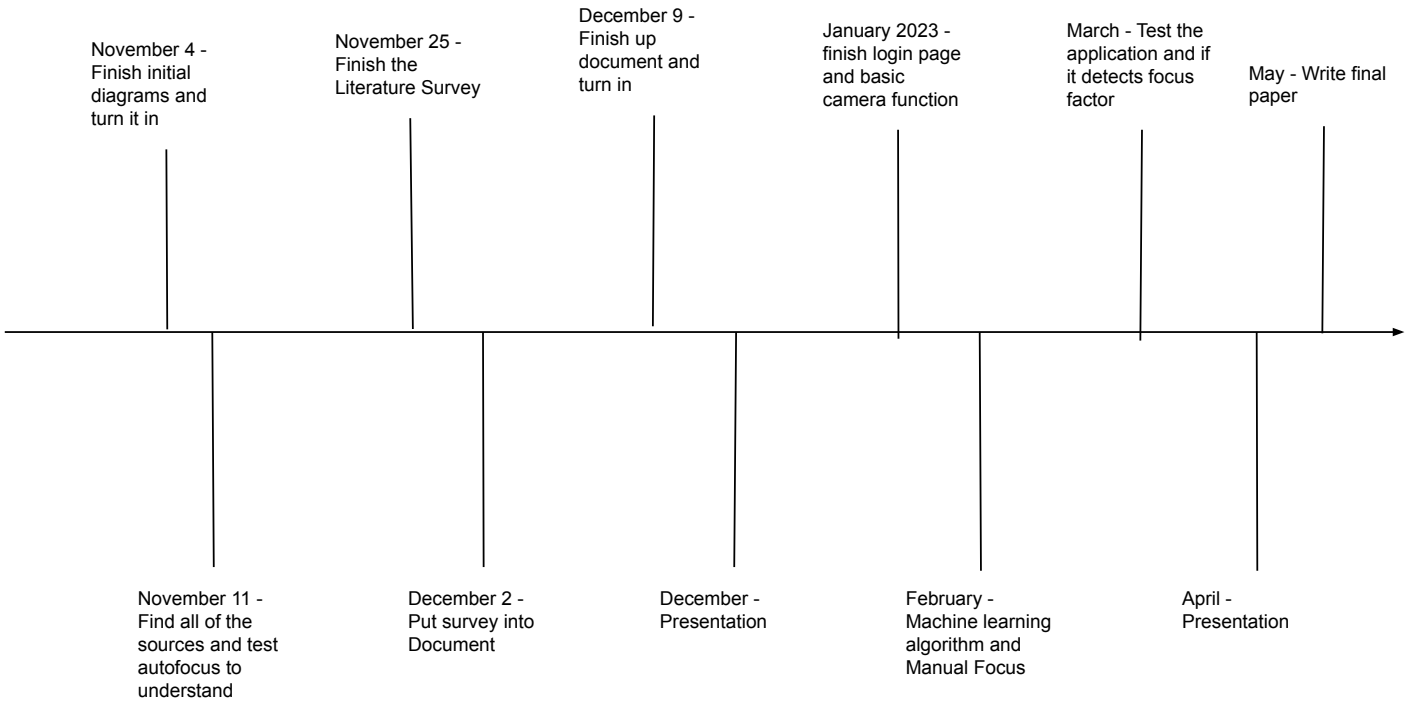


Figure 1.2: Timeline

Chapter 2

Literature Survey

2.1 Introduction to the Literature Review

In this chapter, we review works by researchers related to several areas of the project, such as the smartphone camera, camera focus, deep learning, and machine learning.

2.1.1 Smartphone Camera Technology

Blahnik and Schindelbeck [1] focused on defining smartphone imaging technology and its applications. Smartphone cameras have been a loyal companion for most people now due to their constantly improving portability, connectivity, and image performance. For the last couple of years in smartphones, more and more of them are debuting with multi camera systems, along with 3D acquisition systems such as time-of-flight sensors. When the iPhone came out in 2007, although it was outranked in some areas by other previous smartphones, the revolutionary parts were the controls, which featured only one button and a touchscreen that allowed for precise control, eventually replacing mechanical keyboards. With cameras, there are several requirements. The image sensor should be as large as possible to allow as much light as possible. With light, fundamental disadvantages such as image noise, reduced dynamic range, longer exposure times, and motion blur are all reduced.

Zhang et al. [2] discussed the autofocus system and the evaluation methodologies. The autofocus system in cameras has rapidly become more popular over the last several decades. Autofocus is directly embedded in a camera and is designed to bring the best focused image to the viewer within a few seconds without them needing to do it manually. It is a feedback control system that is composed of three parts. The first is a motor that drives the camera lens to move along the optical axis to find the lens position of best focus, the second is a group of lenses that converge light rays to the image sensor, and the third is a processing unit that computes the focus value per frame and issues control signals to the motor. The evaluation is determined by accuracy and speed. One popular method due to its performance and low cost is contrast-based autofocus. Its basis is the search algorithm, with the simplest one being global search. This is when the autofocus system measures the focus value at every lens step. Other algorithms include rule-based search,

model-based search, coarse to fine search, curve-fitting-based methods, and machine-learning methods.

Abuolaim et al. [3] defined and explained autofocus for smartphone cameras. Autofocus on smartphones is the process of determining how to move a camera's lens so that a certain scene content is focused. Autofocus systems use algorithms such as contrast detection and phase differencing. An issue is that determining a high-level objective regarding how to best focus a particular scene is less clear as different smartphone cameras "employ different AF criteria; for example, some attempt to keep items in the center in focus, others give priority to faces while others maximize the sharpness of the entire scene. The fact that different objectives exist raises the research question of whether there is a preferred objective." The work in this paper aims to revisit autofocus for smartphones within the context of temporal image data by describing the capture of a new 4D dataset that provides access to a full focal stack at each time point in a temporal sequence. Using this dataset, we have developed a platform and associated API that mimic real AF systems by restricting lens motion within the constraints of a dynamic environment and frame capture. API calls let algorithms simulate lens motion, image access, and low level functionality such as contrast and phase detection.

Nguyen et al. [4] discussed the challenges and issues for optical camera communication based android camera 2 API. Smartphones are constantly evolving and upgrading hardware and software. For the camera, the Camera 2 Application Programming Interface is the forefront of the software for smartphone cameras. It configures the camera device as a pipeline, allowing it to change the parameters for capturing a single frame by capture request, raptures the single image per the request, and outputs one capture result metadata packet with all of the requesting information about the state of the device at the time of capture, and the final settings used in TotalCaptureResult. Displaying camera previewed images are done through Surface View, Texture View, or Image View. The CaptureRequest describes all of the capture factors needed by a camera device to capture an image. Using the Camera 2 API, developers can set the manual shutter speed, frame rate to image data output support, and use logical and easy-to-use algorithms. It allows for exact frame rate or shutter speed configurations, although this depends on the hardware of the camera.

Nishiyama and Mizoguchi [5] designed a cognitive support system for cosmetic skin analysis support by having users take pictures of their skin to transmit. This is done for a skin diagnosis that will be used to promote the sales of cosmetics. They do cognitive support through a smartphone so that advice on the operation is transmitted to the user in real time, allowing them to perform their own skin diagnosis.The process of this application is as follows. The user selects a domain by tapping the rectangle on the monitor with a finger and the selected portion is transmitted to the diagnosis and analysis server. The result is then displayed to the user. The application uses Camera API (android.hardware.Camera) and focuses the camera through android.hardware.Camera.AutoFocusCallback. It uses android.app.Activity for the display module and android.hardware.SensorManager for the sensor module. Some other modules such as the speech module, communication module, and cognitive support modules are also used.

Kitano et al. [6] discussed a method used for estimating the distance between objects in photographic images.

This is known as the depth from focus, or DFF method. Although this method has already been implemented through image-side telecentric optical systems because a change in image magnification becomes a problem due to differences in focus positions. Therefore, there have been no examples of this method having been implemented using a general optical system camera, such as a smartphone. In this study, we implemented an image-distance estimation program on the basis of the DFF method using an Android smartphone camera. This paper describes the possibility of estimating the distance between objects in an image taken with the camera of a general smartphone by using the DFF method. This paper also describes how we plan to resolve current flaws with the method that were discovered from this study.

Wachel et al. [7] discussed a simple model for on-sensor phase detection algorithm. In smartphones, focusing is seeking for the image being the best approximation of the captured scene. The proposed autofocusing algorithm is a stochastic optimization type. A scene is in focus when the sensor is in the image plane, where all rays from a single point at the scene converge into a single point A popular approach is to use the sequentially collected images with their variance serving as a focus function, and this is known as contrast-detection auto-focusing which also includes algorithms based on an image histogram or its gradient analysis. Since contrast detection does not require any additional equipment, it is very commonly used in digital cameras. However, a single image does not provide information about either the distance between the sensor and the image plane or the direction toward the sensor that should be shifted in order to attain a focused image. As a result, these algorithms seek it iteratively in the back-and-forth manner, which requires capturing an image in each position. These algorithms are usually derivatives of the stochastic approximation routines and rather slow and not directly applicable in object tracking or video applications. Phase detection algorithms can overcome these deficiencies. In phase detection, a single image is split into two, left- and right-hand side halves, and is achieved with a separate optical path. As a result, this is often used in digital SLRs. If the image is out of focus, then the half-images are shifted with respect to each other. This phase shift maintains information about the distance between the sensor and the image plane and the direction towards the sensor should be moved. Phase detection is faster than contrast detection as a single but split image is enough to determine the correct sensor position. Masking makes it possible to split a single image without additional optical equipment.

2.1.2 Deep Learning

Sanga et al. [8] conducted a study aimed at developing a mobile application for early detection of banana diseases with deep learning by using a dataset of 3000 banana leaves images. They pre-trained the models on Resnet152 and Inceptionv3 Convolution Neural Network architectures with the Resnet152 achieving an accuracy of 99.2 percent and Inceptionv3 an accuracy of 95.41 percent. Inceptionv3 was chosen over Resnet152 for Android as it has lower memory requirements. Once complete, the application that was developed was able to detect the two diseases with a confidence level of 99 percent of the captured leaf area. The researchers employed deep learning models and transfer learning techniques and assessed the performance of the deployed tool to detect banana diseases. They set their result to have

the percentage confidence level in detection to be at least 70 percent as otherwise the app will recommend having a clear image of the captured leaf. The results of this study were that early detection in real time is very important for banana diseases to improve banana yields.

Wang et al. [9] discussed the development of smartphones, laptops, and mobile devices and how they have also furthered the development of artificial intelligence applications on these devices. The information obtained from the cameras, microphones, and sensors of a smartphone such as video, audio, and acceleration can be provided to mobile deep learning applications, or MDLA. Some examples of what mobile deep learning applications provide are malicious software detection, app recommendation, user verification, mobile visual tasks, mobile web browsing optimization, human activity monitoring, medical health monitoring, and other fields. These applications can support distributed machine learning, federated learning, multiple smart IoT applications, and other services that use mobile big data. The main bottleneck of mobile deep learning applications involve storage, calculations, high power and bandwidth consumption, and users' reluctance to download these applications in conjunction with limited resources of mobile devices. The deployment of MDLAs allows for the migration of a large number of centralized applications to the mobile end. For example, instead of a user having to manually record their meal information, they can instead use a smart spoon. Running MDLAs locally without a third party involves reducing resource requirements or optimizing hardware to make it more suitable by either compressing the deep learning model or reusing intermediate computing results or maximizing the rate of utilizing device resources through precise dispatching among multiple deep learning tasks.Another direction is gaining support from background servers by offloading the running of tasks to reduce computing delay.

Sun et al. [10] defined herbal medicine as the practice which includes herbs, herbal materials, herbal preparations, and herbal products. As herbs are basic and are the source and main components of other forms, there must be a focus on quality control of the herbs. For herb recognition, manual and automatic recognition was proposed. Automatic recognition is based on the idea that using herb images and bringing quick candidate categories to help manual recognition find the decisions faster. The researchers proposed a deep learning based network compression algorithm to compress the Deep Neural Network into a smaller one so that it could be used in mobile devices. The three steps of the application are that first, the image processing is operated on each herb image for the preparation of Deep Neural Network computation. Afterwards, a DNN runs on that image to get confidence scores for each category, and lastly, the scores are ranked with top K herb categories to show the results of the recognition. By compressing the algorithm, herb recognition using automatic recognition can be employed in laboratories with limited resources.

Basavaraju et al. [11] discussed learning techniques for mobile Android applications. A smartphone constitutes a sensor carried by humans that can be exploited to provide even more services through data obtained from its cameras, gyroscope, GPS, and accelerometers. Smartphones occupying 52.8 percent of the worldwide market share today are Android devices. There are three machine learning paradigms. They are supervised learning, unsupervised learning,

and reinforcement learning. The researchers in this paper focus on supervised learning, which consists of "algorithms that learn a model from externally supplied instances of known data and known responses to result in a general hypothesis, such that the learned model can be used over new data to predict responses about future instances." In other words, they are performed with examples by being given existing data with examples that have been assigned one or more labels based on their input values and response values. These examples are then used to train the system for the function that it is supposed to learn. Once the machine learns the function, it is then able to work on new data. The two categories of supervised learning are classification, which is used when the response values differentiate between various discrete classes, and regression, which is used when the response values are continuous, such as numerical data. Several applications that implement machine learning were chosen for examination. With WalkSafe, "The contributions of their work are as follows. 1. The app includes intricate design with vehicle detection and pedestrian alert. 2. It incorporates machine learning algorithms on the phone to detect the front views and back views of moving vehicles. 3. It exploits the phone API (Application Programming Interface) to save energy by running the vehicle detection algorithm only during active calls and using mobile sensors such as a back camera to detect vehicles that could be approaching the user." It uses image recognition through a model that was first trained offline, then uploaded for online vehicle recognition. The steps consisted of dataset building using images from different sources to serve as a basis, training for image preprocessing, and feature extraction and classification. However, if there are implementation problems, the image processing technology could drain the battery life. A Fruit detection app uses microphone sensors to estimate the ripeness of watermelons. To detect it, the watermelon has to be thumped in order for the sound to be analyzed and the result to be given. The datasets were trained by thumping acoustic response signals.

Herbst et al. [12] used deep learning to classify images of rapid human immunodeficiency virus, or HIV tests that were acquired in rural South Africa. They used a library of 11,374 images, and deep learning algorithms were trained to classify tests as positive or negative. When the algorithm was deployed as an application, high levels of sensitivity (97.8 percent) and specificity (100 percent) were found, and the number of false positives and false negatives were reduced. Their findings lay the foundations for a new paradigm of deep learning called enabled diagnostics in low and middle income countries. Their diagnostics "have the potential to provide a platform for workforce training, quality assurance, decision support and mobile connectivity to inform disease control strategies, strengthen healthcare system efficiency and improve patient outcomes and outbreak management in emerging infections."

2.1.3 Machine Learning

Ganesan [13] discussed how machine learning and artificial intelligence revolutionizes mobile app development by allowing apps to identify speech, photos, gestures, and translate voices with high accuracy. For example, Google Maps uses machine learning to provide directions and real-time traffic information. Both Apple and Google analyze

the user's wording behavior to recommend and suggest the next word. Facebook and Youtube use machine learning in order to recommend videos and products in addition to people a user may know. Snapchat uses it for their computer vision programs. Machine learning cannot be accomplished in a single step and involves iterative and repeating processes through data exploration, visualization, and experimentation. The steps are to define the machine learning problem, collect, prepare, and enhance the data that is needed, use the data to build the model by choosing the appropriate machine learning type and algorithm, training the model, testing, and evaluating and fine-tuning it, and deploying the model. The four main classifications of machine learning are supervised learning, unsupervised learning, semi supervised learning, and reinforcement learning. Supervised learning uses a labeled dataset with some of the inputs already being mapped to the output. Unsupervised learning is when models are trained with unlabeled datasets and predict the outcome without any human intervention. Semi supervised learning employs labeled and unlabeled datasets. Reinforcement learning learns from the environment in an iterative method. With machine learning, feature extraction is the process of fetching information out of data that can identify the desired result. Classification is when data that has been feature extracted is taken by the algorithm and a formula is created to determine how a new piece of data can be evaluated. Both feature extraction and classification created a trained model, and prediction is taking the trained model, feeding it new data, and seeing how accurately it predicts the expected results.

Sarker et al. [14] discussed how in mobile phones, machine learning algorithms typically find insights or natural patterns in data to make better predictions and decisions in intelligent systems. Deep learning is a part of machine learning that is used to solve complex problems when using a diverse set of data. Natural language processing derives intelligence from unstructured mobile content expressed in a natural language. There are several characteristics of intelligent apps. Action-oriented applications do not wait for the user to make decisions and instead studies their behavior to deliver personalized and actionable results through predictive analytics. Intelligent applications should also be adaptive to each user and the difference in their use. They should generate suggestions and make decisions based on the users' needs and interests, and they should deliver a data-driven output. In addition, being context aware means gathering information about its environment at any time and adapting its behavior. Users should feel the same experience on different platforms through cross-platform operation. Machine learning empowers mobile devices to "learn, explore, and envisage outcomes automatically without user interference." A branch of artificial intelligence is Natural Language Processing, which deals with the interaction between computers and humans using the natural language. "NLP techniques can make it possible for computers to read text, hear speech, interpret it, measure sentiment or to mine opinions, and eventually determine which parts are important in an intelligent system" The ultimate goal if it is to derive intelligence from unstructured data expression in a natural language such as English.

Mcintosh et al. [15] described how machine learning is used to learn functions from data to represent and to classify sensor inputs, multimedia, emails, and calendar events. Machine learning has been appearing more in smartphones through voice recognition, spell checking, word disambiguation, face recognition, translation, spatial reasoning, and

even natural language summarization. However, a major challenge for app developers is that the end-user's device only has a limited battery life which means that computationally intensive tasks can drain the batteries of the smartphone. Since there are not many guidelines for developers to use machine learning on mobile devices while also being concerned about battery consumption, the researchers in this paper combine empirical measurements of different machine learning algorithm implementations with complexity theory to provide recommendations. They were able to conclude that algorithms such as J48, MLP, and SMO generally perform better with regards to energy consumption and accuracy. They also found that consumption of energy was related to algorithmic complexity, and for best results a developer must consider dataset size, number of data attributes, whether the model will require updating, and more.

Kulkarni et al. [16] aimed to evaluate a Pothole Detection System app that uses machine learning on Android. To detect potholes, the application uses an Android device's built in accelerometer to collect the x, y, and z axis accelerations. The pothole detection algorithm runs when the user is driving on the roads and monitors for changes in the acceleration, adding the current time, geographic coordinates and pothole statistics to the event log. By logging the condition of the roads with the locations, they can alert local authorities to fix the road and resolve the complaint.

Torres et al. [17] proposed a biometrics-based machine learning approach that supports user authentication in Android to augment native user authentication mechanisms, making the process more seamless and secure. Our evaluation results show very high rates of success, both for authenticating the legitimate user and also for rejecting the false ones. Finally, we showcase how the proposed solution can be deployed in non-rooted devices.

Loke et al. [18] introduced a sign language converter system using hand gesture recognition feature to recognize the gestures in Indian sign language and convert them to a natural language. It uses Hue, Saturation, Intensity (HSV) color models for hand tracking and segmentation. Supervised training was used to train the neural network for data classification. The developed android application can capture images of hand gestures, and these images are sent to a web hosting server, "from where they are given as input to the neural network in MATLAB for pattern recognition." Following this, the hand gesture is mapped to its natural language equivalent and the converted text is sent back to the device for the user. As the application is easy to use and inexpensive, it can facilitate communication for the deaf, mute, and those who don't understand sign language.

Makhod et al. [19] used machine learning and image processing for egg size classification. The paper proposed an image processing algorithm for classifying eggs by size from an image displayed on an Android device. A coin of known size is used in the image as a reference object. The coin's radius and the egg's dimensions are automatically detected and measured using image processing techniques. Egg sizes are classified based on their features computed from the measured dimensions using a support vector machine (SVM) classifier. The experimental results show the measurement errors in egg dimensions were low at 3.1 percent and the overall accuracy of size classification was 80.4 percent.

Liu et al. [20] created a family doctor app for mobile health service as a result of the improvement of living

standards that have led to the request of people's health increasing paid attention and also increasing demand for hospitalization. This causes hospital overload. The app created uses mobile Internet technology, integrates the idea of service-oriented implementation and is based on Android mobile development technology, in which the function of self-diagnosis is implemented using decision tree classification algorithm. The system will provide online medical treatment and drug information service tools for family members, including consultation with famous doctors, self-diagnosis of symptoms, convenient drug purchase and case record, etc. After the completion of the system, all the functions of the system were tested, and its performance was evaluated. The results of the evaluation were generally in line with expectations. It can run on a variety of operating system platforms and can be flexibly configured and managed.

Chapter 3

Methods

3.1 Languages and Features

The languages that this kidney health application will use are Kotlin, Java, and the extensible markup language, also known as xml. Kotlin is a statically typed programming language that is designed to interoperate with Java. It has type inference and is Google's preferred language for an Android application, with features that make developing applications easier than in Java. For this project, Kotlin is used to program the main features of the application such as the login screen and the main camera function and features, such as manually accessing different parts of the camera to manipulate the focus. The application also makes use of Java code, to work alongside Kotlin and build on other features for the application. Finally, there is the extensible markup language, or xml, which is used to define the UI layout of the application, or user interface. It is used to build a simple and intuitive interface that the user can easily navigate through and find the information needed to complete a kidney health test. Details such as the colors, the layout, the dimensions, and the design of each page is implemented with the extensible markup language.

Development of this application is done through Android Studio, which is the official IDE for Android application development. Development on Android Studio has a great amount of support and assistance, including an emulator that can be used to quickly run and test an application without needing to plug in an Android phone. For the API, the application uses the Camera2API instead of the newer and simpler CameraX as Camera2 is a low level camera package that has deeper controls for more complex uses, like manually manipulating the lens of a camera to change the focus, along with other abilities. However, doing this means that the device specific configurations need to be managed. It provides full access to Google's camera that the original Camera API and CameraX API do not. OpenCV is a computer vision algorithm that will be used for the detection zone of the test strip. The goal is to use the circle detection feature of OpenCV, which requires the hough circle transform to find all of the circles on the test strip, find which ones create an isosceles triangle, and find the distance between the points in the center of the three circles in each corner. The database that will be used to store the user information is SQlite, which is a small, fast, and self-contained SQL database, and is the most used database in the world that comes with Android.

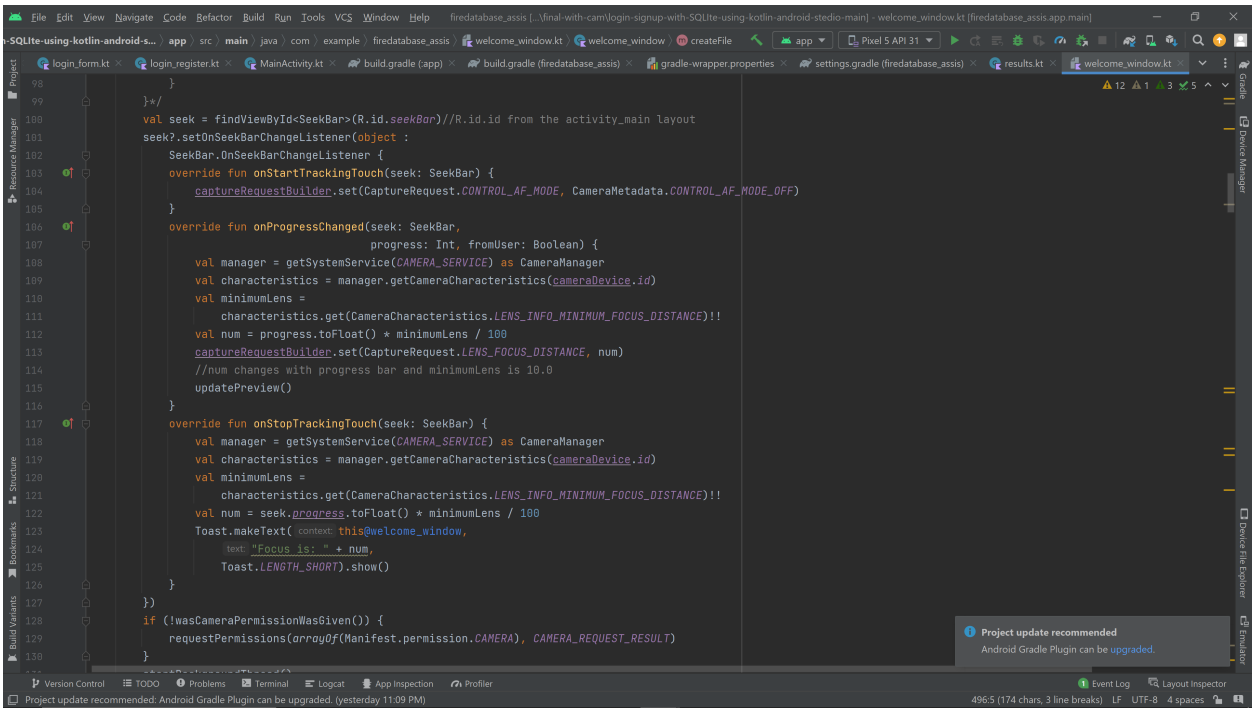


Figure 3.1: Seekbar

3.2 Seekbar

A seekbar is presented on the camera view. To achieve manual focus, The control_af_mode, which is the autofocus mode, is set to off, meaning that the camera will not change the focus by itself when the distance from an object is altered. The code gets the progress of the seekbar that the user sets it to after dragging it and also gets the value of the minimum lens focus distance, which is the shortest distance that the camera can be from the subject and still have it in focus. These two values are then multiplied, and the lens_focus_distance is set to the new value, which then changes the level of focus for the camera.

3.3 Login

For the results and information of the user to be displayed on the readings screen, the user first needs to log in with their username and password. These inputs are then checked in the SQLite database for a matching username and password, and once that is verified, all of the information associated with the account is sent over to the next activity, which is the readings page located in results.kt. Intent is used to navigate from one activity, in this case the login screen, to the next, which is the readings page. Intent.putExtra() transfers data from one activity to the other. In the code shown, on a successful login, the user's name, password, username, demographic, and email are all sent over to be displayed in the readings screen.

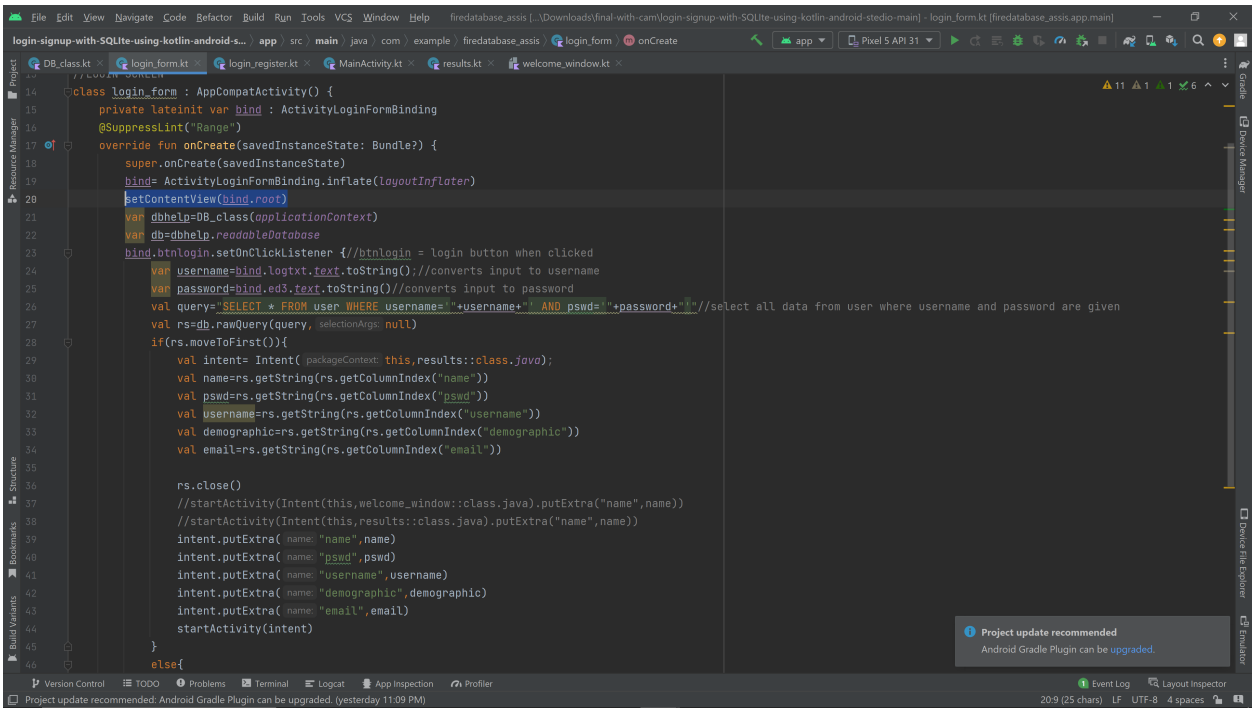
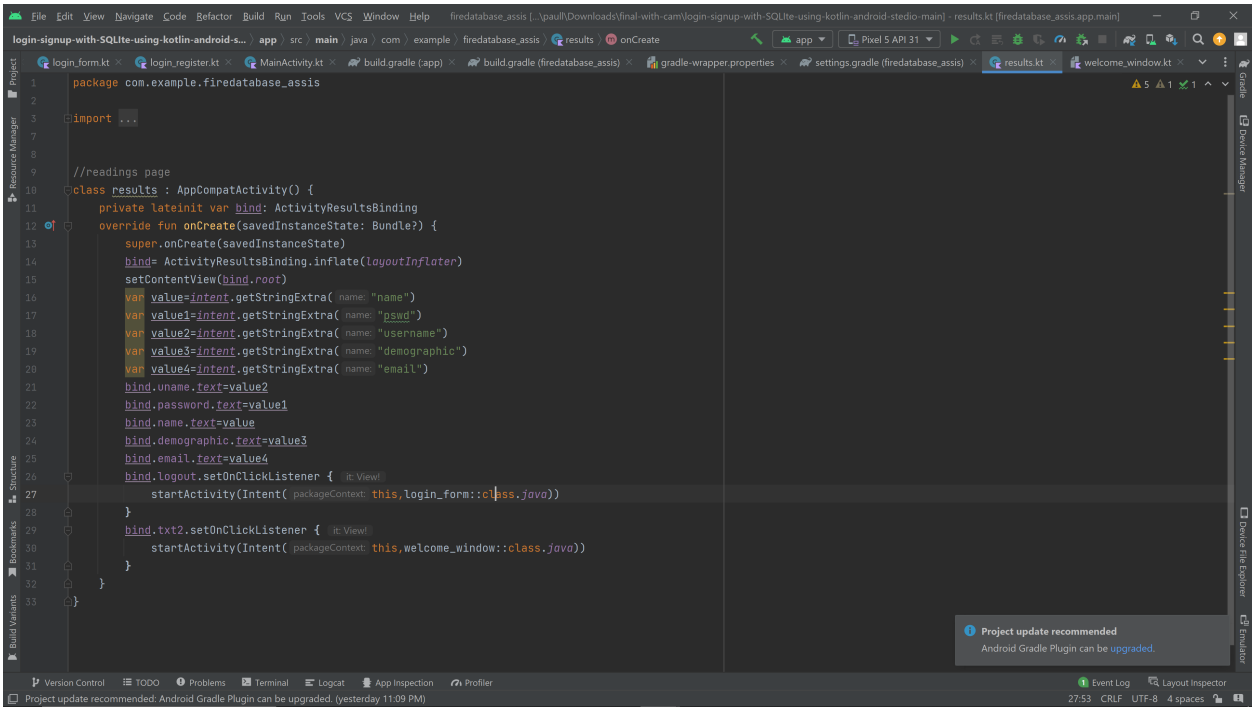


Figure 3.2: Login

3.4 Readings

In the readings screen, Intent.getStringExtra() is used to get all of the associated data from the user and assigns them to different variables from var value to var value4. The variables are bound to each textview in the layout file so that each value can be displayed as text on the readings screen. Figure 3.4 shows how a textview is defined. The android id is used to define which bound value is attached to which textview. In this case, value2 is bound to android:id=”@+id/uname” through bind.uname.text=value2.

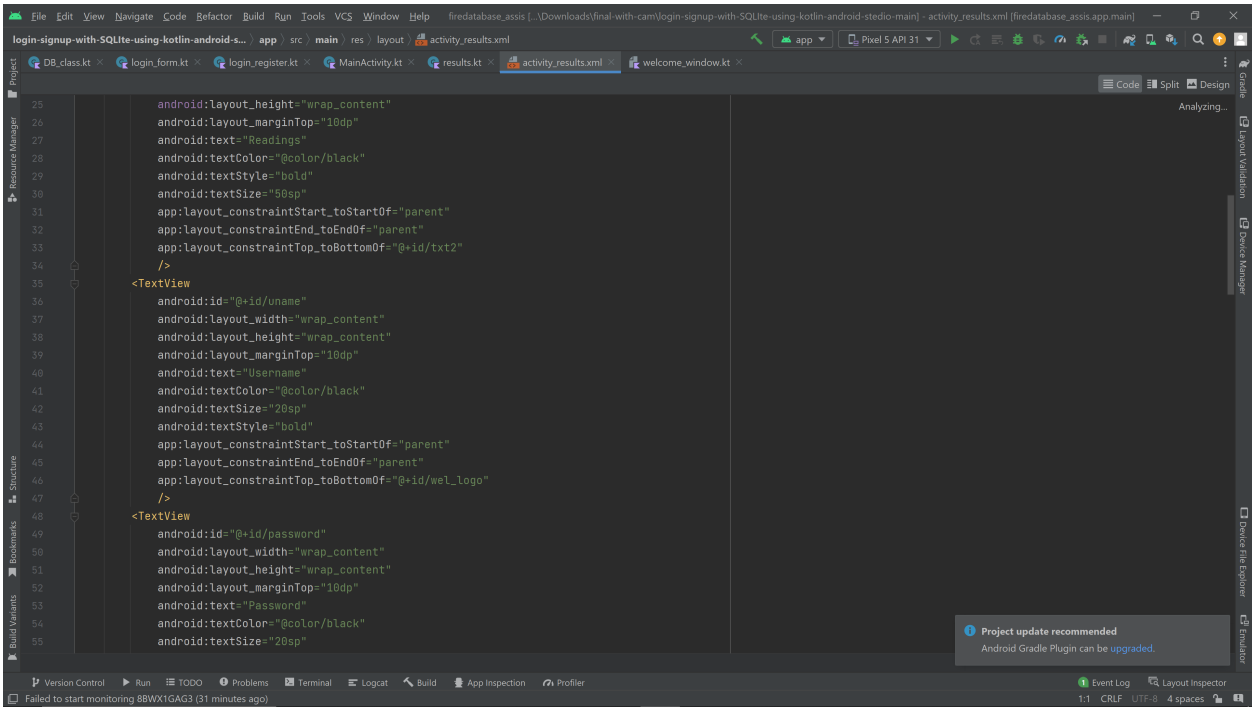


```
package com.example.firedatabase_assis

import ...


//readings page
class results : AppCompatActivity() {
    private lateinit var bind: ActivityResultsBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        bind= ActivityResultsBinding.inflate(layoutInflater)
        setContentView(bind.root)
        var value=intent.getStringExtra( name: "name")
        var value1=intent.getStringExtra( name: "pswd")
        var value2=intent.getStringExtra( name: "username")
        var value3=intent.getStringExtra( name: "demographic")
        var value4=intent.getStringExtra( name: "email")
        bind.uname.text=value2
        bind.password.text=value1
        bind.name.text=value
        bind.demographic.text=value3
        bind.email.text=value4
        bind.logout.setOnClickListener {  it: View!
            startActivity(Intent( packageContext: this,login_form::class.java))
        }
        bind.txt2.setOnClickListener {  it: View!
            startActivity(Intent( packageContext: this,welcome_window::class.java))
        }
    }
}
```

Figure 3.3: Readings



```
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:text="Readings"
            android:textColor="@color/black"
            android:textStyle="bold"
            android:textSize="50sp"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/txt2"
            />
        <TextView
            android:id="@+id/uname"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:text="Username"
            android:textColor="@color/black"
            android:textSize="20sp"
            android:textStyle="bold"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/wel_logo"
            />
        <TextView
            android:id="@+id/password"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:text="Password"
            android:textColor="@color/black"
            android:textSize="20sp"
```

Figure 3.4: Textview Example

Chapter 4

Evaluation

4.1 Design

The user starts at the login screen, where they will be able to login if they already have an account, and if not, they can choose to register instead. In the registration screen, the user inputs their full name, username, password, demographic, email, race, and gender. Afterwards, they can log in, which takes them to the home screen. At the home screen they will see the information associated with their account, and they can also choose to go to the camera to take a new test. In the camera view, the user can manually focus the lens of the camera through the focus distance values to achieve the desired focus, and whatever value the focus is at will be returned to the user to let them know the current value.

4.2 Challenges

A substantial challenge during the development of the application was manual focus, as there was not a lot of information surrounding the feature. The applications that were used as a reference, including the samples that Google provided, all utilized autofocus, which is achieved by simply setting the control_af_mode to on. Some of the other challenges were sending over values to different files of the application such as the name that the user logs into and displaying it in the home page, and that was achieved with packaging the user information into an intent and sending it over to the next activity, or file. A huge hurdle was creating the algorithm to detect and highlight all of the circles on the test strip, find the triangle, and find the detection zone where the algorithm would take place. OpenCV has been a challenge. As this portion of the project was not able to be completed, the alternative that was proposed was to implement an algorithm that would allow the user to take a photo after changing the focus of the camera view and crop the desired detection zone for the final algorithm to analyze the test strip and return the results to the user. Only the pixels in the cropped area would be sent to the machine learning algorithm to be analyzed. This alternative method is easier to implement but requires more work from the user when using the application. Instead of the application automatically finding the detection zone to analyze, the user will need to define the zone themselves.

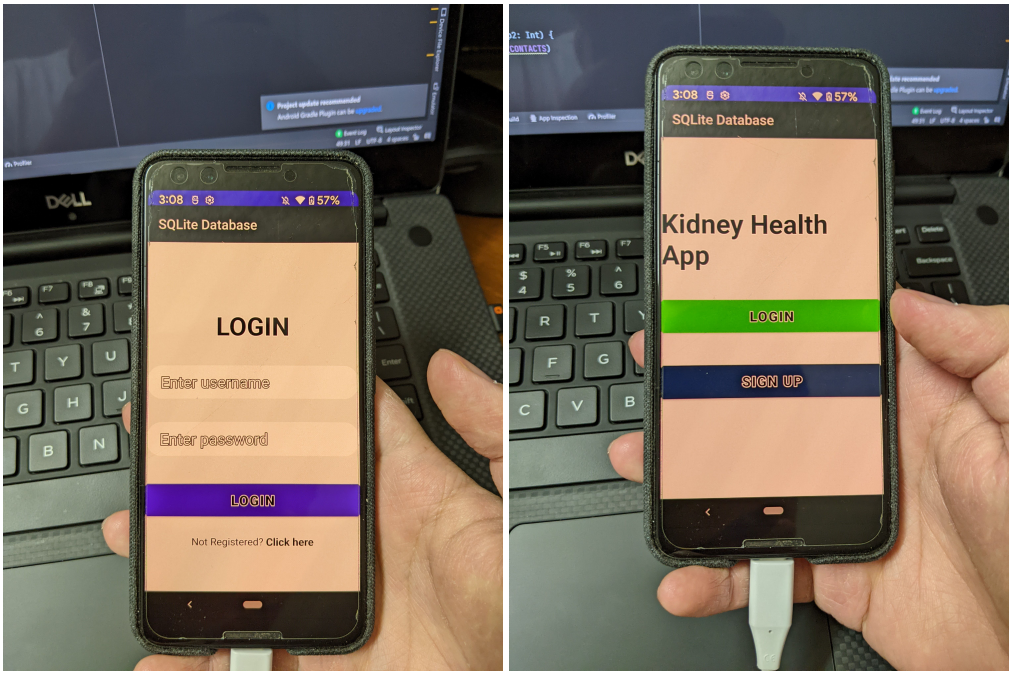


Figure 4.1: Login and Register Screen

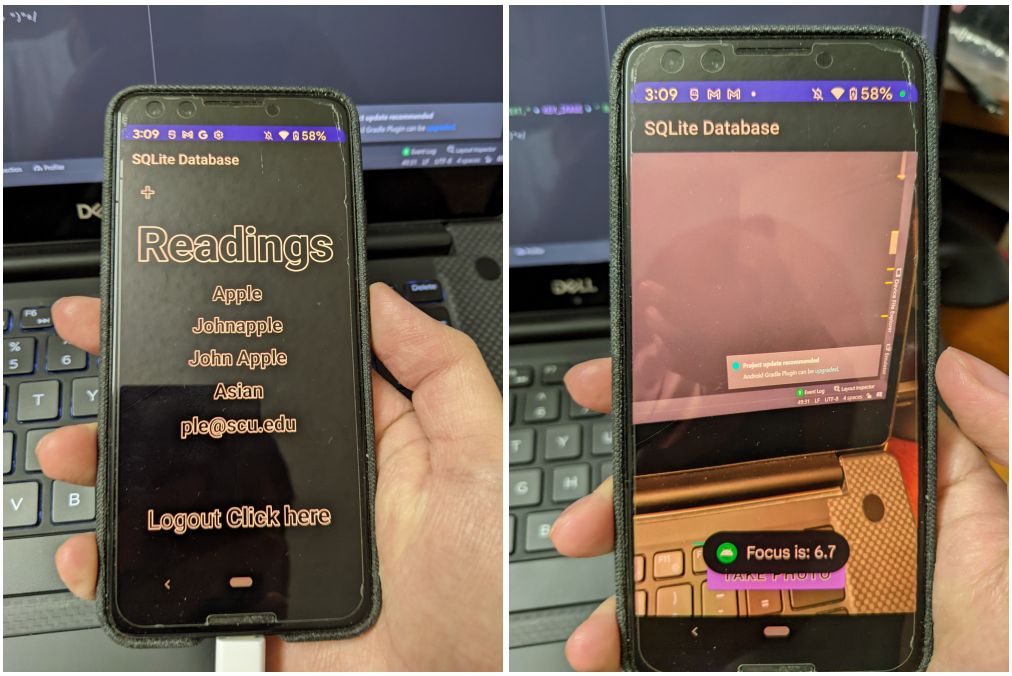


Figure 4.2: Readings Screen and Camera View

Chapter 5

Future Work

In the future, efforts will be directed to developing the detection zone algorithm so that the user will not have to manually crop the image. Other efforts can be made to improve the detection zone algorithm so that it can work even at an angle instead of needing to be at 90 degrees, or perpendicular to the test strip. With this feature implemented, it will make it much easier and convenient for users as they would not have to think about keeping the phone exactly perpendicular to the test strip. It will be able to function at any angle and still be quick and accurate.

Chapter 6

Societal Issues

Ethical Issues. In terms of ethical considerations, the major consideration during the development of this project is privacy, as the application takes a user's private information to run a kidney health test. The application has to ensure that the information will not be leaked as test results should be private. However, SQLite is local to the device, so only the user with the device has the information on hand and can access it. Accessing the database also requires a username and password that the user will need to keep safe. None of the information will be stored on the internet.

Chapter 7

Conclusion

7.1 What We Have Learned

Kidney Health Disease is one of the most common diseases in the world, and although it is easy to detect early on, the current methods of testing are painful, inexpensive, and potentially dangerous. This application will make the process much more simple, efficient, and practical for those who cannot afford to make a trip to the hospital and a lab, and do not want to wait for their results to come in. All of the testing can be done at home in an Android application.

7.2 Why it is Important

The goal of the Kidney Health Android App is to allow for easier access to kidney health testing. It is more comfortable, cheaper for those who do not have money for expensive hospital tests, and is more convenient as it can be done at home with just a blood test strip and an Android smartphone. Kidney disease is very dangerous, common, and difficult to detect in its early stages, and an inexpensive method of testing will encourage people to check the health of their kidney earlier on and with more regularity to get early treatment.

Bibliography

[1] V. Blahnik and O. Schindelbeck, "Smartphone imaging technology and its applications," *Advanced Optical Technologies*, vol. 10, no. 3, pp. 145–232, 2021.

[2] Y. Zhang, L. Liu, W. Gong, H. Yu, W. Wang, C. Zhao, P. Wang, and T. Ueda, "Autofocus system and evaluation methodologies: A literature review," *Sensors and Materials*, vol. 30, pp. 1165–1174, 01 2018.

[3] A. Abuolaim, A. Punnappurath, and M. S. Brown, "Revisiting autofocus for smartphone cameras," in *European Conference on Computer Vision (ECCV)*, pp. 523–537, Springer, 2018.

[4] P. Nguyen, N. T. Le, and Y. M. Jang, "Challenges issues for occ based android camera 2 api," in *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, pp. 669–673, 2017.

[5] H. Nishiyama and F. Mizoguchi, "Cognitive support by smart phone - human judgment on cosmetic skin analysis support -," in *IEEE 10th International Conference on Cognitive Informatics and Cognitive Computing (ICCI-CC'11)*, pp. 175–180, 2011.

[6] K. Kitano and A. Kobayashi, "Implementation of dff using a smartphone camera," in *2017 IEEE 6th Global Conference on Consumer Electronics (GCCE)*, pp. 1–3, 2017.

[7] P. Sliwinski and P. Wachel, "A simple model for on-sensor phase-detection autofocusing algorithm," *Journal of Computer and Communications*, vol. 1, pp. 11–17, 12 2013.

[8] S. Sanga, V. Mero, D. Machuve, and D. Mwanganda, "Mobile-based deep learning models for banana diseases detection," *CoRR*, vol. abs/2004.03718, 2020.

[9] Y. Wang, J. Wang, W. Zhang, Y. Zhan, S. Guo, Q. Zheng, and X. Wang, "A survey on deploying mobile deep learning applications: A systemic and technical perspective," *Digital Communications and Networks*, vol. 8, no. 1, pp. 1–17, 2022.

[10] X. Sun, H. Qian, Y. Xiong, Y. Zhu, Z. Huang, and F. Yang, "Deep learning-enabled mobile application for efficient and robust herb image recognition," *Scientific Reports*, vol. 12, p. 6579, 04 2022.

[11] P. Basavaraju and A. S. Varde, "Supervised learning techniques in mobile device apps for androids," *SIGKDD Explor. Newsl.*, vol. 18, p. 18–29, mar 2017.

[12] V. Turbe, C. Herbst, T. Mngomezulu, S. Meshkinfamfard, N. Dlamini, T. Mhlongo, T. Smit, V. Cherepanova, K. Shimada, J. Budd, N. Arsenov, S. Gray, D. Pillay, A. Herbst, M. Shahmanesh, and R. McKendry, "Deep learning of hiv field-based rapid tests," *Nature Medicine*, vol. 27, pp. 1–6, 07 2021.

[13] V. Ganesan, "Machine learning in mobile applications," *International Journal of Computer Science and Mobile Computing*, vol. 11, pp. 110–118, 02 2022.

[14] I. Sarker, M. Hoque, K. Uddin, and T. Alsanoosy, "Mobile data science and intelligent apps: Concepts, ai-based modeling and research directions," *Mobile Networks and Applications*, vol. 26, 02 2021.

[15] A. Mcintosh, S. Hassan, and A. Hindle, "What can android mobile app developers do about the energy consumption of machine learning?," *Empirical Softw. Engg.*, vol. 24, p. 562–601, apr 2019.

[16] A. Kulkarni, N. Mhalgi, S. Gurnani, and N. Giri, "Pothole detection system using machine learning on android," 2014.

[17] J. Torres, S. de los Santos, E. Alepis, and C. Patsakis, "Behavioral biometric authentication in android unlock patterns through machine learning," in *ICISSP*, 2019.

[18] P. Loke, J. Paranjpe, S. Bhabal, and K. Kanere, "Indian sign language converter system using an android app," in *2017 International conference of Electronics, Communication and Aerospace Technology (ICECA)*, vol. 2, pp. 436–439, 2017.

[19] R. Waranusast, P. Intayod, and D. Makhod, "Egg size classification on android mobile devices using image processing and machine learning," in *2016 Fifth ICT International Student Project Conference (ICT-ISPC)*, pp. 170–173, 2016.

[20] W. Liu, L. Yuhang, P. Lingling, and C. Na, "Design and implementation of family doctor app on android platform," in *2018 9th International Conference on Information Technology in Medicine and Education (ITME)*, pp. 128–132, 2018.