Boston University
Electrical & Computer Engineering
EC464 Capstone Senior Design Project

Final Report

CIR App

Submitted to

Professor Janusz Konrad
8 St. Mary's St
Room 443
(617) 353-1246
jkonrad@bu.edu

Professor Jordana Muroff
(617) 358-4661
jmuroff@bu.edu

Mr. Ozan Tezcan
mtezcan@bu.edu

by

Team 18
CIR App Team

Team Members
Chenjie (Lavi) Zhao lavizhao@bu.edu
Alexander Salmi asalmi@bu.edu
Jiayue (Cindy) Bai jib78@bu.edu
Scott Hom schom@bu.edu
Chin-Hua (Tiffany) Yen chinhuay@bu.edu

Submitted: 4/10/2020

Table of Contents

Executive Summary

The Clutter Image Rating (CIR) Application goal is to help improve the level of care provided by support providers to those who have health conditions with negative affects that manifest themselves as an excessively cluttered home. The core providers, as mentioned by our clients, are clinicians, house professional, and first responders. The level of care is improved by eliminating manual CIR Scoring which is labor intensive and has human bias. Both problems are addressed by automating the process with a deep learning algorithm that automatically rates pictures and outputs a CIR Score. The algorithm is a Residual Network (Res Net) 18 Convolutional Neural Network (CNN) developed by the PhD student Ozan Tezcan [4]. The goal of the CIR App senior design project is to improve the algorithm accuracy by collecting more data to make the training data set more representative of the full range of cluttered rooms on the CIR Scale. Gathering data can be accelerated by allowing many people to use the application. The app is being developed for iOS and Android using the Expo and React Native to put the algorithm in the hands of as many people as possible. The final application is intended to be

released to the public so it needs to be HIPAA compliant, because of the patients data is Private Health Information (PHI). The health information is protected under HIPAA to protect patients confidentiality. The encryption features for part of HIPPA compliances which are TLS encryption for data transfer and FIPS compliant encryption for data storage on AWS [6][7].

In general, the workflow of the app is the following. Choose an image from the camera roll or take a picture to be selected. After the image has been selected users must input a rating. The ratings that the user put in will be used to compare against the scores generated by the algorithm to improve the algorithm. The picture is sent to CIR Algorithm API Endpoint via a POST request in formdata. As well there are two additional features. The first is to view and edit past scores for authorized users. The final feature is to signout.

The storage of user data is in the cloud on a cloud provider which is HIPPA compliant. In terms of expenses all of the cost was taken up by developer accounts to publish the app to the Google Play Store and the Apple App Store. The total cost is $128.66. As well, important resources that did not cost any money were included in the cost (See section 6).

1.0 Introduction

The primary objective of this project is to develop a cross-platform mobile app that will be capable of capturing images of a room and calculating a CIR rating based on the amount of clutter in the captured room. The project cloud service to store images of rooms, their corresponding CIR ratings, as well as other useful data. The CIR rating is calculated in the cloud using a convolutional neural network developed by Ozan Tezcan. The long-term goal for this app is for it to assist medical professionals with mental health assessment and treatment. Due to this, the app must be HIPAA compliant, including encrypting all data being sent to and stored in the database. A secondary objective of this project is to collect data to further train and improve the CIR algorithm.

2.0 Concept Development

At the beginning stage of our design, we decided to split our project into two phases. We did this to both satisfy our clients' need to collect more data to improve their current machine learning algorithm and to have a functional app for patients and clinicians. In the first phase, we will develop a cross-platform app for housing professionals to take and upload pictures of rooms during unit inspection and upload manual CIR scores associated with these images. We can then use these images to improve the accuracy of the algorithm. After we publish our first phase in mid-February, we will start the second phase in which we develop features for patients to get auto-generated CIR scores for self diagnosis and treatment. This also allows clinicians to track their patients' progress and manually overwrite CIR scores. We discussed the possibility of developing the app for patients and clinicians directly as well. However, we all agreed that it is

more beneficial to develop the app for housing professionals first because it will help improve the algorithm before releasing it as a feature and this version of the app has similar features we can reuse later for our final product.

To make a cross-platform app, we discussed two solutions. We can either use a cross-platform development tool or split into two teams for iOS and Android. Although dividing into Android(Java) and iOS(Swift) teams may allow us to develop more finessed features based on specific phone models, it does not allow a smooth experience when incorporating changes and updates and it will divide up labor unnecessarily which can potentially reduce efficiency. We looked into two specific development tools, Expo and Ionic. We eventually chose Expo over Ionic because Expo is more robust, with more testing and publishing options and is very user friendly.

Regarding specific functionalities that our products should possess, we discussed the following requirements for product of each phase:

Our first phase app should have a photo/video taking feature, a photo/video upload feature, and a manual CIR score upload feature. Furthermore, it should allow the user to view images uploaded and edit scores. The app should restrict access to only users trusted and authorized by our clients and provide a personnel training procedure to ensure that the scores uploaded are reliable.

Our second phase app should have all the features of our first app and specific features for patients and clinicians. The app should allow patients to track their scores and . It should allow clinicians to overwrite CIR scores.

For storage of CIR scores and images, we pitched various options including filesystem, SQL databases and noSQL databases. We decided to use Firebase as our database for our preliminary phase app to store image-CIR score pairs and user information. We will continue to explore more HIPAA-compliant database options to replace for our final product, such as Azure and AWS. We decided to use filesystem to store the images taken on the phone temporarily when the app is offline.

To achieve HIPAA compliance, data should be protected both at rest and in transit at a minimum. In our case, we decided to use TLS protocol for data transfer and use a strong encryption standard to encrypt all data on the database. We will use a HIPAA-compliant database as well.

An essential function of the app is to generate automatic CIR scores. We were given a machine learning algorithm implemented in Python with pytorch that uses Resnet 18 as the CNN. We evaluated two solutions to integrate the algorithm to the app. We can host the algorithm on the cloud or on the mobile device.

3.0 System Description

For this project, the main deliverable is the actual cross platform mobile application. The application communicates with the backend system to store all data and image files in the database and file system. To generate the score the mobile application communicates with CIR

Algorithm API Endpoint which is hosted on a Python Flask Server. The general block diagram of the system is shown below in figure 1.

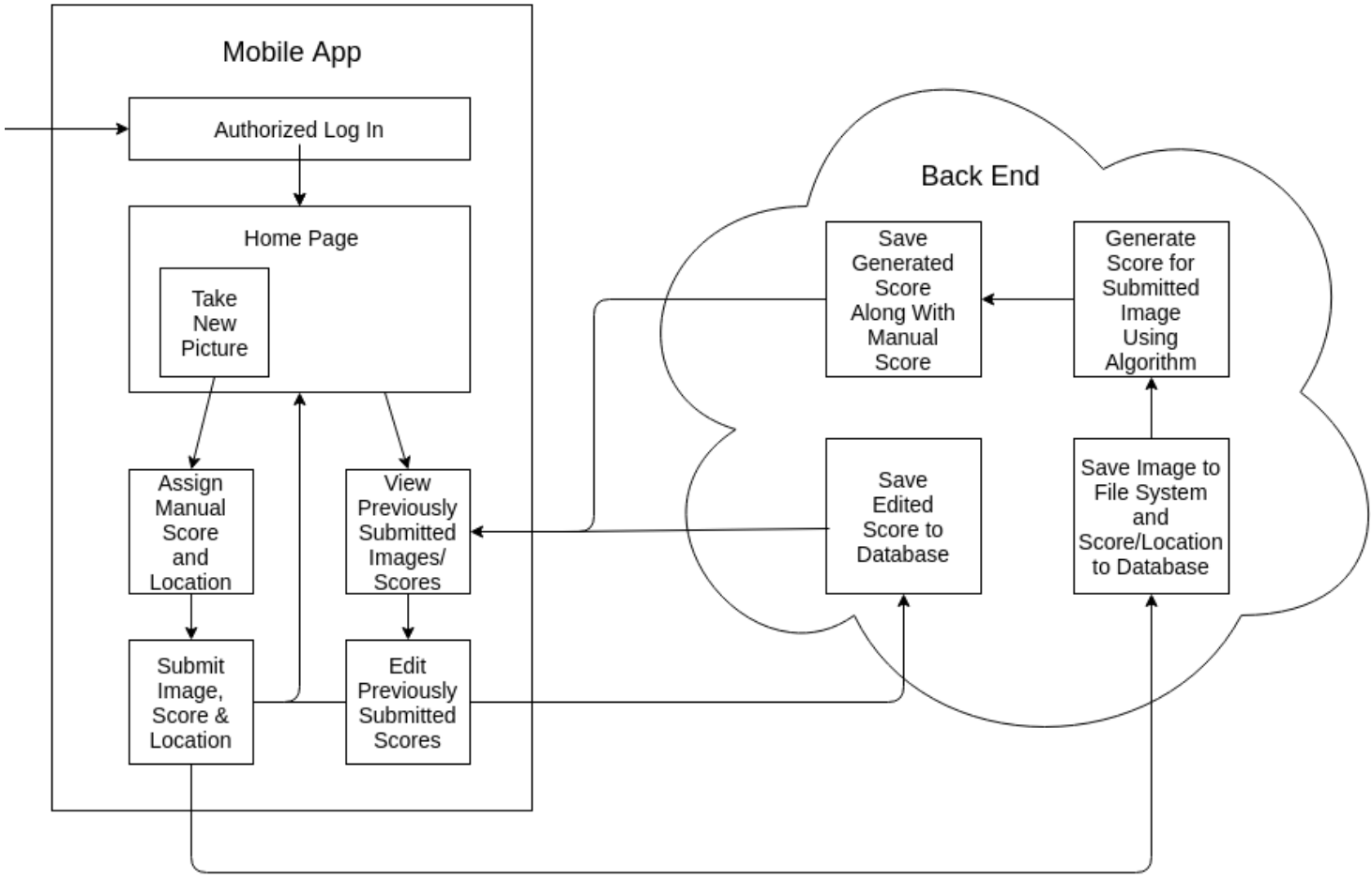


Figure 1. Block diagram of the CIR App system.

The application UX is fairly simple for this project, as there is not much for the app to do. It features an authorized login, where the user's email must be registered in the database before being able to login to the app. This is so that only trusted users may use the app, for security purposes. Once the user has logged in, the app allows the user to take an image, view this image, and then assign a numerical CIR score from 1 to 9, as well as assign it a location/room where the image was taken. This location tag is to be able to keep track of progress made in a specific room over time. The user is then able to submit this image, along with its rating and location, to the back end system, where it will be stored and handled as specified in the next paragraph. Note that, to save the user from extensive roaming costs, if the device that the application is running on is not connected to WiFi, the data and image will be stored locally on the device until it gains a WiFi connection again, and the data will then be sent to the back end. The user is also able to view all of their previously submitted image and rating pairs, as well as edit their assigned

ratings, or delete the image and rating entirely if they so choose to. When editing a rating, this new rating will be stored separately from the original rating, to keep track of changes in the manual ratings. Some screenshots of the current UI for the application can be found in in section 4.5 of this report.

When an image and rating pair is submitted in the application, a couple of tasks will be performed by the back end system. First, the image will be stored in the file system, and the image link, the rating, the location, and the user's id will be stored in the database, along with some other data to be utilized by the application. The database and file system, as well as the SSO authorization system is all being hosted through Google's Firebase system. Second, after the image, manual rating, and location are stored successfully, the image is also sent via POST request to the CIR Algorithm API Endpoint to generate the score. The API works by receiving the POST request on a specific port of the EC2 Server which is ther port the docker container is running on. The request is further routed by the docker container configuration to a specific port on the Flask Server to receive the image and pass it to the deep learning neural network to generate the automatic score. After this automated rating has been calculated, this rating will also be stored in the database along with the image link, the manual rating, and the locationfigure. Also, the score is sent back to the mobile app on the same POST request via promise. See the block diagram of our backend system below in figure 2.

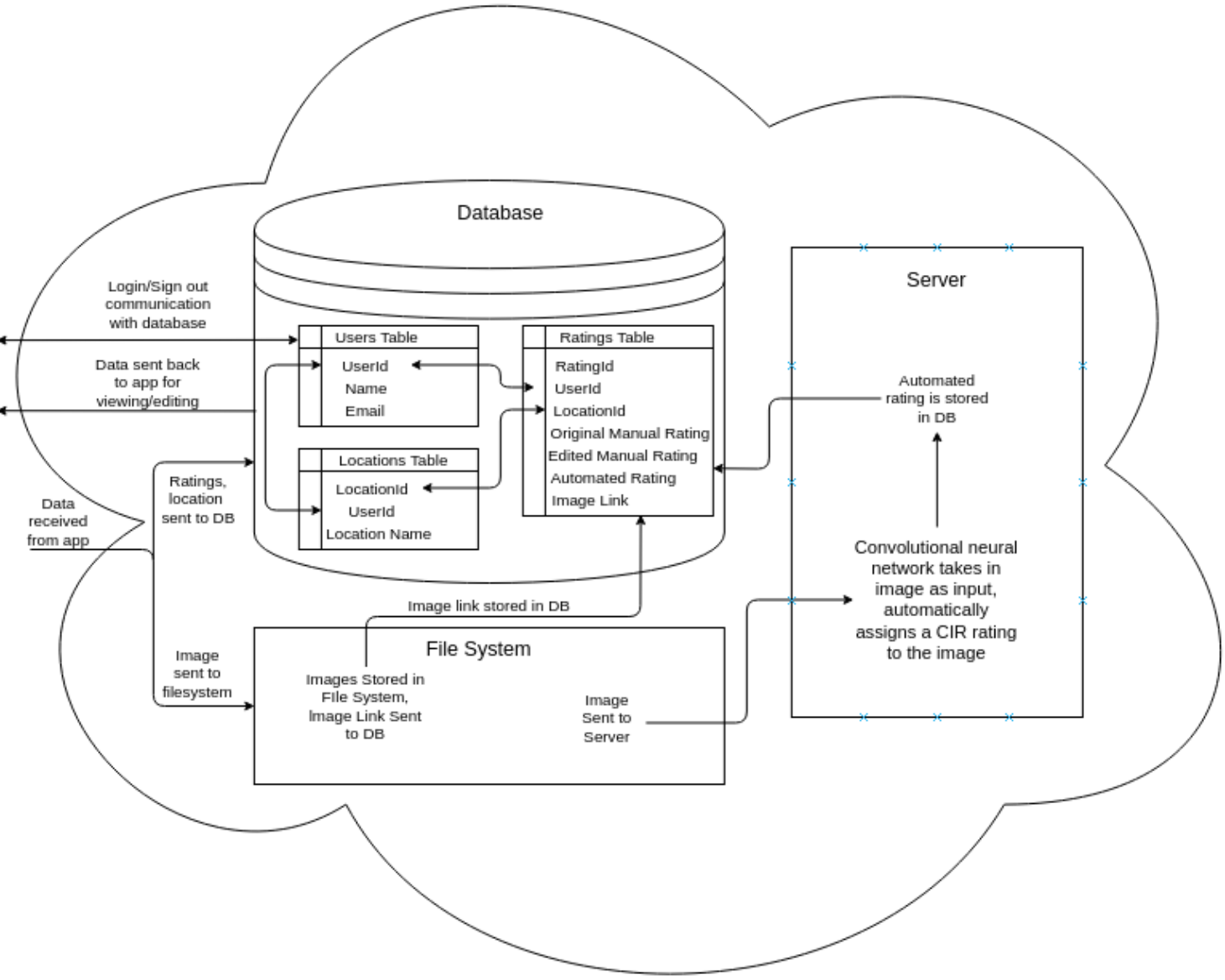


Figure 2. Block diagram of the back end system.

Finally, since this project is intended to eventually be used in medical practices, it involves the challenging task of making everything HIPAA compliant. This includes encrypting all data communication, as well as encrypting the stored data while it is stored in the database and file system. We were planning on working on this as our final sprint of the project, however due to the COVID-19 situation, we were not able to complete this part of the project in time.

4.0 Second Semester Progress

This semester we focused on app development. We worked on front and back end individually

and further did integration and testing. We divided up work into sections to allow us to work individually. Further details will be discussed in different sections as follows.

4.1 Technology and Design

We're using NPM for the package manager. For front end development, the platform for the cross platform mobile application is React Native and Expo, which is a framework and a platform for universal React applications. It is a set of tools and services built around React Native and native platforms that help you develop, build, and quickly iterate on iOS, Android, and web apps from the same JavaScript/TypeScript codebase. Those functionalities are Oauth login, Camera, Rating image to integrate the deep learning algorithm hosted on Node JS server. For storage, we're using google firebase and expo file system. The required materials are listed as follows:

- Package Manager: **NPM**
- Front End Development Tools: **Expo**
    - @expo/vector-icons
    - expo-asset
    - expo-barcode-scanner
    - expo-camera
    - expo-constants
    - expo-face-detector
    - expo-font
    - expo-google-app-auth
    - expo-image-picker
    - expo-media-library
    - expo-permissions
    - expo-google-sign-in
    - expo-app-auth
- Front End Framework: **React Native**
    - lottie-react-native
    - react-native-dialog-input
    - react-native-gesture-handler
    - react-native-maps
    - react-native-paper
    - react-native-reanimated
    - react-native-screens
    - react-native-svg
    - react-native-vector-icons
    - react-native-view-shot
    - react-native-web
    - react-navigation
    - react-navigation-stack
    - react-native-is-iphonex
- Storage
    - **Firebase**

    - expo-file-system
- Cloud Computation
    - AWS Lambda Function
    - Google Cloud Function
- Android Device/Android Studio

4.2 Set Up

In order to set up the project on expo, you will need to do the following:
For Expo Client:

1. Create an expo account: https://expo.io/signup
2. Get the expo app on your phone (on App Store or Google Play)
3. Download node and npm: https://nodejs.org/en/
4. Install Expo CLI: $ npm install expo-cli --global
5. Install React Native CLI: $ npm install -g react-native-cli
6. In the root directory of the project run $ npm install
7. Run application: $ expo start
8. Use your phone to scan the QR code on your computer screen
9. You will be able to use the application on your phone through the app expo.

For Android:

1. Run expo build:android to get the Android version of the binary
2. Run expo publish to push the latest updates for the app over-the-air if the updates don't involve configurations
3. Get an Android device or set up a virtual device on Android Studio(Camera function will not work on a virtual device)
4. Download the binary from the link we send out
5. Open the binary and use the app on your Android device
6. New updates will be pushed to the app automatically after killing the app in the background and restart the app

For iOS:

1. Run expo build:ios to get the iOS version of the binary
2. Run expo publish to push the latest updates for the app over-the-air if the updates don't involve configurations

4.3 API Usage

AWS:

1. Login into AWS Console here (see link below)
   https://console.aws.amazon.com/
2. Create an AWS EC2 Linux Instance with Ubuntu.
3. Download .pem file and save it to somewhere safe to ssh into the server later.
4. SSH into AWS EC2 Instance with this command and fill the appropriate fields (see command below). The default username is ubuntu and MUST be used to ssh.

   $ ssh -i <path/path/*.pem> <username>@<publicDNSofEC2Instance>

5. Install docker with these commands (see commands bellow)

$ sudo apt install docker.io
$ sudo systemctl start docker
$ sudo systemctl enable docker

6. Confirm Docker installation with this command (see command bellow) that prints the Docker version.

   $ docker --version

7. Download docker image with the algorithm with this command (see command below).

   $ docker pull ninja808/cir:latest

8. Get Docker imageName
   $ docker images
9. Run Docker with this command (see command below)

   $ docker run -it –d --name <imageName>:<tagName> -p 5001:5001 -v <hostDir>:/home/shared <imageID> bash

   - --name: <imageName>:<tagName>
   - -d: run in detached mode
   - -p: port
   - –v: link AWS Instance directory to Docker directory

10. Run Flask Server with these commands (see commands below)
    $ cd /home/shared
    $ python inference.py

4.4 Work Distribution

1. Scott Hom: Scott takes roles on the binary generation and the algorithm API deployment. For the binary generation, he tested different methods on Windows and Mac that he concluded the only way is we do it through the expo CLI with a developer account. In terms of the Algorithm API, it is implemented as a Flask Server inside of a Docker Container inside of an EC2 Instance.
2. Jiayue (Cindy) Bai: Cindy focuses on the camera function. Last semester our clients mentioned that due to privacy issues, we cannot save the pictures taken in the app to local camera roll, so Cindy changed that and handled the local database. She focuses on changing and debugging the frontend design. She also worked on debugging the image uploading and score saving features of the app as the images and scores are not uploading and saving to the database as intended after new UI updates.
3. Chenjie (Lavi) Zhao: Lavi takes responsibility for making the login more secure and also for making it to binary. She looked into expo and react native to see how to deploy our app via Firebase. She has been working on making the machine learning model into a

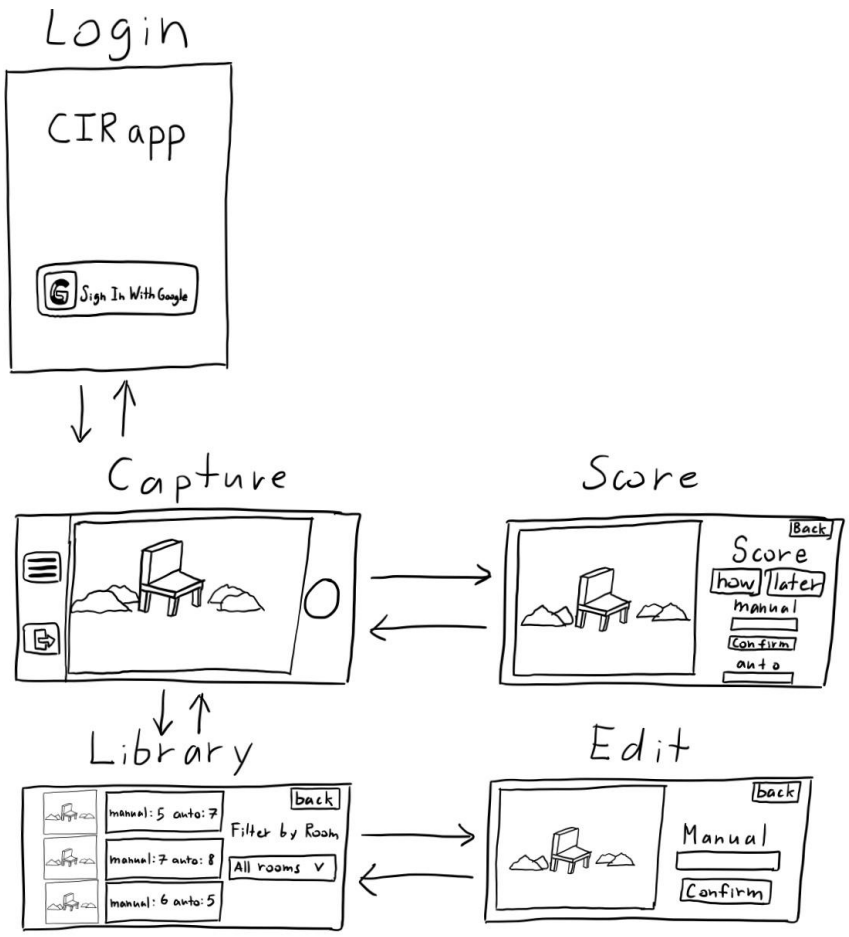
firebase function and incorporating security features. She found three ways to secure the database. She started by implementing security rules in Firebase and investigating Google KMS tools to encrypt data saved in the database.

4. Alexander Salmi: Alex takes care of the backend database part. According to our clients, we are required to categorize the pictures not only by the CIR rating but also by room and location. Alex updated the database structure to include the rooms and locations and also updated the UI for the records and submit pages to include choosing rooms. Alex manages the database and has been working with the frontend team to debug the image uploading and score saving features.
5. Chin-Hua (Tiffany) Yen: Tiffany is in charge of the front end UI for the entire app, making the app more user friendly and intuitive. Our clients requested to put all functions on the same page (home page) last semester, but this semester they wanted us to change it. Therefore, she changed the layout and appearance of the app several times to satisfy client's requests over time. This includes converting the design from vertical orientation to horizontal orientation, changing the home page to camera page, and redesigning score, library, and edit pages.

4.5 Progress Measurement

1. Meeting with clients (Professor Konrad, Professor Muroff, and PhD student Ozan): We met with our clients at least once every two weeks to report our progress as a team. In addition, we sent them the updated app for them to play with and give feedback on.

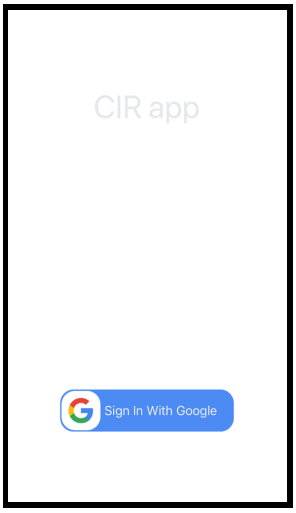
2. Team meeting: We held our team meetings twice every week to work on the project and discuss our progress.
3. App Development: We have a functional and tested app that works on both iOS and Android platforms. The frontend design now met our clients satisfaction. The UI design is converted to horizontal display and the navigation flow is changed to increase the efficiency for the work of housing professionals as shown in the diagram below.



We have a functional algorithm implemented but not yet integrated with the rest of the application. We acquired developer accounts needed for our next steps.

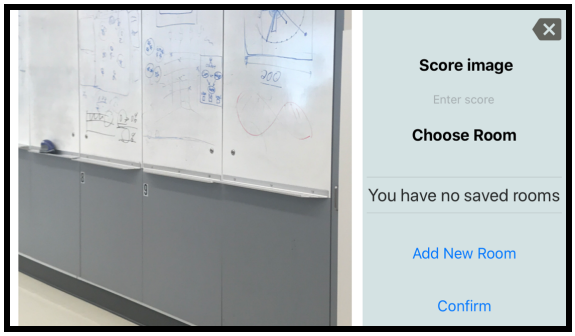
4.6 Result Visualization (Testing steps)

1. Login Screen: When you open the app, you will be able to sign in through google if your email is added to the firebase.

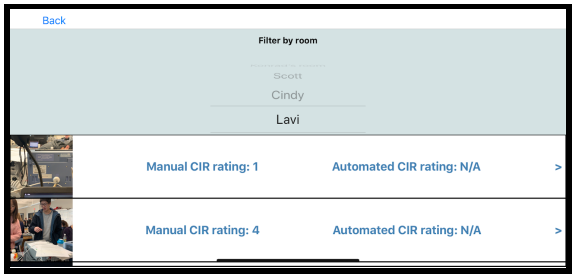


2. Camera Screen: Next, based on our client's request, the home page will be the camera page (landscape mode) where you can take a picture with the button on the right and view records and sign out with the buttons on the left.

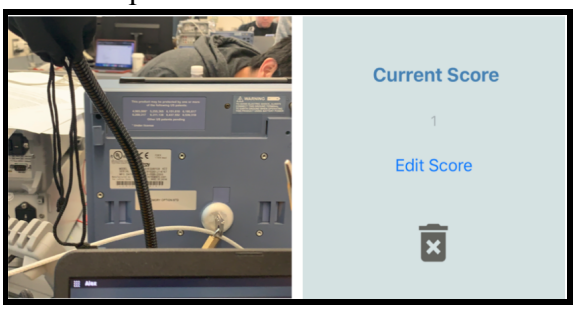
3. Score Screen: After you take a photo, you may manually enter a score and choose a room that you added before or add a new room, and press confirm.



4. Library (Records) Screen: On the records page directed from the home page, you can choose room by scrolling and view your previous pictures and scores.



5. Edit Screen: By clicking on any records on the records screen, you can edit the score or delete the picture.



4.7 Cost

	time	cost
Apple Developer Account (Annual)	1 month	$103.66
Google Developer Account (Lifetime)	1 month	$25.00
AWS T2 Micro Instance	2 month	$0.00
Firebase Database	6 months	$0.00
Firebase Storage	6 months	$0.00
total cost		$128.66

Figure 1: This is the total cost to develop the CIR Application. The time column is included to illustrate how long resources were used which is a significant metric for cloud cost. As well important resources that did not cost any money were included.

4.8 Project Completion Plan

- Security:
    - Implement TLS encryption for HTTP POST requests using Expo React Native Library to comply with HIPAA.
    - Encrypt data in database on Google Firebase and Filesystem on AWS S3 Bucket.
    - Setup API Gateway to manage HTTP requests to CIR Algorithm API Endpoint. Also, implement security measures to prevent cyber security threats such as DDOS. Since the algorithm is already hosted on AWS for security the web firewall named AWS WAF can be used.
    - Remove all metadata except filename and date and time from pictures via Expo React Native Library to ensure no Protected Health Information (PHI) is exposed.
- Networking:
    - Detect when the application is off wifi and store data locally and when connected back to wifi upload the cached information.
- UI:
    - Create an onboarding process for users to signup as general or admin users. For the admin user a onboarding system needs to be created to verify clinicians and housing professionals based upon an official documentation. However, since processing official documents is difficult it would be good to try find an outside company to handle this task.
    - Create a general user mode and admin user mode. The access privileges of the user are stored in the database. The general user mode will not have the ability to edit score vs. admin users will have the ability to edit scores.

- API:
    - Fix the bug that stops the score variable from being updated with the CIR Score, returned as a promise from the POST request. The problem is related to the synchronous function name setState being called inside the

5 Appendix A References

[1] A. Samborski and J. Muroff, "Development and testing of the clutter image rating application," Oct. 17, 2014, undergraduate Research Opportunities Program (UROP) Symposium, Boston University.

[2] A. Tooke, J. Konrad, and J. Muroff, "Towards automatic assessment of compulsive hoarding from images," in *Proc. IEEE Int. Conf. Image Processing*, Sep. 2016, pp. 1324–1328.

[3] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, vol. 1, 2005, pp. 886–893.

[4] M. Ozan Tezcan, J. Konrad, and J. Muroff, "Automatic Assessment of Hoarding Clutter from Images Using Convolutional Neural Networks," 2018, Boston University.

[5] "Breach Notification Guidance." *HHS.gov*, US Department of Health and Human Services, 26 July 2013, www.hhs.gov/hipaa/for-professionals/breach-notification/guidance/index.html.

[6] "NIST.FIPS.140-2.Pdf." U.S. Department of Commerce, 2001.

[7] "TLS Guidelines: NIST Publishes SP 800-52 Revision 2." *NIST*, National Institute of Standards and Technology, 29 Aug. 2019, csrc.nist.gov/News/2019/nist-publishes-sp-800-52-revision-2.

[8] "Guide to Storage Encryption Technologies for End User Devices." *NIST*, National Institute of Standards and Technology, 15 Nov. 2007, csrc.nist.gov/publications/detail/sp/800-111/final.

6 Appendix B Usage

For usage information go to the github page here (see link bellow)

https://github.com/BostonUniversitySeniorDesign/20-18-CIRapp/