Facial Recognition Enabled Social Networking Application

Gautham Vijayaraj[a*], K Mohanakrishnan[a**]

[a]*Department of Computer Science and Engineering SRM Institute of Science and Technology, Potheri, Kattankulathur, Chengalpattu,Tamil Nadu 603203*

Abstract

As we all know most of our social lives are controlled by Social Networking sites. Social Networking refers to the grouping of individuals and organizations together by the help of some medium, in order to share thoughts, interests, activities, etc. There are several web based social network service that are available for example Facebook, Twitter, LinkedIn, Google+ ,etc. all of which offer easy to very use and interactive interface to connect with people with in the country an overseas as well. We can also find several mobile based social networking services in for of apps such as WhatsApp, hike, Line etc. But most of these sites are vulnerable to our personal information being exploited. Also, a lot of cyber criminals use these media as their main base to exploit data. So, what if we could make a platform that not only makes our accounts and data more secure but also makes it a safer environment? That's what our project is going to focus on mainly. And hence, we would like to propose our application that contains all the features that any social networking site contains along with certain additional features which will be making a difference and a huge impact to the society.

1. Introduction

In the current day, many applications and web pages depend upon web services to readily and easily exchange information with one another. Web services provide a way to transfer many different kind of data over the network for the reuse of organizations/individuals across services. We can find that there is a dynamic relationship between the user and the web services, as there are many scenarios that are user input controlled. Web services protocols include Simple Object Access Protocol (SOAP), XML-RPC, and JSON-RPC for message communication; Web Service Description Language (WSDL) and Universal Directory and Discovery Integration (UDDI) for web service description and discovery [1]. For this is the outcome of our continuous yearning for communication across the globe made easier and faster. However just as everything has its own disadvantages even these platforms do to (i.e. the lack of security). For example most web service attacks are XML Injection, XPath Injection, SQL Injection, Spoofing, Denial of Service and Man in the Middle attack [1]. So with this in mind we are going to work on making our own social media application more secure and trustworthy mode of online communication. Here our objective has divided into the main three ideals. The first objective is to provide for a messaging service that is more secure by the help of a custom symmetric key cryptographic algorithm. The second objective is to build up on this service by the implementation of facial recognition for sign-in application of the web application. The third and final objective is to provide for secure and yet overall user friendly environment for the messaging web application.

2. Evaluation of Security Performance for existing social media

This section of the paper contains the security performance for the following topics/scenarios:

- Anatomy of SOAP Security Performance[2]
- Social media security [3]

2.1 Anatomy of SOAP Security Performance

SOAP security activities consist of the following essential two categories such as, cryptography operations and XML processing. Cryptography operations are all byte based, structure neutral and computation intensive. Under this category, we are interested in the performance and differences in signing and verification, in encryption and decryption, and among different algorithms. Though all of the implemented cryptography algorithms are studied

well, only data will provide a chance to confirm with the well known algorithmic properties. Web Services (WS)-Security utilizes existing XML Digital Signature and XML Digital Encryption to specify how to apply these XML general technologies to the SOAP. The system as to how to attach security tokens and a set of processing rules are also included within the specification. We measure WS-Security by measuring time taken for document signing and encryption. WS-Secure Conversation is based on the idea of the conversation sessions, and conversation contexts. It specifies how Security Context Token can be accepted at the beginning and can be used during the session by communication parties. In a scenario where multiple messages are exchanged, use of Security Context Token will help to save sending security keys over and over again. Security Context Token is stored in both side's memory and referenced by its uniquely generated ID. But WS-Secure Conversation recommends not use the Security Context Token for the entire session but to derive a series of keys from the initial token and use these keys in each round of message exchange. [2]

2.2 Social media security and privacy

As a form of presentation and typical application of Web2.0 technology, social media has been constructed by the relations between users and bodies on the basis of modern Internet technology and platform carriers. The original problems related to the confidentiality, completeness, and availability of Internet platforms still exist under social media platforms. With regard to the new platforms, new scenes, and new applications of social media ecosystems, practical application problems have been derived so that are related to security control mechanism, individual privacy protection, and digital copyright protection.[3]Some of the most prominent attacks on social media are as following:

- Identity theft[3]
- Spam attack[3]
- Malware attacks[3]
- Sybil attacks[3]
- Social phishing[3]
- Impersonation[3]
- Hijacking[3]
- Fake requests[3]
- Image retrieval and analysis[3]

Our main intention here is to stop/reduce the attacks based on the messaging and authentication (privacy). Most the above mentioned are just an elaborate study of the various attacks that are prominent in today's social media web applications.

3. Face Authentication

3.1. Related works

In the recent years face detection has gained considerable amount of attention from researchers in biometrics, pattern recognition, and computer vision groups. There is countless security measures, and forensic applications requiring the use of face recognition technologies. As you can see, face detection system is very important in our day to day life. Among the entire sorts of biometric, face detection and recognition system is the most accurate one. In [4], Ashu Kumar et al[4] have presented a survey of face detection techniques. We can see that face detection techniques are increasingly used in real-world applications and products. The most straightforward future direction is to further improve the face detection in presence of some problems like face occlusion and non-uniform illumination. Current research focuses in field of face detection and recognition is the detection of faces in presence of occlusion and non-uniform illumination. A lot of work has been done in face detection, but not in presence of problem of presence of occlusion and non-uniform illumination. If it happens, it will be a big help to face recognition, face expression

recognition etc. Currently many companies are already providing facial biometric in mobile phone for purpose of access. As it has a wide range of application such as for payments, security, healthcare, advertising, criminal identification etc.[4]

We have settled on the use of an API to make the face authentication process efficient and at the same time more accurate. We can see this explained as shown in [9]. There is a strong diminishing return effect and any progress now requires a substantial effort to reduce the number of errors of state-of-the-art methods. DeepFace couples large feedforward-based models with fine 3D alignment. Regarding the importance of each component: 1) Without frontalization: when using only the 2D alignment, the obtained accuracy is "only" 94.3%. Without alignment at all, i.e., using the center crop of face detection, the accuracy is 87.9% as parts of the facial region may fall out of the crop. 2) Without learning: when using frontalization only, and a naive LBP/SVM combination, the accuracy is 91.4% which is already notable given the simplicity of such a classifier. All the LFW images are processed in the same pipeline that was used to train on the SFC dataset, denoted as DeepFace-single. To evaluate the discriminative capability of the face representation in isolation, we follow the unsupervised setting to directly compare the inner product of a pair of normalized features. Quite remarkably, this achieves a mean accuracy of 95.92% which is almost on par with the best performance to date, achieved by supervised transfer learning. An ideal face classifier would recognize faces in accuracy that is only matched by humans. The underlying face descriptor would need to be invariant to pose, illumination, expression, and image quality. It should also be general, in the sense that it could be applied to various populations with little modifications, if any at all. In addition, short descriptors are preferable, and if possible, sparse features. Certainly, rapid computation time is also a concern. We believe that this work, which departs from the recent trend of using more features and employing a more powerful metric learning technique, has addressed this challenge, closing the vast majority of this performance gap. The work demonstrates that coupling a 3D model-based alignment with large capacity feedforward models can effectively learn from many examples to overcome the drawbacks and limitations of previous methods. The ability to present a marked improvement in face recognition, attests to the potential of such coupling to become significant in other vision domains as well. Another major advantage is the already pre-existing database of faces this makes our work as developers easier at same time does not allow any compromise when it comes to accuracy[9].

On of our main basis of belief that a web application where face authentication can be a viable option can be aided with the help of [8]. The system in this paper ([8]) was designed to combine the computation which is run on mobile device and the advantage of cloud computing. The computation on mobile device (onloading) will perform the face detection module and augmented reality to interact with the user. The other computation will run in the cloud server (offloading) using Google App Engine (GAE) dan Face.com service. Actually, the system in this paper was design in three main modules including face detection on Android mobile device, face recognition on cloud server using GAE and Face.com API, and augmented reality as a result of a face recognition module on mobile device. [8]

There are three main underlying modules in this project:

- Face Detection
- Face Recognition
- Augmented Reality

Using the above three we see the result in [8], where authentication is achieved. Thus with the help of [4],[8] and [9] we have come to the conclusion that face authentication using an assisted API. We are using firebase public API system to for Face Authentication. This is the best option as it is a REST API, since it is of low complexity, lightweight payload, fast in terms of performance and execution speed. This helps our application be fast and efficient and reliable at the same time.

3.1. Proposed Method.

The Facial Recognition method we are going to use is FaceNet. FaceNet is a unified system for facial recognition that is based on learning a Euclidean embedding per image using a CNN that consist of 22 layers. The CNN used by FaceNet has been trained to directly optimize the embedding itself in order to replace the intermediate bottleneck layer that has been used by previous deep learning approaches (Schroff, Kalenichenko & Philbin 2015). The authors of FaceNet strongly recommends using their model which has been pre-trained on the VGGFace22 dataset which consists of 3.3 million images for the best results[11].From [11] it was analyzed that FaceNet showed the best accuracy results on a comparison with the other methods.

3.2. Why FaceNet?

From [11] we find that FaceNet achieves the best possible precision of 100%. The second best technique is the scikit-learn implementation of Fisherface using KNN once again with a precision of 70.44%. These results show that FaceNet outperforms all other techniques with at least 29 percent points regarding this metric as well. Precision is calculated by dividing the TPs with all positive predictions, whether correct or not, which mathematically is: $TP/(TP+FP)$.

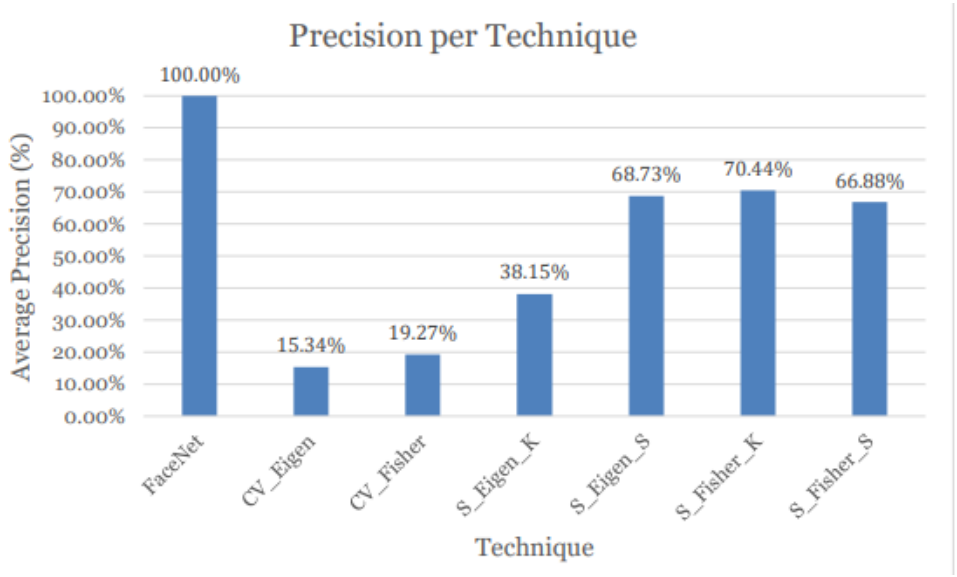


Figure 6 from [11]

Training time is the time it takes to train the model for a specific technique on the 400 training images in the dataset.The average training time taken by the FaceNet method was concluded to be 0.24 seconds

3.3. Implementation of FaceNet using Flutter and TensorFlow

It works with two computer vision models working together, the Firebase ML vision model to perform the face detection and preprocessing in the image, and the MobileFaceNet model to process, classify and transform into a data structure ‘savable’ by a database (an array of numbers).

The process summary is as follows:

Sign up

1. The user takes a photo.
2. The ML models process it and create an output (array of numbers) to be stored in a database.
3. A name and a password are requested

Sign in

1. The user takes a photo.
2. The ML models process it and create an output.
3. The output will be compared against the outputs already stored in the database (it compares by proximity the closest one it finds). As condition, the proximity has to be under the *threshold* (minimum distance), if overcomes it, it will process it as a non-existent user.
4. If the user exists (face already processed) it requests the password for that user, validates and authenticates it.[13]

What is Tensorflow?

TensorFlow is a platform for building and training neural networks, which allow detecting and deciphering patterns and correlations, analogous to the learning and reasoning used by humans.
TensorFlow's flexible architecture enables developers to deploy computation to one or more CPUs or GPUs on desktops, servers, or mobile devices with a single API. It was originally developed by researchers and engineers working in the Google Brain Team, within the Machine Intelligence research department, for the purpose of conducting machine learning and deep neural network research.

TensorFlow Lite is the official framework for running TensorFlow model inference on edge devices. It runs on more than 4 billion active devices globally, on various platforms, including Android, iOS, and Linux-based IoT devices, and on bare metal microcontrollers. [12]

How does it work?

The TensorFlow Lite team provides a set of pre-trained models that solve a variety of machine learning problems. These models have been converted to work with TensorFlow Lite and are ready to use in your applications. TensorFlow Lite is designed to run models efficiently on mobile and other embedded devices with limited memory and compute resources.[12] Part of this efficiency comes from using a special format to store models. TensorFlow models must be converted to this format before TensorFlow Lite can use them. Inference is the process of running data through a model to obtain recognitions. It requires a model, an interpreter, and input data.

Converting Facenet (.pb) to Facenet (.tflite)

We will quantize pre-trained Facenet model having 512 embedding size, but you can choice to use the model with 128 size embeddings. This model is about 95MB in size before quantization. The software requirements for this process are just Python (atleast version *3.4).*After that we need to install the tensor flow package using the pip install command. The Facenet model can be converted from (.pb) to (.tflite) file by following the steps from [12]

How is the .tflite file implemeneted in flutter?

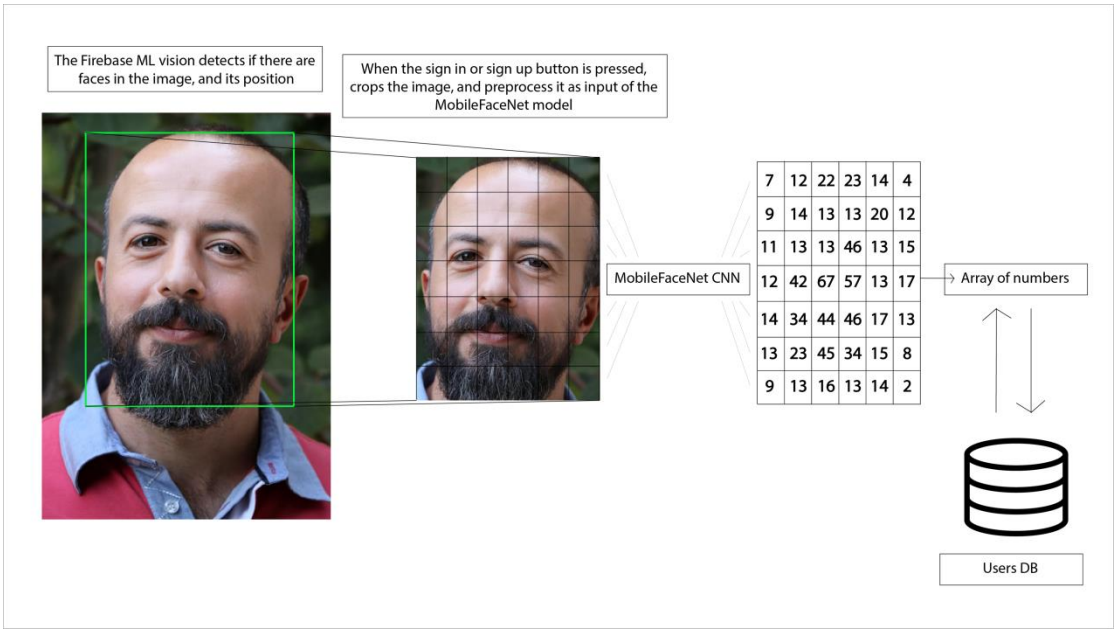
With ML Kit's Face Detection API, you can detect faces in an image, identify key facial features, and obtain the contours of detected faces. It works very well preprocessing the image to detect the zone to be cropped and then processed by the MobileFaceNet model. [14]

The **ML vision** model detects existing human faces in the frames that the camera preview, the class Face contains the coordinates of the points that make up the frame around the face.

```
List<Face> faces = await _mlVisionService.getFacesFromImage(image);
```

This command detects a list of images from the picture taken from the camera of our mobile.

The detected face of the last frame is captured, cropped and then pre-processed to be processed by the MobileFaceNet model. The MobileFaceNet model returns an output (array of numbers).If it's the sign up process , the application requests a name and a password, and later saves the three data (name, password and ML output). If it's the Sign In process, the application performs a search in the database comparing the Euclidean distance between the ML output of the image and the ML outputs stored for each user, the one that matches or is close enough (accuracy above the threshold) the application brings the data and requests a password.



$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

Euclidean distance formula

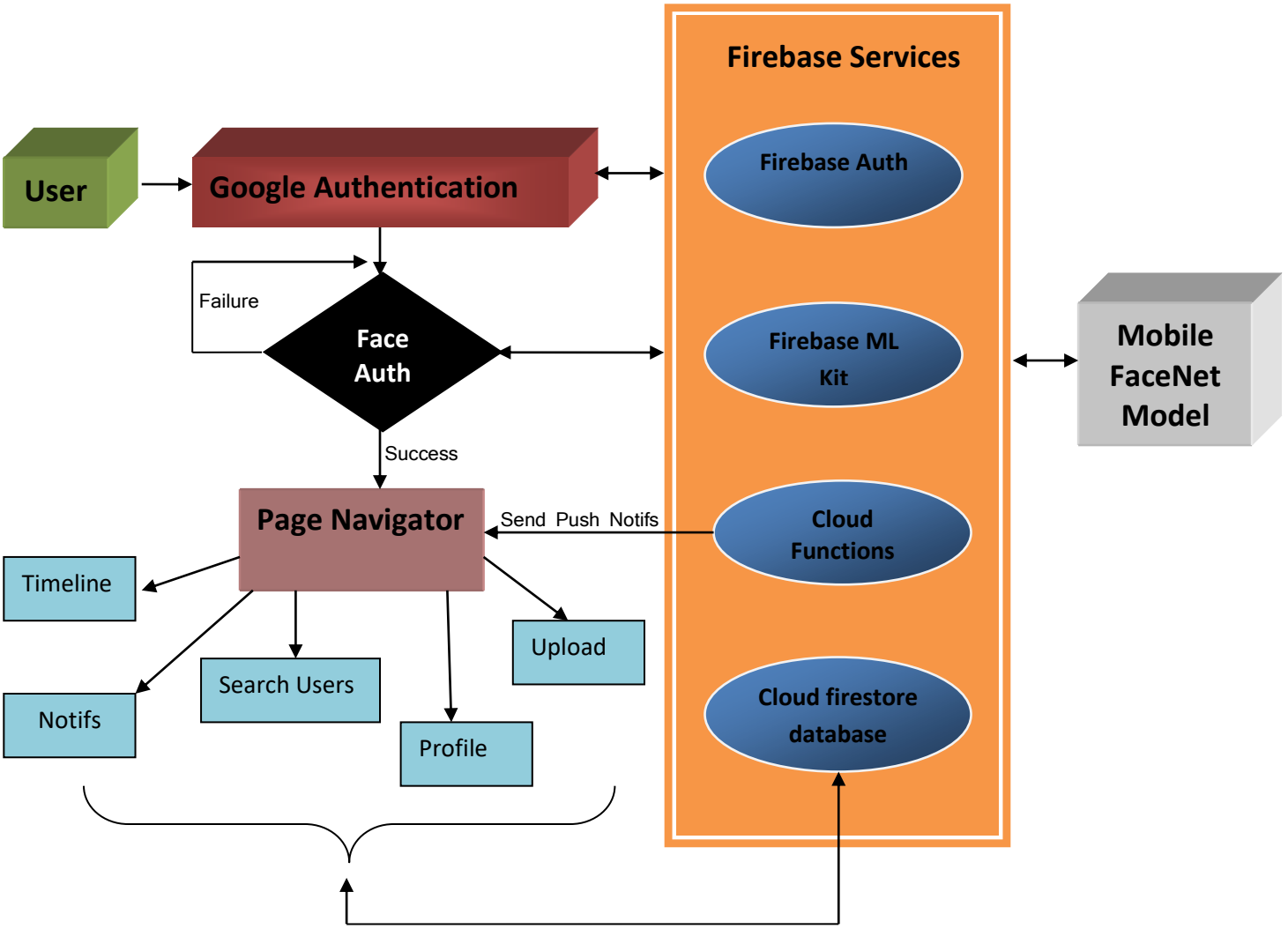
4. Frontend – (Flutter Framework)

Flutter is an open-source UI software development kit created by Google. It is used to develop applications for Android, iOS, Linux, Mac, Windows, Google Fuchsia, and the web from a single codebase. This is the platform we are going to use to design the front end of our application. The coding will be done in the dart language which was also developed by Google.

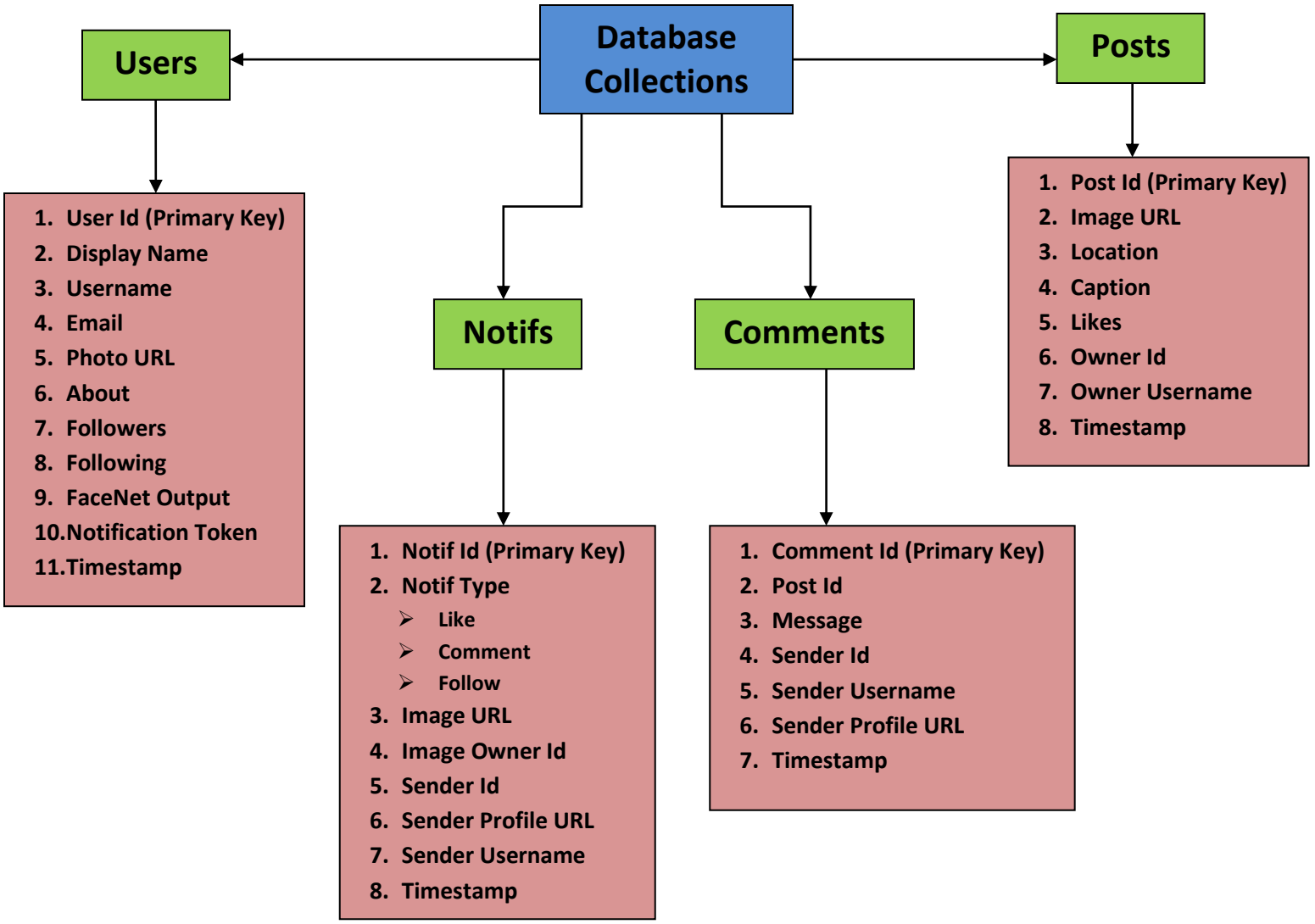
5. Backend – (Firebase and Node Js)

Firebase is also is a platform developed by Google for storing data and accessing using the Cloud Firestore Database. The primary authentication is done through this platform for all the users to sign in.
NodeJs is used for creating trigger functions in order to send push notifications and confirmation emails to all the users in real time.

6. Architecture Diagram



7. Cloud-Firestore Database



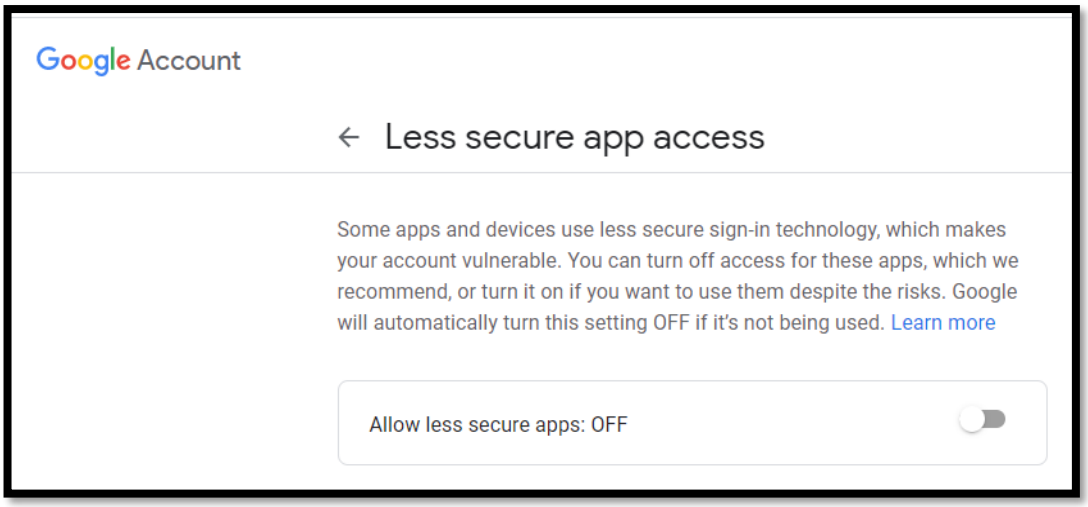
8. Cloud Functions

Once the user signs in we want to program 2 functions. One function to send a welcome message to the registered email and another to send push notifications if that user receives any new notifications

Modules required for the functions:

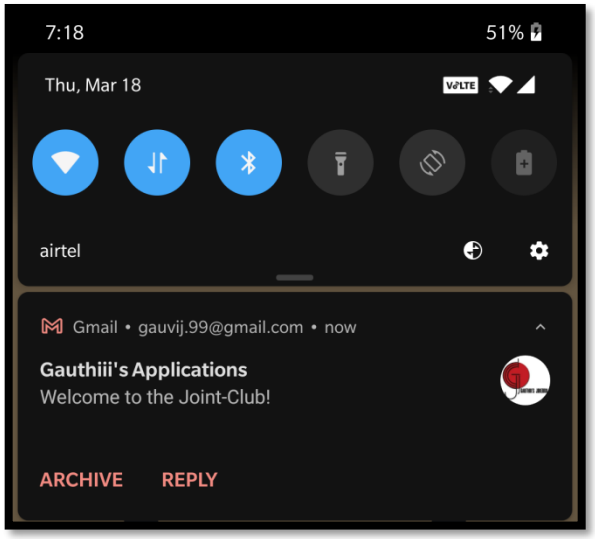
- Firebase-functions
- Firebase-Admin
- Node-Mailer
- Cors

Now we need to go to this link: https://myaccount.google.com/lesssecureapp and enable Less Secure Apps to sign in from the respective Gmail account from which you want to send the automated messgage.




```
let transporter = nodemailer.createTransport({
service: 'gmail',
auth: {
user: '<email address>',
pass: '<password>'  //your password
}
});
exports.sendMail = functions.https.onRequest((req, res) => {
cors(req, res, () => {
// getting destination email by query string
const dest = req.query.dest;
const mailOptions = {
from: "Gauthiii's Applications <gauthamcrazychicken1999@gmail.com>", //
to: dest,
subject: 'Welcome to the Joint-Club!', // email subject
html: `Dear User, Welcome to JC, <p>You have logged into the JC App`
};
// returning result
```

```
return transporter.sendMail(mailOptions, (erro, info) => {
if(erro){
return res.send(erro.toString());
}
return res.send('Email Sent');
});
});
});
```



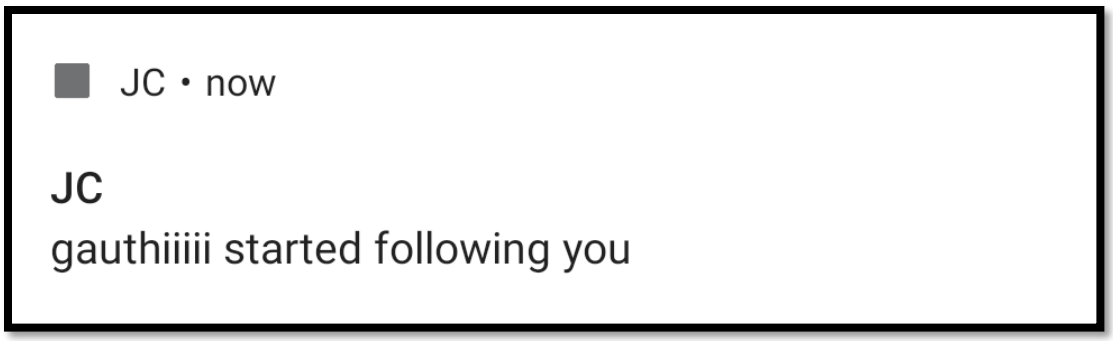
Now in order to create a function for every notification the user gets this is what the code looks like :

```
exports.onCreateActivityFeedItem = functions.firestore
  .document("/feed/{userId}/feedItems/{activityFeedItem}")
  .onCreate(async (snapshot, context) => {
    console.log("Activity Feed Item Created", snapshot.data());
    // 1) Get user connected to the feed
    const userId = context.params.userId;
    const userRef = admin.firestore().doc(`users/${userId}`);
    const doc = await userRef.get();
    // 2) Once we have user, check if they have a notification token; send notification, if they have a token
```

```
const androidNotificationToken = doc.data().androidNotificationToken;
const createdActivityFeedItem = snapshot.data();
if (androidNotificationToken) {
  sendNotification(androidNotificationToken, createdActivityFeedItem);
} else {
  console.log("No token for user, cannot send notification");
}
function sendNotification(androidNotificationToken, activityFeedItem) {
  let body;
  // 3) switch body value based off of notification type
  switch (activityFeedItem.type) {
    case "comment":
      body = `${activityFeedItem.username} replied: ${
        activityFeedItem.commentData
      }`;
      break;
    case "like":
      body = `${activityFeedItem.username} liked your post`;
      break;
    case "follow":
      body = `${activityFeedItem.username} started following you`;
      break;
    default:
      break;
  }
  // 4) Create message for push notification
  const message = {
    notification: { body },
    token: androidNotificationToken,
    data: { recipient: userId }
```

```
    };
    // 5) Send message with admin.messaging()
    admin
      .messaging()
      .send(message)
      .then(response => {
        // Response is a message ID string
        console.log("Successfully sent message", response);
      })
      .catch(error => {
        console.log("Error sending message", error);
      });
  }
});
exports.onCreateAccount = functions.firestore
.document("/users/{userId}")
.onCreate(async (snapshot, context) => {
  console.log("Account Created", snapshot.data());
  // 1) Get user connected to the feed
  const userId = context.params.userId;
  const userRef = admin.firestore().doc(`users/${userId}`);
  const doc = await userRef.get();
  // 2) Once we have user, check if they have a notification token; send notification, if they have a token
  const androidNotificationToken = doc.data().androidNotificationToken;
  const createdAccountItem = snapshot.data();
  if (androidNotificationToken) {
    sendNotification(androidNotificationToken, createdAccountItem);
  } else {
    console.log("No token for user, cannot send notification");
  }
```

```
 function sendNotification(androidNotificationToken, activityFeedItem) {
  let body;
   body=`Welcome to JC ${activityFeedItem.displayName}`;
  // 4) Create message for push notification
  const message = {
   notification: { body },
   token: androidNotificationToken,
   data: { recipient: userId }
 };
  // 5) Send message with admin.messaging()
  admin
   .messaging()
   .send(message)
   .then(response => {
    // Response is a message ID string
    console.log("Successfully sent message", response);
   })
   .catch(error => {
    console.log("Error sending message", error);
   });
  }
});
```



9. Software Requirements

The software requirements to create this application would be:

- Flutter
- Visual Studio Code (Any IDE)
- Android Studio
- NodeJs
- A registered Google Account
- A Firebase account to manage your Console
- Java SE Development kit (version 8 or above)
- A mobile device to test with the developer option enabled
- Git

10. Dependencies Used

A dependency is another package that your package needs in order to work. Dependencies can be installed using the pubspec.yaml file of every Flutter project folder.So the dependencies that we would need are:

- cupertino_icons
- firebase_messaging
- google_sign_in
- cloud_firestore
- firebase_storage
- image_picker
- firebase_auth
- path_provider
- uuid
- geolocator
- cached_network_image
- timeago
- tflite_flutter
- camera
- path
- firebase_ml_vision
- image
- flutter_local_notifications
- fluttertoast
- clippy_flutter

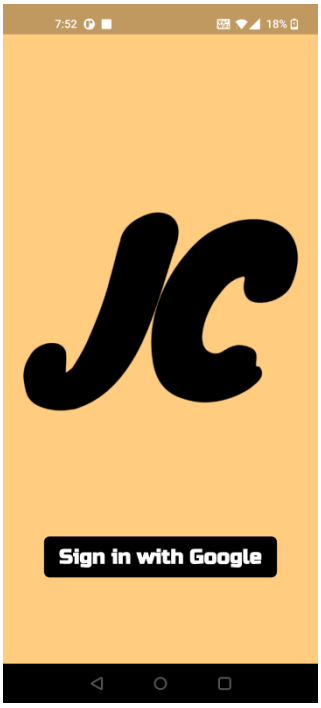
11. Source Code

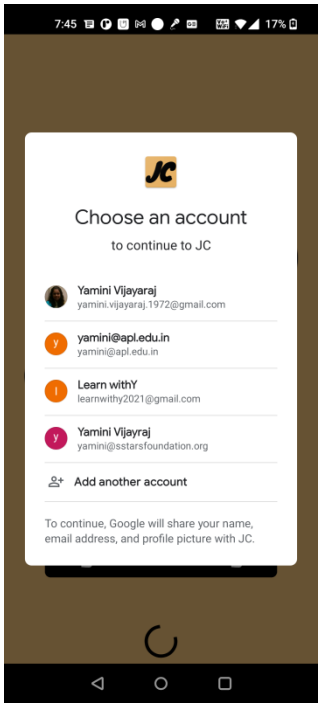
The source code for this application can be accessed from the “lib” folder from the given link to the repository:

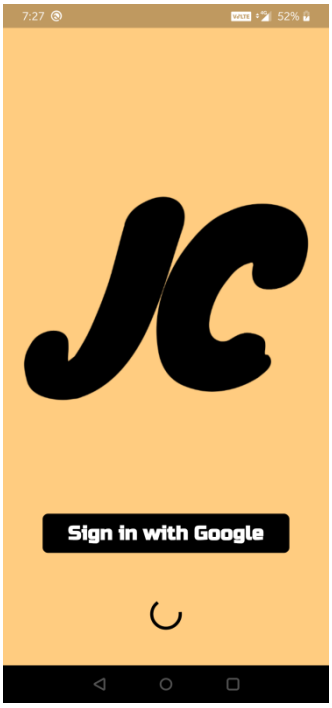
https://github.com/gauthiii/jc-app

12. SCREENS

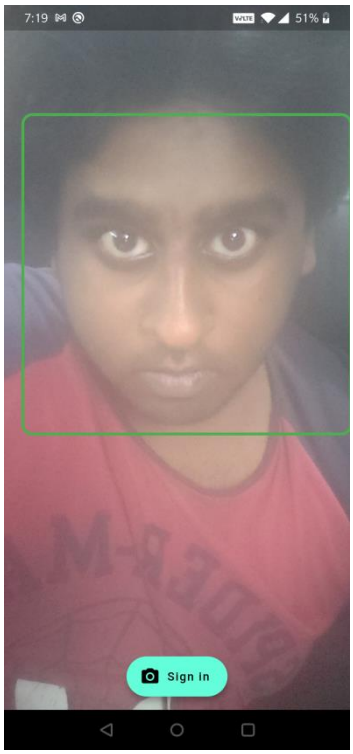
- **Login Screen**

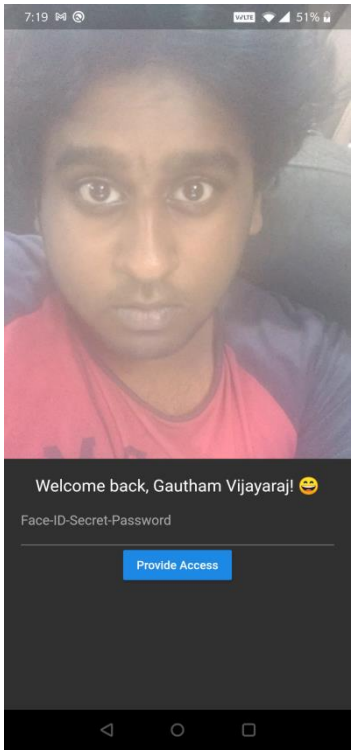


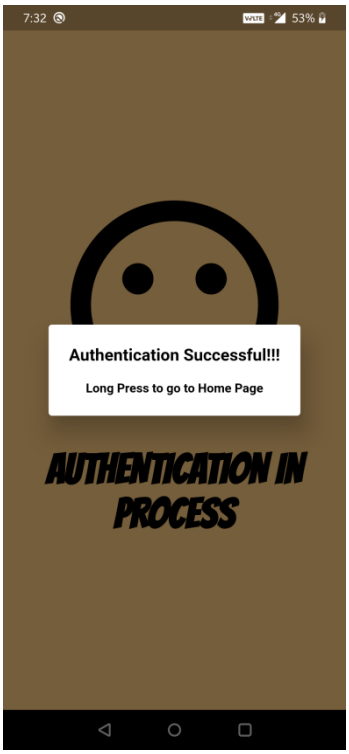




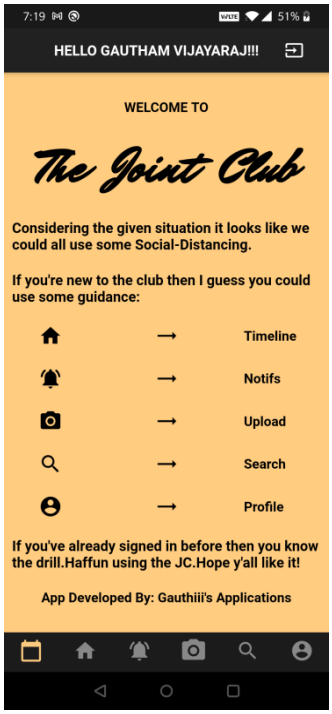
- **Face Authentication Sign In**



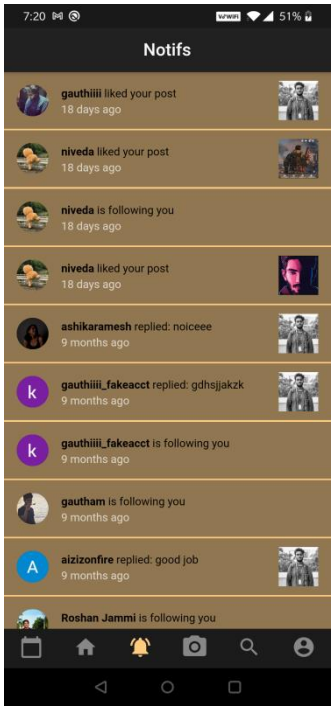




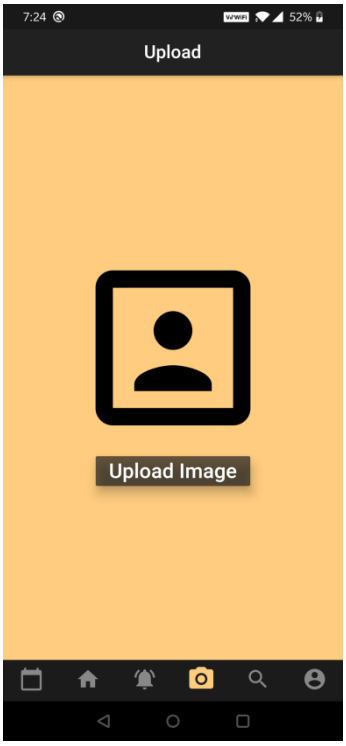
- **Home Screen**

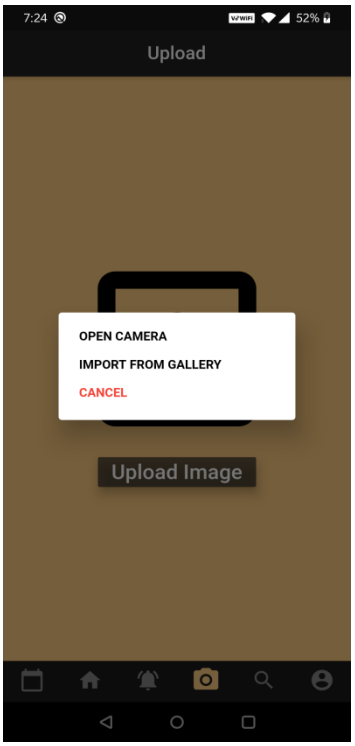


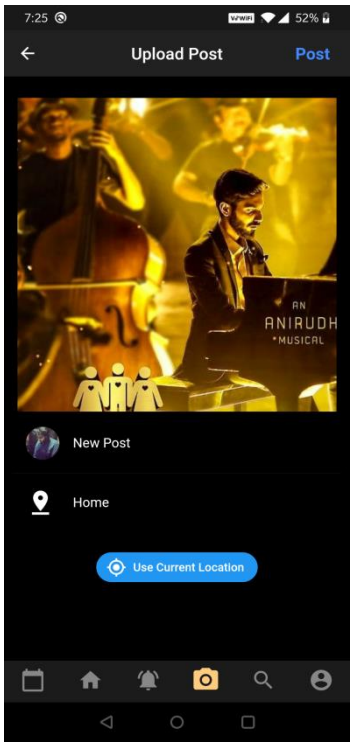
- **Notifications Screen**

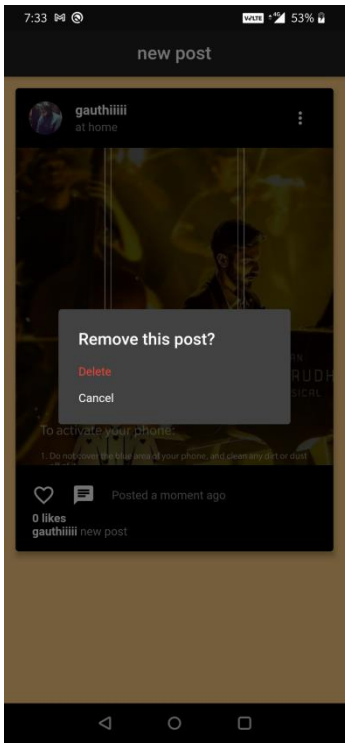


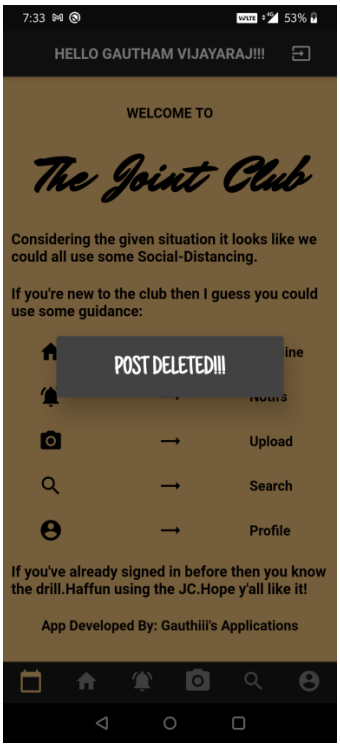
➢ **Upload Screen**



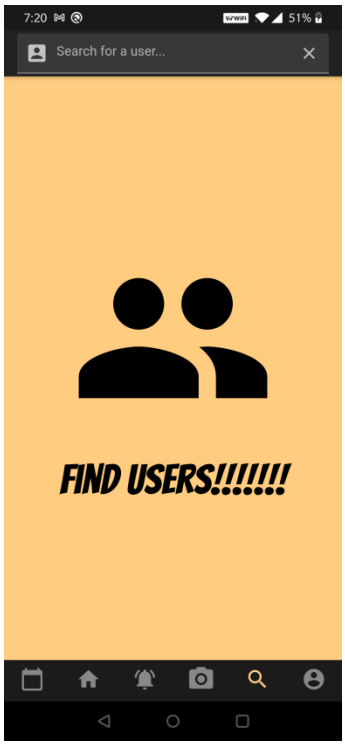


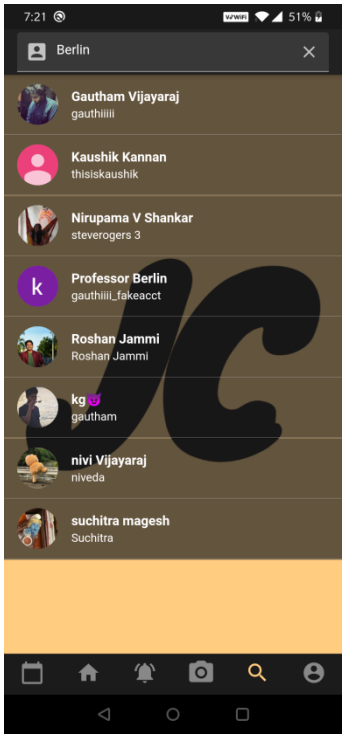




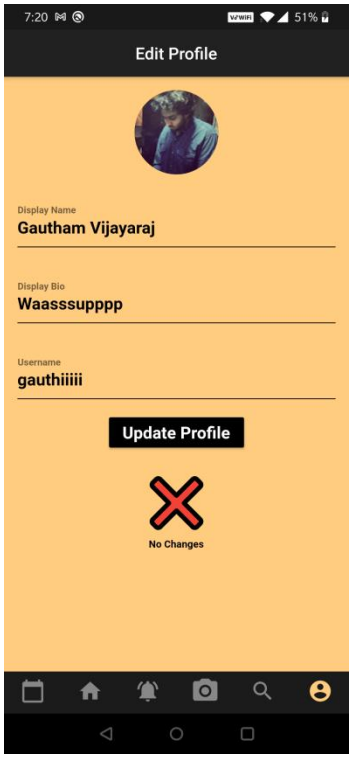


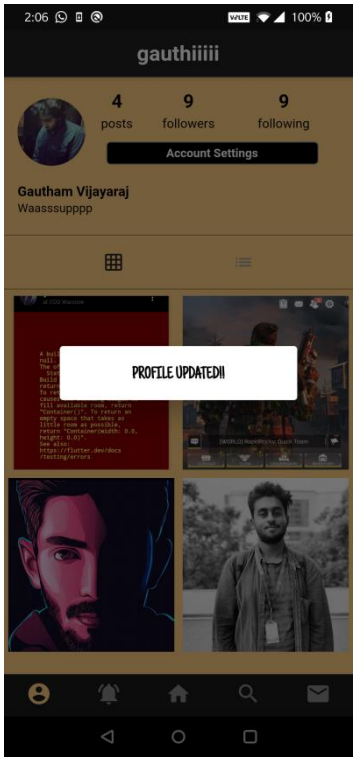
- **Search Page**





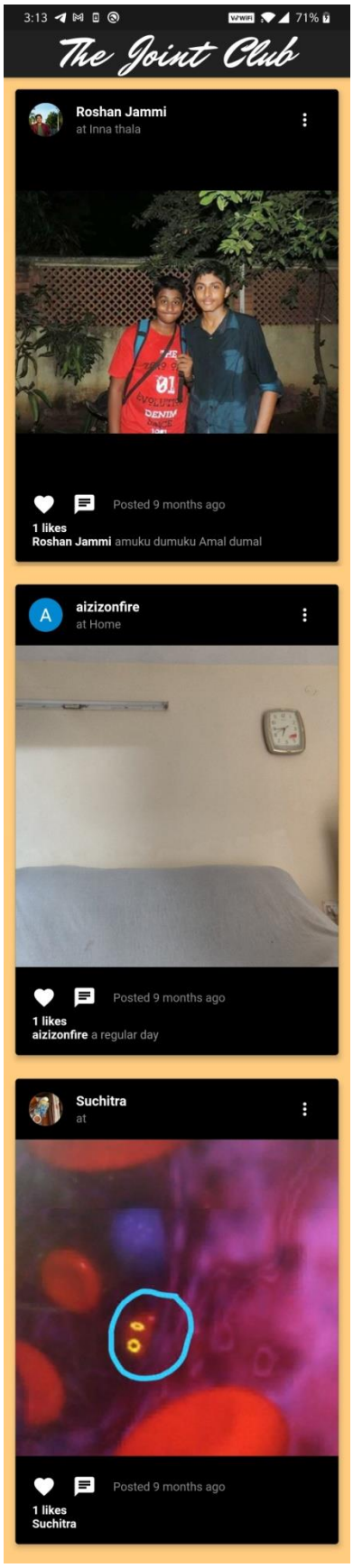
- **Edit Profile**



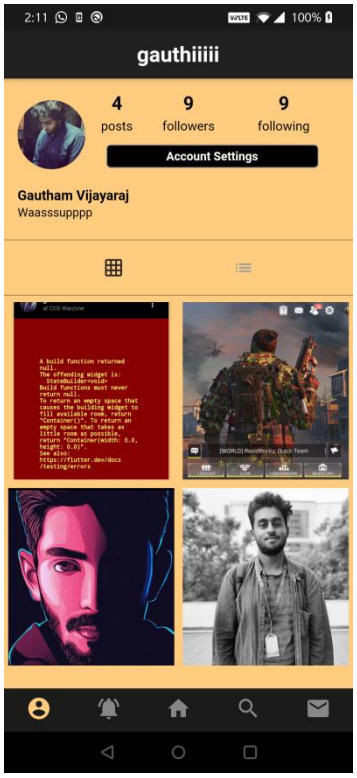


- **Profile (List View)**

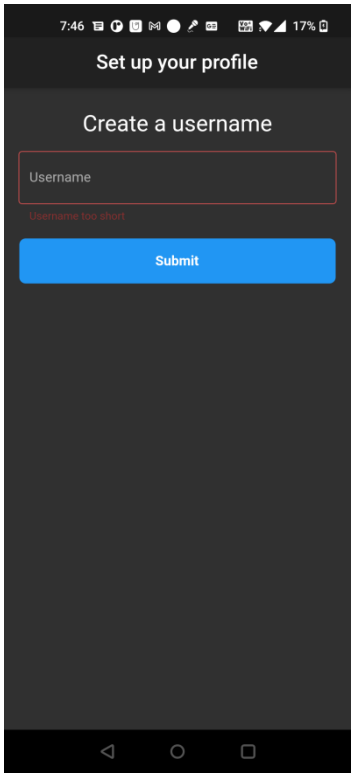
- **Timeline**


3:13
71%
The Joint Club
Roshan Jammi
at Inna thala
Posted 9 months ago
1 likes
Roshan Jammi amuku dumuku Amal dumal
aizizonfire
at Home
Posted 9 months ago
1 likes
aizizonfire a regular day
Suchitra
at
Posted 9 months ago
1 likes
Suchitra

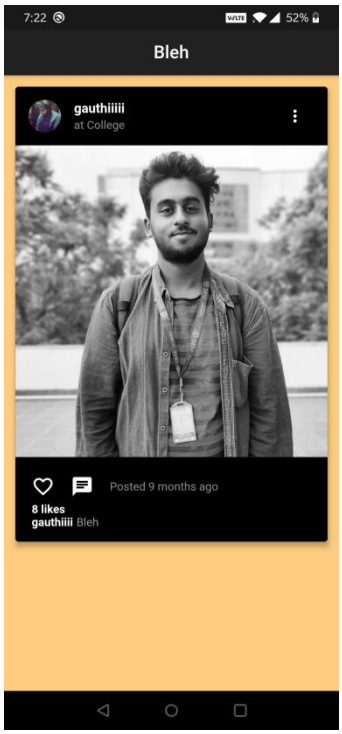
- **Profile (Grid View)**

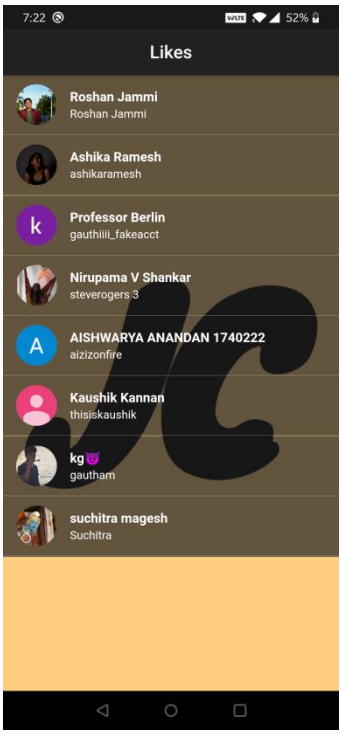


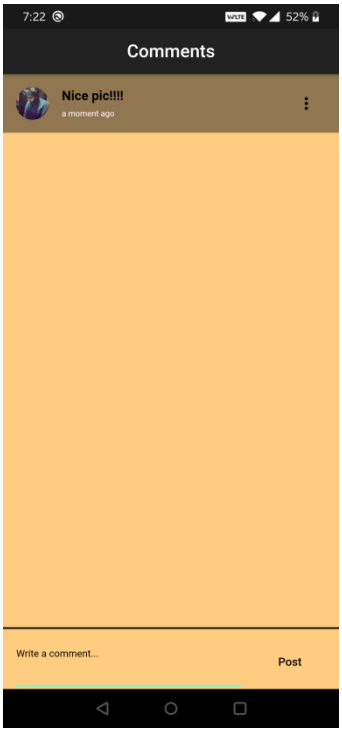
- **Username Setup**

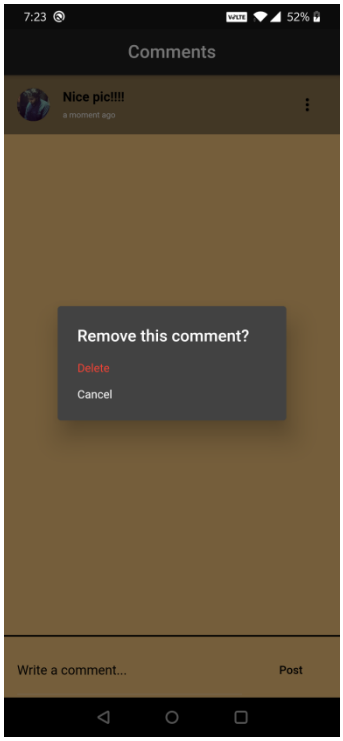


➢ **Post Screen**


7:22
52%
Bleh
gauthiiiii
at College
Posted 9 months ago
8 likes
gauthiiii Bleh


7:22
52%
Likes
Roshan Jammi
Roshan Jammi
Ashika Ramesh
ashikaramesh
Professor Berlin
gauthiiii_fakeacct
Nirupama V Shankar
steverogers.3
AISHWARYA ANANDAN 1740222
aizizonfire
Kaushik Kannan
thisiskaushik
kg
gautham
suchitra magesh
Suchitra

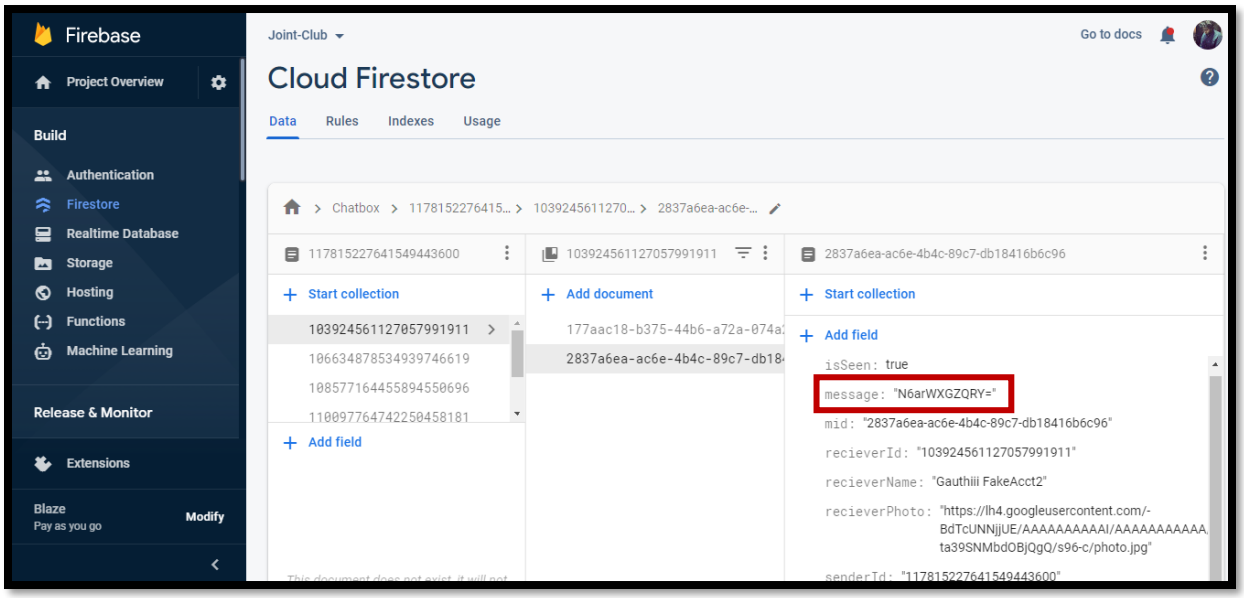

7:22
52%
Comments
Nice pic!!!!
a moment ago
Write a comment...
Post


7:23
52%
Comments
Nice pic!!!!
a moment ago
Remove this comment?
Delete
Cancel
Write a comment...
Post

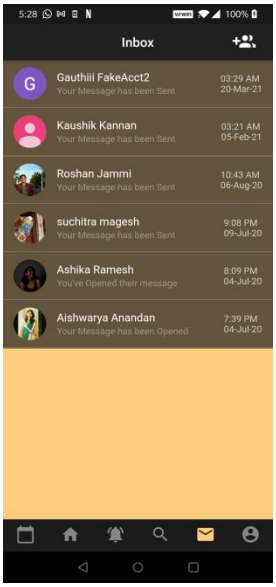
13. MESSENGER FEATURE

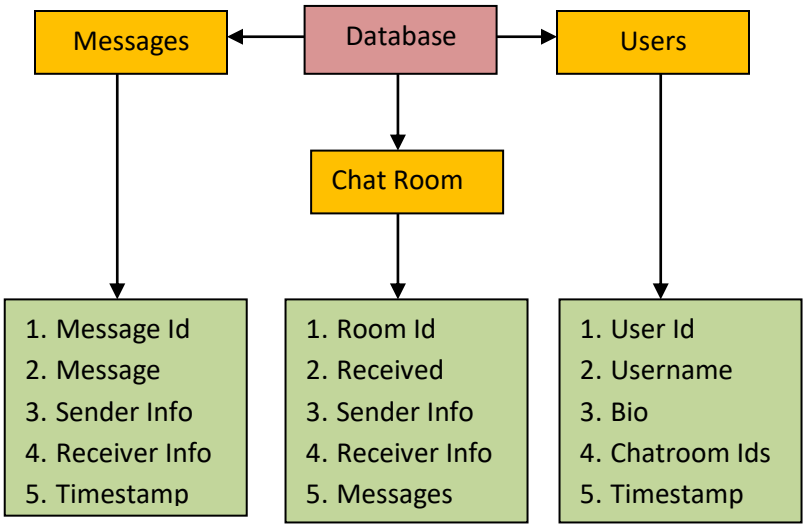
This Application wouldn't be a complete Social Networking App without the chatting feature so that's the next feature that is going to be focused on. The messages would be encrypted using the Triple DES algorithm and stored in the cloud firestore and few more additional collections in the database.

The message which gets stored in the database looks like this:



Once the above are perfectly implemented the Conversation Screen and the Inbox Screen looks like this:





16. HOW IS THIS APPLICATION DIFFERENT FROM THE EXISTING APPLICATIONS

To sum up, JC App (Joint-Club – the name of the app) is different from the rest of the following apps because this app has Face-Recognition enabled authentication. Even if there are other social media apps which facial recognition, the JC uses FaceNet Model for process, classify and transform into a data structure ‘savable’ by a database (an array of numbers). The next additional feature that is implemented is regarding the security of the conversations. Data leaks are very common nowadays and the Chat-Screen is designed in such a way that the conversations can neither be saved by taking Screenshots nor Screen-Recorded by the users. As of now the messages are encrypted using the Triple DES Algorithms. However the application would be updated once new ideas are brainstormed to make this application more efficient and secure.

17. ACCESS TO THE APP

The application can be tested from the apk file that can be downloaded from the QR Code given below.

References

1. Varsha R. Mouli, and K.P. Jevitha, "Web Services Attacks and Security- A Systematic Literature Review" in 6th International Conference On Advances In Computing & Communications(ICACC), 2016, pp.1-8.
2. Hongbin Liu, Shrideep Pallickara, and Geoffrey Fox, "Performance of Web Services Security", Proceedings of the 13th Mardi Gras conference, 2005, pp.1-8.
3. Zhiyong Zhang, and Brij B.Gupta, "Social media security and trust worthiness: Overview and new direction", Future Generation Computer Systems(FGCS), 2016, pp.914-925
4. Ashu Kumar, Amandeep Kaur, and Munish Kumar," Face detection techniques: a review", Artificial Intelligence Review, 2018, pp.1-23.
5. Dr. M. Ilayaraja, Dr. K.Shankar, and Dr. G. Devika, "A Modified Symmetric Key Cryptography Method for Secure Data Transmission", International Journal of Pure and Applied Mathematics(IJPAM), Volume 116 , No. 10 , 2017,pp 301-308.
6. Priyadarshini Patila, Prashant Narayankarb, Narayan D G c, and Meena S Md, "A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish", International Conference on Information Security & Privacy(ICISP), 2015, pp.617-624.
7. Nilanjan Chatterjee, Souvik Chakraborty, Aakash Decosta, Dr. Asoke Nath, "Real-time Communication Application Based on Android Using Google Firebase", International Journal of Advance Research in Computer Science and Management Studies(IJARCSMS), 2018, pp. 74-79.
8. Prasetyawidi Indrawan , Slamet Budiyatno , Nur Muhammad Ridho , and Riri Fitri Sari, "Face Recognition for Social Media With Mobile Cloud Computing", International Journal on Cloud Computing: Services and Architecture (IJCCSA), 2013, pp. 23-35.
9. Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, Lior Wolf, "DeepFace: Closing the Gap to Human-Level Performance in Face Verification", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 1701-1708.
10. Levent Gokrem , Omer Faruk Nasip, "An Encrypted Messaging Application with Multi Fragmented Caesar Encryption Method between Mobile Devices", Journal of New Results in Science (JNRS), 2017, pp. 1-10.
11. Timmy Schenkel, Oliver Ringhage, Nicklas Branding, "A COMPARATIVE STUDY OF FACIAL RECOGNITION TECHNIQUES With focus on low computational power", Bachelor Degree Project in Information Technology Basic level 30 ECTS Spring term 2019.
12. "Real-time Image classification using Tensorflow Lite and Flutter", Marcos Carlomagno,Sep 5 2020
13. Milind Deore , "FaceNet Architecture",Feb 24 2020
14. Marcos Carlomagno , "Face Recognition Authentication using Flutter and Tensorflow Lite , Sep 19 2020