A PROJECT REPORT

ON

ATTENDANCE USING FACE RECOGNITION

SUBMITTED TO

KIIT DEEMED TO BE UNIVERSITY

In partial fulfillment of the Requirement for the award of

BACHELOR's DEGREE IN

INFORMATION TECHNOLOGY

BY

BEDABRATA BORA	**1706123**
PRIYANKA RAJ	**1706145**
SATYAM SHAH	**1706157**
SHASHWATA MUKHERJEE	**1706160**

UNDER THE GUIDANCE OF

PROF. SANKALP NAYAK

SCHOOL OF COMPUTER ENGINEERING
KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY
BHUBANESWAR, ODISHA- 751024

MAY 2020

CERTIFICATE

This is to certify that the project entitled

ATTENDANCE USING FACE RECOGNITION

Submitted by

BEDABRATA BORA

PRIYANKA RAJ

SATYAM SHAH

SHASHWATA MUKHERJEE

is a record of bonafide work carried out by them, in the partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering(Information Technology) at KIIT Deemed to be University, Bhubaneswar. This work is done during year 2019-2020, under our guidance.

Date- 10/06/2020

(Prof. Sankalp Nayak)
Project Guide

ABSTRACT

In recent trends industries, organizations and many companies are using personal identification strategies like fingerprint identification, RFID for tracking attendance etc. Among all these personal identification strategies face recognition is the most natural, less time consuming and highly efficient. The main Strategy involved in this project is taking attendance in organizations, industries etc. using face detection and recognition technology.

This system is most effective, easy and less time consuming for tracking attendance in organizations with period wise without and human intervention. This system also provides a contactless and seamless form of attendance tracking.

Keywords: RFID, Face detection recognition, AWS Rekogniton, Firebase

CONTENTS

TOPIC	Pg. No
1. Introduction	1
1.1 Problem Statement…….	1
1.2 Solution……..	1
1.3 Objective……	1
2. Literature Survey	2
2.1 Literature……	2
3. Requirements Specification	3
3.1 Hardware Requirements……	3
3.2 Software Requirements…….	3
4. Requirement Analysis	4
4.1 Functional Requirements……..	4
4.2 Nonfunctional Requirements……	4
5. System Design	5-7
5.1 Model……	5
5.2 View……..	5
5.3 Controller……..	7
6. System Testing	8
6.1 Test Cases and Results…….	8
7. Project Planning	9
7.1 Project Goals…….	9
7.2 Project Deliverables……..	9
7.3 Project Schedule……	9
7.4 Analyze Project Quality and Identify Risk……	9
8. Implementation	10-14
8.1 Cloud Implementation……...	10
8.2 Android Interface……..	12
9. Project Screenshots…….	15-17
10.Conclusion and Future Scope……..	18
11.References……...	19

LIST OF FIGURES

Figure	Pg. No
Fig 5.2.1	5
Fig 5.2.2	6
Fig 5.2.3	6
Fig 5.3.1	7
Fig 8.1.1	10
Fig 8.1.2	11
Fig 8.2.1	12
Fig 8.2.2	13
Fig 8.2.3	14
Fig 9.1	15
Fig 9.2	15
Fig 9.3	16
Fig 9.4	16
Fig 9.5	17
Fig 9.6	17

1. INTRODUCTION

Facial recognition or face recognition as it is often referred to as, analyses characteristics of a person's face image input through a camera. It measures overall facial structure, distances between eyes, nose, mouth, and jaw edges. These measurements are retained in a database and used as a comparison when a user stands before the camera. One of the strongest positive aspects of facial recognition is that it is non-intrusive. Verification or identification can be accomplished from two feet away or more, without requiring the user to wait for long periods of time or do anything more than look at the camera.

1.1. PROBLEM STATEMENT:

Traditionally student's attendance is taken manually by using attendance sheet, given by the faculty member in class. The Current attendance marking methods are monotonous & time consuming. Manually recorded attendance can be easily manipulated. Moreover, it is very difficult to verify one by one student in a large classroom environment with distributed branches whether the authenticated students are actually responding or not.

1.2. THE SOLUTION

To tackle the above mentioned issues, we came up with a solution to automate the attendance system using facial recognition system them will be able to save the time and effort this tedious process.

1.3. OBJECTIVE

The main objective of the project is to provide an automated attendance system that is practical, reliable and eliminates disturbance and time loss in traditional attendance systems. The primary goal of this project is to develop a cloud-based solution that will automate the attendance taking process in a self-contained classroom environment.

The project will consist of three main components:

- The Cloud component will be responsible for recognizing students.
- The android application component will allow teachers to create students, add classes and view attendance history.
- The Raspberry pie component that will be responsible for uploading classroom images to AWS for processing.

2. LITERATURE SURVEY

2.1. LITERATURE

The facial biometric system has a whole range of use from a big multinational company to universities and schools. Facial recognition is one of the most critical components of the project and that makes it a logical place to start the development process. After doing some extensive research, the first tool tested was OpenCV. OpenCV (Open Source Computer Vision Library) is an open source computer vision library with machine learning capabilities.

The two features that made OpenCV appealing was its ability to use a camera to capture video, and its built-in facial recognition library. OpenCV can be programmed using C++, Java or Python. Python was used to program OpenCV for this project because of its prebuilt libraries and detailed documentation. The image capturing feature of OpenCV was successful in capturing images and will be used for processing classroom images as part of the solution. The Facial recognition feature in OpenCV uses the Local Binary Patterns Histograms LBPH algorithm, but it required multiple images of each
student to achieve a high success rate in recognition. The documentation recommended at least eight images of each student, however that number appeared to be much higher because even after increasing it to 30 images the success rate was still in the 40 percent range. That range does not meet the requirements. Since OpenCV did not meet the success rate, it was necessary to explore other tools.

After exploring other recognition tools, Amazon Web Services (AWS) was chosen because their services met all the requirements. The services that were used include S3 (Simple Storage Service), Rekognition, Lambda, RDS (Relational Database Services), EC2 (Amazon Elastic Compute Cloud) and CloudWatch. All programming was done using the Python programming language because OpenCV and AWS supported that language. Instead of using full video, only classroom images were used for facial recognition processing because it reduced the amount of bandwidth needed to the upload images and reduced cost.

3. REQUIREMET SPECIFICATION

3.1. HARDWARE REQUIREMENTS:

a. Raspberry Pi-3b
b. Camera module for Raspberry pi

3.2. SOFTWARE REQUIREMENTS:

a. Android Studio
b. S3 Bucket
c. Dynamo DB
d. IAM and Lambda

4. REQUIREMENT ANALYSIS

4.1. FUNCTIONAL REQUIREMENTS:

In this project, it was important to gather some requirements that will be needed to achieve the objectives set out previously. With client (user) story a use case analysis was implemented which resulted in the following functional and non-functional requirements were captured. The functional requirements have been gathered from the user story developed from the minutes collected during meetings with the client and are outlined here.

a. Capture face images via webcam or external USB camera.
b. A professional HD Camera.
c. Faces on an image must be detected.
d. The faces must be detected in bounding boxes.
e. Compute the total attendance based on detected faces.
f. Crop the total number of faces detected.
g. Resize the cropped faces to match faces the size required for recognition.
h. Store the cropped faces to a folder.
i. Load faces on database.
j. Train faces for recognition.
k. Perform recognition for faces stored on database.
l. Compute recognition rate of the system.
m. Perform recognition one after the other for each face cropped by Face Detector.
n. Display the input image alongside output image side by side on the same plot.
o. Display the name of the output image above the image in the plot area
p.

4.2. NON FUNCTIONAL REQUIREMENTS

Non-functional requirements are set of requirements with specific criteria to judge the systems operation. These requirements have been collected based on the following after meetings with the client. They cover ease of use to the client, security, support availability, operational speed, and implementation considerations. More specifically:

a. The user will inform the students when taking a photo with clear instructions on how to position their faces.
b. The system is very secure.
c. The system can be easily installed.
d. The system is 100% efficient.
e. The system will have a response time of 10 seconds.
f. The system must be fast and reliable.

5. SYSTEM DESIGN

5.1. MODEL

The basic structure of the Project is that:

- Capturing the Student's image of the face.
- Storing the images in the database.
- Detecting the face and matching the detected face with the database.
- If matched then API request from the database and load data to the Firebase database.
- Data can be retrieved from the Firebase.
- Log in the application and check the attendance.

5.2. VIEW

The application's user interface which is visible to the user is the Application which has been formed in the project.

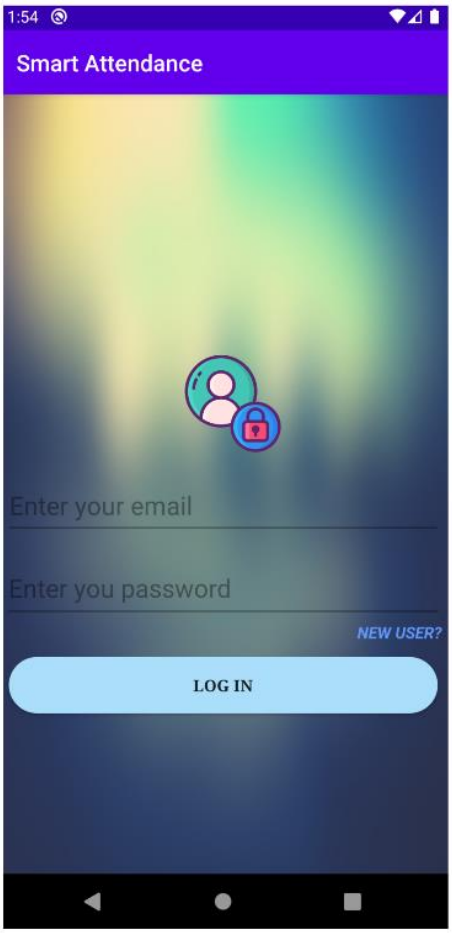


Fig.5.2.1. Login Page Interface for Application

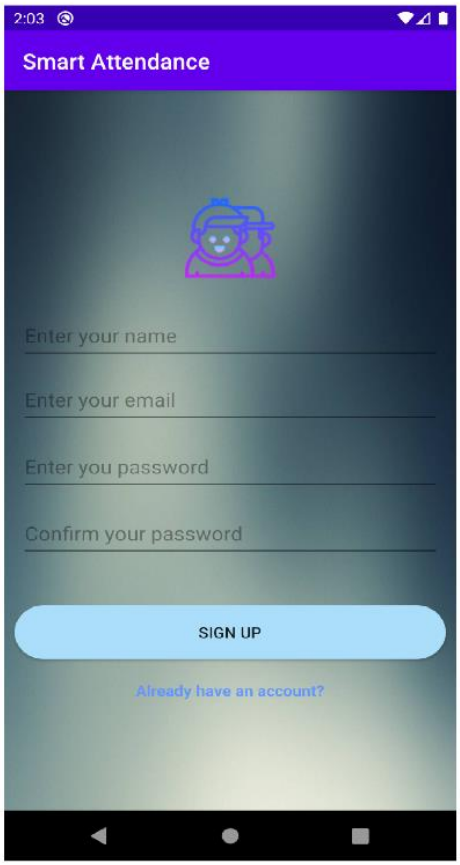


Fig 5.2.2. Registration Page Interface for Application

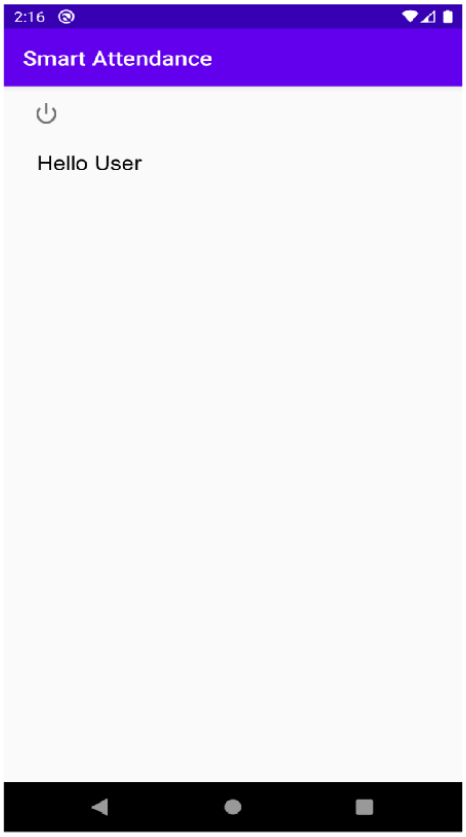


Fig 5.2.3. Home Page of Application

5.3. CONTROLLER

The components that exist between the view and the model comes under the controller.

Here, we are using AWS services to connect the user and the system.

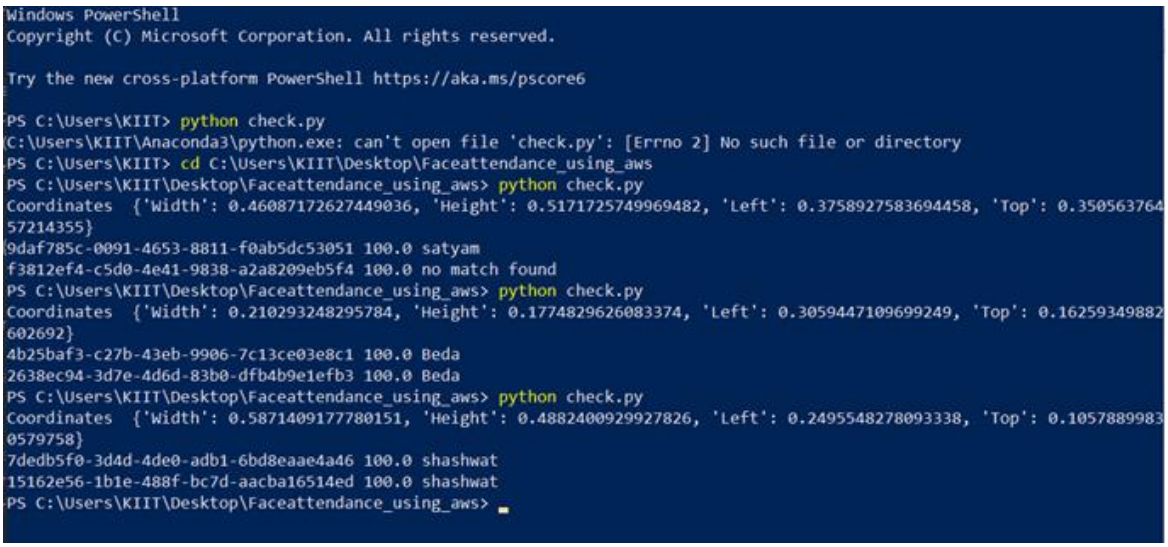


Fig 5.3.1. Connection between AWS and System

6. SYSTEM TESTING

6.1 Test Cases and Test Results

Test ID	Test Case Title	Test Condition	System Behavior	Expected Result
T01	Click on the Login Button with Admin login credentials.	1) Email entered should be valid. 2) Password should be valid.	Admin Home page opens.	Admin Home page will open.
T02	Click on NEW USER	--	Registration Page opens.	Registration Page will open.
T03	Click on Sign Up Button	1) Name should be entered. 2) Email should be according to the requirements. 3) Password should be according to the requirements. 4) Confirm Password should be same as the entered Password.	User Home page opens.	User Home page will open.
T04	Click on the Already have an account Button.	--	Goes back to the Login Page.	Should Go back to the Login Page.
T05	Check for the Face.	1) Some Faces should be present in front of the camera. 2) The faces should be facing towards the camera.	The system recognizes all faces correctly.	The system should recognize all faces correctly.
T06	Automatic update of Attendance on Database.	1) Faces should be detected.	The Attendance was updated.	The Attendance should be updated.

7. PROJECT PLANNING

7.1. PROJECT GOALS

We first defined the goals of the Project. A project in only successful when it has met the needs of the Stakeholder. So, we identified the stakeholders of the project. The customer is the person who receives the deliverables. The users are Teachers and Students of the university.

The Project Manager and the Project team was defined.

After finding the Stakeholders, we created a set of measurable goals to be achieved. We prioritized and divided the goals accordingly.

7.2. PROJECT DELIVERABLES

After creating a list of goals, we calculated what is needed by the project to be executed at highest priority and be delivered initially. This requires setting delivery dates of each goal.

7.3. PROJECT SCHEDULE

We created a list of tasks required for each deliverable by defining the amount of efforts required (in hours or days) and the team member who will be responsible for the task.

For eg. - We planned a schedule of 15 days for the software development part which was divided among two of our team members.

There were a lot of changes in the days due to software changes as the optimized version had to be formed. We also held some team meetings to decide what all approach can be used to optimize the deliverables.

7.4. ANALYZE PROJECT QUALITY AND IDENTIFY RISKS

We tested our system under all conditions required for our system to work. We also found out what all risks can occur in the system and tried to improve it. We analyzed it many number of times to confirm the working of the system so that the user as well as the customer faces no issues while using the product.

8. IMPLEMENTATION

8.1 CLOUD IMPLEMENTATION

1. Raspberry pie takes the picture of the student and sends to Amazon Rekognition via SearchFaceByImage request as a byte stream.
2. Rekognition compares the images in collection and in response, it returns a JSON object containing the metadata (Co-ordinates, Confidence Score, Name) of the matches.
3. If the confidence score is greater than threshold, we update the dynamo db table with Students name and data.Using Android app, Students can access their attendance from dynamo db.

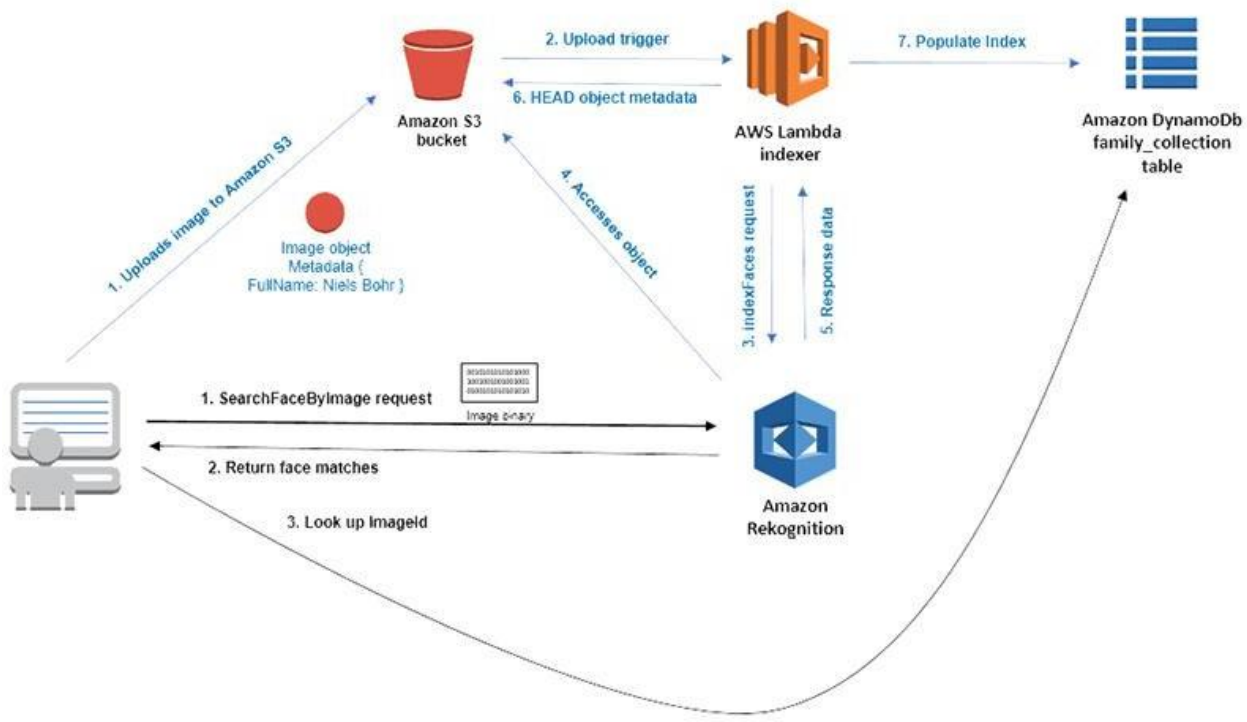


Fig 8.1.1. Cloud Structure

```python
import boto3
import io
from PIL import Image
from pprint import pprint

rekognition = boto3.client('rekognition', region_name='ap-south-1')
dynamodb = boto3.client('dynamodb', region_name='ap-south-1')

image = Image.open("image02.jpeg")
stream = io.BytesIO()
image.save(stream,format="JPEG")
image_binary = stream.getvalue()

response = rekognition.detect_faces(
    Image={'Bytes':image_binary}
        )

all_faces=response['FaceDetails']

# Initialize list object
boxes = []

# Get image diameters
image_width = image.size[0]
image_height = image.size[1]

# Crop face from image
for face in all_faces:
    box=face['BoundingBox']
    x1 = int(box['Left'] * image_width) * 0.9
    y1 = int(box['Top'] * image_height) * 0.9
    x2 = int(box['Left'] * image_width + box['Width'] * image_width) * 1.10
    y2 = int(box['Top'] * image_height + box['Height']  * image_height) * 1.10
    image_crop = image.crop((x1,y1,x2,y2))

    stream = io.BytesIO()
    image_crop.save(stream,format="JPEG")
    image_crop_binary = stream.getvalue()

    # Submit individually cropped image to Amazon Rekognition
    response = rekognition.search_faces_by_image(
            CollectionId='family_collection',
            Image={'Bytes':image_crop_binary}
            )

    if len(response['FaceMatches']) > 0:
        # Return results
        print ('Coordinates ', box)
        for match in response['FaceMatches']:

            face = dynamodb.get_item(
                TableName='family_collection',
                Key={'RekognitionId': {'S': match['Face']['FaceId']}}
                )

            if 'Item' in face:
                person = face['Item']['FullName']['S']
            else:
                person = 'no match found'

            print (match['Face']['FaceId'],match['Face']['Confidence'],person)
```

Fig 8.1.2. Cloud Code Snippet

8.2. ANDROID INTERFACE

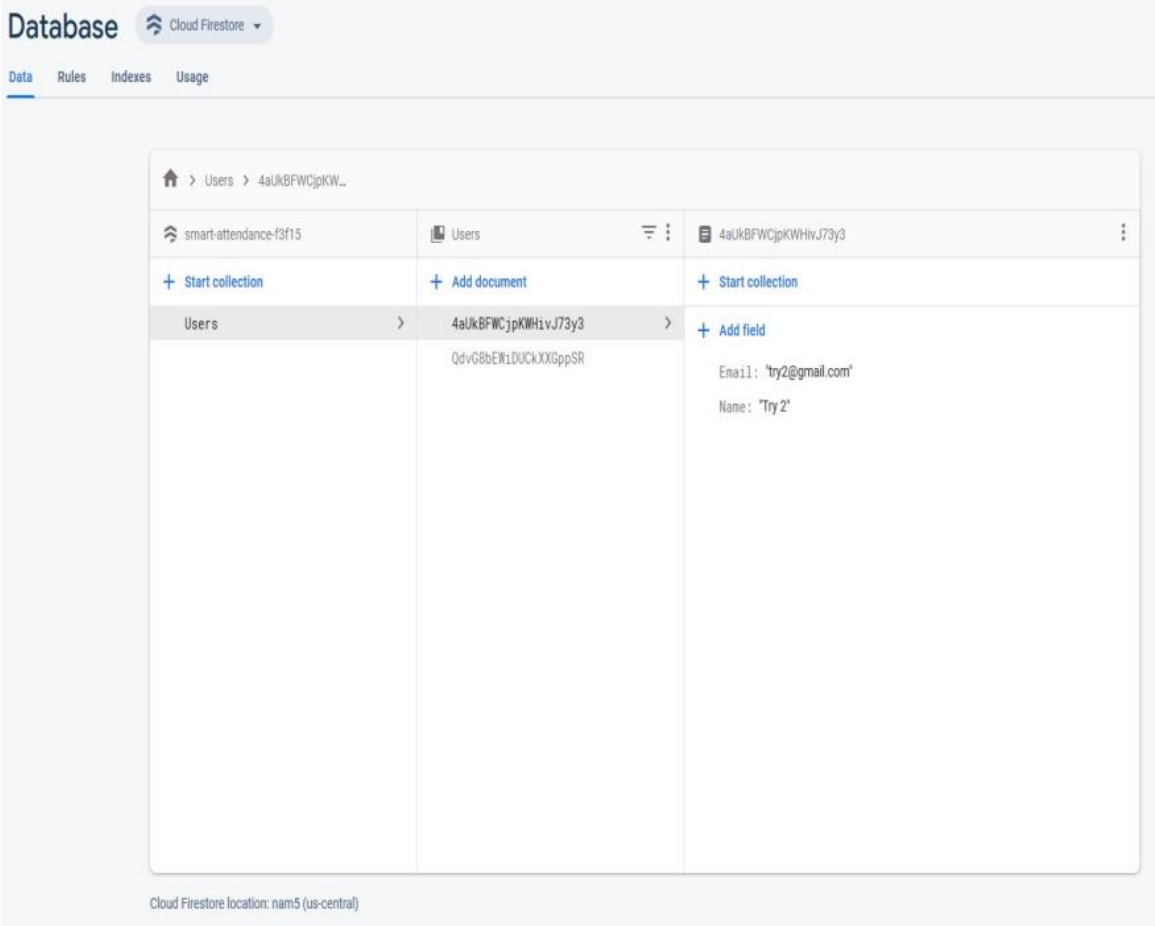


Fig 8.2.1. Firebase Database to store registered users

1. The user is first greeted with a login page. If the user account is already created, then the user can login to their account and view their attendance. Else the can create a new account in the registration Page.

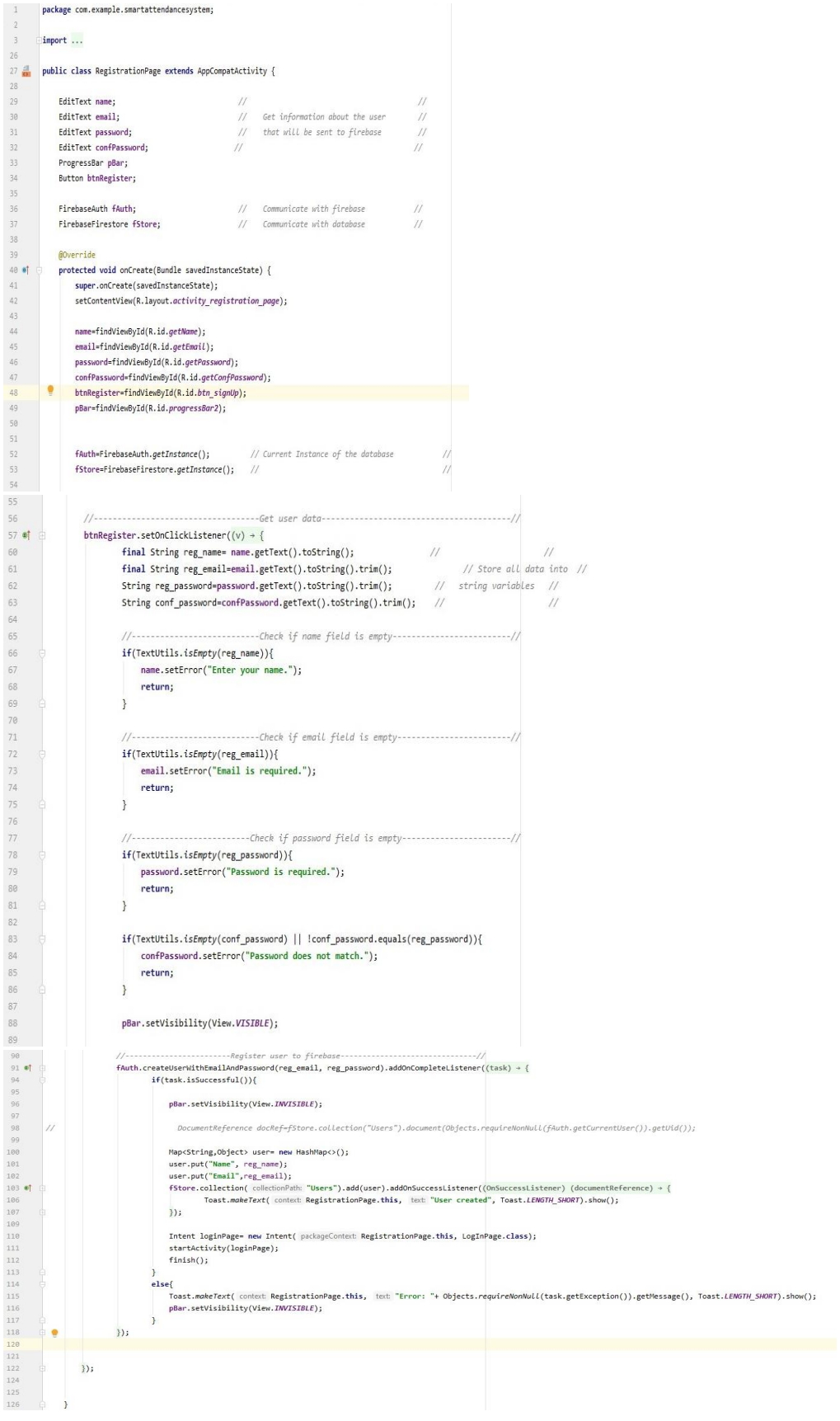
2. For new users, they will be asked for their full name, email and password with certain constraints.

3. The registered data will be added to the firebase database for every unique users.

4. The user data will be displayed on login

```java
package com.example.smartattendancesystem;

import ...

public class LogInPage extends AppCompatActivity {

    EditText email;                    //         User login information         //
    EditText password;                 //                                        //

    Button login;
    ProgressBar pBar;

    FirebaseAuth fAuth;                        //       Communicate with firebase          //
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_log_in_page);

        email= findViewById(R.id.enterEmail);
        password=findViewById(R.id.enterPassword);
        login= findViewById(R.id.btn_signIn);
        pBar=findViewById(R.id.loginProgress);

        fAuth=FirebaseAuth.getInstance();

        //-----------------If user is already loggin in, go to main page-------------------------//
        if(fAuth.getCurrentUser()!=null){
            Intent attendancePage= new Intent( packageContext: LogInPage.this, AttendancePage.class );
            startActivity(attendancePage);
            finish();
        }

        login.setOnClickListener((v) -> {
                String log_email=email.getText().toString();
                String log_password=password.getText().toString();

                //--------------------------Check if email field is empty-----------------------//
                if(TextUtils.isEmpty(log_email)){
                    email.setError("Email is required.");
                    return;
                }

                //-----------------------Check if password field is empty--------------------//
                if(TextUtils.isEmpty(log_password)){
                    password.setError("Password is required.");
                    return;
                }

                pBar.setVisibility(View.VISIBLE);

                //----------------------Authenticate the User---------------------------------//
                fAuth.signInWithEmailAndPassword(log_email,log_password).addOnCompleteListener((task) -> {

                        if(task.isSuccessful()){
                            pBar.setVisibility(View.INVISIBLE);
                            Toast.makeText( context: LogInPage.this,  text: "Logged in successfully.", Toast.LENGTH_SHORT).show();
                            Intent attendancePage= new Intent( packageContext: LogInPage.this, AttendancePage.class);
                            startActivity(attendancePage);
                            finish();
                        }

                        else{
                            Toast.makeText( context: LogInPage.this,  text: "Error: "+ Objects.requireNonNull(task.getException()).getMessage(), Toast.LENGTH_SHORT).show();
                            pBar.setVisibility(View.INVISIBLE);
                        }
                });
    });

    }

    //------------------Connect registration activity to login activity----------------------//
    public void goToRegistration(View v){
        Intent registrationPage= new Intent( packageContext: LogInPage.this, RegistrationPage.class );
        startActivity(registrationPage);

    }
}
```

Fig 8.2.2. Login Code Snippet



```
package com.example.smartattendancesystem;

import ...

public class RegistrationPage extends AppCompatActivity {

    EditText name;                          //                                  //
    EditText email;                         //   Get information about the user //
    EditText password;                      //   that will be sent to firebase  //
    EditText confPassword;                  //                                  //
    ProgressBar pBar;
    Button btnRegister;

    FirebaseAuth fAuth;                     //   Communicate with firebase      //
    FirebaseFirestore fStore;               //   Communicate with database      //

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_registration_page);

        name=findViewById(R.id.getName);
        email=findViewById(R.id.getEmail);
        password=findViewById(R.id.getPassword);
        confPassword=findViewById(R.id.getConfPassword);
        btnRegister=findViewById(R.id.btn_signUp);
        pBar=findViewById(R.id.progressBar2);


        fAuth=FirebaseAuth.getInstance();       // Current Instance of the database      //
        fStore=FirebaseFirestore.getInstance(); //                                       //

        //-----------------------------------Get user data-----------------------------------------//
        btnRegister.setOnClickListener((v) -> {
            final String reg_name= name.getText().toString();               //                  //
            final String reg_email=email.getText().toString().trim();             // Store all data into  //
            String reg_password=password.getText().toString().trim();       //   string variables   //
            String conf_password=confPassword.getText().toString().trim();  //                  //

            //--------------------------Check if name field is empty-------------------------//
            if(TextUtils.isEmpty(reg_name)){
                name.setError("Enter your name.");
                return;
            }

            //--------------------------Check if email field is empty-------------------------//
            if(TextUtils.isEmpty(reg_email)){
                email.setError("Email is required.");
                return;
            }

            //-----------------------Check if password field is empty----------------------//
            if(TextUtils.isEmpty(reg_password)){
                password.setError("Password is required.");
                return;
            }

            if(TextUtils.isEmpty(conf_password) || !conf_password.equals(reg_password)){
                confPassword.setError("Password does not match.");
                return;
            }

            pBar.setVisibility(View.VISIBLE);

            //----------------------Register user to firebase-----------------------------//
            fAuth.createUserWithEmailAndPassword(reg_email, reg_password).addOnCompleteListener((task) -> {
                if(task.isSuccessful()){

                    pBar.setVisibility(View.INVISIBLE);

//                    DocumentReference docRef=fStore.collection("Users").document(Objects.requireNonNull(fAuth.getCurrentUser()).getUid());

                    Map<String,Object> user= new HashMap<>();
                    user.put("Name", reg_name);
                    user.put("Email",reg_email);
                    fStore.collection( collectionPath: "Users").add(user).addOnSuccessListener((OnSuccessListener) (documentReference) -> {
                        Toast.makeText( context: RegistrationPage.this,  text: "User created", Toast.LENGTH_SHORT).show();
                    });

                    Intent loginPage= new Intent( packageContext: RegistrationPage.this, LogInPage.class);
                    startActivity(loginPage);
                    finish();
                }
                else{
                    Toast.makeText( context: RegistrationPage.this,  text: "Error: "+ Objects.requireNonNull(task.getException()).getMessage(), Toast.LENGTH_SHORT).show();
                    pBar.setVisibility(View.INVISIBLE);
                }
            });

        });


    }
```

Fig 8.2.3. Registration Code Snippet

9. PROJET SCREENSHOTS

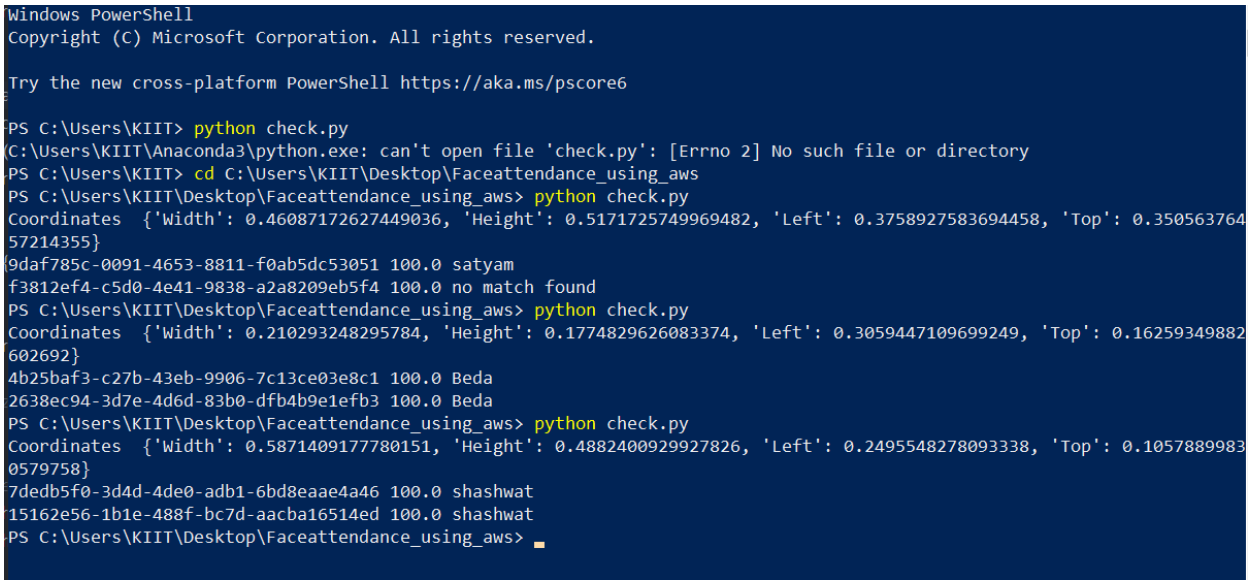


Fig 9.1 Python Code

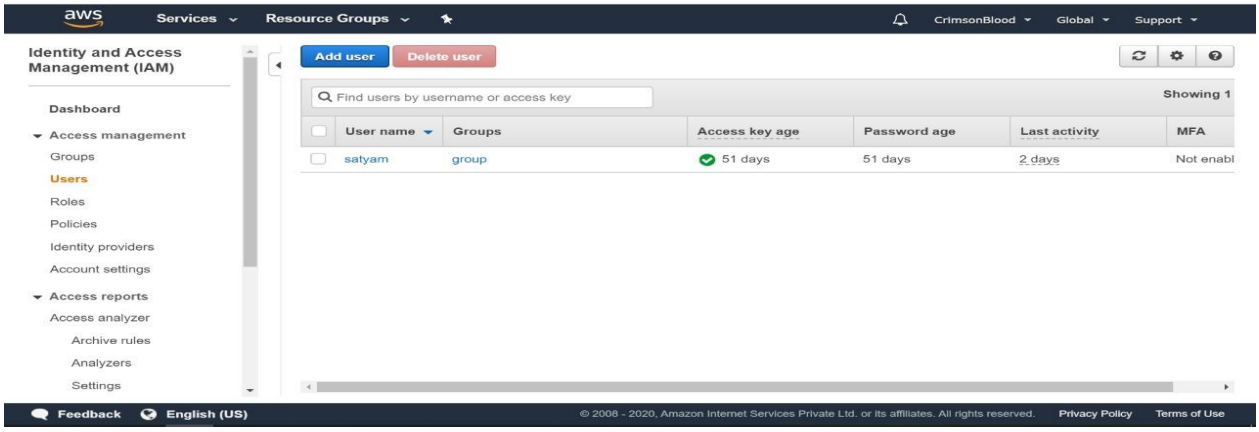


Fig 9.2. AWS Server

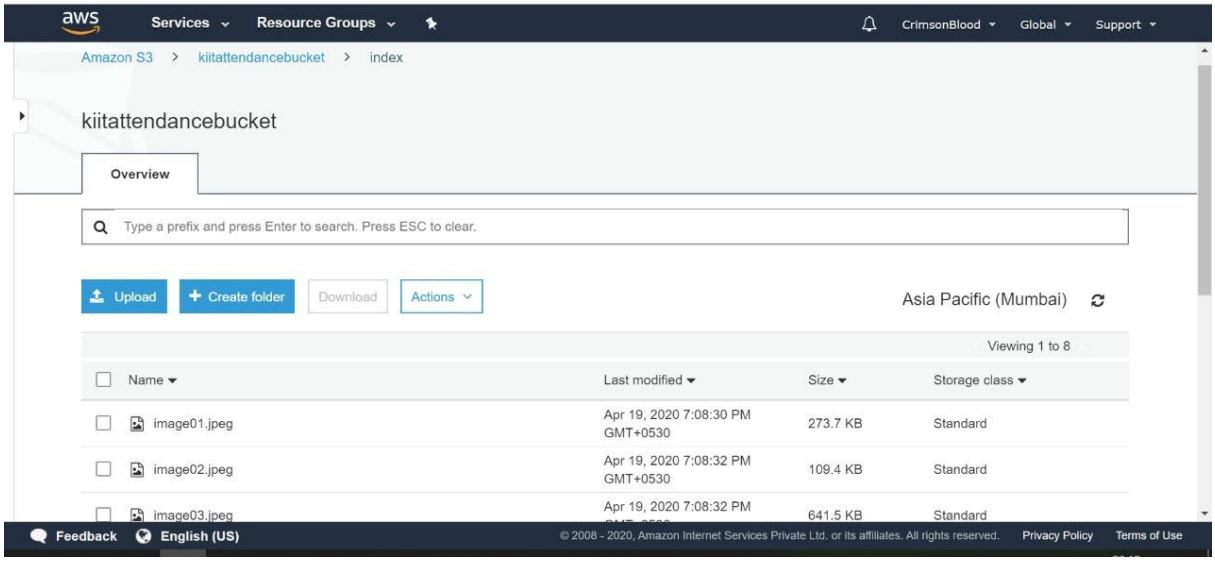


Fig. 9.3 S3 Bucket

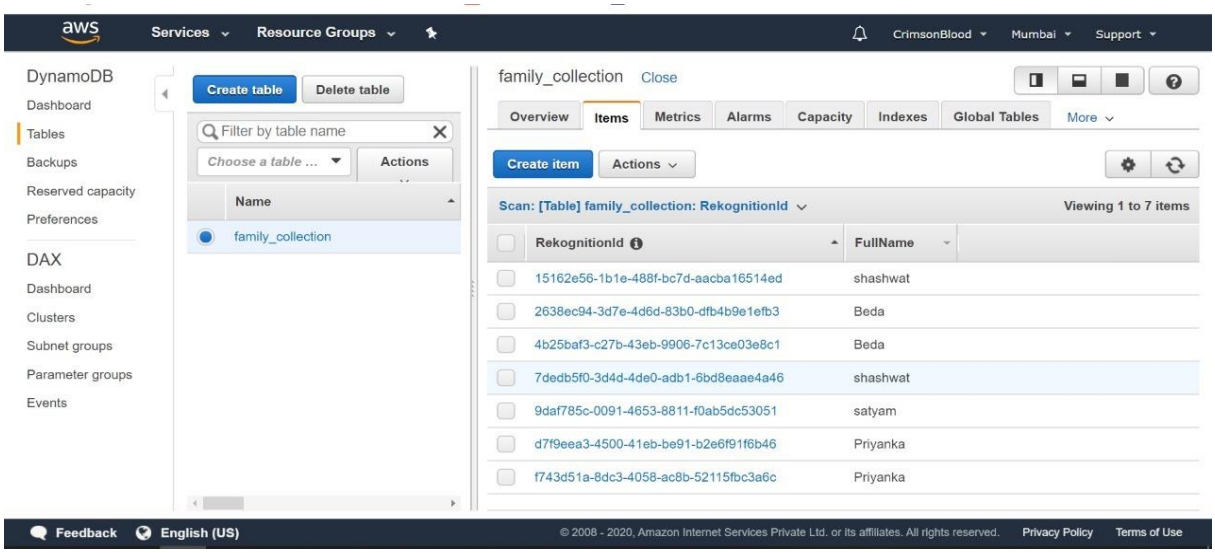


Fig 9.4 Database

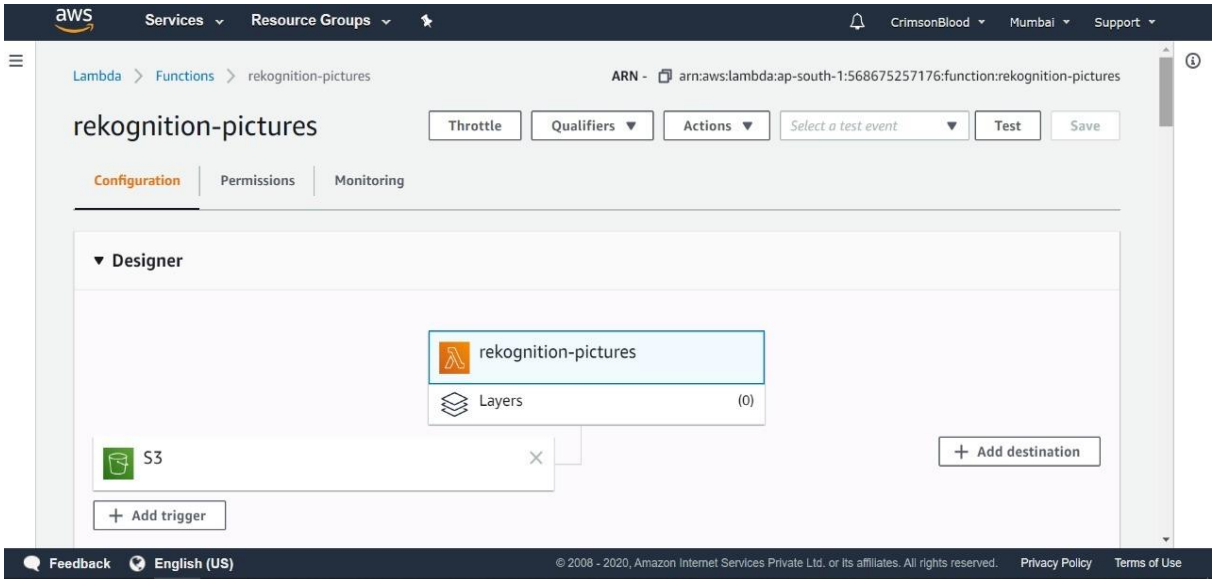


Fig 9.5 AWS Rekognition

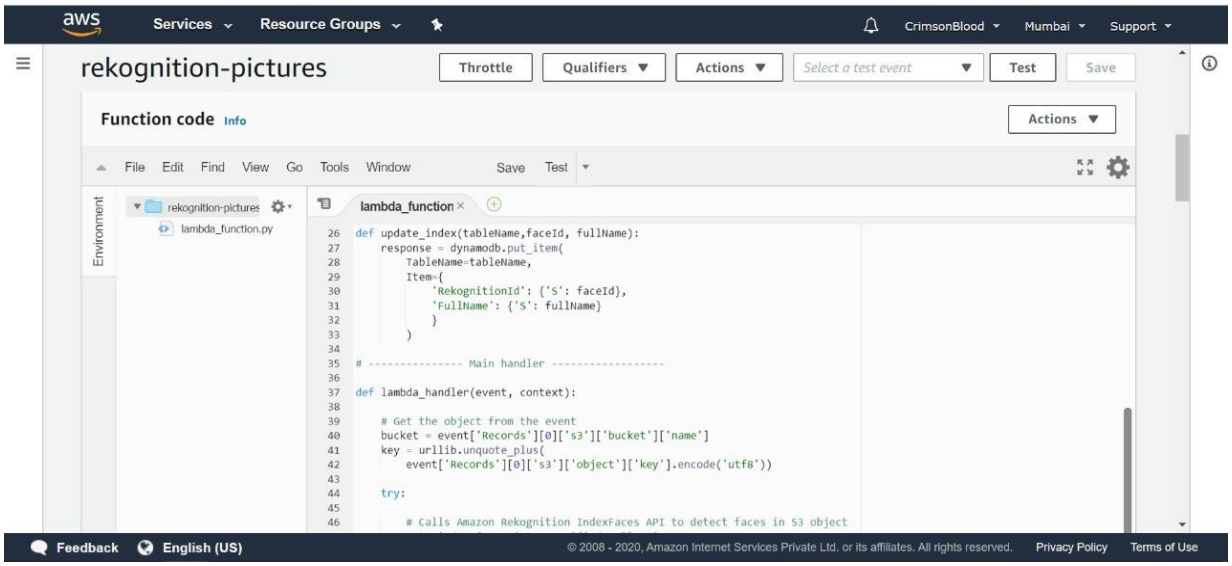


Fig 9.6 Rekognition code

10. CONCLUSION AND FUTURE SCOPE

10.1. CONCLUSION

- This face recognition based attendance management system provides accurate attendance information of the students in easy way and upload the attendance into server using Ethernet cable.
- This system is convenient to user, easy to use and gives better security.
- These systems develop outputs with 88 percent of accuracy.
- When number of student faces increases the accuracy will decreases slightly.
- This system gives the student details as output to a storage device and send message to absent student parent mobile number

10.2. FUTURE SCOPE

- In the current status of the project, individual modules for the face recognition and app development has been done. We will soon integrate these modules together.
- Amazon Rekognition is based on the same proven, highly scalable, deep learning technology developed by Amazon's computer vision scientists to analyze billions of images and videos daily. So, its cutting edge technology will continue to lead in the market with reliability and authenticity.

11. REFERENCES

1. https://docs.aws.amazon.com/rekognition/index.html

2. https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/rekognition.html