**SYNOPSIS**

Attendance management system has become an important software in an organization. The System helps to keeping track of employee’s attendance, login and logout time. But the main problem is the transparency the in system, this creates a trust issue between the organization and its employee. This system is not only for big organization but can be used in schools, colleges, small organizations and office who face the same problem.

To this problem our attendance management system try’s to give the solution by creating an Android Application for the employee to track their attendance, login and logout time every day.

Objectives of the project

* Bring transparency between employee and organization about attendance
* Create a better attendance management system using facial recognition and RFID
* Increase efficiency and effectiveness of attendance management system using better technology
* Create generic attendance management system

The system uses Facial Recognition and RFID technology to mark the attendance of a employee. Both of the technology’s help to create a better fool proof system. Every employee should have a identification card which has RFID embedded and latest photo of the employee is stored in the system for facial recognition. When the employee enter the organization instead of requiring people to place their hand on a reader or precisely position their eye in front of a scanner, face recognition systems takes a picture of employee face as there RFID is authenticated when entered the organization. This information is stored in Firebase which is a cloud based database. Firebase is a product of Google which make the database accessible through the world. Facial recognition analyzes the characteristics of a person's face in images through a digital camera. It measures the overall facial structure, including distances between eyes, nose, mouth, and jaw edges. These measurements are retained in a database and used for comparison while marking attendance. The system create an efficient, effective, cost reduction ,less time consuming attendance management system. This system can be placed at multiple entries and exits of an organization.

**CHAPTER 1**

**INTRODUCTION**

* 1. **PREAMBLE**

Traditional attendance management System helps to keeping track of employee’s attendance, login and logout time, but the main problem is the transparency the in system, this creates a trust issue between the organization and its employee. Traditionally attendance is taken manually by using attendance book and they are monotonous and time consuming. This new system is not only for big organization but can be used in schools, colleges, small organizations and office who face the same problem. Manually recorded attendance can be easily manipulated easy. Keeping record of 1000s of employees in a single book is difficult and it’s difficult to maintain records for longer period of time due to delicate nature of the paper. The system overcome such problems but using cloud based database for maintaining the employee’s attendance records. Employee can use a android app using which he can track his attendance. In traditional system it’s difficult to analyse the data to generate statistics of an employee punctuality because every data is on paper and it should be convert to digital data for analysis. The new systems had a digital database so it’s easy to generate statistic.

To this problem our attendance management system try’s to give the solution by creating an Android Application for the employee to track their attendance, login and logout time every day.

* 1. **LITERATURE REVIEW**

1) Attendance Management System using Face Recognition By- Rohit Chavan, Baburao Phad, Sankalp Sawant, Vinayak Futak, Asha Rawat. Link: http://www.ijirst.org/articles/IJIRSTV1I11060.pdf

Face recognition technology is the least intrusive and fastest bio-metric technology. It works with the most obvious individual identifier the human face. This research aims at providing a system to automatically record the students’ attendance during lecture hours in a hall or room using facial recognition technology instead of the traditional manual methods.

* Better to convert RGB colored photos to Gary scale because it improves accuracy
* Scaling down the captured image can increase efficiency and effectiveness of the system

2) Automatic Attendance Management System Using Face Recognition by- Jomon Joseph, K. P. Zacharia. Link: https://www.ijsr.net/archive/v2i11/MDIwMTM0ODI=.pdf

Recognition is performed by projecting a new image into the subspace spanned by the Eigen faces („face space ‟) and then classifying the face by comparing its position in the face space with the positions of the known individuals. The Eigen face approach gives us efficient way to find this lower dimensional spaceFace has set of characteristic which must be recorded to recognize

* Machine learning helps in identifying individual face accurately
* Using GPU and multiprocessing of CPU face detection works faster

**1.3 PROBLEM STATEMENT**

Keeping track of attendance is getting difficult due to large number of people working in a organization. Using traditional system like attendance book for attendance management has many problems.to overcome those problems new digital attendance management is required which increases the durability of records and create a transparent system and is difficult to manipulate.

* This system will giving access to view attendance, login and logout time using an android application for an employee in organization.

**1.4 METHODOLOGY**

**1.4.1 The Spiral Model**

In 1988, Barry Boehm published a software process model, called the Spiral Model which combines some key aspect of the waterfall model and prototyping methodologies, but provides emphasis in a key area – deliberate alterative risk analysis, particularly suited to large-scale complex systems.

The whole process is presented as a spiral with number of loops. Each loop represents a phase of the software process. Thus, the innermost loop starting at the core of the spiral could represent conceptual development, the next loop with requirements definition, and the next loop with system design and so on.

The Spiral is visualized as a process through a number of iterations, with the four quadrants diagram representing the following activities:

1. Objective Setting: Identify software targets constraints on the process, clarify the project development restrictions, identify project risks and plan alternative strategies.
2. Risk Analysis: A detailed analysis of each identified risk, and to consider how to reduce or eliminate the risk.

**1.5 TECHNICAL FEATURES OF THE PROJECT**

**1.5.1 - PYTHON**

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL).

 Python is a popular for the Internet of things, as new platforms such as Raspberry Pi are based on it. It is wonderful and powerful programming language that's easy to use (easy to read and write) and with Raspberry Pi lets you connect your project to the real world.

**1.5.2 –RASPBERRY PI**

The Raspberry Pi 3 was released in February 2016 with a 64 bit quad core processor, and has on-board Wi-Fi, Bluetooth and USB boot capabilities .The Raspberry Pi 3, with a quad-core processor is described as 10 times the performance of a Raspberry Pi 1.This was suggested to be highly dependent upon task threading and instruction set use.

We here make use of RFID reader and using PI we take input to identify attendance. Web camera is also connected for face recognition and that will be stored to firebase database.

**1.5.3 –ANDROID**

Android is a mobile operating system developed by Google, based on a modified version of the Linux kernel and other open source software and designed primarily for touchscreen mobile devices such as smartphones and tablets. In addition, Google has further developed Android TV for televisions.

We here used android to display employee details for all after authentication. It enable user to view their attendance and thus provide open and transparent environment. We are maintaining single firebase database where we store all details from python and display required using android app.

**CHAPTER 2**

**PROJECT DESCRIPTION**

The attendance management system uses RFID (Radio Frequency Identification) and Facial Recognition technology to create a fool proof system. To create the system Raspberry pi3 is used which feature a Broadcom system on a chip (SoC) with an integrated ARM compatible central processing unit (CPU) and on-chip graphics processing unit (GPU), RFID reader is connected to Raspberry pi to read RFID cards.  RFID 522 sensor is used as a reader. The system also uses Face Recognition for this a high definition we camera is used to capture the face of a employee. Face Recognition works better with proper ambient light, for this multiple led are connected and used while capturing the photo of the employee.

RFID must be bind with the employee in the system, each employee has a employee id which is unique to entire company and every RFID card has a id which is embedded in the RFID card. The system binds the employee id with the card id in a database which helps to identify the employee uniquely. Raspberry pi should be install with the latest version of operating system.  Facial recognition uses a high definition web camera for capturing the face of a employee, some web camera uses device drivers which come with the web camera, we need to install those drivers.

We are using Android app on another side to display attendance to user, and to add holydays , mange users by admin and planed to implement much more service. Android app provides flexibility and easy to view and store attendance . Moreover it creates an open system where there is no ambiguity and accurate. Here we are using calendar to display their attendance based on their Id. This calendar provides a better view where user could easily see his attendance. We are using the real time open source NoSQL database where each employee attendance is scanned and as soon as scanned it will store to database in pre-defined structure. Using that data in our Android app we are retrieving and displaying the attendance to Employee.In future days we can extend this to school and colleges where attendance of student with respective period can be taken easily and thus will save lot of time.

Architecture diagram

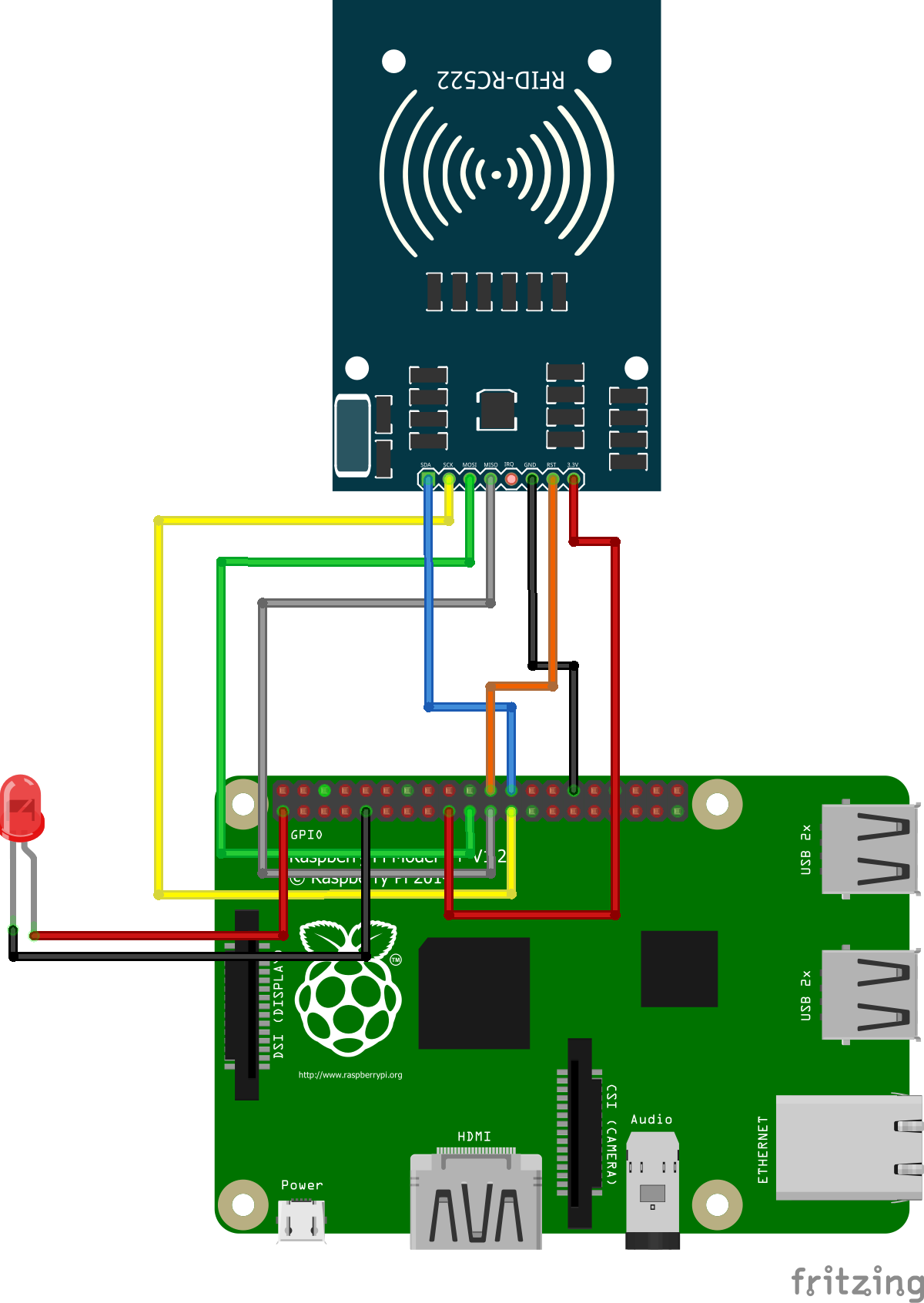


Fig 2.1 Circuit diagram of RFID reader

**CHAPTER 3**

**REQUIREMENT ANALYSIS**

**3.1 FUNCTIONAL REQUIREMENT**

* User Registration
  + Introduction: User should get registered using his RFID tag and face gets captured
  + Input: RFID tag, clear face, Id
  + Processing: All these details gets stored in database for further attendance process
  + Output: User successfully registered
* User Sign Up and Sign In
  + Introduction: Registration and login on android app by user
  + Input: Valid Email, password and their Id
  + Processing: Verify credentials and fetch attendance from database
  + Output: Display user attendance with appropriate message
* Admin Sign Up/Sign In
  + Introduction: Entry for admin to add/delete and maintain details
  + Input: Username and password
  + Processing: Verify login credentials and allow him to do changes
  + Output: Update database values
* Mark Employee Attendance
  + Introduction: Mark user attendance as present or absent
  + Input: RFID and clear Face
  + Processing: Validation of user and update attendance on database
  + Output: Alert message stating success

**3.2 NON-FUNCTIONAL REQUIREMENT**

* Availability: This project can be used anywhere and can run on any systems required the specifications of hardware and software should meet. And anyone can register to this system.
* Integrity: Here each module is linked with another module and they are made as one whole package. And small modules are integrated and made as one whole system.
* Performance requirements: This project is much more responsive compared to other systems. Each module should take less time to open the page and calculation of scores should be done rapidly. Displaying of pages should not take more time.
* Portability: This system can be used on any platform and can be made to run on any systems.
* Reliability: This System is made reliable by performing test cases on each module so that no error should occur.
* Scalability: This system is made such that it will reach users requirements and demands.
* Security: Security is provided to each and every user who uses this system by providing him username and password so that user details and transactions will be kept secured and protected.
* Usability: Test cases are applied to each and every module to know the usage of each module and the whole system.

**CHAPTER 4**

**SOFTWARE REQUIREMENT SPECIFICATION**

**4.1 HARDWARE SPECIFICATION**

Table 3.1: Hardware Specification

|  |  |
| --- | --- |
| **SYSTEM** | **TOOL** |
| Processor | Intel® Core™ i3-4005U CPU @1.60GHz |
| Memory | 6.00 GB RAM |
| Hard Disk | 4 GB |
| USB Camera | Minimum 10MP |
| RFID Reader | RC255 RFID Reader |

|  |  |
| --- | --- |
| **SYSTEM** | **TOOL** |
| Operating System | Linux , Android |
| Front End | XML |
| Back End | Python, Android |
| Database | Firebase |
| Framework | Android Studio |
| Face Recognition | face\_recognition, DLIB |

**4.2 SOFTWARE SPECIFICATION**

Table 3.2: Software Specification

**CHAPTER 5**

**ANALYSIS & DESIGN**

**5.1 ARCHITECTURAL DESIGN**

**5.1.1 BLOCK DIAGRAM**

Generate report

Delete user

Add leaves

Remove leaves

Logout

Select date

Calendar

logout

valid

register

login

Existing user

New user

user

employee

admin

Fig 4.1: Block diagram for calendar app

RECOGNIZE FACE

FIREBASE

IDENTIFY RFID

RECOGNIZE FACE

ADD PHOTOGRAPH

ADD RFID

IDENTIFY USER

USER

Fig 4.2Block diagram for identifying user

**5.2 SYSTEM DESIGN**

**5.2.1 CLASS DIAGRAM**

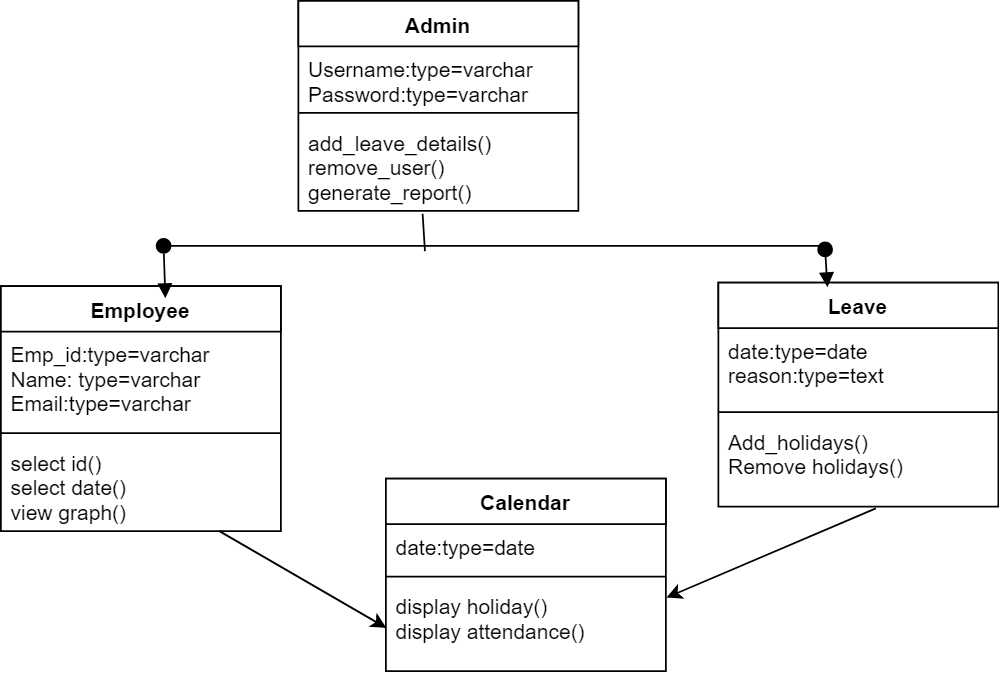


Fig 4.3 Class diagram of calendar app

**5.2.2 USE CASE DIAGRAM**

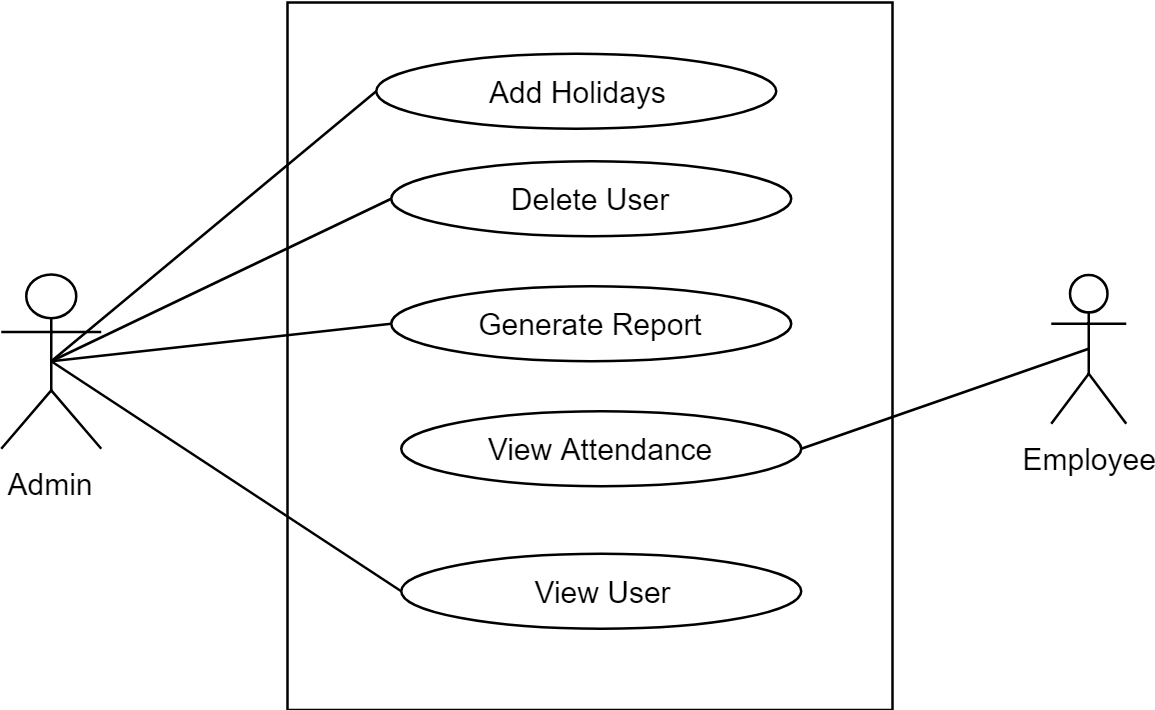
****

Fig 4.4 Use Case Diagram

**5.2.5 TABLE DESIGN**

**Employee Attendance**

Table 4.1: Table design of Employee attendance

|  |  |  |
| --- | --- | --- |
| **ATTRIBUTE** | **DATA TYPE** | **CONSTRAINTS** |
| Employee id | Text | Not null |
| Year | String | Valid year |
| Month | String | Valid month |
| Day | String | Valid day |
| Check in/Check out | String | Valid time |

**Employee**

Table 4.2: Table design of Employee

|  |  |  |
| --- | --- | --- |
| **ATTRIBUTE** | **DATA TYPE** | **CONSTRAINTS** |
| Employee id | String | Not null |
| Name | String | Not null |
| Email | String | Valid Email |

**Holidays**

Table 4.3: Table design of Holiday

|  |  |  |
| --- | --- | --- |
| **ATTRIBUTE** | **DATA TYPE** | **CONSTRAINTS** |
| Year | String | Valid year |
| Month | String | Valid month |
| Day | String | Valid day |

**CHAPTER 6**

**IMPLEMENTATION**

**6.1 CODING**

**Python Raspberry pi**

import face\_recognition

import time

import cv2

import RPi.GPIO as GPIO

import SimpleMFRC522

import blink as blink

import datetime

from firebase import firebase

class f\_rec():

def \_\_init\_\_(self):

self.firebase\_obj=firebase.FirebaseApplication('<https://calenderminiproject.firebaseio.com/>', None)

with open('imageencoding.txt','r') as file:

for line in file:

line=line.replace("\n","")

print(line)

if line!='':

setattr(self,str(line),face\_recognition.load\_image\_file("./img/"+line+".jpg"))

setattr(self,str(line),face\_recognition.face\_encodings(getattr(self,str(line)))[0])

self.reader = SimpleMFRC522.SimpleMFRC522()

print("finish encoding")

def make\_img\_chk(self):

self.getname=self.text+"\_encoding"

print(self.getname)

self.getimg=face\_recognition.load\_image\_file('capture.jpg')

self.src\_encoding = face\_recognition.face\_encodings(self.getimg)

if len(self.src\_encoding)>0:

self.src\_encoding=self.src\_encoding[0]

return True

else:

return False

def find(self):

now = datetime.datetime.now()

obj=getattr(self,self.getname)

self.results = face\_recognition.compare\_faces([obj],self.src\_encoding,tolerance=0.50)

if (self.results):

path="/students/"+self.text+"/"+(str)(now.year)+"/"+(str)(now.month)+"/"+(str)(now.day)

self.firebase\_obj.put(path,"present","1")

self.firebase\_obj.put(path,"come\_in",(str)(now.hour)+":"+(str)(now.minute))

def clean(self):

del(self.getname)

del(self.src\_encoding)

del(self.text)

GPIO.cleanup()

def capture(self):

self.cap = cv2.VideoCapture(0)

input("enter to take photo”)

blink.turn\_onled()

while True:

time.sleep(1)

ret, frame = self.cap.read()

if ret is False:

continue

cv2.imshow('frame',frame)

cv2.imwrite('./capture.jpg', frame)

cv2.destroyAllWindows()

self.cap.release()

blink.turn\_offled()

return True

def getrfid(self):

print("place rfid")

while True:

try:

id, text = self.reader.read()

self.text=text.replace(" ","")

return True

except:

print("rfid not found")

return False

finally:

GPIO.cleanup()

def addrfid(self):

try:

text = input('enter register no:')

print("Now place your tag to write")

self.reader.write(text)

print("Written")

finally:

GPIO.cleanup()

def addphoto(self):

name=input("enter register no")

self.cap = cv2.VideoCapture(0)

while(True):

# Capture frame-by-frame

ret, frame = self.cap.read()

# Our operations on the frame come here

if ret is True:

cv2.imshow('frame',frame)

# gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2BGRA)

else:

continue

# Display the resulting frame

# cv2.imshow('frame',frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

cv2.imwrite("./img/"+name+'\_encoding.jpg', frame)

break

# When everything done, release the capture

self.cap.release()

cv2.destroyAllWindows()

name=name.replace(" ","")

if(self.addencoding(name)):

with open('imageencoding.txt','a') as file:

file.write("\n"+name+'\_encoding')

return True

return False

def addencoding(self,name):

temp=face\_recognition.load\_image\_file("./img/"+name+"\_encoding.jpg")

temp=face\_recognition.face\_encodings(temp)

if len(temp)>0:

setattr(self,str(name+'\_encoding'),(temp)[0])

return True

else:

return False

st=time.time()

p1=f\_rec()

sp=time.time()

print("time to encode-",sp-st)

while True:

ch=input("1.add tag\n2.add Photo\n3.run program")

if ch=="1":

p1.addrfid()

elif ch=="2":

chk=p1.addphoto()

while not chk:

chk=p1.addphoto()

elif ch=="3":

chk=p1.getrfid()

while not chk:

chk=p1.getrfid()

blink.blink\_time(0.1)

p1.capture()

chk=p1.make\_img\_chk()

while not chk:

#continue

print("face not found or multiple faces found")

p1.capture()

chk=p1.make\_img\_chk()

st=time.time()

p1.find()

sp=time.time()

print(p1.results,"-",sp-st)

p1.clean()

Android

**package** com.example.vageesh.calenderminiproject;  
 **import** android.content.Intent;  
 **import** android.graphics.Color;  
 **import** android.os.AsyncTask;  
 **import** android.support.annotation.NonNull;  
 **import** android.support.v4.app.FragmentTransaction;  
 **import** android.support.v4.content.ContextCompat;  
 **import** android.support.v7.app.AppCompatActivity;  
 **import** android.os.Bundle;  
  
**public class** DisplayBasedOnMonth **extends** AppCompatActivity {  
  
 DatabaseReference **myRefToStudents**,**myref\_to\_email**,**myRefToHolyday**;  
 FirebaseUser **currentFirebaseUser**;  
 FirebaseDatabase **database1**;  
  
 MaterialCalendarView **mv**;  
 **int month**[]=**new int**[12];  
 **int year**;  
 **int days**[]=**new int**[30];  
 **int absentdays**[]=**new int**[30];  
 **int holydays**[]=**new int**[30];  
 **int max\_days**;  
 Intent **intent1**;  
 String **usn**,**come\_in**,**go\_out**;  
  
 **int globalcolor**;  
 ProgressBar **pbar**;  
  
 @Override  
 **protected void** onStart() {  
 **super**.onStart();  
 *// To display total number of days,attended dyas* }  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_display\_based\_on\_month***);take\_xml\_view();  
 **new** AsyncCaller().execute();  
  
 }  
  
 **private void** take\_xml\_view() {  
 **pbar**=(ProgressBar)findViewById(R.id.***progressBar2***);  
 }  
  
 **private class** AsyncCaller **extends** AsyncTask<Void, Void, Void> {  
 @Override  
 **protected void** onPreExecute() {  
 **super**.onPreExecute();  
  
 database\_initialization();  
 get\_month\_year\_values();  
  
 }  
  
 @Override  
 **protected** Void doInBackground(Void... params) {find\_max\_days();  
  
 **return null**;  
 }  
  
 }  
  
 **private void** fetch\_values\_from\_database() {myRefToHolyday.child(String.valueOf(year)).child(String.valueOf(month[0])).addListenerForSingleValueEvent(**new** ValueEventListener() {  
  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
  
 }  
 });  
  
  
 }  
  
 **private void** check\_for\_present() {  
 myRefToStudents.child(usn).child(String.valueOf(year)).addListenerForSingleValueEvent(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **if**(dataSnapshot.hasChild(String.valueOf(month[0]))) {  
  
 }  
 **else** {  
  
 startActivity(**new** Intent(DisplayBasedOnMonth.**this**,NoAttendance.**class**));  
 }  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
  
 }  
 });  
 }  
  
 **private void** database\_initialization() {  
  
 database1 = FirebaseDatabase.getInstance();  
 myRefToStudents = database1.getReference(**"students"**);  
 myref\_to\_email=database1.getReference(**"email\_usn"**);  
 }  
  
 **private void** add\_markers\_on\_calendar() {  
  
 **int** days\_to\_display;  
 **int** number\_of\_days=absentdays.length;  
 **int** number\_of\_holydays=holydays.length;  
  
 *//Retrieve current date and stop adding marker on claender up to this.* **final** String curent\_date = **new** SimpleDateFormat(**"dd"**, Locale.getDefault()).format(**new** Date());  
 **final** String curent\_month = **new** SimpleDateFormat(**"MM"**, Locale.getDefault()).format(**new** Date());  
  
 **if**(month[0]<(Integer.parseInt(curent\_month))) {  
 days\_to\_display=number\_of\_days;  
 }  
 **else** {  
 days\_to\_display=Integer.parseInt(curent\_date);  
 }  
  
 pbar.setVisibility(View.INVISIBLE);  
 mv = (MaterialCalendarView)findViewById(R.id.calendarView);  
 mv.setOnDateChangedListener(**new** OnDateSelectedListener() {  
 @Override  
 **public void** onDateSelected(@NonNull MaterialCalendarView widget, @NonNull CalendarDay date, **boolean** selected) {  
 Toast.makeText(getApplicationContext(),**"selected"**,Toast.LENGTH\_SHORT).show();  
  
 **int** month\_click=date.getMonth();  
 **int** day\_click=date.getDay();  
 **int** year\_click=date.getYear();  
  
 **if**(month\_click==Integer.parseInt(curent\_month) && day\_click>Integer.parseInt(curent\_date)) {  
 fragment\_future\_date fudt = **new** fragment\_future\_date();  
 FragmentTransaction ft = getSupportFragmentManager().beginTransaction();  
 ft.replace(R.id.display, fudt);  
 ft.addToBackStack(**null**);  
 ft.commit();  
 } **else** {  
 Log.i(**"Selected"**, String.valueOf(month\_click + day\_click + year\_click));  
  
 show\_come\_in\_out\_time(day\_click, month\_click, year\_click);  
 }  
 }  
 });  
 mv.state().edit()  
 .setFirstDayOfWeek(Calendar.SUNDAY)  
 .setMinimumDate(CalendarDay.from(year, month[0], 1))  
 .setMaximumDate(CalendarDay.from(year, month[0], max\_days))  
 .setCalendarDisplayMode(CalendarMode.MONTHS)  
 .commit();  
  
  
 }  
  
 **private** HashSet<CalendarDay> getCalendarDaysSet(Calendar cal1) {  
 HashSet<CalendarDay> setDays = **new** HashSet<>();  
  
 **int** i;  
 *// while (cal1.getTime().before(cal2.getTime())) {* CalendarDay calDay = CalendarDay.from(cal1);  
 setDays.add(calDay);  
 *//cal1.add(Calendar.DATE, 1);  
 //}* **return** setDays;  
 }  
 **private class** BookingDecorator **implements** DayViewDecorator {  
 **private int** mColor;  
 **private** HashSet<CalendarDay> mCalendarDayCollection;  
  
 }  
  
 @Override  
 **public void** decorate(DayViewFacade view) {  
 view.addSpan(**new** ForegroundColorSpan(mColor));  
 *//view.addSpan(new BackgroundColorSpan(mColor));* view.setBackgroundDrawable(ContextCompat.getDrawable(getApplicationContext(),mColor));  
  
 }  
}

**XML FILE**

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:layout\_margin="10dp"  
 tools:context="com.example.vageesh.calenderminiproject.AdminPage"**>  
  
  
 <**TextView  
 android:id="@+id/textView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="10dp"  
 android:textSize="20dp"  
 android:text="@string/admin\_entry\_msg"  
 android:textAlignment="center"  
 android:textAllCaps="true"  
 android:textStyle="bold"** />  
  
 <**FrameLayout  
 android:id="@+id/framelayout1"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"**>  
  
 <**Button  
 android:id="@+id/button2"  
 android:layout\_marginTop="30dp"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="ADD LEAVE DETAILS"  
 android:onClick="add\_leave\_details"  
 tools:layout\_editor\_absoluteX="122dp"  
 tools:layout\_editor\_absoluteY="61dp"** />  
  
 </**FrameLayout**>  
</**LinearLayout**>

**6.2 Screen Shots**

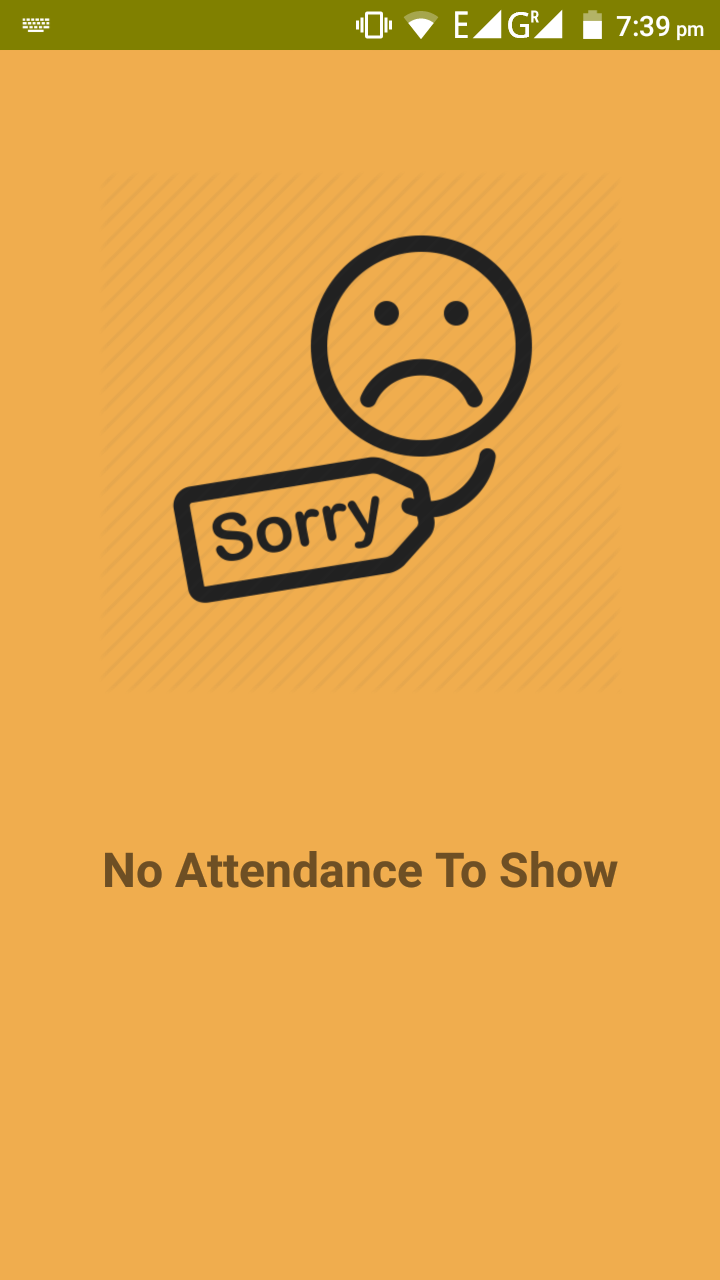
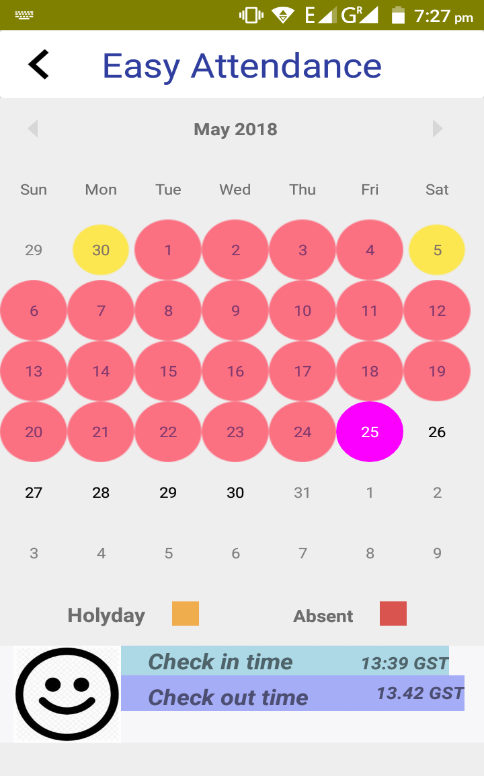
 

Fig 5.1: Check in/out time screen Fig 5.2: No attendance screen

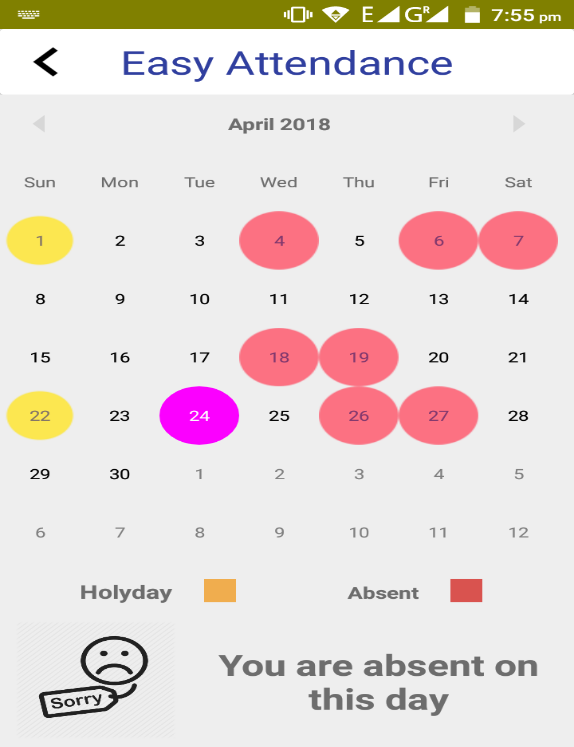
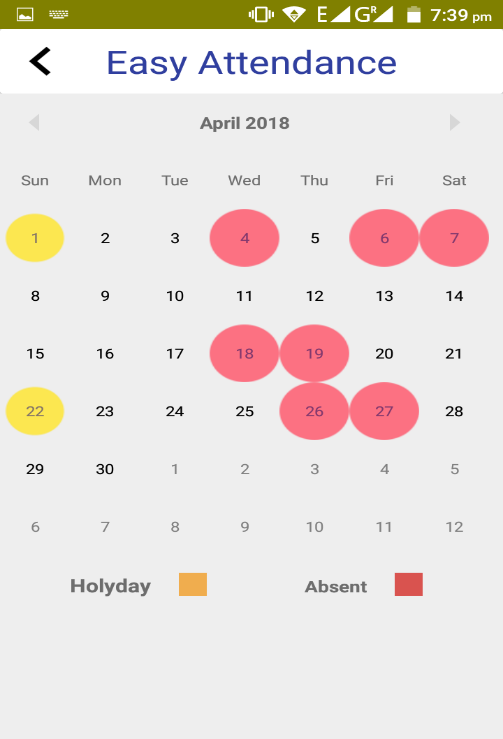


Fig 5.3: Attendance display screen Fig 5.4: Showing absent to user

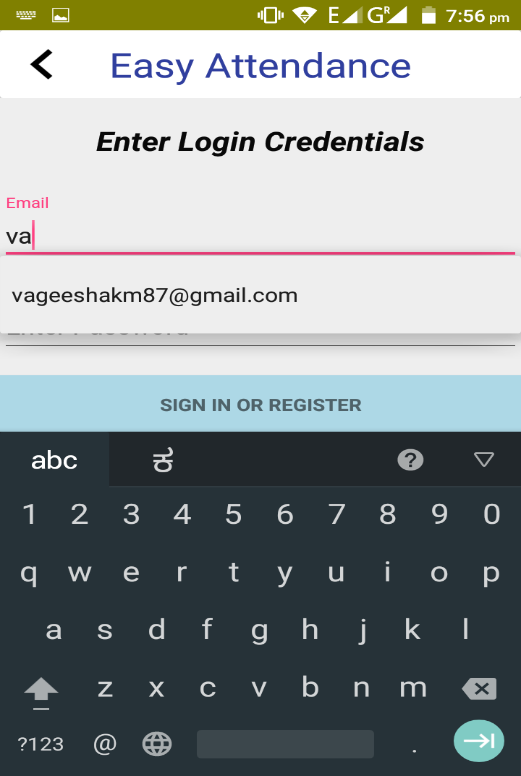
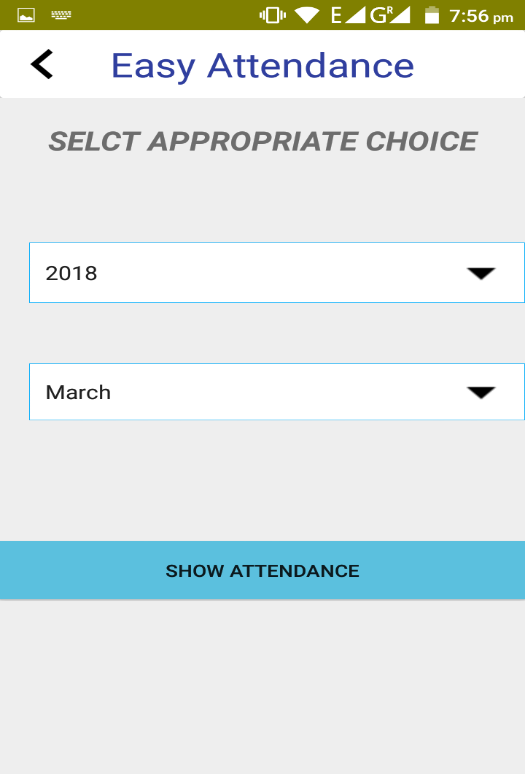


Fig 5.5: Login Screen Fig 5.6: Select date screen

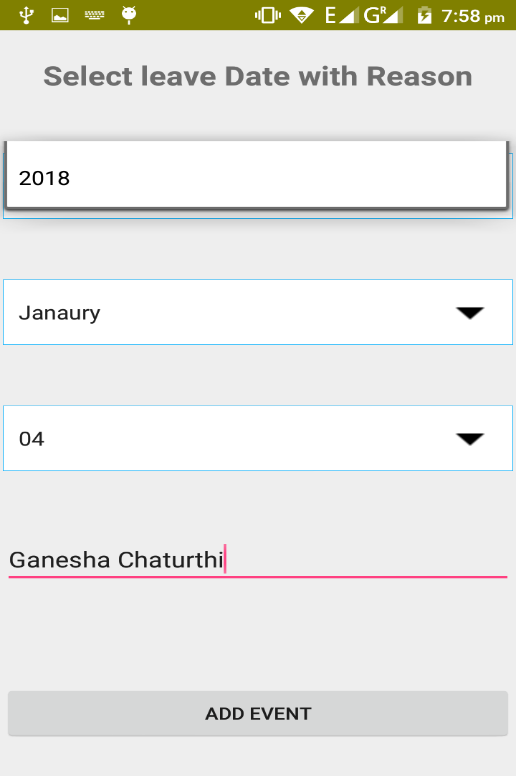


Fig 5.7: Update holidays screen with reason

**CHAPTER 7**

**TESTING**

**7.1 TESTING CASES**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test case | Description | Input | Expected output | Actual output | Remarks |
| Login | User login authentication | * Correct Username and password * Incorrect Username or password | Successful login  Unsuccessful  login | Successful login | Pass |
| RFID tag | Verify RFID id | * RFID tag id | Id matched  Id and Face did not match | Successful marking of attendance | Pass |
| User and admin Registration | Entry of all the fields of the users | * All fields filled correctly * Empty fields | Successful entry  All fields required | Successful entry | Pass |
| Select month and year | Display attendance of user for selected month | * Select Present Date * Select Date that not present | Display Attendance  Display Error Message | Display attendance on calendar and messages | Pass |
| Capture Face | for marking with database face | * Face photo | Face matched  Face Not Matched | Face Matched  Face Not Matched | Pass |

Table 6.1: Table for Testing Cases

**CHAPTER 8**

**CONCLUSION**

**8.1 CONCLUSION AND FUTURE ENCHANCEMENT**

We think not a single project is ever consider as complete forever because with the growing industries and to sustain in that there is always a necessary to be updated with the currently industry.

In this project, we are taking attendance of employee automatically and thus saving time and is more accurate. This system is open where every employee can see his attendance.

In future we can extend this to college students where attendance of each period is taken and saves huge time and gives easy maintenance of records and permanent storage

**8.2 REFERENCES**

[1].Attendance Management System using Face Recognition By- Rohit Chavan

[2].Automatic Attendance Management System Using Face Recognition By- Jomon Joseph, K. P. Zachariah. Link: <https://www.ijsr.net/archive/v2i11/MDIwMTM0ODI=.pdf>

[3]. https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

[4]. “Firebase” documentation:firebase.console.com

[6]. “Android”, Website: https://developper.android.com

[7]. “GitHub”, Website: <https://github.com/ageitgey/face_recognition>

[8]. “Opencv”, Website:  <http://www.face-rec.org/general-info/>

[9]. “Raspberry pi”, Website:  <https://pimylifeup.com/raspberry-pi-rfid-rc522/>.