**SYNOPSIS**

It is a web application that generates the water bill automatically at regular interval of time. The generated water bill includes amount to be paid according to the usage of water by specific user. The web portal will allow users to monitor there house or building water consumption at every minute and also helps them to keep track of previous water consumption details. system is user friendly that enables the user to clearly understand the system and helps him for easy payment. It reduces the manual work of generating bill using paper and also found to be echo-friendly. Since nowadays internet plays a major role, this system helps in easy and quick access using internet and thus enables the user community to work smart using new technology.

Objectives of the project

* Reduces manual work
* Minimizes the entry of incorrect data.
* Bill can be generated automatically on monthly basis
* Generation of result is fast and accurate
* The open system enables user to view his daily usage
* Minimizes the work of employee while generating bill

The aim of water bill generation system is to generate the water bill regularly on monthly basis automatically. User will login to his account based on unique id provided to view his bill and can pay the bill through online. This is echo friendly and also helps the water bill generating department to maintain all the records easily using database. The process is faster and easy to use for both employee and also user.

**CHAPTER 1**

**INTRODUCTION**

* 1. **PREAMBLE**

It is a web application that generates the water bill automatically at regular interval of time. The generated water bill includes amount to be paid according to the usage of water by specific user. The web portal will allow users to monitor their house or building water consumption at every minute and also helps them to keep track of previous water consumption details. System is user friendly that enables the user to clearly understand the system and helps him for easy payment. It reduces the manual work of generating bill using paper and also found to be echo-friendly. Since nowadays internet plays a major role, this system helps in easy and quick access using internet and thus enables the user community to work smart using new technology. 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. The aim of water bill generation system is to generate the water bill regularly on monthly basis automatically. User will login to his account based on unique id provided to view his bill and can pay the bill through online. This is echo friendly and also helps the water bill generating department to maintain all the records easily using database. The process is faster and easy to use for both employee and also user.

* 1. **LITERATURE REVIEW**

1) Smart water meters By- Australian water association. Link:https://stopsmartmetersau.files.wordpress.com/2013/12/smart-water-meters.pdf

A key issue for utilities and consumers is the frequency and temporal spread of water meter reads. Most utilities record water consumption data manually on a monthly, quarterly or half-yearly basis. While monthly data provides better data set for high levels of water usage, quarterly or half-yearly data collection provides a ‘lumpy’ dataset in which a whole year of water consumption is lumped into only two or four sets. This infrequent data collection is sufficient for billing purposes, but gives limited information on actual water use behavior, leakage and seasonal variation. The timely collection and analysis of water use data, and the timely relaying of these data to the water user, can result in significant changes in water use behavior. The benefits include immediate leak detection and consequent remedial action that can save precious quantities of water. The data is also invaluable in designing water efficiency and reuse systems (Butler, 2007) and for the improvement of demand management policies and programs (Giurco et al., 2008). Smart meters are one step closer to bring this dilemma into real-time monitoring of water use, with the added benefit of letting the users know where they use the water most in a dwelling, for example, shower or bath

2) 2. Intelligent Metering for Urban Water Link: www.mdpi.com/journal/water

Metering Technologies There are a range of metering technologies, employing different principles to capture and record water use Common technologies broadly fall within one of four categories:

* displacement meters;
* velocity meters;
* compound (or combination)
* Electromagnetic meters.

Displacement (or mechanical) meters require the movement of water to mechanically displace components within the meter to record water flow. They have the advantage of being inexpensive and accurate at low to moderate flow rates. Velocity meters, such as multi-jet, magnetic flow and ultrasonic measure the velocity of flow through a meter of a known internal capacity. Speed of flow is then converted into volume of flow for usage. Combination meters utilize the strengths of displacement and velocity technologies in the one meter, in cases of variable flow rates (i.e., both high and low flow rates need to be recorded accurately). Electromagnetic flow meters harness the electromagnetic properties of water, which, as it flows, generates voltage as it crosses the force lines of a magnetic field. With no moving parts, these meters are well-suited to flows with contaminants or debris that could otherwise damage a mechanical meter.

**1.3 PROBLEM STATEMENT**

Keeping track of water consumption is getting difficult due to large number of people have access to water. Using traditional system like manual bill generation for billing management has many problems.to overcome those problems new smart water meter is required which increases the durability of records and create a transparent system and is difficult to manipulate.

**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 - NODEMCU**

NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module.The term "NodeMCU" by default refers to the firmware rather than the development kits. The firmware uses the Lua scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson and spiffs.

NodeMCU was created shortly after the ESP8266 came out. On December 30, 2013, Espressif Systems began production of the ESP8266. The ESP8266 is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core, widely used in IoT applications NodeMCU started on 13 Oct 2014, when Hong committed the first file of NodeMCU-firmware to GitHub.

As Arduino.cc began developing new MCU boards based on non-AVR processors like the ARM/SAM MCU and used in the Arduino Due, they needed to modify the Arduino IDE so that it would be relatively easy to change the IDE to support alternate tool chains to allow Arduino C/C++ to be compiled down to these new processors. They did this with the introduction of the Board Manager and the SAM Core. A "core" is the collection of software components required by the Board Manager and the Arduino IDE to compile an Arduino C/C++ source file down to the target MCU's machine language. Some creative ESP8266 enthusiasts have developed an Arduino core for the ESP8266 Wi-Fi SoC that is available at the GitHub ESP8266 Core webpage. This is what is popularly called the "ESP8266 Core for the Arduino IDE" and it has become one of the leading software development platforms for the various ESP8266 based modules and development boards, including NodeMCU.

**1.5.2 –NODE JS**

Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser. Typically, JavaScript is used primarily for client-side scripting, in which scripts written in JavaScript are embedded in a webpage's HTML and run client-side by a JavaScript engine in the user's web browser. Node.js lets developers use JavaScript to write Command Line tools and for server-side scripting running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web application development around a single programming language, rather than different languages for server side and client side scripts.Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in web applications with many input/output operations, as well as for real-time Web applications

Node.js allows the creation of Web servers and networking tools using JavaScript and a collection of "modules" that handle various core functionality Modules are provided for file system I/O, networking, binary data, cryptography functions, data streams, and other core functions Node.js's modules use an API designed to reduce the complexity of writing server applications.

Node.js is primarily used to build network programs such as Web servers. The biggest difference between Node.js and PHP is that most functions in PHP block until completion, while Node.js functions are non-blocking.

Node.js operates on a single thread event loop, using non-blocking I/O calls, allowing it to support tens of thousands of concurrent connections without incurring the cost of thread context switching. The design of sharing a single thread among all the requests that use the observer pattern is intended for building highly concurrent applications, where any function performing I/O must use a callback. To accommodate the single-threaded event loop, Node.js uses the libuv library—which, in turn, uses a fixed-sized thread pool that handles some of the non-blocking asynchronous I/O operations.

**1.5.3 –ANGULAR JS**

AngularJS is a JavaScript-based open-source front-end web application framework mainly maintained by Google and by a community of individuals and corporations to address many of the challenges encountered in developing single-page applications. The JavaScript components complement Apache Cordova, a framework used for developing cross-platform mobile apps. It aims to simplify both the development and the testing of such applications by providing a framework for client-side model–view–controller (MVC) and model–view–view model architectures, along with components commonly used in rich Internet applications. The AngularJS framework works by first reading the HTML page, which has additional custom tag attributes embedded into it. Angular interprets those attributes as directives to bind input or output parts of the page to a model that is represented by standard JavaScript variables. The values of those JavaScript variables can be manually set within the code, or retrieved from static or dynamic JSON resources.

AngularJS is built on the belief that declarative programming should be used to create user interfaces and connect software components, while imperative programming is better suited to defining an application's business logic. The framework adapts and extends traditional HTML to present dynamic content through two-way data-binding that allows for the automatic synchronization of models and views. As a result, AngularJS de-emphasizes explicit DOM manipulation with the goal of improving testability and performance.

### 1.5.4 – YF S201 Hall Effect Water Flow Meter / SENSOR

This sensor sits in line with your water line and contains a pinwheel sensor to measure how much liquid has moved through it. There's an integrated magnetic Hall Effect sensor that outputs an electrical pulse with every revolution. The Hall Effect sensor is sealed from the water pipe and allows the sensor to stay safe and dry.

The sensor comes with three wires: red (5-24VDC power), black (ground) and yellow (Hall Effect pulse output). By counting the pulses from the output of the sensor, you can easily calculate water flow. Each pulse is approximately 2.25 milliliters. Note this isn't a precision sensor, and the pulse rate does vary a bit depending on the flow rate, fluid pressure and sensor orientation. It will need careful calibration if better than 10% precision is required. The pulse signal is a simple square wave so it’s quite easy to log and convert into liters per minute using the following formula.

Pulse frequency (Hz) / 7.5 = flow rate in L/min.

**Features:**

* Sensor Type: Hall effect
* Working Voltage: 5 to 18V DC (min tested working voltage 4.5V)
* Max current draw: 15mA @ 5V
* Output Type: 5V TTL
* Working Flow Rate: 1 to 30 Liters/Minute
* Working Temperature range: -25 to +80℃
* Working Humidity Range: 35%-80% RH
* Accuracy: ±10%
* Maximum water pressure: 2.0 MPa
* Output duty cycle: 50% +-10%
* Output rise time: 0.04us
* Output fall time: 0.18us
* Flow rate pulse characteristics: Frequency (Hz) = 7.5 \* Flow rate (L/min)
* Pulses per Liter: 450
* Durability: minimum 300,000 cycles
* Cable length: 15cm
* 1/2" nominal pipe connections, 0.78" outer diameter, 1/2" of thread
* Size: 2.5" x 1.4" x 1.4"

**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

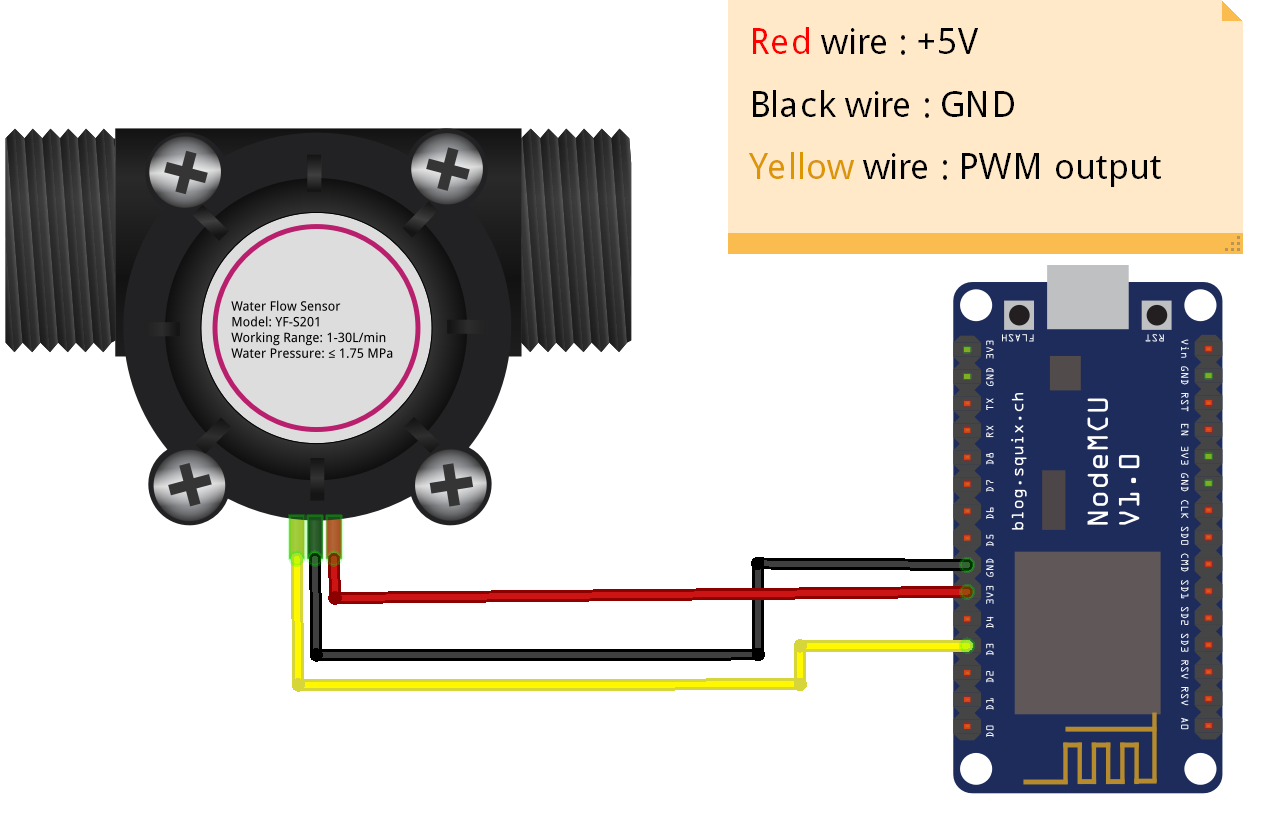


Fig 2.1 Circuit diagram of Water Flow and NodeMcu

**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 4.1: Server Hardware Specification

|  |  |
| --- | --- |
| **SYSTEM** | **TOOL** |
| Processor | Intel® Core™ i3-4005U CPU @1.60GHz |
| Memory | 6.00 GB RAM |
| Hard Disk | 4 GB |

Table 4.2: Water Meter Hardware Specification

|  |  |
| --- | --- |
| **SYSTEM** | **TOOL** |
| Sensor | YF-S201 Hall Effect Water Flow Meter |

|  |  |
| --- | --- |
| **SYSTEM** | **TOOL** |
| Operating System | Linux , Android |
| Front End | HTML |
| Back End | Mean Stack |
| Database | Mlab Mongo dB |
| IDE | Arduino IDE |

**4.2 SOFTWARE SPECIFICATION**

Table 4.3: 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**

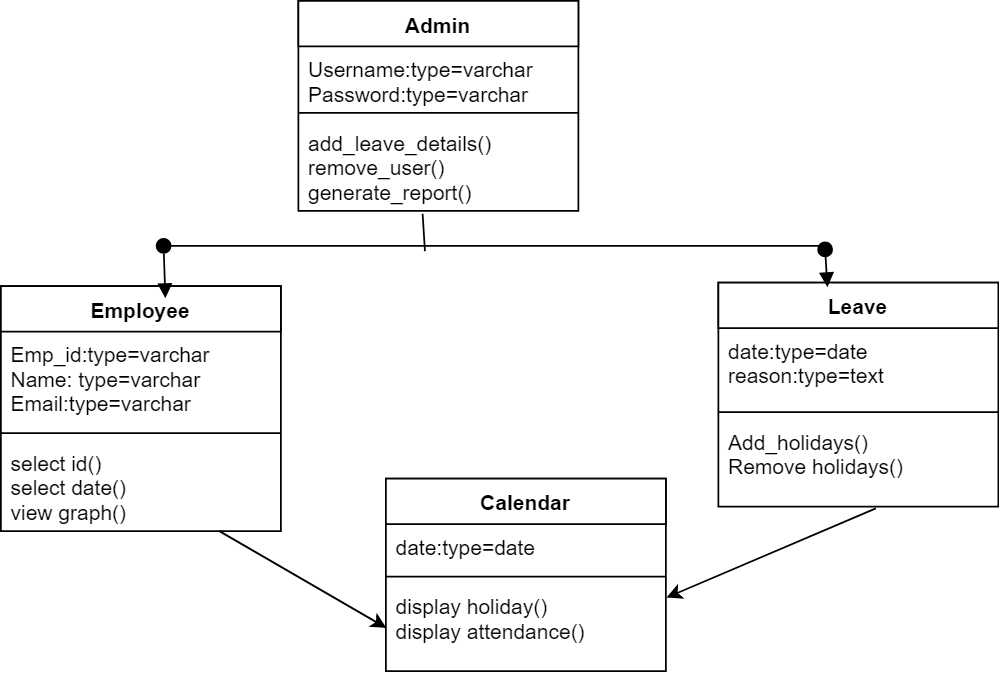


Fig 4.3 Class diagram of calendar app

**5.2.2 USE CASE DIAGRAM**

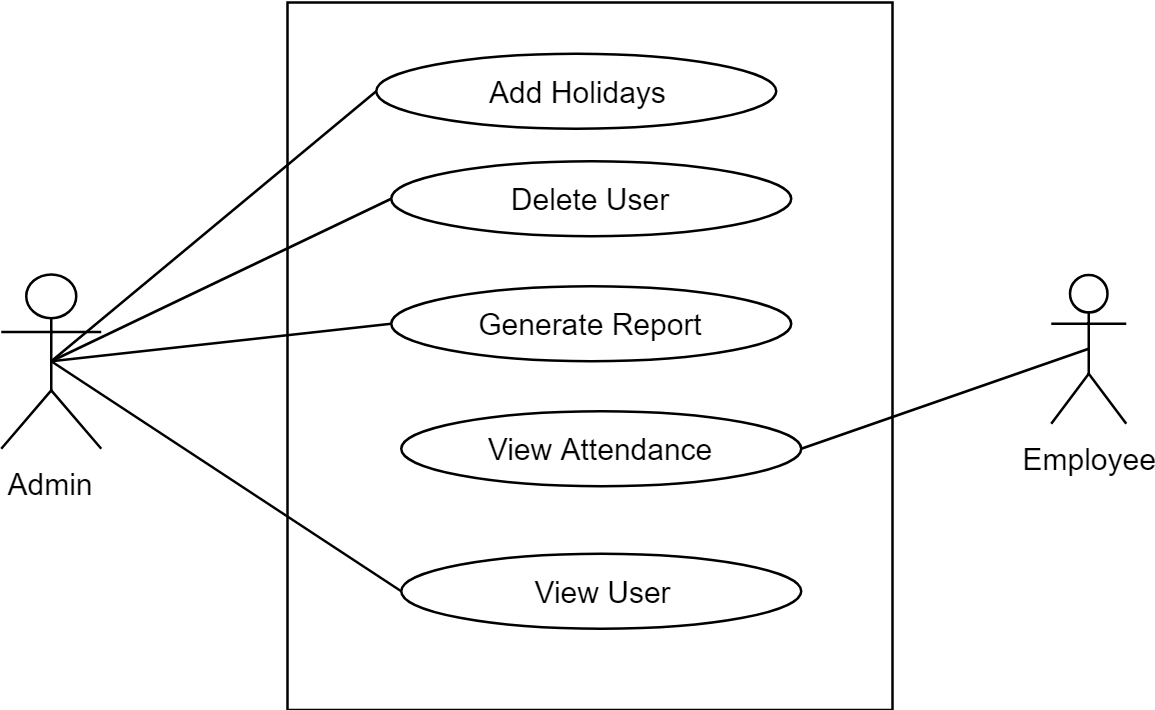
****

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**

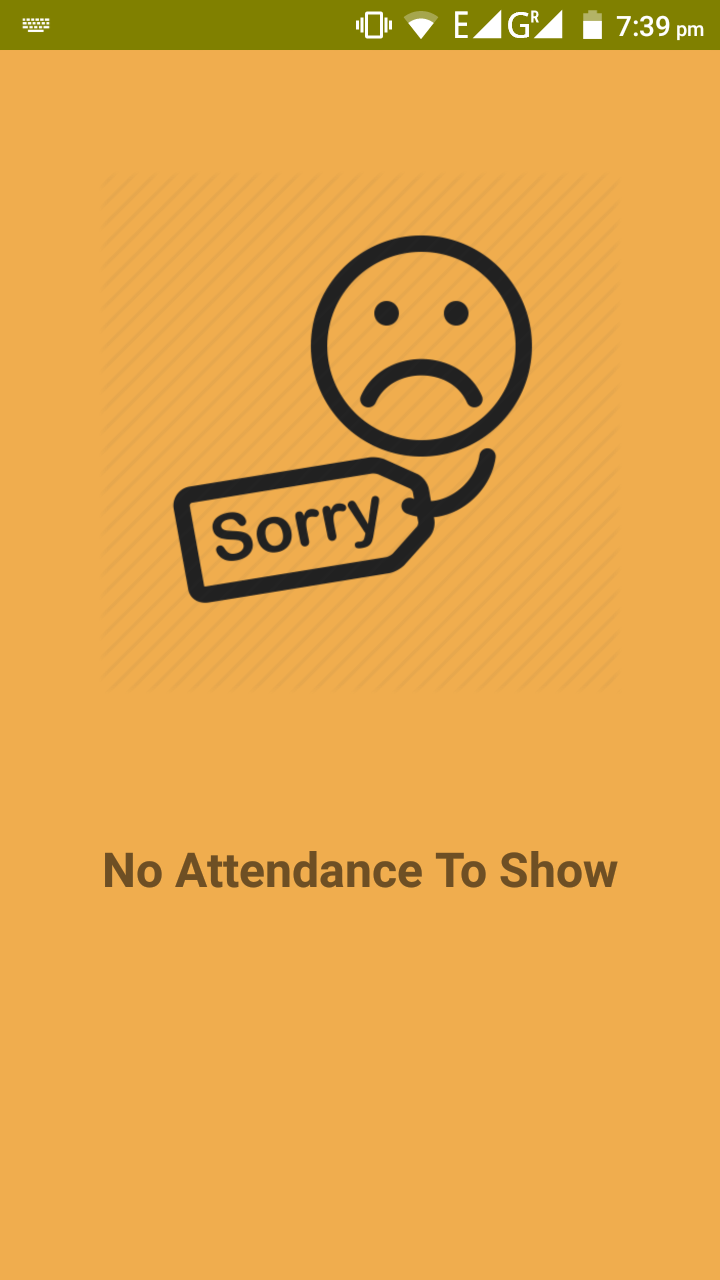
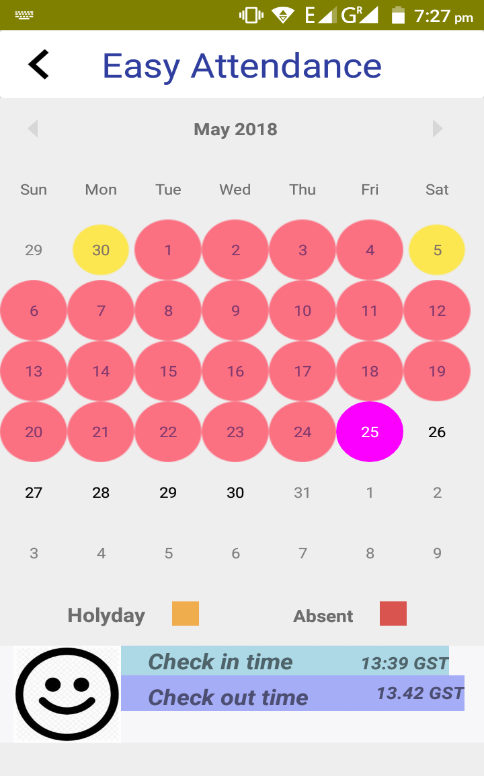
 

Fig 5.1: Check in/out time screen Fig 5.2: No attendance screen

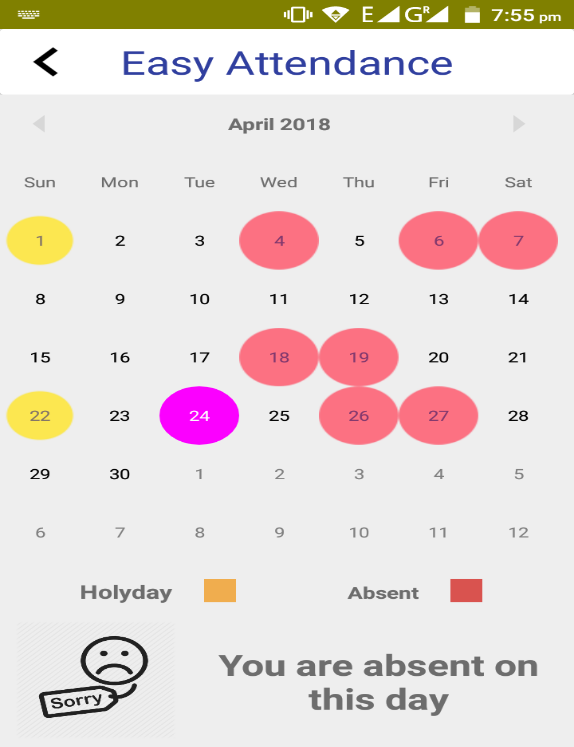
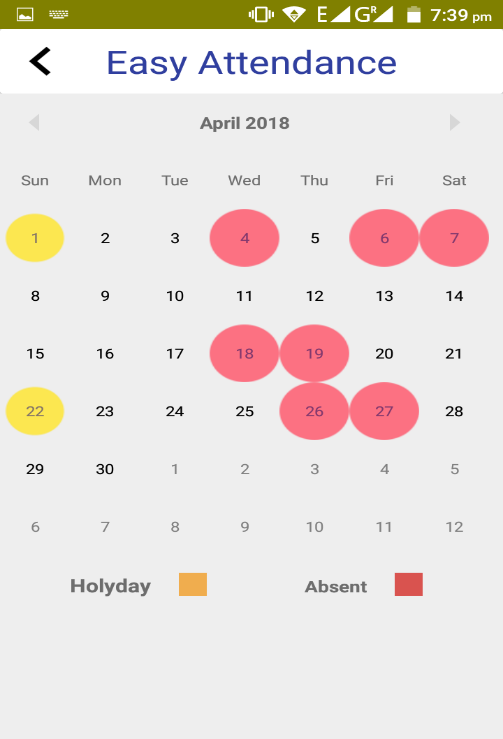


Fig 5.3: Attendance display screen Fig 5.4: Showing absent to user

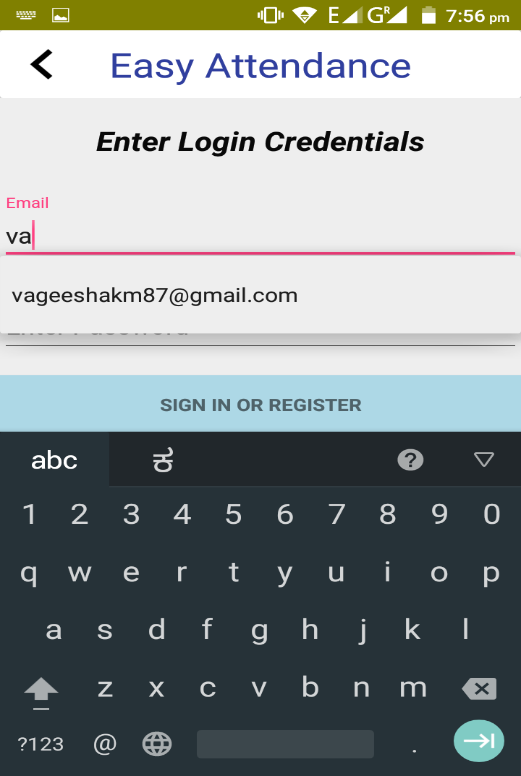
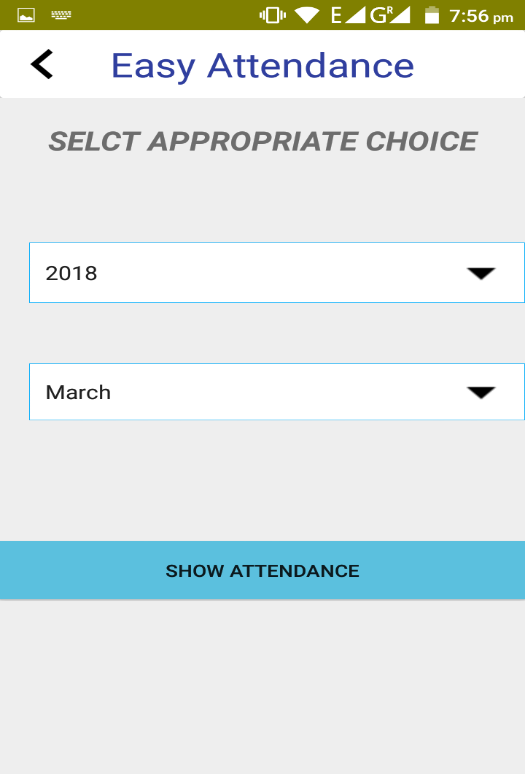


Fig 5.5: Login Screen Fig 5.6: Select date screen

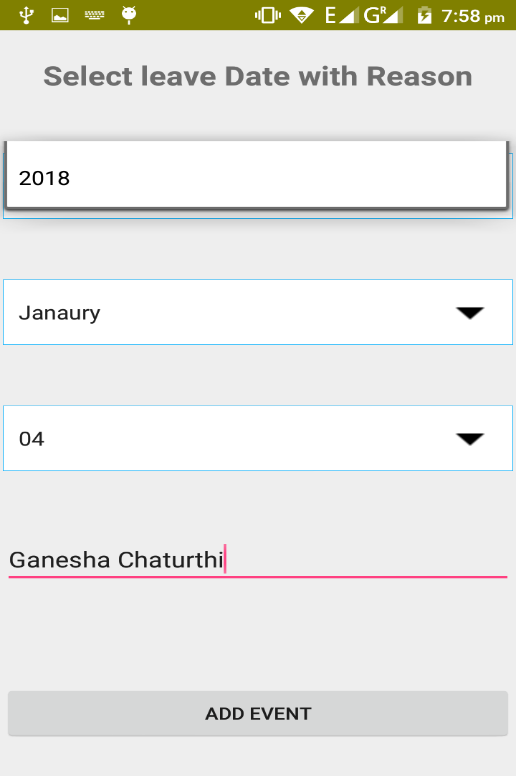


Fig 5.7: Update holidays screen with reason

**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/>.