# LOCATION BASED SMART ATTENDANCE TRACKING SYSTEM

A PROJECT REPORT

submitted

*in the partial fulfillment of the requirements for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

in

**COMPUTER SCIENCE AND ENGINEERING**

by

**V.R.SATHVIK REDDY(17B81A05N0)**

**P.SATWIK REDDY (17B81A05N3)**

**N.SATWIK REDDY (17B81A05N4)**

Under the guidance of

Dr.A.Vani Vathsala

Professor and Head, Department of CSE


**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**CVR COLLEGE OF ENGINEERING**

(***An Autonomous institution, NBA, NAAC Accredited and Affiliated to JNTUH, Hyderabad***)

Vastunagar, Mangalpalli (V), Ibrahimpatnam (M),
Rangareddy (D), Telangana- 501 510

June, 2020

**Cherabuddi Education Society's**

# CVR COLLEGE OF ENGINEERING

**(An Autonomous Institution)**

**ACCREDITED BY NBA, NAAC 'A' Grade**

(Approved by AICTE & Government of Telangana and Affiliated to JNTU Hyderabad)

Vastunagar, Mangalpalli (V), Ibrahimpatnam (M), R.R.District.

Web: http://www.cvr.ac.in, email: info@cvr.ac.in

Ph : 08414 – 252222, 252369, Office Telefax : 252396, Principal : 252396 (O)

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

## CERTIFICATE

This is to certify that the project entitled "**LOCATION BASED SMART ATTENDANCE TRACKING SYSTEM "** being submitted by **V.R.SATHVIK REDDY (17B81A05N0), P.SATWIK REDDY (17B81A05N3), N.SATWIK REDDY (17B81A05N4)** in partial fulfillment of the requirement for the award of degree of Bachelor of Technology in Computer Science and Engineering to Jawaharlal Nehru Technological University (JNTUH), Hyderabad, is a bonafide work carried out by them under my guidance and supervision. The results provided in this report have not been submitted to any other university or institution for the award of any degree.

**Dr.A.Vani Vathsala**

Professor and Head, Department of CSE

CVR College of Engineering

# ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose encouragement and guidance has been a source of inspiration throughout the course of the project.

It is a great pleasure to convey our profound sense of gratitude to our **Principal Dr. Nayanathara K. S** and **Dr.A.Vani Vathsala, Professor and Head, Department of CSE**, CVR College of Engineering for being kind enough for arranging the necessary facilities for executing the project in the college.

We would like to express our sincere gratitude to our guide, **Dr.A.Vani Vathsala, Professor and Head, Department of CSE**, CVR College of Engineering**,** whose guidance and valuable suggestions have been indispensable to bring about the successful completion of our project.

We would also like to express our gratitude to all the staff members and lab faculty, department of **Computer Science and Engineering**, CVR College of Engineering for the constant help and support.

We wish a deep sense of gratitude and heartfelt thanks to management for providing excellent lab facilities and tools. Finally, we thank all those whose guidance helped us in this regard.

# ABSTRACT

Now a days, attendance monitoring and working hour calculation is very essential for almost every institution or organization. Typically there are two types of attendance system available, i) Manual and ii) Automated. Manual system involves the use of sheets of paper or books in taking attendance where employees fill out and managers oversee for accuracy. This method could be erroneous because sheets could be lost or damaged. Also the extraction of relevant data and the manual computation of working time is very time consuming. It takes an extra employee to check for the attendance and timing of other employees which includes cost overhead for the organization as well.

So,we propose a modern day solution that is as follows

- The employer will interact with the web app.
- The employer will enter the details regarding the location where the employee is supposed to work.
- These details will include the employee user name, date and location .
- The employee will use the mobile app to mark the attendance and view their current work schedule.
- The employee will be asked to login and provide biometric details in order to mark their attendance.
- The Mobile will collect the attendance based on the location of the officer through GPS system / Geo-Location System.
- If the employee is not in the vicinity of the assigned location the attendance will not be marked.
- At the end of the work schedule the employee will be asked to end their shift by marking the attendance.

# TABLE OF CONTENTS

# LIST OF FIGURES

# ABBREVIATIONS

HTML – Hyper Text Markup Language

CSS – Cascading Style Sheets

OS – Operating System

RAM – Random Access Memory

API – Application Program Interface

# 1. INTRODUCTION

## 1.1 MOTIVATION:

Every employer wants their staff to perform at the highest level. Managers collect and analyze all data on employees' work for optimum performance and better project delivery, especially when it comes to time. While it's easy to monitor work hours and activities of employees working in the office, things get more complicated for people whose jobs require to go to distant locations away from their main office for work. Here the main problem is to ensure whether the employee is really present at the work site or not. That's where GPS tracking comes into the picture, Following this thought we've developed our project LOCATION BASED SMART ATTENDANCE TRACKING SYSTEM which enables the employer to ensure whether the employee is in the desired location or not.

## 1.2 PROBLEM STATEMENT:

LOCATION BASED SMART ATTENDANCE TRACKING SYSTEM is to develop a location based smart time and attendance tracking system along with face recognition as authentication.

## 1.3 PROJECT REPORT ORGANIZATION:

This book contains six chapters. The first chapter contains motivation and problem statement of the project. The second chapter includes the characteristics of the problem, design challenges and proposed solution. The third chapter includes requirements and specifications.

The fourth chapter includes which contains UML diagrams and Technology Description . The fifth chapter includes Implementation which contains the technologies used for developing the application and code snippets. The fifth chapter also contains testcases and screenshots of the applications. The sixth chapter looks into the future enhancements and conclusion of the project.

# 2. PROPOSED MODEL

## 2.1. INTRODUCTION TO THE CHARACTERISTICS OF THE PROBLEM:

From traditional punch cards to modern access cards and biometric systems, tracking employee working hours and attendance has been an essential process. Currently most of the organizations use automated attendance management systems and biometric devices to manage employee attendance. This type of attendance management systems best suits for the employees who works only in the office but when it comes to the organizations which needs to send particular employees to different locations to work the main problem they encounter is to know whether the employee is in the desired location or not.

Through the existing system we can't know the location of the employee. So in order to overcome this problem we have developed a location based smart attendance tracking system. By using this project we can take the attendance of the employee only if he /she is present at the desired work location so by this the employer can ensure that employee has reached the work location or not. Here the employees attendance and location are taken through mobile app and monitored through a web app by the employer**.**

## 2.2. DESIGN CHALLENGES:

Cross-platform application :

Choosing a cross-platform tool so that the mobile application works for both Android & iOS platforms with common codebase.

Authentication:

In any mobile application framework there is no built-in module for face authentication.

Location Tracking:

Accessing location of employee from work location.

Database:

Selecting a database such that it can synchronize both mobile and web application.

## 2.3.PROPOSED SOLUTION:

Instead of just relying on the attendance we are also including the location tracking system so that we can exactly know whether the employee went to the work location or not. For achieving this we are creating a web application for the employer and mobile application for the employee.

First the employee should login through the mobile application where he can view all the schedules that are assigned to him in the coming days which provides the location of the working place along with the directions. After reaching the work location he/she should give the starting attendance by face authentication. Once the work is done he/she should again give the ending attendance. The employee will be able to give the attendance only when he/she is in the vicinity of the work location.

The schedules for the employees are provided by the employer. Here the employer uses the web application where he can assign schedules to the employees and can check whether the employee went to the work location. Employer can also view the monthly attendance report of a particular employee .

# 3.REQUIREMENTS AND SPECIFICATIONS

## 3.1 SOFTWARE REQUIREMENTS:

### 3.1.1 Functional Requirements:

These requirements define the capabilities and functions that the implemented system must have in order to achieve its intended purpose. It includes a set of inputs, behavior and outputs in line with the objectives of the study. They include:

Web Application:

- Login / Logout: To gain access to the web application employer must login using username and password.
- Add Employee: Employer can add employee details in the system.
- Create Schedule: Employer can assign a particular day's schedule to an employee.
- Location: Employer can view the current location of an employee.
- Monthly Attendance: Employer can view the monthly attendance of an employee.

Mobile Application:

- Login / Logout: To gain access to the mobile application employee must login using username and password.
- View Schedule: The employee can view schedules assigned to him which specifies the work location of a particular day.
- Give Start Attendance: Employee can give the start attendance only within the vicinity of the work location with face biometric.
- Give End Attendance: Employee can give the end attendance only within the vicinity of the work location with face biometric.
- Monthly Attendance: Employee can view his/her monthly attendance.

### 3.1.2 Non-Functional Requirements:

These requirements that specify the criteria used to judge the operation of the system. They were constructed in agreement with functional requirements that define specific behavior and functions. They include:

- Usability: the system interface should be easy to use.
- Reliability and availability: the system should be reliable and always available to perform tasks requested by the user.
- Scalability: the system should be able to adopt additional functionalities. Additional data should be easy to incorporate.
- Integrity: the system being data oriented, it should ensure that the data analyzed and stored is not altered or corrupted.
- Performance: the system should have an acceptable response time while performing its functions.
- Security: The system should allow only authorized users to use its functionalities.

## 3.2 SYSTEM SPECIFICATIONS:

### 3.2.1 Software Specifications:

Operating System: Windows/MAC (Web), Android/iOS (Mobile)

Framework: Flask(Web), Flutter(Mobile)

Database: Firebase

### 3.2.2 Hardware Specifications:

- An android phone with a minimum of 2GB RAM with Marshmallow OS(6.0.1) or higher and a front facing camera / An iOS phone with a minimum of 2GB RAM with iOS 8 or higher and a front facing camera.
- A PC with
    a. Processor : intel i3 or higher
    b. RAM : 2 GB
- Strong internet connectivity.

# 4. ANALYSIS AND DESIGN

## 4.1 USECASE DIAGRAM:

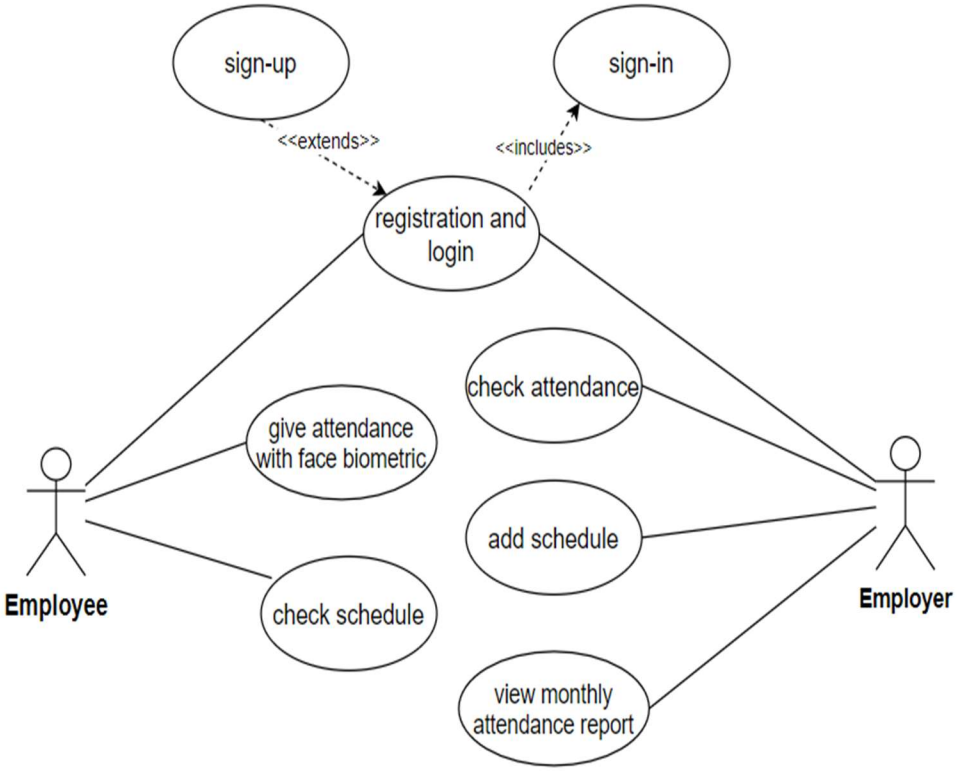


***Fig 4.1*** *Use Case Diagram*

## 4.2 CLASS DIAGRAM:

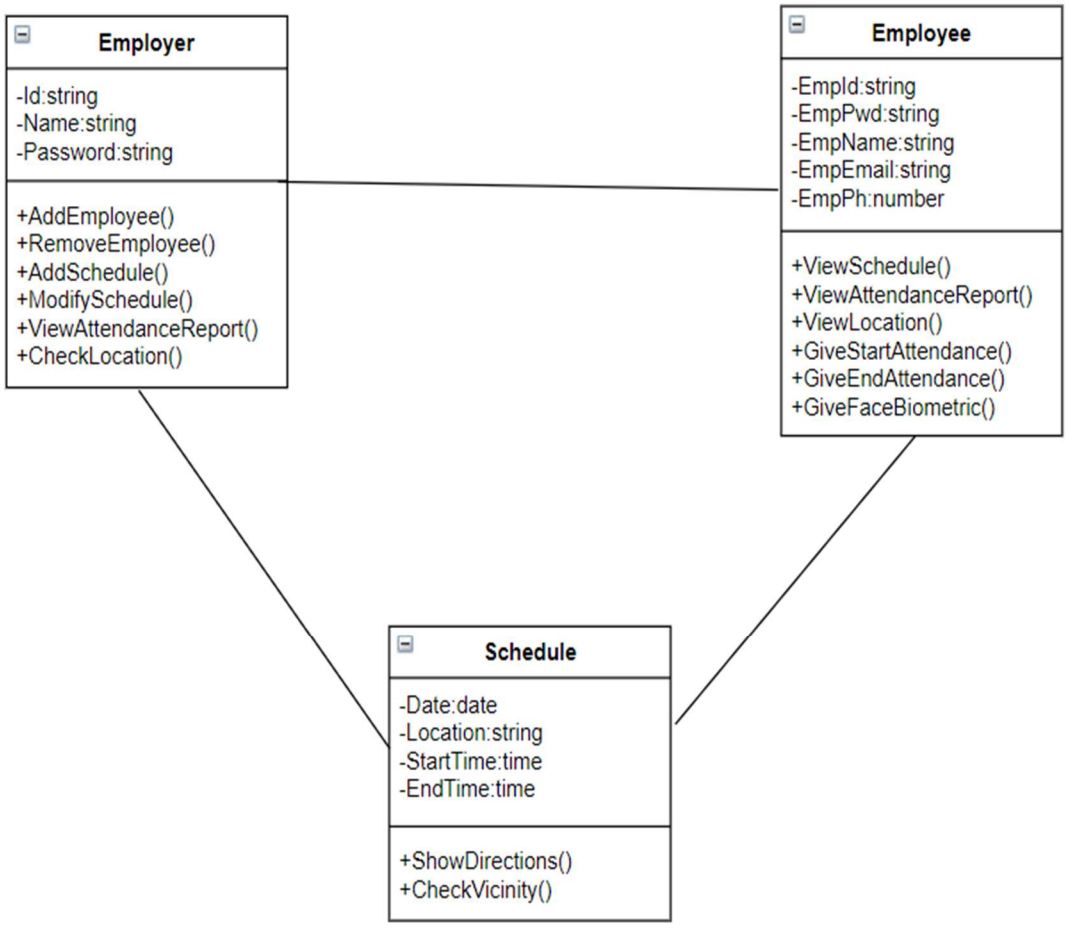


***Fig 4.2*** *Class Diagram*

**4.3 ACTVITY DIAGRAM :**

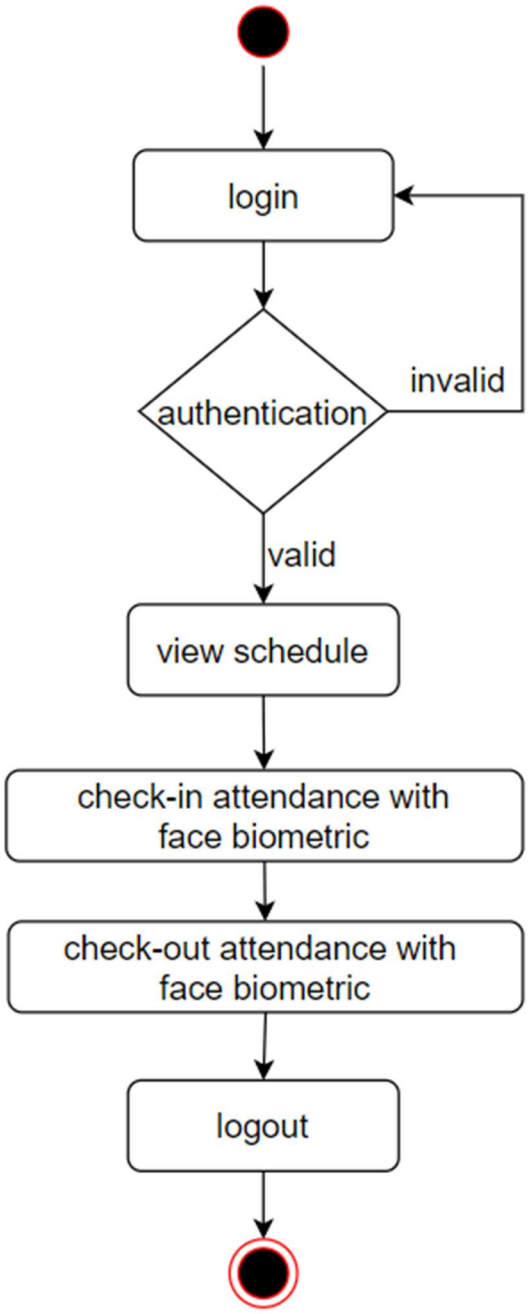


***Fig 4.3.1*** *Activity Diagram For Employee*

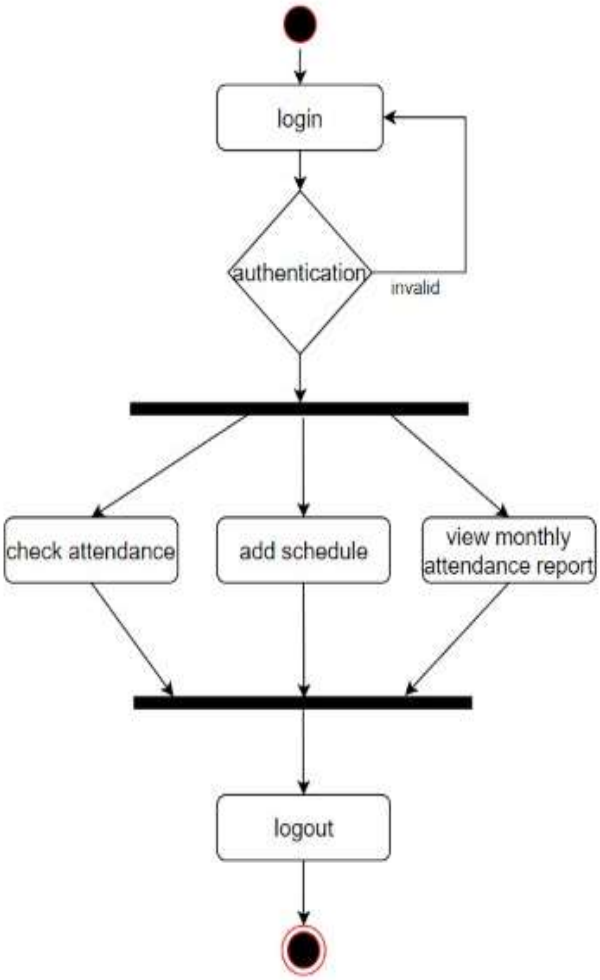


***Fig 4.3.2*** *Activity Diagram For Employer*

## 4.4 SEQUENCE DIAGRAM :

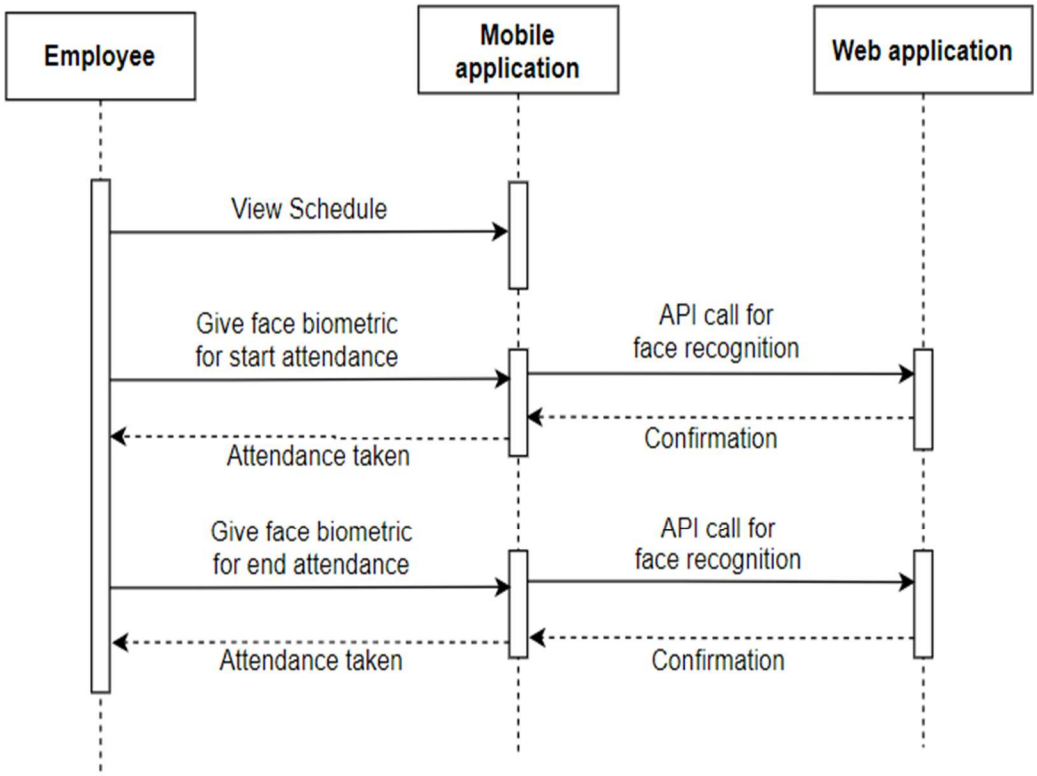


***Fig 4.4*** *Sequence Diagram For Attendance*

## 4.5 SYSTEM ARCHITECTURE:

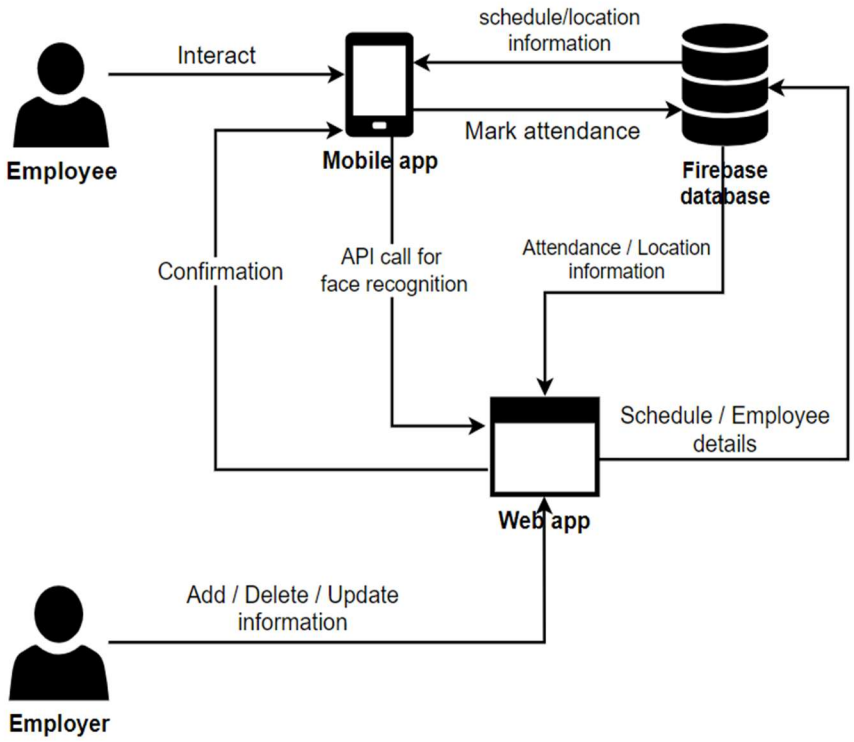


***Fig 4.5*** *System Architecture*

## 4.6. TECHNOLOGY DESCRIPTION:

**Flutter:**

Flutter is an open-source UI software development kit created by Google. It is used to develop applications for Android, iOS, Windows, Mac, Linux, Google Fuchsia and the web from a single codebase.

The major components of Flutter include:

Dart platform:

Flutter apps are written in the Dart language and make use of many of the language's more advanced features. On Windows, macOS and Linux via the semi-official Flutter Desktop Embedding project, Flutter runs in the Dart virtual machine which features a just-in-time execution engine. While writing and debugging an app, Flutter uses Just In Time compilation, allowing for "hot reload", with which modifications to source files can be injected into a running application. Flutter extends this with support for stateful hot reload, where in most cases changes to source code can be reflected immediately in the running app without requiring a restart or any loss of state. Release versions of Flutter apps are compiled with ahead-of-time (AOT) compilation on both Android and iOS, making Flutter's high performance on mobile devices possible.

Flutter engine:

Flutter's engine, written primarily in C++, provides low-level rendering support using Google's Skia graphics library. Additionally, it interfaces with platform-specific SDKs such as those provided by Android and iOS. The Flutter Engine is a portable runtime for hosting Flutter applications. It implements Flutter's core libraries, including animation and graphics, file and network I/O, accessibility support, plugin architecture, and a Dart runtime and compile toolchain. Most developers will interact with Flutter via the Flutter Framework, which provides a modern, reactive framework, and a rich set of platform, layout and foundation widgets.

Design-specific widgets:

The Flutter framework contains two sets of widgets which conform to specific design languages. Material Design widgets implement Google's design language of the same name, and Cupertino widgets implement Apple's iOS Human interface guidelines.

**HTML:**

HTML is a hypertext markup language which is in reality a backbone of any website. Every website can't be structured without the knowledge of html. If we make our web page only with the help of html, than we can't add many of the effective features in a web page, for making a web page more effective we use various platforms such as CSS. So here we are using this language to make our web pages more effective as well as efficient. And to make our web pages dynamic we are using Java script.

**CSS:**

CSS Stands for "Cascading Style Sheet." Cascading style sheets are used to format the layout of Web pages. They can be used to define text styles, table sizes, and other aspects of Web pages that previously could only be defined in a page's HTML. The basic purpose of CSS is to separate the content of a web document (written in any markup language) from its presentation (that is written using Cascading Style Sheets). There are lots of benefits that one can extract through CSS like improved content accessibility, better flexibility and moreover, CSS gives a level of control over various presentation characteristics of the document. It also helps in reducing the complexity and helps in saving overall presentation time. CSS gives the option of selecting various style schemes and rules according to the requirements and it also allows the same HTML document to be presented in more than one varying style.

**Flask:**

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-

relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more frequently than the core Flask program.

**Firebase:**

Firebase is built on Google infrastructure and scales automatically, for even the largest apps. Firebase gives you functionality like analytics, databases, messaging and crash reporting so you can move quickly and focus on your users. Firebase products work great individually but share data and insights, so they work even better together. Firebase frees developers to focus crafting fantastic user experiences. You don't need to manage servers. You don't need to write APIs. Firebase is your server, your API and your datastore, all written so generically that you can modify it to suit most needs. Yeah, you'll occasionally need to use other bits of the Google Cloud for your advanced applications. Firebase can't be everything to everybody. But it gets pretty close.

# 5 . IMPLEMENTATION AND TESTING:

## 5.1 IMPLEMENTATION:

### 5.1.1 Mobile Application:

- Login Module: First the employee has to enter his username and password .These details are compared with the details in the database .If correct, it then redirects to dashboard page.

```
Future<void> loginbutton()
  async {
    if (usern.text !=''){
                  var x=await indatabase(usern.text);
                if ( usern.text!='' && x)
                {
                await userdetails(usern.text);
                setState(() {
                  arr=usern.text;
                   Navigator.push(context, MaterialPageRoute(builder
: (context) =>Dashpage()));
                });
                }
                }
                else{
                  print('not authorized');
                }
  }
```

- View Schedules: Here the employee can view his schedules assigned by the employer which are stored in database. The schedule consists of particular day's work location. When he taps on the schedule it redirects to google maps.

```
 Widget _buildList(BuildContext context, DocumentSnapshot document)
{
    String x = document['date'];
    GeoPoint g = document['locat'];
    var newFormat = DateFormat("dd-MM-yyyy");
    DateTime n = new DateTime(int.parse(x.substring(6)),
        int.parse(x.substring(3, 5)), int.parse(x.substring(0, 2)));
    var y = DateFormat('EEEE').format(n);
    return ListTile(
```

```
      leading: Padding(
        padding: const EdgeInsets.all(8.0),
        child: Icon(
          IconData(0xe878, fontFamily: 'MaterialIcons'),
          color: Colors.blue[700],
          size: 30,
        ),
      ),
      title: Padding(
        padding: const EdgeInsets.all(8.0),
        child: Text('$x',
            style: TextStyle(
                color: Colors.blue[700],
                fontWeight: FontWeight.bold,
                fontSize: 18)),
      ),
      subtitle: Text("  $y",
          style: TextStyle(
              color: Colors.orangeAccent,
              fontWeight: FontWeight.bold,
              fontSize: 16)),
      enabled: true,
      contentPadding: EdgeInsets.symmetric(horizontal: 4, vertical:
4),
      onTap: () {
        callmapsfunction(g.latitude, g.longitude);
      },
    );
  }


static Future<void> openMap(double latitude, double longitude) async
 {
    String googleUrl =
        'https://www.google.com/maps/search/?api=1&query=$latitude,$
longitude';
    if (await canLaunch(googleUrl)) {
      await launch(googleUrl);
    } else {
      throw 'Could not open the map.';
    }
}
```

- Give Start Attendance : Employee can give the attendance by uploading the photo of his face. A REST API call is made to the server to check for authentication. Here we are using face-recognition module in python . The model has an accuracy of 99.38% on theLabeled Faces in the Wild benchmark.It automatically finds the faces in the image with those in database and returns the match. If they are matched then the employee can give the start attendance only when he is in the vicinity of work location.

```
import face_recognition
import os
import cv2
import imutils as im
KNOWN_FACES_DIR='known_faces'
UNKNOWN_FACES_DIR='unknown_faces'
TOLERANCE=0.475
FRAME_THICKNESS=3
FONT_THICKNESS=2
MODEL='cnn'
known_faces=[]
known_names=[]

def preprocess():
    for name in os.listdir(KNOWN_FACES_DIR):
        for filename in os.listdir(f'{KNOWN_FACES_DIR}/{name}'):
            print(name,filename)
            image=face_recognition.load_image_file(f"{KNOWN_FACES_DI
R}/{name}/{filename}")
            encoding=face_recognition.face_encodings(image)
            if len(encoding)==0:
                print('not found')
                continue
            encoding=face_recognition.face_encodings(image)[0]
            known_faces.append(encoding)
            known_names.append(name)

def identify(image):
    image=face_recognition.load_image_file(image)
    x=image.shape
    if x[0]<x[1]:
        image=im.rotate(image,90)
    image=cv2.resize(image,(368,368),127)

locations=face_recognition.face_locations(image)
```

```
    encodings=face_recognition.face_encodings(image,locations)
    image=cv2.cvtColor(image,cv2.COLOR_RGB2BGR)
    matches=list()

    for face_encoding,face_location in zip(encodings,locations):
        results=face_recognition.compare_faces(known_faces,face_encoding,T
OLERANCE)
        match=None
        if True in results:
            match=known_names[results.index(True)]
            matches.append(match)
            print(f'match found :{match}')

    return matches
```

Code for calculating distance

```
Future<double> findingdistance(coords) async {
    double x = 0;
    var position = await Geolocator()
        .getCurrentPosition(desiredAccuracy: LocationAccuracy.high);
    double dis = await Geolocator().distanceBetween(position.latitude,
        position.longitude, coords.latitude, coords.longitude);
    return dis;
  }
```

Code for marking attendance

```
Future<void> valid(GeoPoint coords, user) async {
    Position position = await Geolocator()
        .getCurrentPosition(desiredAccuracy: LocationAccuracy.high);
    double dis = await Geolocator().distanceBetween(position.latitude,
        position.longitude, coords.latitude, coords.longitude);
    GeoPoint p=new GeoPoint(position.latitude, position.longitude);
    setState(() {
      currentloc = position;
      distance = dis;
    });

    print(dis);
    if (dis < 200 && imagecorr) {
       db.collection('schedules').document(user).collection('list').docume
nt(updateddt).updateData({'start': true,'startloc' :p});
       db.collection('fl').document(user).setData({'date':DateTime.now(),'
locat' :p});
      print('in viscinity');
```

```
      setState(() {
        viscinity = null;
      });
    } else {
      print('Not in vicinity');
      setState(() {
        viscinity = 'Not in Vicinity';
      });
    }
  }
```

Uploading image to server.

```
uploadImageToServer() async {
    File imageFile = await ImagePicker.pickImage(source: ImageSource.galle
ry);

    var stream =
        new http.ByteStream(DelegatingStream.typed(imageFile.openRead()));
    var length = await imageFile.length();
    var uri = Uri.parse('http://192.168.1.5:5000/upload');
    print('connection established');
    var request = new http.MultipartRequest('POST', uri);
    var multipartFile = new http.MultipartFile('file', stream, length,
        filename: path.basename(imageFile.path));
    print('continue');
    request.files.add(multipartFile);
    var response = await request.send();
    var x = await http.Response.fromStream(response);
    var y=json.decode(x.body);
    var n=y['name'] ;

    if (n==user){
      setState(() {
        imagecorr=true;
        imagemsg='Face found. Continue';
      });
      }
    else{
      setState(() {
        imagecorr=false;
        imagemsg='Face not found. Please try again';
      });
    }
```

- Give End Attendance : Employee can give the attendance by following the same steps as in Start Attendance.
- Monthly Attendance : Employee can view the days on which he has given the attendance along with the total schedules assigned to him /her.

```
ListView.builder(
     shrinkWrap: true,
     itemCount: snapshot.data.documents.length,
     itemBuilder: (context, index) {
       return Container(
         padding: EdgeInsets.symmetric(
            horizontal: 10, vertical: 5),
              child: Card(
              color: snapshot.data.documents[index]['start']
              ? Colors.greenAccent[400]: Colors.red,
             shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(5.0),),
                   elevation: 2,
                    child: Container(
                    decoration: new BoxDecoration(),
                    child: _buildList(context,
                     snapshot.data.documents[index]))),
                  );
          },

);
```

### 5.1.2 Web Application:

- Login Module: : First the employer has to enter his username and password .These details are compared with the details in the database .If correct, it then redirects to dashboard page.

```
def login():
    err = None
    if request.method == 'POST':
        x=db.collection(u'emp').document(request.form['username']).g
et().to_dict()
        if   request.form['password'] != x['password']:
            err = 'Invalid Credentials. Please try again.'
        else:
            return redirect(url_for('homepage'))

    return render_template('login.html', error=err)
```

- Add Employee: Employer can add details of the employee. The details are then saved in the database.

```
'''adding users'''
@app.route('/addusers', methods=['GET', 'POST'])
def adding_users():
    error=None
    if request.method == 'POST':
        username=request.form['username']
        email=request.form['email']
        phoneno=request.form['pno']
        desig=request.form['designation']
        info={
            u'username' :username,
            u'email' :email,
            u'phone number':phoneno,
            u'designation' : desig
        }
        db.collection(u'userdata').document(str(username)).set(info)
        db.collection("schedules").document(str(username)).set({
            u'count' :0,
        })
        error='Employee added successfully'
    return render_template('addusers.html',error=error)
```

- Create Schedule : Employer can add schedule for a particular employee. The details are stored in the database corresponding to the employee name.

```
@app.route('/addschedule',methods=['POST','GET'])
def addsch():
    err=None
    if request.method=='POST':
        date_t=request.form['dt']
        realdate=date_t[8:]+'-'+date_t[5:7]+'-'+date_t[0:4]
        loca=request.form['loc']
        locator=Nominatim(user_agent='myGeoCoder')
        loc = locator.geocode(loca)
        us=request.form['user']
        prev=db.collection("schedules").document(us)
        if not prev.get().exists:
            return render_template('schedule.html',error="Employee doesn't
exist.")
        if loc==None:
```

```
            return render_template('schedule.html',error="enter correct lo
cation.")
        location=firestore.GeoPoint(loc.latitude, loc.longitude)
        sch={
            u'locdesc':loca,
            u'date' : realdate,
            u'locat':location,
            u'start' :False,
            u'end' :False,
        }
        prev=db.collection("schedules").document(us)
        countofdocs=prev.get().to_dict()['count']
        countofdocs+=1
        db.collection("schedules").document(us).collection("list").documen
t(realdate).set(sch)
        prev.update({u'count' :countofdocs })
        err='Schedule added successfully'
        return render_template('schedule.html',error=err)
```

- MonthlyAttendance : Employer can view the days on which a particular has given the attendance along with the total schedules assigned to him /her.

```
@app.route('/currentdetails')
def index():
    user_refere=db.collection(u'users').stream()
    print(type(user_refere))
    if user_refere==None:
        print('empty')
    else:
        print('not empty')
    return render_template('data.html',di=user_refere)

#showing current location of user with map
@app.route('/map/<name>')
def showmap(name):
    try:
        x=db.collection('users').document(name)
        a=x.get().to_dict()
        locator=Nominatim(user_agent='myGeoCoder')
        cor=str(a['locat'].latitude)+', '+str(a['locat'].longitude)

        location = locator.reverse(cor)
        coord=[a['locat'].latitude,a['locat'].longitude,a['date'],name,
location.address]
```

```
except Exception():
    print('erroe')


return render_template('mapdisplay.html',ordinates=coord)
```

## 5.2 TESTING:

### 5.2.1 Mobile Application:

- Login : Here the entered credentials of the employee are validated. If they are correct then Dashboard screen (fig 5.2.1(b)) appears otherwise it shows an error message as shown in fig 5.2.1(a).


***Fig 5.2.1(a)*** *Login Screen*

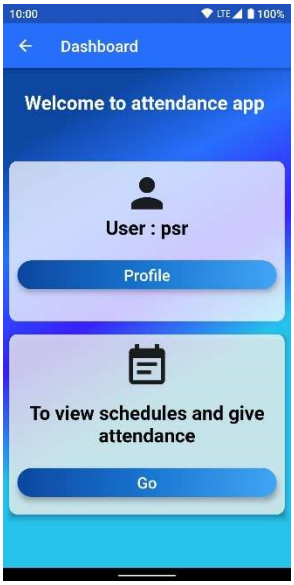


***Fig 5.2.1(b)*** *Dashboard*

- On clicking Profile Button it goes to profile page (fig 5.2.1(d)) , On clicking Go button, it displays available schedules (fig 5.2.1(c)).

- The App Drawer consists of all the features that are present in the app , profile page Consists of details of employee.

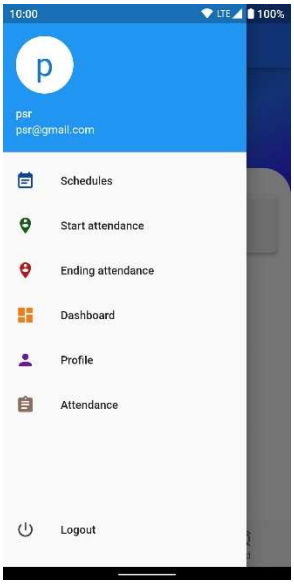


***Fig 5.2.1(c)*** *App Drawer*

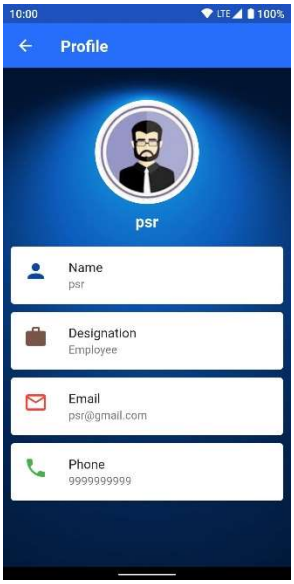


***Fig 5.2.1(d)*** *Profile page*

- Schedule : It shows all Available Schedules, On tapping the schedule, it redirects to google maps

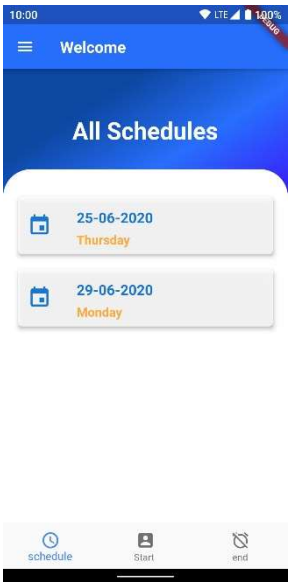


***Fig 5.2.1(e)*** *Schedules*


***Fig 5.2.1(f)*** *Google maps*

- Attendance:
  - The ‘click to get directions to location’ button opens google maps to show directions to work location.
  - For giving attendance first the employee has to click on the ‘Upload photo for biometric’ button if the face is matched then confirmation appears as shown in fig 5.2.1(h) else error message is shown as shown in fig 5.2.1(g).

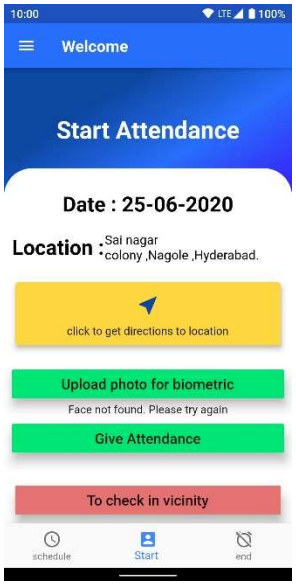


***Fig 5.2.1(g)*** *error message*

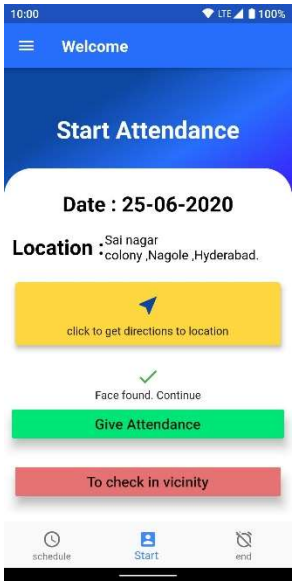


***Fig 5.2.1(h)*** *Confirmation*

  - Once the photo is uploaded successfully, On clicking ‘Give attendance’ button, attendance is taken successfully (fig 5.2.1(i)) otherwise it shows an error message as shown in fig 5.2.1(j).

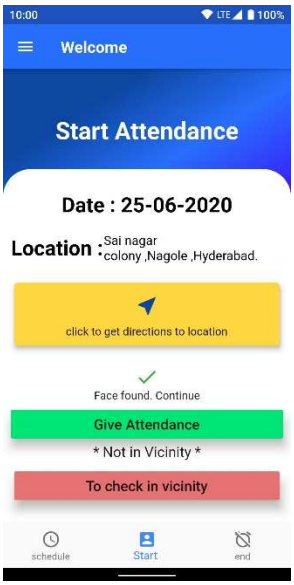


***Fig 5.2.1(j)*** *error message*

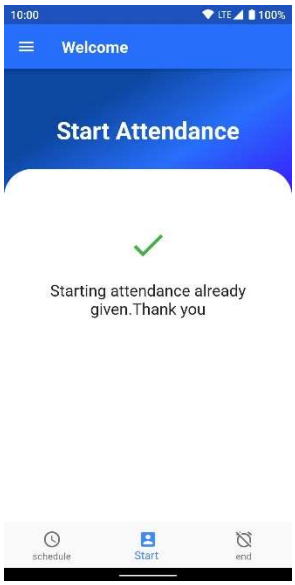


***Fig 5.2.1(i)*** *Confirmation*

- On Clicking 'To check in vicinity' button a popup appears showing how far the employee is from the work location.

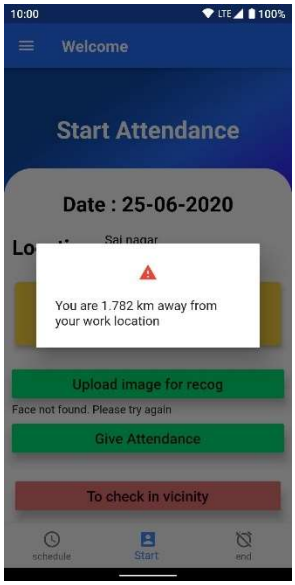


***Fig 5.2.1(k)*** *popup*


***Fig 5.2.1(l)*** *End Attendance*

- The process of end attendance is same as the start attendance.

- Monthly Attendance:
  - The red tile shows the unattended schedule and green tile shows the attended schedule.

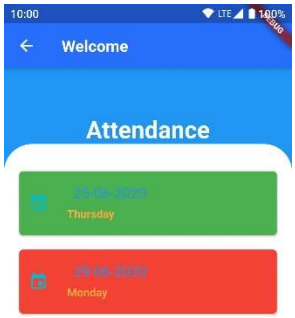


***Fig 5.2.1(m)*** *Monthly Attendance*

### 5.2.2 Web Application:

- Login: Here the entered credentials of the employee are validated. If they are correct then Dashboard appears otherwise it shows an error message as shown in fig 5.2.2(a).

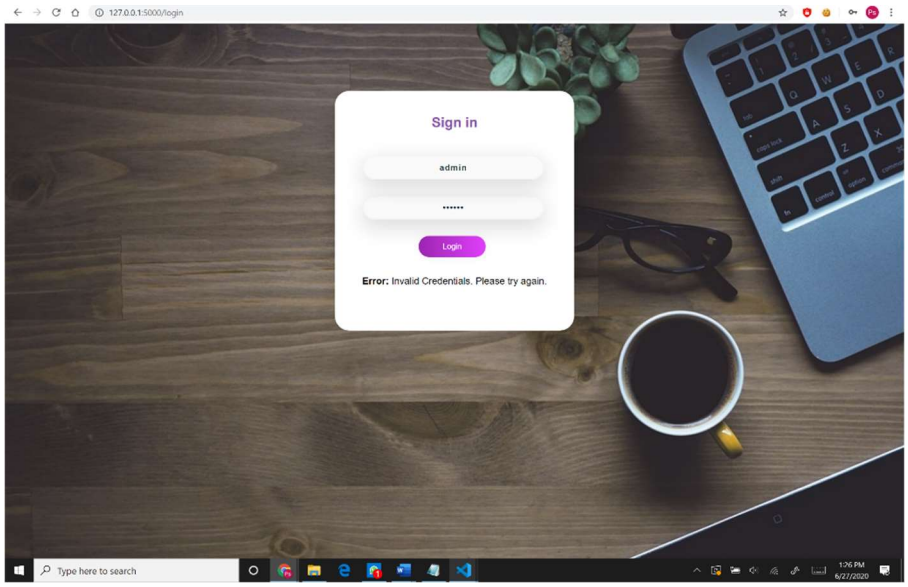


***Fig 5.2.2(a)*** *Login page*

- Add Employee : Employer can add employee by filling the details and clicking submit button then confirmation message appears.

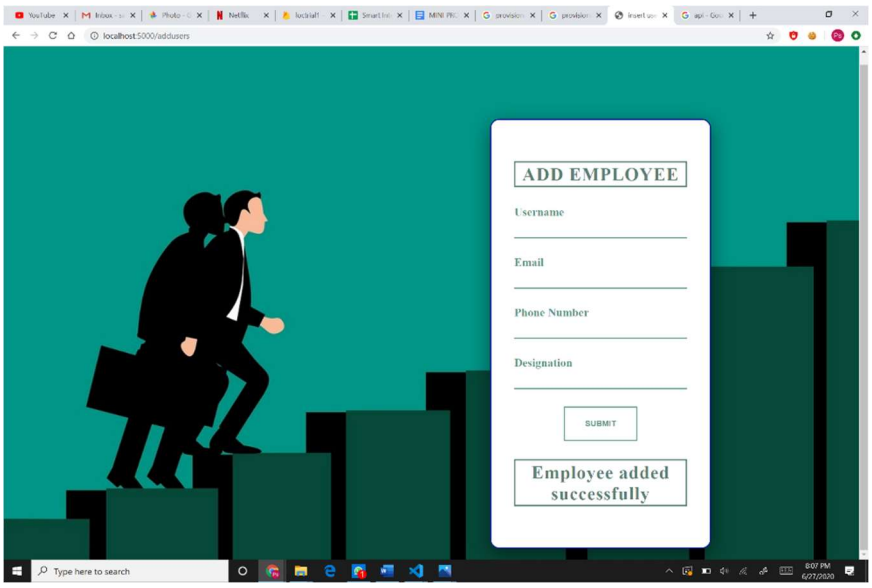


***Fig 5.2.2(b)*** *Adding Employee*

- Create Schedule：Employer can add schedule for a particular employee.

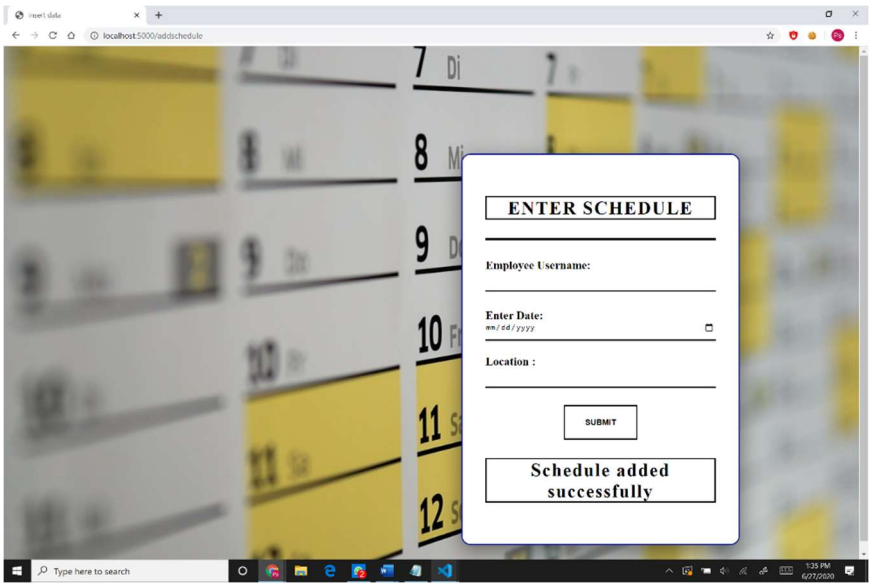


***Fig 5.2.2(c)*** *Adding Schedule*

- If the employer enters wrong username of employee, error message appears.

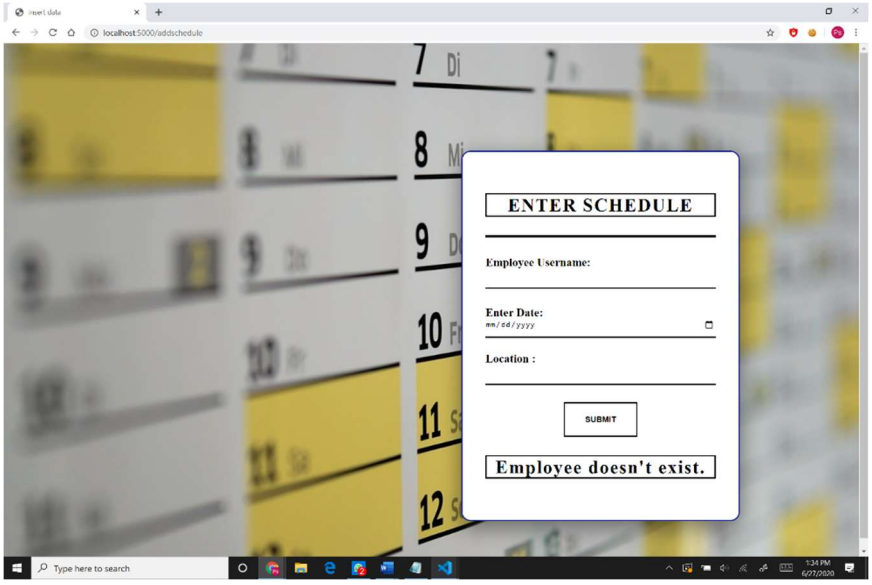


***Fig 5.2.2(d)*** *Schedule with error message*

- Location: Employer can view the current location of the employees

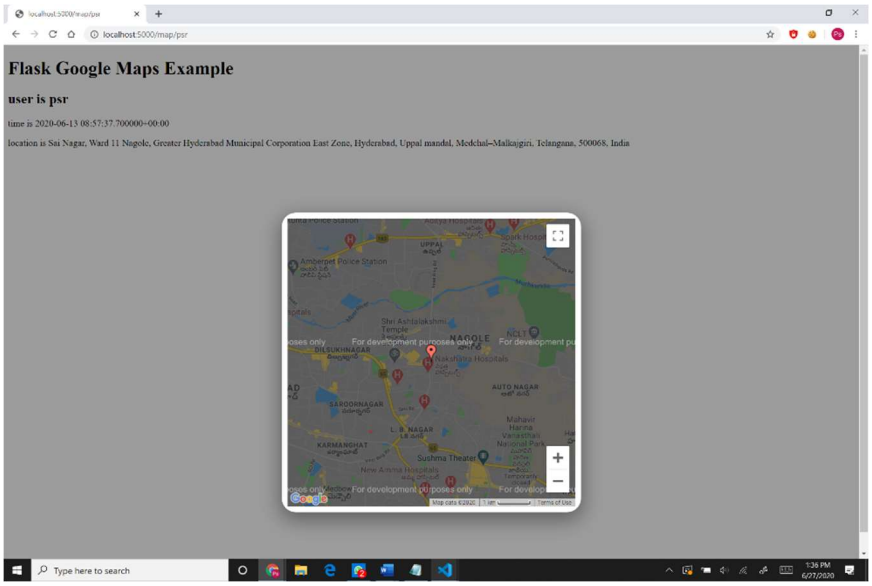


***Fig 5.2.2(e)*** *location of employee*

- Monthly Attendance: : Employer can view the days on which a particular has given the attendance along with the total schedules assigned to him /her.


***Fig 5.2.2(f)*** *Monthly report*

# 6.CONCLUSION AND FUTURE SCOPE

## 6.1 CONCLUSION:

- In this presentation we discussed location based smart attendance tracking system using GPS and as an additional feature, the Face-id technology.
- This app is very much useful for the organizations to track its employees who has to go to distant locations away from their main office for work.
- It is user friendly and has required options, which can be utilized by the user to perform the desired operations.
- It removes the problem of not knowing whether the employee is working according to the schedules provided to him/her.

## 6.2 FUTURE SCOPE:

- In future we are planning to implement time tracking system to calculate the working hours of an employee.
- Creating a mobile application for employer for easy access.
- As the technology emerges, it can be implemented in newer technologies with less code base.
- Based on the future security issues, security can be improved using emerging technologies.

# REFERENCES


1) Flutter: https://flutter.dev/
2) Firebase API: https://firebase.google.com/
3) HTML5 and CSS3: http://www.w3schools.com
4) Flask: https://flask.palletsprojects.com/en/1.1.x/
5) Face Recognition: https://pypi.org/project/face-recognition/