LOCATION TRACKER

Project under the guidance of Arup Kumar Bhattacharjee(RCCIIT), Debanjan Chatterjee , Milan Kamilya of Teratorn Software Private Limited

REPORT OF MAJOR PROJECT SUBMITTED FOR THE FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF COMPUTER APPLICATION.

RAJESH DE

Registration No: 151170510034 of 2015-2016

University Roll No: 11701015034

RCC INSTITUTE OF INFORMATION TECHNOLOGY

Approved by AICTE, New Delhi and Affiliated to MAKAUT, W.B.

An ISO 9001 - 2008 & ISO 14001 - 2004 Certified Institute

Canal South Road, Beliaghata, Kolkata, West Bengal 700015

RCC INSTITUTE OF INFORMATION TECHNOLOGY

श्रमम् बिना न किमपि साध्यम्

Certificate

The report of the project titled – 'LOCATION TRACKER' submitted by Rajesh De (Roll No: 11701015034) of MCA 6th Semester of 3rd Year, has been prepared under my supervision for the partial fulfillment of the requirements for MCA degree in Maulana Abul Kalam Azad University of Technology. The report is hereby forwarded.

OPTIONAL IN CASE:

[Name of guide]

(EXTERNAL SUPERVISOR)

Counter Signed By -

Name of HOD
Department of Computer Application
RCC INSTITUE OF INFORMATION TECHNOLOGY
KOLKATA: 700015, India.

Debanjan Chatterjee
Teratorn Software Private Limited

Milan Kamilya
Teratorn Software Private Limited

RCC INSTITUTE OF INFORMATION TECHNOLOGY

Certificate Of Acceptance

The report of the project titled – 'LOCATION TRACKER' submitted by Rajesh De (Roll No: 11701015034) of MCA 6th Semester of 3rd Year is hereby recommended to be accepted for the fulfillment of the requirements for MCA degree in Maulana Abul Kalam Azad University of Technology.

NAME OF THE EXAMINERS: **SIGNATURE WITH DATE**

_____ _____

_____ _____

_____

PLAGIARISM DECLARATION

1. I know that plagiarism means taking and using the ideas, writings, works or inventions of another as if they were one's own. I know that plagiarism not only includes verbatim copying, but also the extensive use of another person's ideas without proper acknowledgement (which includes the proper use of quotation marks). I know that plagiarism covers this sort of use of material found in textual sources and from the Internet.

2. I acknowledge and understand that plagiarism is wrong.

3. I understand that my research must be accurately referenced. I have followed the rules and conventions concerning referencing, citation and the use of quotations as set out in the Departmental Guide.

4. This assignment is my own work, or my group's own unique group assignment. I acknowledge that copying someone else's assignment, or part of it, is wrong, and that submitting identical work to others constitutes a form of plagiarism.

5. I have not allowed, nor will I in the future allow, anyone to copy my work with the intention of passing it off as their own work.

Name: …… Roll No.: .....

Signature: ……. Date: ……

TABLE OF CONTENT

CONTENTS :

ACKNOWLEDGEMENT

We express our sincere gratitude to Mr. Arup Kumar Bhattacharjee (HOD of Department of CA) and Debanjan Chatterjee & Milan Kamilya of Teratorn Software Private Limited for extending their valuable time for us and guide us for this project.

I am also indebted to the other teachers for their unconditional help and inspiration.

Last but not least I would like to express my gratitude to my HOD - Mr. Arup Kumar Bhattacharjee of our department (Department of CA) who helped me in his own way whenever needed.

(Signature of Student)
Reg. No:151170510034
Roll. No: 11701015034
MCA – 6th Sem
Session – 2015-2018,RCCIIT

INTRODUCTION

As the title goes, it is quite assumable that this web App will track location. Here this web App will track location trail of users for the last certain amount of time period. Each user who should be registered under the “user-app”, will sent geo location data from his/her device to the server. Now this “admin-app” will fetch those data from he server of particular user and will plot the data on the map.

Admins of a company/project might want to track their employee’s movement, based on some location or type of works. It might be for security, surveillance, etc. where the admins will like to know the current position of each user and its last certain time period of trails on map.

It might be used by Military and Army Forces to track each man’s active in the battlefield. Even this App will be useful in zoological surveys and other Wildlife Protection Program. Geo devices can be attached to the body of certain species and it can be tracked from survey centers, further with the oncoming facilities of drones, we can even track at real time. It will surely reduce the threat of them getting killed by poachers.

It’s a basic app which will keep the anyone remain connected with a supervisor for better control, security and management.

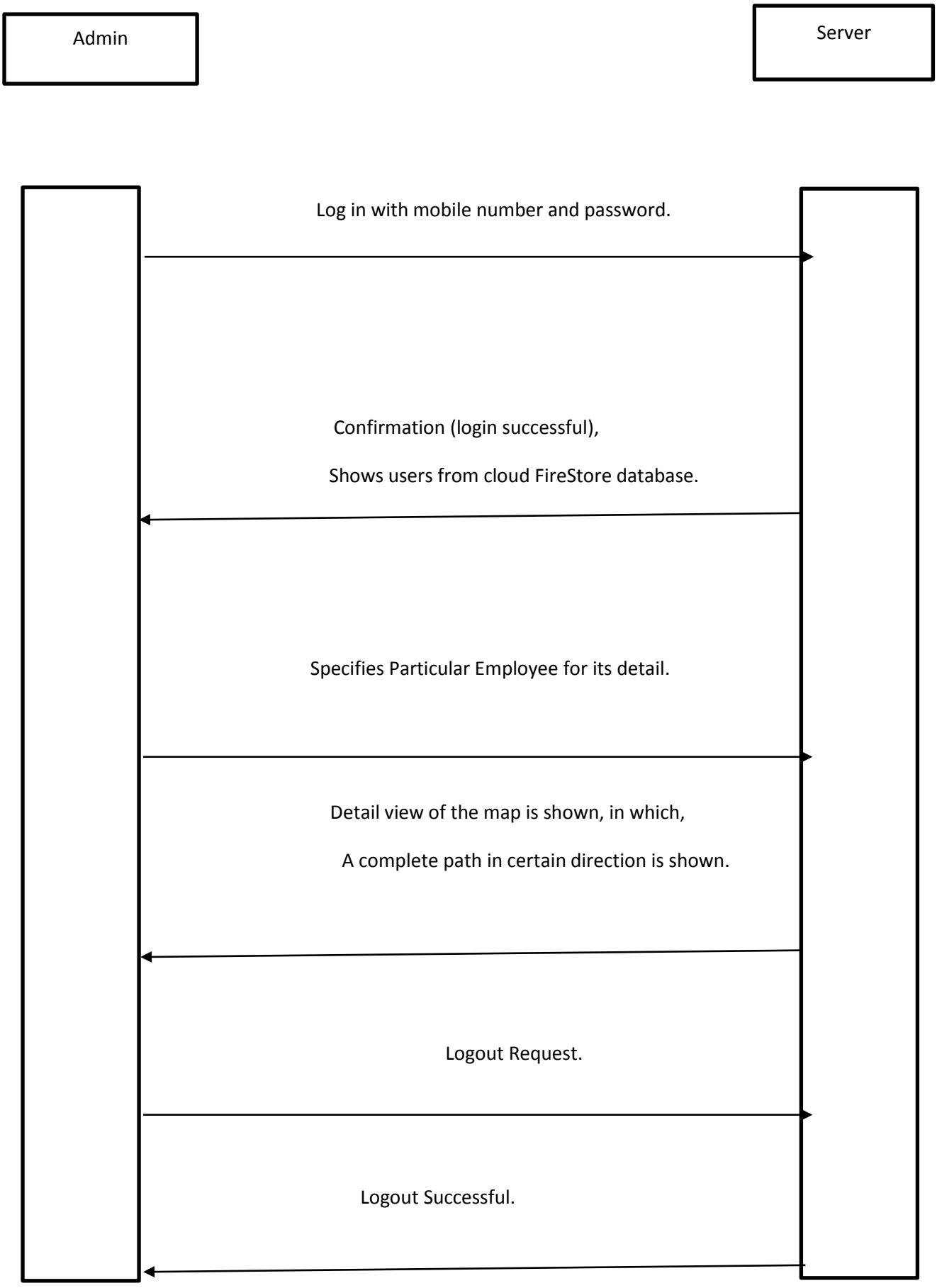
PROBLEM ANALYSIS: -

The Application is designed and developed to collect geo-location data of a person (employee in this case) from the server. Another app that will be installed in the employee's device will send the geo-location to the server at certain intervals. This application pickup location values from the server and will draw the lines between these points, thus showing the exact path in which, the user been through in the specified time period.

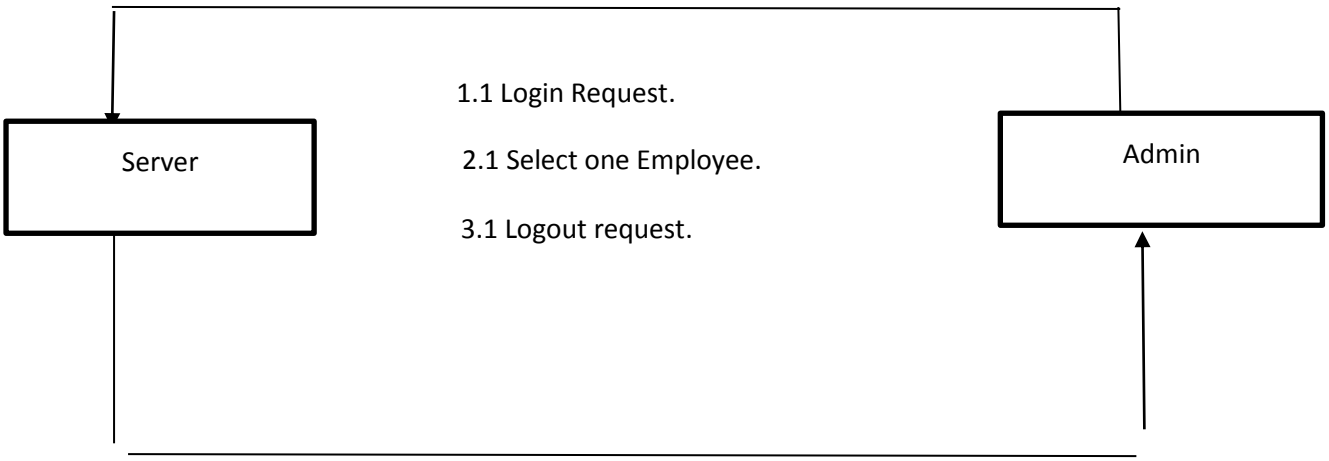
The objectives of this project are to:

- Surveillance on higher priority items.
- Real time tracking of employees, users.
- Tracking of delivery products.
- Protection of wildlife species.

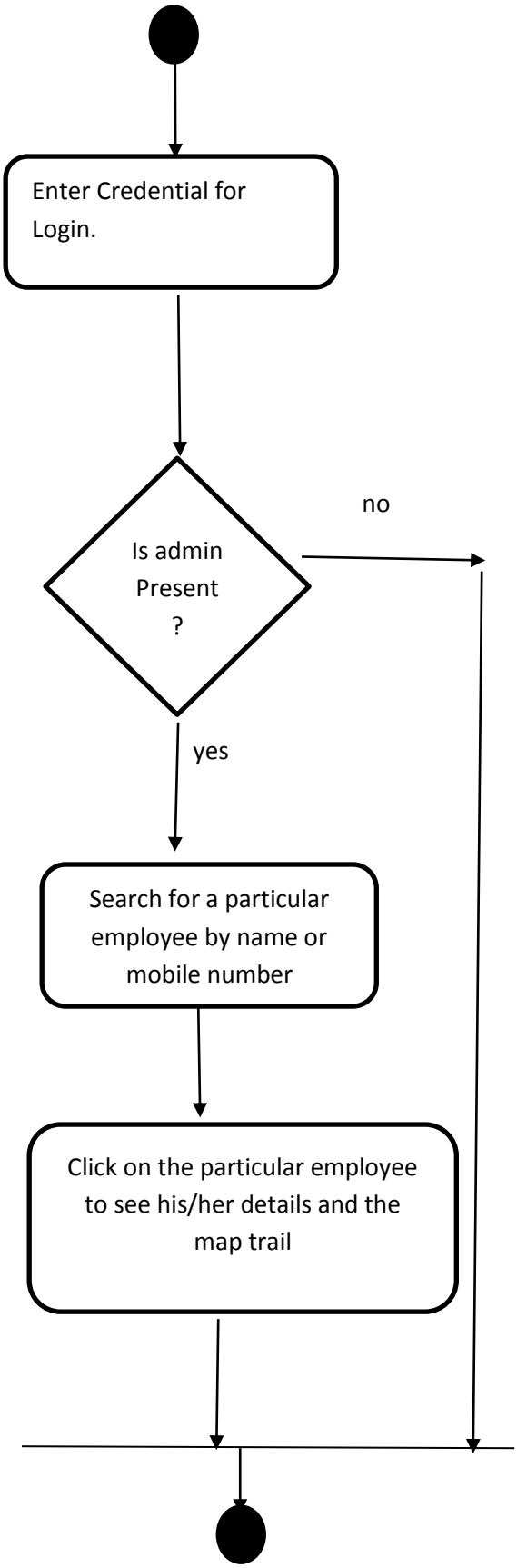
The following diagram contains the application's **Sequence diagram**:-



The following diagram contains the application's **Collaboration diagram**:-



The following diagram contains the application's **Activity diagram**:-



TOOLS AND PLATFORMS:-

The project is based on a web application that runs in a web browser in the front end using Angular, and the entire backend operation is developed using JAVA programming language and its concepts(Collection classes, OOPS concept etc.), Java SE (Standard Edition) features are used to build the project.

The tools that were used to build the project were:

- VISUAL STUDIO CODE- (A code editor for building and debugging modern web and cloud applications) ,
- Google Firebase, Cloud FireStore- (Database)
- Google Map API

Platform & Operating System used:

- LINUX O/S

Development Components:

- Client Side:

  Angular 4, HTML 5, SCSS, TypeScript, Bootstrap, Node server

- Server Side:

  Angular Google Map Package, Google Cloud Firestore (Database)

IMPLEMENTATION DETAILS:-

Our modules in this application are:

LOGIN, DASHBOARD, LOCATION_DETAILS

DASHBOARD:

It is the main module which will show all the employees who are online and the number of users registered under a particular admin. It will fetch the geo location data for each user and will show in table format. The admin can now click on each user to see its details and the trail of that user's location points on Google Map.

LOGIN:

This is a module which is used by the admin to login into the system.

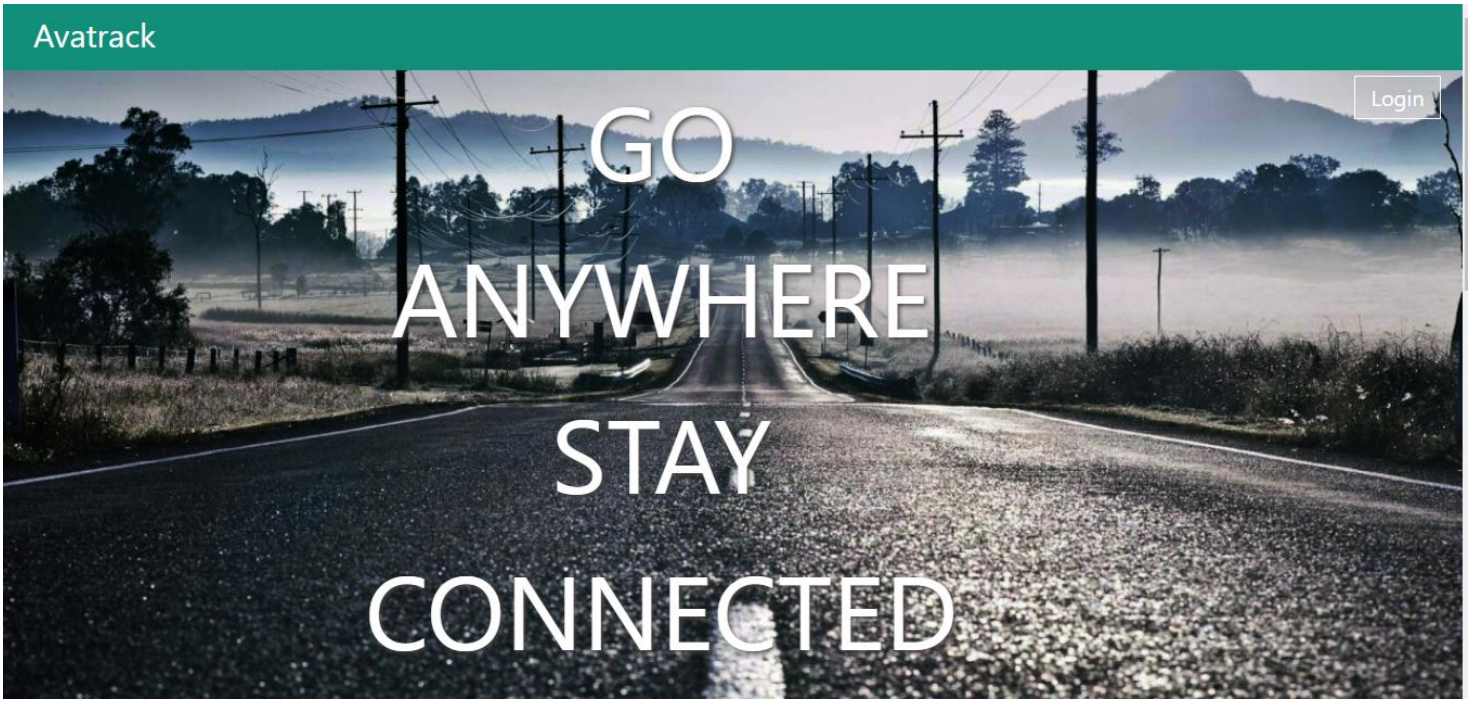
- **SIGN IN:** The admin of the system is already defined in the database so that he can login to the system. In this, USERNAME and PASSWORD are the mandatory fields.
- **FORGET PASSWORD:** It is used to set new passwords for the existing users. In this, FIRST NAME, PHONE NUMBER, NEW PASSWORD are the mandatory fields. On clicking the Submit button, the value of the NEW PASSWORD is taken and updated for the corresponding user in the database, via HTTP service, in the USER table. This will show an alert and automatically reset the form in this page. If the user does not exist in the database, then it will show an alert.
- **LOG OUT:** It is used to log out of the system. It takes the user back to the SIGN IN page.

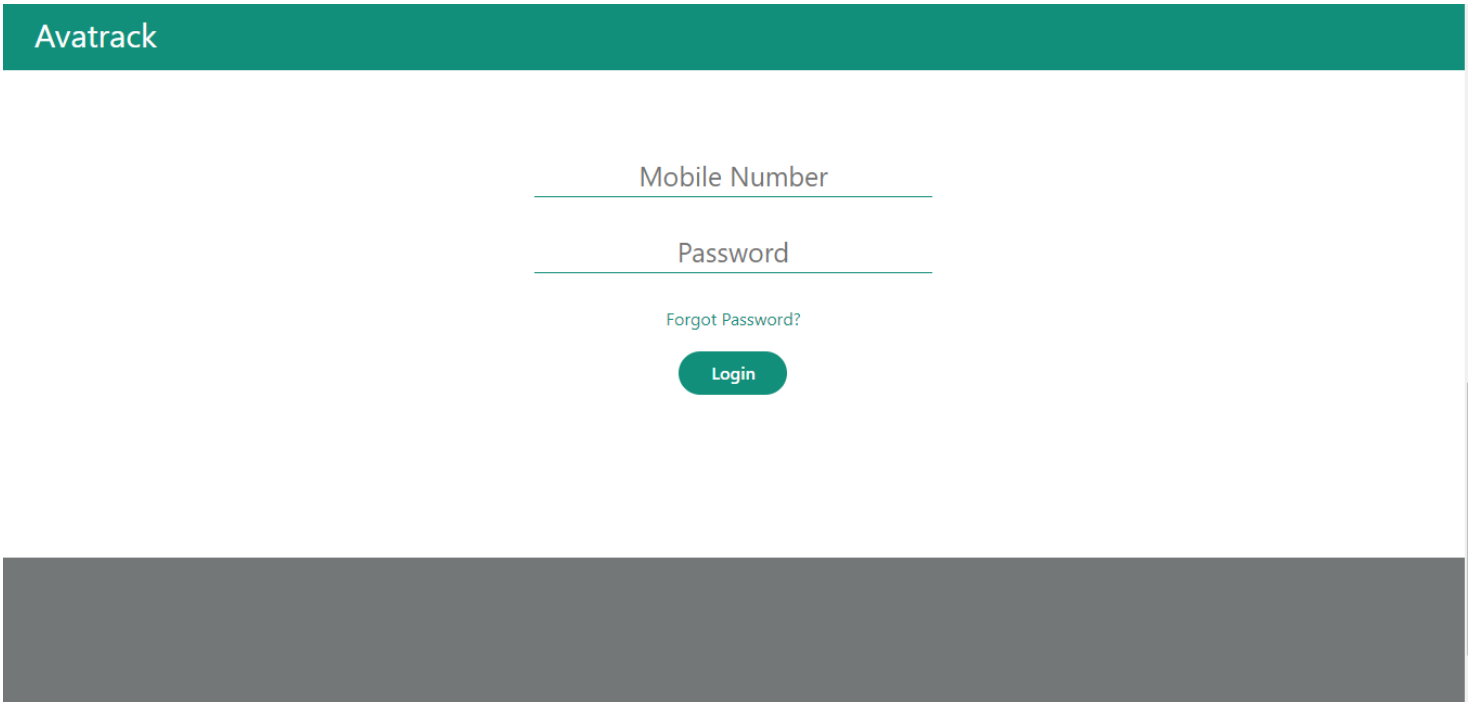
LOCATION DETAILS:

This module will show the detail of a particular selected user by an admin. The field consist of the name, the id (mobile number), and geo-location (latitude, longitude).

Now using those particular fields and with the help of Angular Google Map package it will draw the user's path from the picked up location points, and it will show the path on the Google Map

TESTING


Avatrack
Login
GO
ANYWHERE
STAY
CONNECTED


Avatrack
Mobile Number
Password
Forgot Password?
Login

Abetter

Name or Mobile

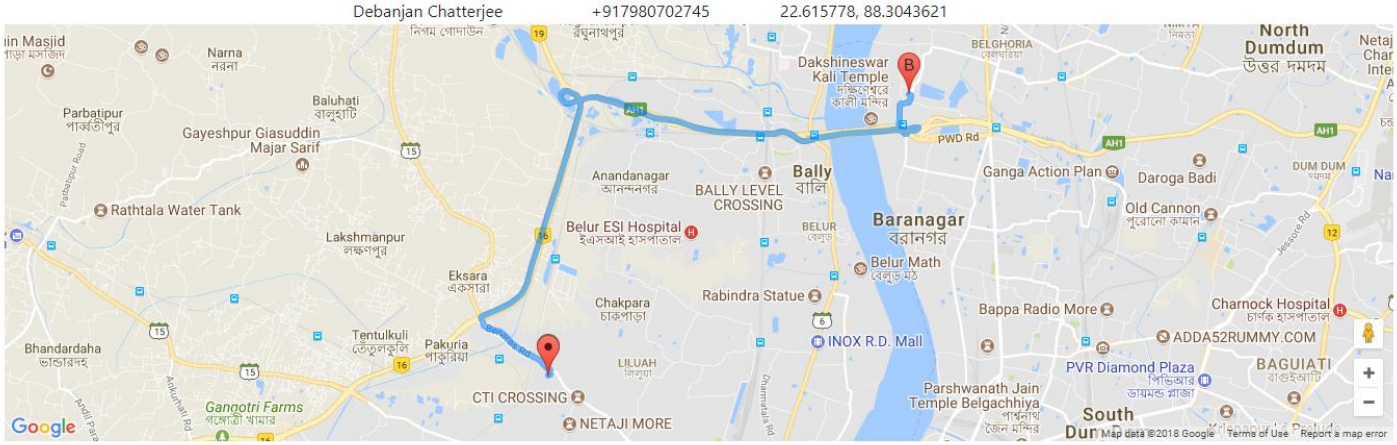
Total Users: 11

Online Users: 2

Name	Mobile	Last Location (lon, lat)
Debanjan Chatterjee	+917980702745	22.615778, 88.3043621
Milan Kamilya	+919038635778	22.615778, 88.3043621

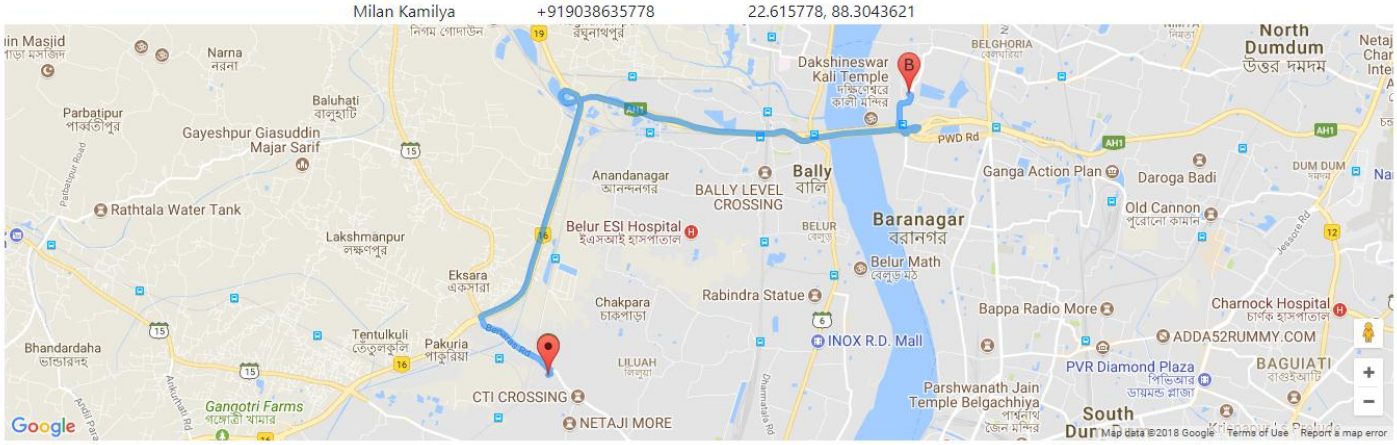
Abetter

Debanjan Chatterjee +917980702745 22.615778, 88.3043621



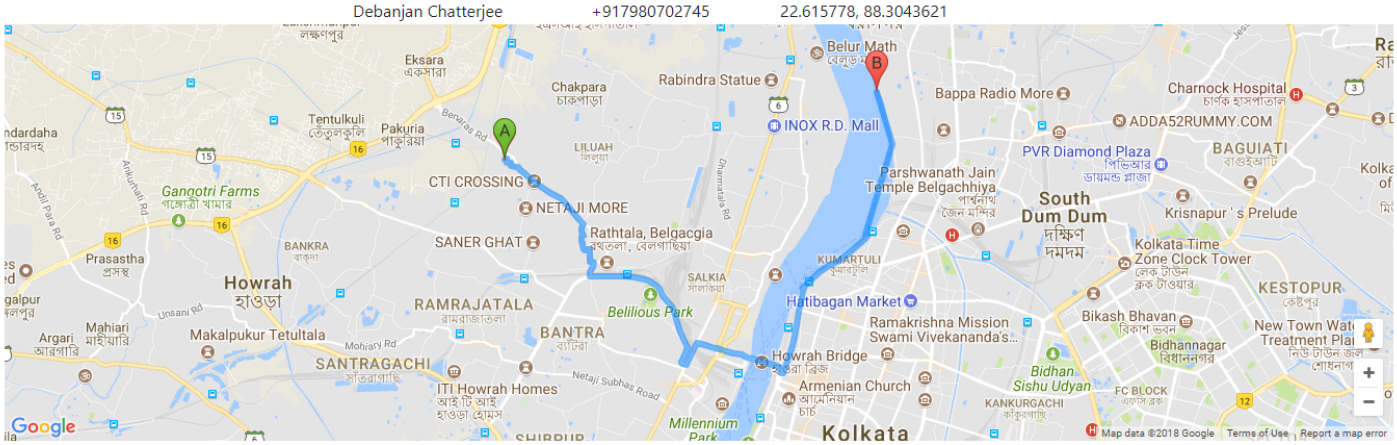
Abetter

Milan Kamilya | +919038635778 | 22.615778, 88.3043621



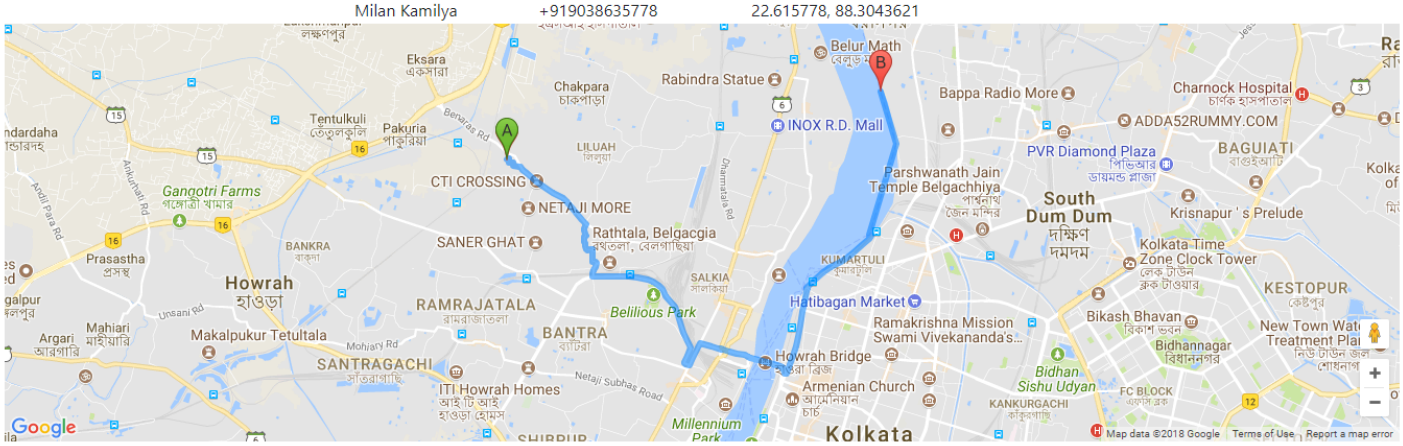
Abetter

Debanjan Chatterjee | +917980702745 | 22.615778, 88.3043621



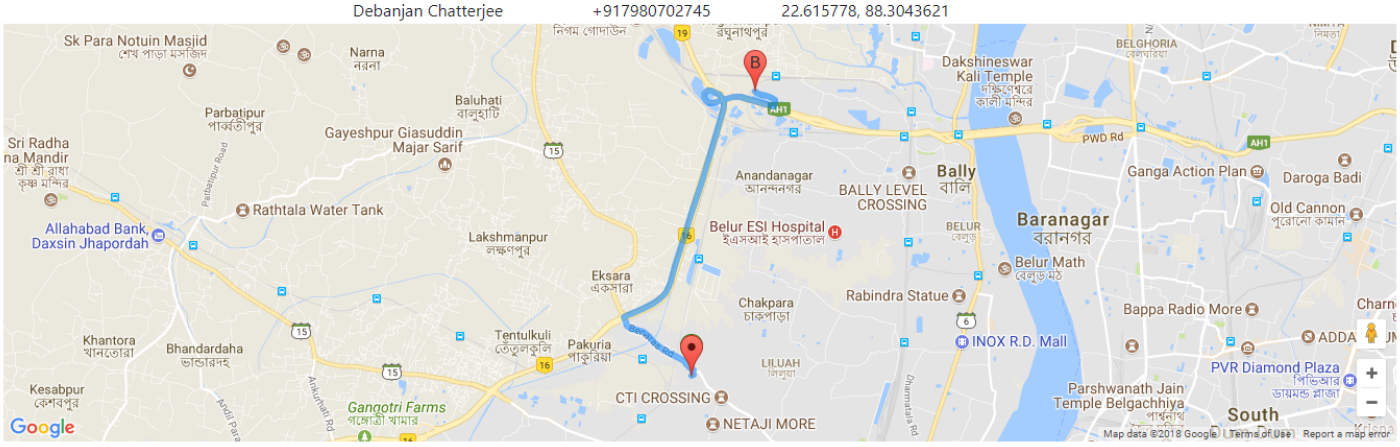
Abetter

Milan Kamilya | +919038635778 | 22.615778, 88.3043621



Abetter

Debanjan Chatterjee | +917980702745 | 22.615778, 88.3043621



MODULE DESCRIPTION

Front End (Visual Studio Code):

Modules:

1. **Login**: This Module is the starting page or the home page of this application. It includes a login section. Here the Admins can use their credentials in the form of Mobile Number and Password to get the access of all the employees under his/her control. In the front end features like smooth scrolling and parallax scrolling have been used to give a nice visual effect.
2. **Dashboard:** It is the main module of this app which contains the most of the data and features. Here we will get to see the total number of users, number of online users. The online user's list will be shown in the form of table having field as name, phone number and latitude and longitude. There are also an option of searching the users based on both name and phone number. Now admin can click on any of the user to see its details and the user's last certain period of trails on Map.
3. **Location Details:** In this module we can see the details of an user in the form of name, phone number, latitude and longitude. Along with this there is a Map section where we can see the starting and ending points of that user for the last certain amount of time and the path on which the person has travelled to.

Back End :

Services: - This is used to get or send the data from or to the database(Cloud Firestore)

1. Mockservice.ts
2. Serverservice.ts

Database(Google Cloud Firestore): - This is used to store the user's and admin's credentials along with the received geo-location data from the user app for a certain period of time.

Collection

Users, Admins, user_loc

Document

User_loc ➔Fields➔{ accuracy: number, alt: number, bearing: number, id: string, name: string, lat: number, lng: number, provider: string, speed: number, time: number }

Classes: - This is used to maintain the connection between front end and back end, also applying the business logic.

1. LoginComponent.ts
2. DashboardComponent.ts
3. LocationDetailsComponent.ts
4.

Main Class: - This is the main class which is used to run the program.

1. AppComponent.ts

SAMPLE CODE

Login:

HTML:

```
div class="container-fluid">
 <nav id="myNavBar" class="navbar fixed-top">
  <div class="row">
   <div id="companyName" class="col">
    Avatrack
   </div>
  </div>
 </nav>
 <div id="backgroundImageContainer">
  <div id="goToLogin">
   <button [ngx-scroll-to]="'loginHolder'" [ngx-scroll-to-offset]="-62"
        [ngx-scroll-to-duration]="1500" [ngx-scroll-to-easing]="'easeInOutQuint'">Login</button>
  </div>
  <div id="captionHolder" class="row">
   <div class="col-10 offset-1">
    GO
   </div>
   <div class="col-10 offset-1">
    ANYWHERE
   </div>
   <div class="col-10 offset-1">
    STAY
   </div>
   <div class="col-10 offset-1">
    CONNECTED
   </div>
  </div>
 </div>
 <div id="loginHolder">
  <div class="row">
   <!-- style="border: 1px solid black" -->
   <div class="col-10 col-md-6 offset-md-3 offset-1" >
    <div class="row">
     <form (ngSubmit)="validation()" [formGroup]="loginForm" class="formLogin col-12">
      <div class="col-12 col-md-8 offset-md-2">
       <div class="row">
        <input type="text" class="col-10 offset-1" placeholder="Mobile Number" minlength=10 maxlength=10
          required formControlName="mobile">
       </div>
       <div class="row">
        <input type="password" class="col-10 offset-1" placeholder="Password" minlength=6 maxlength=15
          required formControlName="password">
       </div>
       <div class="row">
        <label class="checkbox col-10 offset-1">
         <a href="#">Forgot Password?</a>
        </label>
       </div>
```

```
          <div class="">
            <button type="submit">Login</button>
          </div>
        </div>
      </form>
    </div>
  </div>
 </div>
 <div id="footer">

 </div>
</div>
```

Type Script

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { MockService } from '../services/mock.service';
import { FormGroup, FormBuilder, FormControl} from '@angular/forms';
@Component({
 selector: 'app-login',
 templateUrl: './login.component.html',
 styleUrls: ['./login.component.scss']
})
export class LoginComponent implements OnInit {
 loginForm: FormGroup;
 name = new FormControl;
 constructor(public routes: Router,
          public mockservice: MockService,
          private fb: FormBuilder) {}
  smoothScroll(val) {
    console.log(val);
  }
  // smoothScroll(target) {
  //   var scrollContainer = target;
  //   do { //find scroll container
  //       scrollContainer = scrollContainer.parentNode;
  //       if (!scrollContainer) return;
  //       scrollContainer.scrollTop += 1;
  //   } while (scrollContainer.scrollTop == 0);

  //   var targetY = 0;
  //   do { //find the top of target relatively to the container
  //       if (target == scrollContainer) break;
  //       targetY += target.offsetTop;
  //   } while (target = target.offsetParent);

  //   var scroll = function(c, a, b, i) {
  //       i++; if (i > 30) return;
  //       c.scrollTop = a + (b - a) / 30 * i;
  //       setTimeout(function(){ scroll(c, a, b, i); }, 20);
  //   }
  //   // start scrolling
  //   scroll(scrollContainer, scrollContainer.scrollTop, targetY, 0);
  // }
```

```
  validation() {
    const values = this.loginForm.value;
    if( this.loginForm.valid ) {

      // this.store.dispatch(new AuthAction.Login(values['mobile'], values['password']));
      for( let i = 0; i < this.mockservice.adminList.length; i++) {

        if(values['mobile'] === this.mockservice.adminList[i].mobile &&
          values['password'] === this.mockservice.adminList[i].password) {
            this.routes.navigate(['/dashboard']);
        }

      }
    }
  }

  ngOnInit() {
    const mobile = '';
    const password = '';
    this.loginForm = this.fb.group({
      'mobile':[mobile],
      'password':[password]
    });
  }

}
```

SCSS:

```
.container-fluid {
    padding-left: 0px !important;
    padding-right: 0px !important;
}
// nav bar
#myNavBar {
    background-color: rgb(18, 143, 122);
    padding: 10px;
}
#companyName {
    font-size: 30px;
    color: white;
    margin-left: 20px;
}
//image container
#backgroundImageContainer {
    background: url("../../assets/images/bckgrndimg1.jpg") no-repeat center;
    background-size: cover;
    margin-top: 40px;
    height: 890px;
    background-attachment: fixed;
}
#goToLogin {
    float: right;
    margin-top: 30px;
```

```
    margin-right: 20px;
    padding: 0px;
    font-size: 20px;
    button {
        width: 80px;
        height: 40px;
        background-color: transparent;
        border: 1px solid white;
        color: white;
        cursor: pointer;
        position: relative;
        z-index: 3
    }
}
#captionHolder {
    color: white;
    text-align: center;
    margin-right: 0px;
    font-size: 96px;
    text-shadow: 2px 2px 6px rgb(68, 65, 65);
    //transform: translateX(60px);
    padding-top: 15px;
    position: inherit;
    z-index: 1;
}
// login container
#loginHolder {
    padding: 80px 0px 150px 0px;
}
.formLogin {
    text-align: center;
    input {
        border: none;
        border-bottom: 1px solid rgb(18, 143, 122);
        margin-bottom: 30px;
        text-align: center;
        font-size: 25px;
        outline: none;
        color: rgb(18, 143, 122);
    }
    button {
        border: none;
        border-radius: 25px;
        margin-top: 10px;
        background-color: rgb(18, 143, 122);
        color: white;
        font-weight: 500;
        height: 40px;
        width: 100px;
        cursor: pointer;
    }
    a {
        cursor: pointer;
        color: rgb(18, 143, 122);
    }
}
//footer
#footer {
```

```
  width: 100%;
  height: 200px;
  background-color: rgb(116, 119, 119);
}
//media queries
@media only screen and (max-width: 768px) {
  #captionHolder {
    font-size: 72px;
    padding-top: 160px;
  }
}
@media only screen and (max-width: 576px) {
  #captionHolder {
    font-size: 53px;
    padding-top: 139px;
  }
}
@media only screen and (max-width: 420px) {
  #captionHolder {
    font-size: 38px;
    padding-top: 145px;
  }
}
```

Dashboard:

HTML:

```
<div class="container-fluid">
 <nav id="myNavBar" class="navbar fixed-top">
  <div id="navContent" class="row">
   <div id="companyName" class="col-12 col-sm-3">
    Abetter
   </div>
   <div id="searchArea" class="col-12 col-sm-9" [ngClass]="{'show': !showMap, 'hide': showMap}">
    <div class="row">
     <div id="searchField" class="col-10">
       <i class="fa fa-search"></i>
       <input type="text" [(ngModel)]="searchText" placeholder="Name or Mobile">
     </div>
     <div class="col-2">
      <div  id="rightField" class="row">
       <div id="adminField">
         <i class="fa fa-user"></i>
       </div>
      </div>
     </div>
    </div>
   </div>
  </div>
 </nav>
 <div id="mainContainer" class="row mainContainer" [ngClass]="{'show': !showMap, 'hide': showMap}">
  <div class="col-12 col-md-3">
    <h2>Total Users: {{employeeCount}}</h2>
    <h2>Online Users: {{onlineEmployeeCount}}</h2>

  </div>
  <div class="col-12 col-md-9">
```

```
    <div class="row">
      <table class="col-10 offset-1">
        <tr id="headRow">
          <th class="tableRow" >Name</th>
          <th class="tableRow" >Mobile</th>
          <th class="tableRow" >Last Location<br> (lon, lat)</th>
        </tr>
        <tr id="valueRow" *ngFor="let employee of employeeList | filter: searchText; let i = index"
(click)="showLocation(i)">
          <td class="tableData" >{{employee.name}}</td>
          <td class="tableData">{{employee.id}}</td>
          <td class="tableData">{{employee.lat}}, {{employee.lng}}</td>
        </tr>
      </table>

    </div>
  </div>
 </div>
 <app-location-details class="row mainContainer" [empList]="oneEmployeeInfo"
                [ngClass]="{'show': showMap, 'hide': !showMap}">

 </app-location-details>
</div>

<!--  -->
```

Type Script:

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { MockService } from '../services/mock.service';
import { FormGroup, FormBuilder, FormControl} from '@angular/forms';
@Component({
 selector: 'app-login',
 templateUrl: './login.component.html',
 styleUrls: ['./login.component.scss']
})
export class LoginComponent implements OnInit {
 loginForm: FormGroup;
 name = new FormControl;
 constructor(public routes: Router,
          public mockservice: MockService,
          private fb: FormBuilder) {}
  smoothScroll(val) {
    console.log(val);
  }
  // smoothScroll(target) {
  //   var scrollContainer = target;
  //   do { //find scroll container
  //       scrollContainer = scrollContainer.parentNode;
  //       if (!scrollContainer) return;
  //       scrollContainer.scrollTop += 1;
  //   } while (scrollContainer.scrollTop == 0);

  //   var targetY = 0;
  //   do { //find the top of target relatively to the container
  //       if (target == scrollContainer) break;
  //       targetY += target.offsetTop;
```

```
  //   } while (target = target.offsetParent);

  //   var scroll = function(c, a, b, i) {
  //       i++; if (i > 30) return;
  //       c.scrollTop = a + (b - a) / 30 * i;
  //       setTimeout(function(){ scroll(c, a, b, i); }, 20);
  //   }
  //   // start scrolling
  //   scroll(scrollContainer, scrollContainer.scrollTop, targetY, 0);
  // }

 validation() {
  const values = this.loginForm.value;
  if( this.loginForm.valid ) {

    // this.store.dispatch(new AuthAction.Login(values['mobile'], values['password']));
    for( let i = 0; i < this.mockservice.adminList.length; i++) {

      if(values['mobile'] === this.mockservice.adminList[i].mobile &&
        values['password'] === this.mockservice.adminList[i].password) {
          this.routes.navigate(['/dashboard']);
      }

    }
  }
 }

 ngOnInit() {
  const mobile = '';
  const password = '';
  this.loginForm = this.fb.group({
    'mobile':[mobile],
    'password':[password]
  });
 }

}
```

SCSS:

```
.container-fluid {
   padding-left: 0px !important;
   padding-right: 0px !important;
}
// nav bar
#myNavBar {
   background-color: rgb(18, 143, 122);
   padding: 10px;
}
#companyName {
   font-size: 30px;
   color: white;
   margin-left: 20px;
}
//image container
#backgroundImageContainer {
   background: url("../../assets/images/bckgrndimg1.jpg") no-repeat center;
   background-size: cover;
   margin-top: 40px;
```

```
    height: 890px;
    background-attachment: fixed;
}
#goToLogin {
    float: right;
    margin-top: 30px;
    margin-right: 20px;
    padding: 0px;
    font-size: 20px;
    button {
        width: 80px;
        height: 40px;
        background-color: transparent;
        border: 1px solid white;
        color: white;
        cursor: pointer;
        position: relative;
        z-index: 3
    }
}
#captionHolder {
    color: white;
    text-align: center;
    margin-right: 0px;
    font-size: 96px;
    text-shadow: 2px 2px 6px rgb(68, 65, 65);
    //transform: translateX(60px);
    padding-top: 15px;
    position: inherit;
    z-index: 1;
}
// login container
#loginHolder {
    padding: 80px 0px 150px 0px;
}
.formLogin {
    text-align: center;
    input {
        border: none;
        border-bottom: 1px solid rgb(18, 143, 122);
        margin-bottom: 30px;
        text-align: center;
        font-size: 25px;
        outline: none;
        color: rgb(18, 143, 122);
    }
    button {
        border: none;
        border-radius: 25px;
        margin-top: 10px;
        background-color: rgb(18, 143, 122);
        color: white;
        font-weight: 500;
        height: 40px;
        width: 100px;
        cursor: pointer;
    }
    a {
```

```
      cursor: pointer;
      color: rgb(18, 143, 122);
    }
}
//footer
#footer {
    width: 100%;
    height: 200px;
    background-color: rgb(116, 119, 119);
}
//media queries
@media only screen and (max-width: 768px) {
    #captionHolder {
        font-size: 72px;
        padding-top: 160px;
    }
}
@media only screen and (max-width: 576px) {
    #captionHolder {
        font-size: 53px;
        padding-top: 139px;
    }
}
@media only screen and (max-width: 420px) {
    #captionHolder {
        font-size: 38px;
        padding-top: 145px;
    }
}
```

Location Details:

HTML:

```
<div class="container-fluid">
  <div>
  <nav id="myNavBar" class="navbar fixed-top">
    <div id="navContent" class="row">
      <div id="companyName" class="col-12 col-sm-3">
        Abetter
      </div>
      <!-- <div id="searchArea" class="col-12 col-sm-9" [ngClass]="{'show': !showMap, 'hide': showMap}">
        <div class="row">
          <div id="searchField" class="col-10">
            <i class="fa fa-search"></i>
            <input type="text" [(ngModel)]="searchText" placeholder="Name or Mobile">
          </div>
          <div class="col-2">
            <div  id="rightField" class="row">
              <div id="adminField">
                <i class="fa fa-user"></i>
              </div>
            </div>
          </div>
        </div>
      </div> -->
    </div>
```

```
  </nav>
</div>
  <div id="employeeData" *ngIf="oneEmployee; else loading">
    <table class="col-6 offset-3">
      <tr id="headRow">
        <th class="tableRow" ><i class="fa fa-user"></i></th>
        <th class="tableRow" ><i class="fa fa-mobile"></i></th>
        <th class="tableRow" ><i class="fa fa-map-marker"></i><br></th>
      </tr>
      <tr id="valueRow">
        <td class="tableData" >{{oneEmployee.name}}</td>
        <td class="tableData">{{oneEmployee.id}}</td>
        <td class="tableData">{{oneEmployee.lat}}, {{oneEmployee.lng}}</td>
      </tr>
    </table>
  </div>
  <agm-map [latitude]="lat" [longitude]="lng" [zoom]="zoom">
    <agm-marker [latitude]="lat" [longitude]="lng"></agm-marker>
      <agm-direction *ngIf="dir; else loading" [origin]="dir.origin" [destination]="dir.destination"></agm-direction>
  </agm-map>
  <ng-template #loading>Loading.....</ng-template>
</div>
```

Type Script:

```
import { Component, OnInit, Input, ViewChild, ElementRef } from '@angular/core';
import { AgmCoreModule } from '../../../node_modules/@agm/core';
import { AgmDirectionModule } from '../../../node_modules/agm-direction';
// import { GoogleMap } from '@agm/core/services/google-maps-types';
import { ServerService } from '../services/server.service';
import { MockService } from '../services/mock.service';


// const firebase = require("firebase");
// // Required for side-effects
// require("firebase/firestore");

// firebase.initializeApp({
//   apiKey: 'API_KEY_HERE',
//   authDomain: 'DOMAIN_NAME_HERE',
//   projectId: 'PROJECT_ID_HERE'
// });
// var db = firebase.firestore();
interface employee{
  accuracy: string;
  alt: number;
  bearing: number;
  id: string;
  lat: number;
  lng: number;
  provider: string;
  speed: number;
  time: number;
  name: string;
}
```

```
@Component({
  selector: 'app-location-details',
  templateUrl: './location-details.component.html',
  styleUrls: ['./location-details.component.scss']
})
export class LocationDetailsComponent implements OnInit {

  public lat: number;
  public lng: number;
  public dir: any = undefined;
  public zoom:number=15;

  public oneEmployee: employee;
  public mapUrl: string;
  public emp: any[];
  public dummyMobile = '+917980702745';
  public userObj: any={};
  constructor(public serverservice: ServerService,
            public mockservice: MockService) {
   }
  @Input() empList: string[];
  clickMe() {
    // db.collection("user_loc").where("id","==", this.dummyMobile)
    //   .get().then((querySnapshot) => {
    //       querySnapshot.forEach((doc) => {
    //           console.log(`${doc.id} => ${doc.data()}`);
    //           this.userObj = doc.data();
    //           console.log(this.userObj.id,this.userObj.lat);
    //       });
    //   });
  }
  ngOnInit() {
    console.log('one employee', this.mockservice.oneEmployeeDetails);
    this.oneEmployee = this.mockservice.oneEmployeeDetails;
    this.lat = this.oneEmployee.lat;
    this.lng = this.oneEmployee.lng;
    this.dir = {
      // center: {lat: this.lat, lng:this.lng},
      origin: {lat: this.lat, lng: this.lng},
      destination: {lat: this.lat + 0.0099999,lng: this.lng + 0.244257}
    }
    this.serverservice.getServers(this.dummyMobile)
      .subscribe(
        (response) => console.log(response),
        (error) => console.log(error)
      );
  }
}
```

SCSS:

```
#myNavBar {
    background-color: rgb(18, 143, 122);
    padding: 10px;
    // .fa-user, .fa-search {
    //     font-size: 25px !important;
    //     color: white;
    // }
```

```
  //margin-bottom: 80px;
}
#navContent {
  width: 100%;
}
#companyName {
  font-size: 30px;
  color: white;
  padding-left: 40px;
}
#employeeData {
  margin-top:100px;
}

//map section
#iFrameHolder {
  padding-top: 35px;
  align-items: center;
  text-align: center;
}
agm-map {
  height:400px;
  background: center;
}
.fa-user, .fa-mobile, .fa-map-marker {
  font-size: 25px !important;

}
```

Services:

```
import { Injectable } from "@angular/core";
interface employee{
  accuracy: string;
  alt: number;
  bearing: number;
  id: string;
  lat: number;
  lng: number;
  provider: string;
  speed: number;
  time: number;
  name: string;
 }
@Injectable()
export class MockService{
  employeeList: any[] = [
    {"name": "Debanjan Chatterjee", "mobile": "9007319674", "longitude": "87.78", "lattitude": "34.45"},
    {"name": "Milan Kamilya", "mobile": "8334814524", "longitude": "182.40", "lattitude": "14.45"},
    {"name": "Rakesh Jha", "mobile": "9331244524", "longitude": "10.606619", "lattitude": "35.856737"}
  ];
  adminList: any[] = [
    {"mobile": "9007319674", "password": "abcd1234"}
  ]
  oneEmployeeDetails: employee;
}
```

Server Service:

```
import { Injectable } from '@angular/core';
import { Http } from '@angular/http';
@Injectable()
export class ServerService {
   public baseUrl = 'BASE_URL_HERE';
   constructor(private http: Http) {

   }
   // storeServers(servers: any[]){
   //     return this.http.post('SERVER_URL_HERE', servers);
   // }
   getServers(dummyMobile){
      let finalUrl = this.baseUrl + '(default)/documents/' + dummyMobile;
      return this.http.get(finalUrl);
   }
}
```

app.module.ts:

```
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { NgModule } from '@angular/core';
import { ROUTING } from './app.routing';
import { ReactiveFormsModule } from '@angular/forms';
import { ScrollToModule } from '@nicky-lenaers/ngx-scroll-to';

import { Store, StateObservable } from '@ngrx/store';
import { AppComponent } from './app.component';
import { LoginComponent } from './login/login.component';
import { DashboardComponent } from './dashboard/dashboard.component';
import { MockService } from './services/mock.service';
import { FilterPipe } from './app.pipe';
import { LocationDetailsComponent } from './location-details/location-details.component';
import { ServerService } from './services/server.service';
import { HttpModule } from '@angular/http';

import { AgmCoreModule } from '@agm/core';           // @agm/core
import { AgmDirectionModule } from 'agm-direction';   // agm-direction

@NgModule({
  declarations: [
    AppComponent,
    LoginComponent,
    DashboardComponent,
    FilterPipe,
    LocationDetailsComponent
  ],
  imports: [
    BrowserModule,
    ROUTING,
    FormsModule,
    ReactiveFormsModule,
    ScrollToModule.forRoot(),
    HttpModule,
    AgmDirectionModule ,
    AgmCoreModule.forRoot({ // @agm/core
      apiKey: 'API_KEY_HERE'
    })
```

```
  ],
  providers: [MockService, Store, ServerService,AgmCoreModule,AgmDirectionModule],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

app.pipe.ts:

```
import { Pipe, PipeTransform } from '@angular/core';
import { MockService } from '../app/services/mock.service'
// import { MockExecutor } from 'protractor/built/driverProviders';
@Pipe({name: 'filter'})
export class FilterPipe implements PipeTransform {
  transform(items: string[], searchText: string): string[] {
    let empName: string;
    let empMob: string;
    if(!items) return [];
    if(!searchText) {
      return items;
    } else {
      searchText = searchText.toLowerCase();
      return items.filter((value: any, index, array) => {
        empName = value.name;
        empMob = value.mobile;
        return ((empName.toLowerCase().includes(searchText)) || (empMob.includes(searchText)));
      });
    }
  }
}
```

app.routing.ts:

```
import { Routes, RouterModule} from '@angular/router';
import { ModuleWithProviders } from '@angular/core/src/metadata/ng_module';
import { LoginComponent } from './login/login.component';
import { DashboardComponent } from './dashboard/dashboard.component';
import { LocationDetailsComponent } from './location-details/location-details.component';

const AppRoutes: Routes = [
    {path: 'login', component: LoginComponent},
    {path: 'dashboard', component: DashboardComponent},
    {path: 'dashboard/details', component: LocationDetailsComponent},
    {path: '', redirectTo: 'login', pathMatch: 'full'}
];
export const ROUTING: ModuleWithProviders = RouterModule.forRoot(AppRoutes);
```

FUTURE SCOPE:

The implementation of this Location Tracker could be huge. As it can be easily developed in other platforms like Android and iOS, the flexibility increases. We can still improve this app by showing other various data retrieved from the geo-location API. It's not only just the location and the trailing path, rather we can show graphs and charts regarding the altitude, speed, etc. That could be an interesting feature, and some sectors can have huge demand of such an easy to use app, which will show the exact thing they need.

The most importantly, it can be expanded for the use of security. We always get worried about our loved ones, as soon as they step out of the home. By using this app we can really track them, and in some condition we can get some panic message from the user to the admins, i.e in this case the family members. It too can call for medical help, inform police controls, fire brigade and other various mediums of emergency situations.

CONCLUSION

There is always scope for improvement, especially in the developing part, where we can constantly add features along with maintaining the fluency of the application in all platforms. The user interface is kept as simple and easy as possible, just according to the requirement.

The application is built in such a way that any minor requirement changes can be fit so easily without much hassle and changes in code. It is the best way to develop certain applications which can be expanded a lot in future, and as its scope of using vast, therefore the planning of from the beginning should be skeptical.

It's all about the thinking and the philosophy to stay remain connected with the loved ones, the ones who care for us, even the ones who need to check if we are on right place during work or not, everything. Being connected is the key and to reach to the top of this concept. We should be willing enough to use this and share our status, to make the connections more transparent.

REFERENCES

The following links were used by us for learning:-

- https://stackoverflow.com/
- https://www.google.com/
- https://angular.io/guide/architecture
- https://firebase.google.com/docs/
- https://angular-maps.com/guides/getting-started/
- https://developers.google.com/maps/documentation/javascript/tutorial

And other books like:

- Beginning Angular with Typescript – by Greg Lim