**Machine Failure Predictive Maintenance System**

**Ajay Shankar**  (20Z305)

**Shiva Prakash S**  (20Z342)

**Yuvaraj Kesavan P**  (20Z356)

**Chirenjeevi M**  (20Z361)

**Siva M**  (21Z466)

GUIDE: THIRUMAHAL R

### 19Z620 - INNOVATION PRACTICES

Dissertation submitted in partial fulfillment of the requirements for the degree of

**BACHELOR OF ENGINEERING**

**Branch: COMPUTER SCIENCE AND ENGINEERING**



MARCH 2023

### DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

**PSG COLLEGE OF TECHNOLOGY**

(Autonomous Institution)

COIMBATORE – 641 004

**COMPUTER SCIENCE & ENGINEERING**

**MOBILE TRACKING SYSTEM**

Bonafide record of work done by

**Aswath Harish J** (20Z308)

**Shiva Prakash P** (20Z342)

**Yuvaraj Kesavan P** (20Z356)

**Chirenjeevi M** (20Z361)

**Siva M** (21Z466)

Dissertation submitted in partial fulfillment of the requirements for the degree of

### **BACHELOR OF ENGINEERING**

BRANCH: COMPUTER SCIENCE AND ENGINEERING

of Anna University

March 2023

**Mrs. R. Thirumahal Dr. G. Sudha Sadasivam** Faculty Guide Head of the Department

Assistant Professor Department of Computer

Department of Computer Science Science & Engineering

& Engineering

Certified that the candidate was examined in the viva voce examination held on\_\_\_\_\_\_\_\_\_\_\_\_\_

Internal Examiner External Examiner

**CONTENTS**

**CHAPTER PageNo.**

**Acknowledgment……………………………………………………………….………….5**

**Synopsis……………………………………………………………………………………..6**

**List of figures.……………………………………..………………………………………..7**

1. **INTRODUCTION………………………………………………………………….......…….8**
2. **LITERATURE SURVEY...…………………………………………………………………..9**
3. **SYSTEM REQUIREMENTS……………………………………………………………….16**
   1. HARDWARE REQUIREMENTS 16
   2. SOFTWARE REQUIREMENTS 16
   3. FRONT END 16
   4. BACK END 17
4. **SOFTWARE DESIGN…………………………………………………………………..…..18**
   1. API 18
   2. MODEL WORKFLOW18
5. **SYSTEM IMPLEMENTATION……………………………………………………………...21**
   1. **WEB INTERFACE………………………………………………………………….21**
      1. **SUPER ADMIN…………………………………………………………………...21**

5.1.1.1 COMPANY 21

5.1.1.2 EMPLOYEE 22

5.1.1.3 VIEW DETAILS 22

5.1.2 **ADMIN……………………………………………………………………………..23**

5.1.2.1 EMPLOYEE 24

5.1.2.2 TRACkING-DETAILS 25

5.1.2.3 MESSAGE 25

* 1. **APP INTERFACE…………………………………………………………………..26**
     1. LOGIN AND LOGOUT 26
     2. CHECK-IN 27
  2. **LOCATION TRACKING…………………………………………………………...28**
  3. **GEO-FENCING……………………………………………………………………..28**
  4. **IDEAL TIME ALERT……………………………………………………………….28**
  5. **REPORT…………………………………………………………………………….29**

1. **CONCLUSION……………………………………………………………………………….30**
2. **APPENDIX 1…………………………………………………………………………………31**
   1. ABBREVIATION 31
3. **BIBLIOGRAPHY…………………………………………………………………………….32**

**ACKNOWLEDGMENT**

I wish to express my sincere gratitude to our beloved Principal Dr.K.Prakashan for providing an opportunity and necessary facilities in carrying out this dissertation work.

I also wish to express my sincere thanks to Dr.S.Vijayalakshmi -Faculty Guide- Assistant Professor- Department of Computer Science and Engineering, for the encouragement and support that he extended towards this dissertation work.

My sincere thanks are due to Mrs.R.Thirumahal - Project Guide - Assistant Professor- Computer Science and Engineering, whose valuable guidance and continuous encouragement throughout the course made it possible to complete this dissertation work well in advance.

Finally, I would like to thank all my student colleagues, and staff members in the Computer Science Department without whom this dissertation work would not have been completed successfully.

**SYNOPSIS**

In the modern world, there are lots of organizations that can’t trust their employees. Consider a market service, and they can’t trust their employee blindly on where they traveled the whole day and did service. Similarly, every employee does give excuses on why they couldn’t call, saying the battery is down or they couldn’t reach them. Managers of these kinds of organizations feel it is challenging to track every employee's live location every day. This gives them pressure while performing many tasks in collecting their travel history, their attendance timing, tasks they did every day, and so on.

**LIST OF FIGURES**

| **Figures no** | **Figure Description** |
| --- | --- |
| Fig 4.1 | Sequence Diagram |
| Fig 5.1 | Adding Company Details |
| Fig 5.2 | Adding Employee Details |
| Fig 5.3 | Dashboard |
| Fig 5.4 | Adding Employee Details |
| Fig 5.5 | Tracking Details |
| Fig 5.6 | Login Page |
| Fig 5.7 | Check-In and Check-Out Page |

**CHAPTER 1**

**INTRODUCTION**

The goal of this project is to offer a real-time location tracking solution that is both easily accessible and affordable, as well as scalable to meet the needs of businesses and individuals around the world.

Appstrack provides location intelligence through real-time tracking, which offers high accuracy and privacy to its users. This feature enables businesses and individuals to keep track of their assets, monitor employee movements, and optimise their operations for greater efficiency and profitability.

The application is designed to be user-friendly, making it easy for anyone to get started with real-time location tracking, even if they have no prior experience with GPS or location services.

The abstract is also highly scalable, meaning that it can be adapted to meet the needs of businesses and individuals of all sizes. The platform can be customised to suit the unique needs of each user, and its flexible pricing plans make it affordable for businesses and individuals of all budgets.

In summary, Appstrack aims to provide businesses and individuals with a powerful, yet affordable, location-tracking solution that is easily accessible and scalable. With its real-time tracking and location intelligence features, Appstrack offers a reliable and efficient way to monitor assets, optimise operations, and increase productivity.

**CHAPTER - 2**

**LITERATURE SURVEY**

[1] We studied how to install ionic in our systems. We understood what CLI is. Moreover, we tried using their app wizard to set up and installing Ionic CLI in our terminal. We got the commands to start an app by using their pre-made app templates and also learnt how to run an ionic app from our terminal.

[2] We learnt how to use the content area and modify it based on our needs. We understood their usage, where we can use header and footer, how to display it as fullscreen content, and how to assign an element in a fixed position. We understood this is styled using CSS components - left, right, top and bottom. We understood how to use the scroll feature and how to code it so that absolute positioning doesn’t change. We learned the various themes available, which are basically done with the help of CSS. Fullscreen has been used for us in most of the areas where we set the value as true. It takes boolean values, by default it is false. We can scroll in the direction we need. So, we can set it as ScrollX or ScrollY.

[3] Ionicons is an open-source icon set made exclusively for the ionic framework. There are a lot of icons available and it can be taken either as an outline or as a filled icon or even as a sharp icon. Outline icon has only the icon structure and there is no filling of colours in it, whereas it is filled in an filled icon. This open-source library is very useful for icons and there are almost 1500 icons available in it. We imported a lot of icons which are necessary for our application.

[4] Our team required various custom layouts. So, we understood how to use the ion grid. We learnt how to fix the grid size based on the minimum width. We learnt the three ways of setting the column size - Content-based, specified and Responsive.. We tried all these methods and found the difference and their impact in our emulator. The basic inferences we found out was, for Content-based size, setting the size to "auto" the column can size itself based on the natural width of its content, whereas if set the size of a column, then that is specified size. Responsive comes into play where the size changes the column width for all breakpoints. This area covered a lot of topics and was very useful in the alignment, setting the offset values, push and pull and customisation which includes padding in it.

[5] We understood how to use lists and what it is. Our inference was A list is a collection of multiple rows of items that can include different types of content, such as text, buttons, toggles, icons, and thumbnails. Typically, the items in a list have similar data content, such as images and text. Lists in Ionic also support various interactions, such as swiping items to reveal options, dragging to reorder items within the list, and deleting items. These interactions allow users to perform various actions on the list items, making a list more dynamic and responsive to user input. We use lists in various areas in our application. We learnt about their three properties - inset, lines, mode and their method, which is closeSlidingItems. This method generally closes any open sliding item.

[6] We found ion-icon very useful for using the icons imported from ionicons [3]. We used it to display every imported icon in our application and it is flexible to use where we can style it according to our needs and also change its colour if needed.

[7] For our application, ion inputs are frequently used while developing forms that ask for user input. We learnt how users could type data into an ion input to enter it. It offered a user-friendly interface for entering data, making them particularly helpful for mobile devices. It was very useful since we applied to many different form fields, including text, email, password, number, and more. We learnt how to add validation rules to ion inputs to ensure that users enter data in the suitable format and to suit the particular requirements of our application. It is a flexible and effective tool for developing user-friendly forms.

[8] We learnt about ion modal, which is frequently used in our mobile applications to display extra information or nudge users to take action. We found it as a flexible tool that is applied in various situations, such as login or registration forms. When utilised properly, we understood that Ion modal can improve the user experience by offering a user-friendly interface that enables users to engage with the programme without being diverted by other information A user can concentrate on selecting a choice when presented with a confirmation dialogue in an ion modal. To enhance the user experience and promote higher engagement with the program, it's important to use ion modals judiciously. Overusing pop-up boxes can burden users, leading to irritation and potentially reducing their overall engagement with the program. We felt that ion modal is a potent tool for showing more data or inciting users to act in our mobile application.

[9] We learnt about Ion nav which is a user interface element that offers navigational features for our mobile application.We used Ion nav to transport users between different areas of our application by navigating between pages or views. We learnt how to alter the look, feel, and behaviour of the navigation bar in ion nav to suit the particular requirements of our application.We found out that Ion nav frequently works in tandem with other UI elements like ion menu and ion tabs to offer a thorough navigation experience.

[10] We used ionic refresher where Users can refresh content in our mobile applications using the Ionic refresher, a UI component. To provide a refresh functionality, we used it with other UI elements like ion-list or ion-scroll. We learnt how The layout and functionality of the Ionic refresher component can be changed to suit the unique requirements of our application.We learned that It offers a variety of techniques for programmatic control, enabling it to initiate a refresh in response to user actions or application logic. The Ionic refresher component notifies our application via an event that the content needs to be refreshed when a user requests one. The software then updates the user's material by retrieving fresh data. This is made possible in our application only using ionic refresher.

[11] We learnt about a UI element called Ion Router that offers routing capabilities in our mobile applications. It offers ways to navigate between these routes programmatically and helps us to define a set of routes that map to various pages or views within our application.We tailored Ion Router in such a way that it meets the particular requirements of our application, including configuring the routing component's behaviour and specifying transition animations, among other things. We learnt how to use the Ion Router to load the appropriate page or view and modify our application's URL when a user selects a new route. This UI component, Ion Router offers a strong tool for improving the user experience in our mobile application overall. We learnt how to use it properly, so that it can make it simple for our users to move about the app's various areas and locate the information they need. We need to design the routing clearly to prevent user confusion and a bad user experience.

[12] The Ionic framework offers Ion Tabs as a user interface element for developing tabbed user interfaces in mobile apps. We learnt the ability of Ion Tabs which is to make a collection of tabs, each of which stands in for a different view or page in the application. The appropriate view is shown when a user selects a tab, enabling users to switch between the app's many sections. Our inference was that using ion tabs has a lot of benefits. First , Ion Tabs are simple to implement and have an easy-to-understand API, making them simple to use. The content that will be displayed on each tab can be easily defined when creating a tabbed interface. Next, Ion Tabs are highly customisable, enabling us to control the tabs' design and functionality. The text, icons, and colours of the tabs can all be altered to match the look and feel of their particular application. Another great advantage is Integration with other Ionic components. It offers a thorough navigation and user interface experience, Ion Tabs may be used in conjunction with other Ionic components like Ion Router . And lastly, Ion Tabs are made effective and efficient, with optimised rendering and seamless tab transitions which benefited our application while developing.

[13] With the help of the robust and adaptable cross-platform development tool Ionic Capacitor, we can create desktop, mobile, and web applications using web technologies like HTML, CSS, and JavaScript. It is made to integrate easily with React and maintained the same way we maintain the ionic application. The primary advantage we found was access to native features. Ionic Capacitor offers a straightforward and user-friendly API for gaining access to functions that are native to a device, such as the camera, geolocation, and storage which is used in our application. As a result, it helps us create high-performing, feature-rich apps that fully utilise the hardware's capabilities. We found it as an excellent choice when used with ionic.

[14] Local notifications are a feature of Capacitor that enables it to schedule and display notifications to users directly from the user's device. These notifications can be triggered based on time, location, or user actions, and can include text, images, and sounds. We learnt that Local notification can be implemented through the Capacitor Local Notifications plugin, which provides a simple and intuitive API for scheduling and displaying notifications. We must first install the Capacitor Local Notifications plugin to use local notifications and configure it according to their specific requirements. We also learnt how to schedule local notifications in Capacitor, which can be done using a single API call. We can specify the notification title, body, and any additional parameters such as an icon or sound to be played when the notification is triggered. Notifications can also be scheduled to be triggered at a specific time or based on a user's location. We learnt the techniques on how to display notifications. When a local notification is triggered, it is displayed on the user's device according to the parameters which we can specify. Users can interact with the notification, such as tapping on it to launch the application or dismissing it. Local notifications can be displayed even if the application is not currently running in the foreground, making them a powerful tool for keeping users engaged and informed. The LocalNotifications plugin in Capacitor allows us to manage local notification events. We can also track how users interact with notifications, including whether they dismiss or click them to open an application. This is our analysis in capacitor local notifications.

[15] We learnt about Google Maps in Capacitor which is implemented by the Capacitor Google Maps extension, which provides a simple and intuitive API to integrate Google Maps into applications.We learnt how To use Google Maps in Capacitor by installing the Capacitor Google Maps plugin and configuring it according to our requirements. We also understood how to show a map in Capacitor using Google Maps. It can be done by using the Capacitor Google Maps plugin's 'create' technique to create a brand new map item. The map item can then be brought to the application's DOM using fashionable HTML and CSS. The map may be custom designed in quite a few ways, along with specifying its preliminary function and zoom level, including markers and overlays, and customising the map's appearance.We also learnt few methods like 'setCameraPosition' method to pan and zoom the map to a specific location, and the 'addMarker' to add markers to the map that represent points of interest.

[16] To track users we learnt about Geolocation in Capacitor Which is implemented by the Capacitor Geolocation extension, which provides a simple and intuitive API to integrate geolocation into our application. To use geolocation in Capacitor, we must first install the Capacitor Geolocation plugin and configure it according to their specific requirements as we did with [15] Google apps in capacitor. We learnt about how To determine a user's current location in Capacitor using geolocation. This is done by using the getCurrentPosition method of the Capacitor geolocation plugin. This method returns the current location of the user in latitude and longitude, which can be used to provide location-based information and services to the user. We also can watch the user’s region in real-time using the Capacitor Geolocation plugin's 'watchPosition' approach.We understood that this method is very important for our application and helps us greatly. This approach constantly gives updates as the user moves. We also learned an additional plugin called Capacitor Geolocation plugin which helps in geofencing and permits us to outline digital geographic limitations and cause activities while the user enters or exits those limitations. The plugin gives strategies for each including and eliminating geofences, in addition, to help for placing geofence radius and duration.

**CHAPTER 3**

**SYSTEM REQUIREMENTS**

**3.1 Hardware Requirements**

* 8 GB RAM
* 3 GB internal storage (hard disk space)
* Intel(R) Core(TM) i5-10210U CPU
* 4 Core(s), 8 Logical Processor(s)
* 3.0 USB Port and Mobile-PC connector
* Mobile Devices (for testing)

**3.2 Software Requirements**

* IDE (Visual Studio Code)
  + Used in the coding for both web interface and mobile interface
* Operating System - Windows 10 and the latest versions
  + Most used was windows 10 and windows 11 in some systems.
* Framework - Ionic
  + Used for app development.
* MySQL Workbench
  + Used to store data, update date and delete data with SQL Query
* Google Cloud
  + Used to generate google maps API.
* GitHub (for project collaboration)

**3.3 Front End**

* HTML
  + HTML is the standard markup language for creating Web pages.
* CSS
  + Using Cascading Style Sheets (CSS) allows developers to separate content from presentation.
* ReactJS
  + React JS is a JavaScript library used in web development to build interactive elements on websites.
* Ionic
  + Ionic provides a set of tools for building native iOS and Android applications, and mobile-ready Progressive Webapps.

**3.4 Back End**

* MySQL
  + MySQL is a relational database management system
* NodeJS
  + Node. js (Node) is an open source, cross-platform runtime environment for executing JavaScript code.

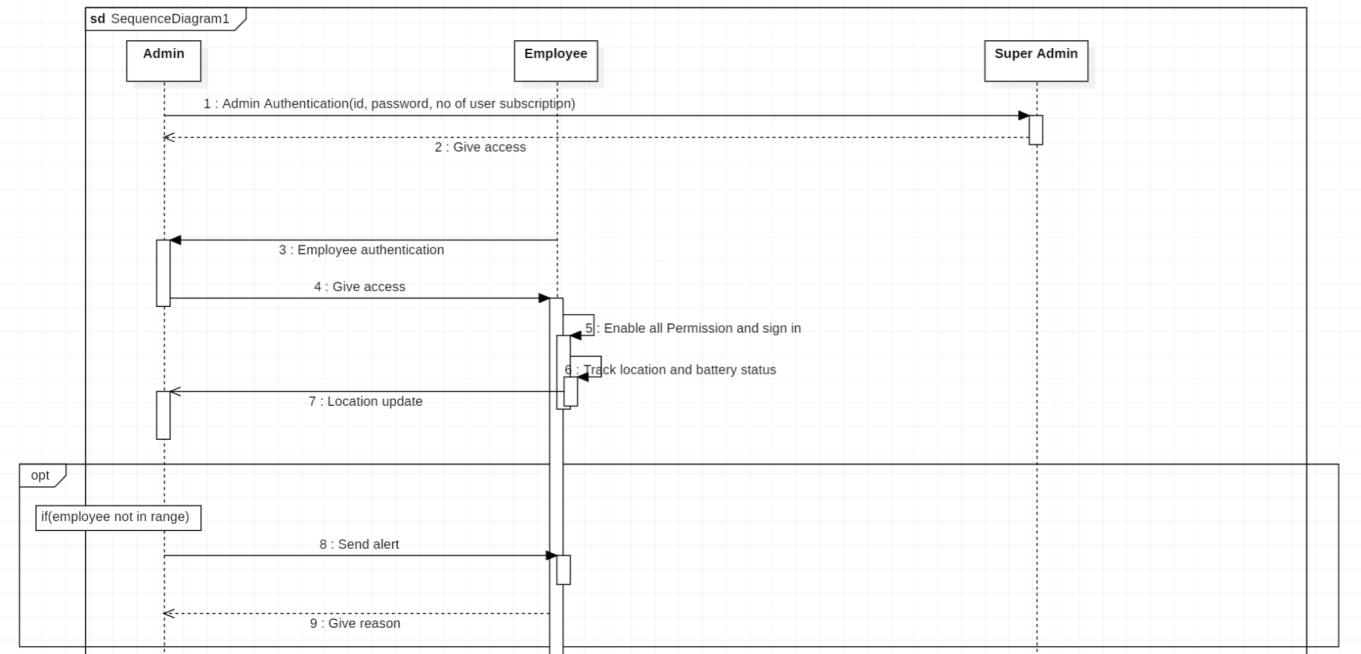
**CHAPTER 4**

**SOFTWARE DESIGN**

**4.1 API**

* Google Maps API - Used to integrate google maps both in our mobiles and web application
* Rest API - Used HTTP requests to access and use data. That data can be used to GET, PUT, POST and DELETE data types, which refers to the reading, updating, creating and deleting of operations concerning resources.

**4.2 Model Workflow**

****

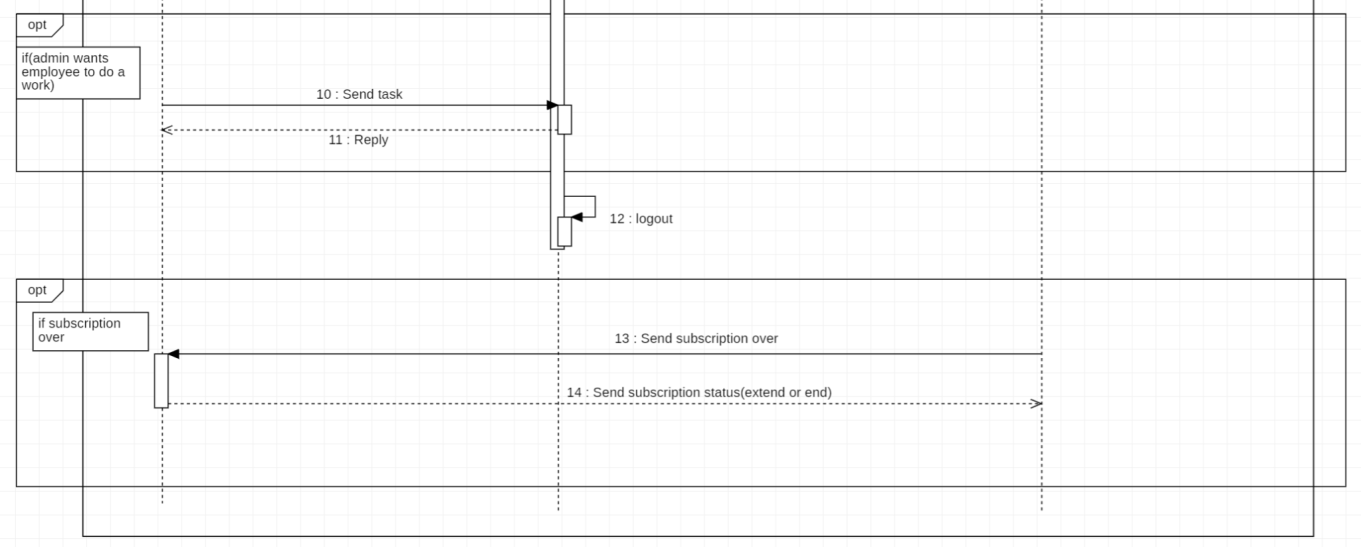


Fig 4.1 Sequence Diagram

At first, When a company subscribes to our application, the administrator of their company will be requesting a username, password, and no of users who can use our application. The super admin of our application will give access to the admin. Employees will be able to log in through their mobile devices when the admin gives access to them with a username and password. When an employee logins through his mobile device, he/she should enable all the permissions required, like location permission. After enabling, the Employee's mobile battery status and location will be shared and it will update the admin. If the employee is not in a given range or he/she went out of the zone(Geolocation), the admin will send an alert to that particular employee and he/she must give a reason for why he/she did that.

There are a few other cases where the admin wants employees to do a task. For that, Admin notifies the employee by sending him a task/s through the website and the employee will receive a notification. Upon getting that, he/she will reply to the admin. When the application’s subscription due is over, the super admin will notify the admin of the company. The admin will reply to the super admin whether he will extend it by subscribing or will terminate his subscription.

**CHAPTER 5**

**SYSTEM IMPLEMENTATION**

**5.1 Web Interface**

The web interface consists of two users and it

* Super Admin
* Admin

**5.1.1 Super Admin**

The super admin is handled by the admins who own this application. They can access the following pages

* Company
* Employee
* View Details

**5.1.1.1 Company**

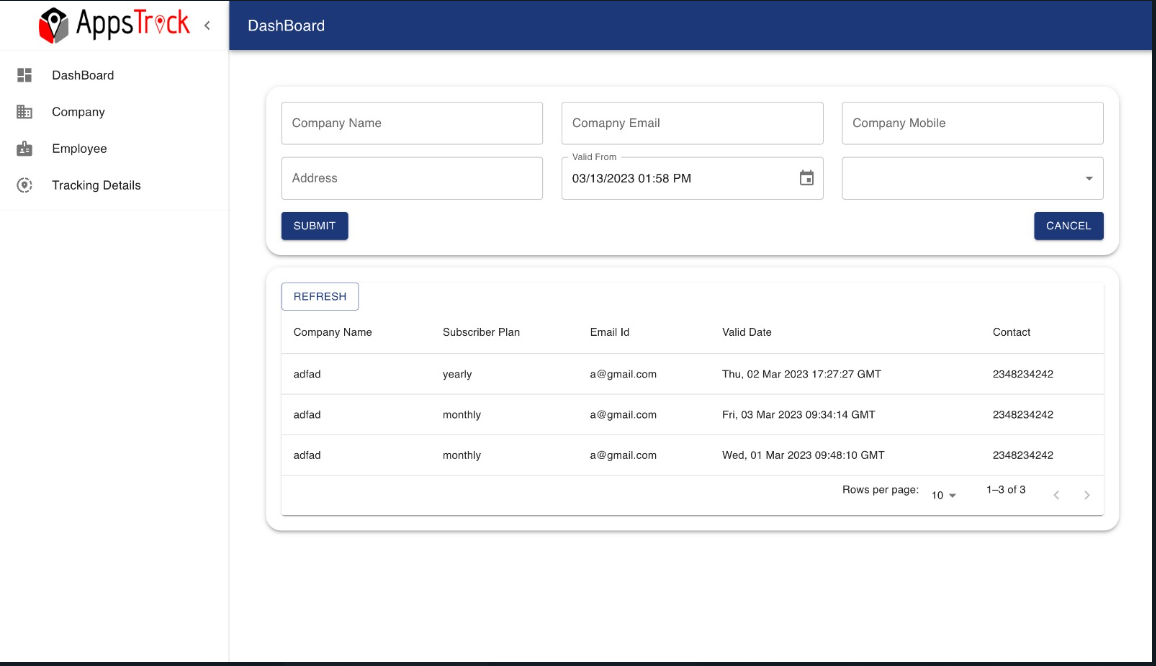
****

Fig 5.1 Adding Company Details

In this module, the super admin enters the company details and their subscription plan along with the number of users who can use the mobile application. They enter the start and end dates, so the company needs to subscribe again once it is over. There are specific plans available. First it can be a yearly subscription or a monthly subscription. Second, they can set the user's limit. Here, we can have 10,30,50,75,100 no of users. The company can buy any one of these subscription plans.

**5.1.1.2 Employee**

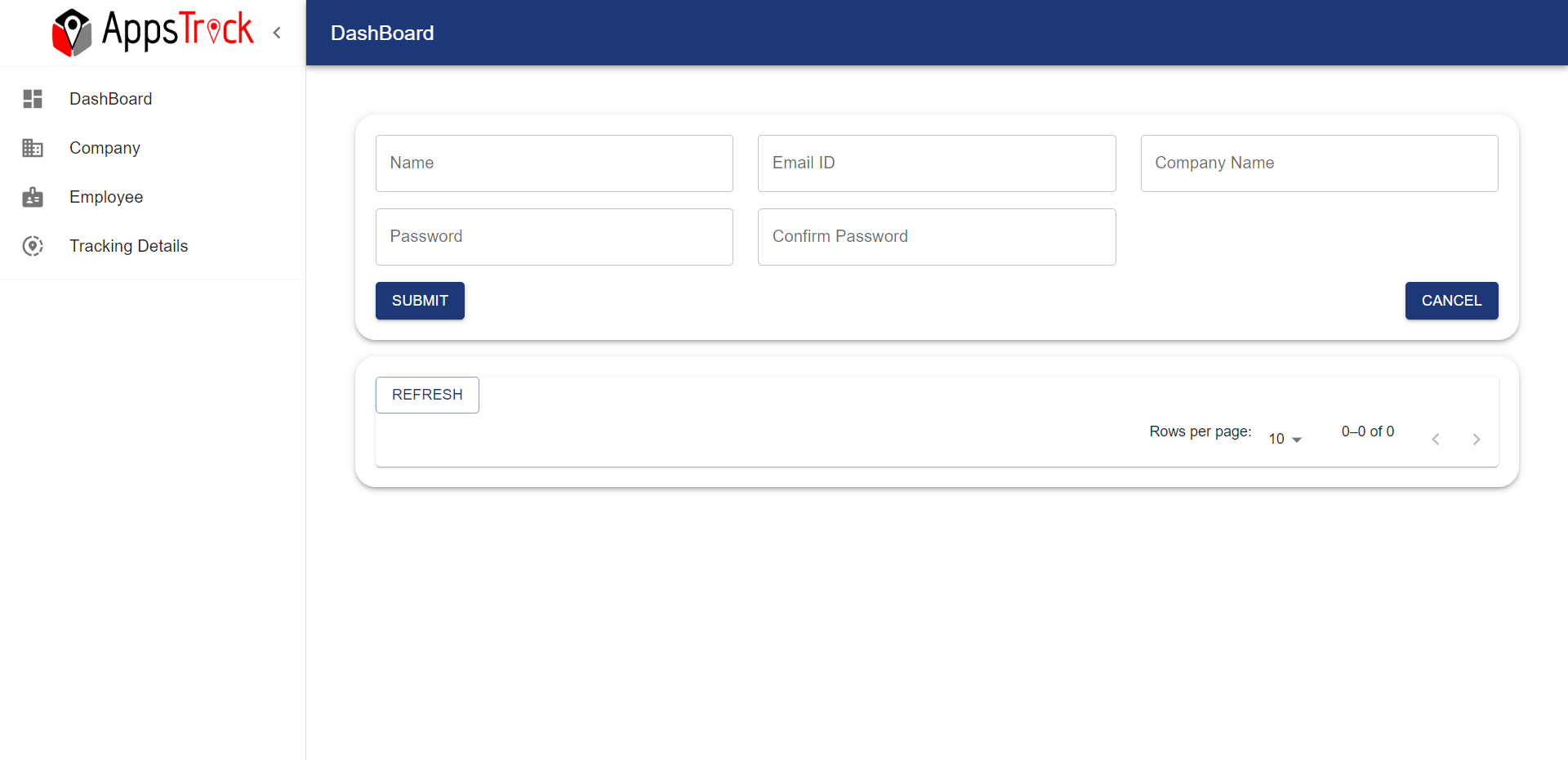


Fig 5.2 Adding Employee Details

In this module, the super admin assigns a unique id and password for the company's admin who subscribed. If the admin forgets their password, they can only contact the super admin to view their password. Hence the super admin has every right to delete the employee if needed.

**5.1.1.3 View Details**

Here, the super admin can view the user details and he has the power to change the password of users only upon the request of the user.

**5.1.2 Admin**

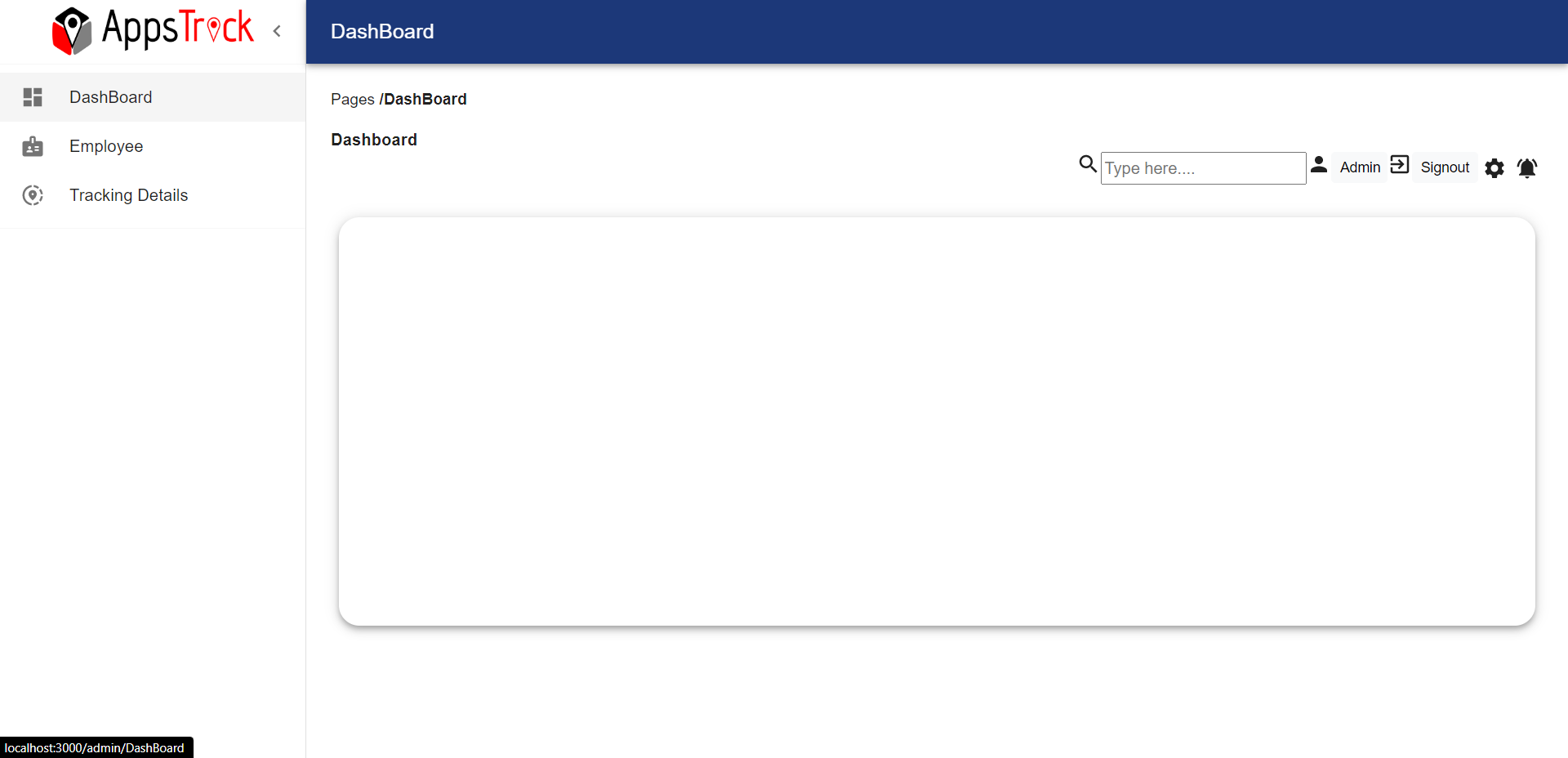
****

Fig 5.3 Dashboard

The admin is the admin of the company/ organisation. They handle their employees and have access to only their company. They can

access the following pages,

* Employee
* Tracking Details
* Message

**5.1.2.1 Employee**

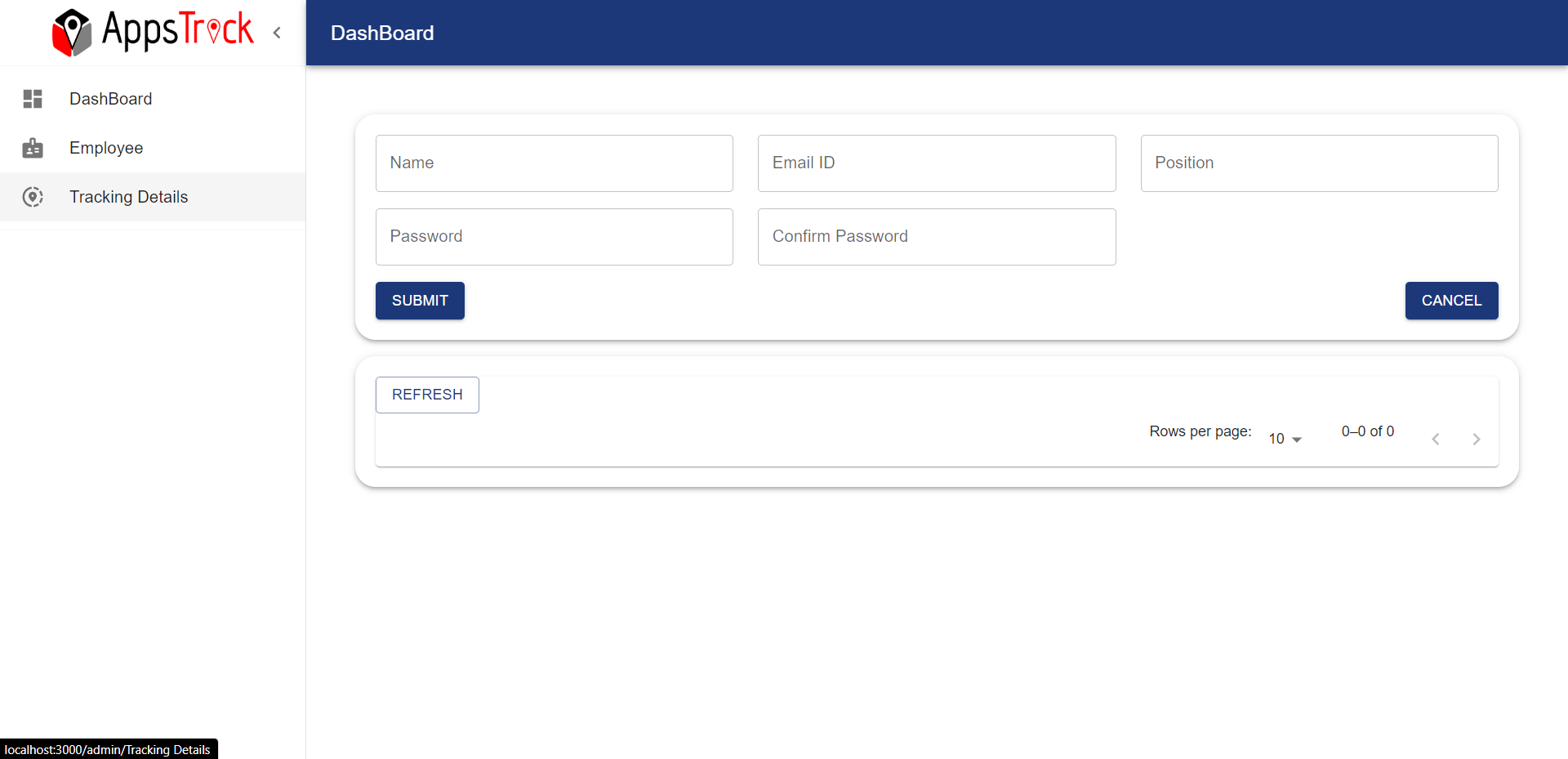


Fig 5.4 Adding Employee Details

This module is used by the admin where they creates employees and to differentiate between them they assigns them a unique employee id and password so that they can use their phones to log in through mobile applications. They are also assigned specific roles.

**5.1.2.2 Tracking Details**

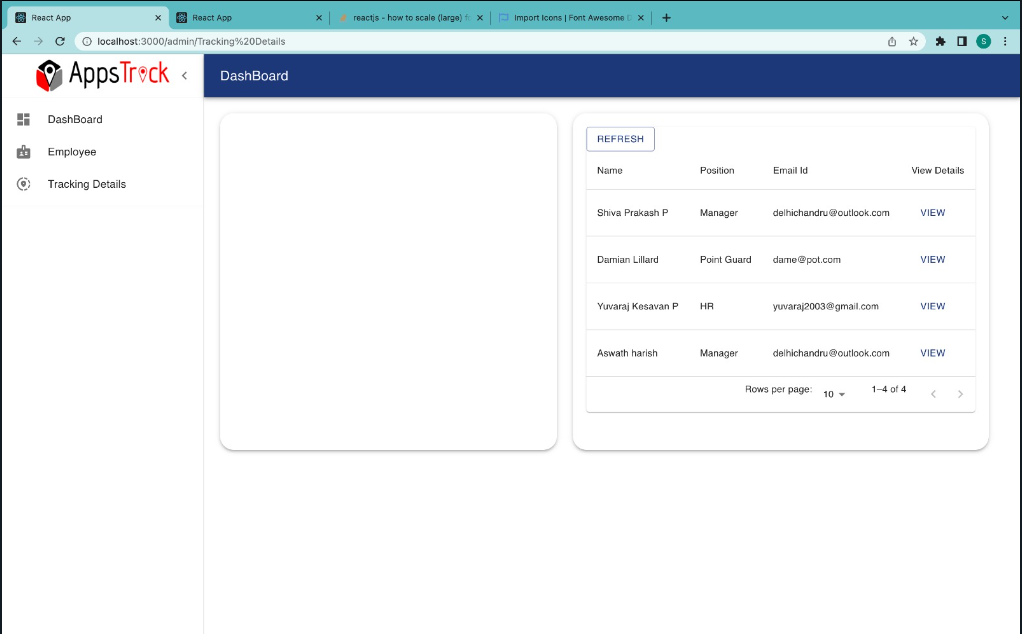
****

Fig 5.5 Tracking Details

This module is the core of the application and the page which explains our idea. Track employees. Here admins can track each and every user and also can view their attendance time, when they checked in and checked out, and in addition to that, can view their battery status.

**5.1.2.3 Message**

This module is used by the admin if they need to convey any message to the employee. They can convey it either as a whole to a group and it is made easy to select by filtering among roles of the employee or they can message any individual. They can also view every chat with employees.

**5.2 App Interface**

The App interface is designed specifically for employees.

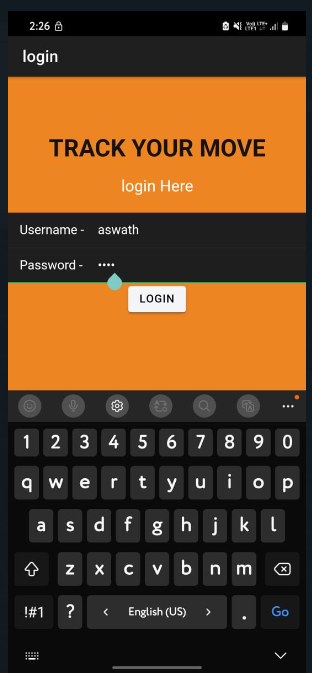
**5.2.1 Login and Logout**

Fig 5.6 Login Page

This module is used by the employees where they can log in and log out using their id and password which will be given by the admin when they register their employees.

**5.2.2 Check-In**

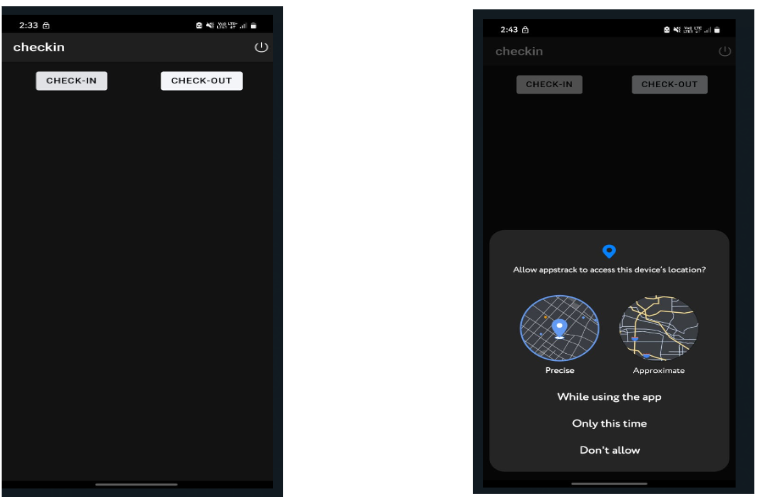
****

Fig 5.7 Check-In and Check-Out

After an employee logs into their account, they can initiate the tracking process by clicking on the Check-In button from the starting point of their travel. This action signals the Appstrack application to begin tracking the employee's movements.

Once the employee has completed their travel, they can click on the Check-Out button to stop the tracking process. This feature ensures that the employee's movements are accurately recorded and helps to prevent any discrepancies in their travel records. By using this system, employees can easily keep track of their travel time and ensure that they are properly compensated for their work.

**5.3 Location Tracking**

The Appstrack application uses GPS and location services to track the employee's movements. The employee's travel history is displayed on a Google map, which provides a clear and accurate representation of their journey.

This feature enables managers to monitor their employees' travel patterns and ensure that they are following the correct routes and procedures. It also provides employees with a clear record of their travel history, which can be useful for reporting and reimbursement purposes. By leveraging the power of GPS and location services, AppTrack provides a reliable and efficient tracking solution for businesses of all sizes.

**5.4 Geo-Fencing**

AppTrack incorporates Google's Geo-fencing technology, which allows businesses to create virtual geographic boundaries using GPS or RFID technology. This feature enables the software to trigger a response when a mobile device enters or leaves a particular area, such as a worksite or restricted zone.

In AppTrack, businesses can set up Geo-fencing boundaries to monitor employee movements and ensure compliance with company policies. If an employee travels outside of the designated area, the application will generate an alert, notifying managers or supervisors of the potential breach. This feature provides an additional layer of security and control, helping businesses to monitor and manage their employees more effectively.

**5.5 Ideal Time Alert**

AppsTrack allows businesses to configure an ideal time setting for their employees, which determines the maximum amount of unproductive time that is considered acceptable. This idle time may be due to factors that are both within and outside of management's control.

If an employee remains idle for longer than the configured time, AppTrack will generate an alert to notify managers or supervisors of the situation. This feature helps to ensure that employees are making productive use of their time and can help businesses identify areas where productivity can be improved.

It should be noted that ideal time generally applies to full-time workers rather than consultants, who typically bill for every hour of their time. By using AppTrack to monitor idle time, businesses can better manage their workforce and optimize their operations for greater efficiency and productivity.

**5.6 Report**

AppTrack provides admins with the ability to generate reports that detail employee tracking data for a specific day or a range of dates. This feature enables admins to quickly and easily access the information they need to make informed decisions about their workforce.

Admins can generate reports that detail employee activity, including check-in and check-out times, locations visited, and distance traveled. By accessing this information, admins can identify areas where employee productivity can be improved and make adjustments to their operations as needed.

Whether generating reports for a single day or a range of dates, AppTrack provides admins with a powerful tool for monitoring and managing their workforce. By leveraging the insights provided by AppTrack's reporting features, admins can make data-driven decisions that help to improve efficiency, productivity, and profitability.

**CHAPTER 6**

**CONCLUSION**

Our web interface has been designed perfectly according to our requirements. Super admins and admins can login through our website APPSTRACK and register companies and employees respectively. Our web interface is connected to the database where all the entries are stored. In our app interface, employees can login with their respective ID and password provided by the admin of their company. Their mobile device can be tracked when employees click the check in button. Before that, they will be asked whether they are interested in this and they need to enable the following permissions.

Our team has been working on completing the google maps part in both the app and web interface where user location must be tracked all the time unless and until he/she clicks the check out button. Our database has been separate for app and web interface and we are working hard to make sure everything is stored in one single database.

Our final goal is to make companies easier in tracking their employees and make work finished so soon. We will be making it with a huge database by expanding our storage where multiple companies buy our product and those entries are stored efficiently and faster retrieval of data. It can be further improved by making the UI even more creative and good-looking for anyone out in this world.

**CHAPTER 7**

**APPENDIX 1**

**7.1 ABBREVIATION:**

1. HTML - Hypertext markup language
2. CSS - Cascading Style Sheets
3. JS -Javascript
4. REST - Representational State Transfer
5. DOM -Document Object Model

**CHAPTER 8**

**BIBLIOGRAPHY**

[1] https://ionicframework.com/docs/api/content

[2] https://ionic.io/ionicons

[3] https://ionicframework.com/docs/api/grid

[4] https://ionicframework.com/docs/api/list

[5] https://ionicframework.com/docs/api/icon

[6] https://ionicframework.com/docs/api/input

[7] https://ionicframework.com/docs/api/modal

[8] https://ionicframework.com/docs/api/nav

[9] https://ionicframework.com/docs/api/refresher

[10] https://ionicframework.com/docs/api/router

[11] https://ionicframework.com/docs/api/tabs

[12] https://capacitorjs.com/docs

[13] https://capacitorjs.com/docs/apis/local-notifications

[14] https://capacitorjs.com/docs/apis/google-maps

[15] https://capacitorjs.com/docs/apis/geolocation