PUNJAB ENGINEERING COLLEGE

(DEEMED TO BE UNIVERSITY)

Department of Computer Science and Engineering

CAPSTONE PROJECT

REPORT

HR MANAGEMENT AND GEO-ATTENDANCE SYSTEM

Under the Guidance of **Dr. Manish Kamboj**

Punjab Engineering College (Deemed to be University)

Submitted by:

Avantika Gargya (16103060)

Arunav Sharma (16103024)

Deepak Tiwari (16103017)

Divyansh Singh (16103073)

Madhav Pruthi (16103041)

DECLARATION

We hereby declare that the project work entitled "HR Management and Geo-Attendance System" is an authentic record of our own work carried out at Punjab Engineering College (Deemed to be University), Chandigarh as per the requirements of “Capstone Project” for the award of the degree of B.Tech. Computer Science and Engineering, under the guidance and supervision of Dr Manish Kamboj.

We further declare that the information has been collected from genuine & authentic sources and we have not submitted this project report to this or any other university for the award of diploma or degree of certificate examination.

Certified that the above statement made by the students is correct to the best of my knowledge and belief.

Dr. Manish Kamboj (Assistant Professor, Punjab Engineering College)

Date:

Place:

Arunav Sharma (16103040)

Avantika Gargya (16103060)

Deepak Tiwari (16103017)

Divyansh Singh (16103073)

Madhav Pruthi (16103041)

PUNJAB ENGINEERING COLLEGE

(DEEMED TO BE UNIVERSITY)

Department of Computer Science and Engineering

CERTIFICATE

It is Certified that the Project work entitled **HR Management and Geo-Attendance System** submitted by **Arunav Sharma, Avantika Gargya, Divyansh Singh, Deepak Tiwari and Madhav Pruthi** for the fulfilment of Capstone Project offered by Punjab Engineering College (Deemed to be University) during the academic year **2019-20** is an original work carried out by the students under my supervision and this work has not framed any basis for the award of and Degree, Diploma or such other titles. All the work related to the project is done by these candidates themselves. The approach towards the subject has been sincere and scientific.

Date:

Dr Manish Kamboj
CSE Department
Punjab Engineering College (Deemed to be University)

ACKNOWLEDGEMENT

We would like to take this opportunity to thank our college **Punjab Engineering College (Deemed to be University), Chandigarh** and **Department of Computer Science and Engineering** for giving us an opportunity to work on this project.

We are immensely grateful to our project mentor **Dr Manish Kamboj (Professor, Department of Computer Science and Engineering)** whose continuous guidance, technical support and moral support at times of difficulty helped us to achieve milestones in the given time. He has been a great source of knowledge and without him, this project could not have been completed.

We are also thankful to the Committee for evaluation who gave their valuable feedback and guidelines for the betterment and completion of this project.

We convey our deep sense of gratitude to all teaching and non-teaching staff for their constant encouragement, support and selfless help throughout the project work. It is a great pleasure to acknowledge the help and suggestion, which we received from the Department of Computer Science and Engineering.

We wish to express our profound thanks to all those who helped us in gathering information about the project. Our families too have provided moral support and encouragement at several times.

Arunav Sharma (16103040)

Avantika Gargya (16103060)

Deepak Tiwari (16103017)

Divyansh Singh (16103073)

Madhav Pruthi (16103041)

ABSTRACT

Our aim with this project is to develop an application that can assist both the employer and the employees that are a part of an organisation, in making the process of tracking attendance, leaves, live location and record of all on-field worker's faster and more efficient. With the use of Flutter, which is a cross-platform software development kit created by Google to develop applications for both Android and iOS, we were able to achieve the same.

The study focuses primarily on explaining our process of employing Google's Firebase and Flutter to accomplish features like Geo-Fencing, Attendance-Management, Leave status record as well as other HR Management activities. The project retrieves the data from Firebase, analysis it, pins the markers to Google Map and lets the employer know the location of the employee at any given point of time if required.

By making use of an HR Management application as proposed by us, any organisation would greatly benefit by reducing the error rate in the maintenance of attendance logs, decrease workload designated for the management of all the employees, as well as increase the efficiency of time and error rate for an organisation.

Installing an attendance tracking app on your phone can help keep a hand on your work. It is especially useful for employees who work on a site with no access to timesheets, for those who leave the office for meetings with clients, or people who travel on a regular basis.

Attendance marking has moved from manual punching to location-tracking based attendance apps within a short span of time.

Security Issues like Multiple Sign-In are smartly handled using Unique Identifier keychain storage for each user.

Table of Contents

DECLARATION 2

ACKNOWLEDGEMENT 4

ABSTRACT 5

LIST OF FIGURES 8

CHAPTER 1: INTRODUCTION 9

CHAPTER 2: BACKGROUND 10

CHAPTER 3: PROPOSED WORK 11

- 3.1 Proposed System Features 11
- 3.2 Operating Environment 11
- 3.3 User Registration 11
- 3.4 Security Requirements 12
- 3.5 Human Resources Head's Access 12
- 3.6 Employee Manager's Access 12
- 3.7 Geo-Attendance/ Location Based Attendance 13
- 3.8 Employee Leave Application and Management 13
- 3.9 Technique Proposed 13

CHAPTER 4: IMPLEMENTATION 14

- Technologies Used 14
- 4.1 Flutter 14
- 4.2 Isolates in Flutter 15
- 4.3 Event Loop 16
- 4.4 Futures 18
- 4.5 Compute 18
- 4.6 Ports 18
- 4.7 IsolateNameServer 19
- 4.8 Firebase 19
- 4.8.1 Features 19
- 4.9 Model-View-Controller (MVC) 20
- 4.9.1 Advantages of MVC 21
- 4.10 Firebase Cloud Functions 22
- 4.11 Firebase Cloud Messaging (FCM) 23
- 4.12 Basic Geo-fencing Implementation 24
- 4.13 Attendance Marking Component 26
- 4.14 Attendance Summary Component (Calendar) 27

4.15 Leave Management System Implementation 28
4.16 Leave Status System Workflow 28
4.17 HR App – Add User Workflow 29
4.18 HR App – Edit User Workflow 29
CHAPTER 5: RESULTS AND DISCUSSIONS 30
5.1 Features Offered by the Project 30
5.2 Location-based Attendance 30
5.3 Leave Management System 30
5.4 Attendance Summary 31
5.5 Leave Summary 31
5.6 HR App 31
5.7 Adding a new Employee 31
5.8 Edit existing Employee 31
5.9 Firebase 32
CHAPTER 6: CONCLUSIONS AND FUTURE WORK 35
REFERENCES 37

LIST OF FIGURES

Figure 1 Event Loop in action 17
Figure 2 Interaction of MVC 21
Figure 3 FCM Operation 23
Figure 4 Geofence Service Initializer 25
Figure 5 Attendance mark Flow-chart 27
Figure 6 Leave Management Flow chart 29
Figure 7 Data Traffic on Database 32
Figure 8 Application Downloads 32
Figure 9 User Activity Graph 33
Figure 10 Time Spent by users 34

CHAPTER 1: INTRODUCTION

India being a developing nation as well as an emerging economy, makes for an ideal breeding ground for many start-ups and small businesses or enterprises.

With this also arises a large pool of problems that require immediate solutions to enable the Indian start-ups to prosper. One of the most widespread issues faced by small start-ups of nearly every category is the lack of a strong technical support system that allows them to function smoothly. Many of these small companies do not have an efficient technical team that could automatize most of their work, manage the employees as well as execute data management.

Therefore, a flexible and modular system that functions as per the requirements of each company is a necessity and would provide easy access with a one-stop management set-up.

It is apparent that different industries have varying technical and management requirements and demands, thus it was our decision to initiate the development of a software eco-system using flutter and firebase by Google. In one such endeavour we were hired by the company **Indus Automotive Pvt. Ltd.** to develop an application for location-based attendance and management of the employees. With back and forth communication from the company, we built a prototype that is secure from multiple and illegit logins.

CHAPTER 2: BACKGROUND

Indus Automotive Pvt Ltd. used rudimentary methods for the marking of attendance and leave management system. So, to overcome the drawbacks provided these ancient methods and to automate their systems a mobile application was proposed as a solution.

The technologies used were decided upon because flutter and firebase are up and coming technologies in the market. So, while developing the project the team wanted to learn new technologies. Firebase, apart from being a real-time database technology also provides various analytical tools which may be beneficial for the company in the future. Flutter, on the other hand is a programming language that allows us to code for both Android and iOS platforms with minor changes only.

As the company has many offices across the country therefore, they wanted location-based attendance system. Also, the app should include basic HR functions like attendance summary, leave application and leave status that makes the maintenance of HR records easier and more efficient. It was specified to the team beforehand that the solution expected should include location-based attendance. There was to and fro between us and their team to finalise documentation on System Requirements and the Data flow in the system. This technique was selected as it was the most suitable to the company instead of biometrics which wasn't feasible to be installed at all the offices, especially at the remote locations.

CHAPTER 3: PROPOSED WORK

The proposed solution for this problem includes a solution centred on Geo-fencing an area around the office, entering into which would enable the employee to mark his attendance. The features proposed by us to the company were:

3.1 Proposed System Features

1. **Employee Registration –** This feature will allow an employee to register a unique identity with the system. This also encompasses role-based access based on regular employee or administrator status.
2. **Administrator Access-** This feature will allow administrators to modify, add, or remove information related to employees and the system.
3. **Location-based Attendance-** This feature will allow employees to mark their attendance once the employee has reached his allocated office.
4. **Leave Management-** This feature will allow employees to apply for leave and track its status.
5. **Leave Approval/Rejection -** With each employee is an associated Manager, who can approve or reject the leaves applied for, by an employee under him.

3.2 Operating Environment

This system will be operated by employees and administrator through an Android/iOS mobile application which is built using Flutter (Google Framework) and Firebase Realtime Database.

3.3 User Registration

This Feature describes how the employees/administrators will register and be managed. An employee/administrator wants to register with the system and should be able to login to use it.

- A user must be able to register with the system
- A user must be classified as a regular employee or administrator i.e. manager
- Once registered, a user must be able to log into the system
- Once logged in, the user must be able to mark his attendance when he reaches the allocated location.
- Administrator must get the notification once the attendance is marked by the employee.

3.4 Security Requirements

The security aspect of an attendance application is important in this case as the company has multiple offices that are far apart and they wish to ensure that their employees have reached their respective office for the day, irrespective of where the manager is.

- A user's password must be encrypted using sha-1 hashing.
- Transactions which will deal with the password of the user will be handled only using the encrypted version of the password
- When resetting a user's password, a confirmation email will be sent to their registered email account. The password will not be reset unless the confirmation email is responded to.
- Multiple log-ins must be prevented from the same user
- A single user must not be able to sign-in using multiple devices.
- Varying level of access to the application/applications and related privileges

3.5 Human Resources Head's Access

- He would be able to perform some action on Employees database.
- An administrator must be able to view attendance history of employee
- An administrator must be able to approve/reject request to changes in employee details
- An administrator must be able to remove a user from the system
- An administrator must be able to grant administrator status
- An administrator must be able to revoke administrator status
- An administrator must be able to view a transaction log which will contain information for events happening in the system

3.6 Employee Manager's Access

- A Manager must be able to grant/reject leaves
- An employee manager is able to view the attendance record of his employees
- Manager must get the notification once the attendance is marked by the employee.

3.7 Geo-Attendance/ Location Based Attendance

An employee will be able to mark his attendance on reaching on-site duty location

- Employee must be able to request for marking attendance at any point of time.
- Attendance request must be accepted if the employee is within a Geofencing area of the office allotted to him that day

3.8 Employee Leave Application and Management

There would be three types of leaves available to the employee – Annual leaves, Medical leaves and Casual Leaves.

- Number of leaves of each type must be fixed and monitored
- Employee must be able to see his leave count and previous leave details
- Employee must be able to see his attendance record and summary
- Employee can submit applications for casual/annual/medical leaves to take a day off

3.9 Technique Proposed

A Geo-fence is a virtual fence or boundary that's placed around a real-world location. It can vary in size, from encircling a small store to covering an entire city. It can also have varying shapes, from circles to polygons. However, most geo-fences are created for a specific area or point of interest.

A geo-fence can be put into use once the geolocation data of an entity is available. The geolocation data could be made available on any GPS-enabled device, like a smartphone or a car with GPS-enabled hardware. The data can then be used by a product or app to understand the location of the phone or car and provide services based on this information. Once the geo-fence is created, the product or app can set up triggers, which then send a text message, email alert, or app notification when an entity under observation enters (or exits) the geo-fence. Creating this geo-fence and setting up the triggers is called Geo-fencing.

CHAPTER 4: IMPLEMENTATION

Technologies Used

4.1 Flutter

Flutter is Google's mobile UI framework for crafting high-quality native interfaces on iOS and Android in record time. Flutter works with existing code, is used by developers and organizations around the world, and is free and open source.

It is a new way to develop apps for Android, iOS and many other platforms

Flutter builds native apps and they are written in the Dart language. Dart is easy to learn and allows a lot of versatility to the Flutter platform. It allowed us to create apps for Android and iOS with the same codebase, thus reducing the work required for the development of separate applications for separate platforms. Flutter builds native apps and they are written in the Dart language. Dart is easy to learn and allows a lot of versatility to the Flutter platform. It allowed us to create apps for Android and iOS with the same codebase, thus reducing the work required for the development of separate applications for separate platforms.ds native apps and they are written in the Dart language. Dart is easy to learn and allows a lot of versatility to the Flutter platform. It allowed us to create apps for Android and iOS with the same codebase, thus reducing the work required for the development of separate applications for separate platforms.

Flutter has been in Beta since February 2018 and since then the Google team and the community has been adding new features and tweaks, fixing bugs and creating great resources.

Dart is a client optimized programming language for applications on multiple platforms. It is developed by Google and is used to build mobile, desktop, and backend as well as web applications. It is object-oriented, class-defined, garbage-collected language using C- style syntax that compiles optionally into JavaScript. It supports interfaces, mixins, abstract classes, reified generics and much more.

- **Compiled as JavaScript**

  To run in mainstream web browsers, Dart relies on a source-to-source compiler to JavaScript. According to the project site, Dart was "designed to be easy to write development tools for, well-suited to modern app development, and capable of high-performance

implementations."[18] When running Dart code in a web browser the code is precompiled into JavaScript using the dart2js compiler. Compiled as JavaScript, Dart code is compatible with all major browsers with no need for browsers to adopt Dart. Through optimizing the compiled JavaScript output to avoid expensive checks and operations, code written in Dart can, in some cases, run faster than equivalent code hand-written using JavaScript idioms.

- **Stand-Alone**

  The Dart software development kit (SDK) ships with a stand-alone Dart VM, allowing Dart code to run in a command-line interface environment. As the language tools included in the Dart SDK are written mostly in Dart, the stand-alone Dart VM is a critical part of the SDK. These tools include the dart2js compiler and a package manager called pub. Dart ships with a complete standard library allowing users to write fully working system apps, such as custom web servers.

- **Ahead-Of-Time Compiled**

  Dart code can be AOT-compiled into machine code (native instruction sets). Apps built with Flutter, a mobile app SDK built with Dart, are deployed to app stores as AOT-compiled Dart code.

  Flutter is a beautiful way to create great applications that require less code-work to create high performance applications that are compatible with a lot of devices.

4.2 Isolates in Flutter

Developers moving to Flutter are most likely new to Dart, and therefore are not aware of architecture and development approaches until faced with laggy UI or Flutter screens of death (error message instead of expected UI elements).

It all started while I was using the FutureBuilder widget. The issue seemed simple: How do I show a dialog in the event the REST API fails because of network issues (i.e. no network).

Dart, despite being a single-threaded language, offers support for futures, streams, background work, and all the other things you need to write in a modern, asynchronous, and (in the case of Flutter) reactive way. This article covers the foundations of Dart's support for background work: *isolates* and *event loops*.

Isolates enable executing code in a different process. Input and output parameters/data is passed using message passing. This reminds me of the notion of processes in Unix that use pipes as parameter/data passing mechanisms. Also, Isolates do not share memory as threads do. I guess this is why they are also called Isolates? They are isolated pieces of execution that rely on input and deliver output.

Why should I use isolates in Flutter? Anywhere we need to compute an expensive operation and do not want to block the UI, an isolate should be used.

There are 2 ways to deal with isolates:

- High level using a convenient Compute function
- Using low level APIs: SendPort and ReceivePort for message passing

An isolate is what all Dart code runs in. It's like a little space on the machine with its own, private chunk of memory and a single thread running an event loop. An isolate has its own memory and a single thread of execution that runs an event loop. In Dart, though, each thread is in its own isolate with its own memory, and the thread just processes events (more on that in a minute).

Many Dart apps run all their code in a single isolate, but you can have more than one if you need it. If you have a computation to perform that's so enormous it could cause you to drop frames if it were run in the main isolate, then you can use Isolate.spawn() or Flutter's compute() function. Both of those functions create a separate isolate to do the number crunching, leaving your main isolate free to say rebuild and render the widget tree.

Two isolates, each with its own memory and thread of execution. The new isolate gets its own event loop and its own memory, which the original isolate — even though it's the parent of this new one — isn't allowed to access. That's the source of the name isolate: these little spaces are kept isolated from one another. In fact, the only way that isolates can work together is by passing messages back and forth. One isolate sends a message to the other, and the receiving isolate processes that message using its event loop. Isolates can send messages to other isolates. This lack of shared memory might sound kind of strict, especially if you're coming from a language like Java or C++, but it has some key benefits for Dart coders.

4.3 Event Loop

An event loop's job is to take an item from the event queue and handle it, repeating these two steps for as long as the queue has items.

Once a Dart function starts executing, it continues executing until it exits. In other words, Dart functions can't be interrupted by other Dart code.

Dart has the concept of isolates. An isolate is similar to a process in a Unix lingo, meaning that as opposed to threads, each isolate do not share memory and the only way to communicate with them is via message passing.

In Flutter, when the main has finished executing the event loop is running in the single isolate Dart has created.

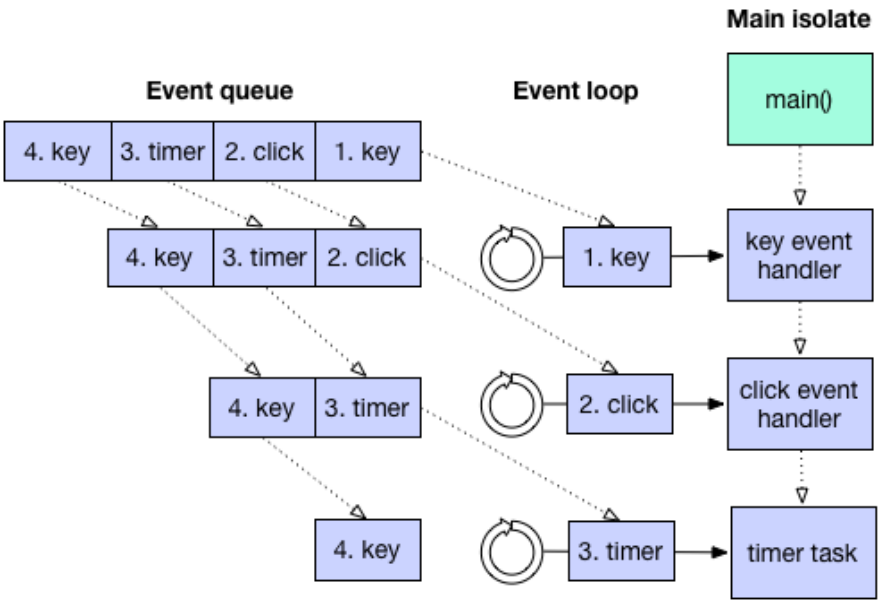

Figure 1 Event Loop in action

So, to summarize the various execution types:

- Method: direct execution
- Future, async and await: placed in the event loop and executed sequentially
- Isolate, executed in a different process

Now that you've had a basic introduction to isolates, let's dig in to what really makes asynchronous code possible: the event loop. Imagine the life of an app stretched out on a timeline. The app starts, the app stops, and in between a bunch of events occur — I/O from the disk, finger taps from the user. The app can't predict when these events will happen or in what order, and it has to handle all of them with a single thread that never blocks. So, the app runs an event loop. It grabs the oldest event from its event queue, processes it, goes back for the next one, processes it, and so on, until the event queue is empty. The whole time the app is running — you're tapping on the screen, things are downloading, a timer goes off — that event loop is just going around and around, processing those events one at a time.
The event loop processes one event at a time.

When there's a break in the action, the thread just hangs out, waiting for the next event. It can trigger the garbage collector. All of the high-level APIs and language features that Dart has for asynchronous programming — futures, streams, async and await — they're all built on and around this simple loop.

4.4 Futures

A Future is a mechanism that allows you to queue execution blocks into the event loop. In the most recent Dart, the async/await mechanisms provide the same functionalities. In order to chain Futures and make sure we can use a Future's result and provide it into another one, we should use the then() mechanism.

In Dart, the code executed by then() is executed directly (method — direct execution), as it is not a Future. The previous code sample explicitly creates a Future in the then() and therefore enqueues the request and gives the event loop a chance to process other events from the event queue. The link provided at the top of the article on the Dart event loop provides some test questions on the order of execution, with code samples. Going through them will help cement your understanding of how it all works.

4.5 Compute

The compute function hides the message passing mechanisms from the developers. The Compute function requires 2 parameters, the function to execute, and the parameter to pass into that function. The function must be static or a global function.

4.6 Ports

This is the low-level way to write isolates, using the *SendPort* and *ReceivePort*. A **ReceivePort** is a non-broadcast stream. This means that it buffers incoming messages until a listener is registered. Only one listener can receive the messages. The messages are sent and received through *SendPort* and *ReceivePort* instances.

To send a message to an isolate port, we need to obtain *ReceivePort* instance corresponding to it. The *ReceivePort* class exposes a *SendPort* getter that is bounded to isolate, so that we can send messages through it.

4.7 IsolateNameServer

The *IsolateNameServer* class is a global register of Dart isolates, from which can register and look up for *SendPorts* and ReceivePorts. An Isolate can register its *ReceivePort* through *IsolateNameSever.registerWithName* method and other isolates can obtain the corresponding SendPort with the *IsolateNameServer.lookupPortByName*() method.

4.8 Firebase

Firebase is a Backend-as-a-Service — BaaS — that started as a YC11 start-up and grew up into a next-generation app-development platform on Google Cloud Platform.

4.8.1 Features

- **Real-time Database**

  The main feature of the application is the Real-time Database. Real-time data is the way of the future. Most databases require you to make HTTP calls to get and sync your data. Most databases give you data only when you ask for it. When you connect your app to Firebase, you're not connecting through normal HTTP. You're connecting through a WebSocket. WebSockets are much, much faster than HTTP. You don't have to make individual WebSocket calls, because one socket connection is plenty. All of your data syncs automagically through that single WebSocket as fast as your client's network can carry it.

  Firebase sends you new data as soon as it's updated. When your client saves a change to the data, all connected clients receive the updated data almost instantly.

- **File Storage**

  Another noticeable feature of Firebase is File Storage. Firebase Storage provides a simple way to save binary files — most often images, but it could be anything — to Google Cloud Storage directly from the client.

  Firebase Storage has its own system of security rules to protect your GCloud bucket from the masses while granting detailed write privileges to your authenticated clients.

- **Authentication**

  Thirdly, Authentication is a key feature as well. Firebase auth has a built-in email/password authentication system. It also supports OAuth2 for Google, Facebook, Twitter and GitHub. We'll focus on email/password authentication for the most part. Firebase's OAuth2 system is well-documented and mostly copy/paste.

  Firebase Auth integrates directly into Firebase Database, so you can use it to control access to your data.

- **Hosting**

  Hosting is another feature of Firebase. Firebase includes an easy-to-use hosting service for all of your static files. It serves them from a global CDN with HTTP/2.

  And to make your development particularly painless, Firebase hosting utilizes Superstatic, which you can run locally for all of your testings. The following implementation uses Gulp, but Gulp is purely optional.

  The BrowserSync + Superstatic development environment is slick. BrowserSync handles reloading your development app across all connected devices and Superstatic replicates Firebase hosting locally in such a way that you can deploy straight to Firebase for production use. Firebase is a Fully-Featured App Platform. The Firebase team has integrated new and existing Google products with Firebase.

4.9 Model-View-Controller (MVC)

MVC is a design pattern used to decouple user-interface (view), data (model), and application logic (controller). This pattern helps to achieve separation of concerns.

The MVC is composed of three components – The Model, the View and the Controller. Correspondingly, we have UI as Views, Models, flutter classes and Services as Controller.

The controller receives the request and generates a model for the same. The model has the classes that describe the data and contains the logic and method to retrieve the data from the database. The View contains the logic to generate the rendered files to present the required data. The Controller chooses

the View to display to the user, and provides it with any Model data it requires. The View renders the final page, based on the data in the Model.

This delineation of responsibilities helps you scale the application in terms of complexity because it's easier to code, debug, and test something (model, view, or controller) that has a single job. It's more difficult to update, test, and debug code that has dependencies spread across two or more of these three areas.

For example, user interface logic tends to change more frequently than business logic. If presentation code and business logic are combined in a single object, an object containing business logic must be modified every time the user interface is changed. This often introduces errors and requires the retesting of business logic after every minimal user interface change.

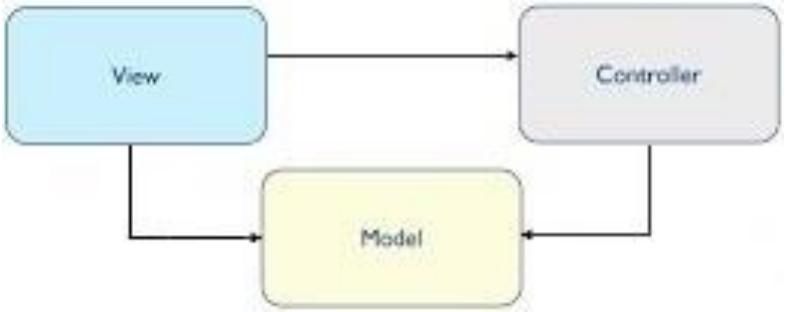

Figure 2 Interaction of MVC

4.9.1 Advantages of MVC

Here, are major benefits of using MVC architecture.

- Easy code, easy maintenance and easy to extend and grow
- MVC Model component can be tested separately from the user
- Easier support for a new type of clients
- Development of the various components can be performed parallelly.

- It helps you to avoid complexity by dividing an application into the three units. Model, view, and controller
- It only uses a Front Controller pattern which process web application requests through a single controller.
- Offers the best support for test-driven development
- It works well for Web apps which are supported by large teams of web designers and developers.
- Search Engine Optimization (SEO) Friendly.
- All classes and objects are independent of each other so that you can test them separately.
- MVC allows logical grouping of related actions on a controller together.

4.10 Firebase Cloud Functions

Web and mobile application often require back-end code to execute tasks like: sending out notifications or processing long running tasks (e.g. scaling images etc). In the traditional approach this back-end code is running on a server. Recently Google's Firebase introduces a new functionality which is called Cloud Functions. With this new service Firebase offers a scalable solution for running back-end code in the cloud. Running code in the cloud has various advantages:

- You do not need to run and maintain your own server
- You do have an isolated code base for back-end code
- You only get billed for the actual executing time of you code
- The cloud infrastructure is highly scalable

Triggers

The functions you write can respond to events generated by Firebase and Google Cloud features. We will call these features triggers.

4.11 Firebase Cloud Messaging (FCM)

Firebase Cloud Messaging (FCM) is a cross-platform messaging solution that lets you reliably deliver messages at no cost.

Using FCM, you can notify a client app that new email or other data is available to sync. You can send notification messages to drive user re-engagement and retention. For use cases such as instant messaging, a message can transfer a payload of up to 4KB to a client app.

An FCM implementation includes two main components for sending and receiving:

- A trusted environment such as Cloud Functions for Firebase or an app server on which to build, target, and send messages.
- An iOS, Android, or web (JavaScript) client app that receives messages.

We can send messages via the Firebase Admin SDK or the FCM server protocols. For testing or for sending marketing or engagement messages with powerful built-in targeting and analytics, you can also use the Notifications composer.

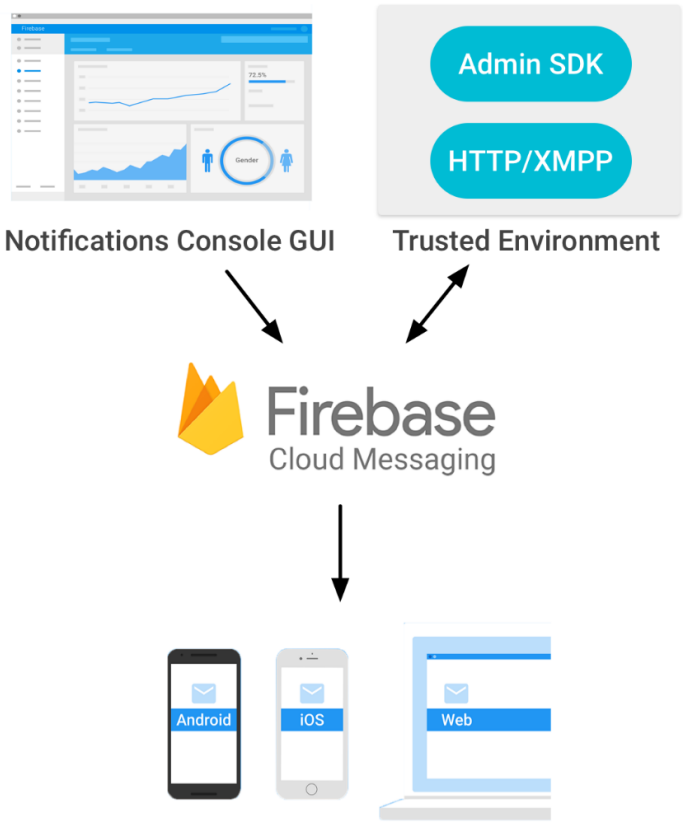

Figure 3 FCM Operation

4.12 Basic Geo-fencing Implementation

The crux of good geo-fencing is a smart alert/notification generation depending on the location of an entity and whether they are entering or exiting the geo-fence. Here is an example of how to support this entering and exiting:

- Store the geofence data as multiple coordinates for a polygon geofence or as a radius and centre point coordinate for a circular geofence.
- Check if the current location coordinate (latitude, longitude) of an entity is entering or exiting a geofence.
- To find out if an entity is entering or exiting, check if the last location coordinate of the entity was inside or outside of the geofence.
- If the last location coordinate of an entity was outside of the geofence and the current location coordinate of the entity is inside the geofence, then the entity is entering the geofence.
- If last location coordinate of an entity is inside the geofence and current location coordinate of the entity is outside the geofence then the entity is exiting the geofence.

Integration with our system

We have used the power of isolates to create Geofencing Service in Flutter. As mentioned above isolates, they have their separate computation power which makes our system to work smoothly. Communication through Ports is done to create callback methods which update our user location state. Basic Work-flow is described below:

- First of all, geofence is initialised by spawning of isolate as background service.
- Coordinates required for creating geofence will be fetched from user database.
- Geofence will be created around the office location
- Then, we initialise ReceivePort which is used to communicate with geofence isolate.
- ReceivePort will keep listening to any changes in user geofence state and update it on the map.
- Callback function is called at the time of geofence state change and update the local variables accordingly.

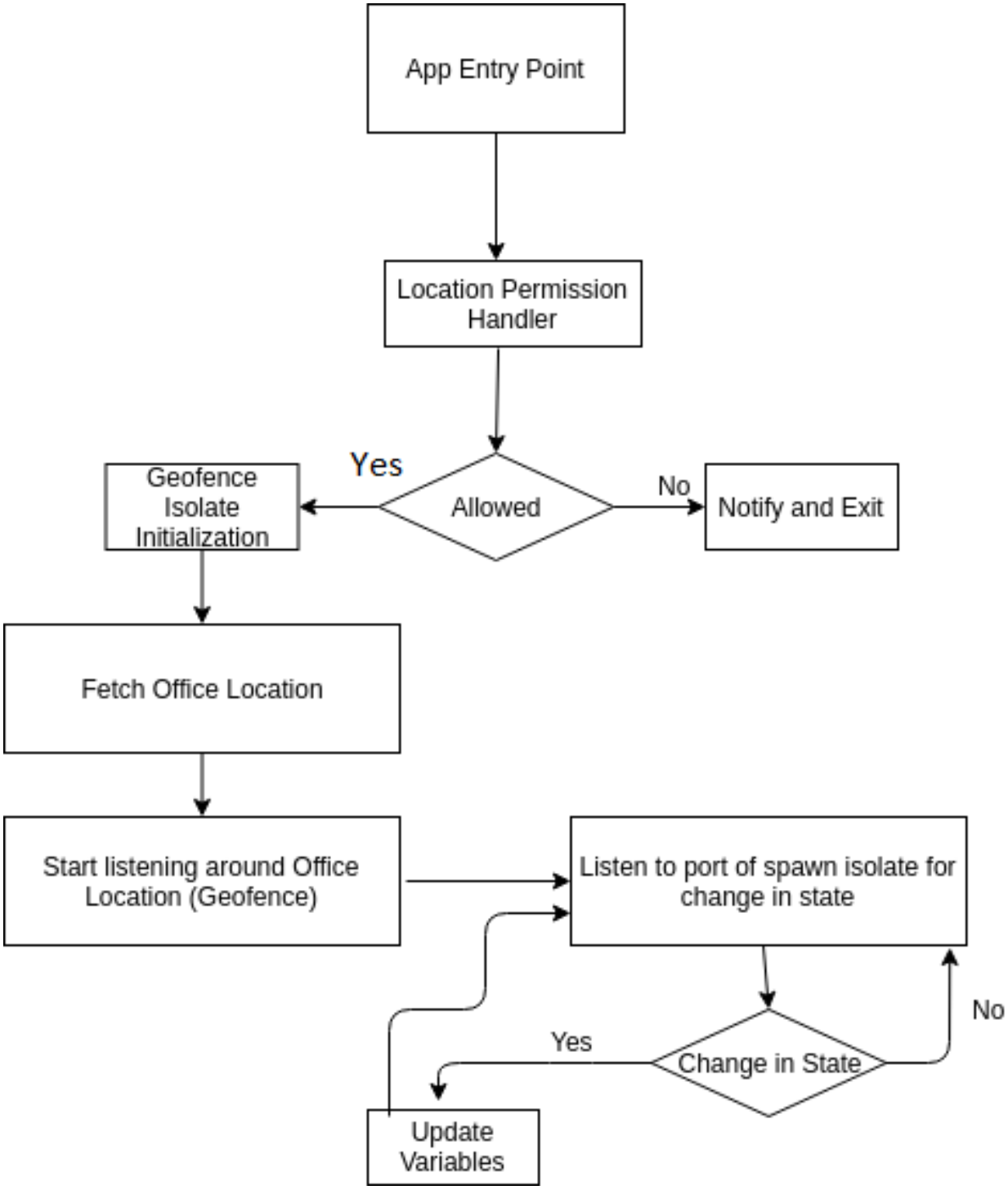

Figure 4 Geofence Service Initializer

Geofencing combines awareness of the user's current location with awareness of the user's proximity to locations that may be of interest. To mark a location of interest, you specify its latitude and longitude. To adjust the proximity for the location, you add a radius. The latitude, longitude, and radius define a geofence, creating a circular area, or fence, around the location of interest.

We can have multiple active geofences, with a limit of 100 per app, per device user. For each geofence, you can ask Location Services to send you entrance and exit events, or you can specify a duration within the geofence area to wait, or *dwell*, before triggering an event. You can limit the duration of any geofence by specifying an expiration duration in milliseconds.

The application retrieves the data from Firebase, analysis it, pins the markers to the Google Map and lets the employer know the location of the employee at any given point of time if required.

By making use of an HR Management application, any organisation would greatly benefit by reducing error rate, decreased workload designated for the management of employees, as well as increase the efficiency of time and error rate for an organisation.

4.13 Attendance Marking Component

From Dashboard, the user will route to the mark attendance activity which shows the location of the user and **Geofence Circle Animation** is also presented on the Google maps. Two buttons are there to mark in and out attendance. User geofence state is determined by the continuously running background service. Upon hitting the marking attendance buttons. Some checks are performed to mark logically correct attendance, as listed below:

- Checking for successive marking of the same type of attendance
- Checking for first in entry before making out an entry
- Checking for time as per policies of the company

When the user geofence state is either enter or dwell and above fulfilling above conditions, the database entry is made and user is notified.

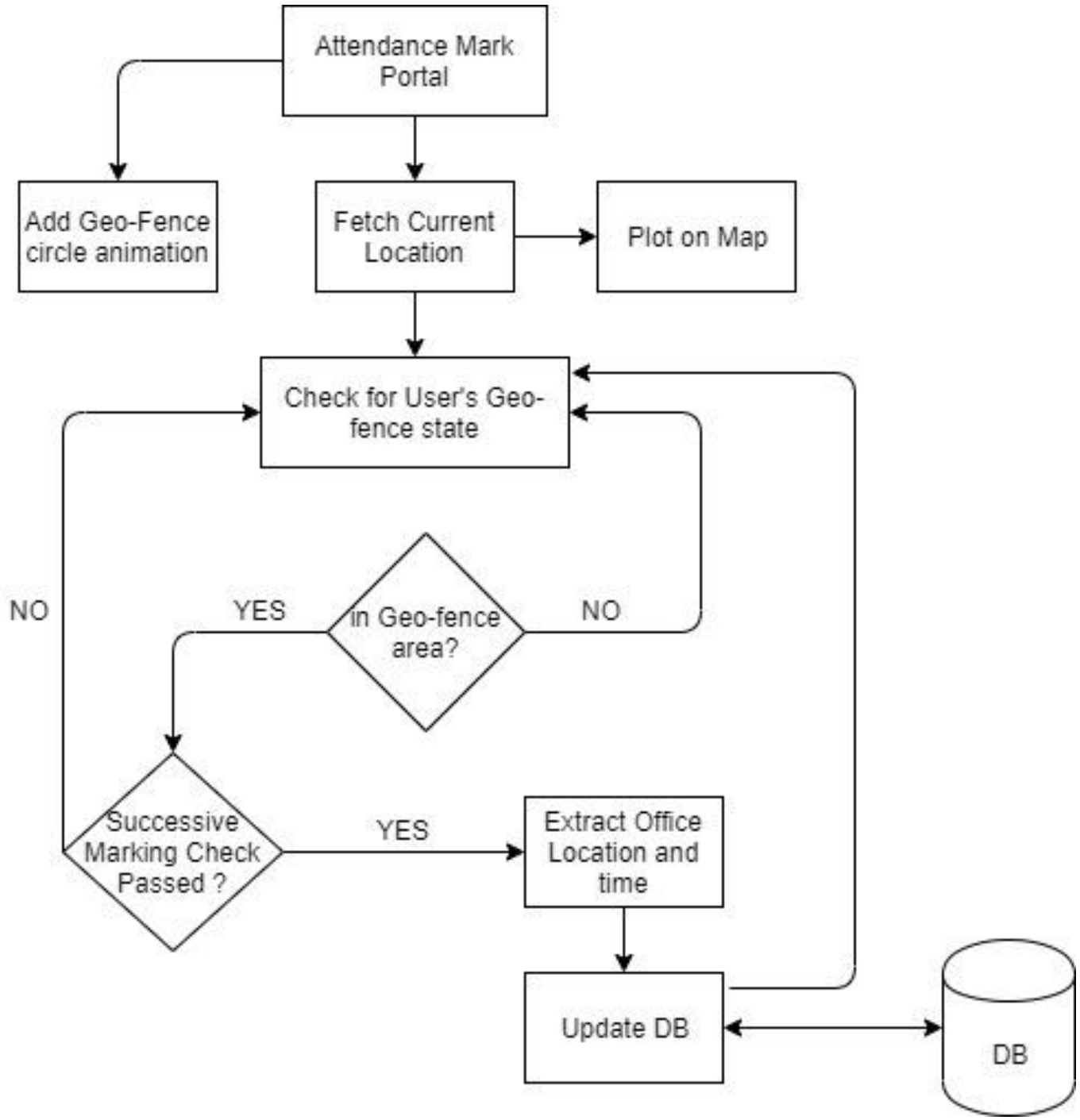

Figure 5 Attendance mark Flow-chart

4.14 Attendance Summary Component (Calendar)

A Calendar component is provided to the user for verifying and checking previous days attendance. The Calendar can be modified according to the convenience, as one week, two weeks or one month. Upon clicking any date, it will display the list of all the previous attendance marked by the employee.

4.15 Leave Management System Implementation

Basic workflow of Leave Application System is described below:

- First basic details are fetched from the database using basic CRUD operations of Google Firebase
- These are fetched using background services where the data fetched is converted into json format
- Then this JSON formatted data is shown by the frontend components of the page.
- Then form is filled by the user
- This data is stored by the models within the application
- This data is then sent back to the database
- Every time an application is submitted firebase generates new ID for that leave
- This ID is then used for the manager side of the leave management system

4.16 Leave Status System Workflow

Basic workflow for Leave Status System is described below:

- The leave ID generated in the previous step is then used to show details regarding the leaves that have been applied
- After the leave has been approved / rejected by the changes are reflected in the database.
- These changes are then reflected on the leave status page.
- Push Notification will be sent to the user using Firebase Cloud Messaging.

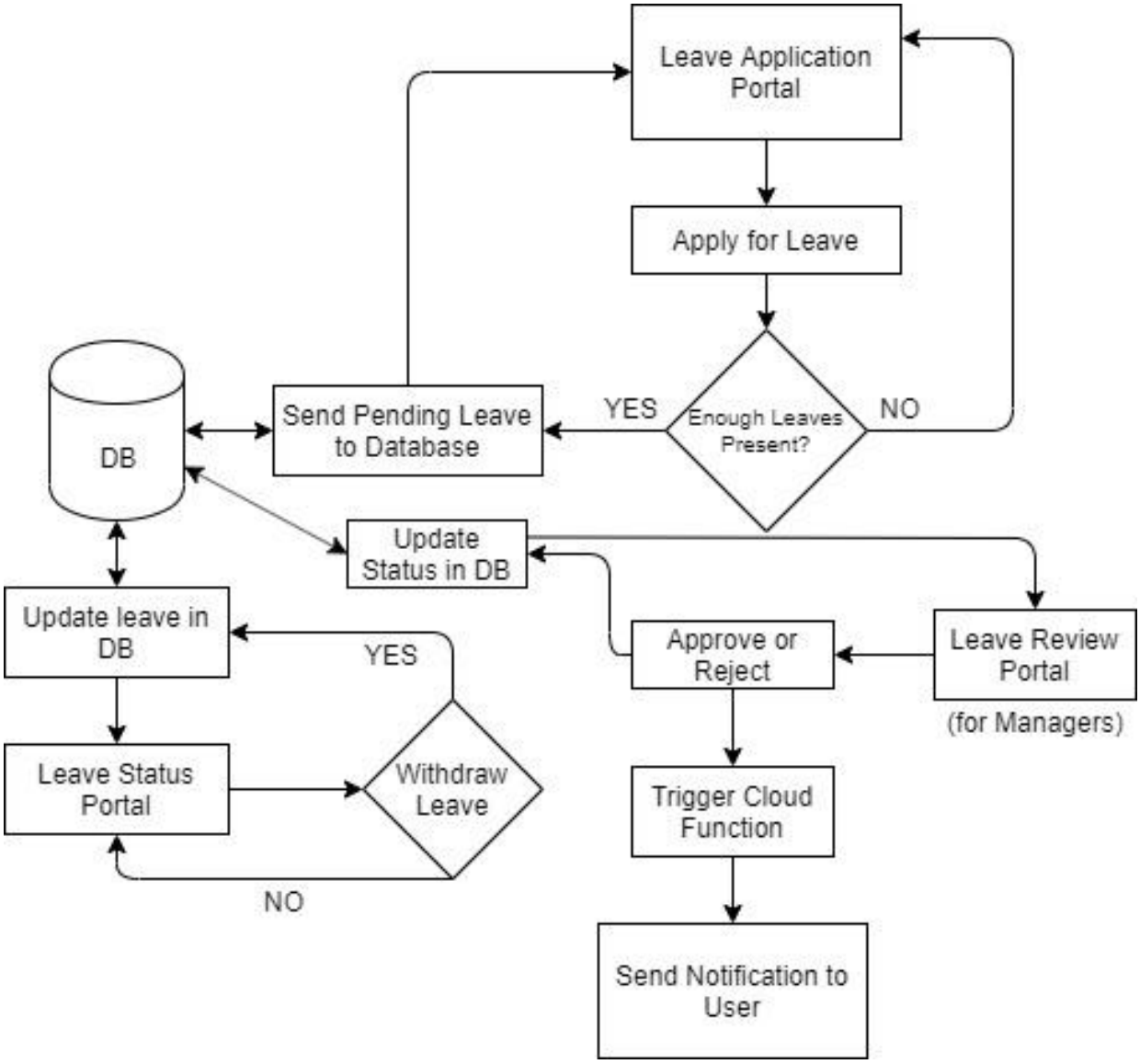

Figure 6 Leave Management Flow chart

4.17 HR App – Add User Workflow

- A form is presented to the HR where he can add new employee details – Employee ID, Name, Email, Password, Phone Number, Address, Office Location and Manger.
- No two Employee IDs, Phone Number and Email can be same.
- All fields are compulsory. Failing to provide any field will show an error.

4.18 HR App – Edit User Workflow

- A list of all the employees will be shown to the HR.
- He can select any employees which will open a new page with existing details of that employee.
- HR can edit those details which will be reflected in the database on saving.

CHAPTER 5: RESULTS AND DISCUSSIONS

5.1 Features Offered by the Project

1. Location-based Attendance for the employee
2. Leave Application System
3. Attendance Summary
4. Leave Summary
5. Leave Approval Facility for Manager
6. HR's Application -
    - Registering new users
    - Management of profiles, official details as well as the office location assignment
    - Edit employee profile

5.2 Location-based Attendance

Attendance will be marked only when the employee enters the radius within which his office lies. This will be done using the geofencing feature developed by enhancing the existing library since the new libraries do not have complete support through flutter. **Geofencing** is a location-based service in which an app or other software **uses** GPS, RFID, Wi-Fi or cellular data to trigger a pre-programmed action when a mobile device or RFID tag enters or exits a virtual boundary set up around a geographical location, known as a **Geofence**.

As there is no Geofence library available in Flutter by default, therefore a **Custom Geofence plugin** was written for the Application for our requirements.

5.3 Leave Management System

The Employee can apply for leaves by specifying various fields i.e. type of leave, reason for leave, and dates for which he shall be absent. The leave request will be approved (or Rejected) by the corresponding manager through a tab which will be only visible to the Manager. When a particular counter of a specific leave reaches zero, employees can no longer use that particular leave type. Each

user can view his leaves classified as annual, casual or medical leaves. Based on how many and if they are left, he can apply for leave. He can also track the status of these leaves if they have been approved or rejected.

5.4 Attendance Summary

Sub-part of the Attendance Management System, it shows and maintains the attendance of the employee and provides an easy to access and read user Interface to see one's own Attendance history. The History can be divided into 1-week, 2-week and a month. He can view his attendance record – i.e. his in time and out-time and his location at that point.

5.5 Leave Summary

Part of Leave Management System, it shows currently requested leaves and details about the leaves. It also has an option to withdraw the leave request.

5.6 HR App

App used by HR consisting of Login Page and a Dashboard from where new employees can be added and details of existing employees can be updated.

5.7 Adding a new Employee

A form where HR can add details of a new employee and add it to the Employee database, assigning or adding his Employee ID in the process.

5.8 Edit existing Employee

Shows list of all the employees and provides a facility to edit the employee details

5.9 Firebase

Firebase provides additional functionality, as apart from providing real-time database support – it also allows us to view certain statistics related to the data.

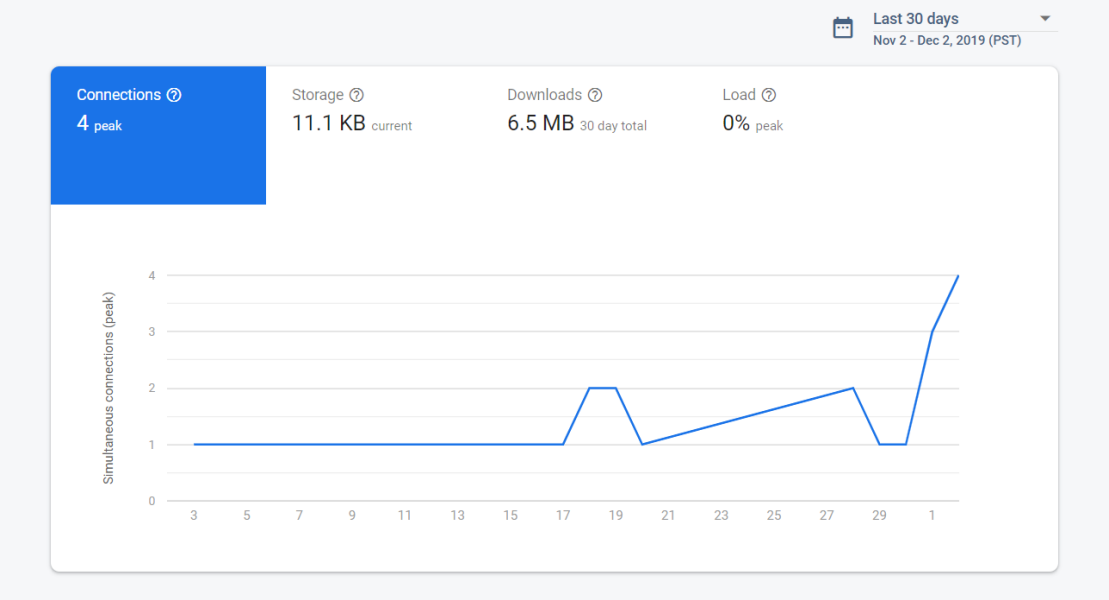

Figure 7 Data Traffic on Database

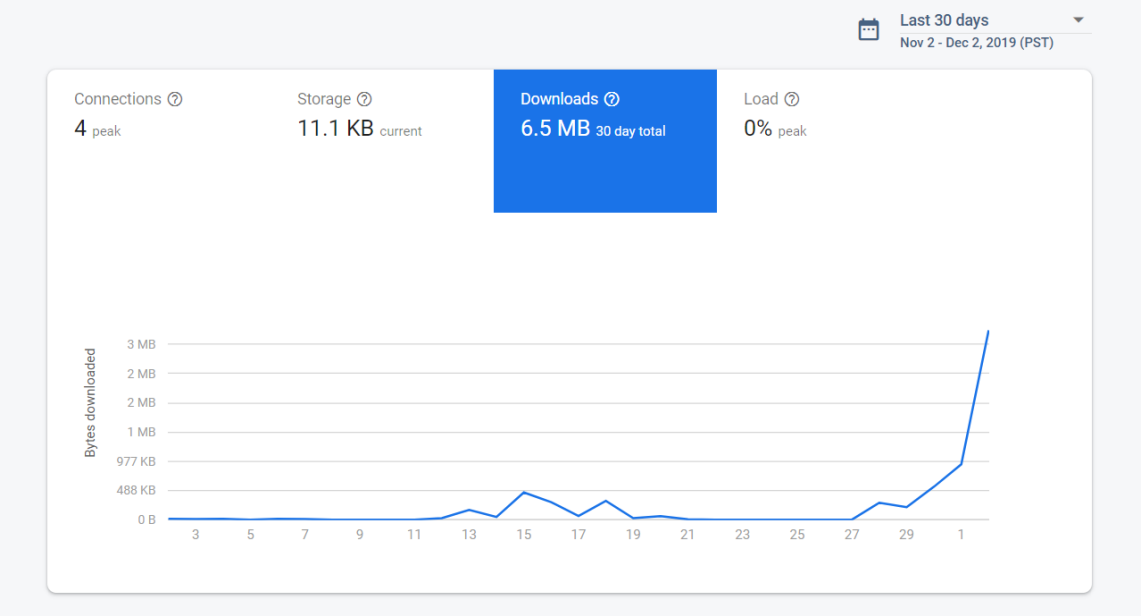

Figure 8 Application Downloads

This graph shows the number of application downloads during the last 30 days. This time period can be adjusted varying from one day to an entire year, hence is beneficial for longer periods of analysis.

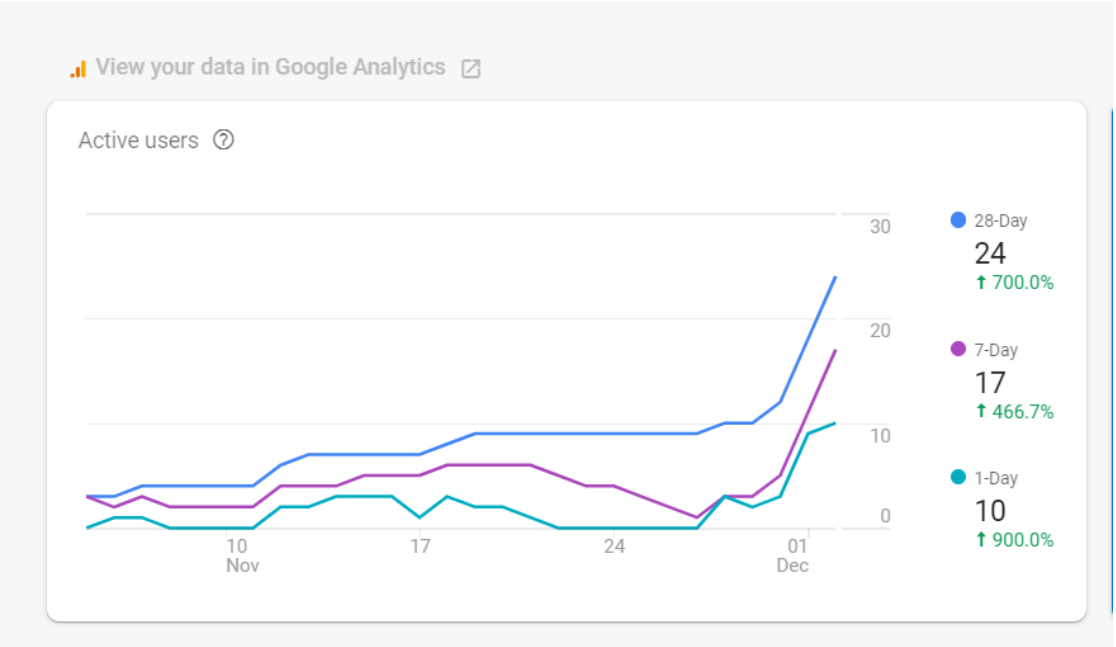

Figure 9 User Activity Graph

This figure represents user Activity in the past 30-days is shown in comparison with activity over the past 7-days and that over the past 1 day, along with the increase percentage.

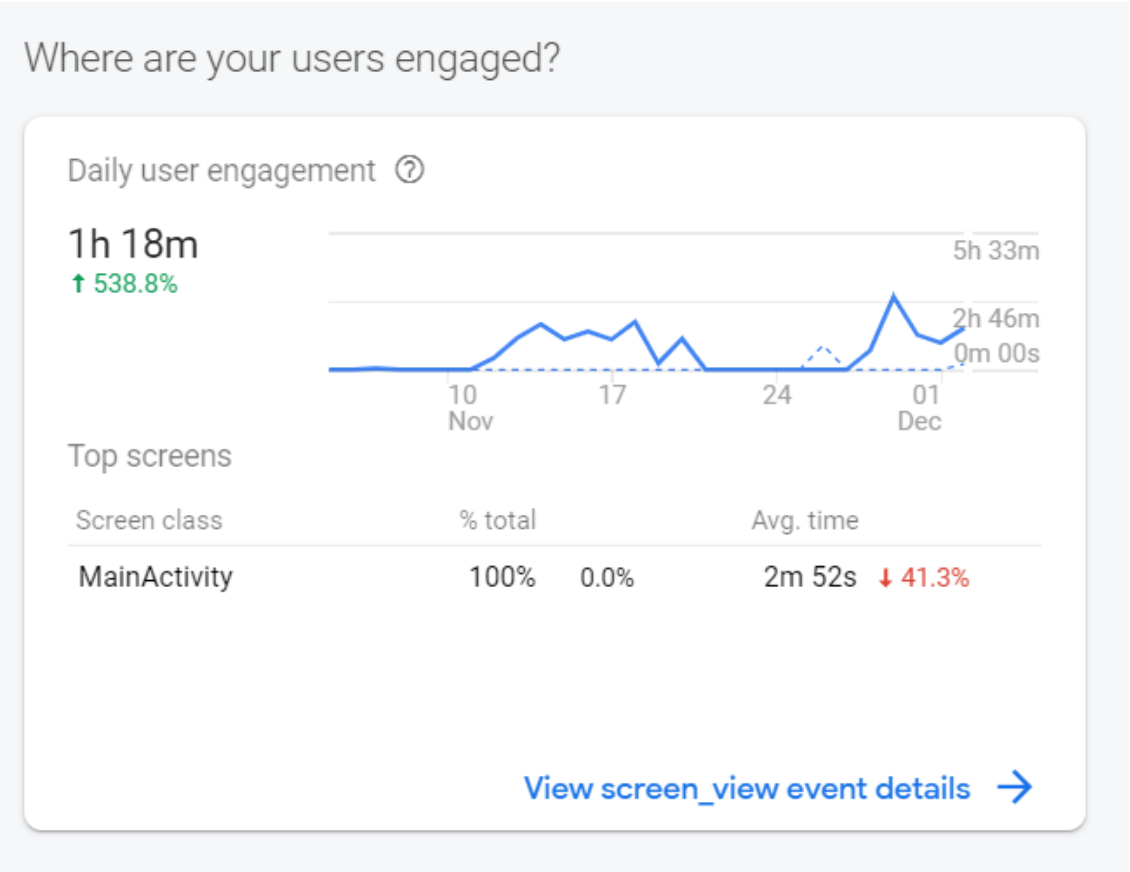

Figure 10 Time Spent by users

An important user statistic provided by Firebase is the time spent on the application, for e.g. the all of the time is spent on the main activity. The graph shows the daily user engagement.

CHAPTER 6: CONCLUSIONS AND FUTURE WORK

- Battery optimization by using a passive state Geofencing in the application so as to enable and disable location from the device when needed only. GPS uses a lot of battery overhead and leads to a very quick drain of battery which would be a problem for employees, especially employees during an on-field duty.
- UI/UX experience could be improved and optimized using better animations, icons as well as loading performance. A simple to use UI that would be easily usable for all age groups and people with varying technical knowledge.
- Code optimization by using better algorithms and data structures. This could also be achieved by using third-party libraries as they would release in the future
- Making the application more modular so as to make it ready to use it in different organizations with different requirements in the future. MVC is used for the structure of the application, thus providing future modularity as well.
- Increasing the Internet Connectivity tolerance of the application so that it can be used with much fluency even during low internet connection speed.
- More Rigorous Testing can be done to remove the slightest of bug still pertaining in the system.
- Checks and constraints will be updated as per the policies and norms of the firm.
- File upload support in HR Application to automate the addition of new employees.
- Automated Notification System using FCM to engage users towards the application

A paramount part in the functioning of any organisation is a streamlined and transparent interface that allows clean documentation of the employees. With this application, the Manager of an organisation would be well aware of the locations and the whereabouts of each employee at any given time of their working hours. As a part of the application, the manager would also be able to access the Regular Attendance Record of each and every employee in the future.

With the help of this application, we were able to provide a much-needed relief to the working at Indus Automotive Pvt. Ltd. Previously, the company was manually compiling the locations and the leaves granted to their workers This caused a major delay in their day to day functions resulting in exhaustion

of time and resources and creating chaos. It also inevitably resulted in the organisation facing losses or hampered outcomes. This created a demand and a requirement for a tool that would be easy to use and greatly reduce the hassle created in the work space as well as outside. This project gave the advantage of an efficient, time-saving and labour-saving method of doing so.

With GeoFlix, many such issues will be solved once the organisation adapts to the technology. The hassle of making calls and consuming time in tasks that our user-friendly application can make the process uncomplicated and efficient. Subsequently, the application nullifies the requirement of manual labour in the process of documentation. We are on our path to completing the development of the application and would be providing the access to the application to begin a trial run in the near future. This would give us the opportunity to get necessary feedback from the organisation and make the required tweaks and changes to ensure an interface that the organisation can duly accept and inculcate in their functioning in the future.

REFERENCES

- https://flutter.dev/docs for the basics of Flutter Framework
- https://api.flutter.dev/ for API Documentation
- https://en.wikipedia.org/wiki/Flutter_(software)
- https://dart.dev/guides/language/effective-dart/documentation
- https://medium.com/flutter/executing-dart-in-the-background-with-flutter-plugins-and-geofencing-2b3e40a1a124 for existing support of background services in Flutter
- https://github.com/bkonyi/FlutterGeofencing for existing support of geofence in Flutter
- https://dart.dev/guides/language/language-tour
- https://sce2.umkc.edu/BIT/burrise/pl/software-process/
- https://mixmastamyk.bitbucket.io/pro_soft_dev/models.html
- https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm
- https://www.guru99.com/mvc-tutorial.html for MVC framework basics
- https://www.smartdraw.com/uml-diagram/ for drawing Dataflow diagram
- https://pub.dev for dependencies
- https://firebase.google.com/ for Real-time database access
- https://firebase.google.com/docs/functions for Integrating Cloud functions for automatically triggering notifications