"Don't try to make right decisions, try to make your decisions right."

List of Figures

Table of Contents

Chapter 1. Introduction

1.1 OVERVIEW:

This project focuses on making it easier for the people to configure their device to their laptops. This will also make it possible for the parents to keep a log of their child's activities.

The user can log into the application and get his mobile phone data onto his laptop by using the same login details.

Also, the parent can create a group and add his child in it. Which will make it possible for him to get all the activities of his child on his on device.

1.2 BACKGROUND AND MOTIVATION:

There are many times when a person does not have his phone around him and he might miss out on any important notification or message. So we came up with an idea to solve this problem by remotely getting all the device messages on the person's system. Also. The there are times when a parent wants to know what their child is doing on his phone to keep him from misusing it. So an application which would give them that information will prove out to be quite helpful.

1.3 PROBLEM STATEMENT AND OBJECTIVE:

Checking on their phones while in office or when not having their phone's around is a complicated task. There is a chance that the person might miss out on some important information. Also keeping an eye on their child's activity was also not possible. They mostly get distracted from studies due to due their phones. This will help the parents to keep a log of his activities.

The application has the following objectives:

Objective 1: To develop an interface which make all the phone data of user available on their desktop systems.

Objective 2: To provide an interface for parents to monitor their children activities on their mobile devices.

Objective 3: To develop an interface on which user can configure and access all his mobile devices

Objective 4: To provide a "find my device" feature through this

Objective 5: To develop an interface through which user can notify/remind other devices by sending notifications.

1.4 SCOPE OF THE PROJECT:

The clear target users for our app are the employees of the company who cannot bring their phone to the working premises and the parents who want to keep an eye on their child's activity. This won't let the user's miss out on any important information during their work hour.

This application will help the user to:

- **Getting the phone notification on laptops:**

This application will allow the user to get all of his phone's notification on the laptop. Mostly, the employees in the company don't have their phones around them. So, this application will give them all the notification on the laptop and save them from missing out on important things.

- **Keeping a log of child's activities:**

This application has a parental mode which will allow the parents to see which application their child used and at what time. This will definitely create a fear in the child's mind and his mobile usage will decrease.

- **Connecting all the devices:**

This will allow a person to create a group and connect all his devices in that single group

1.5 TEAM ORGANISATION:

A **Project Team** is an organized group of people who are involved in performing shared/individual tasks of the project as well as achieving shared/individual goals and objectives for the purpose of accomplishing the project and producing its results. The team consists of the full-time and part-time human resources supposed to collaboratively work on producing the deliverables and moving the project towards successful completion.

A group of people turns into a team when every person of the group is capable of meeting the following conditions:

- Understanding the work to be done within the endeavour.
- Planning for completing the assigned activities.
- Performing tasks within the budget, timeline, and quality expectations.
- Reporting on issues, changes, risks, and quality concerns to the leader.
- Communicating status of tasks.
- Being a person who can jointly work with others.

Our project team consist of although 6 persons with different roles and responsibility explained below:

- Sahaj Sugandhi

  Along with doing preliminary investigation, I did the backend for getting all the mobile notifications on firebase. Then worked on Firebase Hosting and development of Web Application, I studied about firebase, firestone and real time database to achieve the required goal.

- Shahjahan Khan

  I did the research for choosing a technology that could solve our problem and be suitable for making the application. I also helped in making the web interface.

- Shirin Shah

  Helped in doing preliminary investigation. Did the study of current existing systems and played a role in coming up with the idea for project. Along with this, I made the research paper and did the documentation work.

- Shrayansh Jain

  Along with doing preliminary research, I helped in connecting the whole project to firebase and also set up the firebase assistant.

- Soumya Joshi

I did the research work for the project. Then came with the organization and structure for it. Also, I helped in preliminary investigation.

- Vaidik Khandelwal

Did the study of current available systems. Played a part in coming up with the idea of application. Worked on the front end and back end of the parenting module.

1.6 REPORT STRUCTURE

The project Mobile Monitor focuses on maintaining the attendance record, time table and maintain college news updates and the project report is divided into five chapters.

Chapter 1: Introduction: This chapter starts by giving an overview of the project. Then it gives the background of the application. This chapter discusses about the idea of the project. It describes the problem and side by side also discusses the scope of the project. This chapter ends with giving the details of the team members and the roles played by them.

Chapter 2: Review of Literature: The chapter starts by giving the details about the preliminary work done for developing the application. It then gives details about the current system. Then it gives the drawbacks of the existing system which needs to be worked upon and which provide the basis for making the application. The chapter then includes the requirement analysis of the project. Providing a basis to work upon.

Chapter 3: Proposed System: This chapter starts by giving the proposal of the project and then gives the benefits of the proposed system. This chapter shows the block diagrams of the project. Then it covers the different types of feasibility studies to show that the project can be made with the existing tools. Then this chapter gives details about the data structure used and also shows the data flow diagrams. Then it ends by giving the hardware and software requirements for deploying the project.

Chapter 4: Implementation: This chapter gives the knowledge about the technologies and tools used in the development of the project. This also briefs the reader about the languages used in this. Then it also shows some screen shots of the developed project. The chapter ends by describing the testing methodologies used in the testing of the application.

Chapter 5: Conclusion: This chapter gives the conclusion of the report and also tells about the limitations of the work. This chapter also includes future scope of the project.

Chapter 2. Review of Literature

In this chapter it is explained about the application and software that can perform the similar task which can mobile monitor perform, and their limitations because of which we came up with the idea of mobile monitor.

2.1 PRILIMINARY INVESTIGATION:

2.1.1 CURRENT SYSTEM:

The application combines the functionality of multiple applications into one so no single system is a perfect comparison:

- **iMessaging:**

This is a MacOS application. It can Sync Messages from an iPhone, iPad and iPod. It also allows you to send and receive SMS & MMS through your Apple device. But this application can be used by apple device owners only.

- **Your Phone(Windows 10):**

This connects to your Android/Windows mobile devices and allows users to access their text messages and even photos, This application is available for android and windows as well.

2.2 LIMITATIONS OF THE CURRENT SYSTEM

- The current iMessaging doesn’t allow the users of android devices to connect their phones and laptops
- Both of these also do not have a child monitoring system which can help the parent with keeping logs.
- Also, there is no such thing as groups in it.
- They do not tell the parent about the application their child used and at what time.

2.3 REQUIREMENT IDENTIFICATION AND ANALYSIS FOR THE PROJEECT

After studying these current applications, we came to a conclusion that the people need an application that can configure their mobile phones and laptops and get the notification on the laptop itself. Also, an application could be made which will allow the parents to see what their child is doing on his phone.

Also, it is not convenient for the office employees, who cannot get their phones in the office, to get all the notification on their laptop. So that they do not miss out on any important details.

Chapter 3. Proposed System

In this chapter the proposed system is explained which covers all the limitations of all the already existing systems. A proper feasibility study is carried out for this system in all terms and the deployment requirements are attached at the end.

3.1 THE PROPOSAL

The concept is to introduce an android application that can assist and simplify a mobile user's life. From getting all the mobile notification on your laptop to keeping an eye on your children, the application will have it all covered. The application will notify the user's about its phone's information such as battery status; miscall info and message from the Android device to web application. Also it will give the log of a phone to the user as per the requirement.

3.2 BENEFITS OF THE PROPOSED SYSTEM

The proposed system will not only solve the problems of the working people but will also help the parents in keeping an eye on their child's activities.

Also, they wouldn't have to worry about if their child is using the phone and not studying. The app will give them all the information.

The employees will also not miss out on any important information in their working hours and when they do not have their phone around.

3.3 BLOCK DIAGRAM

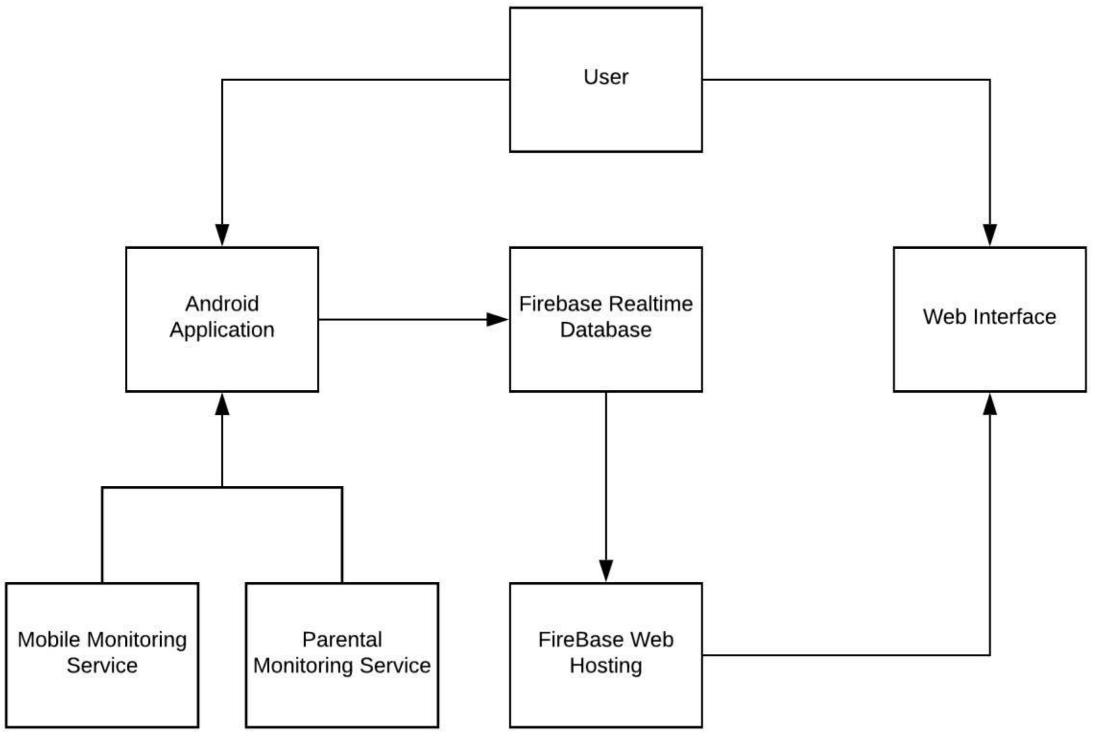

User
Android Application
Firebase Realtime Database
Web Interface
Mobile Monitoring Service
Parental Monitoring Service
FireBase Web Hosting

Figure 3.1 Block Diagram

As demonstrated in fig 3.1 Block Diagram The user can interact with android application and web interface and the services will send data to the application which application will transfer to firebase real time database and then firebase will send it to web hosting which will be displayed by the web interface hosted on firebase web hosting.

3.4 FEASIBILITY STUDY

A feasibility study is an analysis of how successfully a system can be implemented, accounting for factors that affect it such as economic, technical and operational factors to determine its potential positive and negative outcomes before investing a considerable amount of time and money into it.

3.4.1 TECHNICAL FEASIBILITY

The application works on android platform. It will need active internet connection to work properly because the data is fetched and stored to real time data base. The android is user friendly and also robust so our application cannot be easily crashed and it has quite good security.

In the front-end XML is basically used along with some images an icon. We have also used android studio editor. We have used emulator also sometimes to run the application. The project is technically feasible.

3.4.2 ECONOMICAL FEASIBILITY

Each dynamic application required a data house which gives the application the power to store data and perform operations on it. These data houses or database should be real time, structured and with good processing power. The establishment of data house or database is the costliest thing in the development of application. So for this, we have opted for Firebase's Real-Time Database.

Firebase Real Time Database costs a minimal amount which can be purchased at the time of deployment of the application and is scalable.

For the usage of application an Android device is required which supports API level 22 which is Android Lollipop 5.1.

For the session resolutions and work functions Shared Preferences are used which works on every Android Device and also we used firebase for hosting which does not need any server maintenance and is available at minimal cost.

3.4.3 OPERATIONAL FEASI BILITY

The application is quite easy to operate. And the people using it are mostly office going employees, some parents and their children. One needs only the basic knowledge of using android phones and laptops. To serve this motto of easy operation we have kept the UX and UI of application very clean and user friendly.

3.5 DESIGN REPESENTATION

Following are some design representations in UML diagrams:

3.5.1 DATA FLOW DIAGRAM

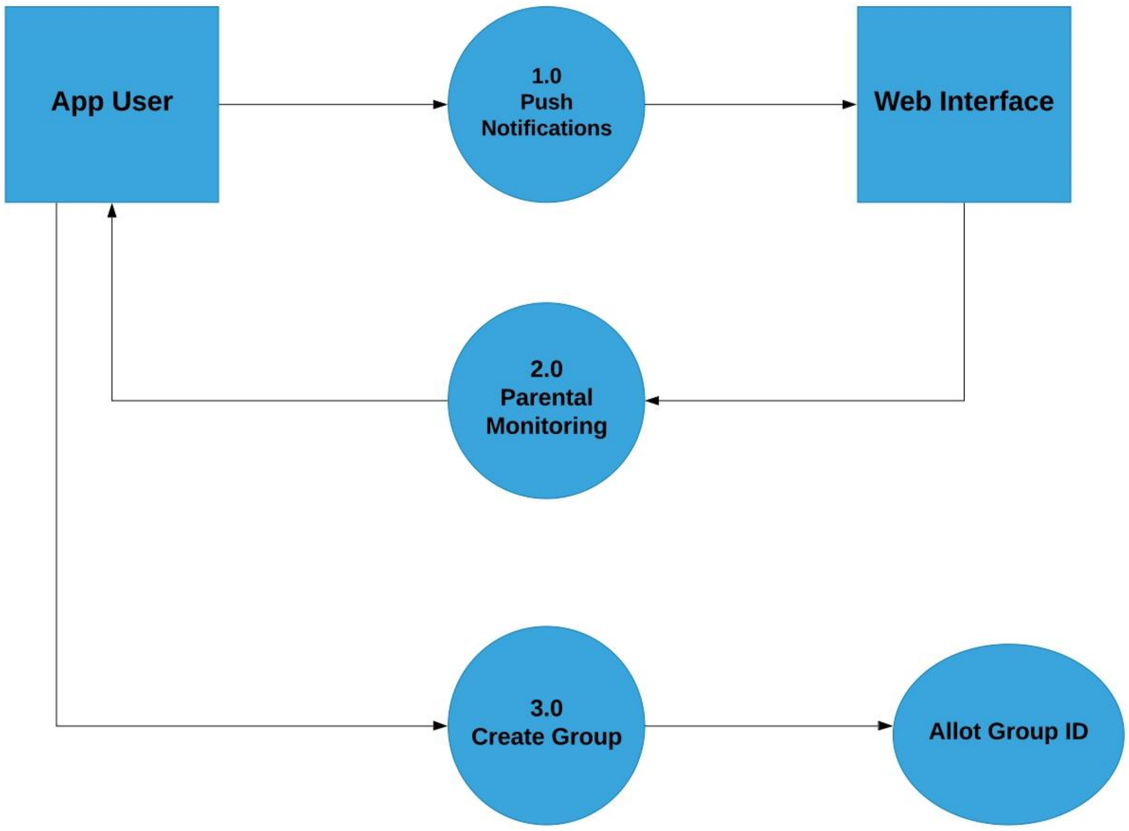


Figure 3.5.1: Level-1 Data flow diagram

This UML diagram represents the flow of data of our system's processes. Also, it provides the inputs and outputs of each entity and the process itself. In simple words it is describing the information system and control flow of our system.

3.5.2 USE CASE DIAGRAM

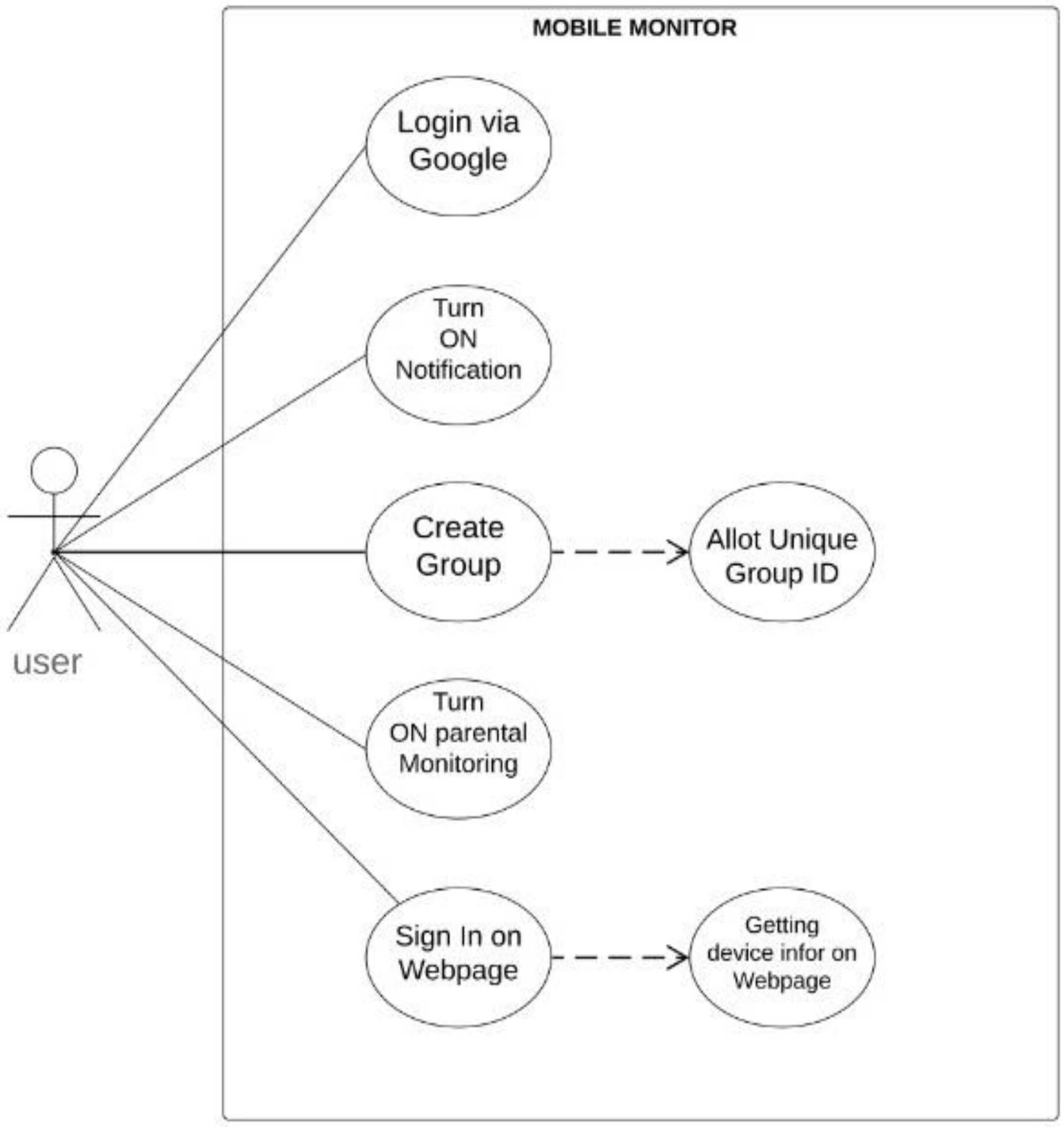


Figure 3.5.2: Use Case Diagram

This UML diagram represents the dynamic behaviour of our system. This use case diagram or behaviour diagram models the functionality of our system by the actors and use cases specified in the diagram. These use cases i.e. the use cases specified above shows the set of services and functions, that our system performs and also shows who performs this use cases on the system.

3.5.3 SEQUENCE DIAGRAM

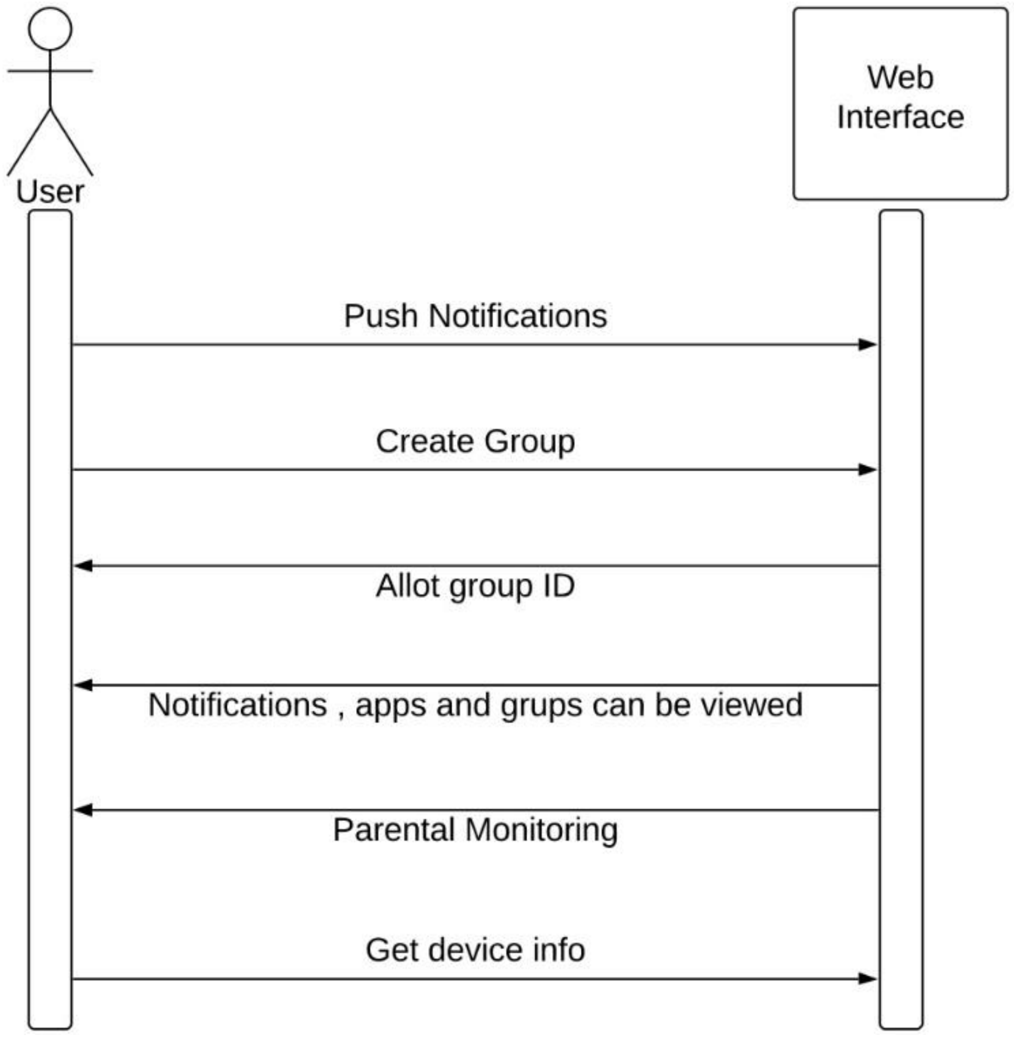
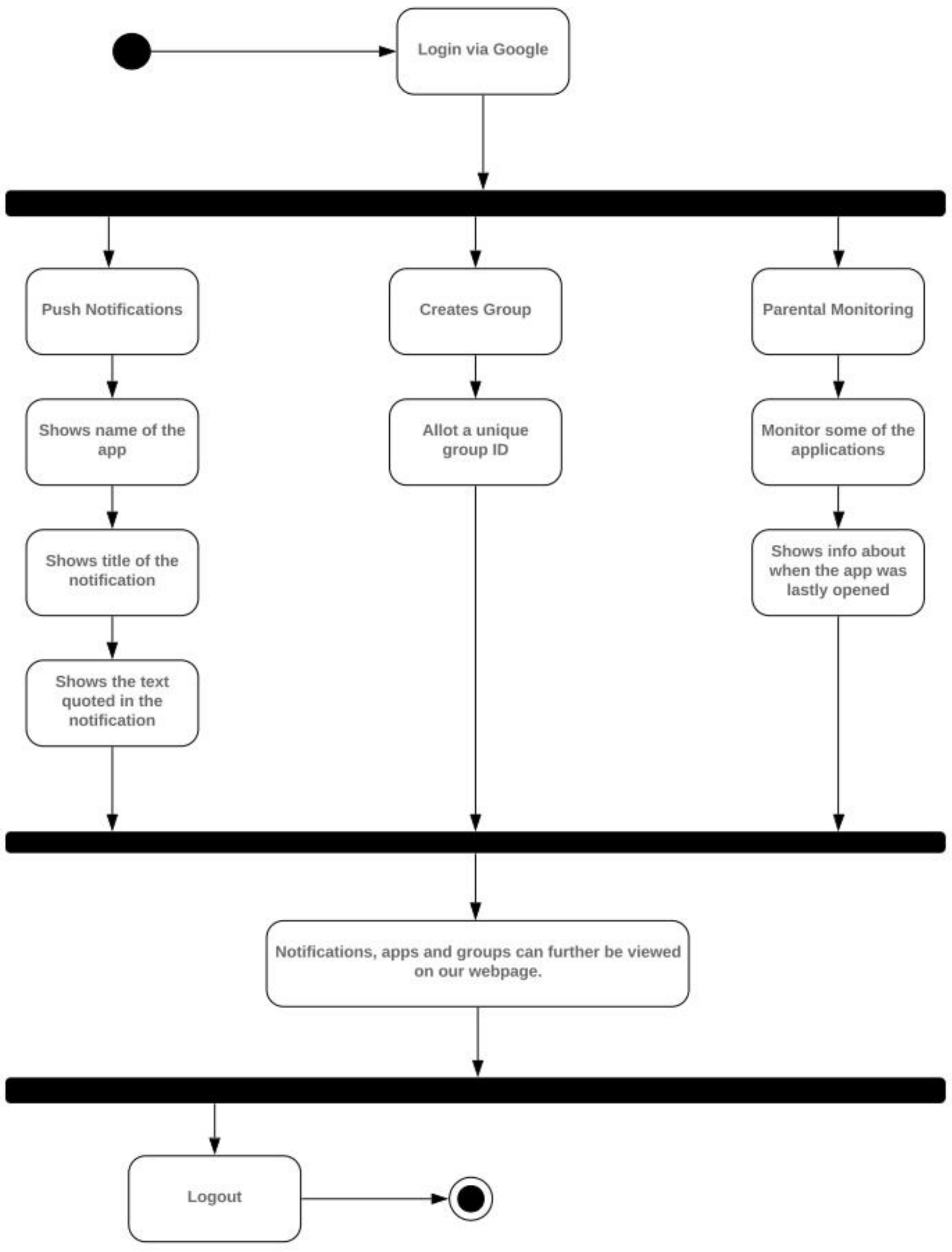


Figure 3.5.3: Sequence diagram

This UML diagram simply depicts i.e. shows the interaction between the objects present in the system in the sequential order i.e. in the order in which these interactions are going to take place. This describes how and in what order the developed system is functioning.

3.5.4 ACTIVITY DIAGRAM



Login via Google
Push Notifications
Creates Group
Parental Monitoring
Shows name of the app
Allot a unique group ID
Monitor some of the applications
Shows title of the notification
Shows info about when the app was lastly opened
Shows the text quoted in the notification
Notifications, apps and groups can further be viewed on our webpage.
Logout

Figure 3.5.4: Activity Diagram

This UML diagram describes the dynamic aspects of the developed system. This is basically describing the flow of the system from one activity to another activity. The activities specified above describes the operations of the developed system and also this UML diagram is describing the control flow from one activity to another activity in our system.

3.6 DEPLOYMENT REQUIREMENTS

There are various requirements to successfully deploy the system. These are mentioned below:

3.6.1 HARDWARE

3.6.1.1 DEVELOPER

- A Virtual or Physical Machine for firebase management.
- A Machine for Server management and Firestore.
- High processing computer for server scouting and editing purposes.

3.6.1.2 USER

- Minimum 2 GB of Ram
- Minimum 50-100 Mb of space in Primary Memory
- Internet connection

3.6.2 SOFTWARE

3.6.2.1 DEVELOPER

- Android Studio 3.0 or above.
- Google Chrome Browser.
- Google ML Kit Support
- Notepad++
- Blocks

3.6.2.2 USER

- Android OS with API Level 22 or above.
- Minimum OS Requirement of 5.0
- Internet Connection

Chapter 4. Implementation

Implementation part will carry all the relevant information regarding the project requirements in terms of hardware and software. All the technologies used are mentioned and explained.

Screenshots for the same are attached at the end.

4.1 TECHNOLOGIES USED

4.1.1 FIREBASE

Firebase is a technology that allows you to make web applications with no server-side programming so that development turns out to be quicker and easier. It supports the web, iOS, OS X, and Android clients. Applications using Firebase can just utilize and control information, without thinking about how information would be put away, and synchronized crosswise over different examples of the application in real time. With Firebase, you don’t need to stress over-provisioning servers or building REST APIs with just a little bit of configuration; you can allow Firebase to make a chance to take every necessary step: storing data, verifying users, and implementing access rules.

There are various benefits of Firebase that relate to the core technology of advancement.

- Firebase Real-time Database
- Firebase Auth
- Firebase AuthUI
- Firebase Storage
- Firebase Analytics
- Firebase Hosting

4.1.2 ANDROID (JAVA)

Android is a mobile operating system (OS) first developed by a Silicon Valley company by the name of Android Inc. A collaboration spearheaded by Google in 2007 through the Open Handset Alliance (OHA) gave Android an edge in delivering a complete software set, which includes the main OS, middleware and specific mobile application, or app.

Patterned after the Linux kernel, the Android also was released as open source code. Development for the Android may be done through Windows, Linux or Mac. Although primarily written in Java, there is no Java Development Machine (JDM) in the platform.

Instead of allowing Java programs to run through the JDM, Google developed Dalvik, a virtual machine specifically for the Android. Dalvik runs recompiled Java code and reads it as Dalvik bytecode and was designed to optimize battery power and maintain functionality in an environment with limited memory and CPU power, such as that of mobile phones, netbooks and tablet PCs.

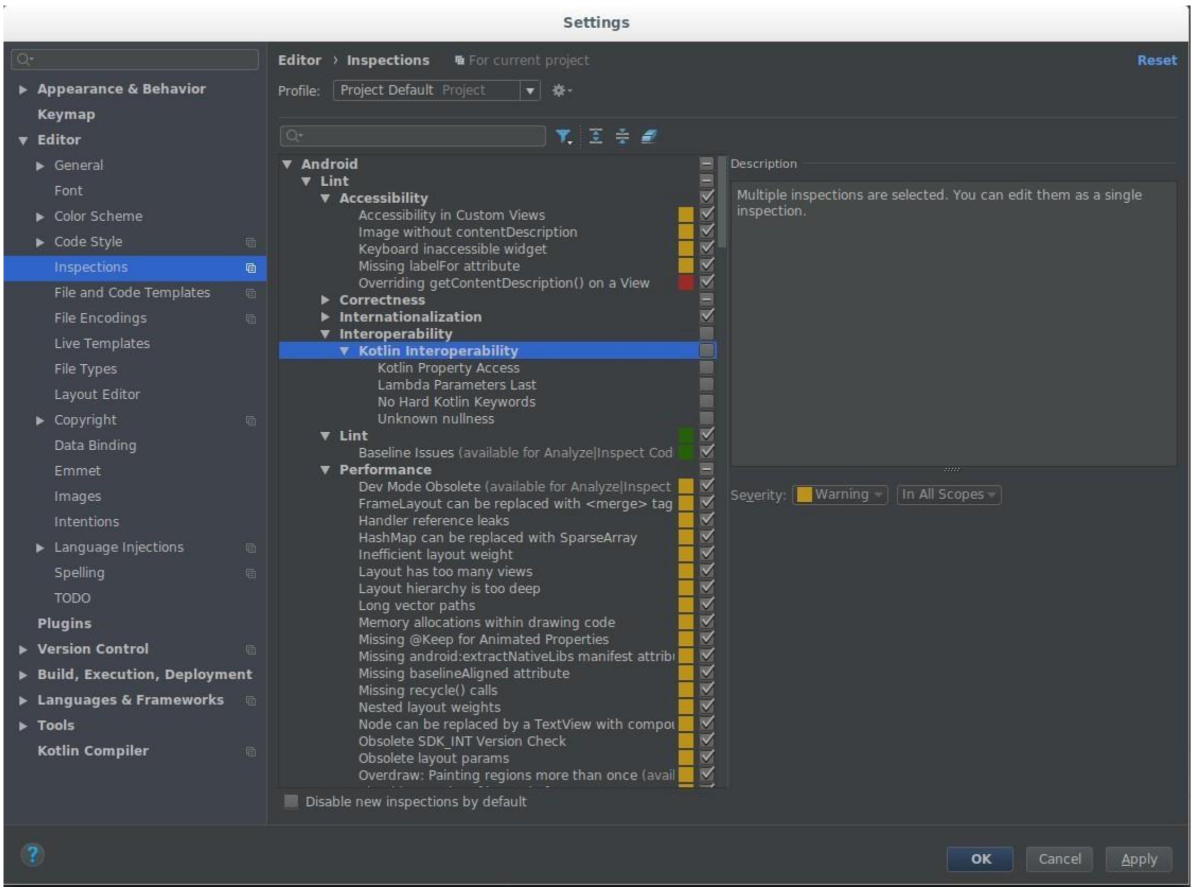
One of the Android's selling points is an ability to break down application boundaries. Another advantage is that it is easily developed, not to mention its speed of app development. A large community of developers continuously devises and designs apps that enhance the capability of devices. These apps are then made available worldwide through Google's Android Market, or other third- party sites.

4.2 TOOLS USED

4.2.1 ANDROID STUDIO

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development.

Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0. The current stable version is 3.2.1, which was released in October 2018.


Settings

Editor › Inspections   For current project   Reset

Profile: Project Default Project

- ▶ Appearance & Behavior
- Keymap
- ▼ Editor
  - ▶ General
  - Font
  - ▶ Color Scheme
  - ▶ Code Style
  - Inspections
  - File and Code Templates
  - File Encodings
  - Live Templates
  - File Types
  - Layout Editor
  - ▶ Copyright
  - Data Binding
  - Emmet
  - Images
  - Intentions
  - ▶ Language Injections
  - Spelling
  - TODO
- Plugins
- ▶ Version Control
- ▶ Build, Execution, Deployment
- ▶ Languages & Frameworks
- ▶ Tools
- Kotlin Compiler

```
▼ Android
  ▼ Lint
    ▼ Accessibility
        Accessibility in Custom Views
        Image without contentDescription
        Keyboard inaccessible widget
        Missing labelFor attribute
        Overriding getContentDescription() on a View
    ▶ Correctness
    ▶ Internationalization
    ▼ Interoperability
      ▼ Kotlin Interoperability
          Kotlin Property Access
          Lambda Parameters Last
          No Hard Kotlin Keywords
          Unknown nullness
    ▼ Lint
        Baseline Issues (available for Analyze|Inspect Cod
    ▼ Performance
        Dev Mode Obsolete (available for Analyze|Inspect
        FrameLayout can be replaced with <merge> tag
        Handler reference leaks
        HashMap can be replaced with SparseArray
        Inefficient layout weight
        Layout has too many views
        Layout hierarchy is too deep
        Long vector paths
        Memory allocations within drawing code
        Missing @Keep for Animated Properties
        Missing android:extractNativeLibs manifest attrib
        Missing baselineAligned attribute
        Missing recycle() calls
        Nested layout weights
        Node can be replaced by a TextView with compou
        Obsolete SDK_INT Version Check
        Obsolete layout params
        Overdraw: Painting regions more than once (avail
```

Description

Multiple inspections are selected. You can edit them as a single inspection.

Severity: Warning   In All Scopes

Disable new inspections by default

OK   Cancel   Apply

Figure 5.1 Android Studio IDE

4.2.2 FIREBASE CONSOLE

Firebase evolved from Envolve, a prior start-up founded by James Tamplin and Andrew Lee in 2011. Envolve provided developers an API that enables the integration of online chat functionality into their websites. After releasing the chat service, Tamplin and Lee found that it was being used to pass application data that weren't chat messages. Developers were using Envolve to sync application data such as game state in real time across their users. Tamplin and Lee decided to separate the chat system and the real-time architecture that powered it. They founded Firebase as a separate company in April 2012.

Firebase Inc. raised seed funding in May 2012. The company further raised Series A funding in June 2013. In October 2014, Firebase was acquired by Google. In October 2015, Google acquired Divshot to merge it with the Firebase team. Since the acquisition, Firebase has grown inside Google and expanded their services to become a unified platform for mobile developers. Firebase now integrates with various other Google services to offer broader products and scale for developers. Firebase Console manages the Firebase platform.

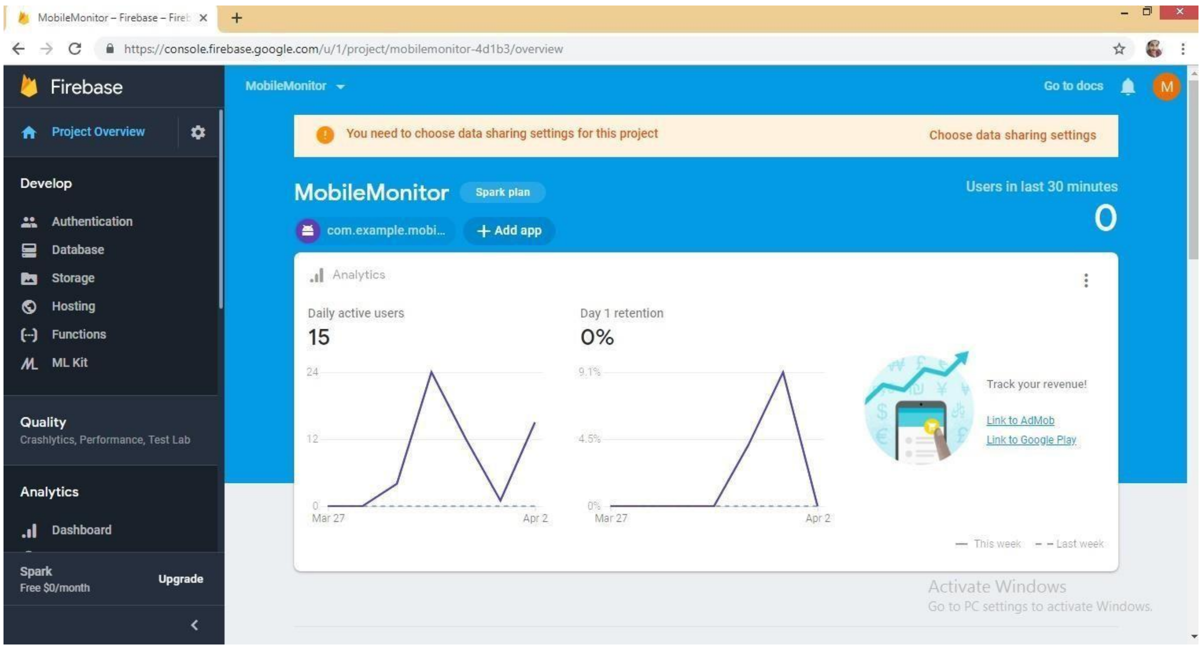


Figure 5.2 Firebase Dashboard

4.2.3 MODELS

MVC

The Model View Controller (MVC) design pattern specifies that an application consist of a data model, presentation information, and control information. The pattern requires that each of these be separated into different objects.

MVC is more of an architectural pattern, but not for complete application. MVC mostly relates to the UI / interaction layer of an application. You're still going to need business logic layer, maybe some service layer and data access layer.

UML Diagram MVC Design Pattern

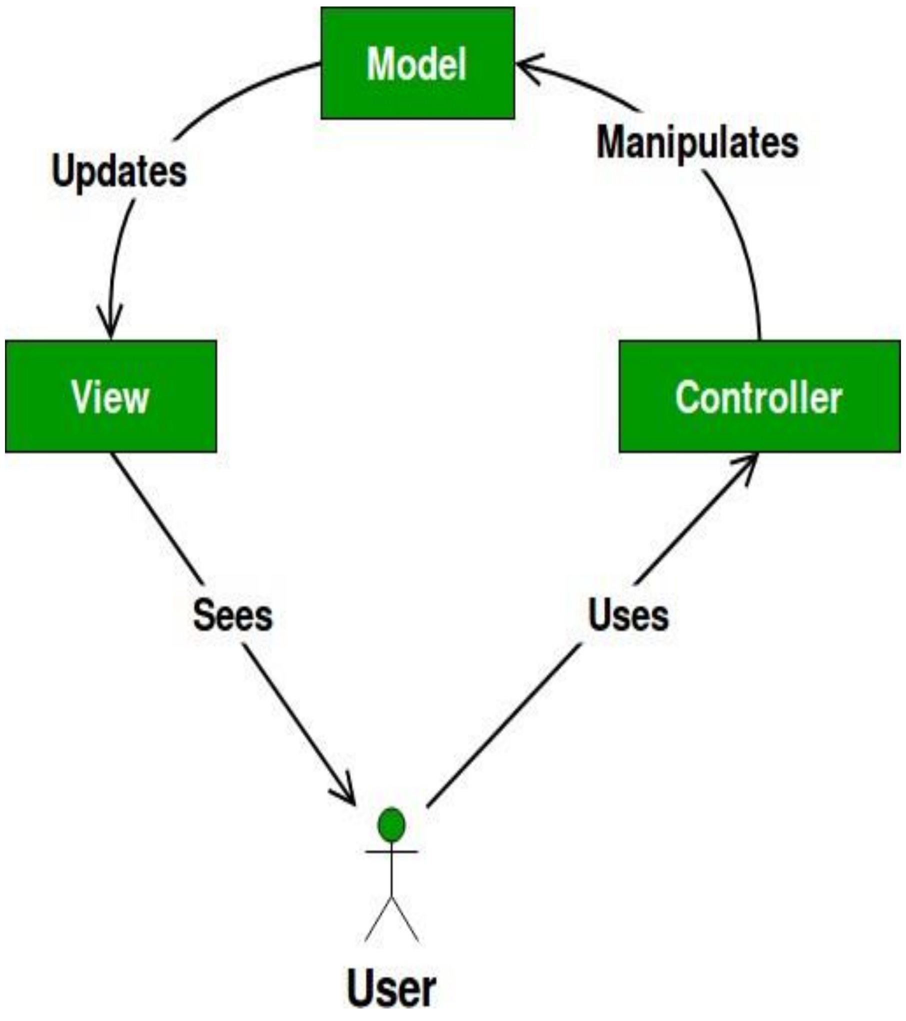


Figure 5.3 MVC Design Pattern

Design components

- The Model contains only the pure application data, it contains no logic describing how to present the data to a user.

- The View presents the model's data to the user. The view knows how to access the model's data, but it does not know what this data means or what the user can do to manipulate it.

- The Controller exists between the view and the model. It listens to events triggered by the view (or another external source) and executes the appropriate reaction to these events. In most cases, the reaction is to call a method on the model. Since the view and the model are connected through a notification mechanism, the result of this action is then automatically reflected in the view.

Advantages

- Multiple developers can work simultaneously on the model, controller and views.
- MVC enables logical grouping of related actions on a controller together. The views for a specific model are also grouped together.
- Models can have multiple views.

Disadvantage

- The framework navigation can be complex because it introduces new layers of abstraction and requires users to adapt to the decomposition criteria of MVC.
- Knowledge on multiple technologies becomes the norm. Developers using MVC need to be skilled in multiple technologies.

4.3 LANGUAGE USED

The app is based on android platform, where we have used java as the development language. Originally developed by Android Inc. and subsequently purchased by Google.

Basically, Android is thought of as a mobile operating system. But it is not limited to mobile only. It is currently used in various devices such as mobiles, tablets, televisions etc.

Android provides a rich application framework that allows us to build innovative apps and games for mobile devices in a Java language environment.

The Android open-source software stack consists of Java applications running on a Java-based, object-oriented application framework on top of Java core libraries running on a Dalvik virtual machine featuring JIT compilation.

Figure 5.4 Android logo and mascot

Features of Android

There are numerous features of android. Some of them are listed below:

- Connectivity

Android supports multiple connectivity technologies including GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and WiMAX

- Storage

SQLite, a lightweight relational database, is used for data storage purposes

- Media support

Android supports various type of audio/video/still media formats like: H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, OggVorbis, WAV, JPEG, PNG, GIF, BMP and WebP

- Web browser

The web browser available in Android is based on the open-source Blink (previously WebKit) layout engine, coupled with Chrome's V8 JavaScript engine supporting HTML5 and CSS3.

- Multi-tasking

Multitasking of applications, with unique handling of memory allocation, is available, using this user can jump from one task to another and at the same time various application can run simultaneously

- Resizable widgets

Widgets are re-sizable, so users can expand them to show more content or shrink them to save space

- Multi-touch

Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero

- Wi-Fi

A technology that lets apps discover and pair directly, over a high-bandwidth peer-to-peer connection.

- Screen capture

Android supports capturing a screenshot by pressing the power and home- screen buttons at the same time. This feature supports after Android 4.0

- Android Beam

A popular NFC-based technology that lets users instantly share, just by touching two NFC enabled phones together

- Multi-Language

Android supports multiple languages, also supports single direction and bi- directional text.

Why Android?

There are so many reasons you should choose Android platform for mobile application development.

- Zero/negligible development cost

The development tools like Android SDK, JDK, and Eclipse IDE etc. are free to download for the android mobile application development. Also, Google charge a small fee $25, to distribute your mobile app on the Android Market.

- Open Source

The Android OS is an open-source platform based on the Linux kernel and multiple open source libraries. In this way developers are free to contribute or extend the platform as necessary for building mobile apps which run on Android devices.

- Multi-Platform Support

In market, there are a wide range of hardware devices powered by the Android OS, including many different phones and tablet. Even development of android mobile apps can occur on Windows, Mac OS or Linux.

- Multi-Carrier Support

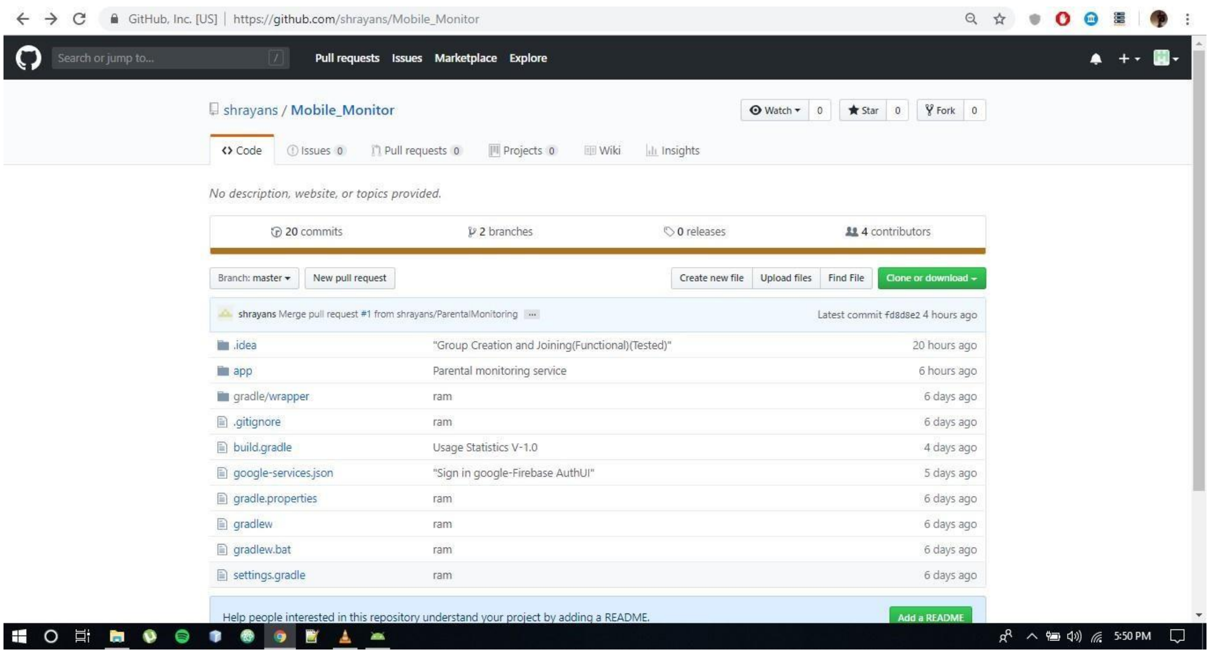
World wide a large number of telecom carriers like Airtel, Vodafone, Idea Cellular, AT&T Mobility, BSNL etc. are supporting Android powered phones.

- Open Distribution Model

Android Market place (Google Play store) has very few restrictions on the content or functionality of an android app. So, the developer can distribute theirs app through Google Play store and as well other distribution channels like Amazon's app store.

4.4 SCREENSHOTS

Tools and IDE


GitHub, Inc. [US] | https://github.com/shrayans/Mobile_Monitor

shrayans / Mobile_Monitor

No description, website, or topics provided.

20 commits · 2 branches · 0 releases · 4 contributors

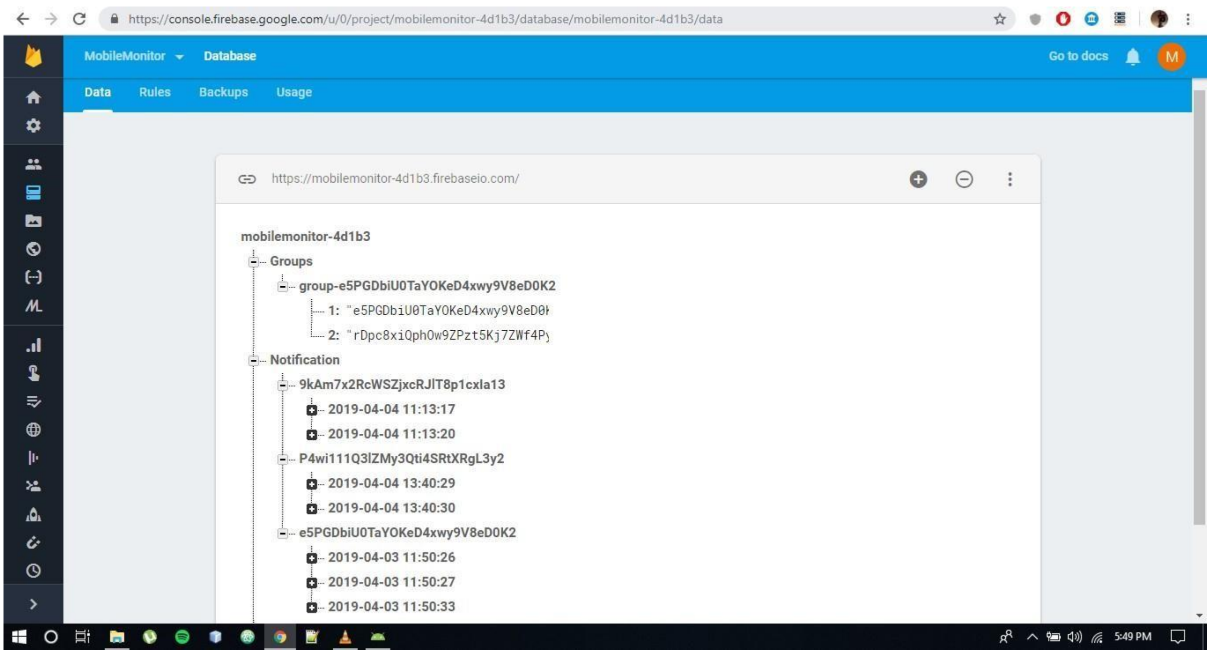
Branch: master · New pull request · Create new file · Upload files · Find File · Clone or download

shrayans Merge pull request #1 from shrayans/ParentalMonitoring · Latest commit fd8d8e2 4 hours ago

.idea	"Group Creation and Joining(Functional)(Tested)"	20 hours ago
app	Parental monitoring service	6 hours ago
gradle/wrapper	ram	6 days ago
.gitignore	ram	6 days ago
build.gradle	Usage Statistics V-1.0	4 days ago
google-services.json	"Sign in google-Firebase AuthUI"	5 days ago
gradle.properties	ram	6 days ago
gradlew	ram	6 days ago
gradlew.bat	ram	6 days ago
settings.gradle	ram	6 days ago

Help people interested in this repository understand your project by adding a README.

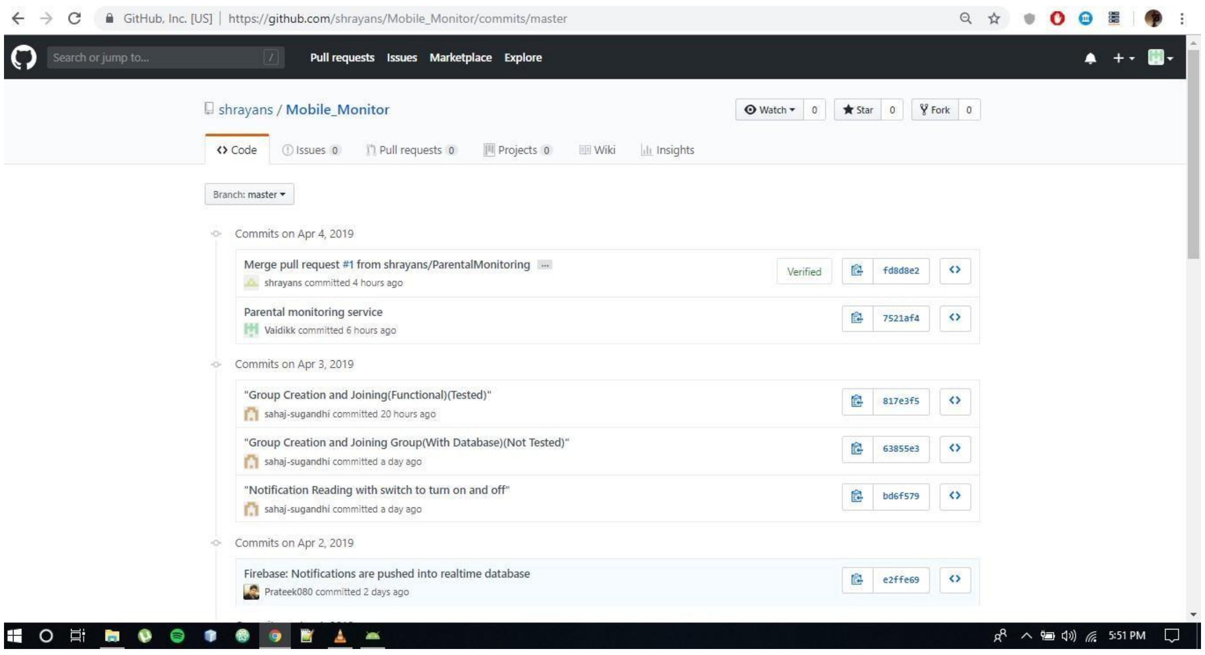
Figure 5.5 GitHub Main, all of our source code is present at this github repository, it helped us in version controlling.


https://console.firebase.google.com/u/0/project/mobilemonitor-4d1b3/database/mobilemonitor-4d1b3/data
MobileMonitor
Database
Go to docs
Data
Rules
Backups
Usage

https://mobilemonitor-4d1b3.firebaseio.com/

```
mobilemonitor-4d1b3
  Groups
    group-e5PGDbiU0TaYOKeD4xwy9V8eD0K2
      1: "e5PGDbiU0TaYOKeD4xwy9V8eD0K
      2: "rDpc8xiQphOw9ZPzt5Kj7ZWf4Py
  Notification
    9kAm7x2RcWSZjxcRJlT8p1cxla13
      2019-04-04 11:13:17
      2019-04-04 11:13:20
    P4wi111Q3lZMy3Qti4SRtXRgL3y2
      2019-04-04 13:40:29
      2019-04-04 13:40:30
    e5PGDbiU0TaYOKeD4xwy9V8eD0K2
      2019-04-03 11:50:26
      2019-04-03 11:50:27
      2019-04-03 11:50:33
```

Figure 5.6 Firebase Realtime database, this is the realtime database structure of our system on firebase



shrayans / Mobile_Monitor

Commits on Apr 4, 2019

Merge pull request #1 from shrayans/ParentalMonitoring
shrayans committed 4 hours ago — Verified — fd8d8e2

Parental monitoring service
Vaidikk committed 6 hours ago — 7521af4

Commits on Apr 3, 2019

"Group Creation and Joining(Functional)(Tested)"
sahaj-sugandhi committed 20 hours ago — 817e3f5

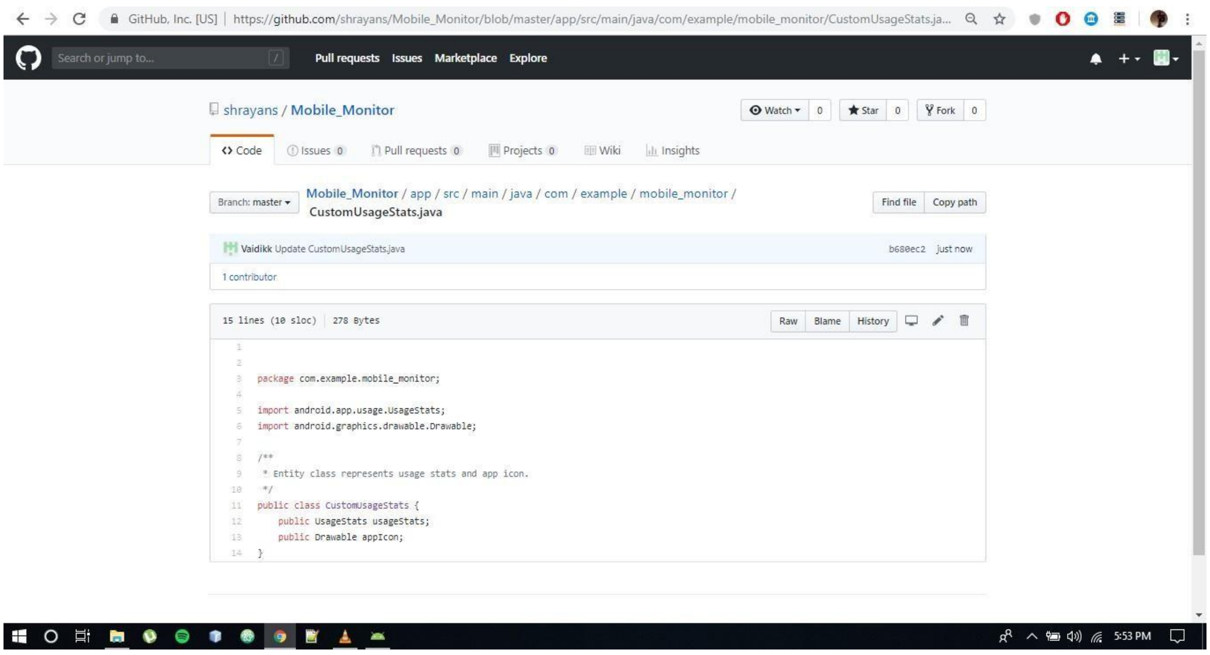
"Group Creation and Joining Group(With Database)(Not Tested)"
sahaj-sugandhi committed a day ago — 63855e3

"Notification Reading with switch to turn on and off"
sahaj-sugandhi committed a day ago — bd6f579

Commits on Apr 2, 2019

Firebase: Notifications are pushed into realtime database
Prateek080 committed 2 days ago — e2ffe69

Figure 5.7GitHub Commits, these are the commits performed by all the team members in there code for the application.



Mobile_Monitor / app / src / main / java / com / example / mobile_monitor / CustomUsageStats.java

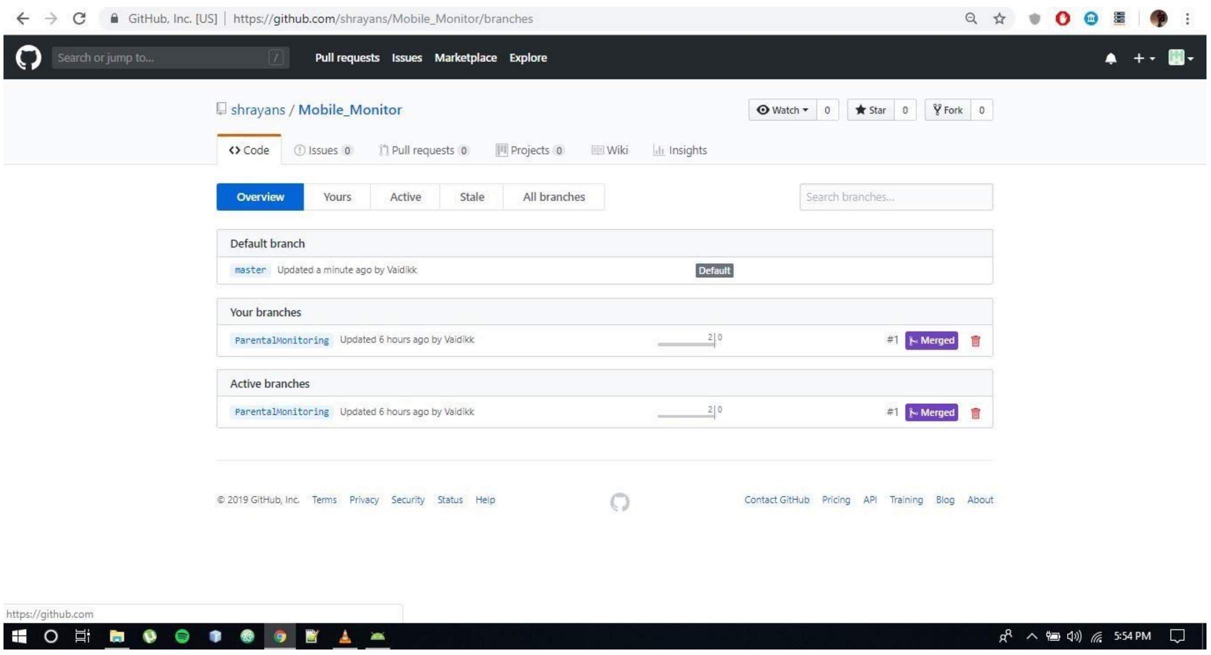
Vaidikk Update CustomUsageStats.java b680ec2 just now

1 contributor

15 lines (10 sloc) | 278 Bytes

```java


package com.example.mobile_monitor;

import android.app.usage.UsageStats;
import android.graphics.drawable.Drawable;

/**
 * Entity class represents usage stats and app icon.
 */
public class CustomUsageStats {
    public UsageStats usageStats;
    public Drawable appIcon;
}
```

Figure 5.8 Editor, this is used by the team members to go through the versions and the changes present in the versions.



Default branch

master Updated a minute ago by Vaidikk Default

Your branches

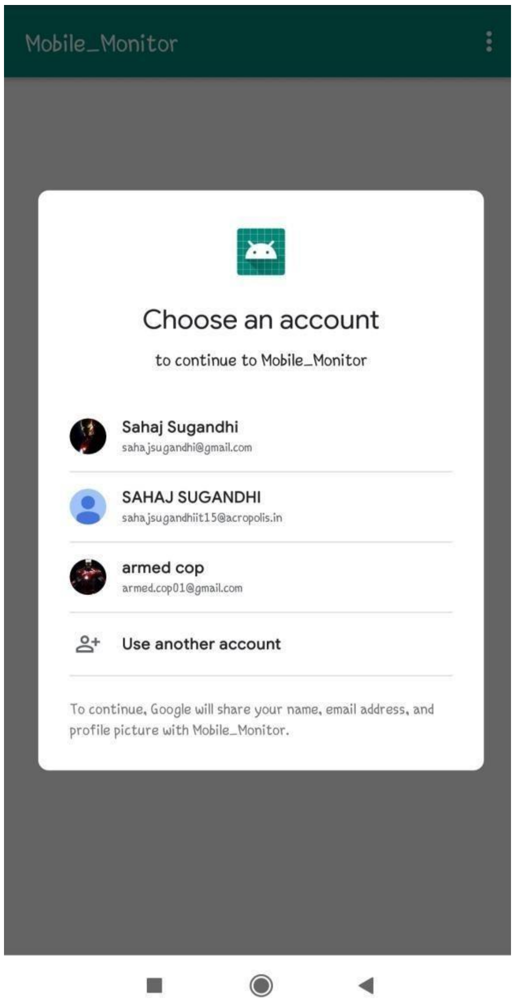
ParentalMonitoring Updated 6 hours ago by Vaidikk 2 | 0 #1 Merged

Active branches

ParentalMonitoring Updated 6 hours ago by Vaidikk 2 | 0 #1 Merged

Figure 5.9 GitHub Branches, different branches are used for the development of different modules of the application.

App



Mobile_Monitor

Choose an account

to continue to Mobile_Monitor

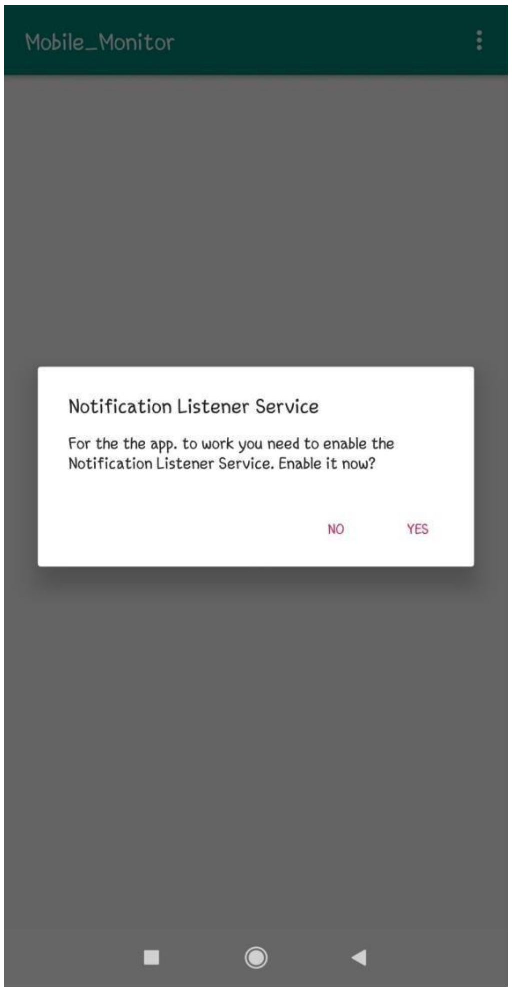
Sahaj Sugandhi
sahajsugandhi@gmail.com

SAHAJ SUGANDHI
sahajsugandhiit15@acropolis.in

armed cop
armed.cop01@gmail.com

Use another account

To continue, Google will share your name, email address, and profile picture with Mobile_Monitor.



Mobile_Monitor

Notification Listener Service

For the the app. to work you need to enable the Notification Listener Service. Enable it now?

Fig 5.10 Fig 5.11

Figure 5.10 shows signup screen of the application, how it will prompt user for signup and login.

Figure 5.11shows how the application is asking for the required permissions. It asks for two permissions which are android special permissions for Notification Reading and Usage Access.

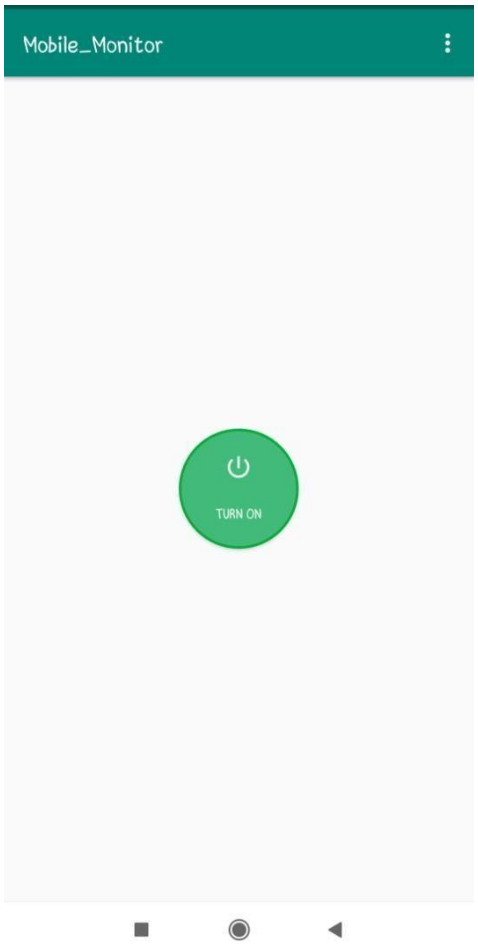


Fig 5.12

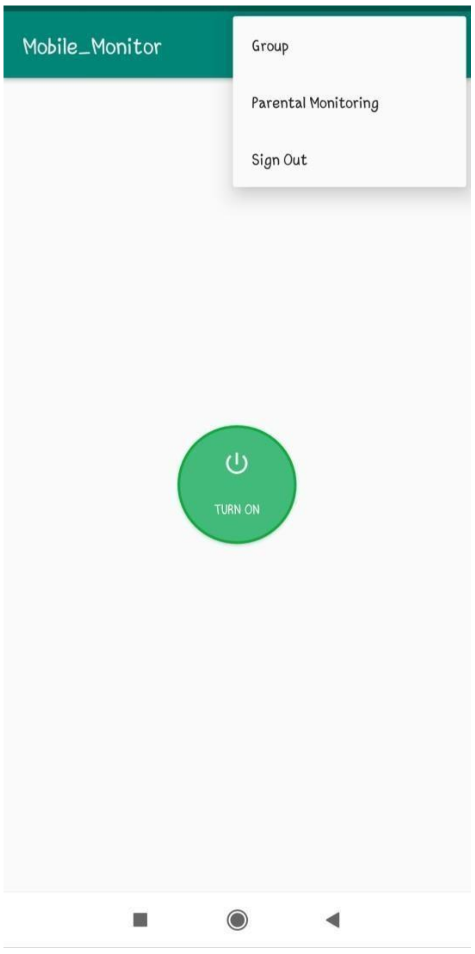
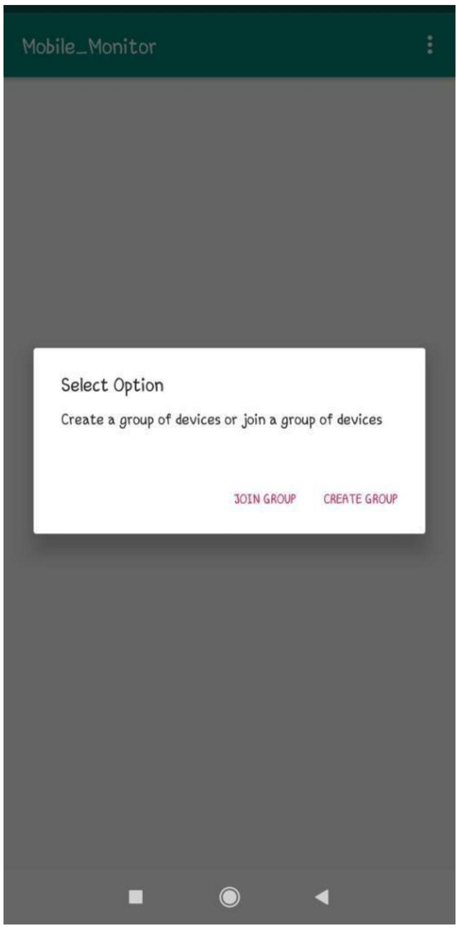


Fig 5.13

Fig 5.12 and 5.13 are of the home screen of the application, it shows what UI will be visible to the user after logging in and what options user will get on home screen.



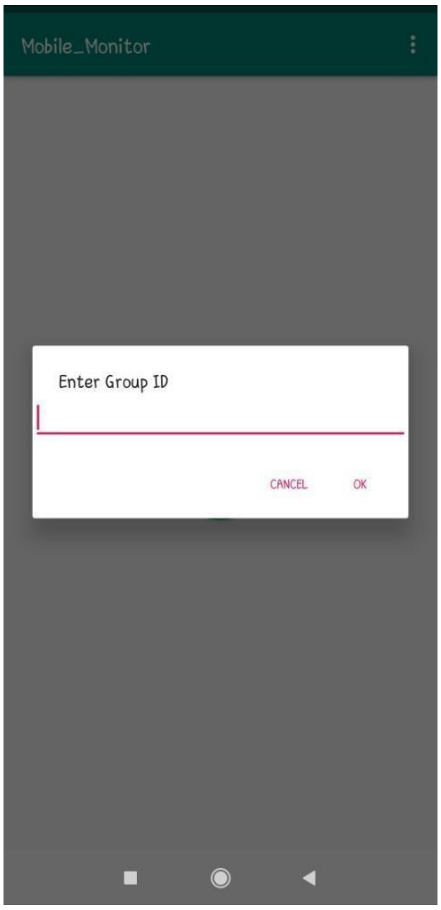
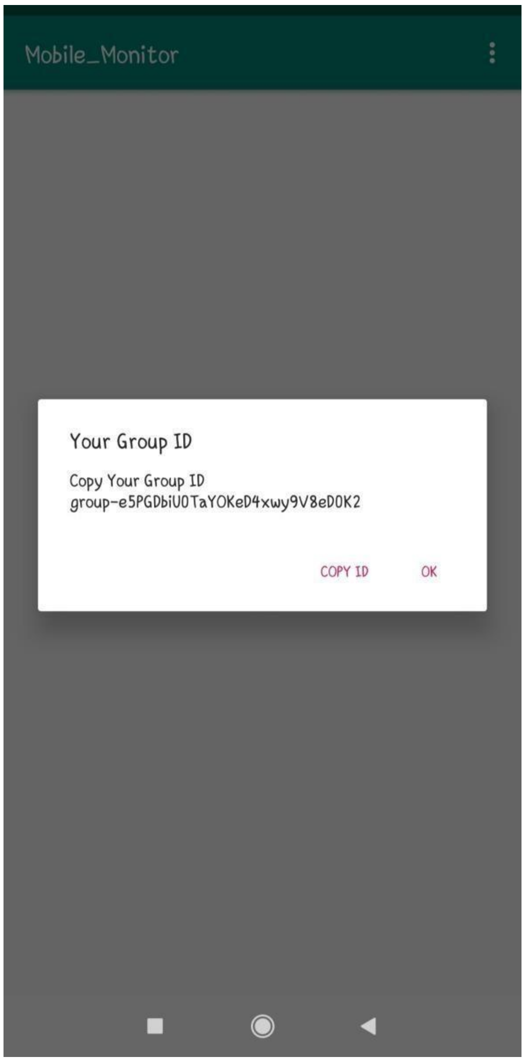


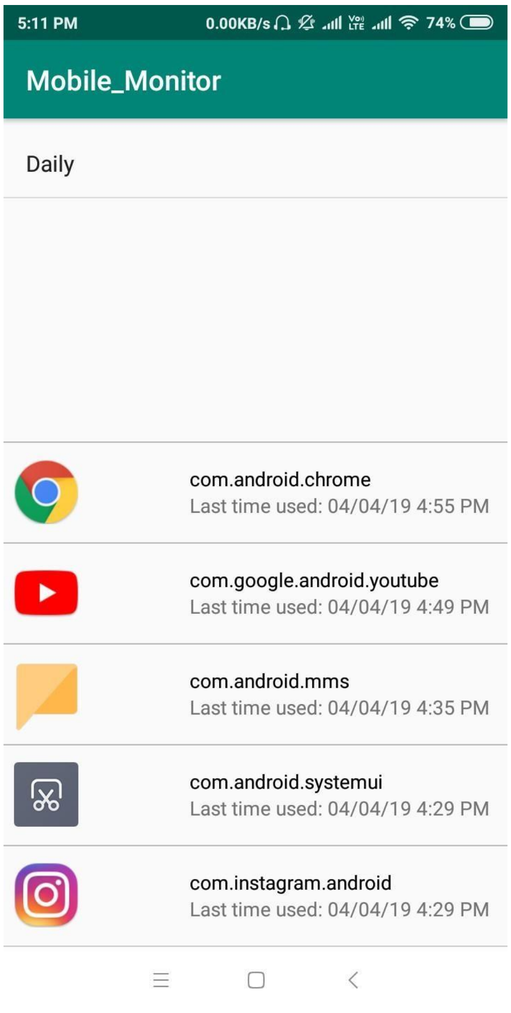
Fig 5.14 Fig 5.15

Figure 5.14 and 5.15 are the pictures of the application show how user will get the options to use the feature of group creation and how user will create and join group


Mobile_Monitor

Your Group ID

Copy Your Group ID
group-e5PGDbiU0TaYOKeD4xwy9V8eD0K2



Mobile_Monitor

Daily

com.android.chrome
Last time used: 04/04/19 4:55 PM

com.google.android.youtube
Last time used: 04/04/19 4:49 PM

com.android.mms
Last time used: 04/04/19 4:35 PM

com.android.systemui
Last time used: 04/04/19 4:29 PM

com.instagram.android
Last time used: 04/04/19 4:29 PM

Fig 5.16 Fig 5.17

Fig 5.16 is the picture of the application shows how the group is visible and accessible to the users.

Fig 5.17 is the picture of the application shows the feature of parental monitoring, how the group admin i.e. parent will monitor the child’s mobile.

4.5 TESTING

Testing is the process of evaluation of a system to detect differences between given input and expected output and also to assess the feature of the system. Testing assesses the quality of the product. It is a process that is done during the development process.

4.5.1 STRATEGY USED

Tests can be conducted based on two approaches:

- Functionality testing
- Implementation testing

The texting method used here is Black Box Testing. It is carried out to test functionality of the program. It is also called 'Behavioural' testing. The tester in this case, has a set of input values and respective desired results. On providing input, if the output matches with the desired results, the program is tested 'ok', and problematic otherwise.

Test Case Analysis

Test Case 1:

Test Case ID	TC001
Test Case Summary	To check authentication and validation
Test Procedure	Try inputting false credentials
Expected Outcome	Login screen shows error
Actual Result	Error message is displayed
Status	Pass

Description: The test case one describes the functionality of the login, signup and sign out. This is used to authenticate and validate the users. The application should display an error message if user enter the wrong credentials and if the credentials are correct then user should be redirected towards home activity. On entering wrong credentials, the application showed error message and on entering correct credentials, the application redirected user to the home screen.
So, the test case is passed.

Test Case 2:

Test Case ID	TC002
Test Case Summary	To check if firebase is getting notifications or not
Test Procedure	Turn on mobile monitoring and showing the data on web page
Expected Outcome	Will show the notification
Actual Result	All the notification will come on the web application
Status	Pass

Description: The test case 2 describes the functionality of notification monitoring, the application should read all the incoming notifications and should display it to the web interface to the user. On turning mobile monitoring on the application showed all the notifications of the android device to the web interface. So the test case is passed.

Chapter 5. Conclusion

5.1 CONCLUSION

This application is an all in one package. It will not only help the people to get their phone's notification on the laptop, but will also allow the parents to keep a log their child's activities.

This will make the lives of office going people a lot easier. As most of the companies these days don't allow mobile in their premises. This application will save them from missing out on any important information.

Also, this will help the parents to get the log of their child's activities and keep them in check.

Overall, this will prove out to be helpful for the working people and parents.

5.2 LIMITATIONS OF THE WORK

- No way of responding through laptop:

This application will only read the message from android device and show it on the web application. One cannot reply from this web application. For, replying the person must use his mobile phone only.

- Limited to android users:

As this application is still in its development phase and is limited to Android users only. There is a possibility of a cross platform application. Which will work on every mobile platform like iOS, Windows OS etc.

- No details about the duration of use:

This application will not give the amount of time a child has used the application. Rather, it will just tell us about the time at which it was opened.

RECOMMENDATIONS FOR FUTURE WORK

- With a high scope of further development this application can be integrated with some more functionalities like mobile tracking, giving alerts on the group
- Possibility of getting the time duration of usage of an application on child’s phone is also possible. Which will increase the control of the parents.
- A way of replying through the web application itself can also be introduced. It will increase the efficiency of the system.

REFERENCES

[1] https://www.google.co.in

[2] https://www.researchgate.net

[3] http://www.engpaper.com

[4] http://www.iosrjournals.org

[5] http://www.mwm.im

[6] https://arxiv.org

[7] https://scholar.google.co.in

[8] https://link.springer.com

[9] https://ieeexplore.ieee.org

[10]https://pdfs.semanticscholar.org

[11]https://eprints.qut.edu.au

[12]http://www.engpaper.com/mobile-jammer.html

Appendix A: Source Code

1. Index.html at Firebase Hosting

```
<html>
<head>
<title>Mobile Monitor</title>
<style>
.hide{
display: none;
}
</style>
</head>
<body>
<h1 id="name">Welcome to Mobile Monitor</h1>
<br><br><br><br>
<h3 id="groupHead" class="hide">Your Group</h3>
<div id="group"></div>
<br><br>
<h3 id="notificationHead" class="hide">Your Notifications</h3>
<br>
<div id="notifications"></div>
<br><br>
<h3 id="parentalHead" class="hide">Parental Monitoring</h3>
<br>
<div id="parental"></div>
<button id="signOut" onclick="signOut()" class="hide">Sign Out</button>
<script src="https://www.gstatic.com/firebasejs/5.9.2/firebase.js"></script>
<script>
  // Initialize Firebase
  var config = {
    apiKey: "AIzaSyCo39ikysdqB0HX7X-AohKBPnFTu4GCl5k",
    authDomain: "mobilemonitor-4d1b3.firebaseapp.com",
    databaseURL: "https://mobilemonitor-4d1b3.firebaseio.com",
    projectId: "mobilemonitor-4d1b3",
    storageBucket: "mobilemonitor-4d1b3.appspot.com",
    messagingSenderId: "418797324871"
  };
  firebase.initializeApp(config);

function login(){
    function newLoginHappened(user){
                    if(user) {
                    //user is signedin
                    document.getElementById("signOut").classList.remove("hide");
                    document.getElementById("notificationHead").classList.remove("hide");
                    document.getElementById("groupHead").classList.remove("hide");
                    document.getElementById("parentalHead").classList.remove("hide");
                    app(user);
                    getGroupDetails(user);
                    getNotificationDetails(user);
                    //getParentalDetails(user);
                    }
                    else{
                    var provider=new firebase.auth.GoogleAuthProvider();
                    firebase.auth().signInWithRedirect(provider);
                    }
    }
    firebase.auth().onAuthStateChanged(newLoginHappened);
}

function getNotificationDetails(user){
    var notificationRefOfUser=firebase.database().ref().child("Notification").child(user.uid);
    notificationRefOfUser.on("value", function(snapshot) {
                    var notificationTableHeader='<table style="width:100%"
border=1><tr><th>Time</th><th>Application</th><th>Title</th><th>Text</th><th>Extra Message</th></tr>';
                    var notificationTable='';
                    snapshot.forEach(function(child) {
                                    var notification=child.val();
                                    var thisLine='<tr
align="center"><td>'+notification.dateAndTime+'</td><td>'+notification.packageName+'</td><td>'+notification.title+'</td><td>
'+notification.text+'</td><td>'+notification.messageLines+'</td></tr>';
                                    notificationTable=thisLine+notificationTable;
                    });
                    var table=notificationTableHeader+notificationTable+"</table><br>";
                    document.getElementById("notifications").innerHTML=table;
    });
}
```

```
function getParentalDetails(user, groupMembersList){
   var groupTable="";
   var defaultLine="<p>You wont get any details if you are not an admin or if you have not enabled parent monitoring</p><br>";
   var parentalRef=firebase.database().ref().child("Parental Monitoring");
   parentalRef.on('value', function(snapshot) {
   var i;
                 for(i=1;i<groupMembersList.length;i++){
                                if(snapshot.hasChild(groupMembersList[i])) {
                                               groupTable=defaultLine+'<table style="width:100%" border=1><caption>For DeviceID:
'+groupMembersList[i]+'</caption><tr><th>Application Name</th><th>Time last used</th></tr>';
                                               snapshot.forEach(function(child) {
                                                              if(child.key===groupMembersList[i]){
                                                                             var activities=child.val();
                                                                             var j;
                                                                             for(j=0;j<activities.length;j++){
                                                                                            var activity=activities[j];

   groupTable=groupTable+'<tr><td>'+activity.appPackage+'</td><td>'+activity.time+'</td></tr>';
                                                                             }
                                                              }
                                               });
                                               groupTable=groupTable+"</table><br>";
                                }
                                document.getElementById("parental").innerHTML=groupTable;
                 }
   });
}

function getGroupDetails(user){
   var groupRef=firebase.database().ref().child("Groups");
   var groupId="group-"+user.uid;
   groupRef.on('value', function(snapshot) {
                 var groupTable="";
                 if(!snapshot.hasChild(groupId)) {
                                groupTable=groupTable+"<p>You don't have Admin permissions of group, or you are not part of any
group</p>";
                 }
                 else {
                                groupTable='<table style="width:100%" border=1><caption>GroupID:
'+groupId+'</caption><tr><th>Member No.</th><th>Member Uid</th></tr>';
                                snapshot.forEach(function(child) {
                                               if(child.key===groupId){
                                                              var groupOfUser=child.val();
                                                              getParentalDetails(user, groupOfUser);
                                                              var i;
                                                              for(i=1;i<groupOfUser.length;i++){
                                                                             groupTable=groupTable+'<tr
align="center"><td>'+i+'</td><td>'+groupOfUser[i]+'</td></tr>';
                                                              }
                                               }
                                });
                                groupTable=groupTable+"</table><br>";
                 }
                 document.getElementById("group").innerHTML=groupTable;
   });
}

function app(user){
   //user.displayName
   //user.photoURL
   //user.email
   //user.uid
   alert("Welcome "+user.displayName+"!");
   document.getElementById("name").innerHTML="Welcome "+user.displayName+"!("+user.uid+")";
}

function signOut(){
   firebase.auth().signOut().then(function() {
                 alert('Signed Out');
                 document.getElementById("signOut").classList.add("hide");
                 document.getElementById("notificationHead").classList.add("hide");
                 document.getElementById("groupHead").classList.add("hide");
                 }, function(error) {
                 alert('Sign Out Error', error);
   });
}

window.onload=login;

</script>
</body>
```

```
</html>
```

2. AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.mobile_monitor">

    <uses-permission
        android:name="android.permission.PACKAGE_USAGE_STATS"
        tools:ignore="ProtectedPermissions" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".ParentalMonitoring"></activity>
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service android:label="@string/service_label" android:name=".NotificationListenerExampleService"
            android:permission="android.permission.BIND_NOTIFICATION_LISTENER_SERVICE">
            <intent-filter>
                <action android:name="android.service.notification.NotificationListenerService"/>
            </intent-filter>
        </service>
    </application>
</manifest>
```

3. activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/mobileMonitoringView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignWithParentIfMissing="false"
        android:layout_above="@+id/on_off_switch"
        android:layout_centerHorizontal="true"
        android:layout_marginStart="25dp"
        android:layout_marginTop="25dp"
        android:layout_marginEnd="25dp"
        android:layout_marginBottom="25dp"
        android:text="Mobile Monitoring is Deactivated"
        android:textColor="@android:color/black" />

    <Button
        android:id="@+id/on_off_switch"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:background="@drawable/circle_green"
        android:drawableTop="@drawable/on_off_switch"
        android:paddingTop="20dp"
        android:textColor="#fff"
        android:layout_centerInParent="true"
        android:text="Turn On"
        android:onClick="switchClicked"/>

</RelativeLayout>
```

4. MainActivity.java

```
package com.example.mobile_monitor;


import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.ClipData;
import android.content.ClipboardManager;
import android.content.ComponentName;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.SharedPreferences;
import android.provider.Settings;
import android.support.annotation.NonNull;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.InputType;
import android.text.TextUtils;
import android.util.Log;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.example.mobile_monitor.Data.NotificationKeeper;
import com.firebase.ui.auth.AuthUI;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.text.SimpleDateFormat;
import java.util.Arrays;
import java.util.Date;

public class MainActivity extends AppCompatActivity {

    public static final int RC_SIGN_IN = 1;

    FirebaseAuth mFirebaseAuth;
    private FirebaseAuth.AuthStateListener mAuthStateListener;

    //FirebaseDatabase database = FirebaseDatabase.getInstance();
    DatabaseReference mDatabase = FirebaseDatabase.getInstance().getReference();
    FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
    String uid;

    Button on_off_switch;
    TextView status;

    private static final String ENABLED_NOTIFICATION_LISTENERS = "enabled_notification_listeners";
    private static final String ACTION_NOTIFICATION_LISTENER_SETTINGS =
"android.settings.ACTION_NOTIFICATION_LISTENER_SETTINGS";
    private AlertDialog enableNotificationListenerAlertDialog;

    private NotificationReceiver nReceiver;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
    mFirebaseAuth=FirebaseAuth.getInstance();
    mAuthStateListener=new FirebaseAuth.AuthStateListener() {
        @Override
        public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {
            FirebaseUser user=firebaseAuth.getCurrentUser();
            if(user!=null){
                setContentView(R.layout.activity_main);
                status=findViewById(R.id.mobileMonitoringView);
                Toast.makeText(MainActivity.this,"WELCOME! You are signed in",Toast.LENGTH_LONG).show();
                MainActivity.this.user = user;

                on_off_switch=findViewById(R.id.on_off_switch);
                SharedPreferences pref = getApplicationContext().getSharedPreferences("MyPref", 0); // 0 - for private mode
                String operation=pref.getString("ReadNotifications",null);
                if(operation!=null&&operation.equals("Activated")) {
                    on_off_switch.setText("Turn Off");
                    on_off_switch.setBackgroundResource(R.drawable.circle_red);
                    on_off_switch.setPadding(0,20,0,0);
                }

                // If the user did not turn the notification listener service on we prompt him to do so
                if(!isNotificationServiceEnabled()){
                    enableNotificationListenerAlertDialog = buildNotificationServiceAlertDialog();
                    enableNotificationListenerAlertDialog.show();
                }

                nReceiver = new NotificationReceiver();
                IntentFilter filter = new IntentFilter();
                filter.addAction("com.example.mobile_monitor.NOTIFICATION_LISTENER_EXAMPLE");
                registerReceiver(nReceiver,filter);
            }
            else{
                startActivityForResult(
                        AuthUI.getInstance()
                                .createSignInIntentBuilder()
                                .setIsSmartLockEnabled(false)
                                .setAvailableProviders(Arrays.asList(
                                        new AuthUI.IdpConfig.GoogleBuilder().build()))
                                .build(),
                        RC_SIGN_IN);
            }
        }
    };
}

@Override
protected void onPause() {
    super.onPause();
    mFirebaseAuth.removeAuthStateListener(mAuthStateListener);
}

@Override
protected void onResume() {
    super.onResume();
    mFirebaseAuth.addAuthStateListener(mAuthStateListener);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater menuInflater=getMenuInflater();
    menuInflater.inflate(R.menu.menu,menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch(item.getItemId()){
        case R.id.parental_monitoring:
            Intent intent = new Intent(this, ParentalMonitoring.class);
            startActivity(intent);
            return true;
        case R.id.group_menu:
            SharedPreferences pref = getApplicationContext().getSharedPreferences("MyPref", 0); // 0 - for private mode
            String operation=pref.getString("GroupID",null);
            if(operation!=null&&!operation.equals("")) {
                showGroupID(this,operation);
            }
            else showDialogForGroups(this,"Select Option","Create a group of devices or join a group of devices");
            return true;
        case R.id.sign_out_menu:
            SharedPreferences prefSignOut = getApplicationContext().getSharedPreferences("MyPref", 0); // 0 - for private mode
            SharedPreferences.Editor editor = prefSignOut.edit();
            editor.putString("GroupID","");
            editor.putString("ReadNotifications", "");
            editor.commit();
            Toast.makeText(this,"You are Signed Out",Toast.LENGTH_LONG).show();
            AuthUI.getInstance().signOut(this);
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}

public void showGroupID(Activity activity, final String groupID) {
    final AlertDialog.Builder builder = new AlertDialog.Builder(activity);
    builder.setTitle("Your Group ID");
    builder.setMessage("Copy Your Group ID\n"+groupID);
    builder.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.cancel();
        }
    });
    builder.setNegativeButton("Copy ID", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            ClipboardManager clipboard = (ClipboardManager) getSystemService(Context.CLIPBOARD_SERVICE);
            ClipData clip = ClipData.newPlainText("ClipBoard Label", groupID);
            clipboard.setPrimaryClip(clip);
        }
    });
    builder.show();
}

public void showDialogForGroups(final Activity activity, String title, CharSequence message) {
    final AlertDialog.Builder builder = new AlertDialog.Builder(activity);
    if (title != null) builder.setTitle(title);
    builder.setMessage(message);
    builder.setPositiveButton("Create Group", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            mDatabase.child("Groups").child("group-"+user.getUid()).child("1").setValue(user.getUid());
            SharedPreferences pref = getApplicationContext().getSharedPreferences("MyPref", 0); // 0 - for private mode
            SharedPreferences.Editor editor = pref.edit();
            editor.putString("GroupID", "group-"+user.getUid());
            editor.commit();
            showGroupID(activity,"group-"+user.getUid());
            dialog.cancel();
        }
    });
    builder.setNegativeButton("Join Group", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            takeGroupID(activity);
            dialog.cancel();
        }
    });
    builder.show();
}

public void takeGroupID(final Activity activity){
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Enter Group ID");
    // Set up the input
```

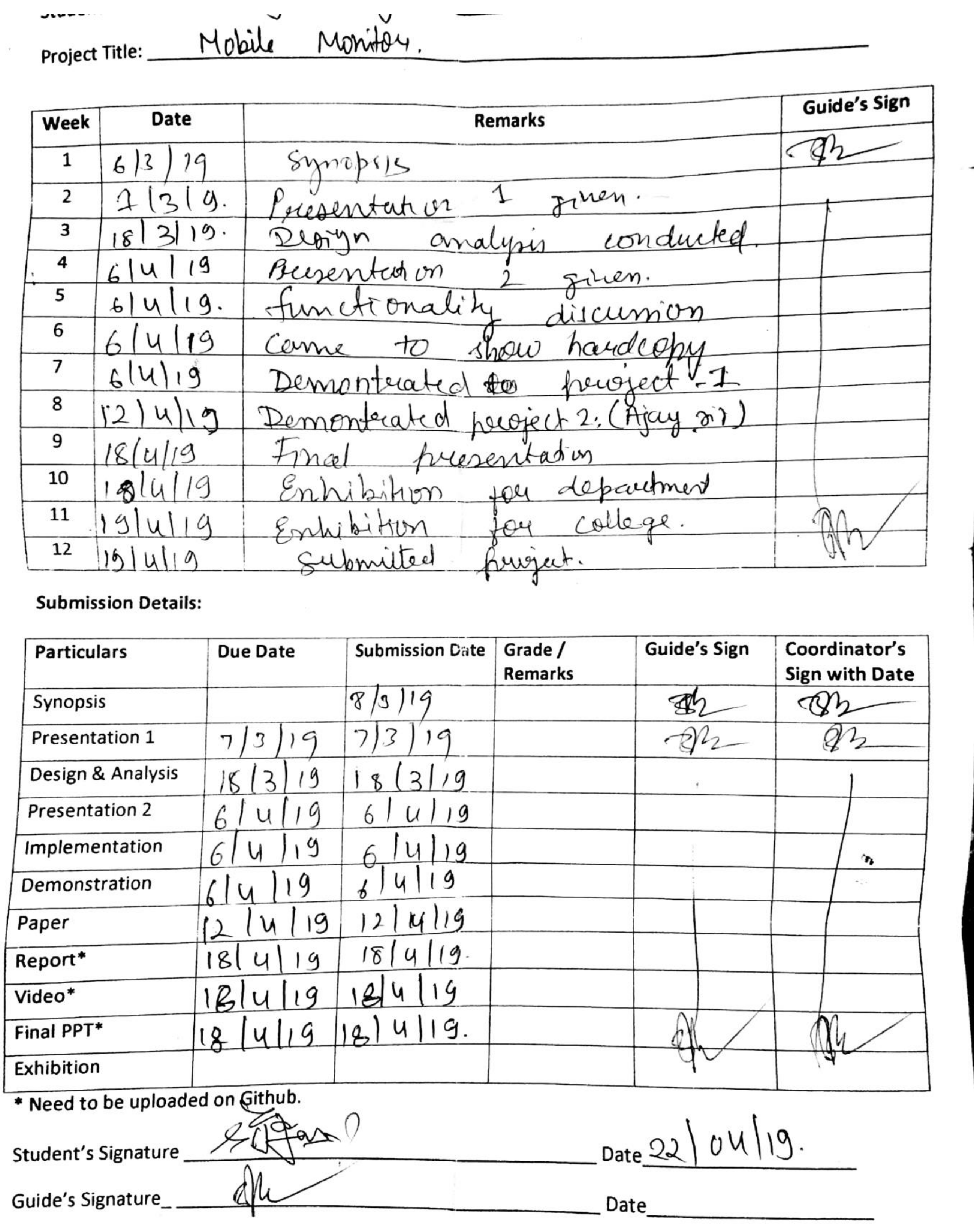
```
final EditText inputField = new EditText(this);
// Specify the type of input expected; this, for example, sets the input as a password, and will mask the text
inputField.setInputType(InputType.TYPE_CLASS_TEXT);
builder.setView(inputField);
// Set up the buttons
builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(final DialogInterface dialog, int which) {
        if(inputField.getText().toString().trim().equalsIgnoreCase("")) {
            Toast.makeText(activity, "This field can not be blank", Toast.LENGTH_SHORT).show();
        }
        else{
            final String inputID=inputField.getText().toString();
            if(!inputID.startsWith("group-")){
                Toast.makeText(activity, "Enter a valid group ID", Toast.LENGTH_SHORT).show();
            }
            else{
                final DatabaseReference rootRef = FirebaseDatabase.getInstance().getReference().child("Groups");
                rootRef.addListenerForSingleValueEvent(new ValueEventListener() {
                    @Override
                    public void onDataChange(DataSnapshot snapshot) {
                        if (snapshot.hasChild(inputID)) {
                            //add the current user(user.getUid()) to this node(inputID)
                            Log.e("MyLog " ,"Count: "+snapshot.getChildrenCount());
                            String oldValue="";
                            for (DataSnapshot postSnapshot: snapshot.getChildren()) {
                                String key=postSnapshot.getKey();
                                if(key.equals(inputID)) {
                                    //oldValue = postSnapshot.getValue(String.class);
                                    //Log.e("MyLog", "key/value: " + key + "/" + oldValue);
                                    DatabaseReference newRef=FirebaseDatabase.getInstance().getReference().child("Groups").child(inputID);
                                    newRef.addListenerForSingleValueEvent(new ValueEventListener() {
                                        @Override
                                        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                                            long count=dataSnapshot.getChildrenCount();
                                            count++;
                                            mDatabase.child("Groups").child(inputID).child(String.valueOf(count)).setValue(user.getUid());
                                        }
                                        @Override
                                        public void onCancelled(@NonNull DatabaseError databaseError) {
                                            Log.e("The read failed: " ,databaseError.getMessage());
                                        }
                                    });
                                }
                            }
                            SharedPreferences pref = getApplicationContext().getSharedPreferences("MyPref", 0); // 0 - for private mode
                            SharedPreferences.Editor editor = pref.edit();
                            editor.putString("GroupID",inputID);
```

```
                              editor.commit();
                              dialog.cancel();
                              showGroupID(activity,inputID);
                          }
                          else{
                              Toast.makeText(activity, "There is no group with this ID", Toast.LENGTH_SHORT).show();
                          }
                      }
                      @Override
                      public void onCancelled(@NonNull DatabaseError databaseError) {
                          Log.e("The read failed: " ,databaseError.getMessage());
                      }
                  });
              }
          }
      }
  });
  builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
      @Override
      public void onClick(DialogInterface dialog, int which) {
          dialog.cancel();
      }
  });
  builder.show();
}

@Override
protected void onDestroy() {
    super.onDestroy();
    unregisterReceiver(nReceiver);
}

/**
 * Is Notification Service Enabled.
 * Verifies if the notification listener service is enabled.
 * Got it from: https://github.com/kpbird/NotificationListenerService-Example/blob/master/NLSExample/src/main/java/com/kpbird/nlsexample/NLService.java
 * @return True if eanbled, false otherwise.
 */
private boolean isNotificationServiceEnabled(){
    String pkgName = getPackageName();
    final String flat = Settings.Secure.getString(getContentResolver(),ENABLED_NOTIFICATION_LISTENERS);
    if (!TextUtils.isEmpty(flat)) {
        final String[] names = flat.split(":");
        for (int i = 0; i < names.length; i++) {
            final ComponentName cn = ComponentName.unflattenFromString(names[i]);
            if (cn != null) {
                if (TextUtils.equals(pkgName, cn.getPackageName())) {
                    return true;
                }
            }
        }
    }
    return false;
}

/**
 * Build Notification Listener Alert Dialog.
 * Builds the alert dialog that pops up if the user has not turned
 * the Notification Listener Service on yet.
 * @return An alert dialog which leads to the notification enabling screen
 */
private AlertDialog buildNotificationServiceAlertDialog(){
    AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);
    alertDialogBuilder.setTitle(R.string.notification_listener_service);
    alertDialogBuilder.setMessage(R.string.notification_listener_service_explanation);
    alertDialogBuilder.setPositiveButton(R.string.yes,
            new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
                    startActivity(new Intent(ACTION_NOTIFICATION_LISTENER_SETTINGS));
                }
            });
    alertDialogBuilder.setNegativeButton(R.string.no,
            new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
                    // If you choose to not enable the notification listener
                    // the app. will not work as expected
                }
            });
    return(alertDialogBuilder.create());
}

public void switchClicked(View view) {
    SharedPreferences pref = getApplicationContext().getSharedPreferences("MyPref", 0); // 0 - for private mode
    SharedPreferences.Editor editor = pref.edit();
    if(on_off_switch.getText().toString().equals("Turn On")) {
        on_off_switch.setText("Turn Off");
        status.setText("Mobile Monitoring is Activated");
        Toast.makeText(this, "Mobile Monitoring is ON now", Toast.LENGTH_SHORT).show();
        on_off_switch.setBackgroundResource(R.drawable.circle_red);
        on_off_switch.setPadding(0,20,0,0);
        editor.putString("ReadNotifications", "Activated");
        editor.commit();
    }
    else{
        on_off_switch.setText("Turn On");
```

```
            status.setText("Mobile Monitoring is Deactivated");
            Toast.makeText(this, "Mobile Monitoring is OFF now", Toast.LENGTH_SHORT).show();
            on_off_switch.setBackgroundResource(R.drawable.circle_green);
            on_off_switch.setPadding(0,20,0,0);
            editor.putString("ReadNotifications", "Deactivated");
            editor.commit();
        }
    }

    class NotificationReceiver extends BroadcastReceiver {

        @Override
        public void onReceive(Context context, Intent intent) {
            NotificationKeeper notificationKeeper = (NotificationKeeper) intent.getSerializableExtra("notificationReceived");
            Log.i("messageInMain","package: "+notificationKeeper.packageName);
            Log.i("messageInMain","title: "+notificationKeeper.title);
            Log.i("messageInMain","text: "+notificationKeeper.text);
            for(int i=1;i<=notificationKeeper.messageLines.size();i++){
                Log.i("messageInMain",i+". MessaageLine: "+notificationKeeper.messageLines.get(i-1));
            }
            //Object(notificationKeeper) is to be uploaded to fire-base

            SharedPreferences pref = getApplicationContext().getSharedPreferences("MyPref", 0); // 0 - for private mode
            String operation=pref.getString("ReadNotifications",null);
            if(operation!=null&&operation.equals("Activated")) {
                //String currentDateTime;

                SimpleDateFormat sdf1 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
                notificationKeeper.dateAndTime = sdf1.format(new Date());

                mDatabase.child("Notification").child(user.getUid()).push().setValue(notificationKeeper);
            }
        }
    }
}
```

Appendix B: Guide Interaction Sheet

This is the guide interaction sheet, which describes on what date we interacted with our project guide about the project and we discussed the mentioned topic on the mentioned date. Also, this sheet describes the submission details, which shows on what date we submitted which document.



Project Title: Mobile Monitor.

Week	Date	Remarks	Guide's Sign
1	6/3/19	Synopsis	[illegible]
2	7/3/19.	Presentation 1 given.	
3	18/3/19.	Design analysis conducted.	
4	6/4/19	Presentation 2 given.	
5	6/4/19.	functionality discussion	
6	6/4/19	Come to show hardcopy	
7	6/4/19	Demonstrated ~~to~~ project -1	
8	12/4/19	Demonstrated project 2. (Ajay sir)	
9	18/4/19	Final presentation	
10	18/4/19	Exhibition for department	
11	19/4/19	Exhibition for college.	[illegible]
12	19/4/19	Submitted project.	

Submission Details:

Particulars	Due Date	Submission Date	Grade / Remarks	Guide's Sign	Coordinator's Sign with Date
Synopsis		8/3/19		[illegible]	[illegible]
Presentation 1	7/3/19	7/3/19		[illegible]	[illegible]
Design & Analysis	18/3/19	18/3/19			
Presentation 2	6/4/19	6/4/19			
Implementation	6/4/19	6/4/19			
Demonstration	6/4/19	6/4/19			
Paper	12/4/19	12/4/19			
Report*	18/4/19	18/4/19.			
Video*	18/4/19	18/4/19			
Final PPT*	18/4/19	18/4/19.		[illegible]	[illegible]
Exhibition					

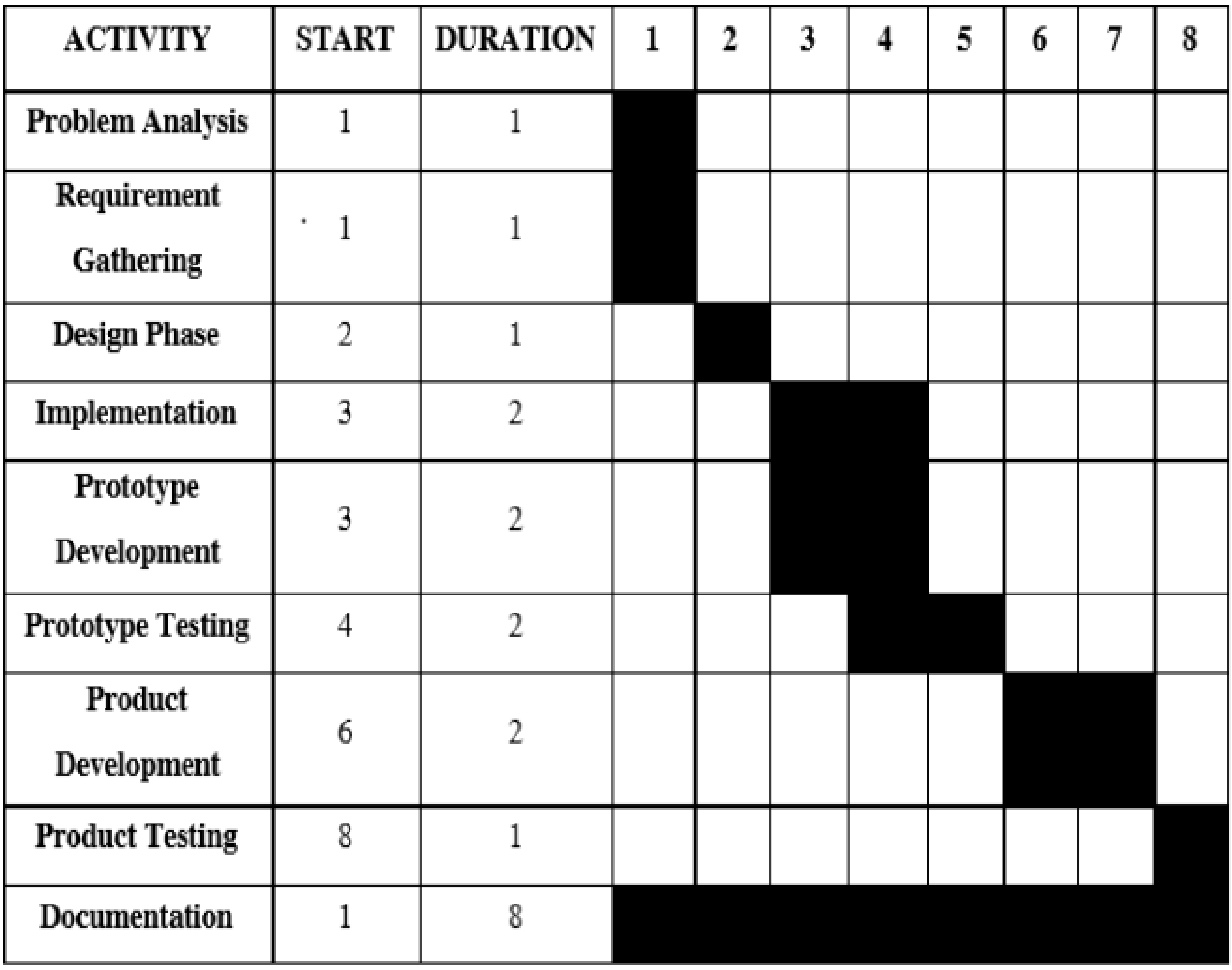
* Need to be uploaded on Github.

Student's Signature [illegible] Date 22/04/19.

Guide's Signature [illegible] Date

Appendix C: Gantt Chart

The below chart describes the project plan, from the starting of the month January i.e. from the first week of the January and here each block represents a week.



ACTIVITY	START	DURATION	1	2	3	4	5	6	7	8
Problem Analysis	1	1	■							
Requirement Gathering	1	1	■							
Design Phase	2	1		■						
Implementation	3	2			■	■				
Prototype Development	3	2			■	■				
Prototype Testing	4	2				■	■			
Product Development	6	2						■	■	
Product Testing	8	1								■
Documentation	1	8	■	■	■	■	■	■	■	■

Gantt Chart(Project Planning)