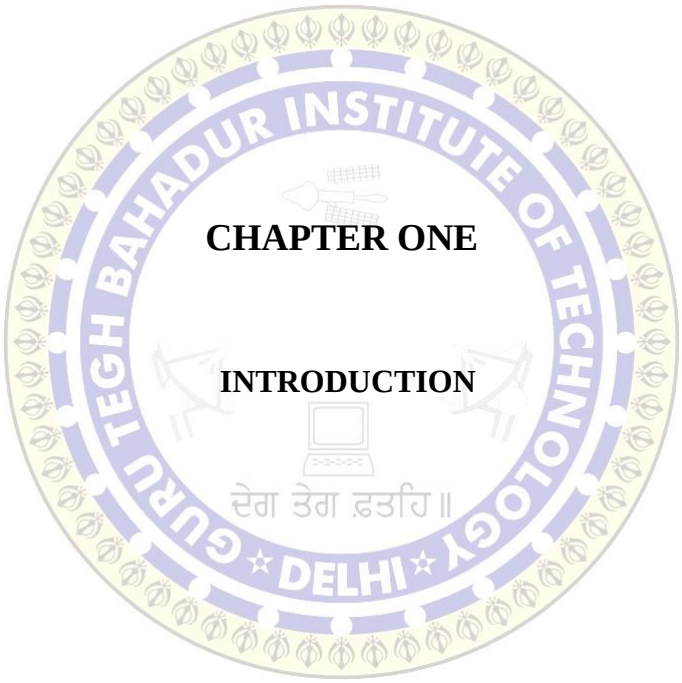


CHAPTER ONE

INTRODUCTION

1. INTRODUCTION

Professor's e-Assistant project is based on IoT Technology and Android. In this project we are using Alexa which will act as an assistant to Professors and will help them in their daily chores. Alexa is an intelligent personal assistant developed by Amazon, it is capable of voice interaction and taking voice commands. Alexa is programmable device which can also be connected to an Android app. Nowadays Alexa is widely used in Home Automation and personal assistance which will fulfill desired tasks as you say "Alexa". In our project this e-Assistant will be connected to an Android app where users can actually see and manage schedules, records and lists.

1.1 Purpose

The idea of making this virtual assistant came into our mind when we saw how much hectic the schedule of teachers in our college is. Features like making lists removes the unnecessary physical work they do while preparing lists such students marks list, attendance list etc. The feature of attendance also focuses to prevent the chaos that students make while teacher takes attendance. Since, Alexa can listen to a person at a time, this will avoid students from making noise and they will patiently listen to their roll calls. Other features will also benefit teachers by assisting them in their routines. They would be able to manage the records of the students of separate classes very easily and hassle-freely.

1.2 Processing

Alexa which responds to when we speak, calls the lambda function through the intents. The lambda function causes the changes in the Firebase database and the changes occurred are reflected to both Alexa and Android Application as these are connected to a realtime firebase database.

1.3 Features

This project aims at developing an e-assistant with the help of Alexa and supported by an Android App. It will assist the professors with following prominent features:

- **Alexa Tell Me My Schedule**: This feature will invoke a task to e-Assistant which will look up to Professor's schedule and call out timings of the class, subject and class number. This feature will automatically remind Professors about their class 10 minutes prior to the lecture.

  Why?

  As we all know how we people have habit of forgetting petty things that are most important sometimes. Similarly, a professor or a teacher could also forget their schedules and tasks.

  How?

  Professor will preset the schedule in the android app and alexa will monitor all those periods and labs and inform professor 10 minutes prior to the class. Alexa will get schedule data from firebase which a professor can edit according to his/her needs.

- **Alexa ask Teacher Helper to Start Attendance**: Alexa will take students attendance and attendance list will be updated on Android App accordingly.

  Why?

  Since we are familiar with the scenarios in which teachers have to handle so much chaos by the students in order to take the attendance completely, hellping professors and teachers in taking attendance can save a lot of energy.

How?

In this feature Professor can add new students to the pre-existing list of students, which has red/green indicator in front of their names.

Red means absent and green means present, all a teacher has to do is to ask alexa to start attendance and the list will be updated realtime in Android app too along with list in firebase.

- **Alexa Make A List:** Using this feature a professor can make any list with desirable column fields just by calling out voice command followed by data of the list. List will be visible on the app in real time. For example making list of Practical marks of students, user can also share this list via e-mail.

Why?

Making a list is a very tedious task when it comes to going through all the Records. Other than making, editing the lists is also requires a great deal of work.

How?

When a professor wants to create a list all they have to do ask Alexa to start creating a list and then decide the appropriate Title and the list will be updated real time to Android App and can be accessible from any remote location.

- **Alexa Tell Me Announcements:** This feature of Android App will make sharing vital information with professors really easy. All HoDs along with Director of college can make announcements, which will be received as push notification as well as Alexa will notify professors.

Why?

Announcements are the important messages that Head of Department or Director want to share with every professor.

Since conveying these announcements could itself be a tough task, therefore, our application which is connected among teachers will help regulate the necessary messages more effectively and efficiently.

How?

This can be done by triggering Alexa and ask it to tell the today's announcements. Once Alexa listens to that, it will respond with today's announcements that will contain the name of the person making the announcement, arrival time of the announcement and the message in it.

- **Attendance Report:** This feature of Android App will show the complete report of attendance of each class along with defaulters marked with red colour and professor could send warning email to all defaulters with one click.

  Why?

  As we know that attendance for a student is mandatory for him or her to learn in the classroom. Also institutes have a certain criteria related to attendance. Thus, this feature helps to regulate attendance of students.

  How?

  This feature is available via android application where every professor gets an option to save the attendance of the current class according to the present date. This also helps to analyse the defaulters by checking the previous references to the attendance and then if the attendance is below 70 precent, then an email can be sent to the defaulters regarding their attendance.

1.4 Uses

This Application can be used in various schools, colleges and institutes and also in the universities. Professor's E-Assistant focuses on providing best facilities to its users in the area of teaching.

1.5 Algorithm

In this project, a smart agent Alexa Dot is used which uses NLP to check what intent matches with the voice input. The matched intent is then called and works in correspondence to firbase. The changes in firebase as also reflected in the Android Application as the former is connected to the latter.

CHAPTER TWO

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

TABLE OF CONTENTS

Table of Contents ……. 8

1. Introduction ……. 8

1.1 Purpose ……..... 9

1.2 Document Conventions ..…… 9

1.3 Intended Audience and Reading Suggestions ……..... 9

1.4 Project Scope ……..……..... 9

2. Overall Description ..…… 9

2.1 Product Perspective ..…… 9

2.2 Product Features ……. 10

2.3 Operating Environment ……... 10

2.4 User Documentation ..……... 11

3. System Features ……..…….. 11

3.1 Alexa Tell Me My Schedule ……... 11

3.2 Alexa Take Attendance ……... 11

3.3 Alexa Make A List …….. 11

3.4 Attendance Report ……... 12

3.5 Announcements ……... 12

4. Other Non- Functional Requirements ……..……... 12

4.1 Understandability …… 12

4.2 Completeness ……. 12

4.3 Conciseness, Consistency and Efficiency ……... 12

4.4 Portability …… 12

4.5 Maintainability …… 13

4.6 Reliability …… 13

1. Introduction

1.1 Purpose

This document specifies the software requirements for the Professor's E-Assistant.

1.2 Document Conventions

The Professor's E-Assistant application is frequently referred to as "The Assistant".

1.3 Intended Audience and Reading Suggestions

This document describes the project scope for software developers.Readers should be familiar with android applications and alexa.

1.4 Project Scope

Professor's e-Assistant project is based on Iot technology. In this project we are using alexa which will act as an assistant to Professors and will help them in their daily chores. This e-Assistant will be connected to an android app where users can actually see and manage schedules,lists and records.

Alexa takes Professor's voice commands as input to do their required task such as taking attendance, telling the schedule , to make a list,.make an announcement etc.. The android app shows the updated status.

2. Overall Description

2.1 Product Perspective

The product aims at minimising the work of the professors in the best possible way. It helps the professors in their day to day work by performing the task designated to it.

The strength of the project should lie in the implementation and cohesion between it's individual parts. These parts work together to create a proven, high quality performance system.

2.2 Product Features

The features of alexa fall into four main categories namely attendance, list, Report and announcements. In this user gives commands to alexa to perform any of these four functions. The alexa uses the lambda function to update the status in the firebase which in turn updates it in the android app.

Alexa tell me my schedule feature will invoke a task to e-Assistant which will lookup to Professor's schedule and call out timings of the class, subject and class number. This feature will automatically remind Professors about their class 10 minutes to the lecture. Alexa take attendance will take students attendance. Attendance list will be updated on the app accordingly. Alexa make a list feature a professor can make any list with desirable column fields just by calling out voice command followed by data of the list. List will be visible on the app in realtime. For example making list of Practical marks of students, user can also share this list via e-mail.Attendance report feature of Android App will show the complete report of attendance of each class along with defaulters marked with red colour and professor could send warning email to all defaulters with one click. Announcements feature of Android App will make sharing vital information with professors really easy. All HoDs along with Director of college can make announcements, which will be received as push notification as well as alexa will notify professors.

2.3 Operating Environment

The system is to be developed using python 2.7,java and xml programming Languages. The hardware used is alexa and android phone and is supported on Windows 7, Windows 8 and Windows 10

operating systems. The tools used are Lambda function of AWS, JSON, Android Studio and Firebase.

2.4 User Documentation

The following documents are to be delivered along with the software :

- An e-Assistant user manual.
- An e-Assistant installation and maintenance manual.

3. System Features

3.1 Alexa Tell Me My Schedule:

This feature will invoke a task to e-Assistant which will lookup to Professor's schedule and call out timings of the class, subject and class number. This feature will automatically remind Professors about their class 10 minutes to the lecture.

3.2 Alexa Take Attendance:

Alexa will take students attendance. Attendance list will be updated on android App accrdingly. If a student is present it will be reflected in the app with a green dot in front of the student's name and in case of an absentee it is reflected using a Red dot.

3.3 Alexa Make A List:

Using this feature a professor can make any list with desirable column fields just by calling out voice command followed by data of the list. List will be visible on the app in realtime. For example making list of Practical marks of students, user can also share this list via e-mail.

3.4 Attendance Report:

This feature of Android App will show the complete report of attendance of each class along with defaulters marked with red colour and professor could send warning email to all defaulters with one click.

3.5 Announcements:

This feature of Android App will make sharing vital information with professors really easy. All HoDs along with Director of college can make announcements, which will be received as push notification as well as alexa will notify professors.

4. Other Non- Functional Requirements:

4.1 Understandability

Applications code and comments should be written descriptively. The names of classes, methods and variables should be self-descriptive. Methods and classes will be commented to detail the purpose.

4.2 Completeness

All external libraries including their respective licenses should be documented.

4.3 Conciseness, Consistency And Efficiency

Application code will be reviewed regularly to remove redundancy, reduce complexity and increase efficiency.

4.4 Portability

Testing on multiple platforms(Windows 7, Windows 8, Windows 10) and implementation should ensure that code and external libraries are not platform or implementation dependent**.**

4.5 Maintainability

Application code will be cohesive and have easily recognisable functionality. Classes will be abstract enough to facilitate changes in data structures. Classes and function modularity should be implemented to avoid the need for major refactoring.

4.6 Reliability

Checked exception handling is highly recommended. Extensive testing is also required . Pending decision, information that is needed, conflicts awaiting resolution , and the like.

CHAPTER THREE

SYSTEM DESIGN

3.1 ER-DIAGRAM

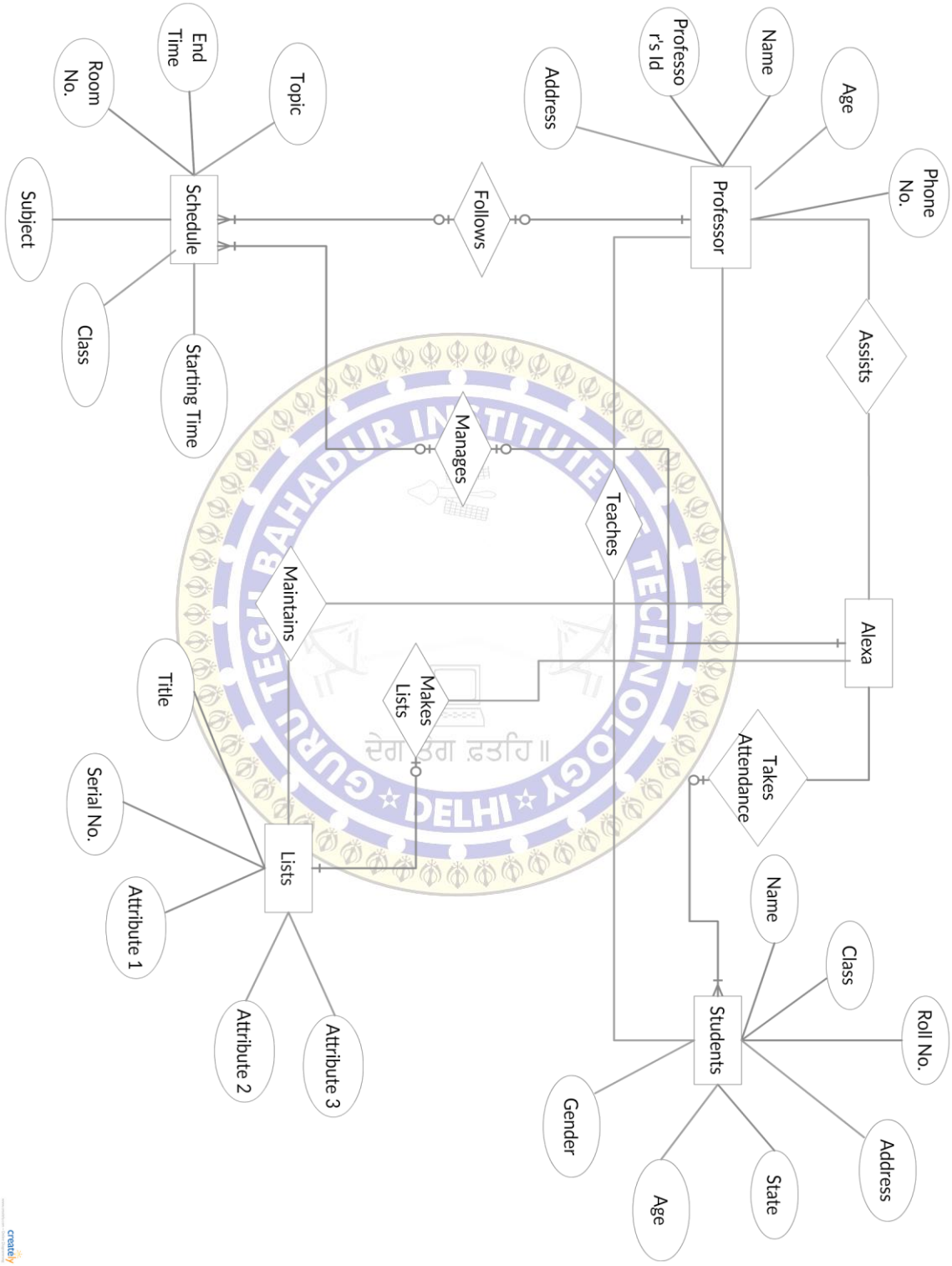


Fig 3.1.1 Entity Relationship Diagram

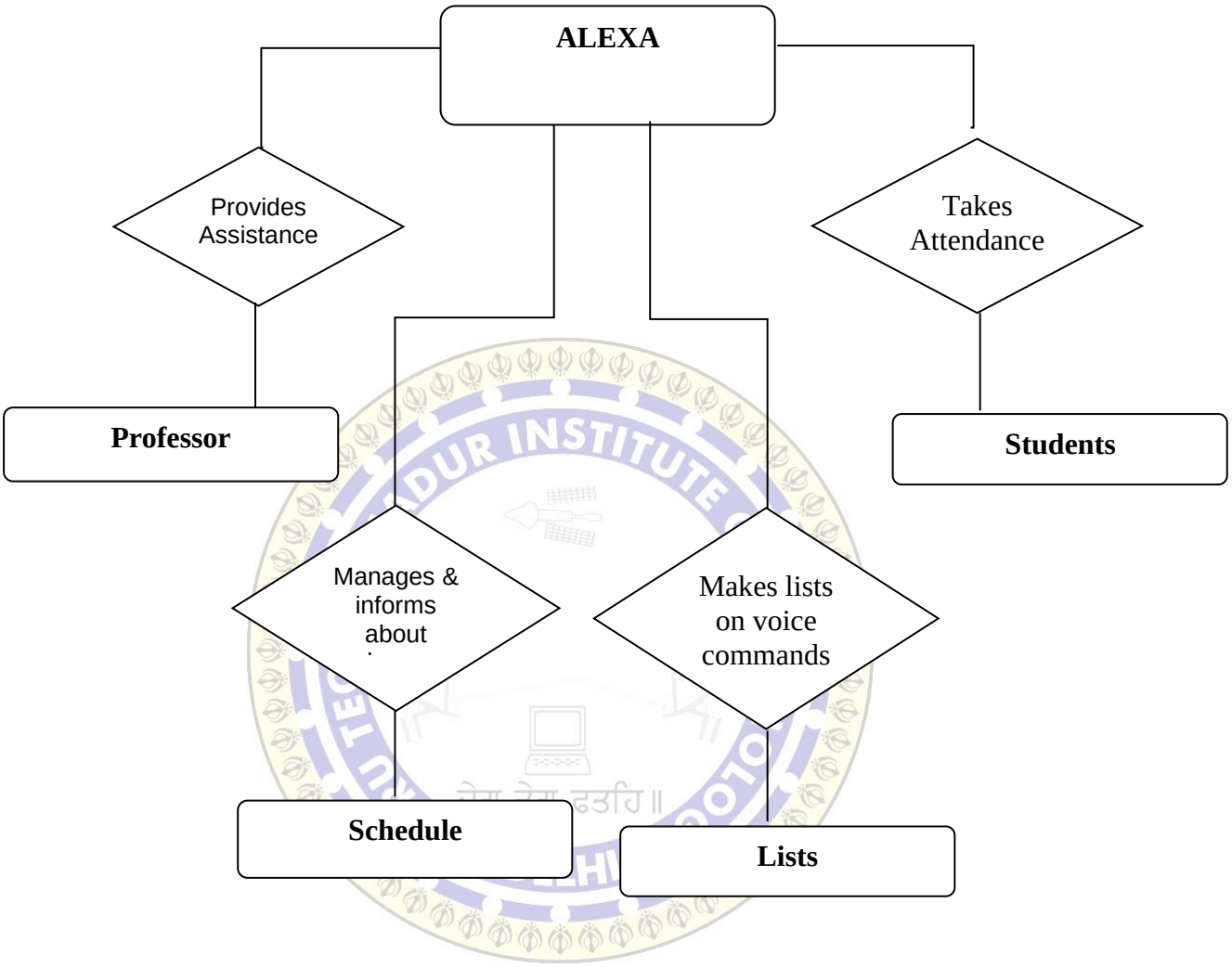


Fig 3.1.2 Relationship between entities

3.2 ACTIVITY LIFE CYCLE

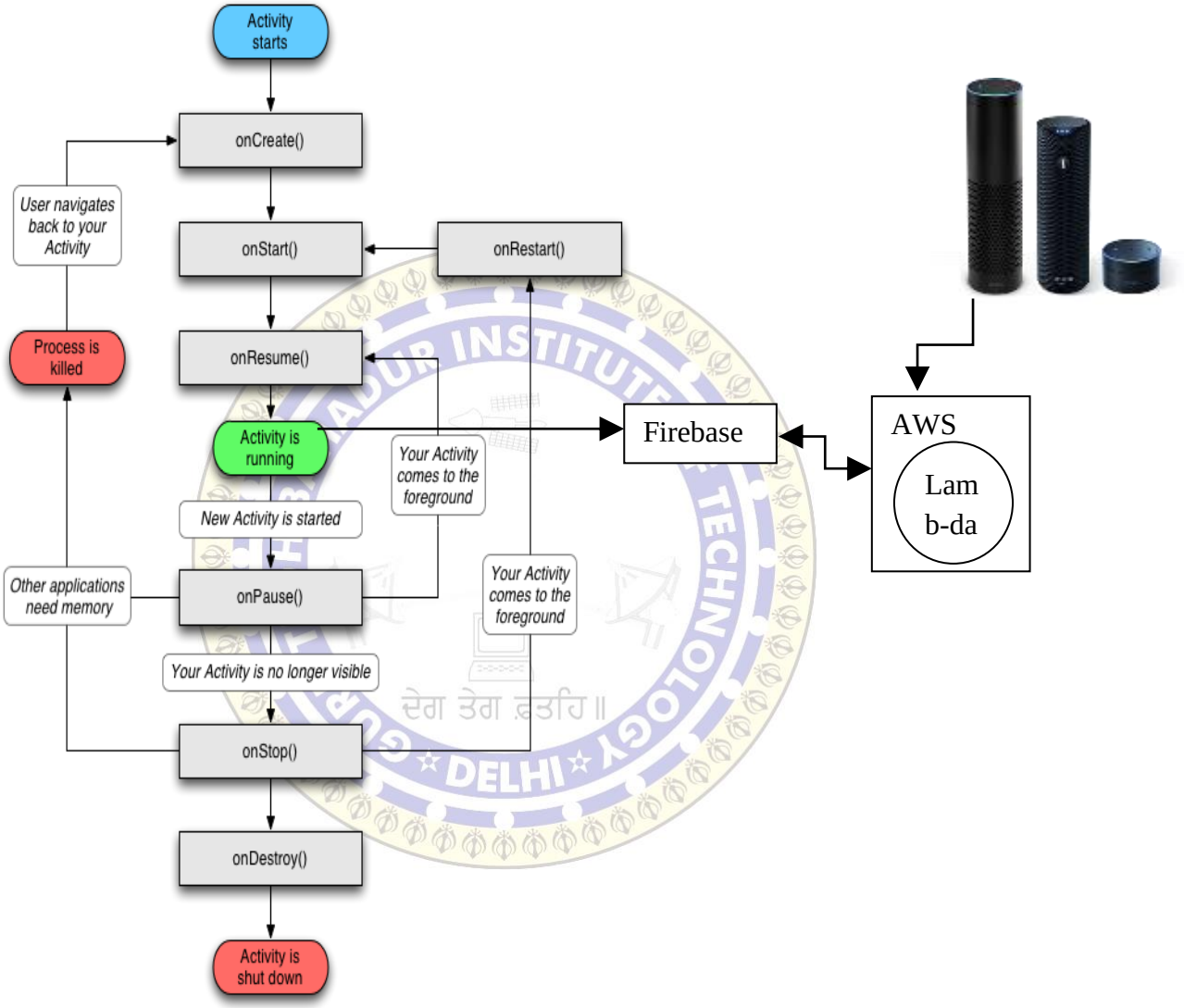


Fig 3.2.1 Activity Life Cycle

3.3 DATA FLOW DIAGRAM

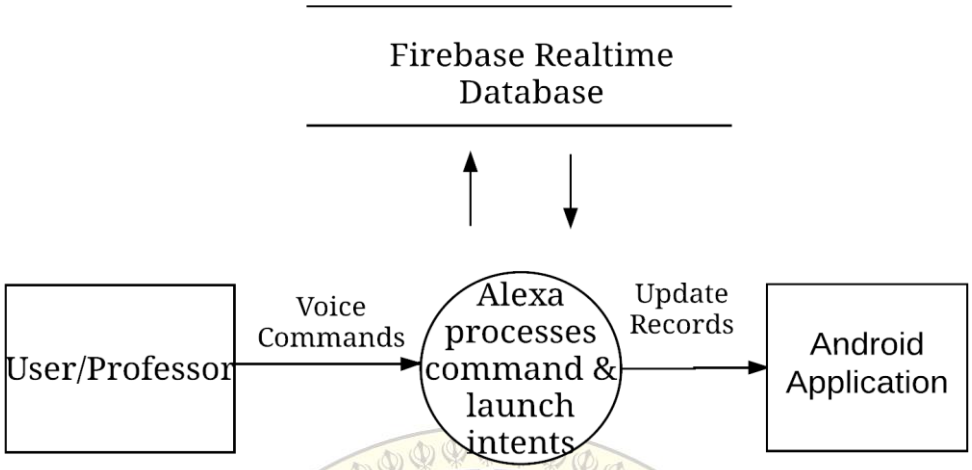


Fig 3.3.1 Level-0 DFD

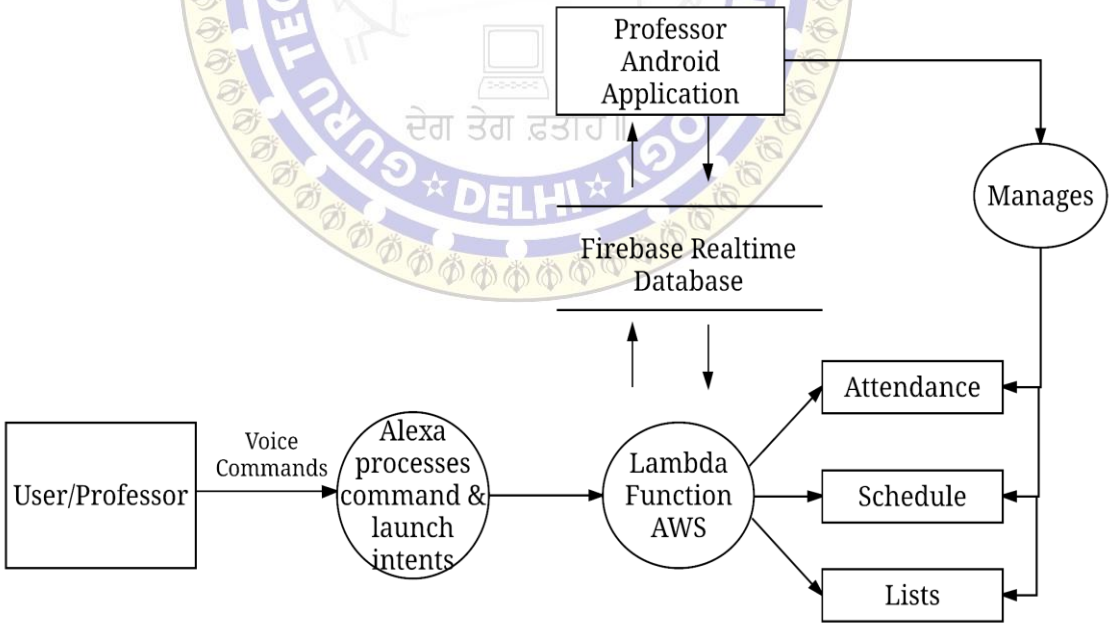


Fig 3.3.2 Level-1 DFD

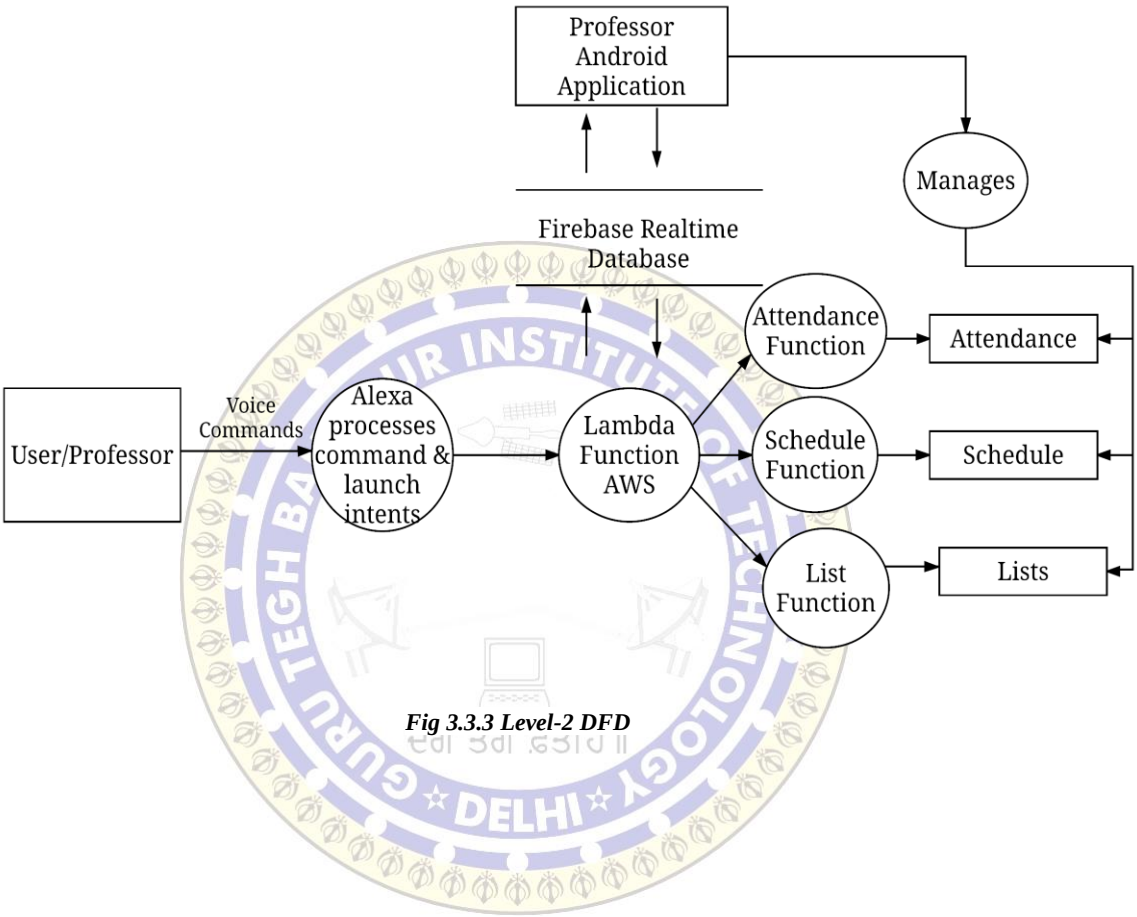


Fig 3.3.3 Level-2 DFD

3.4 USE CASE DIAGRAM

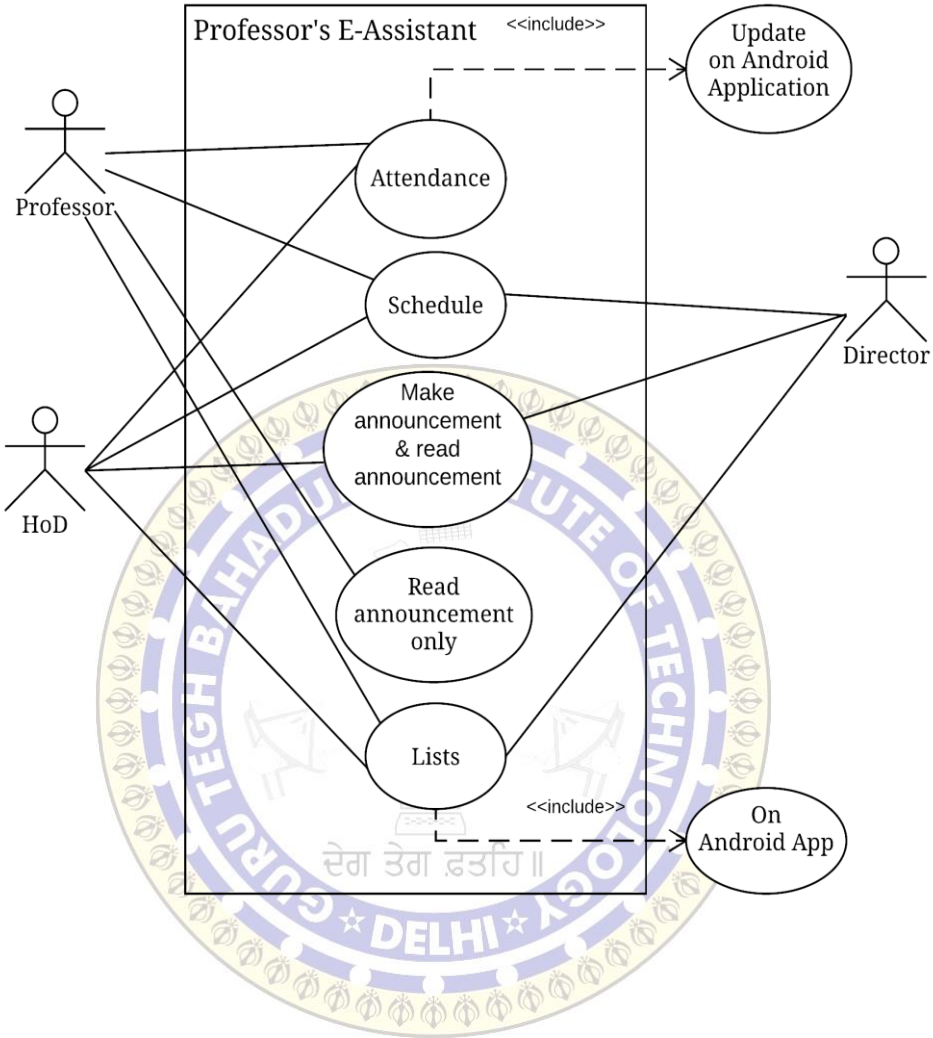


Fig 3.4.1 Use Case Diagram

3.5 CLIENT SERVER ARCHITECTURE

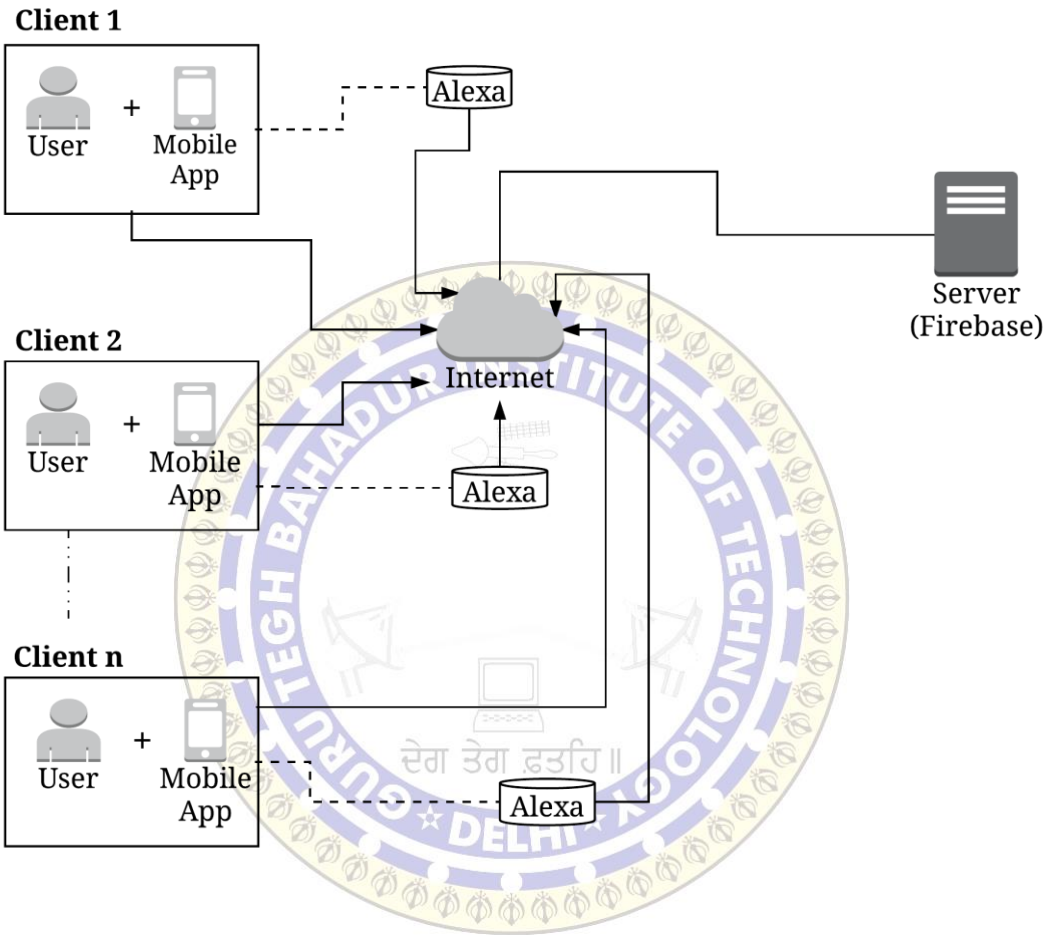


Fig 3.5.1 Client Server Architecture

3.6 FLOWCHART

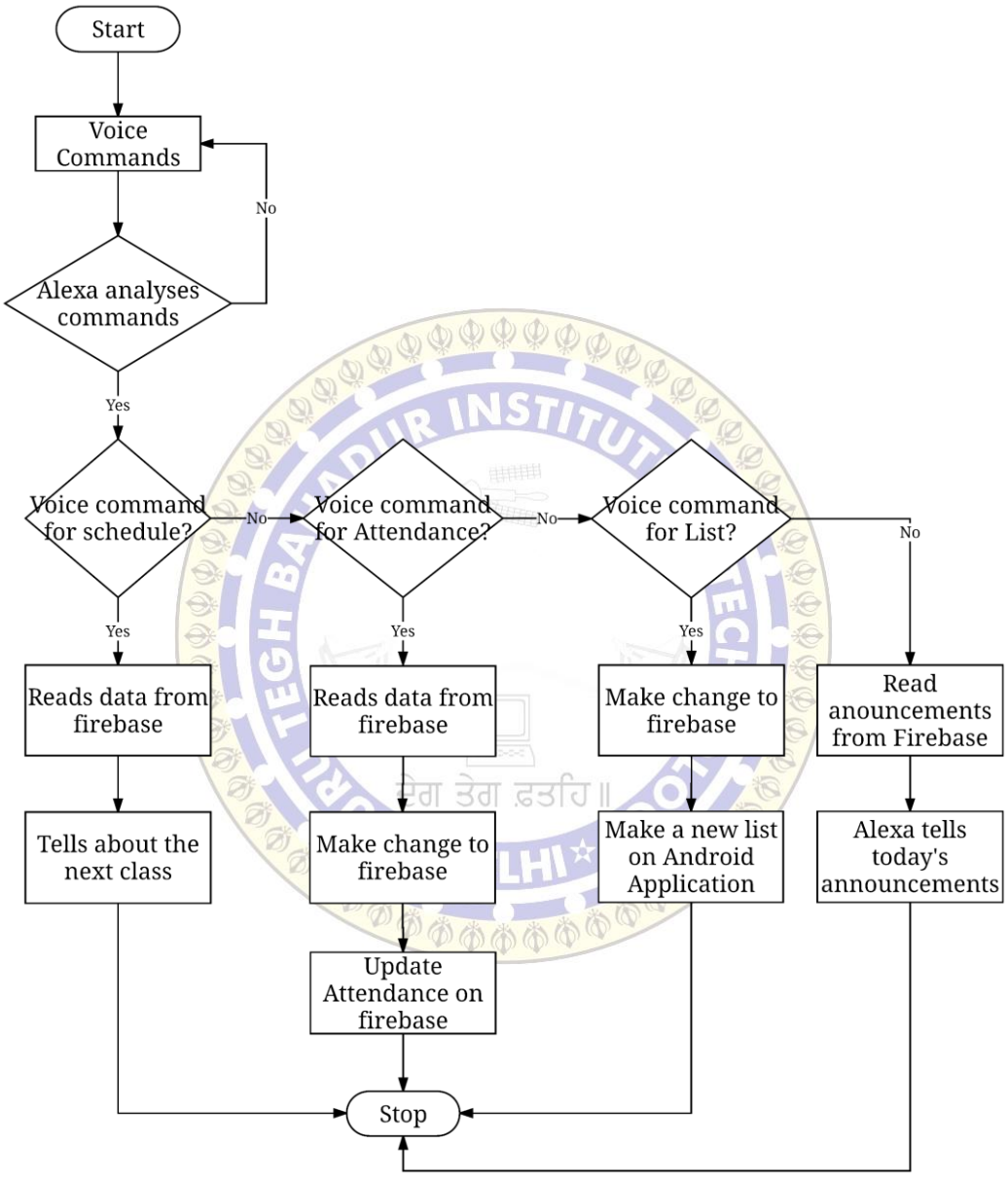


Fig 3.6.1 Flowchart

CHAPTER FOUR

SYSTEM REQUIREMENTS

SYSTEM REQUIREMENTS

Developer's Side:

- **Hardware:**

  Processor: Intel i3 or above

  RAM: 4 GB or more

  Hard Disk: 8 GB or more

  - Any android phone having android version above lollipop
  - Any of the device among Echo Dot, Echo or Echo Plus

- **Software:**

  - Android Studio
  - Python 2.7
  - AWS Platform
  - AWS Lambda Function
  - Firebase Realtime Database
  - JSON

  .

User's Side:

- Any android phone having android version above lollipop.
- Any of the device among Echo Dot, Echo or Echo Plus.

CHAPTER FIVE

EXPLANATION AND WORKING

5. Explanation and Working

5.1 Voice Command to Alexa:

Fig 5.1.1 Voice Command to Alexa

We give voice commands to Alexa, commanding it to do one of its task. We use the word “Alexa” to wake it up. After waking up Alexa the user gives it a command to perform one particular task. Alexa first records this command. Since Alexa cannot understand English language directly, it uses natural language processing at its end to understand the user’s commands.

After this it compares if this processed command matches any of the intents made by the programmer. If yes, it calls that particular intent to perform the desired task.

Whenever you make a voice request, Alexa-enabled devices record or stream audio clips of what you say. Those files are sent to a server, the real brains of the operation to process the audio and formulate a response. The recorded clips are associated to your user account, and that process is enabled by default.

Alexa needs an internet connection to work. It does have a very rudimentary education, though: The only spoken commands it understands on its own are “wake words” or “activation phrases,” things like "Alexa". Once you say those magic words, the voice assistants jump to life, capture your voice request, and sling it to its disembodied cloud brains over Wi-Fi.

5.2 Alexa Connecting to Lambda, Lambda to Firebase:

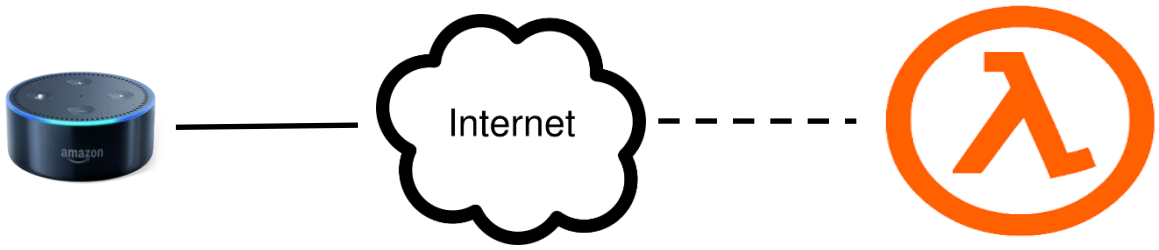


Fig 5.2.1 Alexa connecting to Lambda

Here analysis is done. Now the procedure of the called intent starts executing. These functions or procedures are made in python and they access the firebase to collect the user required information.

Example: If a professor gives command to Alexa to tell his/her schedule, it will run the intent related to scheduling in the background and connect to the firebase to check the schedule timings.
Lambda is the tool used to write the python script i.e. all our functions and procedures which are used to execute a particular task.

The easiest way to build the cloud-based service for a custom Alexa skill is by using AWS Lambda, an Amazon Web Services offering that runs your code only when it's needed and scales automatically, so there is no need to provision or continuously run servers. You upload the code for your Alexa skill to a Lambda function and Lambda does the rest, executing it in response to Alexa voice interactions and automatically managing the compute resources for you.
Using a Lambda function for your service eliminates some of the complexity around setting up and managing your own endpoint:

1.You do not need to administer or manage any of the compute resources for your service.

2.You do not need an SSL certificate.

3.You do not need to verify that requests are coming from the Alexa service yourself. Access to execute your function is controlled by permissions within AWS instead.

4.AWS Lambda runs your code only when you need it and scales with your usage, so there is no need to provision or continuously run servers.

5.3 Android App Connecting to Firebase:

Fig 5.3.1 Android Application connecting to Firebase

Alexa access the firebase, so if needed it makes changes to it.

These changes or these status updates are made to the android app as well because the app accesses the firebase too.

So the changes are updated on the app and we can manage our task eg. attendance, list of schedule etc.. from the android app.

Connect an Android app to Firebase

Step 1:

If you already have an existing Android app, open it in Android Studio or create a new Android project. Open Firebase, go to the console and click on Add Project. Enter Project name and Country and then click on Create Project.

Step 2:

Now click on ‘Add Firebase to your Android App’ and copy the package name of your app. All other fields are optional and can be left blank.

Step 3:

Now download the google-services.json file and copy it to the app folder of your Android project

Step 4:

Copy first dependency in your app's project level build.gradle. Now copy the second dependency at the bottom of the app level build.gradle file and click on Sync Now.

Step 5:

If everything goes well, you will see your project added in the Firebase Console.

5.4 Description of the Technologies Used:

1. Alexa

Fig 5.4.1. Alexa Echo Devices

- Alexa is an intelligent personal assistant developed by Amazon, first used in the Amazon Echo and the Amazon Echo Dot devices developed by Amazon Lab126. It is capable of voice interaction, music playback, making to-do lists, setting alarms, streaming podcasts, playing audiobooks, and providing weather, traffic, and other real time information, such as news. Alexa can also control several smart devices using itself as a home automation system.

1.1 Functions

1. Home Automation

- In the home automation space, Alexa can interact with devices from Belkin WeMo, ecobee, Geeni, IFTTT, Insteon, LIFX, LightwaveRF, Nest Thermostats, Philips Hue, SmartThings, Wink,and Yonomi. The Home Automation feature was launched on April 8, 2015

2. Ordering

- Take-out food can be ordered using Alexa; as of May 2017 food ordering using Alexa is supported by Domino's Pizza, Grubhub, Pizza Hut, Seamless, and Wingstop.Also, users of Alexa in the UK can order meals via Just Eat. In early 2017, Starbucks announced a private beta for placing pick-up orders using Alexa.In addition, users can order meals using Amazon Prime Now via Alexa in 20 major US cities.

3. Music

- Alexa is able to stream media and music directly. To do this, Alexa's device should be linked to the Amazon account, which enables access to one's Amazon Music library, in addition to any audiobooks available in one's Audible library. Amazon Prime members have an additional ability to access stations, playlists, and over two million songs free of charge. Amazon Music Unlimited subscribers also have access to a list of millions of songs.

4. Sports

- Alexa allows the user to hear updates on supported sports teams. A way to do this is by adding the sports team to the list created under Alexa's Sports Update app section.

1.2 Alexa Skills Kit

Amazon allows developers to build and publish skills for Alexa using the Alexa Skills Kit. These skills are third-party developed voice experiences that add to the capabilities of any Alexa-enabled device (such as the Echo). These skills are available for free download using the Alexa app. Skills are continuously being added to increase the capabilities available to the user. A "Smart Home Skill API" is available. All of the code runs in the cloud – nothing is on any user's device. A developer can follow tutorials to learn how to quickly build voice experiences for their new and existing applications.

1.3 Alexa Voice Service

Amazon allows device manufacturers to integrate Alexa voice capabilities into their own connected products by using the Alexa Voice Service (AVS), a cloud-based service that provides APIs to interface with Alexa. Products built using AVS have access to Alexa's growing list of capabilities including all of the Alexa Skills. AVS provides cloud-based automatic speech recognition (ASR) and natural language understanding (NLU). There are no fees for companies looking to integrate Alexa into their products by using AVS.

The voice of Amazon Alexa is generated by a long short-term memory artificial neural network.

2. Firebase

Fig 5.4.2 Firebase Logo

The Firebase Real Time Database is a cloud-hosted database. Data is stored as JSON and synchronized in real time to every connected client. When you build cross-platform apps with our iOS, Android, and JavaScript SDKs, all of your clients share one Real Time Database instance and automatically receive updates with the newest data.

2.1 Key capabilities

1. Realtime

Instead of typical HTTP requests, the Firebase Real Time Database uses data synchronization—every time data changes, any connected device receives that update within milliseconds. Provide collaborative and immersive experiences without thinking about networking code.

2. Offline

Firebase apps remain responsive even when offline because the Firebase Real Time Database SDK persists your data to disk. Once connectivity is re-established, the client device receives any changes it missed, synchronizing it with the current server state.

3. Accessible from Client Devices

The Firebase Real Time Database can be accessed directly from a mobile device or web browser; there's no need for an application server. Security and data validation are available through the Firebase Real Time Database Security Rules, expression-based rules that are executed when data is read or written.

2.2 How it works

The Firebase Realtime Database lets you build rich, collaborative applications by allowing secure access to the database directly from client-side code. Data is persisted locally, and even while offline, realtime events continue to fire, giving the end user a responsive experience. When the device regains connection, the Realtime Database synchronizes the local data changes with the remote updates that occurred while the client was offline, merging any conflicts automatically.

The Realtime Database provides a flexible, expression-based rules language, called Firebase Realtime Database Security Rules, to define how your data should be structured and when data can be read from or written to. When integrated with Firebase Authentication, developers can define who has access to what data, and how they can access it.

3. AMAZON WEB SERVICES

Fig 5.4.3 Amazon Web Services Logo

3.1 Overview

Amazon Web Services (**AWS**) is a subsidiary of Amazon.com that provides on-demand cloud computing platforms to individuals, companies and governments, on a paid subscription basis with a free-tier option available for 12 months. The technology allows subscribers to have at their disposal a full-fledged virtual cluster of computers, available all the time, through the internet. AWS's version of virtual computers have most of the attributes of a real computer including hardware (CPU(s) & GPU(s) for processing, local/RAM memory, hard-disk/SSD storage); a choice of operating systems; networking; and pre-loaded application software such as web servers, databases, CRM, etc. Each AWS system also virtualizes its console I/O (keyboard, display, and mouse), allowing AWS subscribers to connect to their AWS system using a modern browser.

The browser acts as a window into the virtual computer, letting subscribers log-in, configure and use their virtual systems just as they would a real physical computer. They can choose to deploy their AWS systems to provide internet-based services for their own and their customers' benefit.

The AWS technology is implemented at server farms throughout the world, and maintained by the Amazon subsidiary. Fees are based on a combination of usage, the hardware/OS/software/networking features chosen by the

subscriber, required availability, redundancy, security, and service options. Based on what the subscriber needs and pays for, they can reserve a single virtual AWS computer, a cluster of virtual computers, a physical (real) computer dedicated for their exclusive use, or even a cluster of dedicated physical computers. As part of the subscription agreement, Amazon manages, upgrades, and provides industry-standard security to each subscriber's system. AWS operates from many global geographical regions including 6 in North America.

3.2 It's Services and Products

Amazon Web Services offers a broad set of global cloud-based products including compute, storage, databases, analytics, networking, mobile, developer tools, management tools, IoT, security and enterprise applications. These services help organizations move faster, lower IT costs, and scale. AWS is trusted by the largest enterprises and the hottest start-ups to power a wide variety of workloads including: web and mobile applications, game development, data processing and warehousing, storage, archive, and many others.

4. LAMBDA FUNCTION OF AMAZON WEB SERVICES:

Fig 5.4.4 AWS Lambda Logo

AWS Lambda is a compute service that lets you run code without provisioning or managing servers. AWS Lambda executes your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time you consume - there is no charge when your code is not running. With AWS Lambda, you

can run code for virtually any type of application or backend service - all with zero administration. AWS Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, code monitoring and logging. All you need to do is supply your code in one of the languages that AWS Lambda supports (currently Node.js, Java, C# and Python).

4.1 Building Applications with Lambda

- **File processing** – Suppose you have a photo sharing application. People use your application to upload photos, and the application stores these user photos in an Amazon S3 bucket. Then, your application creates a thumbnail version of each user's photos and displays them on the user's profile page. In this scenario, you may choose to create a Lambda function that creates a thumbnail automatically. Amazon S3 is one of the supported AWS event sources that can publish *object-created events* and invoke your Lambda function. Your Lambda function code can read the photo object from the S3 bucket, create a thumbnail version, and then save it in another S3 bucket.

- **Data and analytics** – Suppose you are building an analytics application and storing raw data in a DynamoDB table. When you write, update, or delete items in a table, DynamoDB streams can publish item update events to a stream associated with the table. In this case, the event data provides the item key, event name (such as insert, update, and delete), and other relevant details. You can write a Lambda function to generate custom metrics by aggregating raw data.

- **Websites** – Suppose you are creating a website and you want to host the backend logic on Lambda. You can invoke your Lambda function over HTTP using Amazon API Gateway as the HTTP endpoint. Now, your

web client can invoke the API, and then API Gateway can route the request to Lambda.

- **Mobile applications** – Suppose you have a custom mobile application that produces events. You can create a Lambda function to process events published by your custom application. For example, in this scenario you can configure a Lambda function to process the clicks within your custom mobile application.

5. Android Studio

Fig 5.4.5.1 Android Studio Logo

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on Intellij IDEA. On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices
- Instant Run to push changes to your running app without building a new APK

• Code templates and GitHub integration to help you build common app features .

• Lint tools to catch performance, usability, version compatibility, and other problems

• C++ and NDK support

5.1 Project Structure

Each project in Android Studio contains one or more modules with source code files and resource files. Types of modules include:

- Android app modules
- Library modules
- Google App Engine modules

By default, Android Studio displays your project files in the Android project view. This view is organized by modules to provide quick access to your project's key source files.

All the build files are visible at the top level under Gradle Scripts and each app module contains the following folders:

- manifests: Contains the AndroidManifest.xml file.
- java: Contains the Java source code files, including JUnit test code.
- res: Contains all non-code resources, such as XML layouts, UI strings.

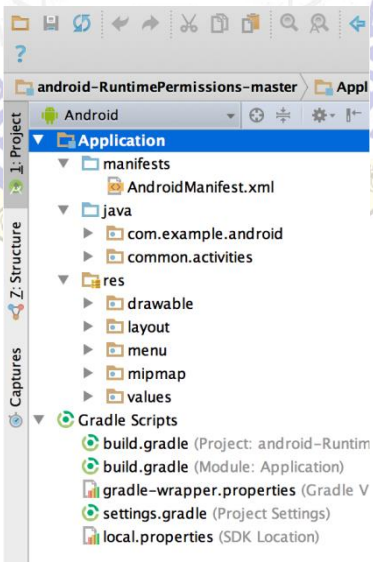


Fig 5.4.5.2 Project Structure

5.2 Gradle Build System

Android Studio uses Gradle as the foundation of the build system, with more Android-specific capabilities provided by the Android plugin for gradle. This

build system runs as an integrated tool from the Android Studio menu, and independently from the command line. You can use the features of the build system to do the following:

- Customize, configure, and extend the build process.
- Create multiple APKs for your app, with different features using the same project and modules.
- Reuse code and resources across source sets.

By employing the flexibility of Gradle, you can achieve all of this without modifying your app's core source files. Android Studio build files are named build.gradle. They are plain text files that use Groovys syntax to configure the build with elements provided by the Android plugin for Gradle. Each project has one top-level build file for the entire project and separate module-level build files for each module. When you import an existing project, Android Studio automatically generates the necessary build files.

5.3 Build Variant

The build system can help you create different versions of the same application from a single project. This is useful when you have both a free version and a paid version of your app, or if you want to distribute multiple APKs for different device configurations on Google Play.

5.4 Multiple APK Support

Multiple APK support allows you to efficiently create multiple APKs based on screen density or ABI. For example, you can create separate APKs of an app for the hdpi and mdpi screen densities, while still considering them a single variant and allowing them to share test APK, javac, dx, and ProGuard settings.

5.5 Performance Profiler

Android Studio provides performance profilers so you can more easily track your app’s memory and CPU usage, find deallocated objects, locate memory

leaks, optimize graphics performance, and analyze network requests. With your app running on a device or emulator, open the **Android Profiler** tab.

5.6 Version Control Basics

Android Studio supports a variety of version control systems (VCS's), including Git, GitHub, CVS, Mercurial, Subversion, and Google Cloud Source Repositories.

After importing your app into Android Studio, use the Android Studio VCS menu options to enable VCS support for the desired version control system, create a repository, import the new files into version control, and perform other version control operations:

1. From the Android Studio **VCS** menu, click **Enable Version Control Integration**.
2. From the drop-down menu, select a version control system to associate with the project root, and then click **OK**.

The VCS menu now displays a number of version control options based on the system you selected.

6. Python 2.7

Fig 5.4.6 Python Logo

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy that

emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale.

Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library.

Python interpreters are available for many operating systems, allowing Python code to run on a wide variety of systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation.

CHAPTER SIX

SCOPE, SUMMARY AND CONCLUSION

Scope, Summary and Conclusion

4.1 Conclusion

The application satisfies all the provided specifications.

4.2 Future Scope

At present this application targets teachers and can be used by any teacher in the domain. In future several other features could be added to the application to make it enterprise ready. Feature like “taking Attendance” could be made more refined by implementing voice profiling which can help to identify people based on their unique voice tones. Since Alexa is an assistant, this application could be similarly expanded to different domains like hospitals, metros, car assistant, kitchen etc. For example, in case of Metro Alexa can be used to provide tickets to commuters whenever they ask for the ticket to their destination.

4.3 Summary

This project aims at providing assistance to professors and teachers in their daily chores.

The project can assist them in specific works that are petty but are most important.

Professor’s e-Assistant is an IOT based project that uses Android and Alexa echo devices for its operation. Alexa is an intelligent personal assistant developed by Amazon, it is capable of voice interaction and taking voice commands. Alexa is programmable device which can also be connected to an Android app.

Professor’s e-Assistant can help professors to store their schedules and get notified about them a few minutes earlier prior to the commencement of the event. It also helps professors take out the most hectic task, attendance, in

which the alexa is used to call out the names and listens to students' replies to mark the attendance.

The main purpose of reducing the workload of teachers and professors is achieved by Professor's e-Assistant.

4.4 Advantages

- This application helps to prevent physically making the lists and records by teachers.
- It helps teachers keep easy track of their schedule.
- It also helps to shortlist the students having short attendance and also allows teacher to warn the students of their performance.
- It provides convenient platform for important notices and announcements to be circulated among teachers regulated by the Director or HoDs of the institutions.

4.5 Limitations

- Amazon Echo Dot devices are needed to be connected via Wi-Fi so as to function properly.
- A user need to have his/her own Echo Dot device.
- Echo Dot doesn't contain any batteries; therefore, it needs to be connected with continuous source of power supply.
- Alexa faces issues while listening to Latin and Indian accents.

REFERENCES

References

[1] **Android Dev Support –**

https://developer.android.com/reference/android/support

[2] **Java: The Complete Reference** by **Herbert Schildt**

[3] **Androidhive -** https://www.androidhive.info/

[4] **Amazon Skill Set –** https://developer.amazon.com/alexa-skills-kit/alexa-skill-quick-start-tutorial

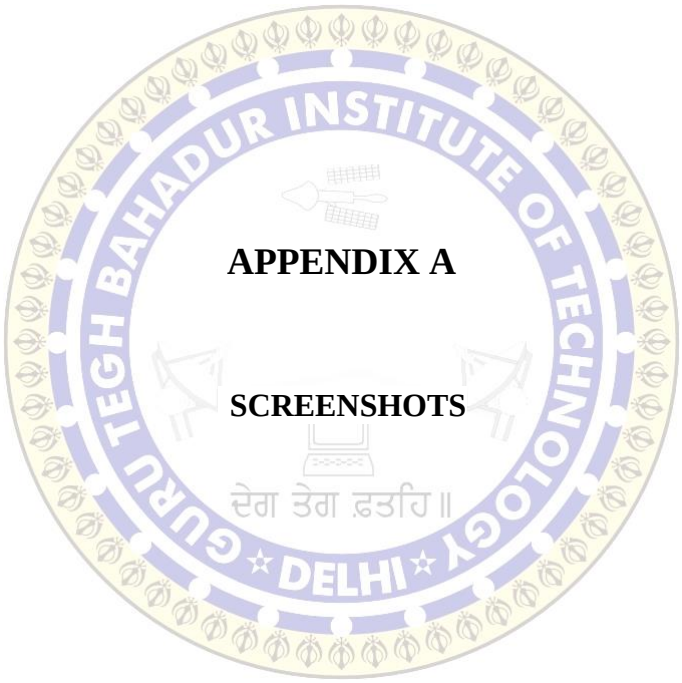
[5] **JournalDev -** https://www.journaldev.com/10096/android-viewpager-example

[6] **Firebase** - https://firebase.google.com/docs/database/

[7] **O'reilly Python Cookbook** by Alex Martelli, David Ascher, Anna Martelli Ravenscroft.

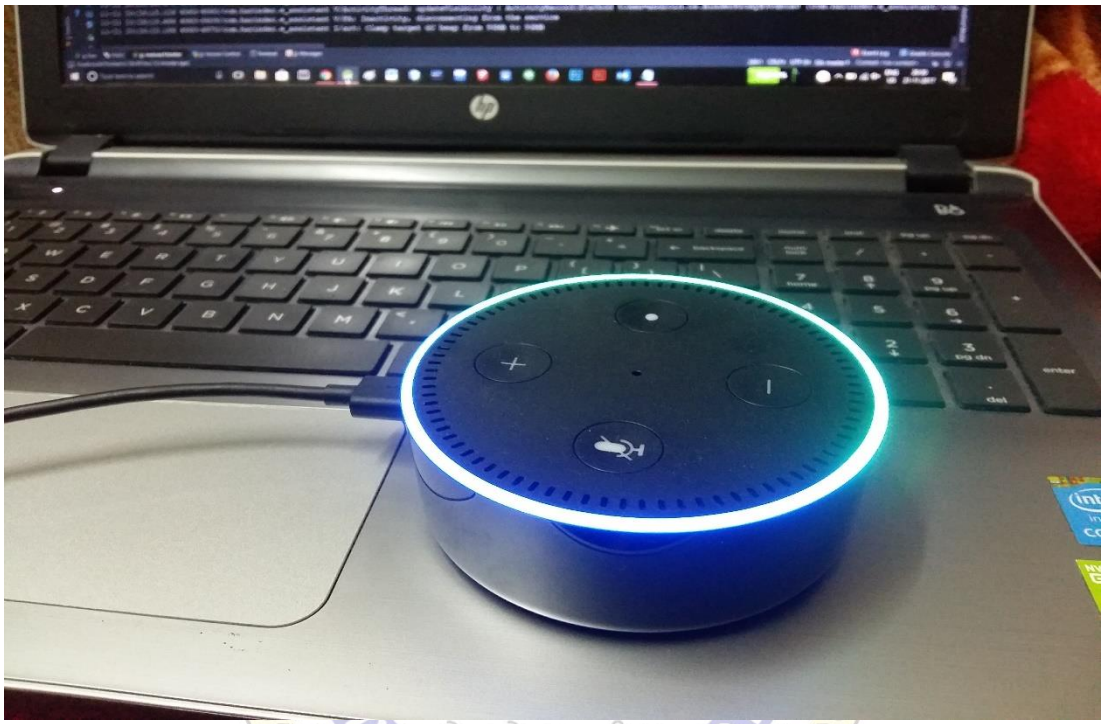
[8] https://alexatutorial.com/

[9] **Python official Documentation -** http://www.python.org/doc/

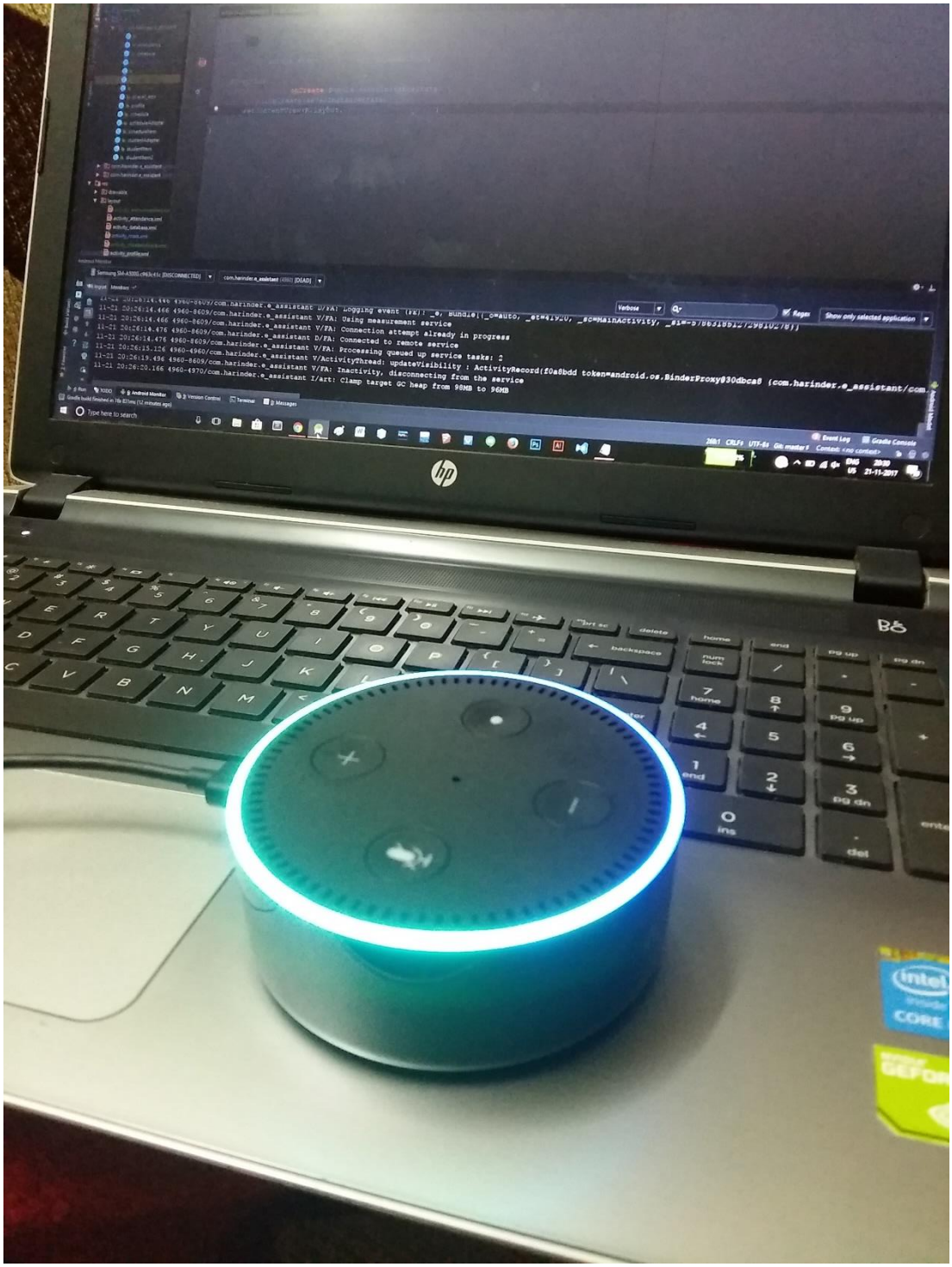


APPENDIX A

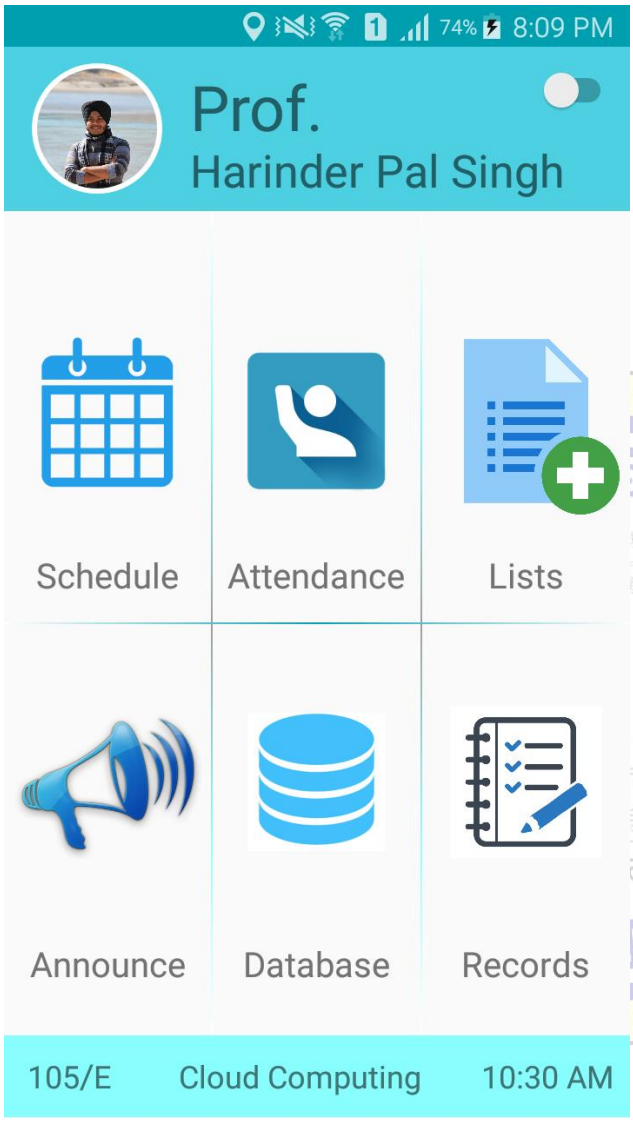
SCREENSHOTS



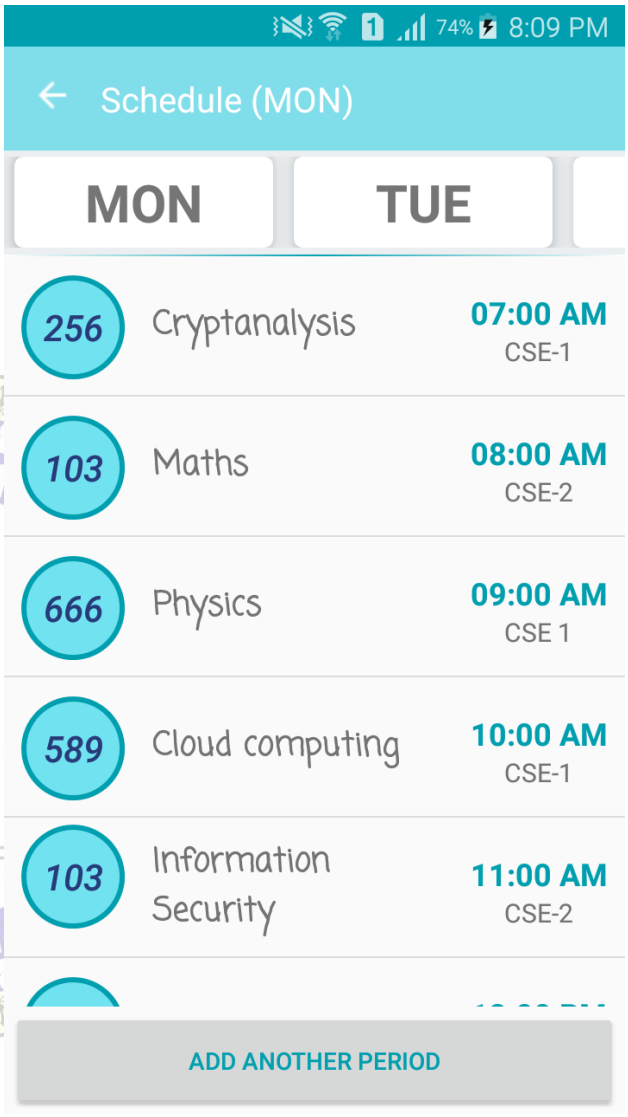
Screenshot 1



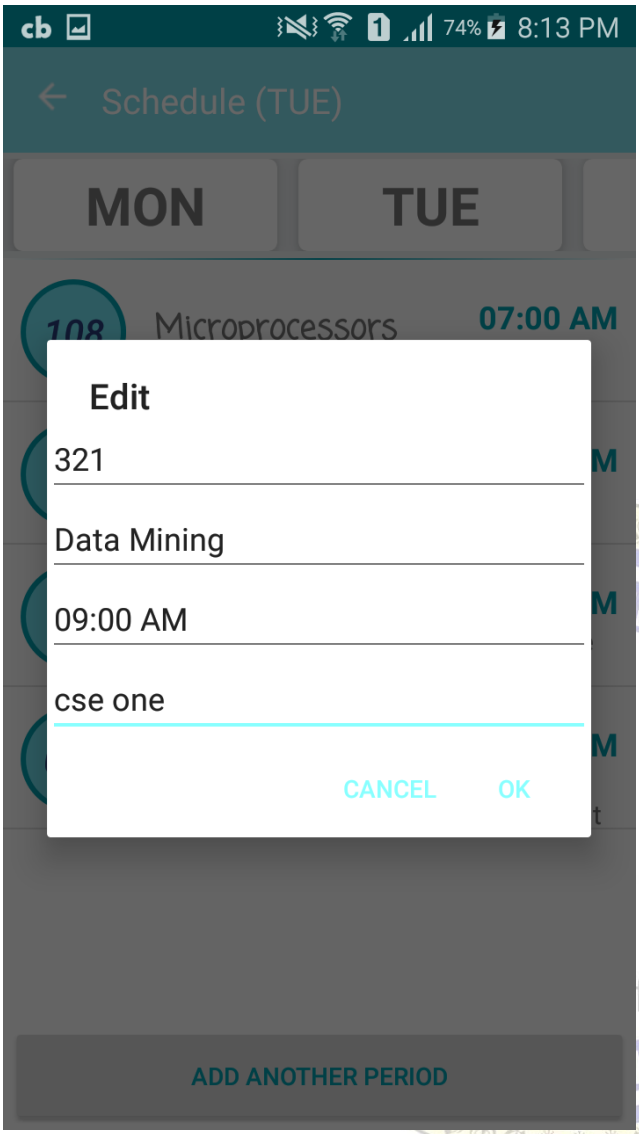
Screenshot 2



Screenshot 3

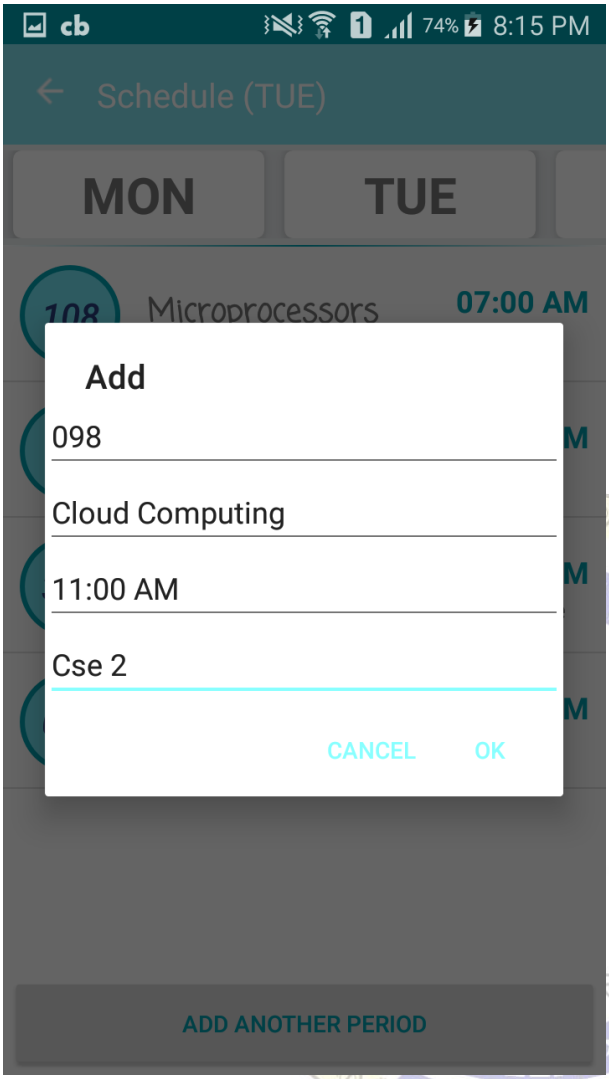


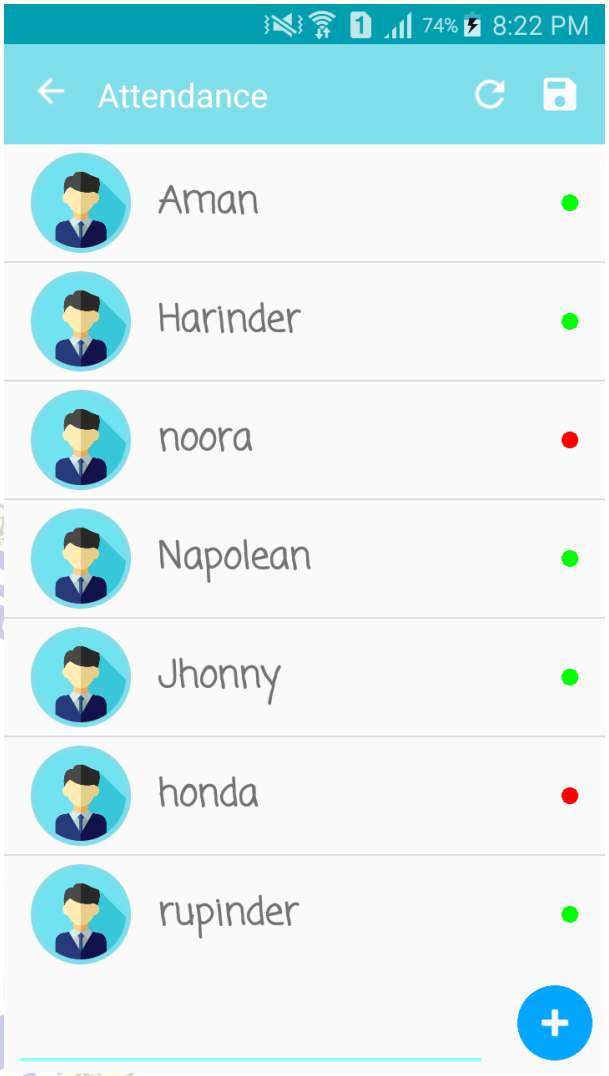
Screenshot 4



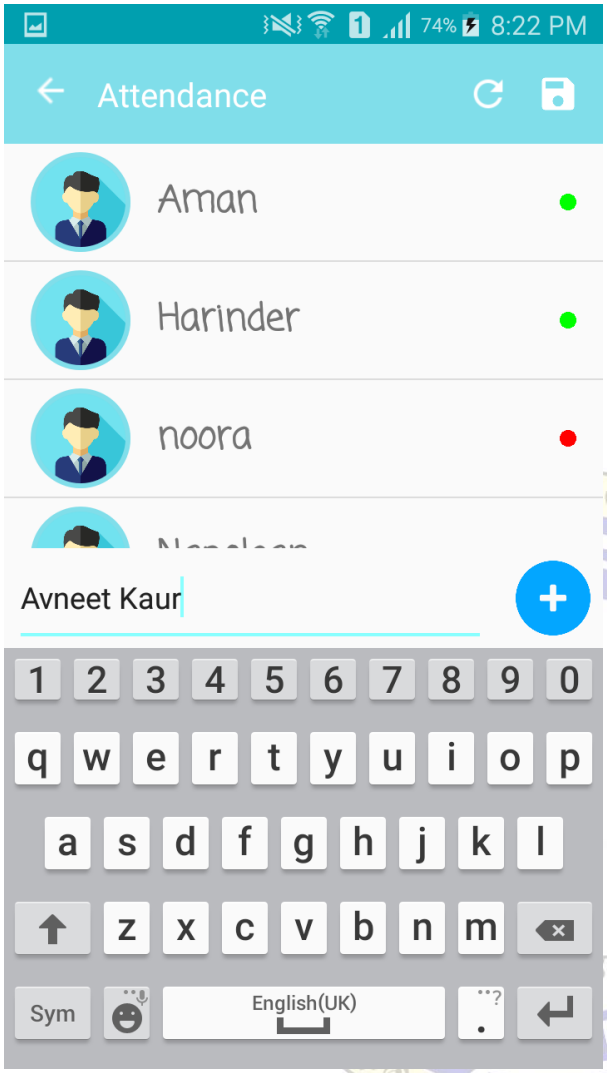
Screenshot 5

Screenshot 6

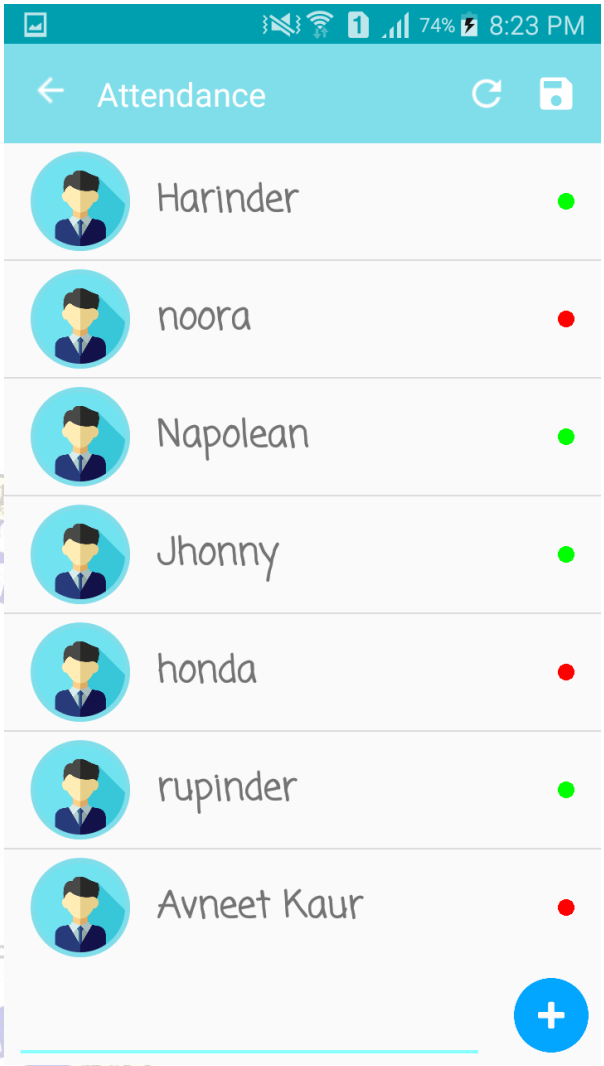




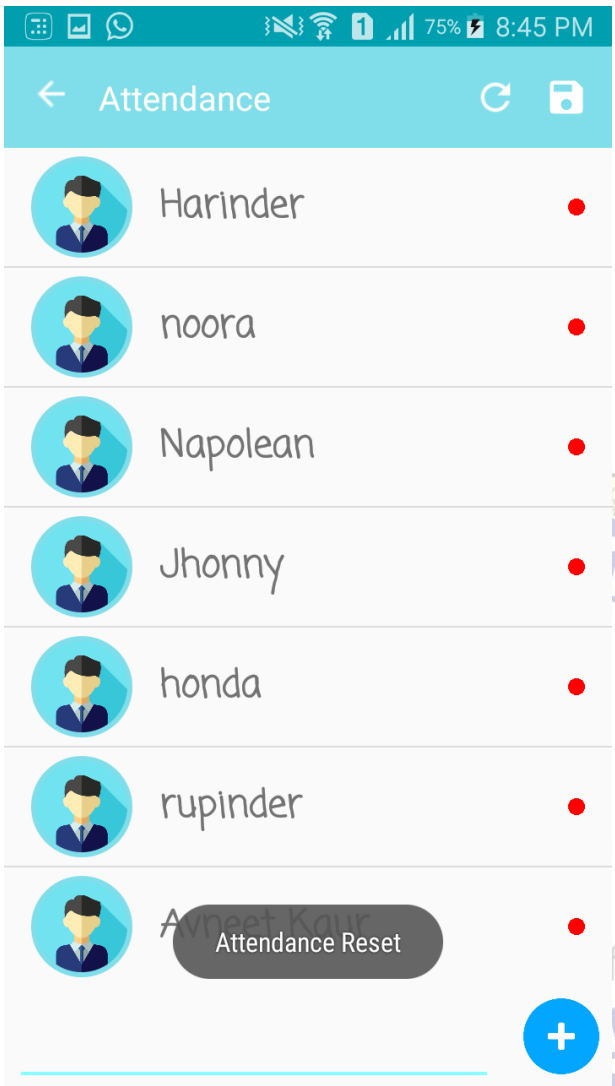
Screenshot 7 **Screenshot 8**

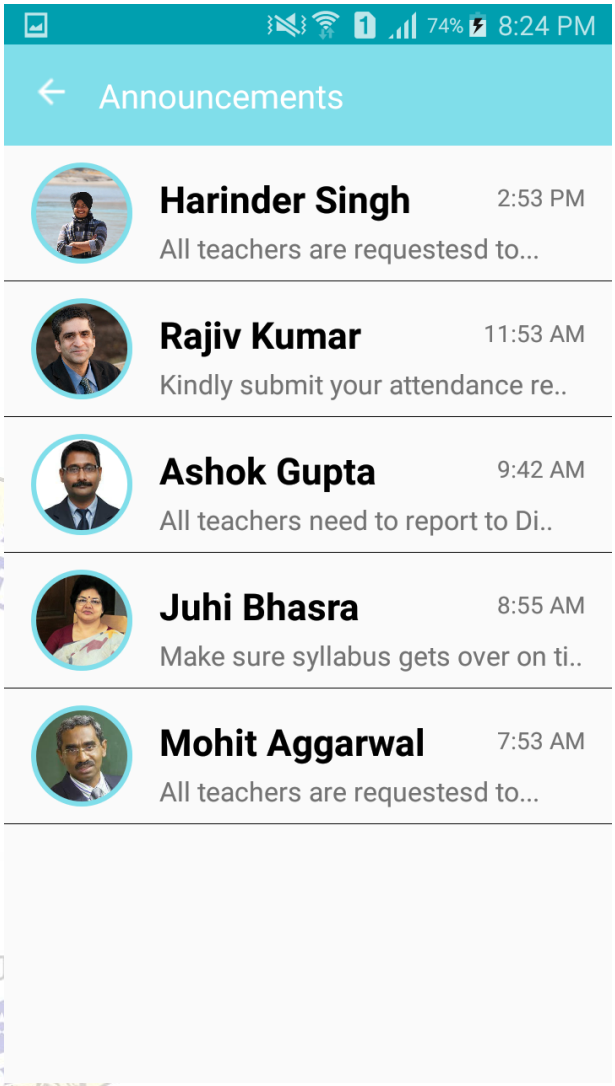


Screenshot 9



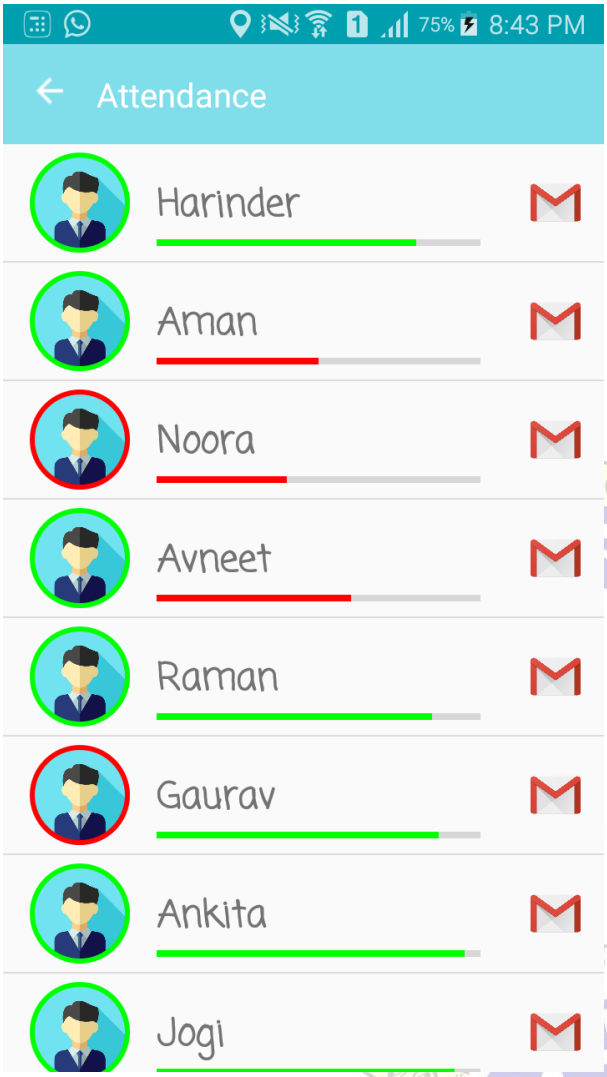
Screenshot 10



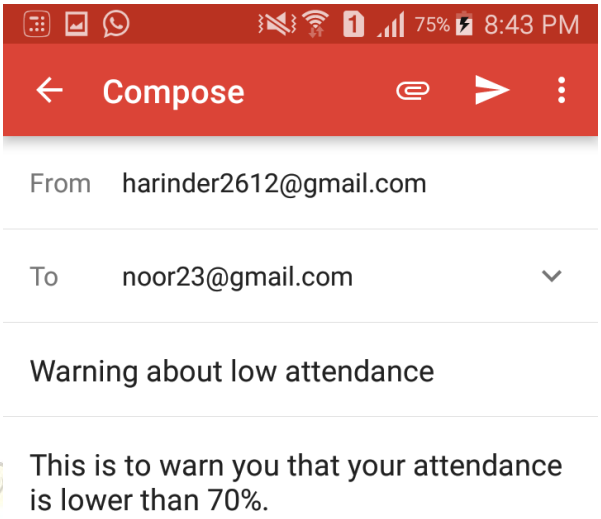


Screenshot 11

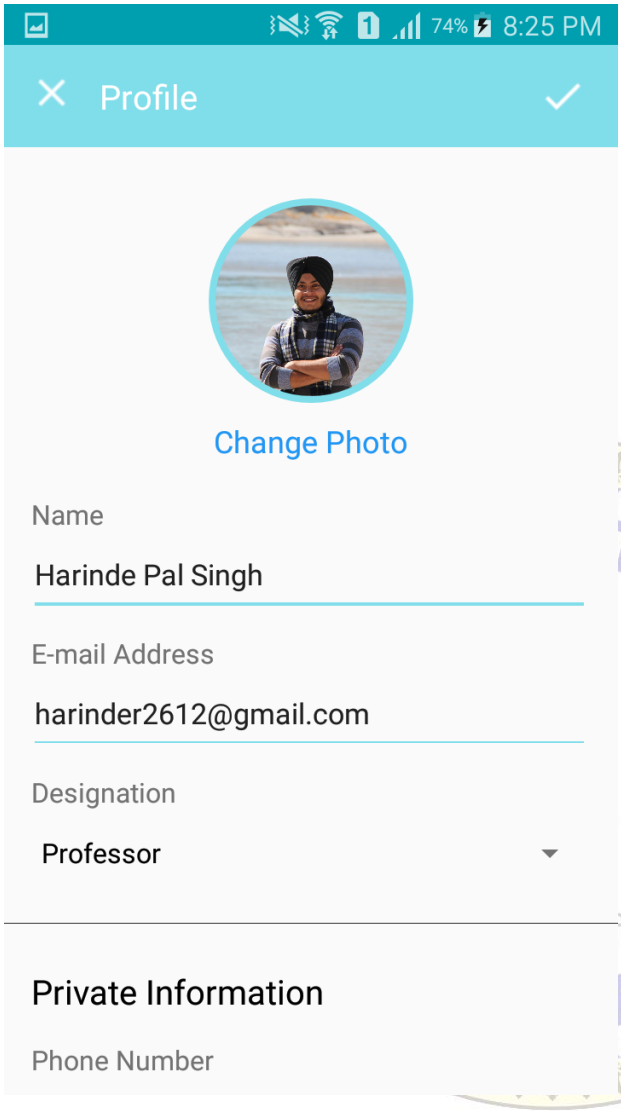
Screenshot 12



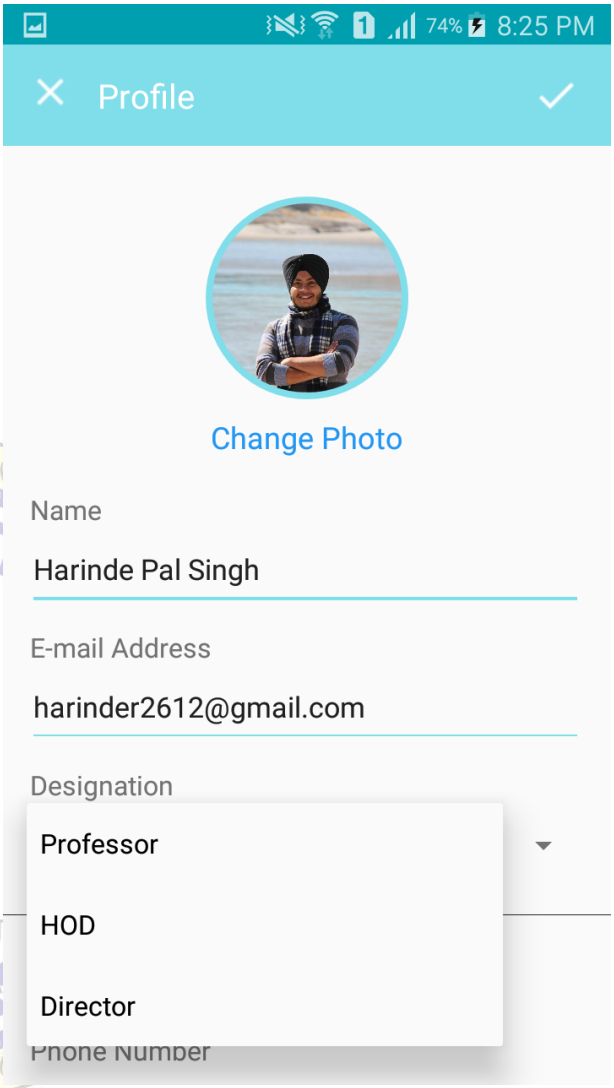
Screenshot 13



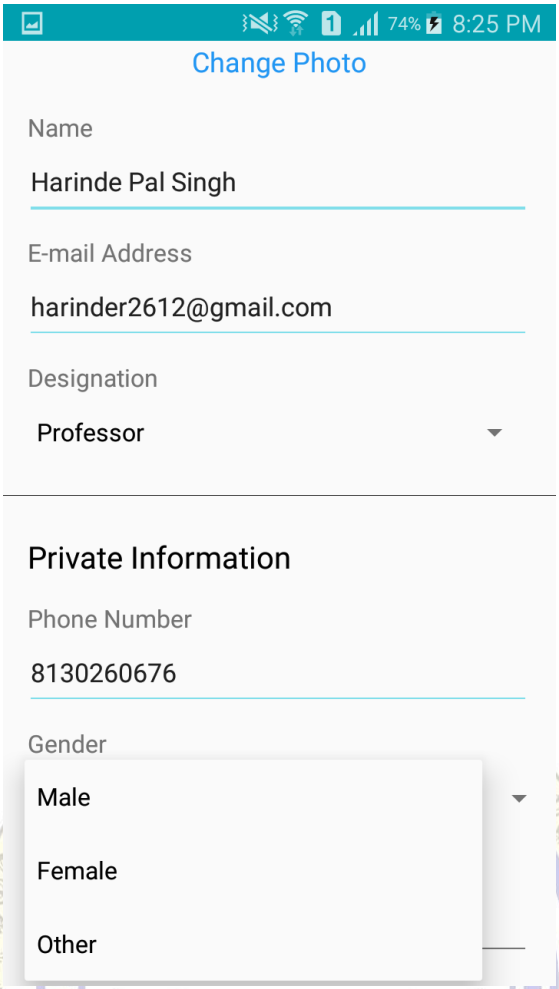
Screenshot 14



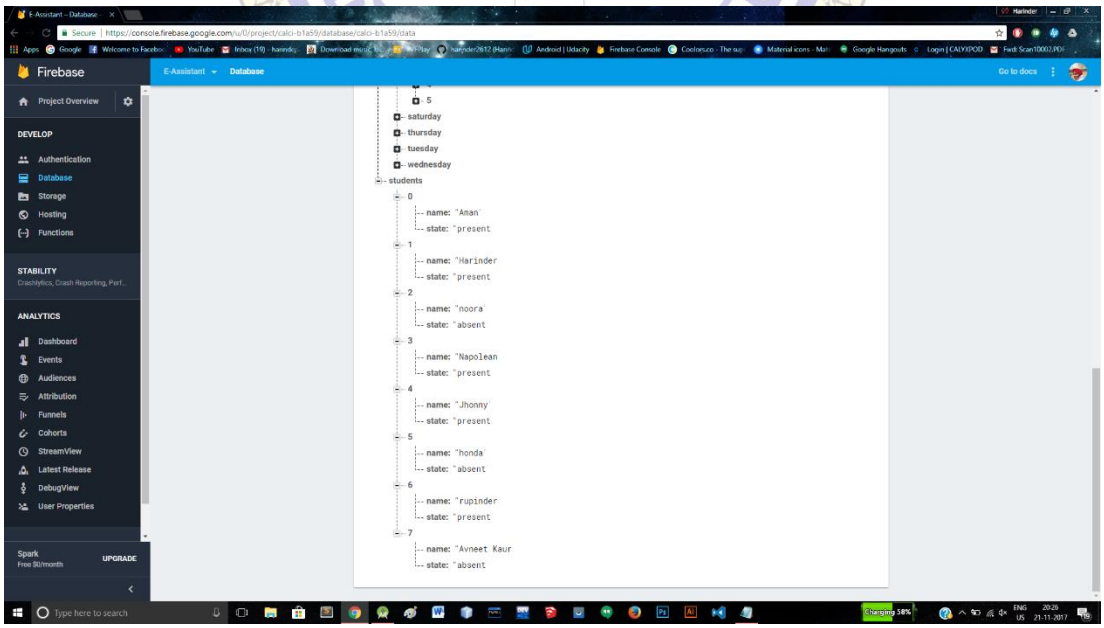
Screenshot 15



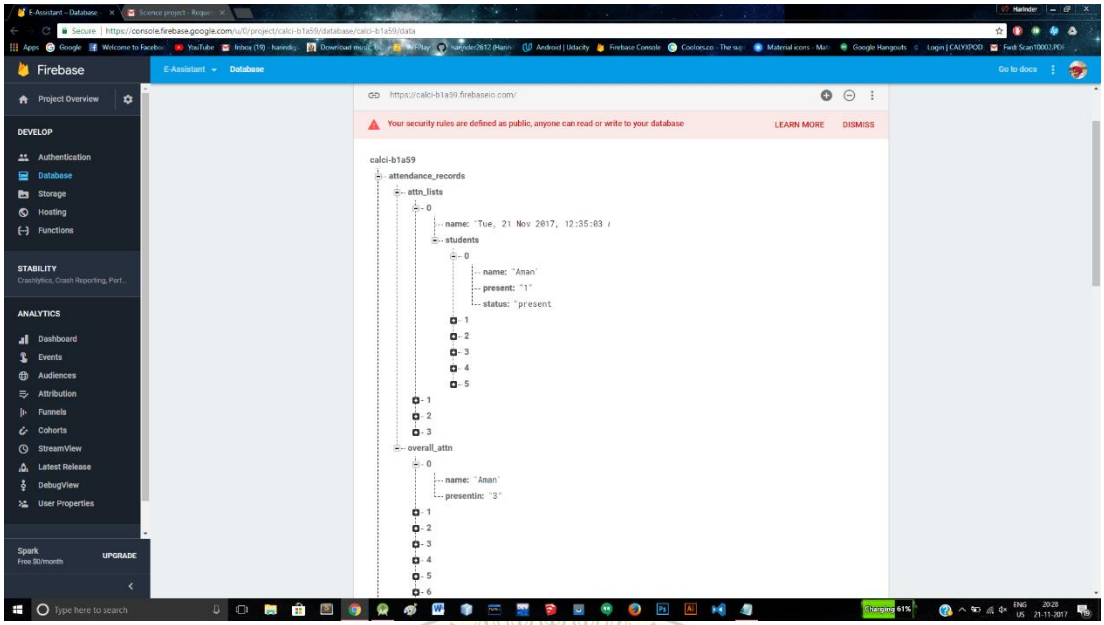
Screenshot 16



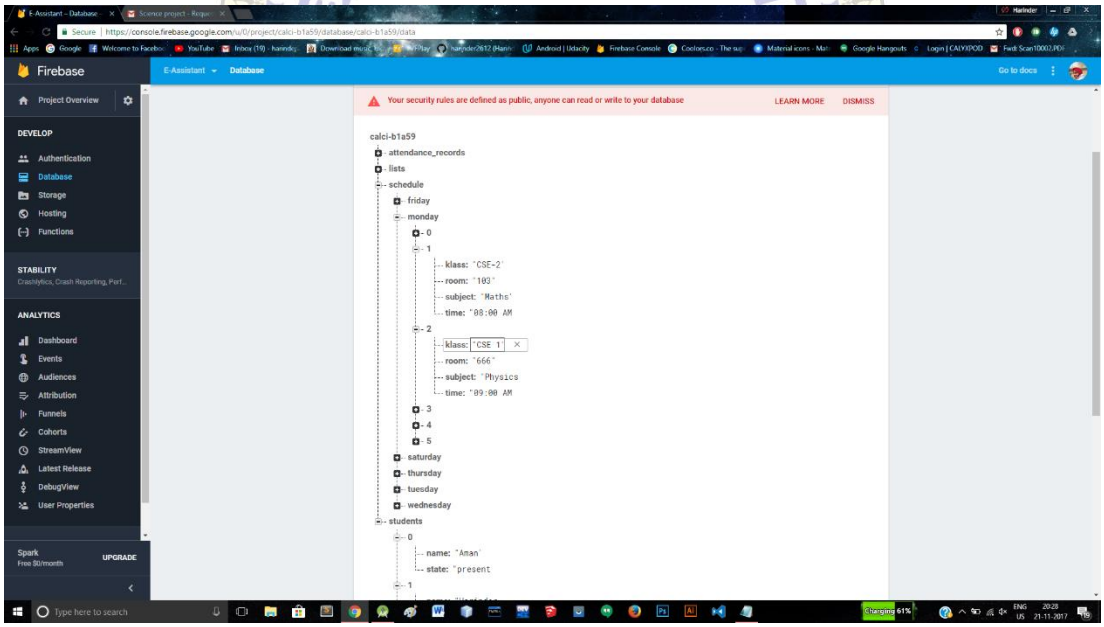
Screenshot 17



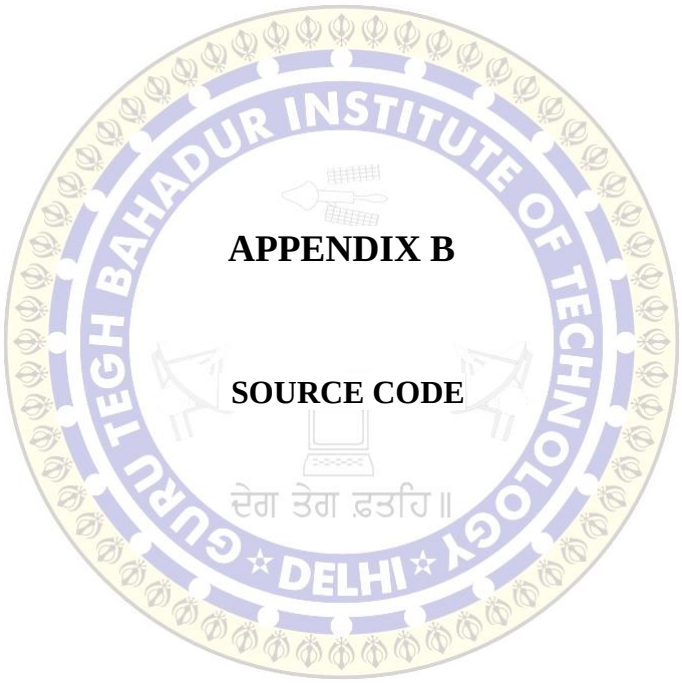
Screenshot 18



Screenshot 19



Screenshot 20



APPENDIX B

SOURCE CODE

lambda_function.py

```
from __future__ import print_function
from firebase import firebase
import json
import copy
import datetime
import pytz
from datetime import datetime

firebase=firebase.FirebaseApplication('https://calci-b1a59.firebaseio.com',None)
first_student=""
students_list=[]
final_list=[]
flag=0
index=0
days=['monday','tuesday','wednesday','thursday','friday','saturday','sunday']

def build_speechlet_response(title,output,reprompt_text,should_end_session):
    return {
    'outputSpeech': {
        'type': 'PlainText',
        'text': output
    },
    'card': {
        'type': 'Simple',
        'title': title,
        'content': output
    },
    'reprompt': {
        'outputSpeech': {
            'type': 'PlainText',
            'text': reprompt_text
        }
    },
    'shouldEndSession': should_end_session
    }

def build_response(session_attributes, speechlet_response):
    return {
        'version':'1.0',
```

```
            'sessionAttributes':session_attributes,
            'response':speechlet_response
            }

def get_help(intent,session):
    card_title="Help"
    speech_output="You can ask me to take attendance of your class."
    should_end_session=False
    return build_response({}, build_speechlet_response(
        card_title, speech_output, speech_output, should_end_session))

def continue_attendance(intent,session):
    card_title="Continue Attendance"
    session_attributes={}
    should_end_session=False
    speech_output=""
    speech=""
    value=str(intent['slots']['presence']['value'])
    print(value)
    if value == "present":
        print('push present to firebase')
        print("Flag: "+str(flag))
        if flag==1:
            result=firebase.patch('/students/0',{'name': first_student, 'state': 'present'})
            speech="Marked present."
        else:
            global index
            result=firebase.patch('/students/'+str(index),{'name': first_student, 'state': 'present'})
            print("Hey")
            print(index)
            speech="Marked present."
    elif value == "absent":
        print('push Absent to firebase')
        print("Flag:"+str(flag))
        if flag==1:
            result=firebase.patch('/students/0',{'name': first_student, 'state': 'absent'})
            speech="Marked absent."
        else:
            global index
            result=firebase.patch('/students/'+str(index),{'name': first_student, 'state': 'absent'})
```

```
            print("Hey")
            print(index)
            speech="Marked absent."
        print(len(students_list))
        if len(students_list)==1:
            should_end_session=True
            session_attributes={}
            speech_output="Attendance Done."
        else:
            students_list.pop(0)
            print(students_list[0])
            speech_output=speech+" "+students_list[0]['name']
            global first_student
            first_student=students_list[0]['name']
            global final_list
            print("Hello Two")
            print(final_list)
            index=final_list.index(students_list[0])
            print("Index defined")
            print(index)
            global flag
            flag=0

        return build_response(session_attributes, build_speechlet_response(
            card_title, speech_output, speech_output,should_end_session))

def handle_session_end_request():
        card_title = "Thank you for ptaking attendance."
        speech_output = "Thank you for taking attendance."
        should_end_session = True
        return build_response({}, build_speechlet_response(
            card_title, speech_output, None, should_end_session))

def schedule_teacher(intent,session):
        card_title="Schedule"
        session_attributes={}
        should_end_session=False
        speech_output=""
        print(intent['slots'])
        print('value' in intent['slots']['days'])
```

```
if('value' in intent['slots']['days']):
    value=str(intent['slots']['days']['value'])
    schedule_result=firebase.get('/schedule',None)
    value=value.lower()
    if(value in schedule_result):
        teacher_classes=schedule_result[value]
        print(len(teacher_classes))
        speech_output="You have "+str(len(teacher_classes))+" classes on "+value+". "
        for x in teacher_classes:
            speech_output+=x['subject']+' with '+x['klass']+' from '+x['time']+'.'+' '
        should_end_session=True
    else:
        speech_output="You do not have any class on this day."
        should_end_session=True
else:
    tz = pytz.timezone('Asia/Kolkata')
    current_time=datetime.now(tz).time()
    current_day_no=datetime.today().weekday()
    current_day=days[current_day_no]
    print(current_time.hour)
    if(current_time.hour>12):
        current_time=current_time.hour-12
    else:
        current_time=current_time.hour
    print("Current Time "+str(current_time))
    schedule_result=firebase.get('/schedule',None)
    classes=schedule_result[current_day]
    hours=[]
    minutes=[]
    for x in classes:
        s=datetime.strptime(x['time'], "%I:%M %p")
        hours.append(s.hour)
        minutes.append(s.minute)
    hours.append(current_time)
    print(hours)
    hours.sort()
    print(hours)
    class_time=""
    m=hours.index(current_time)
    if(m<len(hours)-1):
```

```
            m=m+1
            class_time=str(hours[m])
            for x in classes:
               s=datetime.strptime(x['time'], "%I:%M %p")
               if(str(s.hour)==class_time):
                  speech_output="Your next class is at "+x['time']+' of '+x['subject']+' with '+x['klass']+'
.'
                  should_end_session=True
                  break
         elif(m==len(hours)-1):
            speech_output="No class"+" "+str(current_time)+' .'
            should_end_session=True
      return build_response(session_attributes, build_speechlet_response(
          card_title, speech_output, speech_output, should_end_session))

def make_list(intent,session):
   card_title="Make List"
   session_attributes={}
   should_end_session=True
   speech_output=""
   class_name=intent['slots']['class']['value']
   class_name=class_name.lower()
   list_name=intent['slots']['list']['value']
   students=firebase.get('/lists/'+class_name,None)
   speech_output=list_name+' value for '+students[0]['name']+'is .'
   return build_response(session_attributes, build_speechlet_response(
          card_title, speech_output, speech_output, should_end_session))


def on_intent(intent_request, session):
      """ Called when the user specifies an intent for this skill """
      print("on_intent requestId=" + intent_request['requestId'] +
        ", sessionId=" + session['sessionId'])

      intent = intent_request['intent']
      intent_name = intent_request['intent']['name']

      #Dispatch to your skill's intent handlers
      if intent_name=="StartAttendance":
            return start_attendance(intent,session)
```

```
    if intent_name=="ContinueAttendance":
        return continue_attendance(intent,session)
    if intent_name=="ScheduleIntent":
       return schedule_teacher(intent,session)
    if intent_name=="MakeListIntent":
       return make_list(intent,session)
    elif intent_name=="AMAZON.HelpIntent":
        return get_help(intent,session)
    elif intent_name == "AMAZON.CancelIntent" or intent_name == "AMAZON.StopIntent":
        return handle_session_end_request()
    else:
        raise ValueError("Invalid intent")

def start_attendance(intent,session):
    card_title="Start Attendance"
    session_attributes={}
    should_end_session=False
    speech_output=""
    result = firebase.get('/students', None)
    global students_list
    students_list=result
    print(result[0]['name'])
    student=result[0]['name']
    global first_student
    first_student=student
    global final_list
    final_list=copy.copy(result)
    print("Hello")
    print(final_list)
    speech_output="I am starting attendance. "+student
    print(speech_output)
    global flag
    flag=1
    return build_response(session_attributes, build_speechlet_response(
        card_title, speech_output, speech_output, should_end_session))

def on_session_ended(session_ended_request,session):
    """ Called when the user ends the session.
    Is not called when the skill returns should_end_session=true
    """
```

```
    print("on_session_ended requestId=" + session_ended_request['requestId'] +
        ", sessionId=" + session['sessionId'])
    # add cleanup logic here

def on_session_started(session_started_request, session):
    #Call on session start

    print("on_session_started requestId=" + session_started_request['requestId']
        + ", sessionId=" + session['sessionId'])

def on_launch(launch_request, session):
  """ Called when the user launches the skill without specifying what they
  want
  """
  print("on_launch requestId=" + launch_request['requestId'] +
      ", sessionId=" + session['sessionId'])
  # Dispatch to your skill's launch
  return get_welcome_response()

def get_welcome_response():
    card_title="Welcome to Teacher Helper"
    should_end_session=False
    session_attributes={}
    speech_output="Welcome to Teacher Helper."\
              "You can ask me to take attendance of class."
    reprompt_text=speech_output
    return build_response(session_attributes, build_speechlet_response(
    card_title, speech_output, reprompt_text, should_end_session))



def lambda_handler(event,context):
    print("event.session.application.applicationId="+
        event['session']['application']['applicationId'])

    if event['session']['new']:
        on_session_started({'requestId': event['request']['requestId']},
                    event['session'])

    if event['request']['type']=="LaunchRequest":
```

```
        return on_launch(event['request'], event['session'])
    elif event['request']['type']=="IntentRequest":
        return on_intent(event['request'], event['session'])
    elif event['request']['type']=="SessionEndedRequest":
        return on_session_ended(event['request'], event['session'])
```

attendance.java

```
package com.harinder.e_assistant;

import android.app.ProgressDialog;
import android.os.Build;
import android.support.annotation.RequiresApi;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.view.WindowManager;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.Toast;

import com.google.firebase.database.ChildEventListener;
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.GenericTypeIndicator;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class attendance extends AppCompatActivity {

   ListView attnList;
   EditText studName;
```

```
    ImageView add,refresh,save;
    android.widget.Toolbar toolbar;
    static int totalClasses=-1;
    DatabaseReference myref= FirebaseDatabase.getInstance().getReferenceFromUrl("https://calci-
b1a59.firebaseio.com/");

    @RequiresApi(api = Build.VERSION_CODES.LOLLIPOP)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_attendance);

this.getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_ALW
AYS_HIDDEN);

        attnList= (ListView) findViewById(R.id.attnList);
        refresh= (ImageView) findViewById(R.id.refresh);
        save= (ImageView) findViewById(R.id.save);
        studName= (EditText) findViewById(R.id.studName);
        add= (ImageView) findViewById(R.id.add);
        toolbar= (android.widget.Toolbar) findViewById(R.id.classtitileBar);

        final ProgressDialog pd=new ProgressDialog(this);
        pd.setTitle("Loading...");
        pd.show();


        final ArrayList<studentItem> list= new ArrayList<>();
        final ArrayList<studentItem2> attnRecords= new ArrayList<>();
        final ArrayList<String> ClassesAttended=new ArrayList<>();

        final studentAdapter adapter= new studentAdapter(attendance.this,list);
        attnList.setAdapter(adapter);



        //childeventListener to child of students in firebase
        myref.child("students").addChildEventListener(new ChildEventListener() {
            @Override
            public void onChildAdded(DataSnapshot dataSnapshot, String s) {
```

```
    pd.dismiss();
    Map<String,String> map = (Map<String,String>) dataSnapshot.getValue();
      list.add(new studentItem(map.get("name"),map.get("state")));
       adapter.notifyDataSetChanged();
       attnList.setSelection(adapter.getCount()-1);

   // Toast.makeText(attendance.this, "count: "+count, Toast.LENGTH_SHORT).show();
    System.out.println("OUTPUT:" + map);

}


@Override
public void onChildChanged(DataSnapshot dataSnapshot, String s) {
    Map<String,String> map = (Map<String,String>) dataSnapshot.getValue();
    System.out.println("OUTPUT2:" + map);
    int index=0;
    for(int i=0;i<list.size();i++)
    {
        String name=map.get("name");
        if(list.get(i).getName().equals(name))
        {
            System.out.println(""+name+" "+list.get(i).getName());
            index=i;
        }
    }
    list.get(index).setState(map.get("state"));
    adapter.notifyDataSetChanged();
    attnList.setSelection(index);
}

@Override
public void onChildRemoved(DataSnapshot dataSnapshot) {

}

@Override
public void onChildMoved(DataSnapshot dataSnapshot, String s) {

}
```

```
        @Override
        public void onCancelled(DatabaseError databaseError) {

        }
    });


    myref.child("attendance_records").child("attn_lists").addChildEventListener(new
ChildEventListener() {
        @Override
        public void onChildAdded(DataSnapshot dataSnapshot, String s) {
          totalClasses++;
        }

        @Override
        public void onChildChanged(DataSnapshot dataSnapshot, String s) {

        }

        @Override
        public void onChildRemoved(DataSnapshot dataSnapshot) {

        }

        @Override
        public void onChildMoved(DataSnapshot dataSnapshot, String s) {

        }

        @Override
        public void onCancelled(DatabaseError databaseError) {

        }
    });

    myref.child("attendance_records").child("overall_attn").addChildEventListener(new
ChildEventListener() {
        @Override
```

```
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {
        Map<String,String> map = (Map<String,String>) dataSnapshot.getValue();
        ClassesAttended.add(map.get("presentin"));
    }

    @Override
    public void onChildChanged(DataSnapshot dataSnapshot, String s) {

    }

    @Override
    public void onChildRemoved(DataSnapshot dataSnapshot) {

    }

    @Override
    public void onChildMoved(DataSnapshot dataSnapshot, String s) {

    }

    @Override
    public void onCancelled(DatabaseError databaseError) {

    }
});

//adding new students to the firebase and updating list
add.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        String name=studName.getText().toString();
        if(name.equals(""))
        {}
        else {
            studentItem newStudent = new studentItem(name, "absent");
            DatabaseReference chi = myref.child("students").child(""+list.size());
            chi.setValue(newStudent);

            //adding student to attendance_records
```

```
            DatabaseReference ch =
myref.child("attendance_records").child("overall_attn").child(""+list.size());
            ch.setValue(new overall_attn(name,"0"));


            studName.setText("");
        }
    }
});


//refreshing attendance by marking absent against all students
refresh.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        Toast.makeText(attendance.this, "Attendance Reset", Toast.LENGTH_SHORT).show();
        for(int i=0;i<list.size();i++)
        {
            DatabaseReference chi = myref.child("students").child(""+i).child("state");
            chi.setValue("absent");
        }
    }
});

save.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        // Todays date and time
        SimpleDateFormat formatter1 = new SimpleDateFormat("E, dd MMM yyyy, hh:mm:ss a");
        long day=System.currentTimeMillis();
        final String dateString=formatter1.format(day);

        for(int i=0;i<list.size();i++)
        {
            int num= Integer.parseInt(ClassesAttended.get(i));

            if(list.get(i).getState().equals("present"))
            {
```

```
                num++;
                DatabaseReference chi =
myref.child("attendance_records").child("overall_attn").child(""+i).child("presentin");
                chi.setValue(""+num);
                ClassesAttended.set(i,""+num);
              }
              attnRecords.add(i,new studentItem2(list.get(i).getName(),list.get(i).getState(),""+num));


            }

            Toast.makeText(attendance.this, "Attendance Saved", Toast.LENGTH_SHORT).show();

            DatabaseReference ch =
myref.child("attendance_records").child("attn_lists").child(""+(totalClasses+1)).child("name");
            ch.setValue(dateString);

            for(int i=0;i<list.size();i++)
            {
              DatabaseReference chi =
myref.child("attendance_records").child("attn_lists").child(""+(totalClasses+1)).child("students").chil
d(""+i);
              chi.setValue(attnRecords.get(i));
            }
          }
      });


      toolbar.setNavigationOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
          onBackPressed();
          }

}
```

schedule.java

```
package com.harinder.e_assistant;

import android.app.ProgressDialog;
import android.content.DialogInterface;
```

```
import android.os.Build;
import android.support.annotation.RequiresApi;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.InputType;
import android.view.View;
import android.widget.AdapterView;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import com.google.firebase.database.ChildEventListener;
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 java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Map;

public class schedule extends AppCompatActivity {

    ListView scheduleList;
    TextView tuesday,monday,wednesday,thursday,friday,saturday;
    Button addPeriod;
    DatabaseReference myref= FirebaseDatabase.getInstance().getReferenceFromUrl("https://calci-
b1a59.firebaseio.com/");
    static int flag=0;

    android.widget.Toolbar toolbar;
    @RequiresApi(api = Build.VERSION_CODES.LOLLIPOP)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_schedule);
```

```
scheduleList= (ListView) findViewById(R.id.scheduleList);
tuesday= (TextView) findViewById(R.id.tuesday);
monday= (TextView) findViewById(R.id.monday);
wednesday= (TextView) findViewById(R.id.wednesday);
thursday= (TextView) findViewById(R.id.thursday);
friday= (TextView) findViewById(R.id.friday);
saturday= (TextView) findViewById(R.id.saturday);
addPeriod= (Button) findViewById(R.id.addPeriod);
toolbar= (android.widget.Toolbar) findViewById(R.id.scheduleBar);
toolbar.setNavigationOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View view) {
      onBackPressed();
   }
});

final ProgressDialog pd=new ProgressDialog(this);
pd.setTitle("Loading...");
pd.show();

SimpleDateFormat formatter1 = new SimpleDateFormat("E");
long day=System.currentTimeMillis();
String dateString=formatter1.format(day);



//Arraylist for 6 days of week
final ArrayList<scheduleItem> mon= new ArrayList<>();
final ArrayList<scheduleItem> tue= new ArrayList<>();
final ArrayList<scheduleItem> wed= new ArrayList<>();
final ArrayList<scheduleItem> thu= new ArrayList<>();
final ArrayList<scheduleItem> fri= new ArrayList<>();
final ArrayList<scheduleItem> sat= new ArrayList<>();

//6 adapters for 6 lists, one fr each day
final scheduleAdapter MonAdapter;
final scheduleAdapter TueAdapter;
final scheduleAdapter WedAdapter;
final scheduleAdapter ThuAdapter;
final scheduleAdapter FriAdapter;
```

```
        final scheduleAdapter SatAdapter;

        MonAdapter = new scheduleAdapter(schedule.this,mon);
        TueAdapter = new scheduleAdapter(schedule.this,tue);
        WedAdapter = new scheduleAdapter(schedule.this,wed);
        ThuAdapter = new scheduleAdapter(schedule.this,thu);
        FriAdapter = new scheduleAdapter(schedule.this,fri);
        SatAdapter = new scheduleAdapter(schedule.this,sat);


        //Fetching data of Monday from firebase
        myref.child("schedule").child("monday").addChildEventListener(new ChildEventListener() {
            @Override
            public void onChildAdded(DataSnapshot dataSnapshot, String s) {
                pd.dismiss();
                Map<String,String> map = (Map<String,String>) dataSnapshot.getValue();
                mon.add(new
scheduleItem(map.get("subject"),map.get("room"),map.get("klass"),map.get("time")));
                MonAdapter.notifyDataSetChanged();
                System.out.println("OUTPUT Dekh:" + map);
            }

            @Override
            public void onChildChanged(DataSnapshot dataSnapshot, String s) {
                Map<String,String> map = (Map<String,String>) dataSnapshot.getValue();
                int index=0;
                for(int i=0;i<mon.size();i++)
                {
                    String time=map.get("time");
                    if(mon.get(i).getTime().equals(time))
                    {
                        System.out.println(""+time+" "+mon.get(i).getTime());
                        index=i;
                    }
                }
                mon.set(index,new
scheduleItem(map.get("subject"),map.get("room"),map.get("klass"),map.get("time")));
                MonAdapter.notifyDataSetChanged();
                scheduleList.setSelection(index);
            }
```

```
        @Override
        public void onChildRemoved(DataSnapshot dataSnapshot) {

        }

        @Override
        public void onChildMoved(DataSnapshot dataSnapshot, String s) {

        }

        @Override
        public void onCancelled(DatabaseError databaseError) {

        }
    });


    //fetching data of Tuesday from firebase
    myref.child("schedule").child("tuesday").addChildEventListener(new ChildEventListener() {
        @Override
        public void onChildAdded(DataSnapshot dataSnapshot, String s) {
            pd.dismiss();
            Map<String,String> map = (Map<String,String>) dataSnapshot.getValue();
            tue.add(new
scheduleItem(map.get("subject"),map.get("room"),map.get("klass"),map.get("time")));
            TueAdapter.notifyDataSetChanged();
        }

        @Override
        public void onChildChanged(DataSnapshot dataSnapshot, String s) {
            Map<String,String> map = (Map<String,String>) dataSnapshot.getValue();
            int index=0;
            for(int i=0;i<tue.size();i++)
            {
                String time=map.get("time");
                if(tue.get(i).getTime().equals(time))
                {
                    System.out.println(""+time+" "+tue.get(i).getTime());
                    index=i;
```

```
            }
        }
        tue.set(index,new
scheduleItem(map.get("subject"),map.get("room"),map.get("klass"),map.get("time")));
        TueAdapter.notifyDataSetChanged();
        scheduleList.setSelection(index);
    }

    @Override
    public void onChildRemoved(DataSnapshot dataSnapshot) {

    }

    @Override
    public void onChildMoved(DataSnapshot dataSnapshot, String s) {

    }

    @Override
    public void onCancelled(DatabaseError databaseError) {

    }
});


//fetching data of Wednesday from firebase
myref.child("schedule").child("wednesday").addChildEventListener(new ChildEventListener() {
    @Override
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {
        pd.dismiss();
        Map<String,String> map = (Map<String,String>) dataSnapshot.getValue();
        wed.add(new
scheduleItem(map.get("subject"),map.get("room"),map.get("klass"),map.get("time")));
        WedAdapter.notifyDataSetChanged();
    }

    @Override
    public void onChildChanged(DataSnapshot dataSnapshot, String s) {
        Map<String,String> map = (Map<String,String>) dataSnapshot.getValue();
        System.out.println("OUTPUT Dekh Na:" + map);
```

```
            int index=0;
            for(int i=0;i<wed.size();i++)
            {
               String time=map.get("time");
               if(wed.get(i).getTime().equals(time))
               {
                  System.out.println(""+time+" "+wed.get(i).getTime());
                  index=i;
               }
            }
            wed.set(index,new
scheduleItem(map.get("subject"),map.get("room"),map.get("klass"),map.get("time")));
            WedAdapter.notifyDataSetChanged();
            scheduleList.setSelection(index);
         }

         @Override
         public void onChildRemoved(DataSnapshot dataSnapshot) {

         }

         @Override
         public void onChildMoved(DataSnapshot dataSnapshot, String s) {

         }

         @Override
         public void onCancelled(DatabaseError databaseError) {

         }
      });


      //fetching data of Thusday from firebase
      myref.child("schedule").child("thursday").addChildEventListener(new ChildEventListener() {
         @Override
         public void onChildAdded(DataSnapshot dataSnapshot, String s) {
            pd.dismiss();
            Map<String,String> map = (Map<String,String>) dataSnapshot.getValue();
```

```
            thu.add(new
scheduleItem(map.get("subject"),map.get("room"),map.get("klass"),map.get("time")));
            ThuAdapter.notifyDataSetChanged();
        }

        @Override
        public void onChildChanged(DataSnapshot dataSnapshot, String s) {
            Map<String,String> map = (Map<String,String>) dataSnapshot.getValue();
            System.out.println("OUTPUT Dekh Na:" + map);
            int index=0;
            for(int i=0;i<thu.size();i++)
            {
                String time=map.get("time");
                if(thu.get(i).getTime().equals(time))
                {
                    System.out.println(""+time+" "+thu.get(i).getTime());
                    index=i;
                }
            }
            thu.set(index,new
scheduleItem(map.get("subject"),map.get("room"),map.get("klass"),map.get("time")));
            ThuAdapter.notifyDataSetChanged();
            scheduleList.setSelection(index);
        }

        @Override
        public void onChildRemoved(DataSnapshot dataSnapshot) {

        }

        @Override
        public void onChildMoved(DataSnapshot dataSnapshot, String s) {

        }

        @Override
        public void onCancelled(DatabaseError databaseError) {

        }
    });
```

```
        //fetching data of Friday from firebase
        myref.child("schedule").child("friday").addChildEventListener(new ChildEventListener() {
            @Override
            public void onChildAdded(DataSnapshot dataSnapshot, String s) {
                pd.dismiss();
                Map<String,String> map = (Map<String,String>) dataSnapshot.getValue();
                fri.add(new
scheduleItem(map.get("subject"),map.get("room"),map.get("klass"),map.get("time")));
                FriAdapter.notifyDataSetChanged();
            }

            @Override
            public void onChildChanged(DataSnapshot dataSnapshot, String s) {
                Map<String,String> map = (Map<String,String>) dataSnapshot.getValue();
                System.out.println("OUTPUT Dekh Na:" + map);
                int index=0;
                for(int i=0;i<fri.size();i++)
                {
                    String time=map.get("time");
                    if(fri.get(i).getTime().equals(time))
                    {
                        System.out.println(""+time+" "+fri.get(i).getTime());
                        index=i;
                    }
                }
                fri.set(index,new
scheduleItem(map.get("subject"),map.get("room"),map.get("klass"),map.get("time")));
                FriAdapter.notifyDataSetChanged();
                scheduleList.setSelection(index);
            }

            @Override
            public void onChildRemoved(DataSnapshot dataSnapshot) {

            }

            @Override
```

```
        public void onChildMoved(DataSnapshot dataSnapshot, String s) {

        }

        @Override
        public void onCancelled(DatabaseError databaseError) {

        }
    });



    //fetching data of Saturday from firebase
    myref.child("schedule").child("saturday").addChildEventListener(new ChildEventListener() {
        @Override
        public void onChildAdded(DataSnapshot dataSnapshot, String s) {
            pd.dismiss();
            Map<String,String> map = (Map<String,String>) dataSnapshot.getValue();
            sat.add(new
scheduleItem(map.get("subject"),map.get("room"),map.get("klass"),map.get("time")));
            SatAdapter.notifyDataSetChanged();
        }

        @Override
        public void onChildChanged(DataSnapshot dataSnapshot, String s) {
            Map<String,String> map = (Map<String,String>) dataSnapshot.getValue();
            System.out.println("OUTPUT Dekh Na:" + map);
            int index=0;
            for(int i=0;i<sat.size();i++)
            {
                String time=map.get("time");
                if(sat.get(i).getTime().equals(time))
                {
                    System.out.println(""+time+" "+sat.get(i).getTime());
                    index=i;
                }
            }
            sat.set(index,new
scheduleItem(map.get("subject"),map.get("room"),map.get("klass"),map.get("time")));
            SatAdapter.notifyDataSetChanged();
```

```
        scheduleList.setSelection(index);
    }

    @Override
    public void onChildRemoved(DataSnapshot dataSnapshot) {

    }

    @Override
    public void onChildMoved(DataSnapshot dataSnapshot, String s) {

    }

    @Override
    public void onCancelled(DatabaseError databaseError) {

    }
});



monday.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        scheduleList.setAdapter(MonAdapter);
        toolbar.setTitle("Schedule "+"("+"MON"+")");
        flag=1;
    }
});

tuesday.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        scheduleList.setAdapter(TueAdapter);
        toolbar.setTitle("Schedule "+"("+"TUE"+")");
        flag=2;
    }
});

wednesday.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
    public void onClick(View view) {
        scheduleList.setAdapter(WedAdapter);
        toolbar.setTitle("Schedule "+"("+"WED"+")");
        flag=3;
    }
});

thursday.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        scheduleList.setAdapter(ThuAdapter);
        toolbar.setTitle("Schedule "+"("+"THU"+")");
        flag=4;
    }
});

friday.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        scheduleList.setAdapter(FriAdapter);
        toolbar.setTitle("Schedule "+"("+"FRI"+")");
        flag=5;
    }
});

saturday.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        scheduleList.setAdapter(SatAdapter);
        toolbar.setTitle("Schedule "+"("+"SAT"+")");
        flag=6;
    }
});


//To edit the existing periods from the lists and also update on firebase
scheduleList.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {
    @Override
```

```
    public boolean onItemLongClick(AdapterView<?> adapterView, View view, final int i, long l)
{
        Toast.makeText(schedule.this, ""+i, Toast.LENGTH_SHORT).show();
        AlertDialog.Builder builder = new AlertDialog.Builder(schedule.this);
        builder.setTitle("Edit");

        LinearLayout layout = new LinearLayout(schedule.this);
        layout.setOrientation(LinearLayout.VERTICAL);

        // Set up the input
        final EditText room = new EditText(schedule.this);
        final EditText subject= new EditText(schedule.this);
        final EditText time= new EditText(schedule.this);
        final EditText klass= new EditText(schedule.this);

        room.setHint(" Enter Room No.");
        subject.setHint(" Enter Subject Name");
        time.setHint(" Enter Time ");
        klass.setHint(" Enter Class Name");

        room.setInputType(InputType.TYPE_CLASS_TEXT);
        subject.setInputType(InputType.TYPE_CLASS_TEXT);
        time.setInputType(InputType.TYPE_CLASS_TEXT);
        klass.setInputType(InputType.TYPE_CLASS_TEXT);

        layout.addView(room);
        layout.addView(subject);
        layout.addView(time);
        layout.addView(klass);

        builder.setView(layout);

        // Set up the buttons
        builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(schedule.this, "Period Edited", Toast.LENGTH_SHORT).show();

                switch (flag)
                {
```

```
                case 1:{
                    DatabaseReference chi = myref.child("schedule").child("monday").child(""+i);
                    chi.setValue(new
scheduleItem(subject.getText().toString(),room.getText().toString(),klass.getText().toString(),time.get
Text().toString()));
                }break;

                case 2:{
                    DatabaseReference chi = myref.child("schedule").child("tuesday").child(""+i);
                    chi.setValue(new
scheduleItem(subject.getText().toString(),room.getText().toString(),klass.getText().toString(),time.get
Text().toString()));
                }break;

                case 3:{
                    DatabaseReference chi = myref.child("schedule").child("wednesday").child(""+i);
                    chi.setValue(new
scheduleItem(subject.getText().toString(),room.getText().toString(),klass.getText().toString(),time.get
Text().toString()));
                }break;

                case 4:{
                    DatabaseReference chi = myref.child("schedule").child("thursday").child(""+i);
                    chi.setValue(new
scheduleItem(subject.getText().toString(),room.getText().toString(),klass.getText().toString(),time.get
Text().toString()));
                }break;

                case 5:{
                    DatabaseReference chi = myref.child("schedule").child("friday").child(""+i);
                    chi.setValue(new
scheduleItem(subject.getText().toString(),room.getText().toString(),klass.getText().toString(),time.get
Text().toString()));
                }break;

                case 6:{
                    DatabaseReference chi = myref.child("schedule").child("saturday").child(""+i);
                    chi.setValue(new
scheduleItem(subject.getText().toString(),room.getText().toString(),klass.getText().toString(),time.get
Text().toString()));
```

```
                }break;
            }

        }
    });
    builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.cancel();
        }
    });

    builder.show();
    return false;
  }
});


//To add new features to lists as well as on firebase
addPeriod.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View view) {

    AlertDialog.Builder builder = new AlertDialog.Builder(schedule.this);
    builder.setTitle("Add");

    LinearLayout layout = new LinearLayout(schedule.this);
    layout.setOrientation(LinearLayout.VERTICAL);

    // Set up the input
    final EditText room = new EditText(schedule.this);
    final EditText subject= new EditText(schedule.this);
    final EditText time= new EditText(schedule.this);
    final EditText klass= new EditText(schedule.this);

    room.setHint(" Enter Room No.");
    subject.setHint(" Enter Subject Name");
    time.setHint(" Enter Time ");
    klass.setHint(" Enter Class Name");
```

```
            room.setInputType(InputType.TYPE_CLASS_TEXT);
            subject.setInputType(InputType.TYPE_CLASS_TEXT);
            time.setInputType(InputType.TYPE_CLASS_TEXT);
            klass.setInputType(InputType.TYPE_CLASS_TEXT);

            layout.addView(room);
            layout.addView(subject);
            layout.addView(time);
            layout.addView(klass);

            builder.setView(layout);

            // Set up the buttons
            builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
               @Override
               public void onClick(DialogInterface dialog, int which) {
                  Toast.makeText(schedule.this, "New Period Added",
Toast.LENGTH_SHORT).show();

                  switch (flag)
                  {
                     case 1:{
                        DatabaseReference chi =
myref.child("schedule").child("monday").child(""+mon.size());
                        chi.setValue(new
scheduleItem(subject.getText().toString(),room.getText().toString(),klass.getText().toString(),time.get
Text().toString()));
                     }break;

                     case 2:{
                        DatabaseReference chi =
myref.child("schedule").child("tuesday").child(""+tue.size());
                        chi.setValue(new
scheduleItem(subject.getText().toString(),room.getText().toString(),klass.getText().toString(),time.get
Text().toString()));
                     }break;

                     case 3:{
                        DatabaseReference chi =
myref.child("schedule").child("wednesday").child(""+wed.size());
```

```
                    chi.setValue(new
scheduleItem(subject.getText().toString(),room.getText().toString(),klass.getText().toString(),time.get
Text().toString()));
                }break;

                case 4:{
                    DatabaseReference chi =
myref.child("schedule").child("thursday").child(""+thu.size());
                    chi.setValue(new
scheduleItem(subject.getText().toString(),room.getText().toString(),klass.getText().toString(),time.get
Text().toString()));
                }break;

                case 5:{
                    DatabaseReference chi =
myref.child("schedule").child("friday").child(""+fri.size());
                    chi.setValue(new
scheduleItem(subject.getText().toString(),room.getText().toString(),klass.getText().toString(),time.get
Text().toString()));
                }break;

                case 6:{
                    DatabaseReference chi =
myref.child("schedule").child("saturday").child(""+sat.size());
                    chi.setValue(new
scheduleItem(subject.getText().toString(),room.getText().toString(),klass.getText().toString(),time.get
Text().toString()));
                }break;
            }

        }
    });

    builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.cancel();
        }
    });
```

```
            builder.show();
        }
    });


    switch (dateString)
    {
        case "Mon":monday.performClick();
            break;
        case "Tue":tuesday.performClick();
            break;
        case "Wed":wednesday.performClick();
            break;
        case "Thu":thursday.performClick();
            break;
        case "Fri":friday.performClick();
            break;
        case "Sat":saturday.performClick();
            break;
        case "Sun":monday.performClick();
            break;
    }
  }


}
```

studentAdapter.java

```
package com.harinder.e_assistant;


import android.content.Context;
import android.support.annotation.LayoutRes;
import android.support.annotation.NonNull;
import android.support.v4.content.ContextCompat;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
```

```
import android.widget.TextView;

import java.util.ArrayList;
import java.util.List;

public class studentAdapter extends ArrayAdapter<studentItem>{


    public studentAdapter(Context context,List<studentItem> objects) {
        super(context,0, objects);
    }

    @Override
    public View getView(final int position, View convertView, ViewGroup parent) {

        View view= convertView;
        if(view==null)
        {
            view=LayoutInflater.from(getContext()).inflate(R.layout.attendance_item, parent, false);

        }

       studentItem currentStudentItem=getItem(position);
        TextView name=(TextView) view.findViewById(R.id.studname);
        ImageView state= (ImageView) view.findViewById(R.id.state);

        name.setText(currentStudentItem.getName());
        if(currentStudentItem.getState().equals("present"))
        {
            state.setImageResource(R.drawable.greendot);
        }
        else
        {
            state.setImageResource(R.drawable.reddot);
        }

        return view;
    }
}
```

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
    android:orientation="vertical"
    tools:context="com.harinder.e_assistant.MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="0.2"
        android:background="#4dd0e1"
        android:orientation="horizontal">

        <de.hdodenhof.circleimageview.CircleImageView
            android:id="@+id/profile"
            android:layout_width="75dp"
            android:layout_height="75dp"
            android:layout_gravity="center"
            android:layout_marginLeft="16dp"
            android:src="@drawable/harry2"
            app:civ_border_color="#ffffff"
            app:civ_border_width="3dp"
            android:background="?attr/selectableItemBackgroundBorderless"
            android:clickable="true"/>

        <LinearLayout
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:orientation="vertical">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="0dp"
                android:layout_weight="1"
```

```
        android:orientation="horizontal">

        <TextView
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="16dp"
            android:layout_weight="1"
            android:gravity="bottom"
            android:text="Prof."
            android:textSize="36sp" />

        <Switch
            android:id="@+id/switch1"
            android:layout_width="wrap_content"
            android:layout_height="50dp"
            android:layout_gravity="right"
            android:layout_marginRight="6dp" />

    </LinearLayout>


    <TextView
        android:layout_width="match_parent"
        android:layout_height="30dp"
        android:layout_marginBottom="6dp"
        android:layout_marginLeft="16dp"
        android:layout_marginRight="10dp"
        android:gravity="bottom"
        android:text="Harinder Pal Singh"
        android:textSize="26sp" />


  </LinearLayout>

</LinearLayout>

<LinearLayout
  android:layout_width="match_parent"
  android:layout_height="0dp"
  android:layout_weight="1"
```

```
android:orientation="vertical">

<LinearLayout
   android:layout_width="match_parent"
   android:layout_height="0dp"
   android:layout_weight="1"
   android:orientation="horizontal">

   <LinearLayout
      android:id="@+id/schedule"
      android:layout_width="0dp"
      android:layout_height="match_parent"
      android:layout_weight="0.33"
      android:background="?attr/selectableItemBackgroundBorderless"
      android:clickable="true"
      android:orientation="vertical">

      <ImageView
         android:layout_width="match_parent"
         android:layout_height="0dp"
         android:layout_weight="1"
         android:paddingBottom="10dp"
         android:paddingLeft="16dp"
         android:paddingRight="16dp"
         android:paddingTop="60dp"
         android:src="@drawable/schedules" />

      <TextView
         android:layout_width="match_parent"
         android:layout_height="0dp"
         android:layout_weight="0.3"
         android:gravity="center"
         android:text="Schedule"
         android:textSize="20sp" />

   </LinearLayout>

   <View
      android:layout_width="1dp"
      android:layout_height="match_parent"
```

```
    android:background="@drawable/fadedline"></View>

<LinearLayout
    android:id="@+id/attendance"
    android:layout_width="0dp"
    android:layout_height="match_parent"
    android:layout_weight="0.33"
    android:background="?attr/selectableItemBackgroundBorderless"
    android:clickable="true"
    android:orientation="vertical">

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:paddingTop="60dp"
        android:paddingLeft="10dp"
        android:paddingRight="10dp"
        android:src="@drawable/attendancep" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="0.3"
        android:gravity="center"
        android:text="Attendance"
        android:textSize="20sp" />

</LinearLayout>

<View
    android:layout_width="1dp"
    android:layout_height="match_parent"
    android:background="@drawable/fadedline">

</View>

<LinearLayout
    android:layout_width="0dp"
    android:layout_height="match_parent"
```

```
        android:layout_weight="0.33"
        android:background="?attr/selectableItemBackgroundBorderless"
        android:clickable="true"
        android:orientation="vertical">

        <ImageView
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="1"
            android:paddingBottom="6dp"
            android:paddingLeft="6dp"
            android:paddingRight="6dp"
            android:paddingTop="66dp"
            android:src="@drawable/addlist" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="0.3"
            android:gravity="center"
            android:text="Lists"
            android:textSize="20sp" />

    </LinearLayout>


</LinearLayout>

<View
    android:layout_width="match_parent"
    android:layout_height="1dp"
    android:background="@drawable/fadedline2">

</View>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:orientation="horizontal">
```

```
<LinearLayout
    android:layout_width="0dp"
    android:layout_height="match_parent"
    android:layout_weight="0.33"
    android:background="?attr/selectableItemBackgroundBorderless"
    android:clickable="true"
    android:orientation="vertical">

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:src="@drawable/announce" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="0.3"
        android:gravity="center_horizontal"
        android:text="Announce"
        android:textSize="20sp" />

</LinearLayout>

<View
    android:layout_width="1dp"
    android:layout_height="match_parent"
    android:background="@drawable/fadedline3"></View>

<LinearLayout
    android:layout_width="0dp"
    android:layout_height="match_parent"
    android:layout_weight="0.33"
    android:background="?attr/selectableItemBackgroundBorderless"
    android:clickable="true"
    android:orientation="vertical"
    android:id="@+id/database">

    <ImageView
```

```
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:padding="20dp"
        android:paddingLeft="14dp"
        android:paddingRight="14dp"
        android:src="@drawable/databasee" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="0.3"
        android:gravity="center_horizontal"
        android:text="Database"
        android:textSize="20sp" />

</LinearLayout>

<View
    android:layout_width="1dp"
    android:layout_height="match_parent"
    android:background="@drawable/fadedline3">

</View>

<LinearLayout
    android:layout_width="0dp"
    android:layout_height="match_parent"
    android:layout_weight="0.33"
    android:background="?attr/selectableItemBackgroundBorderless"
    android:clickable="true"
    android:orientation="vertical">

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:padding="16dp"
        android:src="@drawable/lists" />
```

```
        <TextView
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="0.3"
            android:gravity="center_horizontal"
            android:text="Records"
            android:textSize="20sp" />

    </LinearLayout>
  </LinearLayout>
</LinearLayout>

<LinearLayout
  android:layout_width="match_parent"
  android:layout_height="0dp"
  android:layout_weight="0.1"
  android:background="#88ffff"
  android:orientation="horizontal">

  <TextView
    android:layout_width="0dp"
    android:layout_height="match_parent"
    android:layout_weight="0.4"
    android:gravity="center"
    android:text="105/E"
    android:textSize="18sp" />

  <TextView
    android:id="@+id/subject"
    android:layout_width="0dp"
    android:layout_height="match_parent"
    android:layout_weight="1"
    android:gravity="center"
    android:text="Cloud Computing"
    android:textSize="18sp" />

  <TextView

    android:layout_width="0dp"
    android:layout_height="match_parent"
    android:layout_weight="0.5"
```

```
        android:gravity="center"
        android:text="10:30 AM"
        android:textSize="18sp" />

    </LinearLayout>


</LinearLayout>
```

activity_schedule.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    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"
    android:orientation="vertical"
    tools:context="com.harinder.e_assistant.schedule">

    <Toolbar
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:id="@+id/scheduleBar"
        android:background="@color/colorPrimary"
        android:title="Schedule"
        android:titleTextColor="#ffffff"
        android:navigationIcon="@drawable/backbutton">

    </Toolbar>

    <HorizontalScrollView
        android:layout_width="wrap_content"
        android:layout_height="60dp"
        android:scrollbars="vertical"
        android:background="#ECEFF1">

        <LinearLayout
```

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="horizontal"
android:layout_gravity="center_vertical">


<android.support.v7.widget.CardView
   android:layout_width="150dp"
   android:layout_height="match_parent"
   android:layout_gravity="center"
   app:cardCornerRadius="5dp"
   app:cardElevation="8dp"
   android:layout_marginRight="6dp"
   android:layout_marginLeft="6dp">

   <TextView
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:layout_margin="6dp"
      android:gravity="center"
      android:layout_gravity="center"
      android:text="MON"
      android:id="@+id/monday"
      android:textSize="30sp"
      android:textStyle="bold"
      android:clickable="true"
      android:background="?attr/selectableItemBackgroundBorderless"/>

</android.support.v7.widget.CardView>

<android.support.v7.widget.CardView
   android:layout_width="150dp"
   android:layout_height="match_parent"
   android:layout_gravity="center"
   app:cardCornerRadius="5dp"
   app:cardElevation="8dp"
   android:layout_marginRight="6dp"
   android:layout_marginLeft="6dp">
```

```
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="6dp"
        android:gravity="center"
        android:layout_gravity="center"
        android:id="@+id/tuesday"
        android:text="TUE"
        android:textSize="30sp"
        android:textStyle="bold"
        android:clickable="true"
        android:background="?attr/selectableItemBackgroundBorderless"/>
</android.support.v7.widget.CardView>

<android.support.v7.widget.CardView
    android:layout_width="150dp"
    android:layout_height="match_parent"
    android:layout_gravity="center"
    app:cardCornerRadius="5dp"
    app:cardElevation="8dp"
    android:layout_marginRight="6dp"
    android:layout_marginLeft="6dp">


    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="6dp"
        android:gravity="center"
        android:layout_gravity="center"
        android:text="WED"
        android:id="@+id/wednesday"
        android:textSize="30sp"
        android:textStyle="bold"
        android:clickable="true"
        android:background="?attr/selectableItemBackgroundBorderless"/>

</android.support.v7.widget.CardView>

<android.support.v7.widget.CardView
```

```
    android:layout_width="150dp"
    android:layout_height="match_parent"
    android:layout_gravity="center"
    app:cardCornerRadius="5dp"
    app:cardElevation="8dp"
    android:layout_marginRight="6dp"
    android:layout_marginLeft="6dp">


    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="6dp"
        android:gravity="center"
        android:layout_gravity="center"
        android:text="THU"
        android:id="@+id/thursday"
        android:textSize="30sp"
        android:textStyle="bold"
        android:clickable="true"
        android:background="?attr/selectableItemBackgroundBorderless"/>

</android.support.v7.widget.CardView>

<android.support.v7.widget.CardView
    android:layout_width="150dp"
    android:layout_height="match_parent"
    android:layout_gravity="center"
    app:cardCornerRadius="5dp"
    app:cardElevation="8dp"
    android:layout_marginRight="6dp"
    android:layout_marginLeft="6dp">


    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="6dp"
        android:gravity="center"
        android:layout_gravity="center"
```

```
            android:text="FRI"
            android:id="@+id/friday"
            android:textSize="30sp"
            android:textStyle="bold"
            android:clickable="true"
            android:background="?attr/selectableItemBackgroundBorderless"/>
        </android.support.v7.widget.CardView>

        <android.support.v7.widget.CardView
            android:layout_width="150dp"
            android:layout_height="match_parent"
            android:layout_gravity="center"
            app:cardCornerRadius="5dp"
            app:cardElevation="8dp"
            android:layout_marginRight="6dp"
            android:layout_marginLeft="6dp">


            <TextView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_margin="6dp"
                android:gravity="center"
                android:layout_gravity="center"
                android:text="SAT"
                android:id="@+id/saturday"
                android:textSize="30sp"
                android:textStyle="bold"
                android:clickable="true"
                android:background="?attr/selectableItemBackgroundBorderless"/>

        </android.support.v7.widget.CardView>


    </LinearLayout>

</HorizontalScrollView>

<View
    android:layout_width="match_parent"
```

```
        android:layout_height="1dp"
        android:background="@drawable/fadedline2"/>
    <ListView
       android:layout_width="match_parent"
       android:layout_height="0dp"
       android:layout_weight="1"
       android:id="@+id/scheduleList">

    </ListView>

       <Button
          android:layout_width="match_parent"
          android:layout_height="50dp"
          android:layout_margin="6dp"
          android:text="Add another period"
          android:textColor="@color/colorPrimaryDark"
          android:id="@+id/addPeriod"/>

    </LinearLayout>
```